Proyecto 1 - Herencia y polimorfismo

Yuen Law

Semestre I, 2022

Objetivos

1. Aplicar conceptos de ingeniería de software.
2. Aplicar conceptos básicos de POO: Herencia y polimorfismo

Descripción

El objetivo de este proyecto es desarrollar una simulación de enjambre utilizando herencia y polimorfismo para modelar los diferentes tipos de agentes. Para este, se debe implementar un agente básico que tenga comportamientos y atributos generales. De este agente heredarán los otros tipos de agentes.

Enjambre

El comportamiento de enjambre que queremos simular es el de hormigas o abejas que buscan recursos y se defienden de amenazas. Cada agente se moverá y tomará decisiones independientemente, pero basado en información de su entorno, el cual puede incluir la presencia de otros agentes. Por ejemplo, si está cerca de otros agentes, puede decidir acercarse o alejarse a estos según su propio estado.

Agentes

Deben implementar 3 tipos de agente: el agente base, recolectores y defensores.

Como se mencionó, el agente base tendrá los atributos y métodos comunes para los otros 2 tipos de agente, pero durante la simulación no es necesario que se creen instancias de éste. El agente recolector buscará recursos y los llevará a la base. Cuando detecta una amenaza, se alejará. El agente defensor también recolecta, pero además si detecta una amenaza, la atacará.

Para los agentes recolectores y defensores debe implementar comportamientos para las siguientes situaciones:

1. El agente está buscando recursos
2. El agente tiene recurso
3. El recolector detecta una amenaza
4. El defensor detecta una amenaza

Interacciones

Los agentes se moverán independientemente buscando recursos, pero también deben modificar su comportamiento si tienen otros agentes cerca. Por ejemplo, si un recolector lleva recursos de regreso a la base y tiene otro agente cerca que también lleva recurso, ajustará ligeramente su camino para tratar de moverse con el otro. Deben incorporar ajustes de comportamiento para al menos los siguientes casos:

1. Un agente lleva recursos, y ve otro(s) que llevan recursos.
2. Un agente está buscando recurso, y ve otro(s) que llevan recursos.
3. Un agente está buscando recurso, y ve otro(s) que también están buscando recursos.
4. Un defensor ve otro(s) agentes que huyen o atacan una amenaza.
5. Un recolector ve otro(s) agentes que huyen o atacan una amenaza.

Objetos

Además de los agentes, existen otros objetos. Estos son: recursos, amenazas y obstáculos. Los obstáculos son objetos fijos que no se pueden mover y aparecen desde un inicio en el mapa. Estos bloquean el camino y no permiten que los agentes pasen. Los recursos y amenazas aparecen aleatoriamente en el tiempo. Los recursos permanecerán en el mapa hasta que sean recolectados. Cada objeto recurso tiene 10 unidades que pueden ser recolectadas de una en una por agentes. De manera similar, las amenazas tienen un contador de vida que inicia en 10. Después de 10 ataques, la amenaza desaparece.

Mapa

El mapa es un tablero de al menos 50×50. Los agentes ocupan 1 solo casilla y se mueven en 4 direcciones: arriba, abajo, izquierda y derecha. Los otros objetos ocupan 4 casillas. La base es una casilla en alguna de las esquinas.

Simulación

No es necesario que la simulación corra en tiempo real, es decir, no es necesario usar hilos. Se recomienda tener estructuras de datos, por ejemplo listas, que se recorren en ciclos para determinar la siguiente acción de cada agente y el estado de los objetos. Aquí es necesario aplicar polimorfismo, es decir, solo se puede usar una estructura para todos los agentes y el procesamiento llamará al mismo

método de cada uno. No debe haber código de la forma (if (tipo==a) ejecute x(); else ejecute y()). Una vez que se hayan procesado todas las estructuras, se actualiza el mapa para reflejar todos los cambios.

La simulación completa se ve entonces de la siguiente manera:

1. Se genera el mapa, incluyendo obstáculos, recursos y amenazas.
2. Se crea la lista de agentes. Los agentes pueden iniciar ya se todos en la base o en posiciones aleatorias en el mapa.
3. En cada paso ("tick") de la simulación[1]:
   a) Se recorre la estructura de agentes para determinar su siguiente acción, i.e. hacia donde se moverán, etc.
   b) Si un recurso o una amenaza desaparece, se crea uno nuevo para reponerlo.
   c) Se actualizan las posiciones de los agentes y objetos según corresponda.

[1] Cada tick se controla manualmente, por ejemplo con un botón "Siguiente".

Requerimientos

- Se debe programar usando el paradigma orientado a objetos.
- Se debe aprovechar al máximo la herencia y el polimorfismo, recuerde, no debe haber código de la forma (if (tipo==a) ejecute x(); else ejecute y()).
- Debe preparar y presentar un avance
- Debe implementar una interfaz gráfica
- Para el control de desarrollo se debe usar la pizarra de proyecto en Github y el sistema kanban para registrar el progreso de las actividades realizadas.

Entregables

1. Documentación PDF, usando una plantilla de Overleaf con una descripción general de la solución implementada y el diagrama de clases y una breve explicación de cada clase. Además deben incluir una sección donde detallen cómo se aplicó la herencia y polimorfismo, así como ventajas y desventajas que observaron durante el desarrollo del proyecto. Por último debe incluir una sección de referencias donde detalle los recursos externos utilizados para el desarrollo del proyecto, por ejemplo, tutoriales, ejemplos de código, videos, etc.
2. Código en un repositorio de Git. El último commit realizado debe ser antes de la fecha y hora límite de entrega.

Aspectos administrativos

1. Fecha de entrega:
2. El pdf debe ser enviado por medio de Teams
3. El proyecto debe ser desarrollado en Java.

Evaluación

Rubro	Valor
Documento PDF	(15)
- Descripción	2.5
- Diagrama de clases	5
- Herencia y polimorfismo	5
- Referencias	2.5
Código	(65)
- GUI	5
- Mapa	5
- Clase agentes	7.5
- Clase Objetos	7.5
- Uso de la Herencia	7.5
- Comportamiento e interacciones	10
- Simulación	
– Uso correcto de polimorfismo	7.5
– Estructura de datos	5
– Procesamiento	5
– Actualización	5
Avance	(10)
Workflow	(10)
- Uso de git	5
- uso del board de github	5
Total	100