Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Guadalajara

Modelación de sistemas multiagentes con gráficas computacionales (Gpo 302)

Evidencia 3. Actividad Integradora

TC2008B

Equipo Warlocks | Integrantes:

Christian Fernando Aguilera Santos	ITC | A01643407
Aarón Hernández Jiménez	ITC | A01642529
Maxime Parienté	ITC | A01764161
Pablo Esteban Reyes	ITC | A01643307
Luis Marco Barriga Baez	ITC | A01643954
Aram Barsegyan	ITC | A01642781

29 nov 2024

Conformación del equipo

Fortalezas y áreas de oportunidad de cada miembro

Para el equipo de trabajo, detallamos los puntos fuertes de cada miembro para la realización del proyecto en los puntos que se presentan a continuación.

- **Aarón**: Posee muy buenos conocimientos en gráficos y en programación en general, se encargará prioritariamente de hacer la parte de Unity y la conexión con Python, según yo no comprendo en su totalidad los agentes y tengo problemas con cálculo.
- **Maxime**: Tiene mucha facilidad con el aspecto matemático (matrices, tensores...) para la parte computacional, y se ocupa de la parte de Python y la teoría del código. Este curso le permitirá adquirir nuevas habilidades, especialmente en lo referente a gráficos.
- **Luis Marco Barriga Baez:** Es muy hábil en el desarrollo con Python, destacando en la programación de agentes y sus comportamientos. Tiene una gran capacidad para resolver problemas algorítmicos, lo que lo convierte en un buen elemento para el equipo del proyecto. Aunque aún puede mejorar en la integración gráfica con herramientas como Unity.
- **Pablo:** Profundo conocimiento de la abstracción de los conceptos relacionados a los sistemas multiagentes. Conciso pero detallado al realizar trabajos para no dejar lugar a la ambigüedad. Carece de habilidades avanzadas para interactuar con los agentes con Unity.
- **Christian:** Capacidad para diseñar e implementar modelos de sistemas multiagentes usando AgentPy, comprendiendo las dinámicas y las interacciones entre agentes.
- **Aram:** Tiene un buen manejo de Unity y entiende bien el motor gráfico, destacando en la integración con Python y protocolos de comunicación entre agentes. Puede mejorar en el funcionamiento de los agentes y el trasfondo matemático relacionado.

De manera general, esperamos aprender sobre cómo se pueden modelar múltiples agentes, sus interacciones y sus acciones para poder modelar sistemas más complejos. El aprendizaje de las bibliotecas AgentPy y del software Unity nos motiva mucho para el futuro.

Objetivos del equipo

- **Desarrollar una simulación funcional:** Crear un sistema multiagente en Unity que integre un dron, cámaras de seguridad y agentes de vigilancia, asegurando una interacción fluida y realista.

- **Implementar agentes inteligentes:** Programar los agentes utilizando Python para que puedan realizar patrullajes autónomos, detectar actividades sospechosas y responder de manera adecuada a las alertas generadas por el sistema.
- **Optimizar la colaboración entre tecnologías:** Integrar las capacidades de Unity y Python de manera eficiente, aprovechando sus fortalezas para lograr una simulación visualmente atractiva y técnicamente robusta.
- **Fomentar el aprendizaje y la colaboración:** Consolidar conocimientos en modelación de sistemas multiagentes, uso de bibliotecas como AgentPy y diseño de entornos virtuales, mientras se fortalecen las habilidades de trabajo en equipo.
- **Evaluar el desempeño del sistema:** Establecer métricas claras de éxito, como tiempo de respuesta del dron, cobertura del área monitoreada y precisión en la detección de amenazas.

Establecer compromisos

- Cumplir con los objetivos del proyecto: Algo que claramente nos vamos a comprometer es desarrollar un sistema de multi agentes que es funcional donde básicamente se entregue un dron, cámaras de seguridad y un agente de vigilancia en un entorno virtual. Queremos que este proyecto sea técnicamente funcional y visualmente atractivo.

- Entregar resultados de calidad: También nos vamos a comprometer muchísimo con todo el tema de la entrega de un sistema integrado bien documentado que pueda cumplir con todo el tema de las especificaciones establecidas. Realizaremos revisiones constantemente para así poder garantizar que todo el trabajo está básicamente completo y también está implementado de manera muy eficiente.

- Respetar los tiempos establecidos: También cada uno de nosotros básicamente nos vamos a comprometer a complementar todo el tema de las tareas asignadas dentro de los plazos que acordamos como equipo y también en el plan de trabajo ya que en caso de imprevistos podremos comunicarnos de cualquier aspecto de manera inmediata para buscar soluciones en equipo.

- Colaborar activamente: Una de las cosas que nos comprometemos como equipo es estar súper activos en el proyecto desde poder dar ideas y también poder ayudar a resolver problemas. Estamos súper comprometidos en apoyar a cualquier persona del equipo que necesite ayudar para garantizar si o si el éxito del proyecto.

- Mantener una comunicación constante y efectiva: Pensaremos algunas herramientas como WhatsApp, Discord e incluso github como equipo para poder mantenernos en contacto e interconectados para así poder responder los mensajes y las dudas que nos puedan surgir en un plazo mucho más rápido.

- Participar en la integración y pruebas del sistema: Todo el equipo se compromete a colaborar en la fase de integración y pruebas finales del sistema para garantizar que todos los componentes funcionen correctamente y que el sistema sea estable.

Descripción del problema

En entornos de seguridad, especialmente en áreas extensas como la Dungeon, la vigilancia efectiva es un desafío constante debido a la cantidad de recursos necesarios para monitorear y responder a incidentes. La supervisión manual por parte del personal de seguridad puede ser ineficiente, ya que existe un margen considerable de error humano y limitaciones en la cobertura. Además, los costos asociados con mantener una vigilancia constante pueden ser elevados.

Este proyecto busca abordar estas problemáticas mediante el desarrollo de un sistema multiagente que simule la interacción entre un dron autónomo, cámaras de seguridad y personal encargado. La meta es explorar cómo estas tecnologías pueden complementarse para optimizar la vigilancia, reducir el margen de error y responder de manera más ágil a posibles amenazas en el ambiente monitoreado.

