

**Instituto Tecnológico y de Estudios Superiores de Monterrey**  
**Campus Guadalajara**



**Modelación de sistemas multiagentes con gráficas  
computacionales (Gpo 302)**

**Evidencia 1. 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 Marco Barriga Baez ITC | A01643954

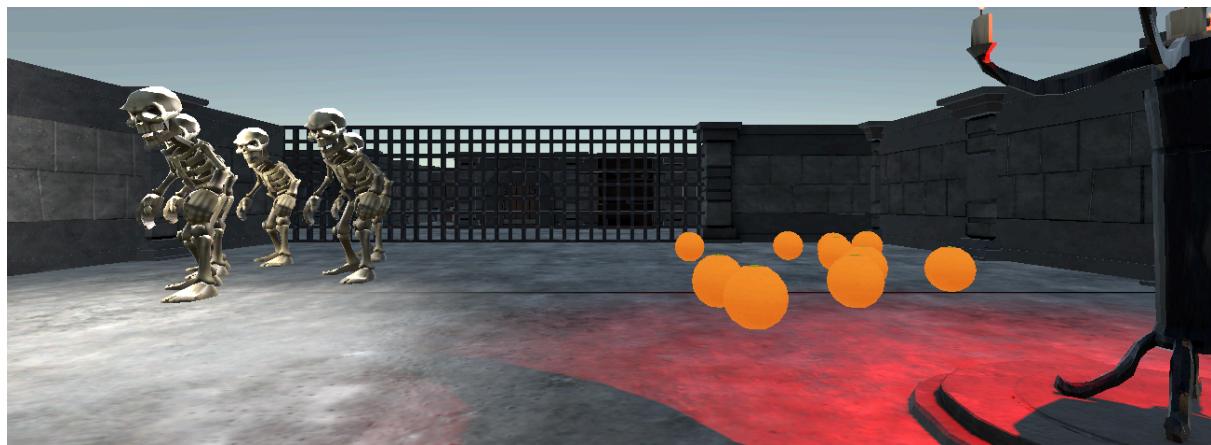
<b>Introducción.....</b>	<b>3</b>
Visión computacional.....	4
Navegación Autónoma.....	4
Métricas en tiempo real.....	5
<b>Parte 1. Sistemas multiagentes.....</b>	<b>6</b>
<b>Propiedades de Agentes y Ambiente.....</b>	<b>6</b>
<b>Tipo de Agentes: Robots (Robot Agent).....</b>	<b>6</b>
Capacidades.....	6
Sensores.....	6
Propiedades del Ambiente.....	6
Comportamientos Implementados.....	7
Sistema de Métricas (Establecido en tiempo Real!):.....	7
<b>Métrica de Utilidad o Éxito de Cada Agente.....</b>	<b>7</b>
Descripción General.....	7
Métricas Principales.....	8
1. Cubos Entregados (Cubes Delivered).....	8
2. Distancia Total Recorrida (Total Distance).....	8
3. Ratio de Eficiencia (Efficiency Ratio).....	8
4. Tasa de Entrega (Delivery Rate).....	8
Interpretación de Resultados.....	8
Rendimiento Destacado.....	8
Rendimiento Medio.....	8
Rendimiento Bajo.....	9
Conclusiones de las métricas.....	9
<b>Diagramas de Clases de los Agentes.....</b>	<b>10</b>
<b>Diagramas de Clases de las Ontologías.....</b>	<b>12</b>
<b>Estrategia de Mejora de la Eficiencia de los Agentes.....</b>	<b>13</b>
1. Optimización de Selección de Objetivos.....	13
2. Coordinación Entre Agentes.....	13
3. Gestión de Rutas.....	13
4. Optimización de Recursos.....	13
5. Mejoras en el Sistema de Decisiones.....	13
6. Optimización de Entrega.....	13
7. Sistema de Aprendizaje y Adaptación.....	14
8. Gestión de Congestión.....	14
9. Optimización de Exploración.....	14
10. Sistema de Recuperación.....	14
<b>Conclusión.....</b>	<b>14</b>
<b>Parte 2. Gráficas Computacionales.....</b>	<b>15</b>
Modelos con materiales (colores) y texturas (usando mapeo UV):.....	15
Animación (tienen diferentes animaciones ya sea correr o caminar).....	15
Iluminación.....	16
Detección de colisiones básica.....	16
<b>Parte 3. Visión Computacional.....</b>	<b>16</b>

# Introducción.

En este trabajo, se ha creado una simulación donde cinco robots, ilustrados por esqueletos animados, tienen que ordenar un grupo de objetos (anteriormente botellas de Coca-Cola ahora naranjas debido a que era más sencillo que fueran naranjas) en un entorno desordenado. El entorno se compone de un laberinto construido en piedra, con la función particular de mover los objetos desde una base roja (A) hasta otra azul (B) las cuales han sido actualizadas y ahora únicamente se destacan por las luces azul y rojo. Los robots deben desplazarse por el laberinto, superando barreras, y garantizar que los objetos sean colocados de manera ordenada en el lugar final.

