Tecnológico de Monterrey

Evidencia 2

Jessica Odette Guizado Hernández A01643724

Santos Alejandro Arellano Olarte A01643742

Luz Patricia Hernández Ramírez A01637277

Carlos Alberto Mentado Reyes A01276065

Ángel Eduardo Esquivel Vega A01276114

Modelación de sistemas multiagentes con gráficas computacionales (Gpo 101)

Profesores:

Iván Axel Dounce Nava

Luis Raúl Guerrero Aguilar,

Obed Nehemías Muñoz Reynoso,

Carlos Johnnatan Sandoval Arrayga

Viernes 06 de septiembre, 2024

Evidencia 2: Entrega Intermedia

Propuesta Formal del Reto

Descripción del Problema

El objetivo principal del proyecto es desarrollar un sistema multiagente en un entorno de almacén o bodega, donde un dron y varios otros agentes (cámaras y personal de seguridad) trabajan juntos para mantener la seguridad del espacio. Este entorno simulado deberá permitir la interacción entre los diferentes agentes, optimizando la detección y respuesta ante situaciones sospechosas.

El proyecto contempla:

1. **Patrullaje del dron**: El dron despegará desde su estación y realizará un patrullaje rutinario por el almacén, siguiendo una ruta predeterminada y aterrizando de nuevo en su base una vez completado el recorrido.
2. **Detección de amenazas**: Las cámaras fijas ubicadas en diferentes puntos estratégicos del almacén estarán encargadas de detectar movimientos o figuras sospechosas. Si se detecta algo inusual, las cámaras notificarán al dron para que investigue.
3. **Inspección y respuesta**: Una vez notificado, el dron se dirigirá a la zona sospechosa para realizar una inspección más detallada. Si el dron detecta una amenaza, pasará el control al personal de seguridad, quienes decidirán la acción a seguir, como activar una alarma o desactivar una falsa alarma.

Identificación de los Agentes Involucrados:

- **Dron**: Responsable del patrullaje, detección de anomalías y comunicación con el personal de seguridad.
- **Cámaras Fijas**: Encargadas de monitorear el entorno del almacén, detectando movimientos o figuras sospechosas.
- **Personal de Seguridad**: Controla el dron en caso de amenaza y toma decisiones críticas para la seguridad del almacén.

Objetivos del Equipo:

1. **Desarrollar un entorno de simulación 3D altamente eficiente y realista**, donde todos los agentes interactúen de manera coherente y sincrónica.
2. **Integrar y coordinar eficazmente los diferentes agentes involucrados en el proyecto**, asegurando que cada uno cumpla su función dentro del sistema de seguridad del almacén.
3. **Asegurar que el sistema de seguridad del almacén funcione de manera óptima con el dron y el personal de vigilancia**, detectando y respondiendo a amenazas potenciales de manera eficiente.
4. **Implementar un sistema de comunicación robusto y eficiente entre los agentes**, priorizando la rapidez y la exactitud en la transmisión de información crítica.
5. **Documentar cada fase del proyecto**, asegurando que todas las decisiones y cambios sean registrados para futuras referencias y evaluaciones.

Diagramas de Clase

cameraAgent
agentType : int knownObjects : List[str] objects_seen : List[Entity] input_sent : bool
setup(self) : None step(self) : None update(self) : None end(self) : None
sendMessage(receiver: Agent, content: dict) : void send_objects_seen(drone: Agent) : void
see(e: Entity) : void
Capaz de detectar objetos dentro de su rango de visión y registrar los que observa. Puede enviar la lista de objetos detectados a los drones, facilitando la inspección y acción por parte de estos. Además, maneja la configuración inicial de objetos conocidos y gestiona la actualización de su estado y la comunicación en cada paso del ciclo de simulación.
El agente es responsable de detectar y registrar objetos en su área de visión, enviando la lista de objetos detectados a los drones para que puedan actuar en consecuencia. Maneja su estado actual actualizando la lista de objetos conocidos y configurando la comunicación durante la simulación. En cada paso del ciclo de simulación, realiza la detección de nuevos objetos y envía actualizaciones a los drones si es necesario, asegurándose de estar listo y configurado para comenzar sus tareas al inicio del ciclo.
Protocólo de Comunicación "CameraAgent - DronAgent"
Cameras