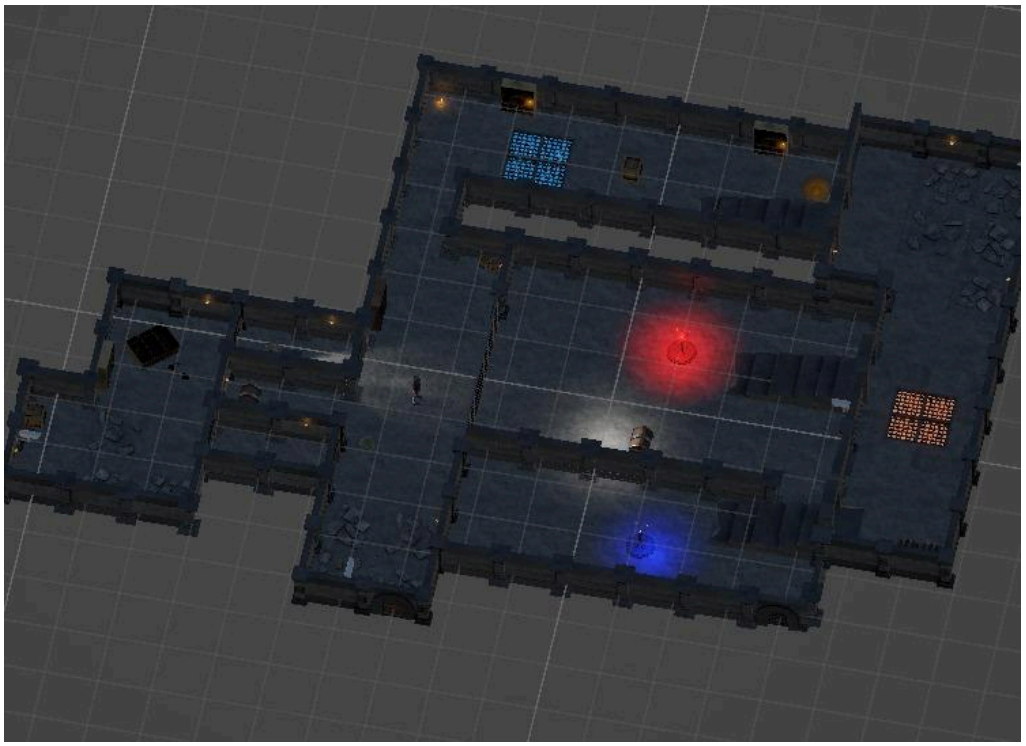
Descripción del proyecto

En este proyecto, desarrollamos una simulación en Unity de un dron y su entorno utilizando un enfoque de sistemas multi-agente. El objetivo es modelar las interacciones entre un dron, cámaras fijas y personal de seguridad en un ambiente virtual. El dron realizará patrullajes autónomos, respondiendo a alertas de las cámaras o detectando actividades sospechosas directamente, inspeccionando las áreas indicadas. En caso de amenaza, el dron colaborará con el personal de seguridad, quien tomará el control y decidirá si activar una alarma. Python será empleado para la programación de los agentes y sus comportamientos, mientras que Unity se encargará de la simulación gráfica y la integración de visión computacional.

- **Ambiente virtual y agente de seguridad** : El entorno virtual elegido es una Dungeon oscura, custodiada por un esqueleto. Las texturas con mapas UV y materiales se han descargado de Internet.

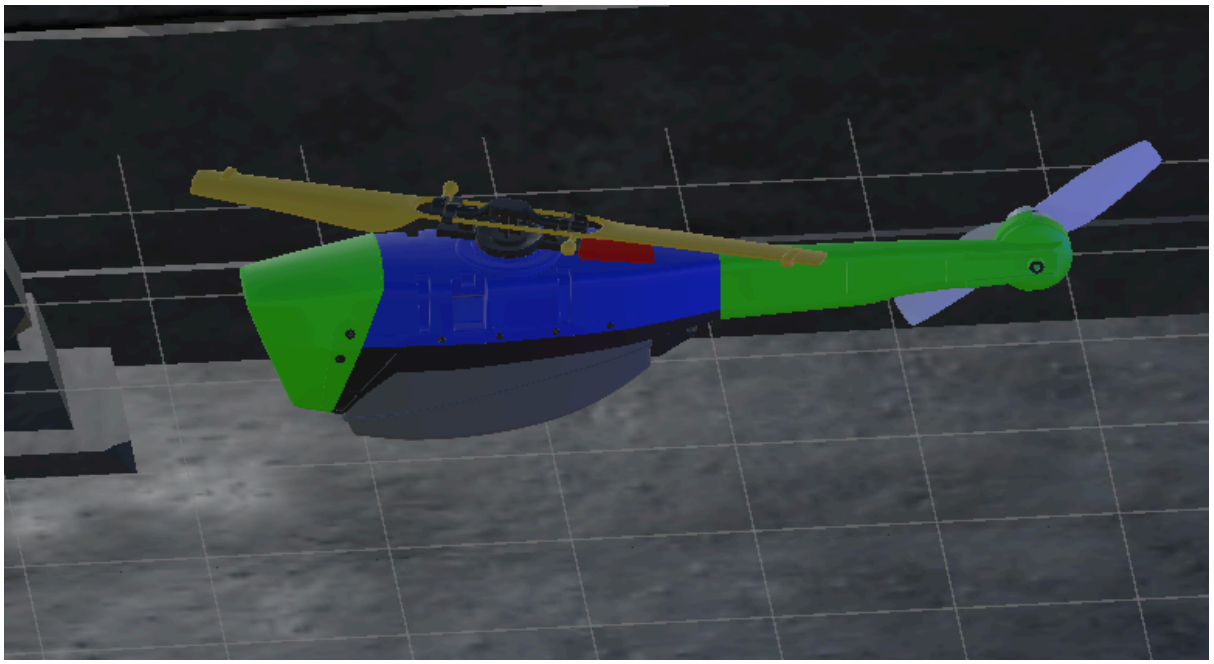
Agentes y Ambiente

Ambiente virtual



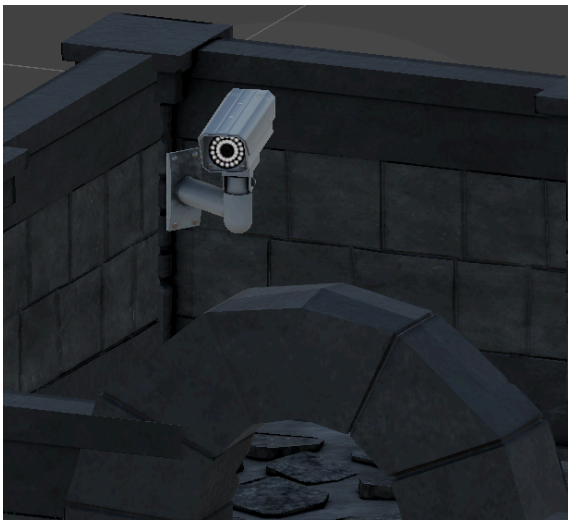
Agente (Seguridad)