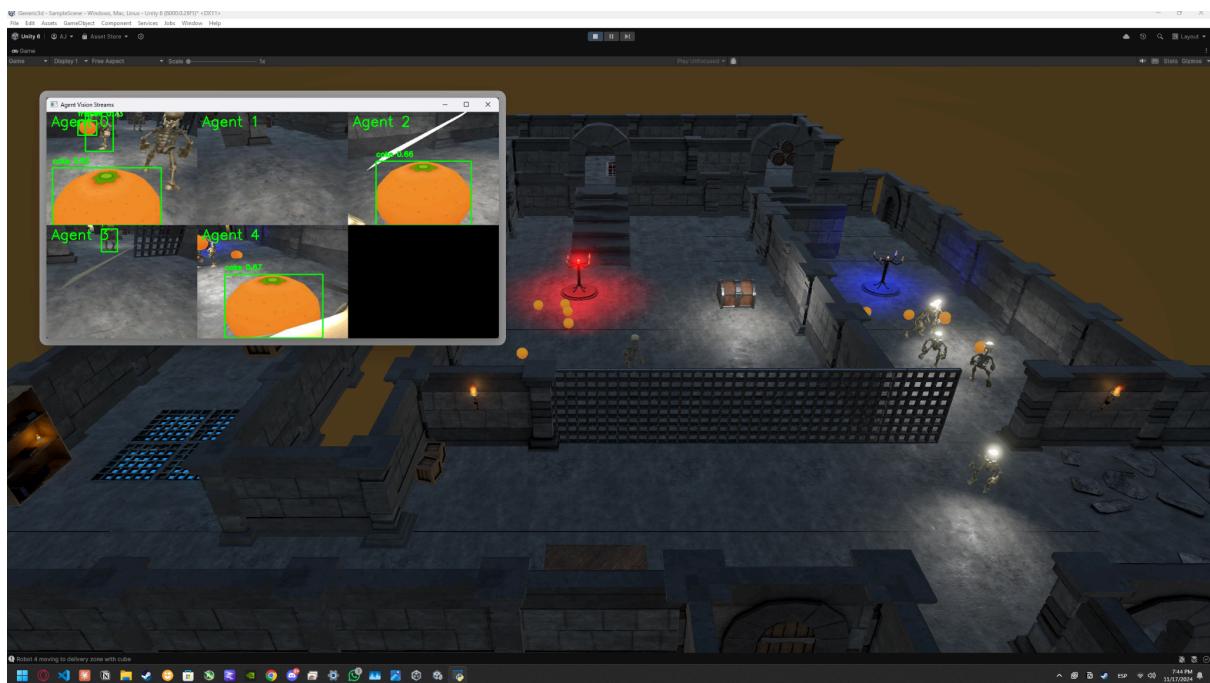
Cada robot dispone de visión computacional que facilita realizar la identificación de sus objetos, al igual que otros que contenga la base de datos YOLOV5, tiene información como los robots y la cantidad de objetos en el entorno no entregados. La simulación se centra en cómo los robots cooperan para ejecutar la labor de llevar los objetos de manera eficaz, incrementando la rapidez en la organización de los objetos y reduciendo los movimientos superfluos.

El propósito principal del proyecto es valorar el desempeño de estos agentes en términos de utilidad, empleando diversas tácticas de coordinación como un navMesh para que puedan encontrar obstáculos y cambiar su dirección, al mismo tiempo que se recopila información acerca de la distancia en los movimientos efectuados y el tiempo requerido para finalizar la labor.

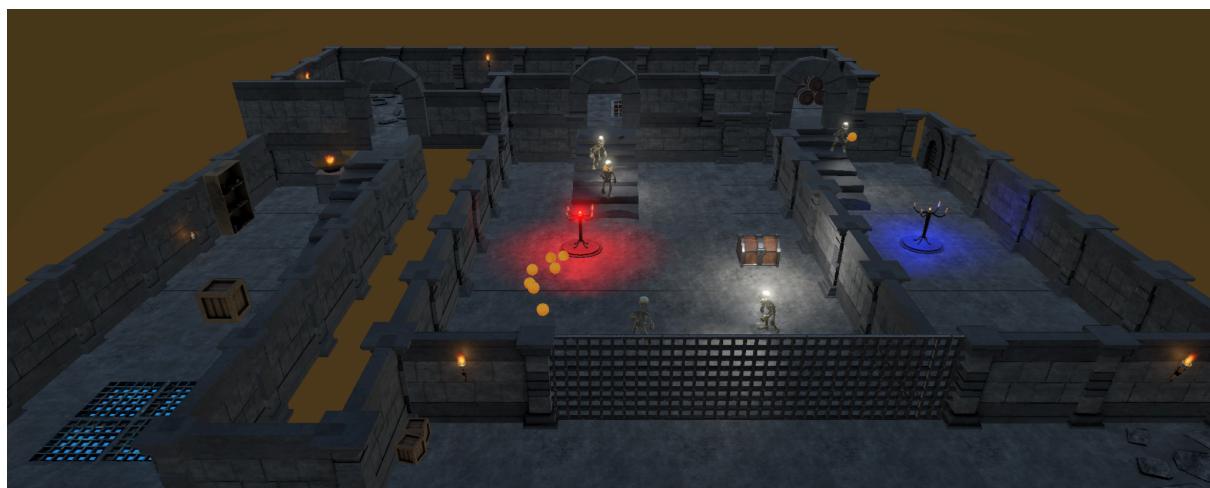


Tenemos diversos ejemplos de lo que hicimos en este proyecto como por ejemplo:

