Segunda extensión: Agregando viento

Modelación de sistemas multiagentes con gráficas computacionales

Omar Jiménez Armendáriz	A01732097
Francisco Rocha Juárez	A01730560
Alejandro Alfonso Ubeto Yañez	A01734977

2.1.- Versión de la extensión

Haz click **aquí** para acceder a nuestro repositorio de BitBucket.

Id de commit: **3187ba2**

2.2.- Documentación de cambios

Los cambios realizados durante esta extensión son los siguientes:

- Implementación de parámetros *SoutWindSeed* y *WestWindSpeed* al constructor de la clase *Tree*.
- Declaración de condiciones de propagación:
    - *neighbor_fine*: el árbol vecino no se está quemando ni se ha quemado.
    - *spread_rand* = un número aleatorio entre 0 y 100.
    - *wind_condition* = la velocidad del viento favorece la propagación del fuego en dirección al árbol vecino (se inicializa como falso)
- Se compara el la velocidad del viento junto con las posiciones del árbol actual y el árbol vecino para volver a *wind_condition* verdadera y aumentar o disminuir la variable spread rand.
- Si se cumplen las condiciones de propagación y *spread_rand* es menor a *probability_of_spread*, el árbol vecino se enciende en llamas.
- Implementación de parámetros *SoutWindSeed* y *WestWindSpeed* inicializados con valor 0 en el constructor de la clase *Forest*.
- Cambio del color de los árboles a azul pues nuestro uno de los integrantes del equipo no es capaz de distinguir entre el rojo y verde.
- Implementación de input de tipo Slider para controlar las variables de *SoutWindSeed* y *WestWindSpeed* del modelo.

2.3.- Versión actual del código

```
from mesa import Agent, Model
from mesa.space import Grid
from mesa.time import RandomActivation

from mesa.visualization.modules import CanvasGrid
from mesa.visualization.ModularVisualization import ModularServer

from mesa.visualization.UserParam import UserSettableParameter

from mesa.datacollection import DataCollector
from mesa.visualization.modules import ChartModule

# Clase Agente: Arbol
class Tree(Agent):
    FINE = 0
    BURNING = 1
    BURNED_OUT = 2
    def __init__(self, model: Model, probability_of_spread, south_wind_speed,
west_wind_speed):
        super().__init__(model.next_id(), model)
        self.condition = self.FINE
        self.probability_of_spread = probability_of_spread
        self.south_wind_speed = south_wind_speed
        self.west_wind_speed = west_wind_speed

    def step(self):
        if self.condition == self.BURNING:
            neighbors = self.model.grid.neighbor_iter(self.pos, moore=False)
            for neighbor in neighbors:

                # Condiciones de propagación
                neighbor_fine = neighbor.condition == self.FINE
                spread_rand = self.random.random() * 100
                wind_condition = False

                # Velocidad del viento Sur
                if self.south_wind_speed > 0 and self.pos[1] > neighbor.pos[1]: # Sur
                    wind_condition = True
                    spread_rand -= self.south_wind_speed
                elif self.south_wind_speed < 0 and self.pos[1] < neighbor.pos[1]:   # Norte
                    wind_condition = True
                    spread_rand += self.south_wind_speed

                # Velocidad del viento Oeste
                if self.west_wind_speed > 0 and self.pos[0] > neighbor.pos[0]:  # Oeste
                    wind_condition = True
                    spread_rand -= self.west_wind_speed
                elif self.west_wind_speed < 0 and self.pos[0] < neighbor.pos[0]:    # Este
                    wind_condition = True
                    spread_rand += self.west_wind_speed

                # Cambio de estado
```

```
            if neighbor_fine and spread_rand < self.probability_of_spread and
wind_condition:
                neighbor.condition = self.BURNING
        self.condition = self.BURNED_OUT

# Clase Modelo: Forest
class Forest(Model):
    def __init__(self, height=50, width=50, density=0.90, probability_of_spread=50,
south_wind_speed=0, west_wind_speed=0):
        super().__init__()
        self.schedule = RandomActivation(self)
        self.grid = Grid(height, width, torus=False)
        for _,x,y in self.grid.coord_iter():
            if self.random.random() < density:
                tree = Tree(self, probability_of_spread, south_wind_speed, west_wind_speed)
                if x == 0:
                    tree.condition = Tree.BURNING
                self.grid.place_agent(tree, (x,y))
                self.schedule.add(tree)

        # Recolector de información: procentaje de arboles quemados
        self.datacollector = DataCollector({"Percent burned": lambda m: self.count_type(m,
Tree.BURNED_OUT) / len(self.schedule.agents)})

    # Método estático para referenciar al modelo Forest dentro de la función lambda
    @staticmethod
    def count_type(model, condition):
        count = 0
        for tree in model.schedule.agents:
            if tree.condition == condition:
                count += 1
        return count

    def step(self):
        self.schedule.step()
        self.datacollector.collect(self)
        if self.count_type(self, Tree.BURNING) == 0:
            self.running = False

# Coloreado de agentes
def agent_portrayal(agent):
    if agent.condition == Tree.FINE:
        portrayal = {"Shape": "circle", "Filled": "true", "Color": "Blue", "r": 0.75, "Layer": 0}
    elif agent.condition == Tree.BURNING:
        portrayal = {"Shape": "circle", "Filled": "true", "Color": "Red", "r": 0.75, "Layer": 0}
    elif agent.condition == Tree.BURNED_OUT:
        portrayal = {"Shape": "circle", "Filled": "true", "Color": "Black", "r": 0.75, "Layer": 0}
    else:
        portrayal = {}

    return portrayal

grid = CanvasGrid(agent_portrayal, 50, 50, 450, 450)
```

```
# Creación de tabla que grafica datacollector
chart = ChartModule([{"Label": "Percent burned", "Color": "Black"}],
data_collector_name='datacollector')

# Implementación de Slider para manipular densidad de bosque y la probabilidad de
incendio
server = ModularServer(Forest,[grid, chart],"Forest",
                {"density": UserSettableParameter(
                    "slider", "Tree density", 0.45, 0.01, 1.0, 0.01),
                "probability_of_spread": UserSettableParameter(
                    "slider", "Spread probability", 50, 0, 100, 1),
                "south_wind_speed": UserSettableParameter(
                    "slider", "South wind speed", 0, -25, 25, 1),
                "west_wind_speed": UserSettableParameter(
                    "slider", "West wind speed", 0, -25, 25, 1),
                "width":50,
                "height":50
                })

server.port = 8522 # The default
server.launch()
```