Dron (seguridad)



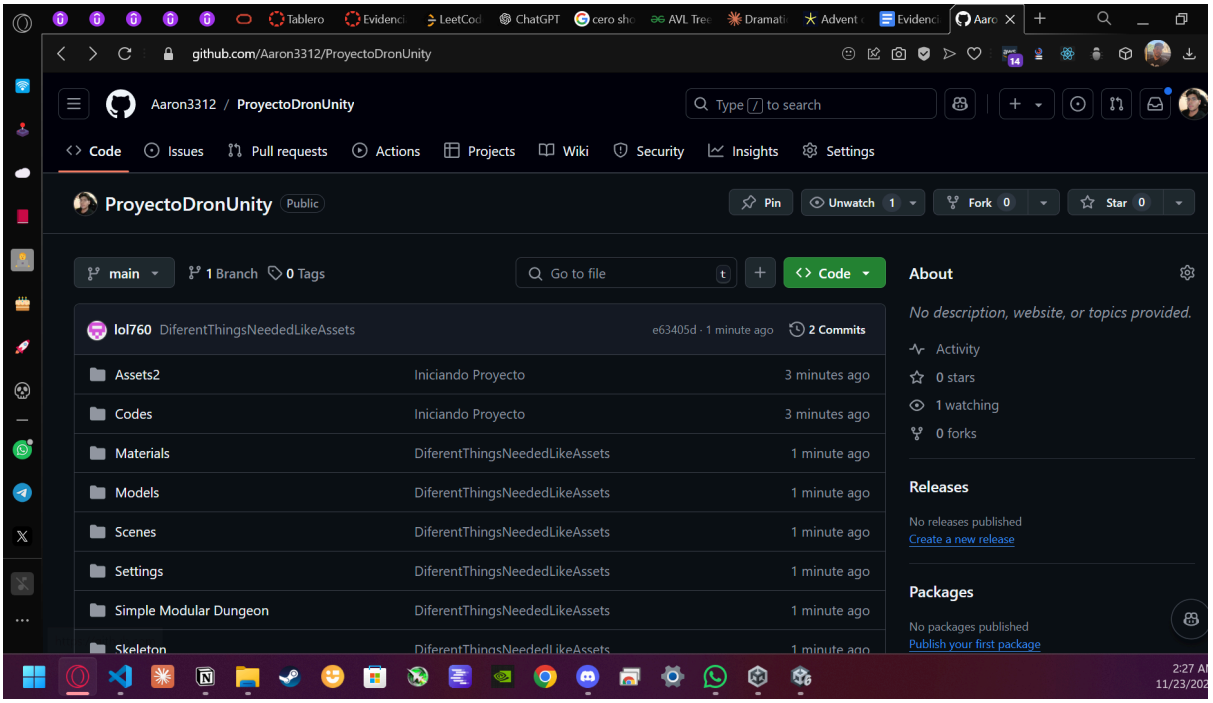
Ladrón

Camaras de seguridad



Herramientas de Trabajo Colaborativo

https://github.com/Aaron3312/ProyectoDronUnity.git



Identificación de los agentes

Agentes Involucrados y Propiedades

Dron:

- Propósito: Realiza patrullaje aéreo autónomo y responde a alertas de las cámaras, se puede desplegar y guardar a la voluntad del guardia.
- Capacidades: Despega y aterriza de forma autónoma, detecta figuras sospechosas, se desplaza al lugar de la alerta, y transmite video en tiempo real que se utiliza para detectar a cualquier intruso.
- Propiedades:
    - Dinámico: Su entorno cambia constantemente mientras patrulla.
    - Continuo: Se mueve en un espacio tridimensional continuo.
    - Accesible: Tiene acceso completo a su estado y al del entorno a través de sus sensores.
    - Socialización: Se comunica con las cámaras y el personal de seguridad.
    - Proactividad: Toma la iniciativa de investigar áreas sospechosas y responder a alertas.

Cámaras Fijas:

- Propósito: Monitorear constantemente el área asignada y detectar figuras sospechosas.
- Capacidades: Identificación de figuras y movimientos inusuales, envío de alertas al dron y al sistema central.
- Propiedades:
    - Estático: Permanecen en una posición fija.
    - Discreto: Capturan imágenes a intervalos regulares.
    - Accesible: Tienen acceso completo a su campo de visión.
    - Socialización: Comparten información con otras cámaras y el dron.
    - Proactividad: Inician alertas cuando detectan actividad sospechosa.

Personal de Seguridad:

- Propósito: Supervisar las alertas generadas por los agentes autónomos y tomar control de la visión del dron para evaluar si el encuentro de un intruso es mayor al 90% de confianza.
- Capacidades: Evaluar la situación a través de la cámara del dron, clasificar el evento como amenaza o falsa alarma, y emitir la señal de alarma correspondiente.
- Propiedades:
    - Dinámico: Responde a situaciones cambiantes en tiempo real.
    - Continuo: Opera en un entorno físico continuo.
    - No accesible: No tiene acceso directo a toda la información del sistema, depende de los reportes de otros agentes.
    - Socialización: Interactúa con el dron y recibe información de las cámaras.
    - Proactividad: Toma decisiones basadas en la información recibida y puede iniciar acciones como el despliegue del dron o la activación de alarmas.

Modelo seleccionado:

Para este proyecto, se eligió un modelo multiagente basado en la interacción autónoma y la coordinación supervisada. El modelo multiagente permite a cada componente del sistema (dron, cámaras y personal de seguridad) actuar como un agente independiente con objetivos específicos, pero también colaborar con otros agentes en función de las necesidades emergentes del entorno.

Este enfoque facilita la división del trabajo y permite que cada agente se especialice en tareas particulares, como la patrulla, la detección de movimientos sospechosos y la respuesta manual por parte del personal de seguridad. La comunicación entre los agentes es esencial para garantizar

que el sistema funcione de forma óptima, especialmente en situaciones donde la detección automática del dron o de las cámaras pueda requerir intervención humana.