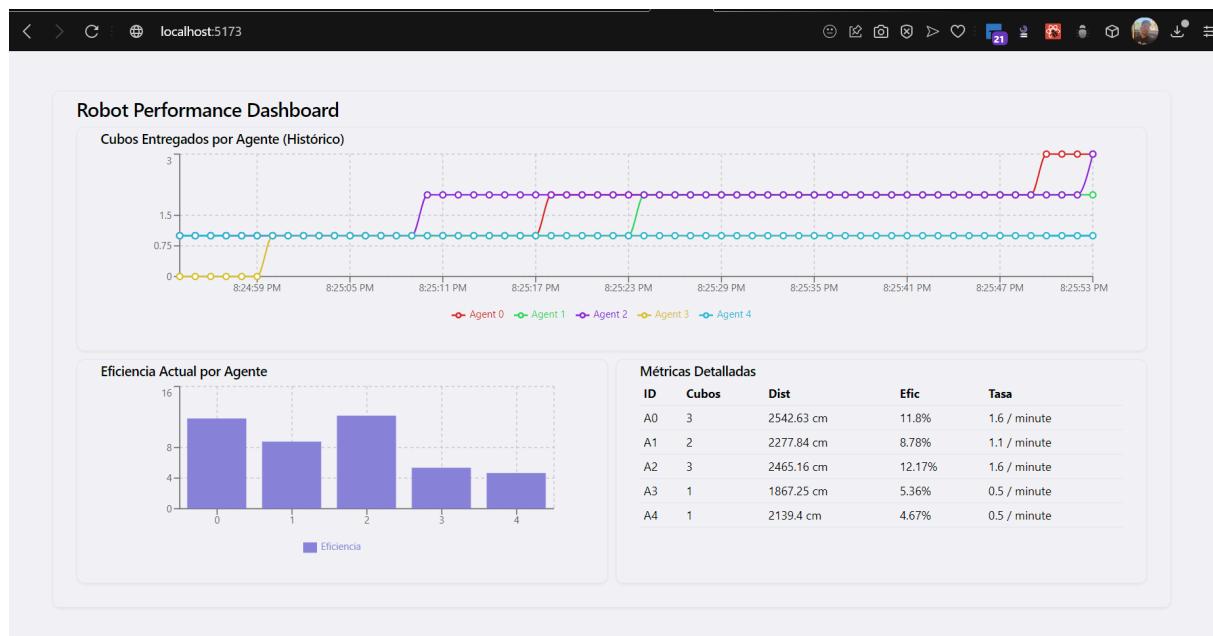
## Visión computacional



## Navegación Autónoma



## Métricas en tiempo real



# Parte 1. Sistemas multiagentes

## Propiedades de Agentes y Ambiente.

### Tipo de Agentes: Robots (Robot Agent)

Capacidades:

- Movimiento basado en NavMeshAgent con velocidad de 15 unidades
- Detección y manejo de colisiones usando NavMesh
- Capacidad de recoger y soltar objetos (cubes) usando sistema de attachment points
- Comunicación entre agentes mediante un sistema de bloqueo (lockObject) para evitar conflictos al recoger cubos
- Sistema de salto para superar obstáculos (jumpForce = 5f)
- Animaciones de movimiento controladas por Animator Script

Sensores:

- Raycast para detección de obstáculos altos (heightCheckDistance = 0.5f)
- Detección de distancia para interacción con cubos (stoppingDistance = 1.5f)
- Sistema de verificación de suelo (IsGrounded()) //no dejarlos saltar mientras están en el aire
- Detección de estado de carga (hasCube boolean)

Propiedades del Ambiente:

- Espacio de navegación definido por NavMesh
- Tiempo requerido para las funciones de los agentes aprox 2 min, la ejecución de este ejemplo dura exactamente 2 minutos.
- Es una dungeon de aproximadamente 120 x 25 x 180 debido a que tiene 2 pisos y muchas habitaciones.
- basado en logs de python de las veces que se conectó a ellos para tomar una decisión aproximadamente tenemos que son 70 movimientos por objeto entregado.
- Dos zonas distintas:
  - Plano A (rojo): Zona de recolección inicial (-15, -1.19, 0)
  - Plano B (azul): Zona de entrega (30, -1.19, 0)
- Sistema de cubos:
  - 10 cubos distribuidos aleatoriamente en la zona roja
  - Posiciones: x(-20 a -10), z(-10 a 10)
- Dimensiones navegables:
  - Área de exploración: x(-20 a 20), z(-10 a 10) // siempre regresan a esta área una vez entregados los cubos, expandible para buscarlos en todo el mapa.
  - Radio de navegación del agente: 0.3f

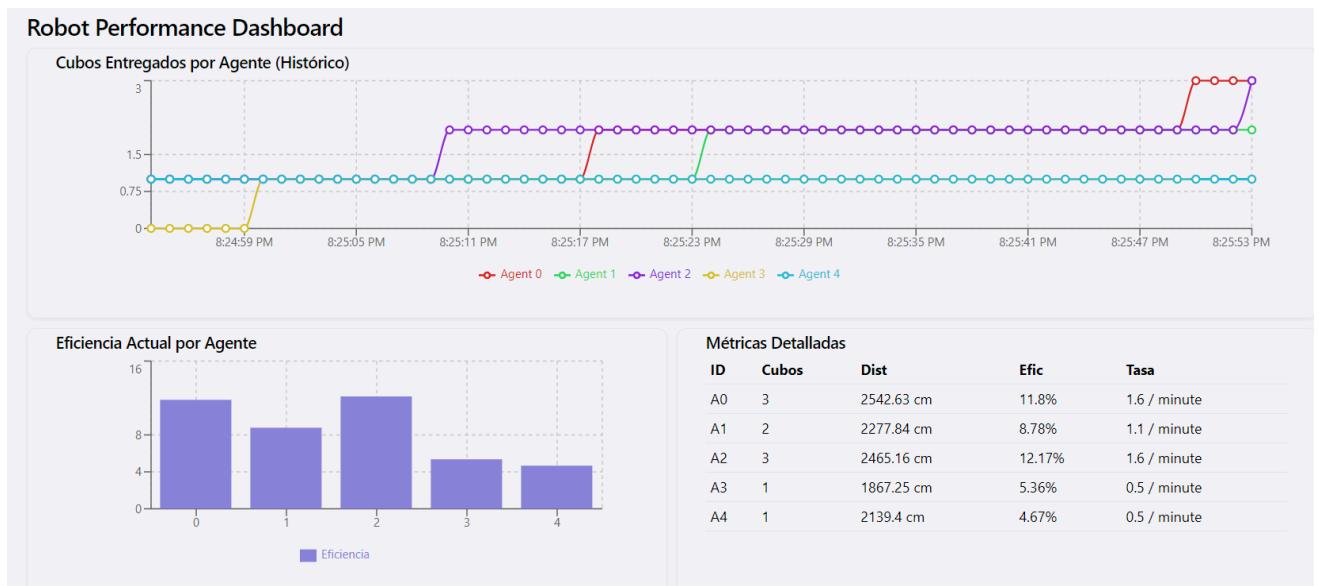
## Comportamientos Implementados:

