

Documentación para correr la simulación

Para correr la simulación y poder visualizarla, lo que se tiene que hacer es lo siguiente. Nuestro programa fue escrito usando Google Colab. Al abrir el programa, se pueden ver varios bloques de código que tendrán que ser inicializados al presionar los botones de play.

```
▶ !pip3 install mesa
```

El primer botón que tiene que ser presionado es para la instalación de Mesa, lo cual viene siendo un framework que nos permite crear modelos basados en interacciones entre agentes.

```
▶ # La clase `Model` se hace cargo de los atributos a nivel del modelo, maneja los agentes.
# Cada modelo puede contener múltiples agentes y todos ellos son instancias de la clase `Agent`.
from mesa import Agent, Model

# Debido a que puede haber más de un agente por celda elegimos `MultiGrid` que permite más de un objeto por celda.
from mesa.space import MultiGrid

# Con `SimultaneousActivation` hacemos que todos los agentes se activen de manera simultanea.
from mesa.time import SimultaneousActivation

# Vamos a hacer uso de `DataCollector` para obtener el grid completo cada paso (o generación) y lo usaremos para graficarlo.
from mesa.datacollection import DataCollector

# matplotlib lo usamos para graficar/visualizar como evoluciona el autómata celular.
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib.colors import ListedColormap, BoundaryNorm
plt.rcParams["animation.html"] = "jshtml"
matplotlib.rcParams['animation.embed_limit'] = 2**128

# Definimos los siguientes paquetes para manejar valores numéricos.
import numpy as np

# Definimos otros paquetes que vamos a usar para medir el tiempo de ejecución de nuestro algoritmo.
import time
import datetime

# Para sumar tuplas
import operator
```

A continuación, se tiene que presionar el segundo botón para importar ciertos paquetes como numpy y matplotlib.

```
▶ def get_grid(model):
    grid = np.zeros((model.grid.width, model.grid.height))
    for cell in model.grid.coord_iter():
        cell_content, x, y = cell

        if len(cell_content) == 0:
            if x == 4 or x == 5 or y == 4 or y == 5:
                grid[x][y] = 7
            else:
                grid[x][y] = 8
        else:
            for content in cell_content:
                if isinstance(content, Semaforo):
                    grid[x][y] = content.estado + 4
                elif isinstance(content, CarrilInterseccion):
                    grid[x][y] = 7
                elif isinstance(content, Carro):
                    grid[x][y] = content.numCarril

    return grid

class Carro(Agent):
    def __init__(self, unique_id, model, numCarril, direccionCarril, destino, posFinal, semaforo):
        super().__init__(unique_id, model)
        self.numCarril = numCarril
        self.direccionCarril = direccionCarril
        self.destino = destino
```

Después, se tiene que presionar el tercer botón para declarar a todos los agentes y al modelo.

```
▶ print('Número del 1-5 indicando la frecuencia con la que aparecen nuevos carros,')
  FRAMES = int(input('1 siendo lo más frecuente y 5 siendo lo menos frecuente: '))
  TIEMPO_MAX = float(input('Tiempo de ejecución (segundos): '))

  start_time = time.time()
  model = Interseccion(FRAMES+1)

  while((time.time() - start_time) < TIEMPO_MAX):
      model.step()

  print('\n' + str(model.cuentaCarrosCruzados) + ' carros cruzaron la intersección.')
```

```
☞ Número del 1-5 indicando la frecuencia con la que aparecen nuevos carros,
  1 siendo lo más frecuente y 5 siendo lo menos frecuente: 2
  Tiempo de ejecución (segundos): 0.05
```

