```python
#!/usr/bin/python3
#Fuzzy
import os, sys
import random

from pysimbotlib.Window import PySimbotApp
from pysimbotlib.Robot import Robot
from kivy.core.window import Window
from kivy.logger import Logger

# Number of robot that will be run
ROBOT_NUM = 1

# Delay between update (default: 1/60 (or 60 frame per sec))
TIME_INTERVAL = 1/60 #10frame per second

# Max tick
MAX_TICK = 3000

# START POINT
START_POINT = (20, 560)

# Map file
MAP_FILE = 'maps/default_map.kv'

class FuzzyRobot(Robot):

    def __init__(self):
        super(FuzzyRobot, self).__init__()
        self.pos = START_POINT

    def update(self):
        ''' Update method which will be called each frame
        '''
        self.ir_values = self.distance()
        self.target = self.smell()

        # initial list of rules
        rules = list()
        turns = list()
        moves = list()

        # rule 0
        rules.append(self.front_far() * self.left_far() * self.right_far())
        turns.append(0)
```

```
        moves.append(3)

        # rule 1
        rules.append(self.right_near())
        turns.append(-20)
        moves.append(2)

        # rule 2
        rules.append(self.left_near())
        turns.append(30)
        moves.append(2)

        # rule 3
        rules.append(self.right_mid_near())
        turns.append(-80)
        moves.append(0)

        # rule 4
        rules.append(self.left_mid_near())
        turns.append(50)
        moves.append(0)

        # rule 5
        rules.append(self.smell_left() )
        turns.append(-10)
        moves.append(3)

        # rule 6
        rules.append(self.smell_right() )
        turns.append(10)
        moves.append(3)

        # rule 7
        rules.append(self.front_near() )
        turns.append(20)
        moves.append(-1)

        # rule 8
        rules.append(self.left_near()*self.left_mid_far())
        turns.append(0)
        moves.append(3)

        # rule 9
        rules.append(self.right_near()*self.right_mid_far())
        turns.append(0)
```

```
        moves.append(3)

        ans_turn = 0.0
        ans_move = 0.0
        for r, t, m in zip(rules, turns, moves):
            ans_turn += t * r
            ans_move += m * r

        print('move', ans_move)
        print('turn', ans_turn)
        self.turn(ans_turn)
        self.move(ans_move)

    def front_far(self):
        irfront = self.ir_values[0]
        if irfront <= 10:
            return 0.0
        elif irfront >= 40:
            return 1.0
        else:
            return (irfront-10.0) / 30.0

    def front_near(self):
        return 1 - self.front_far()

    def left_far(self):
        irleft = self.ir_values[6]
        if irleft <= 10:
            return 0.0
        elif irleft >= 30:
            return 1.0
        else:
            return (irleft-10.0) / 20.0

    def left_mid_far(self):
        irleft = self.ir_values[7]
        if irleft <= 10:
            return 0.0
        elif irleft >= 30:
            return 1.0
        else:
            return (irleft-10.0) / 20.0

    def left_mid_near(self):
        return 1-self.left_mid_far()
```

```
    def left_near(self):
        return 1 - self.left_far()

    def right_far(self):
        irright = self.ir_values[2]
        if irright <= 10:
            return 0.0
        elif irright >= 30:
            return 1.0
        else:
            return (irright-10.0) / 20.0

    def right_mid_far(self):
        irright = self.ir_values[1]
        if irright <= 10:
            return 0.0
        elif irright >= 30:
            return 1.0
        else:
            return (irright-10.0) / 20.0

    def right_mid_near(self):
        return 1- self.right_mid_far()

    def right_near(self):
        return 1 - self.right_far()

    def smell_right(self):
        target = self.smell()
        if target >= 90:
            return 1.0
        elif target <= 0:
            return 0.0
        else:
            return target / 90.0

    def smell_center(self):
        target = abs(self.smell())
        if target >= 45:
            return 1.0
        elif target <= 0:
            return 0.0
        else:
            return target / 45.0
```

```
    def smell_left(self):
        target = self.smell()
        if target <= -90:
            return 1.0
        elif target >= 0:
            return 0.0
        else:
            return -target / 90.0

if __name__ == '__main__':
    app = PySimbotApp(FuzzyRobot, ROBOT_NUM, mapPath=MAP_FILE,
interval=TIME_INTERVAL, maxtick=MAX_TICK)
    app.run()
```