- `get_cube`: Movimiento hacia y recolección de cubos
- `deliver_cube`: Movimiento hacia zona de entrega
- `put_cube`: Depositar cubo en zona de entrega
- `explore`: Movimiento aleatorio en el espacio navegable

## Sistema de Métricas (Establecido en tiempo Real!):

- Conteo de cubos entregados
- Distancia total recorrida
- Ratio de eficiencia (cubos/distancia)
- Tasa de entrega (cubos por minuto)
- Tiempo transcurrido en minutos

# Métrica de Utilidad o Éxito de Cada Agente.



## Descripción General

Las métricas de utilidad son indicadores clave que miden el rendimiento y la eficiencia de cada agente robótico en el sistema. Estas métricas se calculan continuamente durante la operación y proporcionan una visión integral del éxito de cada robot en su tarea de recolección y entrega de cubos.

# Métricas Principales

## 1. Cubos Entregados (Cubes Delivered)

- **Qué mide:** El número total de veces que se manda llamar la función de dejar un cubo.
- **Importancia:** Indica directamente la productividad del agente
- **Ejemplo:** El Agente A0 y A2 muestran el mejor rendimiento con 3 cubos entregados cada uno

## 2. Distancia Total Recorrida (Total Distance)

- **Qué mide:** La distancia acumulada en centímetros que el robot ha recorrido durante su operación
- **Importancia:** Ayuda a evaluar el desgaste y el consumo de energía del robot
- **Ejemplo:** El Agente A0 ha recorrido 2542.63 cm

## 3. Ratio de Eficiencia (Efficiency Ratio)

- **Qué mide:** La relación entre cubos entregados y distancia recorrida (cubos/distancia tiempo)
- **Importancia:** Indica qué tan eficientemente el robot está utilizando su movimiento y tiempo
- **Ejemplo:** El Agente A2 muestra la mejor eficiencia con 12.17%

## 4. Tasa de Entrega (Delivery Rate)

- **Qué mide:** Número de cubos entregados por minuto
- **Importancia:** Indica la velocidad de operación del agente
- **Ejemplo:** Los Agentes A0 y A2 tienen la mejor tasa con 1.6 cubos/minuto

# Interpretación de Resultados

## Rendimiento Destacado

- Los Agentes A0 y A2 son los más efectivos, con:
  - Mayor número de entregas (3 cubos)
  - Mejores tasas de entrega (1.6/minuto)
  - Mejores ratios de eficiencia (11.8% y 12.17% respectivamente)

## Rendimiento Medio

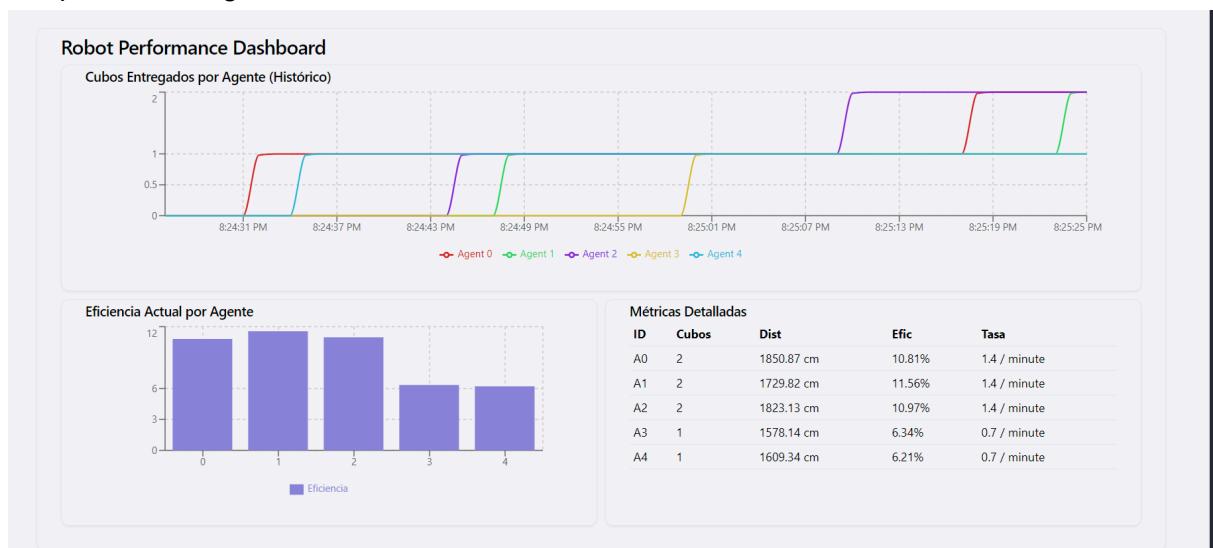
- El Agente A1 muestra un rendimiento intermedio con:
  - 2 cubos entregados
  - Una tasa de entrega de 1.1/minuto
  - Eficiencia de 8.78%

## Rendimiento Bajo

- Los Agentes A3 y A4 muestran el rendimiento más bajo:
  - Solo 1 cubo entregado cada uno
  - Menor tasa de entrega (0.5/minuto)
  - Eficiencia más baja (5.36% y 4.67% respectivamente)

## Conclusiones de las métricas.

- Existe una clara variación en el rendimiento entre agentes y es curioso porque de momento únicamente los cambios son variaciones del ambiente, posición de aparición y el reconocimiento de Yolo.
- La eficiencia no solo depende del número de entregas, sino también de la optimización de rutas, choques con objetos y otros agentes, también tiene razones con la visión computacional.
- Los agentes más exitosos combinan alta productividad con movimiento eficiente y tiempos de entrega correctos.



