Tercera extensión: Permitiendo propagación de largo alcance

Modelación de sistemas multiagentes con gráficas computacionales

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

3.1.- Versión de la extensión

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

Id de commit: **3b5b194**

3.2.- Documentación de cambios

- Implementación de parámetro *big_jumps* a constructor de clase *Tree*.
- Condicional lógica: En caso de que *big_jumps* sea verdadero, conseguir las coordenadas de un árbol del modelo que se encuentre en la posición en x e y afectadas por la velocidad del viento dividida por un factor de escala (8).
- Implementación de parámetro *big_jumps* a constructor de clase *Forest*.
- Creación de incendio en coordenadas con x=15 para una demostración del esparcimiento en las 4 direcciones.
- Implementación de input de tipo Checklist para controlar valores de la variable *big_jumps*.

3.3.- 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, big_jumps):
        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
        self.big_jumps = big_jumps

    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

                #We need to chceck if big jumps is enabled
                if self.big_jumps == True:
                    # Big Jump
```

```
                jumpX = int(self.pos[0] + ((self.west_wind_speed * -1) / 8))
                jumpY = int(self.pos[1] + ((self.south_wind_speed * -1) / 8))

                #We need to check if the jump is valid
                if jumpX >= 0 and jumpX < self.model.grid.width and jumpY >= 0 and jumpY <
self.model.grid.height:
                    #We need to check if the grid position contains a tree
                    if self.model.grid.is_cell_empty((jumpX, jumpY)) == False:
                        jumpTree = self.model.grid.get_cell_list_contents((jumpX, jumpY))[0]
                        #We need to check if the tree is fine
                        if jumpTree.condition == self.FINE:
                            jumpTree.condition = self.BURNING

        # Se apaga el arbol emisor
        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, big_jumps=False):
        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,
big_jumps)
                if x == 15:
                    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", 0.75, 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),
                "big_jumps": UserSettableParameter(
                    "checkbox", "Big Jumps", False),
                "width":50,
                "height":50
                })

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

3.4.- Cambios observados

A continuación se muestra un ejemplo fascinante en el que dos simulaciones muestran la implementación de la variable booleana *big_jumps* y su efecto en cada uno de los pasos de tiempo del modelo. Cabe mencionar que una de ellas inicia su incendio desde el centro del bosque, esto se implementó para mostrar con más facilidad el movimiento del fuego en múltiples direcciones.

El primer ejemplo muestra una ejecución con una velocidad vertical del viento positiva y en su máximo nivel, lo que indica que los vientos vienen desde el norte hacia el sur, así mismo tiene una velocidad horizontal del viento positiva lo que significa que los vientos soplan desde el este hacia el oeste, dejándonos una diagonal que parte desde el centro hasta la esquina inferior izquierda dando pequeños tropezones o saltos que ocurren por la increíble fuerza del viento que empuja chispas hacia los árboles más alejados.

La segunda simulación parte de una línea vertical que se esparce en línea recta de izquierda a derecha pues solo se cuenta con una velocidad horizontal negativa, lo que indica que los vientos soplan del oeste al este. Nuevamente, el fuego se mueve dando saltos entre árboles llegando incluso a dejar unos pocos intactos.

