

Robocup Federation
Robocup 2021
JuniorSoccer Simulation
Team name: Radiant

AmirAli Heidarzadeh¹, Soheil AmirAhmadi¹, Mahan Satari², Ramtin Barzkar²

¹Allame Helli 2 High school, Tehran, Iran

f_daemi@yahoo.com

Abstract

With the introduction of RoboCup simulator in the Junior Football league, we started working on this simulator. After getting acquainted with the software environment and programming, we implemented various algorithms and strategies. Using the data received from the simulator and using the coordinates and robots, we were able to implement various algorithms so that one of the robots as a goalkeeper and other robots play the role of defender and attacker. After the team approved the code sent by the team, we worked more seriously. We followed and we were able to get better conditions to participate in RoboCup 2021.

Keywords: software, programming, robot, simulator, algorithm

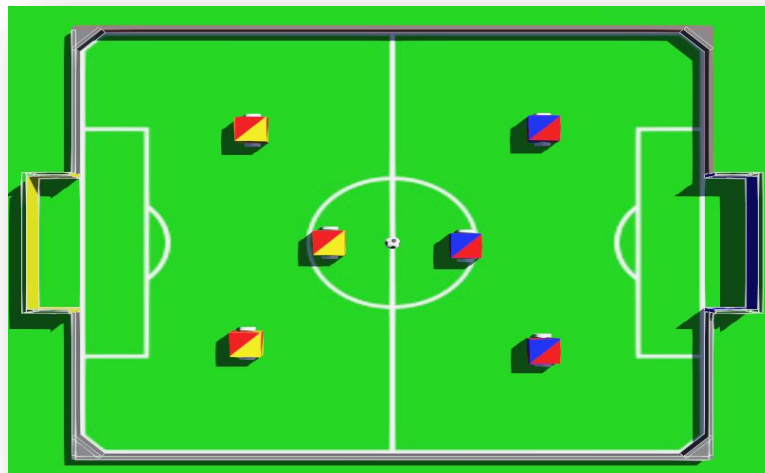
1- AmirAli Heidarzadeh
2- Soheil AmirAhmadi
3- Mahan Satari
4- Ramtin Barzkar

1 Introduction

Robotic is one of the most popular subjects that has attracted many students to itself that has many leagues and one of them is soccer sim league. Team Radiant from Allame Helli 2 High school in Tehran, Iran, we worked on real light weight soccer robots for a year and followed the hardware design and mechanics of soccer robots as well as robot programming professionally. With the introduction of RoboCup simulator, we decided to try this simulator. After installing the webot software and running the junior the junior soccer sim program on it, we became interested in this simulator and decided to participate in RoboCup 2021, so we sent the initial code to confirm our team in the Asian region, and after Confirmation our team is currently preparing for the World RoboCup. We are programming, making TDP, ... by dividing tasks among team members. figure_1

Table_1: strategy, programming and software

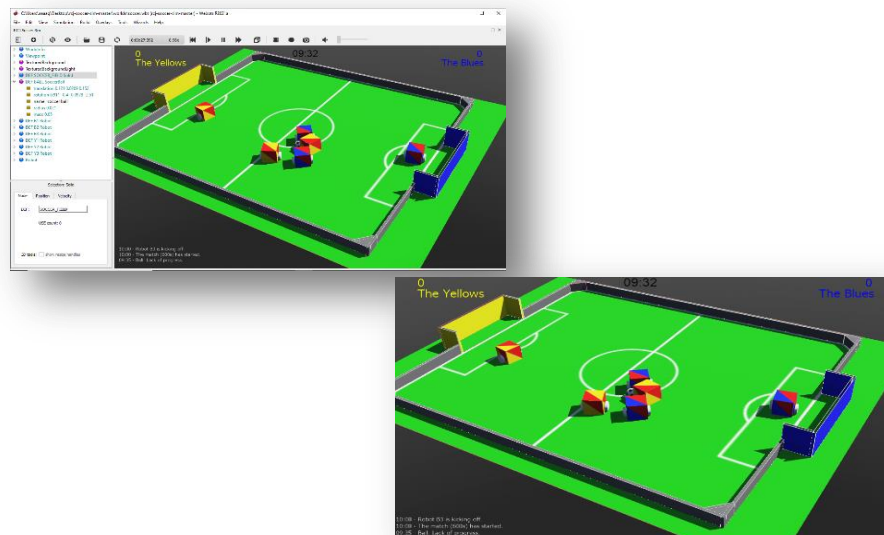
Strategy	Programming	Software
Ideas (making robots play better)	Detecting ball	Webot
Goalkeeper robot	Following the ball	Visual Studio Code (vs code)
Forward robot	Scoring	Python
Defender robot		