- Las métricas van cambiando con cada uso del entorno

# Diagramas de Clases de los Agentes.

## RobotAgent

### State Description:

The agent represents an autonomous robot that collects cubes from one zone and delivers them to another. It maintains states for:

- Unique ID
- Current position
- Whether it's carrying a cube
- Target cube information
- Movement status (isMoving, isJumping)
- Navigation status via NavMeshAgent

### Actions:

- MoveToCube(): Navigates to and picks up targeted cubes
- MoveToDeliveryZone(): Transports cubes to designated delivery area
- Explore(): Moves randomly when no specific task
- PutCube(): Drops carried cube at current location
- Jump(): Overcomes obstacles through vertical movement

### Methods:

- MoveToCube(): Navigates to and picks up targeted cubes
- MoveToDeliveryZone(): Transports cubes to designated delivery area
- Explore(): Moves randomly when no specific task
- PutCube(): Drops carried cube at current location
- Jump(): Overcomes obstacles through vertical movement

### Capabilities, Service Description, Supported Protocols:

- Autonomous navigation using Unity's NavMesh system
- Obstacle detection and avoidance
- Cube manipulation (pickup/drop)
- Synchronized multi-agent cube targeting system
- Visual feedback through animations
- Physics-based movement and interactions

### [Constraint] Society:

Operates in a shared environment with other robots, requiring:

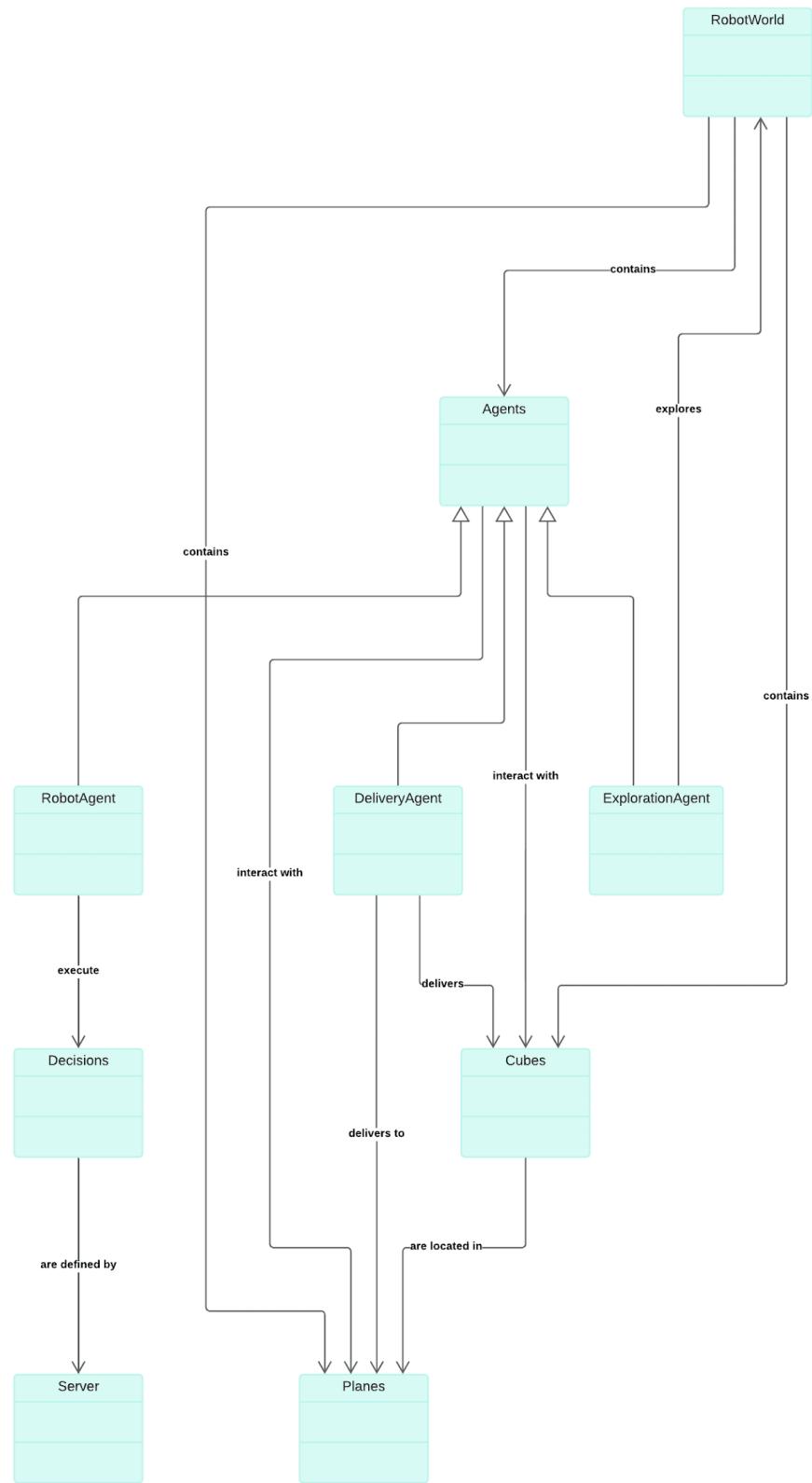
- Cube claim coordination to prevent conflicts

- Shared delivery zone management
- NavMesh pathfinding constraints
- Physics-based movement limitation

