Room Cleaner

> Fernando López Gómez | A01639715

In [ ]:
```
import agentpy as ap
from random import choice, randint

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import IPython
```

About the Room cleaner

This model simulates a dirty room and a robot cleaning system. The robots strart at the top left tile and start searching for dirty tiles and cleaning them until the room is clean or the number of iterations is exceeded. The robots can move one tile at the time to any of their neighbor tiles.

The model assumes there are no obstacles in the room that could affect robot movement

> You cand find the library documentation here:
> https://agentpy.readthedocs.io/en/latest/overview.html

Model Definition

In [ ]:
```
MOVES = [(0,1),(1,0),(0,-1),(-1,0), (-1,1),(1,-1),(1,1),(-1,-1)]
class roomModel(ap.Model):

    def setup(self):
        # Create agents (dirtyTiles)
        n_dirtyTiles = int(self.p['Dirt density'] * (self.p.sizeX * self.p.sizeY))
        dirtyTiles = self.dirtyTiles = ap.AgentList(self, n_dirtyTiles)

        # Create agents (robots)
        robots = self.robots = ap.AgentList(self, self.p['numRobots'])
        robotList = [(1,1)] *self.p['numRobots'] #List where every tuple is the positi

        # Create grid (room)
        self.room = ap.Grid(self, [self.p.sizeX, self.p.sizeY], track_empty=True)
        self.room.add_agents(dirtyTiles, random=True, empty=True)
        self.room.add_agents(robots, robotList, empty=True)

        # Initiate a dynamic variable for all dirtyTiles
        self.dirtyTiles.isClean = False

        #Identify agents
        self.dirtyTiles.isTile = True
        self.robots.isTile = False


    def step(self):
```

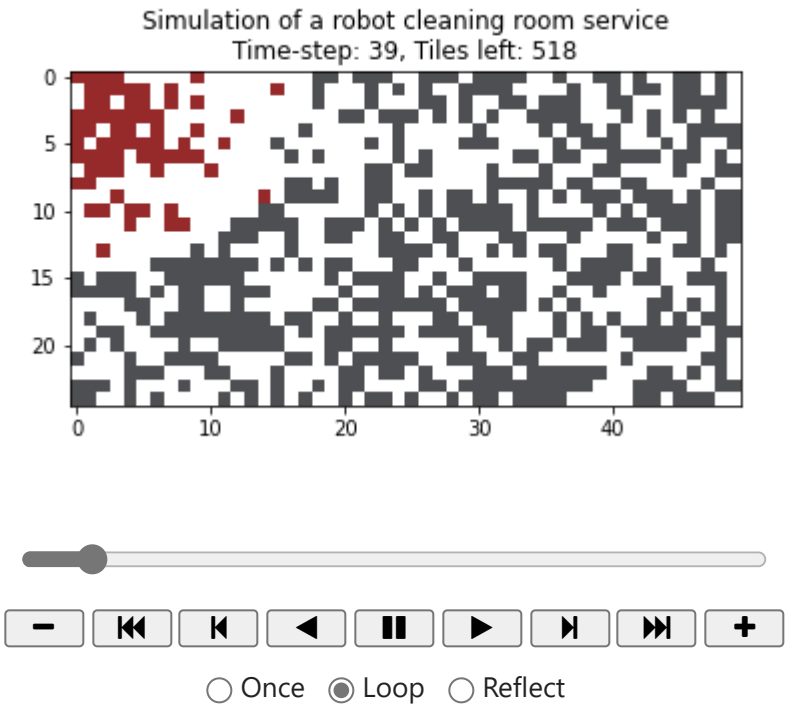
```
        # Select cleaning robots
        robots = self.robots

        # Spread fire
        for robot in robots:
            for neighbor in self.room.neighbors(robot):
                if neighbor.isTile and not neighbor.isClean:
                    neighbor.isClean = True # Robot starts cleaning
                    neighbor.isTile = None #Change color
                    break
            else:
                #for-else: If you dont hit any break statement, excecute else block
                # so if we dont find any dirtyTile near us, we move to another tile

                self.room.move_by(robot, choice(MOVES)) #(randint(-1,1),randint(-1,1)



        # Stop simulation if no dust is left
        if len(self.dirtyTiles.select(self.dirtyTiles.isClean == False)) == 0:
            self.stop()

    def end(self):

        # Document a measure at the end of the simulation
        cleaned_tiles = len(self.dirtyTiles.select(self.dirtyTiles.isClean == True))
        self.report('Percentage of cleaned tiles',
                    cleaned_tiles / len(self.dirtyTiles))
```

Single Run Animation

In [ ]:

```
# Define parameters

parameters = {
    'Dirt density': 0.5, #Percentage of the room covered in dirt
    'sizeX': 25,
    'sizeY': 50,
    'numRobots': 200,
    'steps': 500
}


# Create single-run animation with custom colors

def animation_plot(model, ax):
    attr_grid = model.room.attr_grid('isTile')
    color_dict = {True:'#4e4f52', False:'#962a2a', None:'#ffffff'}
    ap.gridplot(attr_grid, ax=ax, color_dict=color_dict, convert=True)
    ax.set_title(f"Simulation of a robot cleaning room service\n"
                 f"Time-step: {model.t}, Tiles left: "
                 f"{len(model.dirtyTiles.select(model.dirtyTiles.isClean == False))}")

fig, ax = plt.subplots()
model = roomModel(parameters)
animation = ap.animate(model, fig, ax, animation_plot)
IPython.display.HTML(animation.to_jshtml(fps=15))
```

Out[ ]:



Observaciones

El tiempo de ejecución del programa depende de los parámetros que se le asignen al programa:

1. *Número de agentes robots:* El número de agentes influye en la cantidad de celdas que limpian debido a que que pueden ejecutar la acción de moverse un mayor número de veces seguidas y poder desplazarse por el mapa de manera más rápida, pudiendo así alcanzar más celdas sucias. Este parámetro está representado con la key word `numRobots`.
2. *Tamaño del cuarto:* El tamaño del cuarto que se quiere limpiar influye en el tiempo de ejecución debido a que muchos robots en un cuarto pequeño pueden terminar la ejecución del programa antes del número máximo de ejecuciones. Este parámetro está representado por las key words `sizeX` y `sizeY`.

3. *Densidad de suciedad:* Entre más limpio esté el cuarto, más rápido terminarán los robots de limpiar el cuarto; esto puede tener diferentes casos, siendo el mejor caso el hecho de que toda la basura se concentre cerca del área en la que los robots aparecem, y el peor caso cuando haya muy poca basura lejos del área de aparición de robots, caso en el cual pudieran no terminar de limpiar el cuarto. Este parámetro está representado por la key word `Dirt density`.

Observations

The execution time of the program depends on the parameters that are assigned to the program:

1. *Number of agent robots:* The number of agents influences the number of cells they clean because they can execute the action of moving a greater number of times in a row and be able to move around the map more quickly, thus being able to reach more dirty cells. This parameter is represented by the key word `numRobots`.
2. *Size of the room:* The size of the room to be cleaned influences the execution time because many robots in a small room can finish the execution of the program before the maximum number of executions. This parameter is represented by the key words `sizeX` and `sizeY`.

3. *Dirt Density:* The cleaner the room, the faster the robots will finish cleaning the room; this can have different cases, the best case being that all trash is concentrated near the area where the bots spawn, and the worst case being that there is very little trash far from the bot spawn area, in which case they might not finish cleaning the room. This parameter is represented by the key word `Dirt density`.