Al presionar el cuarto botón, nos permite crear una instancia del modelo la cual será desplegada a continuación. Al presionar este botón, se pedirá que el usuario escriba dos valores: Un número del 1-5 indicando la frecuencia con la que aparecen nuevos carros (1 siendo lo más frecuente y 5 siendo lo menos frecuente) y el tiempo total de ejecución (es recomendado un tiempo alrededor de 0.03 a 0.07 segundos para que no tarde mucho la simulación).

```
[15] all_grid = model.datacollector.get_model_vars_dataframe()
```

```
[16] %%capture

  cmap = ListedColormap([(139/255, 0, 139/255), (0, 0, 1), (139/255, 69/255, 19/255), (1, 165/255, 0),
                        (0, 1, 0), (1, 1, 0), (1, 0, 0), (67/255, 67/255, 67/255), (180/255, 213/255, 170/255)])
  bounds = [0,1,2,3,4,5,6,7,8,9]
  norm = BoundaryNorm(bounds, cmap.N)

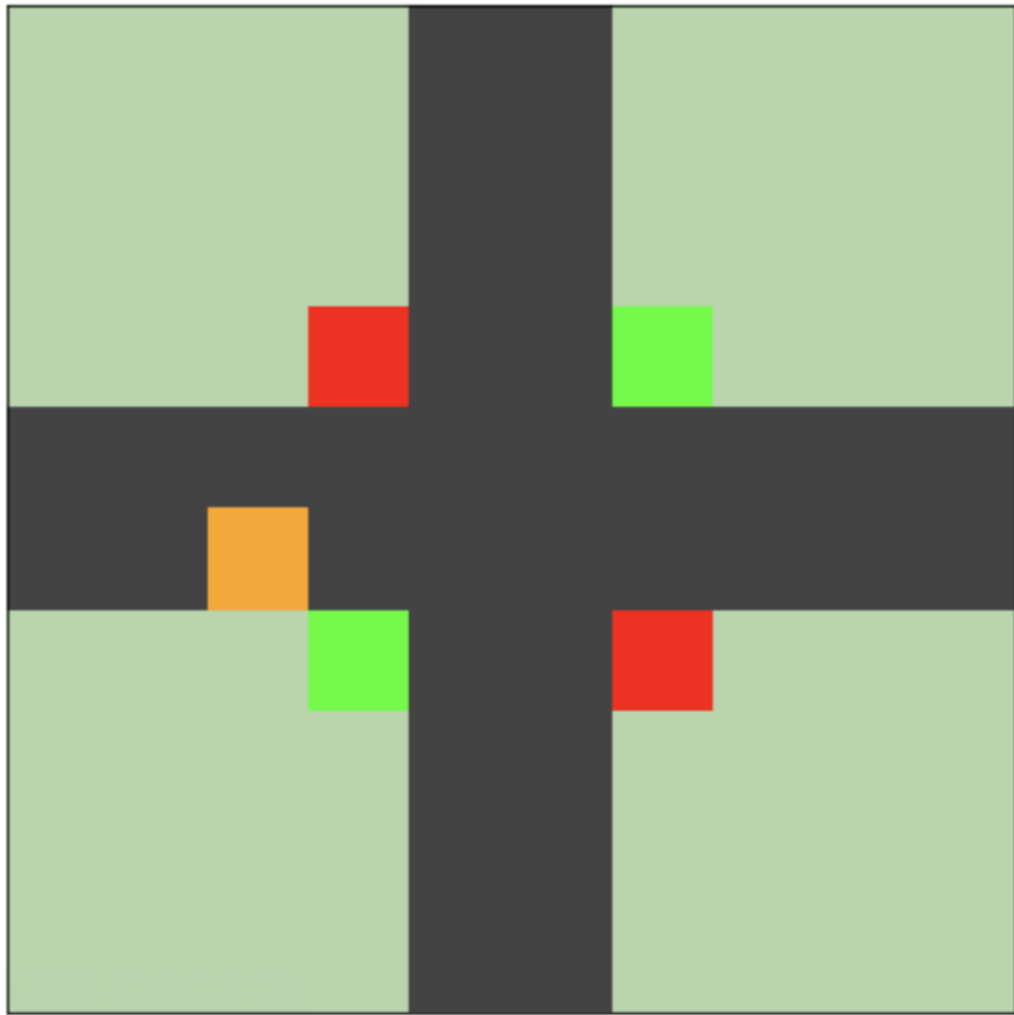
  fig, axs = plt.subplots(figsize=(7,7))
  axs.set_xticks([])
  axs.set_yticks([])
  patch = plt.imshow(all_grid.iloc[0][0], cmap, norm)

  def animate(i):
      patch.set_data(all_grid.iloc[i][0])

  anim = animation.FuncAnimation(fig, animate, frames=len(all_grid))
```

```
[17] anim
```

Los siguientes tres bloques permiten que la instancia del modelo pueda ser visualizada a través del uso del framework Matplotlib. Al presionar los 3 botones, se podrá ver en la parte de abajo la simulación.