#### **List of Communicative Acts (CA):**

- What: Coordinate cube collection and delivery
- Why: To efficiently transport cubes from source to delivery zone
- How: Through synchronized state management and decision making via Python backend
- Examples:
  - "Robot {id} claimed cube {cubeId}"
  - "Robot {id} moving to pick up cube"
  - "Robot {id} moving to delivery zone with cube"
  - "Robot {id} dropped cube {cubeId}"

# Diagramas de Clases de las Ontologías.



# Estrategia de Mejora de la Eficiencia de los Agentes

Para mejorar la eficiencia de los robots en la organización de los objetos, se podrían implementar las siguientes estrategias:

## 1. Optimización de Selección de Objetivos

- Implementar un sistema de priorización de cubos basado no solo en la distancia, sino también en la densidad de cubos en el área
- Considerar la dirección actual del movimiento del robot para favorecer cubos que estén en su trayectoria
- Evitar que múltiples agentes se dirijan a áreas con pocos cubos

## 2. Coordinación Entre Agentes

- Establecer zonas de operación dinámicas para cada agente basadas en la distribución actual de cubos
- Implementar un sistema de comunicación entre agentes para evitar interferencias en sus rutas
- Crear un mecanismo de "relevos" donde los agentes pueden transferir cubos entre sí para optimizar las entregas

## 3. Gestión de Rutas

- Implementar pathfinding con memoria para evitar áreas donde el agente ha experimentado obstáculos
- Desarrollar un sistema de "rutas preferenciales" basado en el histórico de entregas exitosas
- Considerar el establecimiento de "puntos de control" intermedios para optimizar las rutas largas

## 4. Optimización de Recursos

- Implementar un sistema de "descanso estratégico" cuando no hay cubos disponibles en lugar de exploración aleatoria esto apoya mucho ya que la eficiencia del agente disminuye significativamente si camina largas distancias.
- Crear zonas de espera optimizadas cerca de áreas de alta densidad de cubos, (actualmente hacen algo parecido a esto)
- Establecer puntos de retorno estratégicos después de las entregas (lo hacen no es necesario)

## 5. Mejoras en el Sistema de Decisiones

- Implementar un sistema de predicción de movimientos de otros agentes para evitar colisiones
- Desarrollar un mecanismo de prioridad basado en la eficiencia histórica de cada agente (interesante)
- Crear un sistema de "especialización" donde los agentes se optimizan para rutas cortas o largas según su rendimiento

## 6. Optimización de Entrega

- Establecer puntos de entrega dinámicos en la zona B basados en la congestión actual (muy interesante, evitaremos el amontonamiento y el caso accidental de sacar del área objetos)

- Implementar un sistema de cola para entregas cuando múltiples agentes llegan simultáneamente (tienen algo parecido actualmente, no es implementado a propósito)
- Crear zonas de aproximación optimizadas para reducir el tiempo de entrega final (derrapan al llegar por lo que es interesante esta idea)

## 7. Sistema de Aprendizaje y Adaptación

- Implementar un sistema de registro de rutas exitosas para mejorar futuras decisiones (path saving)
- Desarrollar métricas de eficiencia más detalladas para identificar áreas de mejora específicas
- Crear un sistema de adaptación de parámetros basado en el rendimiento histórico

## 8. Gestión de Congestión

- Implementar un sistema de detección y evitación de áreas congestionadas
- Crear rutas alternativas predefinidas para situaciones de alto tráfico
- Desarrollar un sistema de prioridad de paso en zonas estrechas o congestionadas

## 9. Optimización de Exploración

- Implementar patrones de exploración sistemáticos en lugar de movimientos aleatorios
- Desarrollar un sistema de mapeo de áreas productivas basado en hallazgos históricos
- Crear zonas de exploración prioritarias basadas en la probabilidad de encontrar cubos

## 10. Sistema de Recuperación

- Implementar estrategias de recuperación cuando un agente se queda atascado
- Desarrollar mecanismos de redistribución de tareas cuando un agente está teniendo un bajo rendimiento
- Crear protocolos de "reinicio de tarea" cuando una misión actual se vuelve ineficiente

# Conclusión

La simulación creada evidencia la habilidad de los robots para cooperar en un ambiente retador, tal como un almacén con barreras. Pese a que se consigue la tarea, la eficacia de los robots puede incrementarse considerablemente mediante la aplicación de estrategias sofisticadas de coordinación y planificación de rutas. La distribución de roles, la aplicación de algoritmos de ruta óptima y la puesta en marcha de protocolos de prioridad son algunas de las optimizaciones que podrían disminuir el tiempo total de la tarea y los movimientos innecesarios.

