Matemáticas y ciencia de datos para la toma de decisiones

Prof. Luis Palomino Ramírez

Prof. Omar Mendoza Montoya

Actividad integradora

Grupo 1

Tecnológico de Monterrey | Campus Guadalajara

Joshua S. Hernandez Reza A01246538

Noviembre 2021

Introducción.

Esta actividad integradora consta de generar una simulación grafica del tráfico que transita y una calle de doble sentido con solo dos carriles y dos semáforos (uno para cada sentido) que facilita la funcionalidad de un paso peatonal. Para poder lograr esto la solución tendrá dos partes una encargada de generar una visualización de la simulación y la otra encargada de simular el comportamiento de carros y semáforos.

Sistema multiagentes.

Agentes.

La función de esta parte de la solución utilizara Python y su librería de [agentpy] para generar un ambiente continuo bidimensional donde existirán dos tipos de agentes autos y semáforos. Ambos tipos de agentes fueron diseñados como agentes de estados con su respectivo diagrama ilustraciones uno y dos.

Ilustración 1 (Diagrama de agente semáforo)

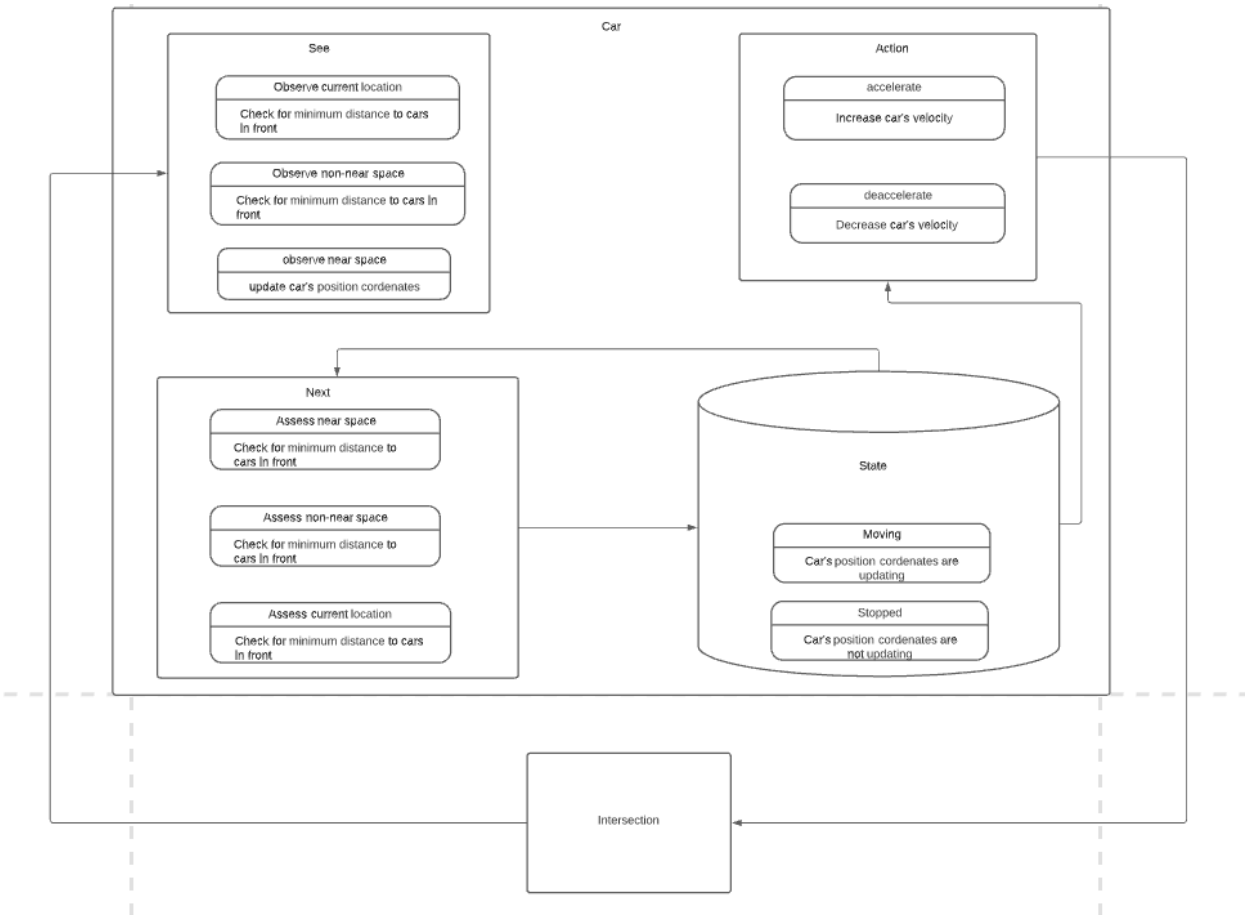

Ilustración 2 (Diagrama de agente carro)

Como podemos observar el agente semáforo es muy sencillo ya que solo cambia de estado (color) evaluando el tiempo que debe durar el estado actual (rojo o amarillo) o el input del ambiente (verde) para simular tránsito de peatones. EL tiempo que se le asigna a cada estado del semáforo es importante ya que si el rojo o el verde se mantiene por mucho tiempo no permitiría apreciar la interacción entre los semáforos y los carros o si el amarillo dura muy poco esto puede causar choques entre los vehículos debido al frenado abrupto.