Diagrama de clase de los agentes

Métricas de utilidad/éxito

Para evaluar el éxito y la eficacia del sistema multiagente, se establecen las siguientes métricas:

Dron

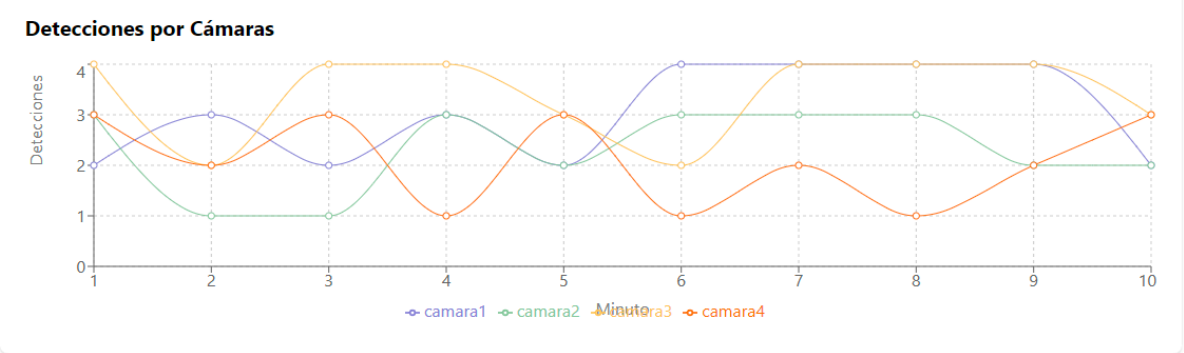
a. Tasa de detección exitosas de figuras sospechosas
b. Tiempo de simulación

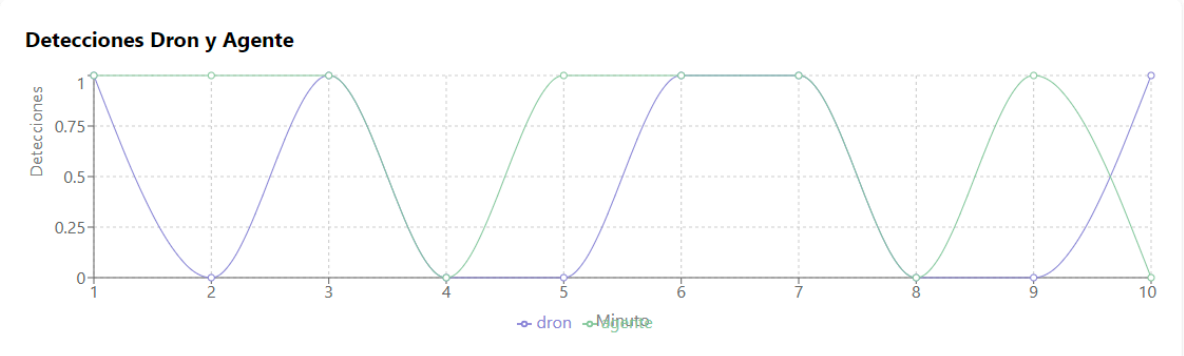
Cámaras Fijas

a. Tasa de detección exitosas de figuras sospechosas
b. Tiempo de simulación

Rendimiento del personal de seguridad

a. Tasa de detección exitosas de figuras sospechosas
b. Tiempo de simulación

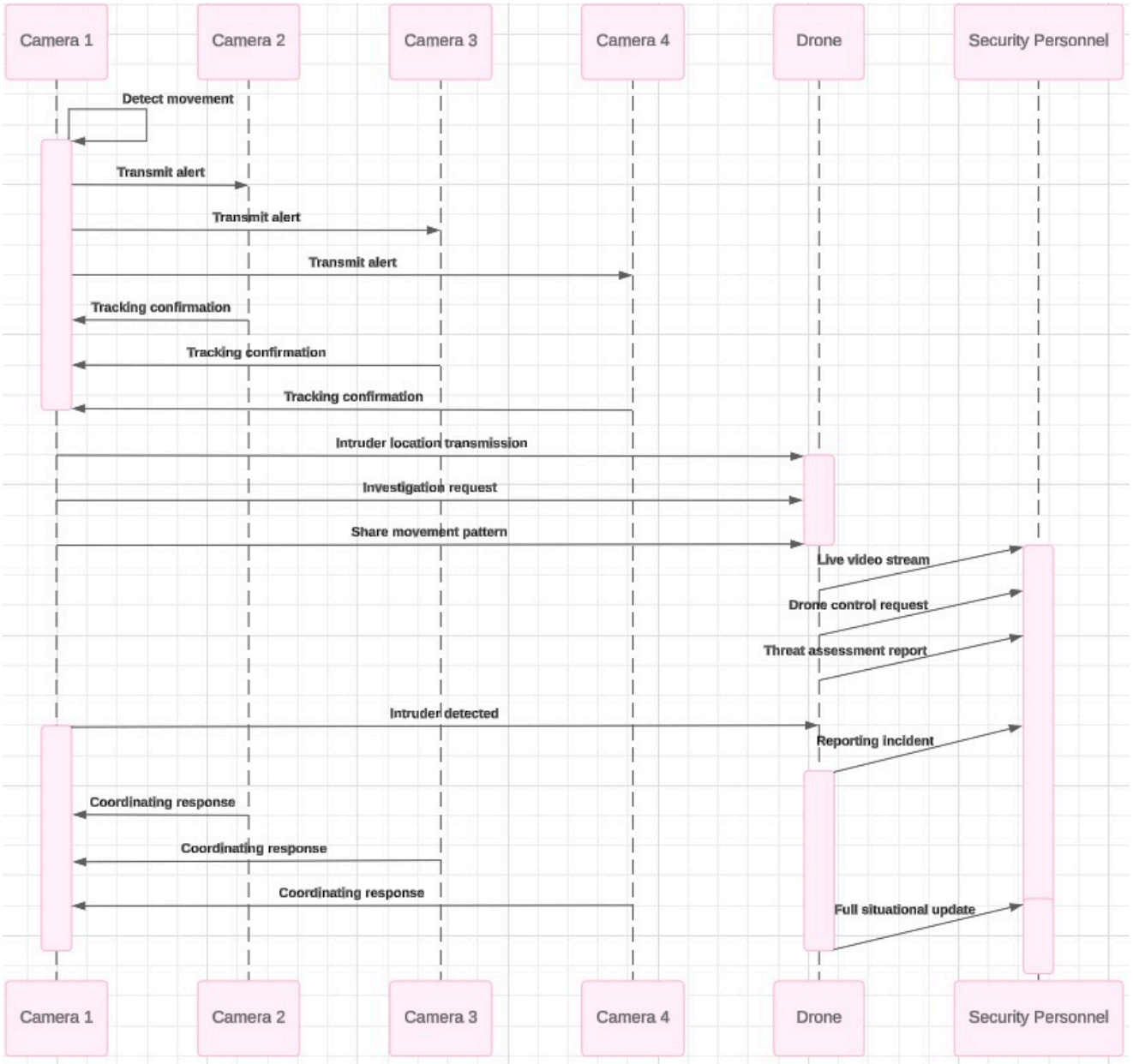




Utilidad Final (Detecciones/Tiempo)

