Campus Monterrey

Análisis de la Solución Desarrollada para el proyecto de Sistemas Multiagentes

Materia
Modelación de sistemas multiagentes con gráficas computacionales (Gpo 4)

Profesores
Edgar Covantes Osuna
Jorge Cruz Duarte

Integrantes

Nicolás Cárdenas Valdez A01114959

Septiembre 8, 2021

Reflexión

Primero que nada me gustaría discutir el razonamiento detrás nuestra decisión de escoger el modelo de multiagentes que hicimos, antes de haber realizado prácticas con la plataforma de MESA en conjunto con Python nosotros ideamos un modelo idealista donde todo tenía sus características y lo que iba a necesitar. Sin embargo, ya habíamos usado MESA y tras realizar estas prácticas nos dimos cuenta que cambiaban mucho las cosas para poder simular lo que se nos requería. Cambiaron muchas cosas en cuanto a los agentes, sus métodos y sus atributos. La herramienta de MESA fue utilizada por su gran funcionalidad en ayudar y complementar simulación de sistemas multiagentes lo cual es justo lo que necesitábamos, y el lenguaje python es muy amigable lo cual fue un inmenso beneficio al momento de programar la simulación, ya que al principio no entendíamos mucho de cómo funcionaba y tuvimos que meternos profundamente en la documentación de está gran herramienta. Como lo mencione anteriormente, el modelado fue muy diferente antes de haber tocado MESA al modelado que realizamos posteriormente a haber utilizado MESA, algunos detalles grandes que cambiamos fue el hecho de que no necesitábamos agentes de conductor o de los sensores. Los sensores los convertimos en atributos de los semáforos ya que fue más fácil programarlos de está manera y creo que asimila más como funciona en vida real. En cuanto al conductor, encontramos que no es necesario tenerlo como una variable separada ya que todas sus acciones vendrían siendo las acciones del carro, por lo menos en este contexto. Después de haber tenido un poco mas de experiencia tanto con el modelado y las herramientas para dicho modelado creo que al final terminamos con las funciones y atributos que en realidad necesitábamos, al final del día, pudimos modelar el sistema de MESA muy similarmente a como lo habíamos planeado, es decir, sus funcionalidades y con las semejanzas que se necesitan en vida real: semáforos con sensores, cada carro haciendo sus propias decisiones. Al final de todo, todos los agentes realizaban las “acciones” que tenían que realizar dentro del programa, por ejemplo, el semáforo no “movía” al carro dentro del programa, el carro “checaba” el semáforo para ver si se podía mover lo cual nos ayuda mucho a asimilarlo a vida real y sea una solución más cercana a lo que se nos pide en la situación problema, además de esto nos permite hacer las modificaciones a solamente los carros cuando queremos cambiar algún atributo de los carros, no tener que estar cambiando los métodos o atributos en otras clases de los agentes y creo que esa es una de las ventajas de nuestra solución pero también de hacer el modelado y simulación de está manera porque al final del dia, buscamos representar la vida real lo más que se pueda. Algunas cosas que podría delimitar como desventajas es que nuestra simulación hace uso del espacio *toroidal* por lo cual no crea nuevos carros en la simulación si no que “rota” los carros alrededor del espacio y es el mismo carro por lo cual no se acerca a la realidad pero creo que es la única limitación que tenemos en nuestra solución que desarrollamos. . La manera que se nos presentó para visualizar la simulación creada por MESA fue usar las funciones de animación de matplotlib, la cual es igualmente una librería de Python. Está se nos presentó y se nos enseñó desde un principio por lo cual no hubo muchas decisiones en ese aspecto. Lo que decidimos cambiar fue la *colormap* que se usó, usamos uno que abarcara todos los colores del arcoiris para poder seleccionar en específico que color usamos para cada elemento de la simulación y se pudiera

entender mejor. Los colores los cambiábamos en su función de *get_grid* por lo cual fue fácil tener diferentes colores para los semáforos activos e inactivos. Tratamos de asimilar todos los colores a sus aspectos en la vida real. Todos los carros en la simulación de MESA eran del mismo color (azul) pero en nuestra visualización en Unity, cada uno tenía aspectos diferentes ya que solo pasabamos las "coordenadas" de los objetos a Unity para que se pudiera mover de acuerdo al movimiento que se generó durante la simulación en MESA. En cuanto visualización utilizamos Unity ya que es el que más sabemos utilizar y en el cual nos dieron las herramientas para poder llevar a cabo la solución por lo tanto no había otra alternativa. La verdad se me hizo muy conveniente la manera en la que realizamos la solución, es decir, los pasos o fases que tomamos para poder llevarlo a cabo. Hacer la simulación en Python/MESA y después conectarlo a Unity con las herramientas que me han dado ya que la conexión fue más simple, en mi opinión, de programar que hacer toda la simulación en Unity. Fue muy práctico y lo considero una de las ventajas más grandes de nuestra solución ya que fue en base a cómo fuimos aprendiendo las herramientas. Finalmente creo que la solución fue desarrollada a como se nos pidió a pesar de varios retos en la conexión de Python a Unity pero nuestro equipo lo pudo sacar adelante ya que lo hicimos con tiempo y varios de nosotros se especializan en cosas diferentes. Si tuviera que volverlo a hacer con algunas modificaciones, en Unity haría que se crearán nuevos carros cuando los carros dejaban la intersección y no que volvieran mediante el espacio toroidal, ya que el hecho de que sea un nuevo carro asimila más la vida real y podríamos darles nuevos colores o atributos a cada carro cada vez que se acerca a la intersección, esto seria más en preparación para futuras mejoras al modelo, pero en general creo que ha quedado muy bien la solución gracias al trabajo y esfuerzo de todos mis compañeros..