2.4.- Cambios observados

Antes de dar detalles sobre los resultados de la implementación, es muy importante resaltar lo importante que es para este equipo el hacer este proyecto accesible para todas las personas, es por esto que se decidió cambiar el color de los árboles a azul para que las personas daltónicas sean capaces de distinguir las celdas en llamas de las celdas en perfecto estado. Otro detalle visual es la implementación de dos sliders a la interfaz los cuales permitirán controlar los valores de las dos nuevas variables.

A continuación se muestra un gráfico que ejemplifica el nuevo comportamiento del modelo mediante ejecuciones del mismo utilizando diferentes valores para las nuevas variables de *SoutWindSeed* y *WestWindSpeed* del modelo. Cabe añadir que ambas simulaciones cuentan con valor alto para la variable que controla la densidad del bosque, esto es para que se note mejor el efecto de la velocidad del viento sobre los agentes.

El primer gráfico cuenta con vientos en dirección al sur y al este en su máximo nivel, esto hace que el fuego tienda a esparcirse de arriba hacia abajo en diagonal hacia la esquina inferior derecha del modelo pues el incendio está siendo empujado por la fuerza del viento desde el norte hacia el sur y del oeste hacia el este.

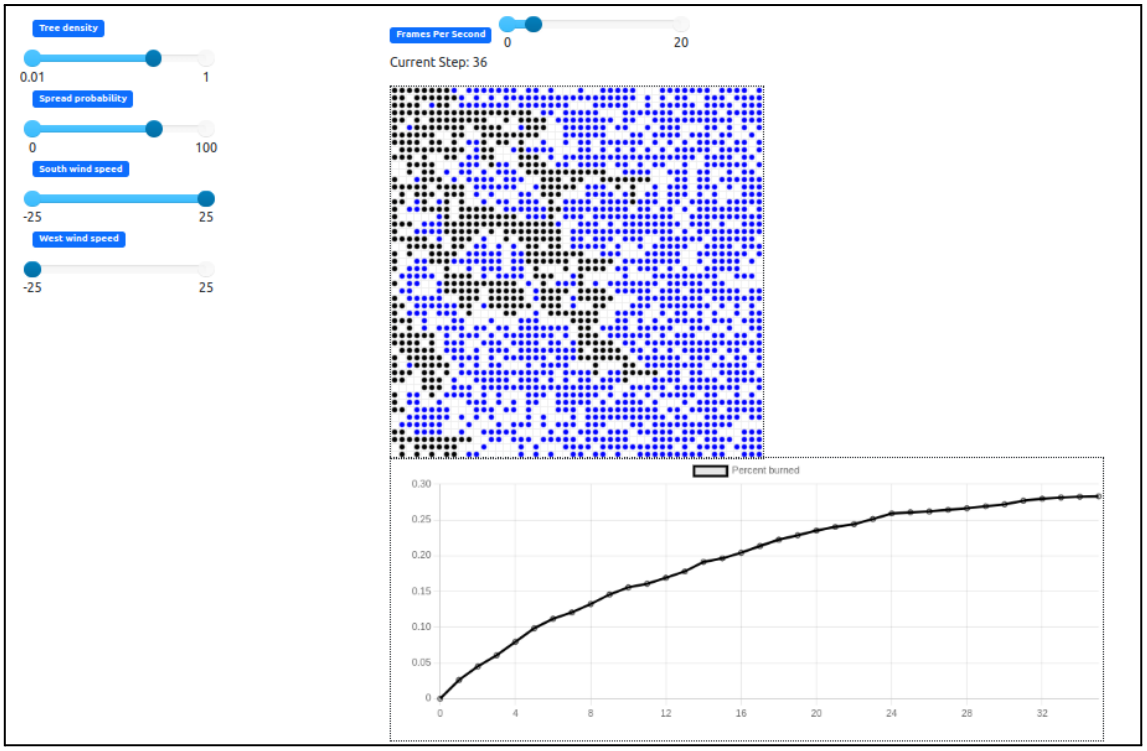

Imagen1.- Ejecución con densidad más realista y 25 de SouthSpeed