Cámaras	Dron	Agente
2.70	0.50	0.70

Diagrama de secuencia de protocolos de interacción

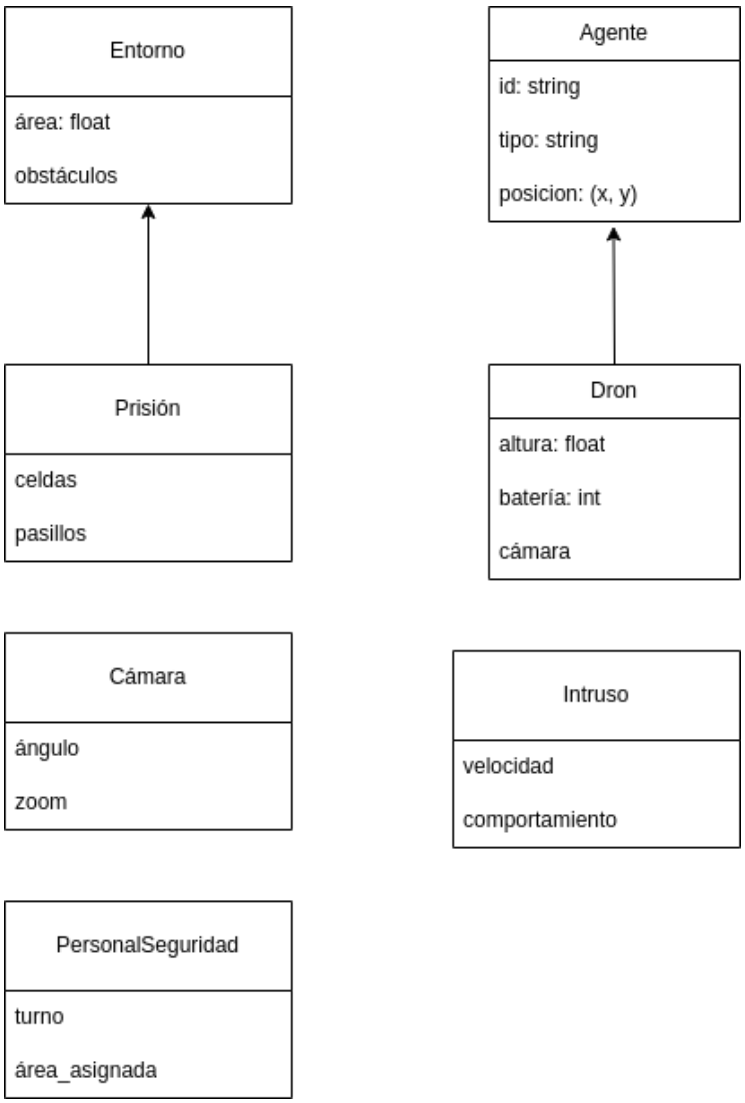


Explicación del diagrama de secuencia

Los diagramas de secuencia ilustran las interacciones del protocolo cooperativo entre los agentes en la simulación.
Las instancias de **FixedCameraAgent** actúan como agentes benevolentes, compartiendo información de intrusos detectados con otras cámaras y **DroneAgent** para garantizar un seguimiento perfecto. Cuando un **FixedCameraAgent** detecta actividad sospechosa, transmite una alerta a sus pares y notifica al **DroneAgent**, que ajusta su patrulla para investigar la última posición conocida del intruso.

DroneAgent transmite video en tiempo real a **SecurityPersonnelAgent**, quien monitorea la transmisión, evalúa la situación y determina si se emite una alarma o clasifica el evento como una falsa alarma.
SecurityPersonnelAgent también puede anular el comportamiento autónomo del dron para control manual. Los diagramas representan un sistema totalmente colaborativo donde los agentes se comunican de manera eficiente, lo que garantiza un seguimiento preciso de los intrusos y una respuesta eficaz a las amenazas.

Diagrama de clases simple de la ontología utilizada



Explicación del diagrama de clases simple de la ontología utilizada

El diagrama de clases que describe la ontología de nuestro sistema de seguridad carcelaria muestra cómo organizamos los diferentes elementos del proyecto. Básicamente, tenemos una clase principal llamada Entorno que representa todo el espacio donde ocurre la acción, y de ahí sale una clase más específica que es la Dungeon. Luego está la clase Agente, que es como el molde para todos los agentes del sistema, ya sea el dron, las cámaras o el personal de seguridad. Cada uno de estos tiene sus propias características especiales. Por ejemplo, el dron tiene cosas como altura y batería, mientras que las cámaras tienen ángulo y zoom. También incluimos una clase para el Intruso, que es lo que nuestro sistema intenta detectar.
Lo interesante de esta estructura es que nos permite representar de manera sencilla cómo interactúan todas las partes del sistema. Podemos ver claramente cómo el

dron, las cámaras y el personal de seguridad operan dentro de la Dungeon, y cómo cada uno tiene su papel en la detección de intrusos. Además, esta forma de organizar las cosas hace que sea más fácil programar los comportamientos de cada agente y asegurarnos de que todo funcione bien junto. En resumen, este diagrama nos ayuda a tener una visión clara de cómo funciona todo nuestro sistema de vigilancia y cómo se relacionan sus diferentes partes.

Plan de trabajo

Actividades pendientes	Tiempo para realizar la actividad (días)	Responsable de
Configuración del ambiente de desarrollo (Unity + Python)	2	Aarón y Chris
Diseño del ambiente virtual de la Dungeon	3	Luis y Maxime
Establecimiento de la arquitectura de comunicación Unity-Python	3	Pablo, Aram y Aaron
Implementación del agente dron y sus comportamientos básicos	4	Aaron y Maxime
Desarrollo del sistema de cámaras de seguridad	4	Aaron y Pablo
Programación del agente de seguridad (esqueleto)	2	Aaron y Chris
Integración de detección de intrusos	3	Equipo completo
Implementación de protocolos de comunicación entre agentes	3	Maxime, Aram y Aaron
Desarrollo del sistema de alertas y respuestas	2	Aaron y Chris
Integración de la visión computacional	2	Pablo y Aaron
Pruebas de integración del sistema completo	3	Equipo completo

Optimización de rendimiento	2	Aaron y Maxime
Documentación final y preparación de la presentación	2	Equipo completo

Justificación de Propiedades Finales

Las propiedades finales del sistema multiagente se justifican por los siguientes aspectos:

