Tecnológico de Monterrey

Campus Santa Fe

Modelación de sistemas multiagentes con gráficas computacionales.

TC2008B.302

Situación Problema:

Movilidad Urbana

Alumnos:

David Santiago Vieyra García - A01656030

Fernanda Cantú Ortega - A01782232

Profesores:

Gilberto Echeverría Furió

Octavio Navarro Hinojosa

Fecha de entrega:

30 de noviembre de 2023

1. Introducción

Problemática

La problemática de la movilidad urbana en México se centra en el aumento descontrolado del uso de automóviles, generando impactos negativos en los ámbitos ambiental, social y económico. El incremento en los Kilómetros-Auto Recorridos (VKT) de 106 millones en 1990 a 339 millones en 2010 ha desencadenado problemas de contaminación, congestión vehicular, riesgos para la seguridad vial, dependencia excesiva del automóvil y pérdidas económicas. Estos efectos adversos afectan la calidad de vida, la salud pública, la productividad y el medio ambiente. Es vital implementar estrategias que promuevan alternativas de transporte sostenible y mejoren la eficiencia del sistema de movilidad para abordar este desafío de manera integral.

Propuesta de Solución

La propuesta de solución es una simulación de movilidad urbana que evite generar congestiones viales. Para lograr esto, se utilizaron la librería Mesa, del lenguaje de programación Python, creada para Agent-Based Modelling, y el framework Flask, utilizado para conectar nuestra simulación con nuestro modelo en Unity. Con el conjunto de estas herramientas tecnológicas, se aborda la implementación de una simulación de tráfico 3D que cuenta con agentes inteligentes que perciben y reaccionan al ambiente en el que están situados.

El proyecto se estructura en cinco elementos principales:

- *server.py*: Creación de la interfaz mediante CanvasGrid de la librería Mesa. Reproduce una visualización bidimensional de la simulación y el modelo con una simple rejilla de 24x25 en un servidor web.
- *agent.py*: Definición de las clases para los agentes *Car* (encargados de desplazarse a su destino en nuestro ambiente simulado evitando generar congestiones). También se definen los agentes *Traffic_Light*, *Destination*, *Initialization*, *Obstacle* y *Road*.
- *model.py*: Definición e inicialización del entorno y del modelo. Se definen funciones para la lectura del mapa base, el movimiento y percepción del agente Car, el pathfinding generado con el algoritmo A*. Posteriormente para la competencia se incluye una función que realiza un POST request a un servidor.
- *flask_server.py*: API que permite conectar la definición del modelo y de los agentes en el motor de juegos Unity
- *Unity*: La simulación es desplegada en un ambiente tridimensional conformado por modelos construidos utilizando Blender y archivos OBJ. El movimiento de los coches es controlado utilizando matrices de transformación en un archivo de C#. De igual manera, la recolección de datos de los agentes es controlada dentro de Unity.

2. Diseño de Agentes

La simulación cuenta con 6 diferentes agentes, los cuales representan los semáforos, las calles, el nodo de inicialización, los destinos, los obstáculos, y los coches. De todos, el único agente inteligente son los coches. Los demás agentes son parte del ambiente y afectan directamente las decisiones de los coches, y son factores que el coche debe considerar con el fin de alcanzar su objetivo.

Agentes Car

Los agentes Car, tienen como objetivo principal llegar a su destino siguiendo la ruta más corta, esto es logrado mediante la creación de un grafo del mapa, el cuál es posteriormente procesado por un algoritmo A* de la librería NetworkX que recibe la posición inicial de nuestro agente, la posición final y la heurística. La heurística se obtiene calculando la distancia euclidiana de la posición y el destino.

El agente Car tiene diversas capacidades perceptivas y efectoras que le permiten interactuar con su entorno en un modelo basado en agentes.

Capacidades Perceptivas:

1. Detección de Contenido de Celdas:

   El agente puede percibir el contenido de la celda en la que se encuentra utilizando el método grid.get_cell_list_contents(). Esto le permite identificar la presencia de otros agentes en la misma ubicación.
2. Identificación de Objetos:

   Utiliza el método determine_node_type para identificar el tipo de objeto en la celda actual, como obstáculos, carreteras, nodos de inicialización, destinos, semáforos, etc.
3. Detección de Semáforos:

   Puede identificar la presencia de semáforos y su estado (rojo o verde), lo que afecta su capacidad para moverse.

