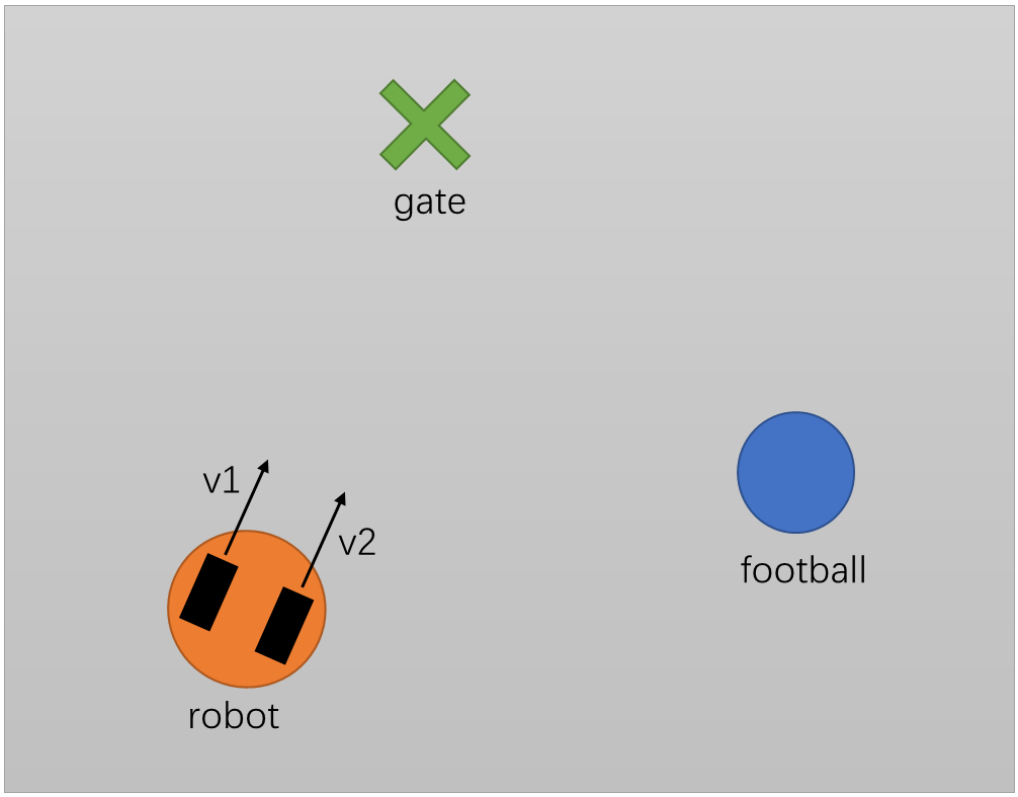
scenario:

A robot is playing with football and it wants the football to enter into the gate. There are two ways for it to complete the goal- shooting or driblbling. Undoutedly, it is most time-efficient for the robot to shoot, but there is some sliding when robot tries to knick the football. In this way, the football will not move as expected. So what is the robot's strategy now?




It is a world based on velocity instead of acceleration. The action of robot is the velocity of two wheels, which will give the velocity and orientation of the robot. When hitting the football, the new velocity of the robot and the ball will be calculated by law of conservation of momentum(1) and law of conservation of mechanical energy(2).

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2' \tag{1}$$

$$\frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_1 v_1'^2 + \frac{1}{2} m_2 v_2'^2 \tag{2}$$

The action_space and observation_space of the environments.

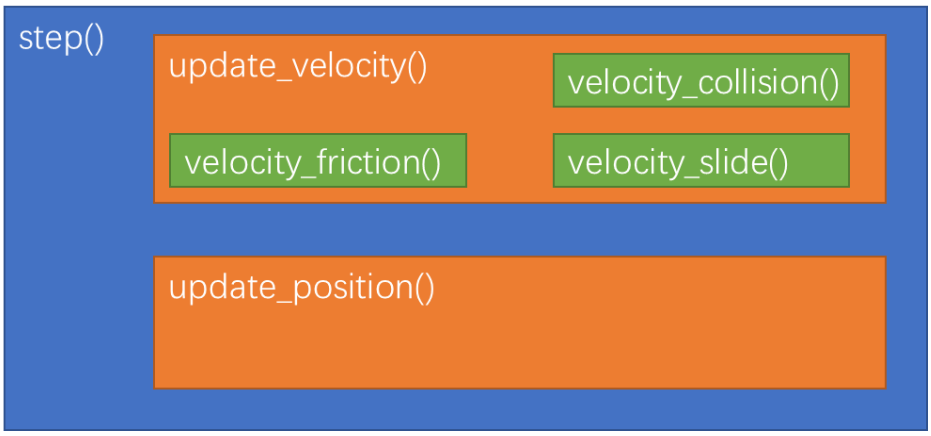
```
Oberservation:
        robot position, football position, robot speed, football speed, robot orientation,
gate position
Action:
        robot speed
```

Some parameters in this environment.

```
MAP_WIDTH    = 1000          # the width of the map
```

```
MAP_LENGTH  = 1000          # the length of the map
ROB_SIZE    = 5             # the radius of robot (supposing circle)
BALL_SIZE   = 2             # the radius of football (supposing circle)
ROB_POS     = [10,10]       # the initial location of robot (valid if random location is
closed)
BALL_POS    = [500,500]     # the initial location of football (valid if random location is
closed)
GATE_POS    = [700,700]     # the initial location of gate (valid if random location is
closed), this can be n-D array for n gates
GATE_NUM    = 1             # the number of gates
                            # range[1,inf] 1-only one gates
CE_FRI      = 0             # coefficient of friction
                            # range[0,1]. 0-no friction
CE_CRH_WALL = 1             # when hitting the wall, the coefficient of maintaining energy
                            # range[0,1] 1-fully elastic collision
MASS_RAT    = 1             # the ratio of robot to football
                            # range(0,inf] inf-robot is far heavier than football
SLD_THD     = 0             # the random sliding's threshold when robot knicking the ball
                            # range[0,inf] 0-there is no random sliding on robot
```

The whole core function is the step(). And its process is shown below.




update_velocity() is to get the new velocity of the robot and the football. Due to several factors like collision, friction and sliding, the velocity calculation needs to consider all these things.

update_position() is to get the new position of the robot and the football. The position is the integration of the velocity based on the timespan we set.

In order to test the environments more directly and to have a better presentation both, it is meaningful to build a UI.

We play to use pyglet library to help us to do it.