1. Arquitectura de los Agentes:
    - El dron requiere autonomía para el patrullaje y capacidad de respuesta a alertas, justificando sus propiedades de vuelo autónomo y detección de figuras sospechosas.
    - Las cámaras fijas necesitan mantener una vigilancia constante y comunicación efectiva, lo que justifica sus capacidades de identificación y envío de alertas.
    - El personal de seguridad (representado por el esqueleto) debe mantener supervisión y control final, justificando sus propiedades de toma de decisiones y control manual del dron.
2. Sistema de Detección:
    - El umbral de confianza del 80% para observación y 90% para alarma se justifica por la necesidad de equilibrar entre sensibilidad en la detección y minimización de falsas alarmas.
    - La transmisión de video en tiempo real del dron al personal de seguridad se justifica por la necesidad de verificación humana en situaciones críticas.
3. Comunicación entre Agentes:
    - La estructura de comunicación permite que las cámaras fijas compartan información de intrusos con otras cámaras y el dron, justificando el protocolo cooperativo implementado.
    - El sistema de alertas jerárquico (cámaras → dron → personal) se justifica por la necesidad de una respuesta escalonada y verificada ante posibles amenazas.
4. Ambiente Virtual:
    - La elección de una Dungeon como entorno se justifica por ser un espacio que requiere vigilancia constante y tiene áreas críticas definidas.
    - El diseño permite una clara visualización de las interacciones entre agentes y facilita la detección de actividades sospechosas.

5. Integración Tecnológica:
    - La combinación de Unity para la simulación visual y Python para la lógica de los agentes se justifica por la necesidad de un entorno realista con comportamientos complejos.
    - La arquitectura modular permite la escalabilidad y mantenimiento del sistema, justificando la separación de responsabilidades entre los diferentes componentes.

Aprendizaje adquirido

Sistemas Multiagente

- Diseño e implementación de agentes autónomos
- Protocolos de comunicación entre agentes
- Coordinación de comportamientos complejos

Herramientas y Tecnologías

- Dominio de Unity para simulaciones 3D
- Programación avanzada en Python
- Uso de la biblioteca AgentPy
- Integración de diferentes tecnologías en un sistema cohesivo

Habilidades blandas

- Trabajo colaborativo en equipos multidisciplinarios
- Gestión efectiva del tiempo y recursos
- Comunicación técnica y documentación
- Resolución de problemas complejos

Análisis individual - Aaron

Justificación de selección del modelo multiagentes

La elección del modelo multiagentes para este proyecto de seguridad carcelaria se justifica por la necesidad de coordinar múltiples entidades autónomas (dron, cámaras, personal) que deben trabajar en conjunto. El documento muestra que el sistema requiere una comunicación constante y toma de decisiones distribuida, donde cada agente tiene roles específicos pero interdependientes. El modelo multiagentes permite manejar esta complejidad de manera modular y escalable.

Justificación de diseño

Escogimos este diseño puramente por razones que ya teníamos los modelos descargados y trabajar con ellos resultó bastante sencillo por estas razones los mantuvimos sin cambios, si bien se nos presentó la idea de una bodega amplia o un campus la verdad fue bastante mas complicado cambiar el modelo y argumentar porque un esqueleto estaría en una bodega o un campus.

Analizar variables de decisión

Las principales variables que afectan la toma de decisiones en el sistema son:

- Nivel de confianza en la detección de intrusos (umbral del 80% para observación y 90% para alarma)
- Tiempo de respuesta del dron ante alertas
- Cobertura del área de vigilancia
- Priorización de zonas críticas
- Estado actual del sistema (normal/alerta)
- Condiciones ambientales que afectan al dron

Ventajas y desventajas de la implementación

Ventajas:

- Integración efectiva entre Unity y Python para combinar visualización 3D con lógica de agentes
- Sistema modular que permite añadir o modificar agentes fácilmente
- Automatización de tareas rutinarias de vigilancia
- Reducción del error humano mediante sistemas de detección automática
- Capacidad de supervisión humana en decisiones críticas

Desventajas:

- Complejidad en la coordinación entre múltiples tecnologías

- Necesidad de mantener sincronización entre agentes
- Dependencia de la calidad de la detección visual
- Posibles retrasos en la comunicación entre agentes
- Curva de aprendizaje pronunciada para el desarrollo

Análisis de la solución desarrollada.

La solución implementada combina efectivamente diferentes tecnologías (Unity, Python, AgentPy) para crear un sistema de vigilancia integrado. La arquitectura permite que cada agente opere de manera autónoma mientras mantiene la capacidad de coordinación con otros agentes. El uso de umbrales de confianza en la detección y la supervisión humana final proporciona un balance entre automatización y control humano.

Reflexión sobre el proceso de aprendizaje: El documento muestra que el proyecto ha permitido desarrollar habilidades en:

- Programación avanzada en Unity y Python
- Diseño e implementación de sistemas multiagentes
- Integración de diferentes tecnologías
- Trabajo en equipo y gestión de proyectos
- Resolución de problemas complejos en tiempo real

Reflexión sobre mi proceso de aprendizaje.

En primer lugar, profundicé considerablemente mis conocimientos en Unity y su integración con Python. Aunque ya tenía experiencia previa en gráficos y programación anteriormente también había hecho un juego en Unity y había trabajado con yolov5 en un proyecto, el desafío de conectar ambas tecnologías me permitió comprender mejor las complejidades de la comunicación entre diferentes sistemas y la importancia de una arquitectura bien diseñada, en general es curioso, seria muchísimo mas rapido haber tenido una arquitectura ya en mente y no empezar a ver como funciona esto o aquello.

Un aspecto particularmente desafiante fue el desarrollo de los comportamientos de los agentes en el entorno virtual. Aprendí a implementar patrones de movimiento complejos con nav mesh para el dron y a sincronizar sus acciones con las cámaras de seguridad con diferentes conexiones a diferentes códigos de python lo cual en sí mismo es interesante, anteriormente había trabajado con diferentes proyectos que requieren esto pero nunca lo había aplicado correctamente, lo que mejoró mi comprensión de los sistemas de coordinación multiagente, y los sistemas de conexión por sockets y protocolos como el UDP.

También desarrollé una mejor apreciación por la importancia del trabajo en equipo. Como responsable principal de la parte de Unity y la conexión con Python al igual que responsable principal del proyecto, tuve que colaborar con mis compañeros, especialmente en la integración de los diferentes componentes del sistema. Esto me ayudó a entender muchas cosas del trabajo en equipo, y siendo sincero siento que no se trabajar en equipo, y este bloque no ayudó particularmente con ese aspecto.

