Instituto Tecnológico y de Estudios Superiores de Monterrey

Modelación de sistemas multiagentes con gráficas computacionales

Gpo. 102

Evidencia #3

Avance 3

Nombre de los profesores:

Raul V. Ramirez Velarde

Edgar Convantes Osuna

Integrantes:

Luis Alberto Portilla López | A00829935

Maruca Cantú Valdés | A00834245

Francisco Javier Lugo Gutierréz | A01571142

Bryan Alejandro Cortés Guzmán | A01284228

Maruca Cantu Valdes | A00834245

06 de septiembre de 2023 Monterrey, N.L.

```
%pip install mesa
from mesa import Agent,Model
from mesa.time import RandomActivation
from mesa.space import MultiGrid
from mesa.datacollection import DataCollector
import random
class Coche(Agent):
    def __init__(unique_id,model):
        super().__init__(unique_id,model)
        self.velocidad = 0

    def get_semaforo():
        cell = self.model.grid.get_cell_list_contents([self.pos])
        if cell:
            neighbors = self.model.grid.get_neighbors(self.pos,
moore=False)
            for neighbor in neighbors:
                if isinstance(neighbor, Semaforo):
                    return neighbor.status

    def get_neighbors():
        neighbors = self.model.grid.get_neighbors(self.pos,
moore=False)
        cars = []
        for neighbor in neighbors:
            if isinstance(neighbor, Coche):
                cars.append(neighbor)
        return cars

    def step(self):
        semaforo_status = get_semaforo()

        if semaforo_status == 'Rojo':
            self.velocidad = 0
        elif semaforo_status == 'Verde':
            self.velocidad = 5
        elif semaforo_status == 'Amarillo':
            self.velocidad = 2.5

        neighbors = get_neighbors()
        if neighbors:
            self.velocidad = min(self.velocidad,
neighbors[0].velocidad)

        self.model.grid.move_agent(self, (self.pos[0], self.pos[1] +
self.velocidad))
class Semaforo(Agent):
    def __init__(self,unique_id,model):
        super().__init__(unique_id,model)
        self.status = "Verde"
        self.time = 0
        self.carros_en_frente = 0
        self.duracion_verde = 5
        self.duracion_amarillo = 6
        self.duracion_rojo = 11
        self.has_btn = False
        self.btn_pressed = False
    def get_cars():
```

```
            cell = self.model.grid.get_cell_list_contents([self.pos])
            if cell:
                cell = cell[0]
                neighbors = self.model.grid.get_neighbors(self.pos,
moore=False)
                for neighbor in neighbors:
                    if isinstance(neighbor, Coche):
                        self.carros_en_frente += 1

    def get_other_cars():
        neighbors = self.model.grid.get_neighbors(self.pos,
moore=False)
        cars = []
        for neighbor in neighbors:
            if isinstance(neighbor, Semaforo):
                cars.append(neighbor.carros_en_frente)

        return cars

    def get_status():
        neighbors = self.model.grid.get_neighbors(self.pos,
moore=False)
        estados = []
        for neighbor in neighbors:
            if isinstance(neighbor, Semaforo):
                estados.append(neighbor.status)
        return estados

    def step(self):
        self.time += 1
        self.get_cars()
        carros = get_other_cars();
        for amnt in carros:
            if self.carros_en_frente > amnt
                self.time = 0
        if self.time == self.duracion_verde:
            self.status = "Amarillo"
        elif self.time == self.duracion_verde + self.duracion_amarillo:
            self.status = "Rojo"
        elif self.time == self.duracion_verde + self.duracion_amarillo
+ self.duracion_rojo:
            self.status = "Verde"
            self.time = 0
        if self.has_btn:
            if self.btn_pressed:
                self.status = "Verde"
                self.time = 0
                self.btn_pressed = False
        self.model.grid.move_agent(self, (self.pos[0], self.pos[1] +
self.carros_en_frente))
        self.model.grid.move_agent(self, (self.pos[0], self.pos[1] +
1))
        self.get_other_cars()
        self.get_status()
class Peaton(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.caminando = True
```

```
        self.velocidad = 1

    def get_semaforo_status(self):
        cell = self.model.grid.get_cell_list_contents([self.pos])
        if cell:
            neighbors = self.model.grid.get_neighbors(self.pos,
moore=False)
            for neighbor in neighbors:
                if isinstance(neighbor, Semaforo):
                    semaforo_status = neighbor.get_status()
                    if semaforo_status == 'Verde':
                        return "No pasa"
                    elif semaforo_status == 'Amarillo':
                        return "No pasa"
                    elif semaforo_status == 'Rojo':
                        return "Pasa"
        return "Pasa"  # En caso de no encontrar semáforo, se considera
que puede pasar

    def semaforo_tiene_boton(self):
        cell = self.model.grid.get_cell_list_contents([self.pos])
        if cell:
            neighbors = self.model.grid.get_neighbors(self.pos,
moore=False)
            for neighbor in neighbors:
                if isinstance(neighbor, Semaforo):
                    if neighbor.has_btn:
                        return True
        return False

    def presionar_boton(self):
        cell = self.model.grid.get_cell_list_contents([self.pos])
        if cell:
            neighbors = self.model.grid.get_neighbors(self.pos,
moore=False)
            for neighbor in neighbors:
                if isinstance(neighbor, Semaforo) and neighbor.has_btn:
                    neighbor.presionado = True

    def step(self):
        semaforo_status = self.get_semaforo_status()
        if semaforo_status == "Pasa":
            self.caminando = True
        else:
            self.caminando = False

        if self.semaforo_tiene_boton():
            self.presionar_boton()
class Sensor(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.presionado = False
```