securityGuardAgent
agentType : int drone_detected_objects : List[object] objects_to_investigate : List[object] message_queue : List[Message] dangerous_objects : List[str] received_from_drone : bool message_received : bool ready_to_reply : bool
setup(self) : None step(self) : None update(self) : None end(self) : None
receive_message(message: Message) -> None send_message(receiver: ap.Agent, content: dict) -> None process_messages() -> None send_objects_to_investigate(drone: ap.Agent) -> None filter_dangerous_objects() -> List[object]
Puede recibir y procesar mensajes de drones, enviar mensajes a los drones sobre objetos que necesitan ser investigados, y filtrar objetos peligrosos para gestionar amenazas. Mantiene una cola de mensajes para una comunicación eficiente y actualiza su estado según la información recibida. Estas capacidades permiten una coordinación efectiva con los drones y una adecuada priorización en la investigación de objetos peligrosos.
El agente es es responsable de recibir información de drones sobre objetos detectados, procesar esta información para identificar objetos peligrosos, y enviar instrucciones a los drones sobre qué objetos investigar. Además, mantiene una cola de mensajes para la comunicación y asegura que los mensajes se procesen correctamente, colaborando con los drones para gestionar amenazas y garantizar la seguridad en el entorno.
Protocólo de Comunicación "SecurityAgent - DronAgent"
SecurityCrew

droneStationAgent
agentType : int ponderation : float
setup(self) : None step(self) : None update(self) : None end(self) : None
El agente está diseñado para ser una estación de drones con la capacidad de proporcionar un punto de partida o destino para los drones. Mantiene una ponderación que podría representar la importancia o prioridad de la estación.
DroneStations

objectAgent
agentType : int object_is : str
setup(self) : None step(self) : None update(self) : None end(self) : None
El agente tiene la capacidad de representar y gestionar un objeto en el entorno simulado.
Objects

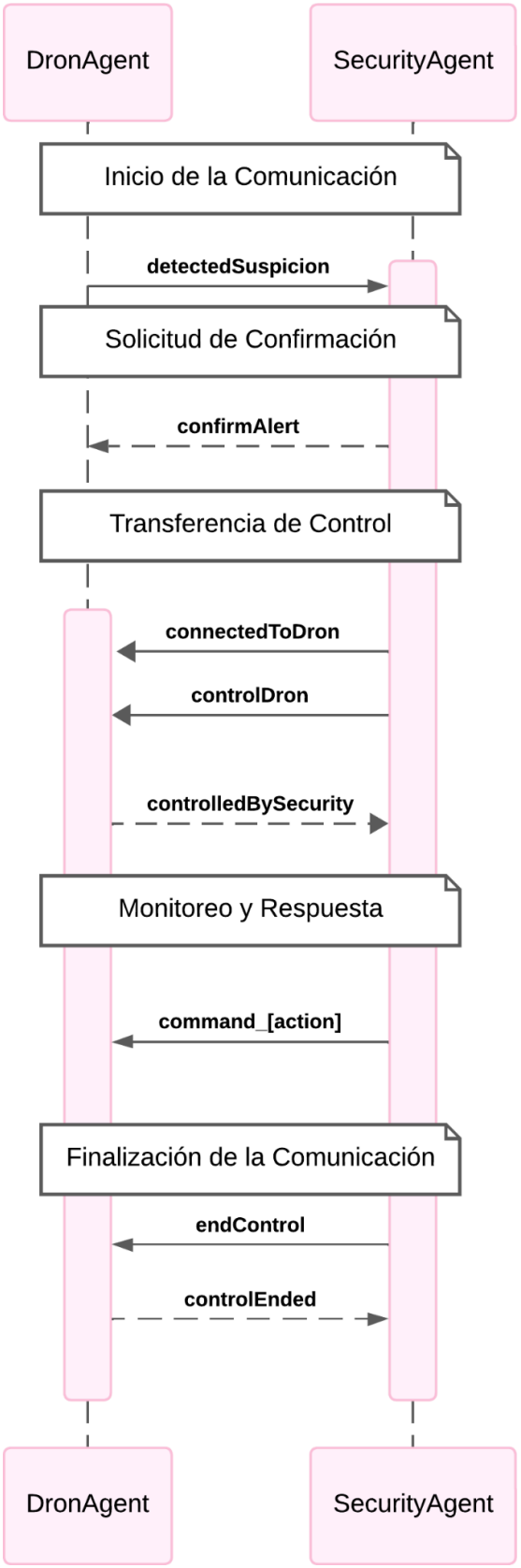
Medición de la Utilidad o Éxito

El éxito del sistema se medirá a través de los siguientes criterios:

