Noviembre 2022

Evidencia 1

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

Modelación de sistemas multiagentes con gráficas computacionales

TC2008B.302

Edgar Covantes Osuna

Jose Daniel Azofeifa Ugalde

Presenta:

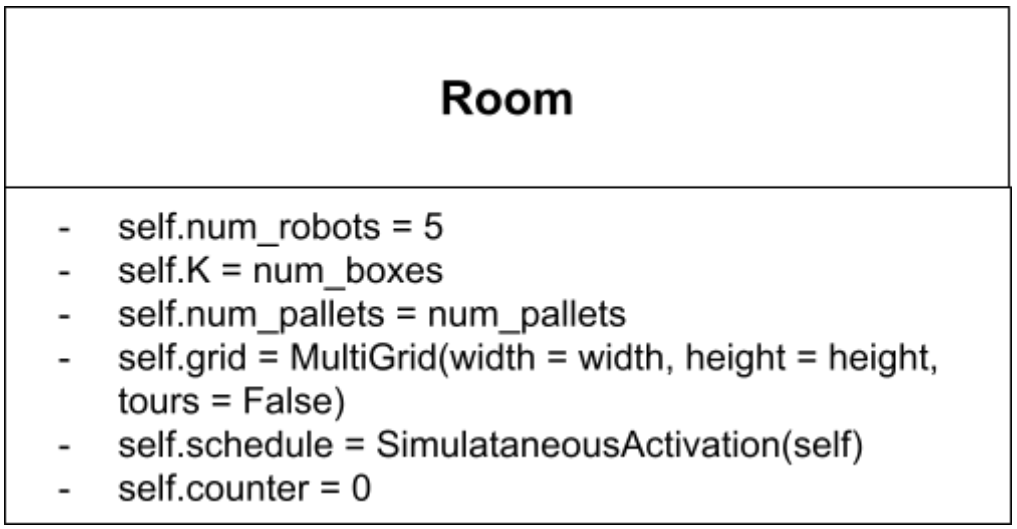
Alberto Estrada Guerrero | A01276671
José Gerardo Cantú García | A00830760
Franco Sotomayor Casale | A00831450
Arturo Garza Campuzano | A00828096
Marcelo Márquez Murillo | A01720588
Edgar Alexandro Castillo Palacios | A00830568

Diagramas de clases

Pallet
- unique_id - model - self.box_count = 0

Box
- unique_id - model - self.is_grabbed = False - self.on_pallet = False - self.pallet_pos = None

Robot
- unique_id - model - self.grabbed_box = False - self.on_pallet = False - self.new_pos = None - self.movements = 0



Protocolo de agentes:

Agentes reactivos simples - Estos agentes deciden qué realizar de acuerdo a la percepción actual, ignorando el pasado. Se toma una decisión en base al input actual y se realiza una actividad. Por ejemplo: si el robot de enfrente está frenando entonces su posición se elimina en la lista de posibilidades de los otros robots. Esta conexión se denomina regla de condicíon-accion (también llamadas reglas de situación-accion, producciones, o reglas si-entonces).

Estrategia cooperativa para la solución del problema

- Parte Python-Mesa

Para esta parte de la evidencia, primero decidimos anotar sobre los agentes que estarían en nuestra escena al igual que los métodos o atributos que cada uno contendrá. Ciertos agentes como el de la tarima o la caja no tendrían método de *pasos* pero sí atributos necesarios para mantener el conteo y representación correcta para iterar hasta terminar de recoger cada caja al igual que poder representar visualmente un diferente color a base de la cantidad de cajas que contiene.

Originalmente pensábamos en simplemente quitar la caja de nuestra cuadrícula y decirle al *robot* que estaba cargando algo pero “físicamente” no lo hacía. En vez, decidimos agregarle dos métodos para el acto de poner y recogerse a sí mismo de manera que cuando nuestro agente *robot* quiera actuar con la caja o con la tarima; que nuestro *robot* guarde el objeto dentro de una variable y lo lleve consigo hasta llegar a la tarima. Por lo tanto, cuando hacemos contacto con esa tarima se llama el método de poner caja dentro de nuestra clase de *caja* para que se coloque de manera correcta.

Para el funcionamiento de nuestro agente, decidimos por cada movimiento revisar a dónde se quiere mover al igual de que ve a su alrededor para que siempre esté listo de recoger o colocar cajas al igual de no entrar a posiciones que sean redundantes como sobre otro robot, una tarima y no cargar una caja o sobre una caja pero ya

carga con una, de esa manera nos aseguramos que los robots mientras que si se muevan alrededor de la cuadrícula de una manera aleatoria si puedan ser más eficientes al moverse.

- Parte Unity

Para esta segunda parte, desarrollamos diferentes casos de prueba en las colisiones que tendría el objeto, en este caso el robot, con el objetivo de que reaccionara a cada impacto de forma inteligente, cambiando su dirección de acuerdo a la posición en la que se encontraba. Por lo que cada objeto de la escena contiene un box collider que controla la nueva orientación del robot, asimismo cada robot posee una prioridad, ya que si existe un colisión entre 2 objetos de tipo robot, este debe esperar a que uno avance para si proceder. Inicialmente tuvimos la idea de tener 3 objetos vacíos que contengan su propio box collider para poder tener un sistema que detecte anteriormente de algún tipo de colisión y de qué lado vendría. Al final decidimos implementar el primer sistema a causa de tiempo y practicabilidad.