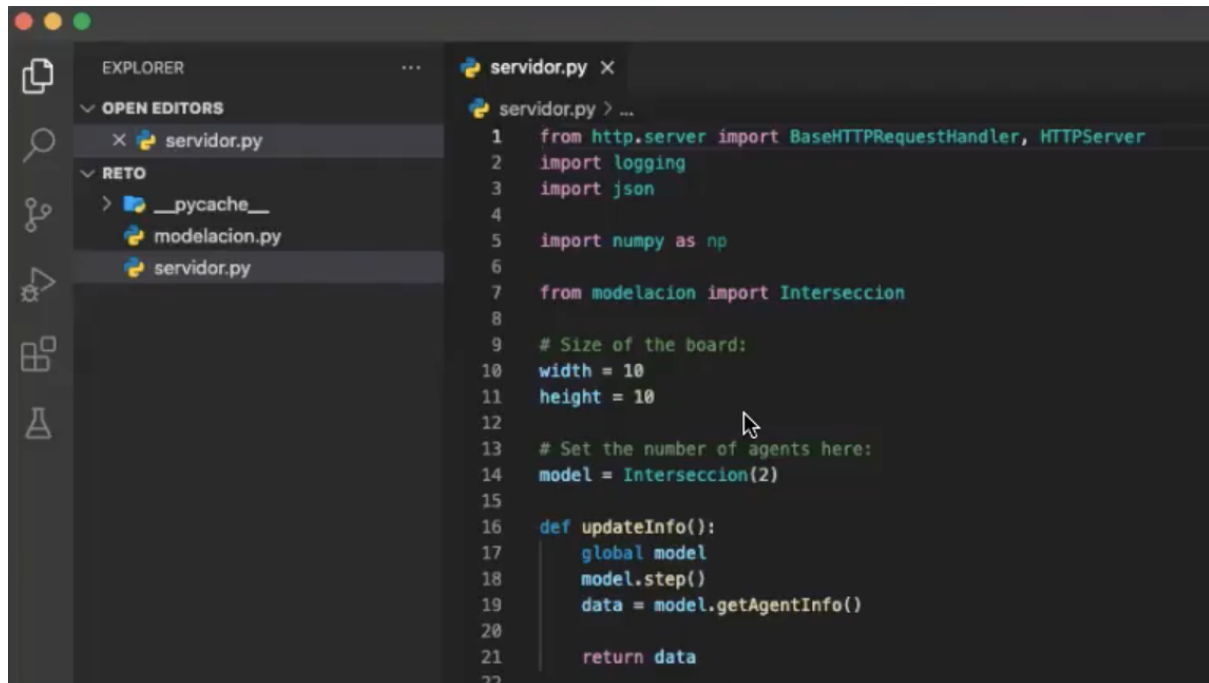


☐ Once ☒ Loop ☐ Reflect

Al picarle al botón de play, se podrá visualizar la simulación.

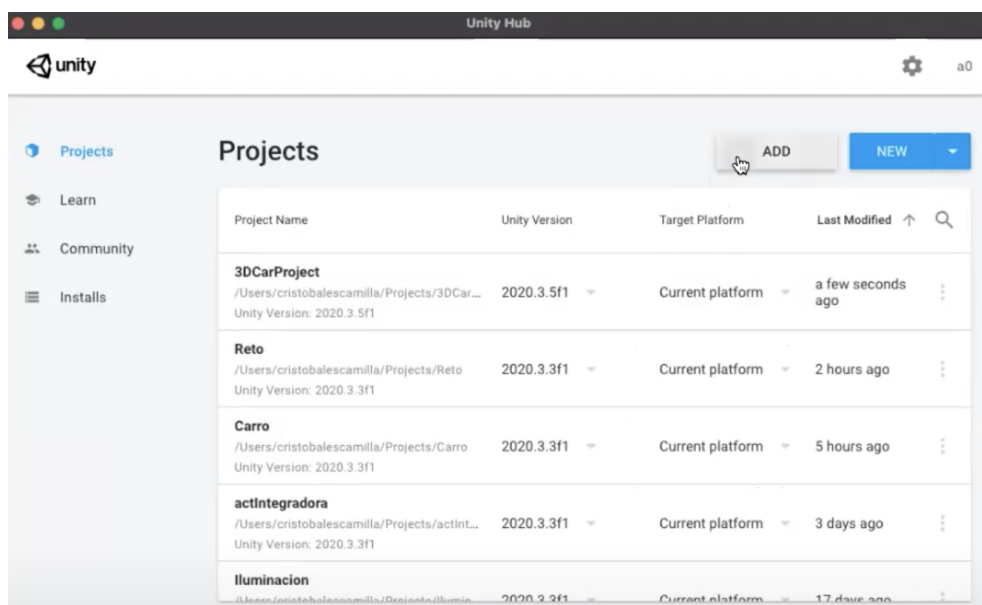
Documentación para correr la simulación en Unity

Lo primero que se tiene que hacer es descargar los archivos “modelacion.py” y “servidor.py”. Cuando se descarguen los dos archivos, el siguiente paso es abrir los dos archivos en el IDE de preferencia (en este caso usaremos VS Code) y correr solamente el archivo “servidor.py”.



```
1 from http.server import BaseHTTPRequestHandler, HTTPServer
2 import logging
3 import json
4
5 import numpy as np
6
7 from modelacion import Interseccion
8
9 # Size of the board:
10 width = 10
11 height = 10
12
13 # Set the number of agents here:
14 model = Interseccion(2)
15
16 def updateInfo():
17     global model
18     model.step()
19     data = model.getAgentInfo()
20
21     return data
22
```

El siguiente paso es descargar el folder que contiene todos los archivos de la simulación en Unity. Al ya tener ese folder, lo que se tiene que hacer es abrir el Unity Hub, picarle al botón en la parte superior derecha que dice “ADD” y seleccionar el folder que acabamos de descargar. Luego se tiene que dar doble click en la sección de Projects al folder correspondiente.



Cuando se abre el proyecto en Unity, lo último que se tiene que hacer es picarle al botón de play que se encuentra en la parte superior y se podrá ver la simulación de nuestro modelo.