Una área de crecimiento importante fue mi comprensión de los agentes inteligentes. Aunque inicialmente me sentía menos seguro en este aspecto jajaj creía que el algoritmo de descendencia genética era en sí los agentes jajaj, el proyecto me obligó a profundizar en los conceptos teóricos y su aplicación práctica, lo que resultó en una mejor comprensión de los sistemas multiagentes en general.

Finalmente, aprendí la importancia de la planificación y la documentación adecuada en proyectos complejos que si bien tenía bases fuertes ahora se como deben quedar repositorios bien hechos en github. La necesidad de coordinar múltiples componentes y tecnologías me enseñó valiosas lecciones sobre la gestión de proyectos de software.

Propuestas de mejora

- ☐ Implementar un sistema de aprendizaje automático para mejorar la precisión de detección
- ☑ ~~Desarrollar un sistema de redundancia para casos de fallo en alguno de los agentes~~
- ☑ ~~Añadir más métricas de rendimiento para evaluar la eficacia del sistema hecho y en tiempo real~~
- ☑ ~~Optimizar los protocolos de comunicación para reducir latencia~~
- ☐ Implementar un sistema de registro y análisis de incidentes para mejorar el sistema con el tiempo

Análisis individual - Aram

Modelo multiagente seleccionado

El modelo multiagente elegido integra comunicación constante entre un dron, cámaras fijas y personal de seguridad. Este enfoque fue seleccionado por su capacidad de combinar agentes autónomos con intervención manual, creando un sistema eficiente y adaptable a diferentes escenarios. Durante el desarrollo, comprendí que trabajar con sistemas multiagente no solo implica programar tareas específicas, sino también entender cómo los agentes toman decisiones de manera autónoma, algo que al inicio se me dificultó, pero con el tiempo logré entender mejor.

Variables consideradas:

- Área de vigilancia efectiva.
- Latencia en la comunicación entre agentes.
- Tasa de detección de amenazas.

Interacción de las variables:
La interacción fluida entre el dron y las cámaras asegura que las alertas lleguen al personal de seguridad, minimizando falsos positivos y mejorando la precisión. Este proceso, al visualizarlo en Unity, mostró cómo las conexiones entre agentes y sus decisiones impactan directamente en los resultados del sistema. Aunque fue desafiante al principio, especialmente al integrar múltiples agentes y entender cómo operaban, con práctica y trabajo en equipo, estas interacciones se volvieron más claras.

Diseño gráfico seleccionado

Se utilizó Unity por su flexibilidad para representar ambientes realistas y su compatibilidad con Python, permitiendo controlar agentes externos. Este motor gráfico me resultó familiar, ya que tenía un conocimiento moderado sobre su funcionamiento y siempre me ha entusiasmado el diseño de videojuegos, algo que podría relacionarse con mi futuro profesional. Sin embargo, enfrentarme a la conexión entre Unity y Python fue un gran reto, ya que migrar código entre plataformas con diferentes características es un proceso complejo que nunca había hecho antes.

Ventajas:

- Alta personalización del entorno.
- Representación clara de zonas monitoreadas.

Desventajas:

- Complejidad técnica para nuevos usuarios.
- Tiempo adicional para ajustar y renderizar escenas.

Modificaciones:

- Simplificar modelos gráficos en áreas no críticas.
- Automatizar procesos de pruebas para reducir tiempos.

Trabajar con Unity fue una experiencia gratificante pero también demandante. La creación de escenas en 3D y la asignación de atributos a objetos específicos resultaron tareas a veces tediosas, pero el resultado final siempre fue satisfactorio. Visualizar las simulaciones y observar cómo los agentes tomaban decisiones basadas en sus protocolos me permitió apreciar la importancia de la inteligencia artificial en la tecnología moderna.

Este proyecto no solo me ayudó a reforzar mis habilidades técnicas, sino que también me permitió descubrir lo fascinante que es el mundo de los agentes y cómo todo en inteligencia artificial gira en torno a ellos. Fue un aprendizaje invaluable, enriquecido por el trabajo en equipo, donde cada desafío nos llevó a soluciones que contribuyeron al éxito del resultado final.

Análisis individual - Luis Marco

1. Modelo multiagente seleccionado

Básicamente se seleccionó un modelo basado en agentes que están especializados en tener capacidades autónomas, para así ser coordinados por protocolos de comunicación eficientes.

Algunas de las variables que fueron consideradas:

Capacidad del dron para patrullar.

Frecuencia de alertas generadas por cámaras.

Tiempo promedio para resolver amenazas.

Interacción de las variables:
Se buscaba al inicio un sistema que estuviera super bien sincronizado que asegurase que las alertas se procesen rápidamente, para así poder evaluar de la mejor manera los riesgos.

2. Diseño gráfico seleccionado

Para ser sinceros, Unity fue seleccionado por los profesores pero el diseño en Unity facilita muchísimo como se visualiza y cómo interactúan los agentes en tiempo real, mejorando así la comprensión del sistema.

Desventajas:

- Dificultades para integrar funcionalidades avanzadas sin experiencia previa.
- Uso intensivo de tiempo en ajustes gráficos.

Ventajas:

- Ambiente visual impactante para la evaluación del proyecto.
- Soporte para tecnologías como mapas UV.

Modificaciones:

- Algunas de las modificaciones que yo realizaría sería introducir opciones gráficas adaptativas para equipos menos potentes y que cualquiera lo pudiera correr. Además creo que utilizar herramientas de monitoreo para optimizar la latencia seria de vital importancia.

Reflexión sobre mi proceso de aprendizaje.

Básicamente durante este proceso aprendí y reforce muchísimo todo el tema de mis habilidades en programación de agentes y también en coordinación de sistemas complejos. La verdad es que al inicio la integración con Python para mí fue

complicadísima pero finalmente logré mejorar el manejo, a futuro debería enfocarme en también en mejorar la integración gráfica con Unity.

Análisis individual - Pablo

Modelo multiagente seleccionado

El sistema multiagente se seleccionó para dividir eficientemente las tareas de monitoreo, detección y respuesta. Cada agente opera de manera independiente pero colaborativa.

Variables consideradas:
Consumo de recursos computacionales, velocidad de respuesta del dron y precisión en la detección visual.

Interacción de las variables:
El balance entre la detección automática y la intervención manual permite adaptarse a diferentes escenarios.

Diseño gráfico seleccionado
Unity fue elegido para proporcionar un entorno inmersivo y comprensible, crucial para la presentación y análisis del proyecto.

