INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY

SEDE: Santa Fe

Reflexión Individual Simulación Tráfico

Modelación de sistemas multiagentes con gráficas computacionales (Gpo 301)

Alumno:

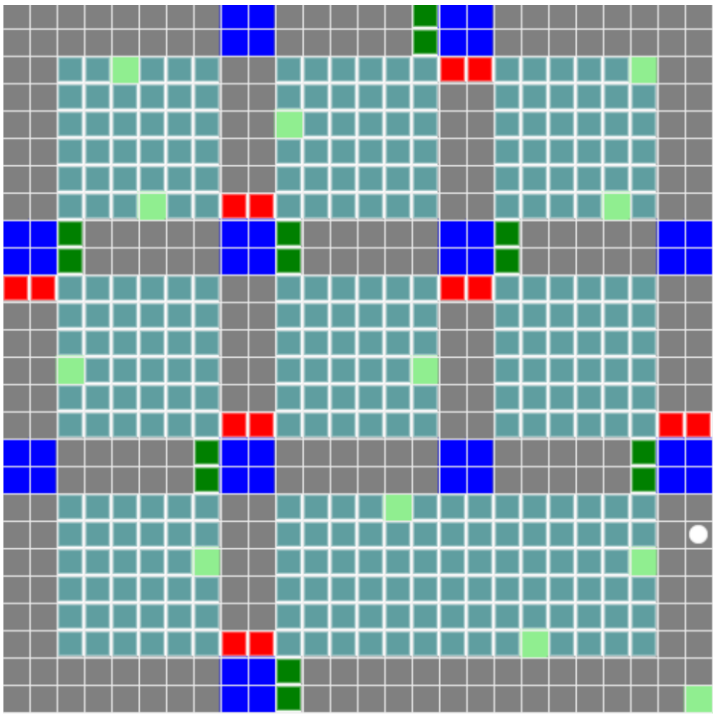
Fernanda Nava Moya A01023896

Análisis de la solución desarrollada

Para la situación problema se crearon los siguientes Agentes con sus respectivas acciones:

- **Traffic_Light**
    - Estado (booleano: verde, rojo)
    - Tiempo en que cambian entre estos valores booleanos
- **Destination**
    - Coordenadas en donde se encuentra
- **Obstacule**
    - Edificio dentro de la ciudad
- **Intersection**
    - Es un nodo
- **Road**
    - Dirección de tráfico (arriba, abajo, derecha, izquierda)
- **Car**
    - Estado (booleano: andando, parado)
    - Destino (coordenadas)
    - Intersección actual (ID de intersección)
    - Intersección próxima (ID de intersección)
    - Posición (coordenadas de vehículo)
    - Se mueve diferente en intersecciones y calles
    - Encuentra la ruta más rápida

Algo que se tomó en consideración fué la manera en que los coches iban a encontrar la ruta para llegar a su destino respetando el sentido de las calles y el estado de los semáforos. Por lo que decidimos que nuestras intersecciones fueran nodos los cuales estan conectados entre si:



De esta manera podemos utilizar el algoritmo de Floyd Warshall el cual consiste en crea una matriz de una dimensión n*n donde ‘n’ es el número de vértices, en este caso la cantidad de intersecciones, sus índices representan los vértices y sus valores es el peso de la conexión entre esos vértices. Una vez que la matriz es creada, entonces se encuentra la intersección intermedia del camino más corto desde el origen al destino. Esta intersección es la variable ‘u’ para que en A[i][j] se inserte (A[i][u] + A[u][j]) if (A[i][j] > A[i][u] + A[u][j]) para que si la distancia directa desde la fuente al destino es mayor que la ruta a través de la intersección ‘n’, entonces se inserta A[i][n] + A[n][j]. Se calcula la distancia desde la intersección de origen a la intersección de destino a través de esta intersección ‘n’. La complejidad de este algoritmo es de $O(V^3)$**:**

```
for k in range(len(self.adjacent)):
        for i in range(len(self.adjacent)):
            for j in range(len(self.adjacent)):
                if(self.adjacent[i][j] > (self.adjacent[i][k] + self.adjacent[k][j])):
                    self.adjacent[i][j] = self.adjacent[i][k] + self.adjacent[k][j]
                    self.next[i][j] = self.next[i][k]
```

Consecuentemente, para asi encontrar la ruta más óptima entre la posición del vehículo y la posición de su destino. De esta manera, el vehículo no da vueltas por toda la ciudad tratando de encontrar su destino.

Dentro de la simulación el agente Road nunca cambia su estado ni se mueve, unicamente tiene un atributo *destination* el cual nos dice el sentido de la calle (arriba, abajo, derecha o izquierda) y los agentes de tipo Destination no Similarmente, los agentes de tipo Obstacle no se mueven en ningún momento simplemente existen y son los vehículos los que los esquivan. De igual manera, el semáforo no se va a mover nunca durante la simulación pero su estado si: entre rojo y verde. Por otro lado, el carro se mueve de acuerdo al sentido de la calle, respetando el estado del semáforo, y buscando las ruta más eficiente para llegar a su destino.

Nuestra solución es óptima ya que un vehículo no da iteraciones de mas intentando buscar su destino ya que obtiene la mejor ruta posible respetando el sentido de las calles, el estado del semáforo y esquivando obstáculos y otros vehículos. Sin embargo, algo que podríamos mejorar es permitir que los vehículos puedan dar vuelta desde el segundo carril de esta manera se disminuye el trafico y, en caso de haber muchos vehículos que llega a haber fila hasta el alto, no obstruye el paso a otros vehículos

Reflexión sobre proceso de aprendizaje

Al principio de este bloque no sabía que eran multiagentes y mucho menos como conectar estos multiagentes con Unity para poder crear una simulación. Me gustó aprender a utilizar mesa ya que realmente se puede llegar a simular muchas cosas con esta herramienta. De igual manera, utilizar Unity para modelar objetos fue muy interesante ya que se puede importar objetos o se pueden *moldear* desde ahí. Es muy gratificador ver como Unity refleja la simulación de mesa.

De igual manera, algo que me sorprendió fue ver comportamiento emergente de los agentes tipo vehículos dentro de la simulación ya que algo que no tomamos en cuenta fue la manera en que se incorporan otros vehículos a un carril de la calle. Ya que estos no toman en consideración alguna medida de seguridad por lo que al incorporarse a un nuevo carril este se incorpora aunque solo haya un espacio libre antes de un vehículo se mete a esto lo que no habíamos tomada en cuenta.