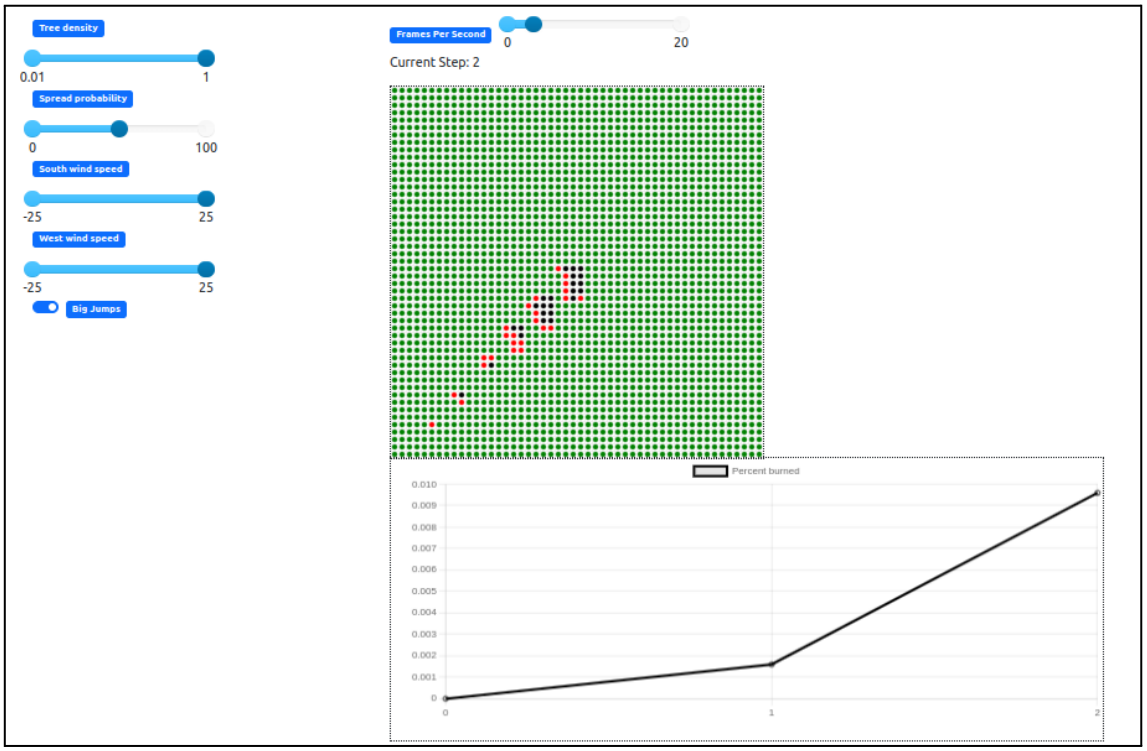

Imagen1.- Ejecución de prueba con punto al centro y Big Jumps activado

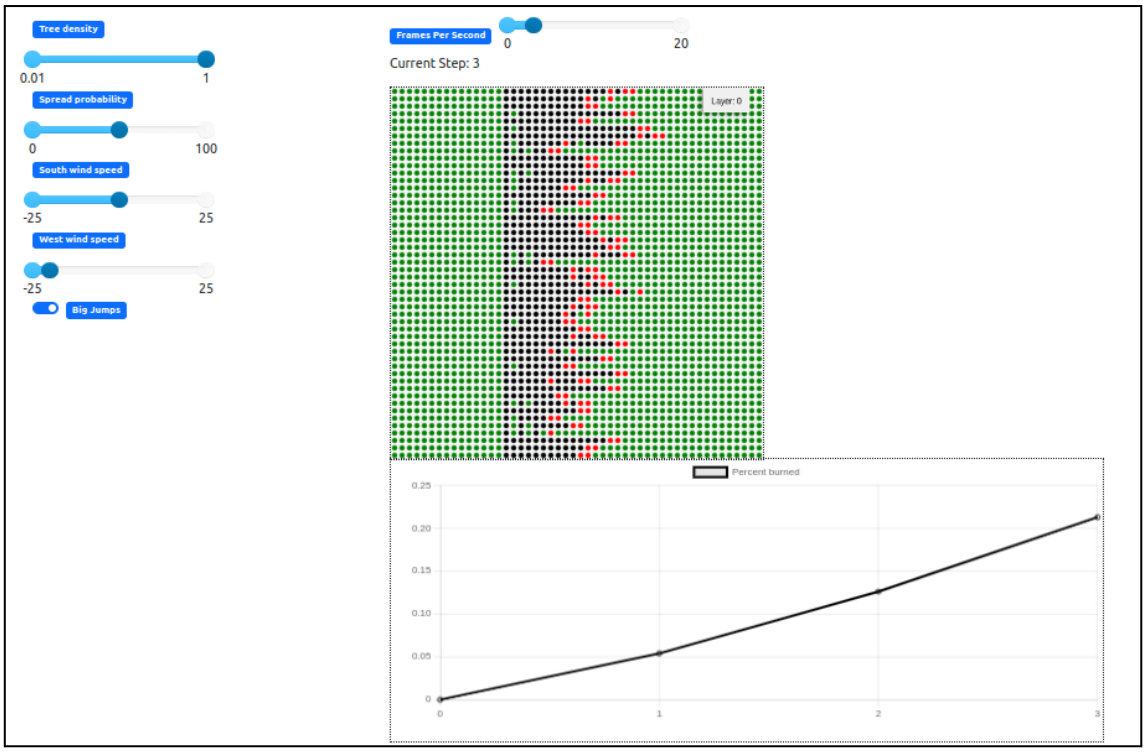

Imagen2.- Ejecución en línea recta con Big Jump activado