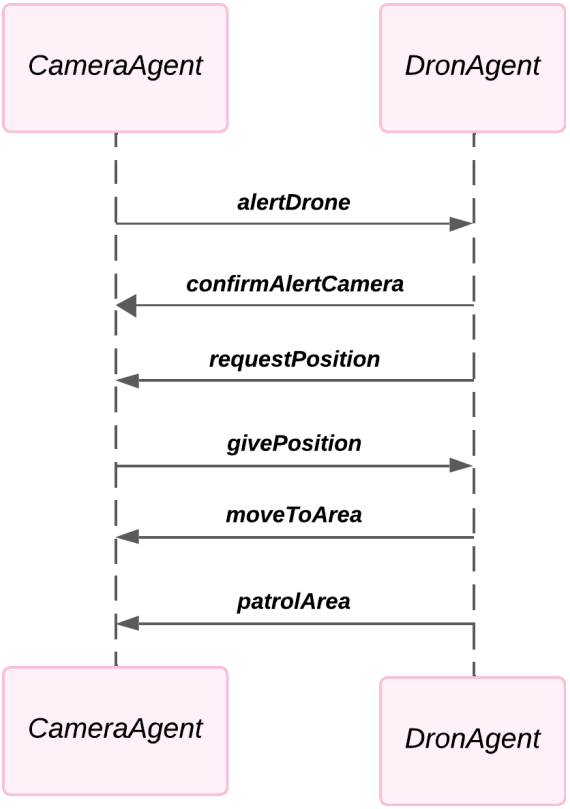
- **Tasa de detección de amenazas**: Proporción de amenazas detectadas y manejadas correctamente por el sistema.
- **Eficiencia del patrullaje**: Tiempo que tarda el dron en realizar su patrullaje completo y en responder a una notificación de amenaza.
- **Interacción entre agentes**: Evaluación de la eficacia y velocidad de la comunicación y coordinación entre el dron, cámaras y personal de seguridad.
- **Tiempo de respuesta**: Velocidad con la que el personal de seguridad puede tomar el control del dron y responder a una amenaza detectada.
- **Precisión de la detección**: Evaluación de la tasa de falsas alarmas generadas por el sistema.

Diagramas de Protocolos de Interacción

Protocólo de Comunicación "SecurityAgent - DronAgent"



Protocólo de Comunicación "CameraAgent - DronAgent"



Propiedades de los agentes:

- **Agente Dron:** Reactivo pues reacciona a los objetos que hay a su alrededor, proactivo, ya que toma decisiones propias y social porque se comunica con las cámaras de seguridad y con el guardia de seguridad
- **Agente Cámara:** Reactivo, ya que reacciona a los objetos del ambiente y social porque se comunica con el dron
- **Agente Guardia de Seguridad:** Reactivo porque reacciona a lo que el dron le dice y social porque se comunica con el dron
- **Agente Objeto:** Ninguna
- **Agente Estación de Dron:** Ninguna

Propiedades del ambiente:

- **No accesible:** No todos los objetos saben lo que pasa en el sistema todo el tiempo
- **Determinista:** Solo hay una posible reacción para cada acción
- **Estático:** El ambiente siempre se mantiene igual, no cambia
- **Discreto:** El ambiente es discreto ya que es una simulación basada en steps
- **Episódico:** Se podría considerar episódico ya que la interacción de los agentes dependerá del protocolo de razonamiento que tenga el dron en un momento particular de la simulación

Análisis individuales

Jessica Odette Guizado Hernández A01643724

Esta actividad me ayudó a entender cómo utilizar librerías avanzadas para trabajar con modelos entrenados. Por otra parte desarrollé una servidor de RESTful API para trabajar transacciones que mantuvieran comunicados los diferentes aplicaciones. Aprendí a consumir un API desde Unity y trabajar con formatos JSON. Está implementando nos facilitó mucho el desarrollo y flujo de información. Serialización y deserialización son procesos fundamentales para el procesamiento de los datos de forma segura y eficiente.

Comprendí cómo implementar trabajar con ObjectMover utilizando múltiples objetos y modificar sus propiedades mediante la implementación de algoritmos.

Al trabajar con flask además de crear un servicio de Rest API de una forma simplificada, práctica e intuitiva, aprendí el uso de los decoradores de funciones que nos ayudan a agregar operaciones ya sea antes o después de lo definido en nuestras funciones. En el caso de flask, se utilizaron para agregar toda la lógica y prefiguración de los Endpoints que desarrollamos.

