UNIVERSIDADE DA CORUÑA

Grado en Intelixencia Artificial
Sistemas Multiaxente

Práctica 2: Sistemas Multiagente

Jesus Silva Vicente
Miguel López Lópe
Grupo de clase: martes

Mundo Virtual

El entorno virtual desarrollado en Unity recrea un escenario cerrado compuesto por varias estancias conectadas por pasillos, diseñado para simular una situación de vigilancia y persecución entre agentes. El objetivo principal es capturar a un agente ladrón mientras se protege un tesoro (el oro) y se evita su fuga por una puerta.

- Componentes del escenario:

  Estructura arquitectónica: El mapa incluye salas rectangulares, pasillos de interconexión, esquinas ciegas y puertas de paso. Las paredes y objetos como puertas utilizan colisionadores estáticos marcados con la capa Obstaculos, que afectan tanto a la navegación como a la visión.

  NavMesh y navegación: Se ha horneado un NavMesh global sobre las superficies transitables del mapa, lo que permite a los agentes usar NavMeshAgent para desplazarse evitando obstáculos.

- Puntos clave del mapa:

  goldPoint: una posición fija donde se encuentra el oro. Es protegida dinámicamente por los policías cuando el ladrón es avistado o se sospecha que intentará robarlo.

  doorPoint: ubicación de la puerta de escape. Si el ladrón no es interceptado a tiempo, puede dirigirse a esta zona para fugarse.

  destinos: lista de waypoints usados para patrullar. Se recorren cíclicamente mientras no haya otras tareas de mayor prioridad.

- Interacción con el entorno : Se han integrado sistemas de interacción que refuerzan la credibilidad del entorno

  Detector de presencia: Zonas delimitadas por colisionadores trigger asociadas a un script DetectorPresencia. Estas detectan la entrada y salida del ladrón (o cualquier objeto con etiqueta específica como "Player") y activan una puerta (PuertaCorredera) mediante el método PersonaDetectada(bool).

  Sensores visuales: Cada policía cuenta con un trigger adicional (esfera o cápsula) que permite detectar si el ladrón entra en su rango de visión. La visibilidad se calcula con raycasting, teniendo en cuenta ángulo, distancia y obstrucción por obstáculos.

Arquitectura individual

Cada agente policial ha sido diseñado como una entidad completamente autónoma, dotada de capacidades de percepción, navegación, planificación jerárquica y comunicación asíncrona, sin depender de ningún controlador central. Para estructurar su comportamiento, se ha seguido una arquitectura en tres capas funcionales que interactúan entre sí:

1. Navegación autónoma

   Los desplazamientos se gestionan mediante el componente NavMeshAgent, que permite a cada policía moverse por el entorno siguiendo rutas calculadas dinámicamente. Se accede a posiciones clave como:

   goldPoint: zona del oro.

   doorPoint: salida del ladrón.

   thiefTransform.position: posición estimada del ladrón en tiempo real.

   El destino del agente se actualiza constantemente en función de su estado y de los eventos que percibe o recibe mediante mensajes. Las decisiones de navegación están condicionadas por varios flags booleanos que definen el contexto del agente:

   ocupado: indica si está ejecutando una tarea HTN prioritaria.

   isPatrolling: si está en modo patrulla.

   ladronViendo, ladronPerdido: estados de percepción del ladrón.

Esto garantiza que el movimiento se adapte al comportamiento del agente y no se base únicamente en scripts fijos.

2. Planificación jerárquica (HTN) Para estructurar su comportamiento, cada agente cuenta con una instancia del planificador PlaneadorHTN, que sigue el modelo de planificación jerárquica de tareas (HTN). Este planificador mantiene una cola dinámica de tareas (QueueTareaHTN), generadas en función del estado actual del agente y de eventos recientes.

   Comportamiento del planificador:acceptpproposal Si ladronViendo es true: → Se encola TareaPerseguir.

   Si el ladrón no está a la vista y el agente está libre (!ocupado y !isSearching): → Se encola TareaPatrullar.

   Tras recibir un acceptpproposal en una subasta: → Se encola TareaOro, TareaPuerta o TareaInterceptar.

   Ejecución: Las tareas se ejecutan mediante una coroutine (EjecutarPlan), que extrae tareas de la cola y ejecuta cada una mediante su propia coroutine. Esto permite que cada tarea controle su propia lógica temporal (esperas, desplazamientos, condiciones de parada, etc.).

