UNIVERSIDADE DA CORUÑA

Grado en Intelixencia Artificial
Sistemas Multiaxente

Práctica 2: Sistemas Multiagente

Martín Barros Iglesias
Marcos Grobas Martínez
Grupo viernes

Mundo Virtual

Diseño del Entorno

- **Distribución espacial**:
  - Un mapa que representa a una nave espacial dividida en 4 áreas que presentan diferentes salas y pasillos, permitiendo dar muchas oportunidades tanto al jugador como al equipo de guardias y cámaras que conforman el sistema multiagente.
  - Se incluyen un conjunto de puntos en el mapa de bloqueo estratégico y otros de patrullaje.
  - Puertas automáticas con sensor de proximidad que se abren al acercarse tanto los guardias como el jugador y se cierran después de un tiempo de espera

Agentes Adicionales

- **Cámaras de vigilancia (2 unidades)**:
  - Campo de visión seleccionable
  - Protocolo de alerta prioritario a guardias asignados (`assignedCoordinator`)
  - Retardo de actualización configurable (`checkInterval`)

Interacción Sensorial

- **Visión**:
  - Sistema de detección basado en capas (`playerLayer`) para identificación precisa

  - Protocolo de dos pasos:
    1. Chequeo de distancia (`directionToPlayer.magnitude < sightRange`)
    2. Raycast direccional con `Physics.Raycast` para verificar línea de visión
  - Configuración modular de rango de visión (`sightRange`)

- **Sonido**:
  - Arquitectura basada en emisores (`SoundEmitter`):
    * Radio sonoro configurable por evento
    * Duración temporal del sonido
  - Detección colaborativa:
    * Cálculo de radio combinado (`hearingRadius + emitter.GetSoundRadius()`)
    * Verificación espacial mediante `Vector3.Distance`
    * Actualización dinámica de emisores activos (`allSoundEmitters`)
  - Visualización en editor con Gizmos personalizados

Arquitectura individual

Arquitectura General (MultiAgentSystem)

El sistema sigue una **arquitectura híbrida** que combina elementos reactivos y deliberativos, implementada mediante una jerarquía de clases donde `MultiAgentSystem` es la clase base abstracta.

- **Características principales**:
  - Comunicación basada en mensajes ACL (FIPA-ACL)
  - Coordinación mediante protocolo de subastas
  - Roles dinámicos (chase, patrol, treasure, etc.)
  - Estado compartido entre agentes

- **Justificación**: Adecuada para vigilancia por:
  - Reactividad rápida a eventos
  - Toma de decisiones coordinada
  - Flexibilidad para diferentes agentes

GuardScript (Agentes Guardias)

Arquitectura: Híbrida reactiva/BDI (Belief-Desire-Intention).

- **Estructura**:
  - Sensores: `VisionSensor` y `HearingSensor`
  - Comportamientos: Patrol, Chase, Treasure, Exit, Blockage
  - Protocolo de subasta heredado
- **Modo de funcionamiento**:
  1. Patrulla en estado normal
  2. Al detectar jugador:
     - Intenta ser coordinador
     - Inicia subasta si es coordinador
     - Cambia comportamiento según rol asignado
- **Justificación**:
  - Autonomía cuando no hay amenazas
  - Coordinación eficiente ante detecciones
  - Adaptabilidad situacional

CameraAgent (Agentes Cámaras)

Arquitectura: Puramente reactiva con asignación jerárquica.