Capacidades Efectoras:

1. Movimiento:

   El agente tiene la capacidad de moverse en la cuadrícula. El método move calcula y ejecuta el siguiente movimiento en función de una ruta pre calculada utilizando el algoritmo A*.
2. Actualización de Posición:

   Utiliza grid.move_agent() para actualizar su posición en la cuadrícula después de un movimiento válido.
3. Identificación de Destino:

   Puede determinar si ha llegado a su destino y realizar acciones específicas, como eliminarse del modelo y la programación si ha alcanzado el destino.
4. Búsqueda de Ruta:

   Tiene la capacidad de buscar una ruta utilizando el algoritmo A* a través del método find_path. Este método considera las restricciones de movimiento y la topología de la cuadrícula.

5. Verificación de Obstáculos y Semáforos:

   Antes de realizar un movimiento, el agente verifica la presencia de obstáculos y el estado de los semáforos utilizando el método can_move_to, evitando movimientos inválidos.
6. Manejo de Eventos:

   Puede manejar eventos específicos, como la llegada al destino, donde se retira del modelo y la programación.

PEAS

1. Performance: El agente recorre la ciudad en búsqueda de su destino, respetando las restricciones que los demás agentes imponen sobre él.
2. Environment: El agente se enfrenta a un entorno que cuenta con obstáculos y agentes que definen la posibilidad del movimiento del mismo.
3. Actuators: Los actuadores que posee el agente para interactuar con el entorno son la detección de celdas, la asignación del destino al que debe llegar, y la implementación de A* para encontrar el mejor camino.
4. Sensors: Cómo se mencionó anteriormente, el agente Car tiene la habilidad de detectar las celdas en las que se encuentra y puede reaccionar ante esto.

Agentes Traffic Light

Los agentes que representan a los semáforos sirven la función de permitir o bloquear el paso de los agentes Car, con el fin de controlar y regular el movimiento vial. Los semáforos están programados con la función toggle_traffic_lights para cambiar de estado cada cierto tiempo.

Agentes Road

La función de los agentes que representan las calles es guiar a los coches e indicarles a qué dirección se pueden mover cuando el coche comparte celda con ellos. La clase de los agentes Road crea los respectivos agentes. Posteriormente, dentro de la definición del modelo, se define dónde debe ser posicionada cada tipo de calle de acuerdo al mapa base.

Agente Initialization

La clase de los agentes de inicialización genera los nodos donde serán generados los coches. Estos nodos son colocados en cada una de las cuatro esquinas de la rejilla.

Agentes Destination

La clase que define los destinos generan los puntos del mapa a los que los coches deben llegar. Dentro de la definición del modelo se le asigna aleatoriamente un destino a cada coche.

Agentes Obstacles

La clase de los agentes de obstáculos crea los puntos del mapa donde estarán colocados los edificios, los cuáles impiden el paso de los coches.

Arquitectura de Subsunción

La arquitectura de subsunción empleada permite priorizar las acciones de los agentes Car, garantizando que lleguen a su destino, evitando obstáculos, respetando semáforos y siguiendo la dirección de las calles como objetivos primordiales.

Comportamiento Prioritario - Evitar colisiones con obstáculos

- Condición de Activación: Detección de obstáculos en el radio de percepción
- Acción: Cambiar la dirección de movimiento para evitar el obstáculo

Comportamiento - Manejo de destino

- Condición de Activación: Compartir celda con el destino asignado
- Acción: El agente es removido de la simulación

Comportamiento - Planificación de ruta

- Condición de Activación: Tener un destino asignado
- Acción: Encontrar la ruta más corta desde el origen hasta el destino correspondiente

Comportamiento - Responder a semáforos

- Condición de activación: Percibir estado del semáforo
- Acción: Si el semáforo está en rojo, parar. Continuar el camino cuando el semáforo esté en verde.