3. Conjunto de tareas disponibles (HTN)

   · TareaPerseguir:

   La finalidad de esta tarea es que el agente policial pueda perseguir al ladrón de forma inmediata y continua mientras permanezca dentro de su campo visual. Esta tarea se activa cuando el estado interno del agente indica que el ladrón ha sido detectado (ladronViendo = true) y que se tiene acceso al transform del ladrón. Esta tarea cancela cualquier patrulla activa previamente e inicia un bucle donde, cada 0.2 segundos, se actualiza la posición objetivo del agente con la posición actual del ladrón (thiefTransform.position). Esta aproximación garantiza un seguimiento reactivo y preciso, adaptándose a los movimientos del objetivo en tiempo real.

   El agente mantiene este comportamiento mientras siga viendo al ladrón. En cuanto se pierde la línea de visión o se produce una colisión que indique captura, la tarea finaliza y se transfiere el control al planificador HTN para reasignar comportamientos. Este diseño simula un comportamiento realista de un guardia de seguridad que reacciona de

inmediato ante una amenaza visual, utilizando el sistema de navegación de Unity (NavMeshAgent) para ejecutar la persecución.

· TareaPatrullar:

La tarea de patrullar tiene como finalidad garantizar la cobertura pasiva del entorno cuando no existen amenazas inmediatas. Se trata del comportamiento por defecto de los agentes cuando no están ocupados ni están viendo al ladrón. Consiste en recorrer una serie de puntos prefijados (contenidos en el array destinos) en un orden circular, avanzando al siguiente solo si el agente ha alcanzado su destino anterior. Este control se lleva a cabo mediante la verificación del campo remainingDistance del NavMeshAgent.

Esta tarea ofrece un patrón de vigilancia sistemático y cíclico que asegura que todas las zonas del entorno son supervisadas con regularidad. Al estar condicionada por el estado del agente (no debe estar ocupado ni debe haber detección visual del ladrón), se activa únicamente en escenarios sin incidentes activos. Además, su implementación como tarea atómica en HTN permite que pueda ser fácilmente interrumpida por eventos externos, como la aparición del ladrón o la recepción de un mensaje relevante.

· TareaOro y TareaPuerta:

Ambas tareas están diseñadas para permitir a los agentes desplazarse a puntos estratégicos del entorno tras ganar subastas cooperativas (AuctionGold o AuctionDoor). La finalidad de estas tareas es mantener vigilancia y control en áreas sensibles del mapa, como la cámara acorazada o la salido. La activación de estas tareas ocurre exclusivamente cuando el agente ha sido elegido como ganador de la correspondiente subasta, lo que garantiza una asignación eficiente y descentralizada de roles.

El funcionamiento consiste en que el agente, al ganar la subasta, obtiene la posición objetivo (goldPoint o doorPoint) y se desplaza hacia ella utilizando NavMeshAgent. En el caso de la TareaOro, el agente permanece allí durante un tiempo predefinido, actuando como guardia estático. En el caso de la TareaPuerta, el agente se queda en la puerta vigilando indefinidamente o hasta que se produzca una interrupción por eventos externos.

Estas tareas permiten simular comportamientos defensivos y estratégicos, reforzando zonas clave mediante una coordinación emergente entre los

agentes. No existe un coordinador central que asigne estas posiciones, sino que el mecanismo de subasta distribuye los roles de forma dinámica y eficiente en función de la disponibilidad y la cercanía de cada agente.

· TareaInterceptar:

Esta tarea actúa como respuesta táctica intermedia cuando el ladrón ha sido visto por uno de los policias y este solicita un refuerzo. Su propósito es que el agente se dirija a la última posición conocida del ladrón, con la esperanza de poder recuperarlo visualmente o interceptarlo si sigue cerca. La tarea se activa tras ganar la subasta AuctionIntercept.

Durante su ejecución, el agente se marca como ocupado y se dirige al punto recibido como última posición del ladrón. Si durante el trayecto el ladrón reaparece en el campo de visión, la tarea se cancela automáticamente y el sistema transiciona a TareaPerseguir. Si el agente llega al punto y no se ha restablecido contacto visual, se considera que el ladrón ha escapado y se relanza el mecanismo de subastas para redistribuir los agentes disponibles hacia nuevas posiciones clave como el oro o la puerta.

Esta lógica simula de forma realista el comportamiento de un guardia que acude rápidamente a un lugar donde se ha detectado una amenaza, aunque no la tenga localizada en ese momento. Es una tarea crítica para dar continuidad a la persecución y evitar que los agentes regresen inmediatamente a patrullar sin haber asegurado la zona.

· TareaAuction:

Esta tarea representa el inicio de un proceso de coordinación distribuida entre agentes basado en el protocolo FIPA. Su finalidad es lanzar subastas para cubrir puntos estratégicos del entorno cuando se detecta al ladrón. Por ejemplo, al ver al ladrón, un agente puede iniciar automáticamente las subastas AUCTIONGOLD, AUCTIONDOOR y AUCTIONINTERCEPT, enviando mensajes CFP al resto de agentes.