Figure_1: junior soccer simulator with webot

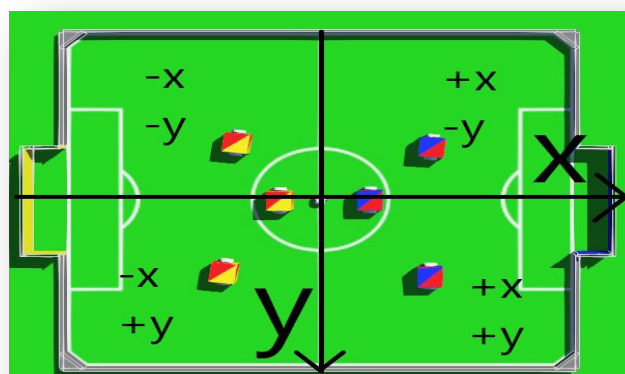
2 Software

Webot is a software which all real robots can be simulated and simulator of that robot can be designed. The junior football robot simulator is also designed with webot software and there are 3 robots for playing football and each team must have these 3 robots. Write the game program webot software allows us to make changes in programming on robots and evaluate the performance of our robot in the game with the opposing team. The programming language of robots is Python, and using this. The language of different algorithms can be implemented on robots. figure_2



Figure_2: webot software

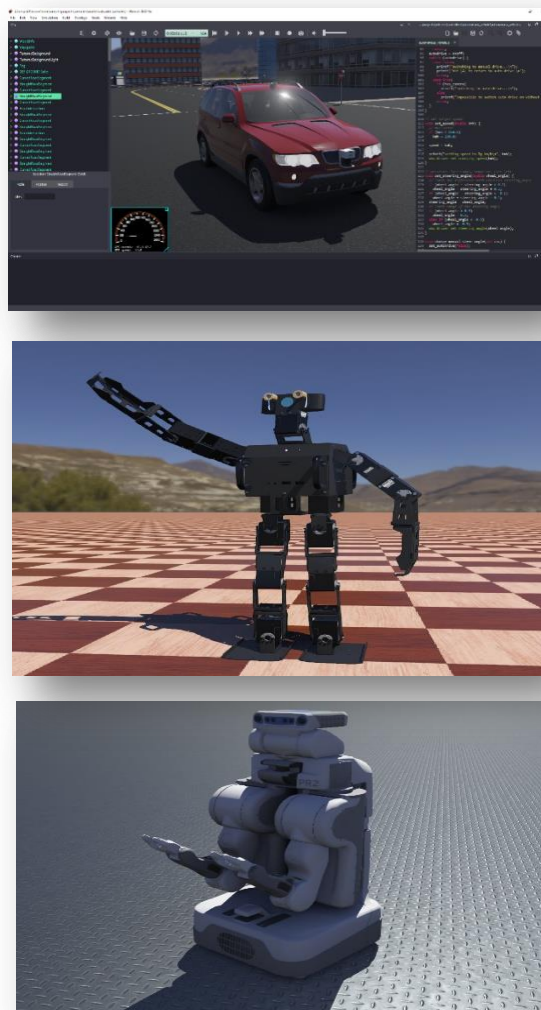
Webot software provides us with robot and ball data using Python commands and defining different functions. These data show the position of the robots and ball on the ground relative to the center of the ground. They are important to chase the ball. Figure_3