Comportamiento - Movimiento General

- Condición de Activación: Cuando no se activan comportamientos prioritarios.
- Acción: Moverse en la dirección correspondiente a la celda de la calle en la que esté posicionado el agente.

Acciones (ordenadas de menor a mayor prioridad)

- Capa de Navegación Básica: Esta capa maneja el movimiento básico del coche, seguir las reglas básicas del camino (como seguir la dirección del camino), y moverse a la siguiente posición de su path.
- Capa de Semáforos: Esta capa se encarga de interactuar con los semáforos. Si un semáforo está en rojo, el coche debe detenerse. Esta lógica se puede manejar en la función can_move_to, comprobando si hay un semáforo en rojo en la próxima posición.
- Capa de Planificación de Ruta: Esta capa utiliza un algoritmo de búsqueda de rutas para determinar la mejor ruta hacia el destino del coche, teniendo en cuenta las condiciones actuales del tráfico y otros factores. Se basa en la función find_path.
- Capa de Manejo de Destino: Esta capa verifica si el coche ha llegado a su destino. Si es así, realiza las acciones necesarias, como eliminar el agente del modelo o actualizar las estadísticas. Esto se maneja en la función arrived_at_destination.
- Capa de Prevención de Colisiones: Esta capa supervisa si en la próxima posición hay obstáculos como otros coches para evitar colisiones.

Métricas de Desempeño

En el contexto de la simulación, cuyo propósito es diseñar y proponer un sistema de vialidad eficiente para prevenir congestiones, consideramos que la evaluación del rendimiento recae en dos principales métricas. Para esta evaluación, es importante considerar que los agentes vehiculares en esta simulación tienen un objetivo singular: llegar a sus destinos asignados. En consecuencia, se pueden identificar dos métricas clave para medir el éxito de nuestro sistema.

La primera métrica es la tasa de éxito, que se refiere al número de vehículos que logran llegar a sus destinos en un período de tiempo específico. Cuanto mayor sea esta tasa, más efectivo será nuestro sistema en garantizar que los vehículos cumplan sus objetivos sin demoras innecesarias.

La segunda métrica es la capacidad operativa de la simulación, que se relaciona con la cantidad máxima de vehículos que el sistema puede acomodar simultáneamente. Un sistema bien implementado permitirá que un mayor número de vehículos circulen de manera fluida en la simulación sin crear congestiones, lo que se traduce en una vialidad más eficiente y menos tráfico.

Estas métricas combinadas proporcionarán una evaluación integral del desempeño de nuestro sistema de vialidad, permitiendo así una evaluación precisa de su efectividad en la prevención de congestiones y la gestión eficiente del tráfico vehicular.

3. Diseño del Ambiente

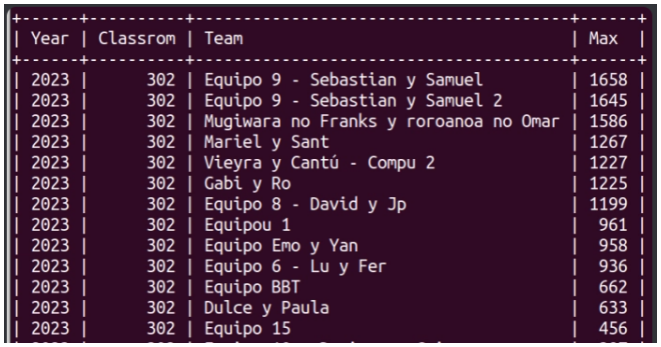
El entorno simulado se despliega como un espacio tridimensional cuidadosamente diseñado para emular con gran fidelidad una ciudad. Este espacio virtual cobra vida con la presencia de imponentes edificios, intrincadas redes de calles y semáforos completamente operativos. La accesibilidad al entorno es sobresaliente, ya que nuestros agentes disponen de información completa, precisa y en tiempo real sobre su estado. Este entorno ha sido meticulosamente configurado como determinista, lo que significa que no existe incertidumbre en relación con las consecuencias de las acciones que emprenden nuestros agentes. La capacidad de los coches de influir en el entorno es limitada; en lugar de modificarlo, su principal función es navegar por él, asegurando así que el efecto de sus acciones esté plenamente garantizado.