Por otro lado, el agente de carro es mas complejo aun cuando tiene menos estados (en movimiento o detenido). Esto se debe a que el vehículo tiene que evaluar mas variables para escoger una acción como lo es la distancia del vehículo que está enfrente (si es que hay uno), la distancia del semáforo al agente y el estado del semáforo para decidir si frenar, acelerar o mantener la velocidad. También es importante la prioridad que se le da a estas variables.

Por último, todos los estados del semáforo son discretos mientas que el estado de “en movimiento” de un vehículo es continuo ya que es una simplificación de la velocidad del vehículo cuando es mayor a cero.

Ambiente.

En este caso los agentes por si solos no es posible generar una simulación sin un entorno en el que los agentes sean posicionados y en el caso de los agentes carro se puedan desplazar. El entorno que se utiliza en esta actividad es un entorno continuo (space) ya que permite que el movimiento de los vehículos sea más realista a comparación de un entorno discreto (grid). Otra ventaja del entorno continuo es que permite que los limites del espacio se conecten, permitiendo así que los vehículos que salen del entorno por el límite superior aparezcan por el limite inferior. Aun que parezca una funcionalidad cualquiera nos permite reutilizar los mismos agentes una y otra vez sin necesidad de eliminar los agentes que salieron del entorno y generar nuevos desde una posición inicial.

Animación Gráfica.

La visualización de los resultados de la simulación es hecha con javascript utilizando la librería de THREE. Esta recibe como entrada dos archivos uno para los carros y otra para los semáforos.

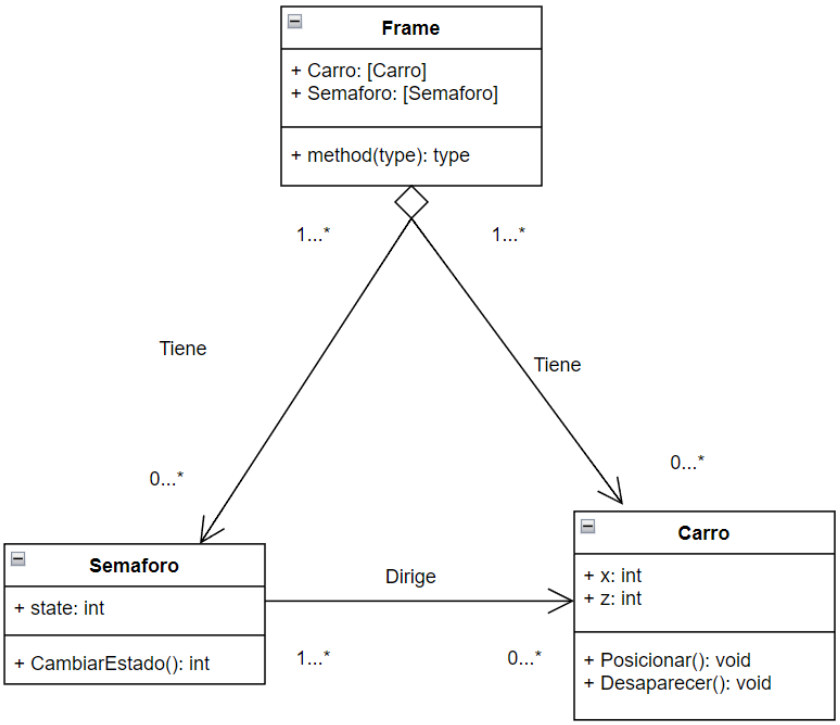

Ilustración 3 (Diagrama de clases)

En el diagrama de clases podemos apreciar que por cada cuadro de la animación el programa utiliza el estado de cada semáforo para saber que color debe de estar en el cuadro actual, así como las coordenadas “x” y “z” para posicionar al

vehículo dando la apariencia de que el carro se mueve. El objeto json para los carros que la animación utiliza se ve de la siguiente manera:

Car_data.json

```
{
        0: [
                {
                        “x”:  10.53,
                        “z”: 15.34,
                        “id”: 0
                },
                {
                        “x”:  10.53,
                        “z”: 15.60,
                        “id”: 1
                },
                {
                        “x”:  10.53,
                        “z”: 15.89,
                        “id”: 2
                }
        ]
        .
        .
        .
}
```

Este objeto contiene una llave que va del 0 a k – 1; donde k es la cantidad de cuadros totales. Dichas llaves mapean a una lista de objetos la cual es de tamaño c; donde c es la cantidad de vehículos en la animación. Cada objeto dentro de esta lista representa un vehículo único con las coordenadas donde se encuentra el vehículo en ese cuadro y su id único. Como podemos apreciar el valor de x de todos los carros que

están en el mismo, solo el valor de “z” cambia cada cuadro. Esto de debe a que los carros se mueven en línea recta.

Traffic_light_data


```
{
    0: [
        {
            “state”: 0,
            “id”: 0
        },
        {
            “state”: 0,
            “id”: 1
        }
    ]
    .
    .
    .
}
```


Este objeto es mas sencillo ya que mantiene la misma estructura que el objeto de car_data.json con la excepción de que los objetos dentro de las listas representan un semáforo con su estado y id único. El estado de los semáforos se puede mantener igual por varios cuadros ya que el estado de un semáforo tarda mas en cambiar que un solo cuadro.

Estos datos son leídos antes de que la animación comience y los valores de cada vehículo y semáforo son actualizado a través de la función update_scene().

Por último, debido a que los edificios y los semáforos son objetos estáticos la creación de estos fue hardcore.