Figure_3: python commands

2.1 Webot

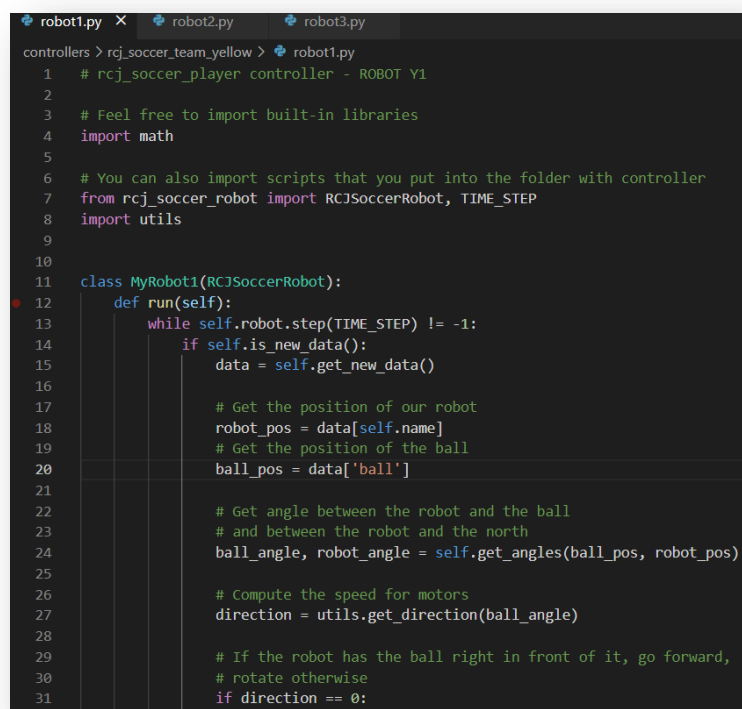
Webots is a free and open-source 3D robot simulator used in industry, education and research. Webots includes a large collection of freely modifiable models of robots, sensors, actuators and objects. In addition, it is also possible to build new models from scratch or import them from 3D CAD software. When designing a robot model, the user specifies both the graphical and the physical properties of the object. The physical properties include the mass, friction factor, as well as the spring and damping constants. Simple fluid dynamics is present in the software. Webots includes a set of sensors and actuators frequently used in robotic experiments, e.g., lidars, radars, proximity sensors, GPS, accelerometers, cameras, emitters, and receivers, servo motors (rotational & linear), position and force sensors, LEDs, grippers, gyros, compass, IMU, etc. The robot controller programs can be written outside of webots in C, C++, Python, ROS, Java using a simple APL. figure_4



Figure_4: 3D robot simulator with webot

2.2 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy; a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source; the fast edit-test-debug cycle makes this simple approach very effective. figure_5

A screenshot of a code editor with a dark background and light-colored text. The editor has three tabs at the top: 'robot1.py', 'robot2.py', and 'robot3.py'. The 'robot1.py' tab is active. The code is written in Python and is a soccer robot controller. It includes comments in green, single-line imports, and a class definition 'MyRobot1' that inherits from 'RCJSoccerRobot'. The 'run' method of the class contains a while loop that checks for new data, gets the robot and ball positions, calculates angles, and determines the direction for the robot to move. The code is line-numbered from 1 to 31.

```
robot1.py X robot2.py robot3.py
controllers > rcj_soccer_team_yellow > robot1.py
1  # rcj_soccer_player controller - ROBOT Y1
2
3  # Feel free to import built-in libraries
4  import math
5
6  # You can also import scripts that you put into the folder with controller
7  from rcj_soccer_robot import RCJSoccerRobot, TIME_STEP
8  import utils
9
10
11 class MyRobot1(RCJSoccerRobot):
12     def run(self):
13         while self.robot.step(TIME_STEP) != -1:
14             if self.is_new_data():
15                 data = self.get_new_data()
16
17                 # Get the position of our robot
18                 robot_pos = data[self.name]
19                 # Get the position of the ball
20                 ball_pos = data['ball']
21
22                 # Get angle between the robot and the ball
23                 # and between the robot and the north
24                 ball_angle, robot_angle = self.get_angles(ball_pos, robot_pos)
25
26                 # Compute the speed for motors
27                 direction = utils.get_direction(ball_angle)
28
29                 # If the robot has the ball right in front of it, go forward,
30                 # rotate otherwise
31                 if direction == 0:
```