Santos Alejandro Arellano Olarte A01643742

Seleccionamos el modelo multiagente porque es ideal para simular entornos donde varios actores, como drones, guardias y cámaras, deben trabajar de manera independiente pero con un objetivo común: mejorar la seguridad y organización en un almacén desordenado. Las variables más importantes que consideramos fueron el número de agentes, el tamaño del almacén, la cantidad de objetos y el tiempo máximo de simulación. Estas variables interactúan de forma clave; por ejemplo, si aumentamos los objetos sin ajustar la cantidad de drones o guardias, el sistema puede volverse menos eficiente. Además, el diseño gráfico fue pensado para que la visualización sea clara y podamos identificar la posición y función de cada agente fácilmente, lo cual es fundamental para monitorear su desempeño. Entre las ventajas que encontramos están la flexibilidad del modelo, que puede adaptarse a distintos escenarios, y su escalabilidad. Sin embargo, una desventaja que notamos es que, si no se optimiza bien, el sistema puede sobrecargarse con muchos agentes o pasos de simulación, haciéndolo más lento. Para mejorar esto, podríamos optimizar los algoritmos de movimiento de los agentes y mejorar las reglas de interacción para reducir la redundancia. En cuanto a mi reflexión individual, creo que este proyecto ha sido una experiencia muy enriquecedora porque me permitió no solo profundizar en la simulación multiagente, sino también encontrar

soluciones prácticas para desafíos reales, como la optimización de un almacén. Me di cuenta de la importancia de ajustar las variables de manera adecuada para obtener resultados óptimos y cómo pequeños cambios pueden tener un gran impacto en la simulación.

Luz Patricia Hernández Ramírez A01637277

La parte más difícil del reto fue establecer y llevar a cabo la conexión entre Unity y Python AgentPy, y fue de la que más aprendizajes adquirí. Implementar ontologías, taxonomías y pensar como agentes me dio un mayor entendimiento sobre la situación. En nuestro caso, decidimos usar un modelo práctico a fin de construir un sistema que permitiera la interacción eficiente entre los agentes en un entorno controlado. Optamos por un modelo práctico que se enfocara en la colaboración entre drones, cámaras y guardias de seguridad, utilizando ontologías para definir claramente las relaciones entre los objetos y agentes en el entorno. Este enfoque facilitó la gestión de información y la toma de decisiones de los agentes, mejorando la capacidad del sistema para adaptarse a cambios en el entorno y cumplir sus objetivos de manera eficiente.

Carlos Alberto Mentado Reyes A01276065

Al pensar en el caso de estudio para el proyecto, empecé a pensar cuales serían los pasos que se verían en la simulación, al principio no entendí bien la situación y se empezó la simulación pensando en que las cámaras de vigilancia harían todo el proceso de filtro de objetos.
Después con la situación correctamente definida, se definieron las variables y los agentes:
Las cámaras de seguridad, los objetos en el modelo, el dron, el guardia de seguridad, y el checkpoint del dron, para que la simulación funcione los agentes interactúan entre sí, los agentes con razonamiento y protocolos de comunicación saben sobre la existencia de los agentes objetos.
Las ventajas que tiene el código de la simulación es que está pensada para cualquier tamaño de grid, por lo que no debería tener fallas en su funcionamiento, las desventajas son más aplicables en la presentación de la solución gráfica en Unity. Debida la forma que se manejan las coordenadas y la actualización de estas dentro de la simulación (int) en la solución gráfica se ve como si fuera muy lento y trabado, además, el sistema de visión computacional no está conectado a la simulación.
Este proyecto me ayudo mucho a entender como funcionan los agentes, pues yo me encargué de la parte del código de la simulación, logré cosas que a mi parecer son muy buenas

Ángel Eduardo Esquivel Vega A01276114

Se seleccionó este modelo, ya que permite simular la interacción dinámica entre varios agentes en un ambiente simulado. Como cada agente interactúa de maneras específicas y diferentes, consideramos que este enfoque es el ideal. Las principales variables qué consideramos y definimos fueron los tipos de agentes necesarios, la interacción que tendrían entre ellos y el ambiente donde interactúan. Finalmente, estas variables interactúan en conjunto para generar la simulación. Para el diseño gráfico desarrollamos lluvias de ideas como equipo para llegar al producto final, así mismo identificamos diferentes posibles assets, herramientas y puntos de vista. Esto con el objetivo de seleccionar lo más apropiado para el proyecto. Algunas ventajas que tiene el proyecto es la interacción dinámica, la claridad del comportamiento y representación gráfica atractiva. En cuanto a desventajas destaca el rendimiento, ya que la simulación es un poco lenta y consume muchos recursos, estos sin duda igualmente son puntos que podrían ser mejorados y optimizados. Para hacer esto tendríamos que mejorar la eficacia del código y su interacción con Unity.