- **Estructura**:
  - Detección por ángulo/distancia de visión
  - Comunicación con guardia asignado
- **Modo de funcionamiento**:
- Al detectar envía mensaje ACL al guardia asignado
- **Justificación**:
  - Sensores especializados sin lógica compleja
  - Delegación de decisiones
  - Bajo costo computacional

Característica	Guardias	Cámaras
Arquitectura	Híbrida (reactiva+BDI)	Puramente reactiva
Toma de decisiones	Participan en subastas	Solo detección
Movilidad	Móviles (NavMesh)	Estáticas
Sensores	Visión+audición	Solo visión
Roles	Múltiples dinámicos	Fijo (detección)

Table 1: Comparativa entre tipos de agentes

Diferencias clave entre agentes

Ventajas del diseño:

- Escalabilidad: fácil añadir más agentes
- Robustez: tolerancia a fallos
- Eficiencia: distribución adecuada de complejidad
- Coordinación: mecanismo de subastas óptimo

Comunicación entre agentes

El sistema implementa un protocolo de comunicación basado en FIPA-ACL (Foundation for Intelligent Physical Agents - Agent Communication Language) con adaptaciones específicas para el dominio de vigilancia.

Estructura de mensajes

Los mensajes siguen el formato:

```
ACLMessage {
    Performative: "REQUEST_BID"|"INFORM"|"ASSIGN_ROLE"|"CAMERA_ALERT",
    Sender: GameObject,
    Receiver: GameObject,
    Content: string (formato estructurado),
    Protocol: "auction"|"chase_protocol"|"camera_protocol",
    Language: "guard_communication"
}
```

Flujo de comunicación

- **Creación**: Mediante el método `SendACLMessage()` que valida los campos requeridos según el protocolo.
- **Envío**: Comunicación directa punto-a-punto entre agentes mediante llamadas a métodos.
- **Recepción**: Todos los mensajes llegan al método `ReceiveACLMessage()` que los deriva según protocolo.

Protocolos implementados

- **Protocolo de subasta**:
  1. Coordinador envía `REQUEST_BID`
  2. Guardias responden con sus distancias calculadas
  3. Coordinador asigna roles con `ASSIGN_ROLE`
- **Protocolo de cámara**:
  1. Cámara envía `CAMERA_ALERT` al guardia asignado
  2. Guardia propaga la información con `INFORM`

Diagrama AUML (Protocolo de Subasta)

```
[Coordinator]                  [Participant]
   | --- CFPAuctionRequest ---> |
   | <-- Propose (Bid) -------- |
   | --------- Accept-Proposal-> |
   | <-- Inform (Result) -------|
   | --------- Confirm --------> |
```

Formato del contenido de los lenguajes

El contenido de los mensajes utiliza un lenguaje estructurado basado en pares clave-valor y listas delimitadas.

Estructuración

- **Posiciones**: Formato `"X;Y;Z"` con coordenadas separadas por punto y coma:

  ```
  "12.345;0.000;-7.890"
  ```

- **Asignaciones de roles**: Simple cadena de texto con el nombre del rol:

  ```
  "chase" | "treasure" | "blockage1"
  ```

Procesamiento

- **Parsing**: Método `ParsePosition()` que:
  1. Divide la cadena por delimitadores
  2. Convierte valores a float con cultura invariante
  3. Valida que haya exactamente 3 componentes
- **Validación**: Chequeo de valores numéricos dentro de rangos válidos del escenario.
- **Manejo de errores**: Excepciones específicas para formatos inválidos.

Estrategia Multiagente

La estrategia implementada sigue un modelo híbrido reactivo-deliberativo con coordinación descentralizada.

Mecanismos de cooperación

- **Subastas holónicas**: Cuando un agente detecta al jugador:
  1. Inicia proceso de subasta como coordinador temporal
  2. Asigna roles óptimos según distancias calculadas
  3. Disuelve el grupo tras la asignación
- **Jerarquía dinámica**: Las cámaras notifican solo a su guardia asignado, creando sub-equipos locales.

Roles adaptativos

- **Asignación dinámica**: Los roles se reasignan según:
  - Proximidad al jugador
  - Estado del tesoro (robado/no robado)
  - Posiciones estratégicas
- **Transiciones**: Los agentes pueden cambiar entre los diferentes roles cada vez que uno de los agentes detecte al jugador e inicie subasta.

Toma de decisiones

- **Local**: Cada agente decide su participación en subastas
- **Global**: El coordinador temporal consolida información y asigna roles
- **Estado compartido**: Variables como `playerHasTreasure` sincronizan conocimiento