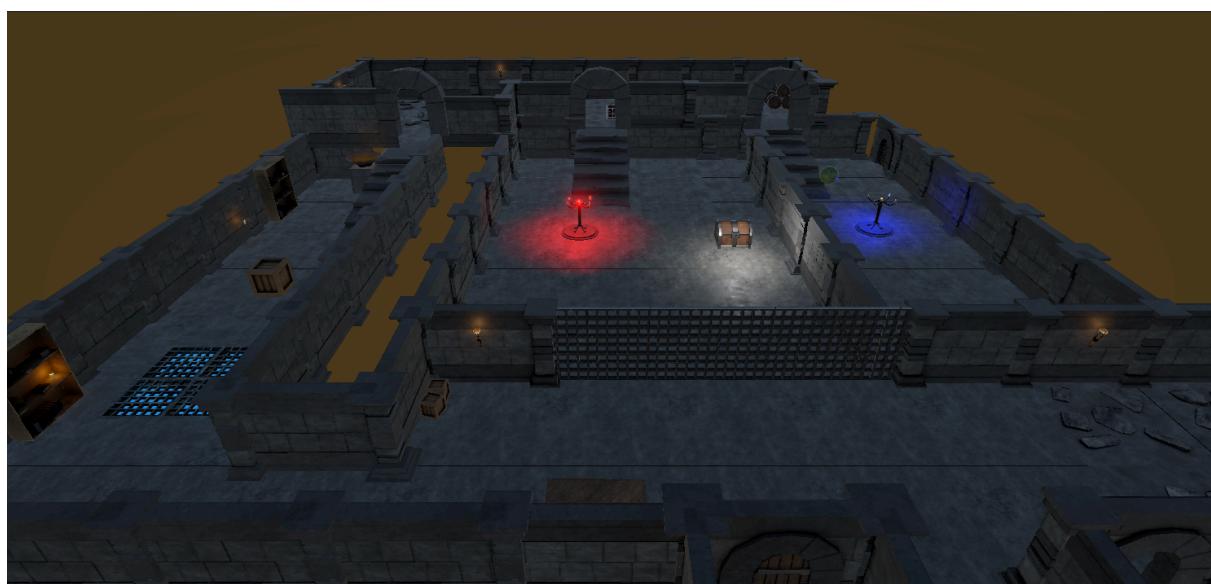
Además, la implementación de tecnologías como mapas compartidos podría facilitar a los robots una mejor adaptación a su ambiente, previendo dificultades y mejorando sus elecciones. En términos generales, la puesta en marcha de estas mejoras no solo aceleraría

el trabajo, sino que también proporciona una mayor adaptabilidad y flexibilidad ante variaciones en el ambiente o en las condiciones de la labor.

Este proyecto resalta la importancia de la planificación y la coordinación en los sistemas multiagentes, y ofrece una base sólida para futuras investigaciones en la optimización de agentes autónomos en entornos complejos.

## Parte 2. Gráficas Computacionales

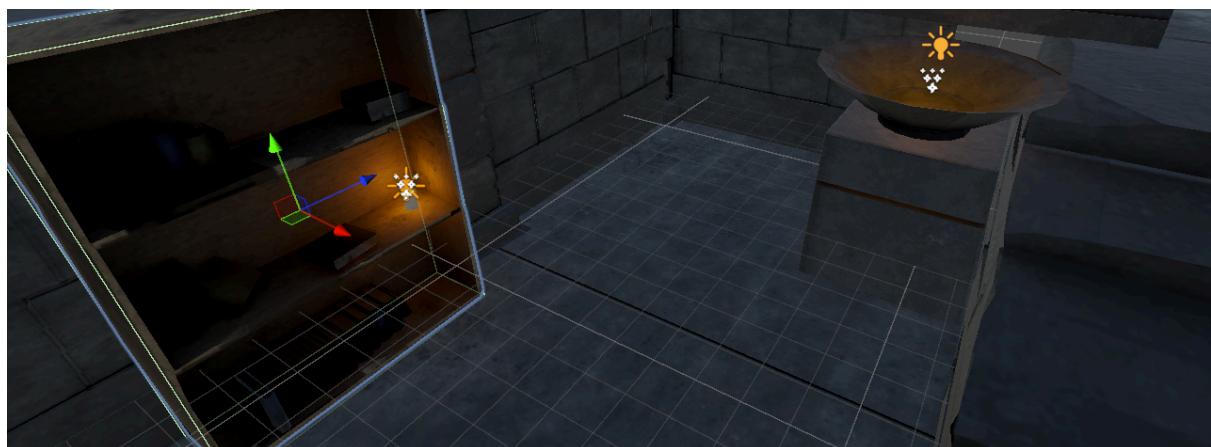
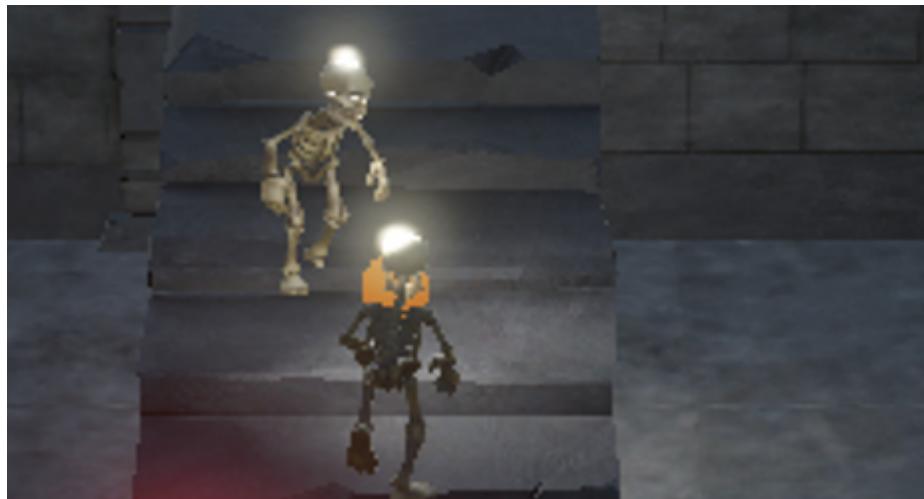
**Modelos con materiales (colores) y texturas (usando mapeo UV):**