Ventajas: Capacidades avanzadas para simular movimientos de los agentes y las herramientas para pruebas iterativas rápidas.

Desventajas: Dependencia de hardware con altos requisitos y las limitaciones para simular agentes en gran escala.

Modificaciones: Explorar bibliotecas alternativas para reducir la carga gráfica y también incluir métricas de rendimiento más detalladas en el sistema.

Reflexión sobre mi proceso de aprendizaje.

Al inicio tenía un poco de complicación con el comprendimiento de los multiagentes pero al final de este curso me he dado cuenta de que he aumentado mi capacidad para comprenderlo. Logré avances super grandes a medida que seguíamos en el curso de cuando hablamos de la programación, solamente quedaría resaltar que debo trabajar mucho en mejorar la interacción con herramientas gráficas, ya que creo que este es un pequeño punto débil en mi contra.

Análisis individual - Máxime

Justificación de la selección del modelo multiagente

La solución implementada integra de manera eficiente tecnologías diversas como los lenguages Unity, Python y el framework AgentPy para construir un sistema de vigilancia completamente integrado. La arquitectura diseñada permite que los agentes operen de forma autónoma, manteniendo a la vez la capacidad de coordinarse entre ellos. Además, la implementación de umbrales de confianza en la detección, complementada con una supervisión humana en los casos críticos, asegura un equilibrio adecuado entre la automatización y el control humano. El diseño seleccionado fue elegido principalmente por la practicidad de trabajar con modelos preexistentes, lo que facilitó su integración y manejo. Aunque inicialmente se propusieron alternativas como un entorno de almacén o campus, cambiar el modelo base resultó complejo, especialmente al justificar la presencia de un esqueleto en dichos contextos. Por ello, se optó por mantener el diseño original sin modificaciones significativas. Además, cada agente opera de forma autónoma, manteniendo la capacidad de coordinación con otros agentes dentro del sistema. El uso de umbrales de confianza en la detección, combinado con la supervisión humana en momentos clave, equilibra la automatización con el control humano. Esto asegura un sistema confiable y adaptable a diferentes escenarios.

Análisis de las variables de decisión

Las variables clave que afectan la toma de decisiones en el sistema incluyen:

- **Nivel de confianza en la detección de intrusos**: Umbral mínimo del 80% para observación y 90% para emitir una alarma.
- **Tiempo de respuesta del dron**: Capacidad del dron para actuar rápidamente ante alertas.
- **Cobertura de vigilancia**: Supervisión efectiva de toda el área designada.
- **Priorización de zonas críticas**: Atención inmediata a áreas con mayor riesgo.
- **Estado del sistema**: Modos de operación (normal o en alerta).
- **Condiciones ambientales**: Factores como viento que pueden afectar el rendimiento del dron.

Ventajas y desventajas de la implementación

Ventajas:

- Integración eficiente entre Unity y Python, combinando visualización 3D con lógica basada en agentes.

- Diseño modular que permite añadir o modificar agentes fácilmente.
- Automatización de tareas rutinarias de vigilancia, reduciendo errores humanos.
- Supervisión humana en decisiones críticas, garantizando un control adecuado.
- Escalabilidad del sistema para futuros requerimientos.

Desventajas:

- Alta complejidad en la coordinación de múltiples tecnologías.
- Dependencia de una sincronización precisa entre los agentes.
- Sensibilidad a la calidad de las detecciones visuales.
- Posibles retrasos en la comunicación entre agentes.
- Curva de aprendizaje considerable para el desarrollo e implementación.
- No sabemos si lo que funciona bien en virtual va a funcionar en el mundo real (especificamente con drones)

Para reducir los puntos negativos, podríamos trabajar con un modelo que se acerque más a la realidad (y por lo tanto sea más complejo), y utilizar una plataforma como ROS, que permite la conexión entre el mundo real (dron) y la simulación (Sim2Real). Esta mejora permitiría implementar un proceso incremental para verificar, a intervalos regulares, que la simulación refleja las capacidades del dron en el mundo real.

Reflexión sobre mi proceso de aprendizaje.

Para empezar, este proyecto me permitió descubrir la parte gráfica de los proyectos (en este caso, Unity) y mejorar drásticamente mi manera de programar y visualizar los agentes. Ya he tenido la oportunidad de realizar una estancia de investigación en visión por computadora, donde trabajé con herramientas como SAM2 y Yolov8. Además, ya había estudiado temas relacionados con el aprendizaje por refuerzo, que a menudo se basa en el enfoque MAS (Sistemas Multi-Agentes). Así, en cuanto a la parte teórica y el desarrollo del código de los agentes, pude profundizar en temas que ya conocía. Sin embargo, fue muy interesante porque el proyecto se basa en las demandas reales de una empresa, lo que exige más que un simple conocimiento teórico, requiriendo una implementación mucho más completa.

Además, disfruté mucho el hecho de trabajar en un equipo donde cada individuo aportó algo único al proyecto. En particular, me gustó mucho cuando mis compañeros me ayudaron a entender Unity y la conexión con Python y C#. A su vez, me encantó poder ayudar a mis compañeros en la parte matemática relacionada con las operaciones matriciales para los gráficos.

Finalmente, este proyecto fue muy interesante de llevar a cabo porque nos exigió mucho trabajo con plazos muy cortos, lo que nos obligó a colaborar, realizar un trabajo eficiente y comunicarnos constantemente sobre las diferentes problemáticas.

Análisis individual - Chris

Reflexión sobre mi proceso de aprendizaje.

La elección del modelo multiagentes para este proyecto de seguridad tiene sustento por la necesidad de coordinar múltiples entidades autónomas que deben trabajar en conjunto. El sistema requiere una comunicación constante y toma de decisiones distribuida entre el dron, las cámaras de seguridad y el personal, donde cada agente tiene roles específicos pero interdependientes. Este modelo permite manejar la complejidad del sistema de manera modular, facilitando la integración de diferentes componentes y comportamientos.

Durante este proyecto, he profundizado mis conocimientos en el diseño e implementación de sistemas multiagentes. He mejorado mis habilidades en la programación con AgentPy y en la integración de diferentes tecnologías como Unity y Python. El trabajo en equipo ha sido fundamental, permitiéndome desarrollar habilidades de colaboración y comunicación técnica.
La implementación de los agentes y sus comportamientos me ha permitido comprender la importancia de la coordinación entre diferentes componentes. He aprendido a equilibrar la autonomía de los agentes con la necesidad de supervisión humana en situaciones críticas.

Anexos

- Enlace al repositorio de GitHub.
  https://github.com/Aaron3312/ProyectoDronUnity.git