Figure_5: Python is an interpreted

3 Algorithm

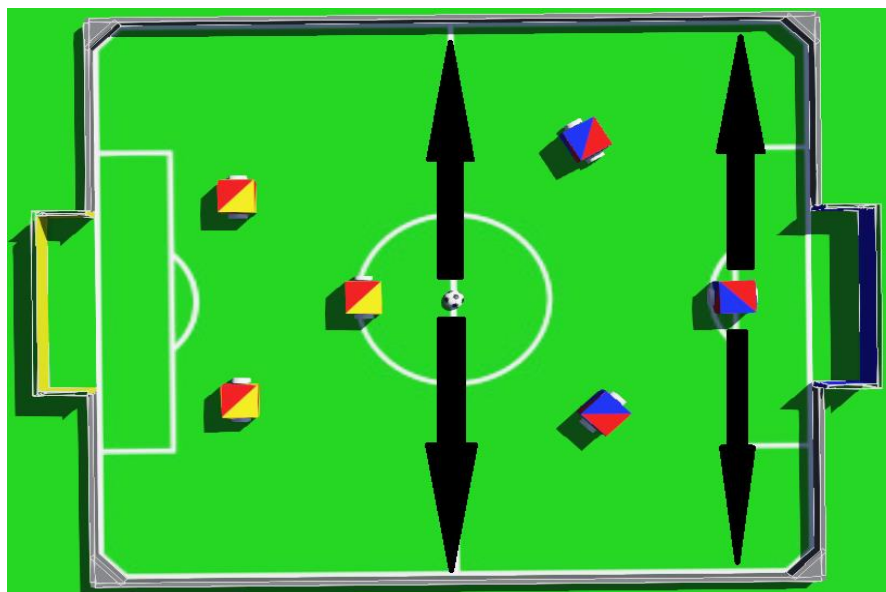
Finding the ball is the first step in the soccer robot. This is done by a variable called direction and is defined and set in the common utils.py program. So we decided that our robots recognize four directions so that we can move faster and more carefully. When competition starts the robot that selected to goalkeeper goes to our goal and start moving on the y-axis toward moving the ball. The attacker robot start chasing the ball when it enters the opponent's half of the field and tries to enter the ball into the opponent's goal and score goals. The shooter robot has the task of helping to attack and defense, and when the ball is fixed in a center of the field. figure_6



Figure_6: Finding the ball is the first step in the soccer robot

3.1 Game strategy

After launching webot software and programmer in Python environment, we implemented many ideas to make robots play better and came to the conclusion that the goalkeeper has an important role in the game. To avoid scoring goals, we must have a goalkeeper. So we place one of the robots (usually a robot farther away from you) in the goal area to prevent the opponent from scoring. An attacking robot can advance the game in our favor, so one of the robots plays the role of the attacker robot and the ligament. The third robot has a defensive role and is always in the center of the field to be ready to attack the opponent's goal while defending his own goal. figure_7

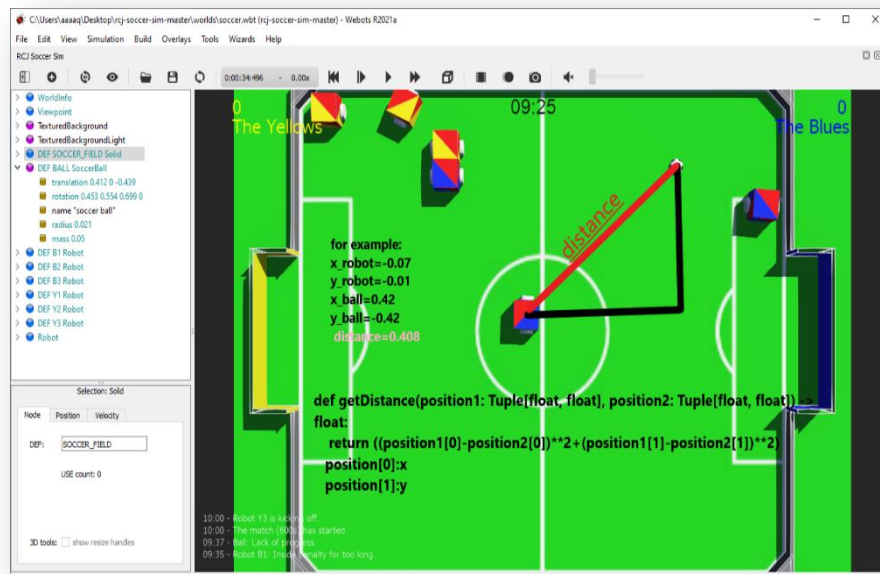


Figure_7: Finding the ball is the first step in the soccer robot

3.2 Ball distance

The distance of each robot from the ball is very important because we need this distance to write game strategies and algorithms.

To calculate the distance of the ball from the robots, we use a mathematical equation such as Pythagoras according to the position of the robot and the ball on the axis **x** and **y**. figure_8



Figure_8: distance of each robot from the ball

4 References:

- <https://drive.google.com/file/d/1WdsB8ayPtIh3qTKui0jDZdmWI-ShMa8/view>
- <https://github.com/RoboCupJuniorTC/rcj-soccer-im/archive/refs/heads/master.zip>
- <https://www.python.org/downloads>
- <https://github.com/>
- <https://www.avrfreaks.net>
- <http://www.hpinfotech.ro>