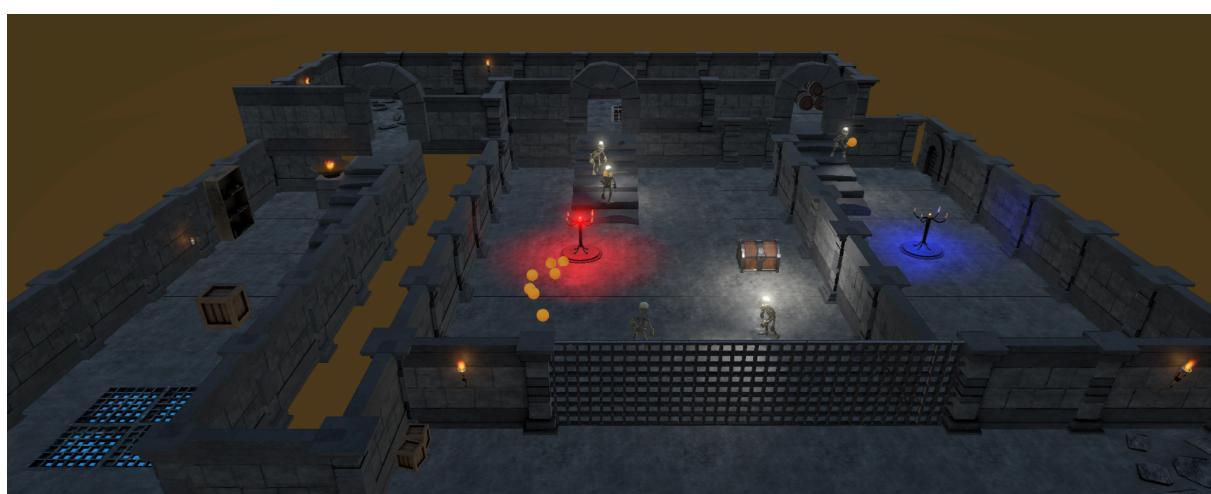
**Animación (tienen diferentes animaciones ya sea correr o caminar)**



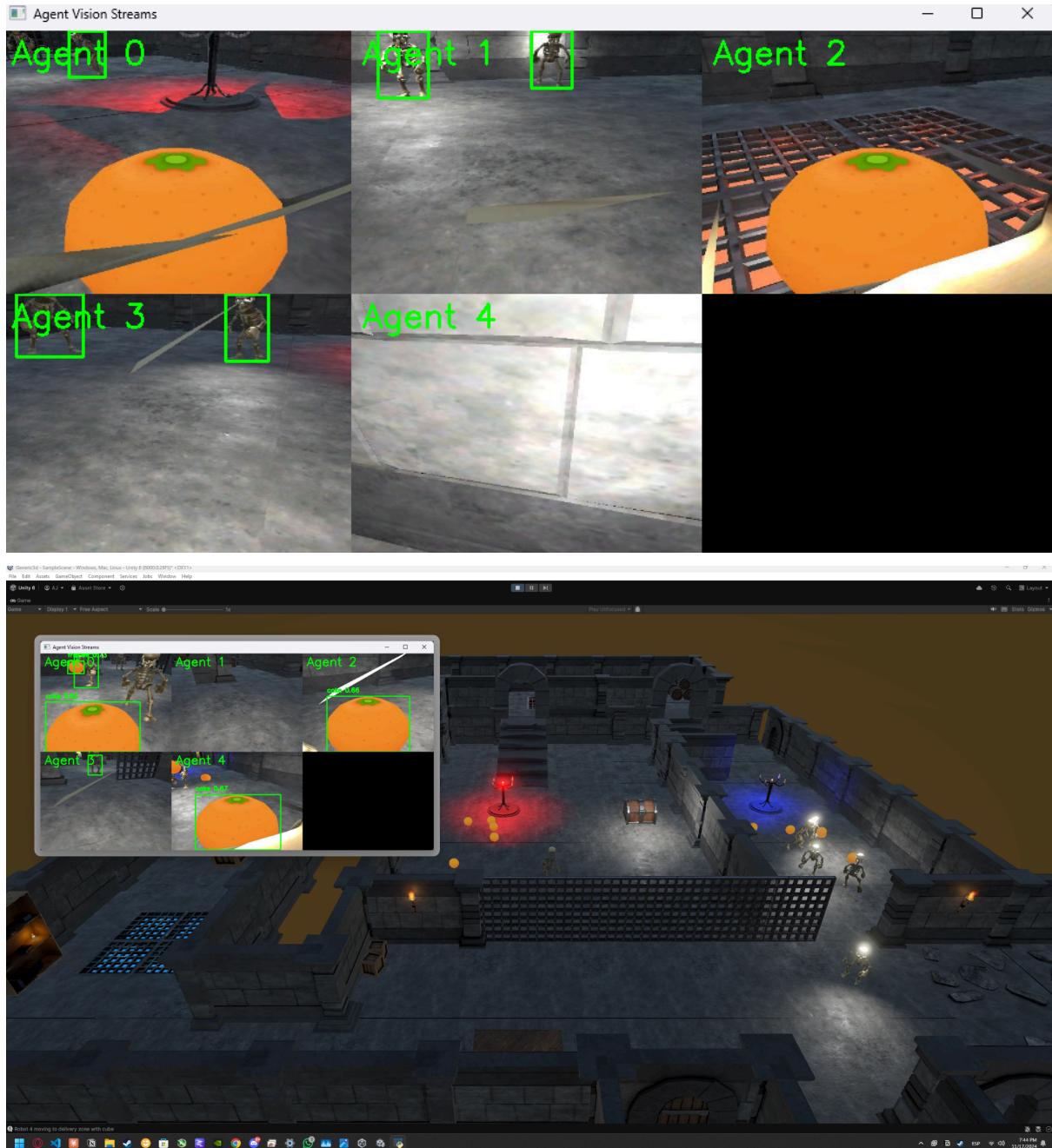
## Iluminación



## Detección de colisiones básica



## Parte 3. Visión Computacional



Cada agente está mandando en tiempo real con un protocolo UDP un stream de su cámara para que YOLOV5 lo analize.

Anteriormente utilizaba un sistema más rudimentario para esto comprimiendo imágenes base64 y mandandolas a python por json en base64 pero era demasiado lento así que cambié el enfoque, una vez que reconocen el objeto que necesitan dan un anuncio en la consola de objeto reconocido.