El funcionamiento se basa en comprobar si ya existe una subasta activa con el mismo ID; si no es así, se crea un nuevo estado de subasta, se formatea la información de destino (coordenadas del oro, la puerta o la posición del ladrón) y se lanza el mensaje CFP. El agente recolecta propuestas durante un tiempo limitado, selecciona al mejor candidato (usualmente el más cercano y disponible) y le envía un mensaje de

ACCEPTPROPOSAL, rechazando al resto. Posteriormente, el agente seleccionado ejecuta la tarea asociada.

Esta tarea es esencial para la cooperación y coordinación entre agentes sin necesidad de una jerarquía ni un centro de control. Permite una gestión dinámica, escalable y robusta de los recursos, asegurando que las zonas más críticas estén siempre cubiertas por al menos un agente.

Comunicación entre agentes

Cada policía extiende de CommunicationAgent, una clase base que implementa el protocolo de comunicación FIPA ACL. Esta arquitectura permite emitir, recibir y procesar mensajes asincrónicos entre agentes.

Implementación del protocolo FIPA. Los agentes procesan los siguientes performatives:

- CFP (Call For Proposal): Convoca una subasta. Incluye ID y coordenadas del objetivo.
- PROPOSE: Envia una oferta (bid) basada en la distancia al objetivo, penalizada si el agente está ocupado.
- ACCEPT_PROPOSAL / REJECT_PROPOSAL: Notifica el resultado de la subasta.
- INFORM: Empleado para difundir eventos globales como detección, pérdida o captura del ladrón.
- NOT_UNDERSTOOD: Se envía cuando un mensaje no se ha entendido.

Lógica de subastas:

1. Cada subasta tiene un identificador (auctiondoor, auctiongold, auctionintercept).
2. El iniciador calcula un target y lo envía a los demás agentes junto con un CFP.
3. Los agentes responden con sus bids. Tras un tiempo de espera (proposalTimeout), se elige al ganador con el menor bid no ocupado.

4. El ganador ejecuta la tarea correspondiente, y los perdedores reanudan su planificación HTN.

El flujo de comunicación se ha diseñado para ser descentralizado y basado en eventos. Cada agente mantiene una cola de mensajes y procesa un número limitado por frame para evitar bloqueos.

Flujo típico:

1. Un agente detecta al ladrón mediante raycast y OnTriggerEnter.
2. Se llama a LadronVisto(thiefTransform), lo que actualiza estados internos y emite thiefspotted.
3. Inicia las correspondientes subastas para detener al ladrón de forma organizada(auctiondoor, auctiongold, auctionintercept).
4. Los demás agentes responden con PROPOSE si son elegibles, incluyendo en el mensaje la distancia al objetivo de la tarea.
5. El agente iniciador recoge propuestas y envía ACCEPT/REJECT.
6. El ganador marca ocupado = true y ejecuta la tarea. Los demás continúan su rutina.

Formato del contenido de los lenguajes

Los mensajes intercambiados son cadenas de texto formateadas en estructuras legibles por humanos:

- Coordenadas:

  "thiefspotted: 14.5,0.0,8.9"
  "CFP: auctiongold: 22.1,0.0,4.3"

- Propuestas:

  "auctiongold:12.34"
  Donde 12.34 es la distancia al objetivo, calculada con Vector3.Distance.

Se utiliza CultureInfo.InvariantCulture para evitar errores por comas decimales en sistemas internacionales.

Estrategia Multiagente

La estrategia colectiva del sistema se basa en cooperación descentralizada mediante subastas dinámicas y planificación local reactiva:

1. Coordinación sin agente central

   No existe un agente supervisor. Cada policía actúa como un nodo autónomo, capaz de:

   - Detectar y compartir información (INFORM).
   - Convocar y participar en subastas (CFP / PROPOSE).
   - Ejecutar planes según su situación actual (HTN local).

   Se realiza un reparto de tareas dinámico cuando el ladrón es detectado, se lanzan unas subastas para proteger y revisar el oro, bloquear la salida principal y dar refuerzo con más policías en las zonas donde el ladrón es detectado.

   Solo un agente por subasta asume la tarea; los demás adaptan su comportamiento para evitar redundancias.

2. Reactividad y adaptabilidad. Si un agente gana una subasta pero detecta al ladrón, puede cancelar la tarea que le fue asignada e iniciar la persecución, ya que la tarea de persecución se considera prioritaria con respecto a las demás.

   Los agentes monitorizan permanentemente su entorno (visión, triggers) y ajustan su planificación constantemente en función de los cambios percibidos.

   Las tareas son interruptibles si cambia el contexto (por ejemplo, alguien captura al ladrón antes o el estado del mundo cambia durante la realizacion de la tarea).