Una característica esencial de este entorno es su naturaleza episódica, lo que significa que nuestros agentes pueden tomar decisiones basadas exclusivamente en el episodio actual, sin necesidad de proyectar posibles futuros episodios. Además, este ambiente exhibe dinamismo gracias a la presencia de semáforos cuyos procesos son externos a los agentes y que influyen activamente en el estado del entorno. Por último, este entorno se presenta como una amalgama de aspectos discretos y continuos: inicialmente, permitimos que la simulación se ejecutara sin restricciones temporales, lo que nos permitió explorar exhaustivamente el comportamiento de los agentes y detectar cualquier anomalía en la simulación. Sin embargo, durante la competencia, se nos indicó que debíamos imponer un límite de pasos para medir con precisión la métrica de éxito, que como se mencionó anteriormente, se basa en el número de agentes que logran alcanzar sus destinos en un período de tiempo/steps específico.

4. Análisis de Resultados y Conclusiones

Resultados

Tras llevar a cabo la simulación y varias pruebas, se observó que la cantidad de coches que llegan a su destino en un determinado tiempo es bastante alta. Se hizo una prueba de 1000 pasos, la cuál dió como resultado un total de 1227 coches que alcanzaron su destino. Esto nos posicionó en el 4to lugar de la final de la competencia.



Year	Classrom	Team	Max
2023	302	Equipo 9 - Sebastian y Samuel	1658
2023	302	Equipo 9 - Sebastian y Samuel 2	1645
2023	302	Mugiwara no Franks y roroanoa no Omar	1586
2023	302	Mariel y Sant	1267
2023	302	Vieyra y Cantú - Compu 2	1227
2023	302	Gabi y Ro	1225
2023	302	Equipo 8 - David y Jp	1199
2023	302	Equipou 1	961
2023	302	Equipo Emo y Yan	958
2023	302	Equipo 6 - Lu y Fer	936
2023	302	Equipo BBT	662
2023	302	Dulce y Paula	633
2023	302	Equipo 15	456

Basado en las métricas de desempeño establecidas anteriormente y en comparación con los demás equipos, este es un gran resultado. También se observó que, con un tiempo de actualización de 0.3, la mayor cantidad de coches que aguanta la simulación es de 121. Llegar a dicho punto tomó aproximadamente 1700 pasos, con 1531 carros que llegaron a su destino.

Conclusiones

Analizando las pruebas y las métricas de desempeño, se puede concluir que la simulación fue un éxito. Sin embargo, en el futuro nos gustaría implementar mejoras para que la simulación sea más eficiente y que soporte más coches al mismo tiempo sin que haya conglomeraciones. De igual manera, nos gustaría implementar parámetros para que los coches sean más reactivos. Un ejemplo de esto es la paciencia, y que si un coche lleva esperando más de X tiempo, se “desespere” y cambie de carril o recalcule su ruta. Estaría interesante incluir otros elementos de la ciudad, como peatones, patrullas y camiones,

5. Conclusiones Finales

La simulación demostró cumplir con el objetivo de la situación problema: Proponer y generar un modelo de vialidad que impida congestiones viales y muestre movimientos viales eficientes. Los conocimientos adquiridos nos permitirán seguir diseñando soluciones a problemas cotidianos y nos impulsarán como alumnos y como científicos en computación. Además, este tipo de simulaciones son ejemplos perfectos de cómo la tecnología puede ser utilizada para mejorar la calidad de vida de las personas, facilitando sistemas de transporte más eficientes y sostenibles. Esto nos posiciona en un lugar privilegiado al adentrarnos en la contribución a la solución de retos cotidianos mediante el uso de la tecnología y la ciencia de datos.

Referencias:

Medina Ramírez, Salvador. (2012). Transforming Urban Mobility in Mexico: Towards Accessible Cities Less Reliant on Cars. Institute for Transportation and Development Policy (ITDP Mexico). Recuperado el 29 de noviembre de 2023, de: http://mexico.itdp.org/wp-content/uploads/Transforming-Urban-Mobility-in-Mexico.pdfLinks