Práctica 1: Búsquedas

Introducción

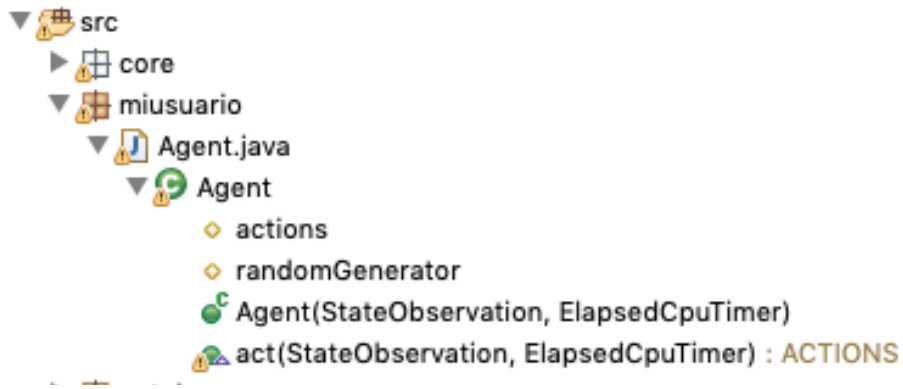
El alumno debe resolver un problema de agentes reactivos utilizando las estrategias de búsqueda informada A*.

Problema y entorno

El problema a resolver esta basado en la plataforma de juegos GVGIA (General Video Game AI Competition) cuya página principal se encuentra en http://www.gvgai.net. Se parte de una posición aleatoria de partida y el algoritmo debe explorar el espacio de búsqueda devolviendo la mejor solución obtenida. Para ello buscará el camino más corto entre la posición actual (que va cambiando a medida que avanza) y la posición de la meta.

Una vez instalado el SW desde github (https://github.com/GAIGResearch/GVGAI) se debe programar un agente reactivo para alcanzar la meta del juego de "cruzar la carretera", que corresponde al 43.

La práctica consistirá en la creación de un paquete que debe **OBLIGATORIAMENTE** llamarse como el **usuario de correo del alumno sin puntos**. Dentro de este paquete se debe implementar la clase Agent:



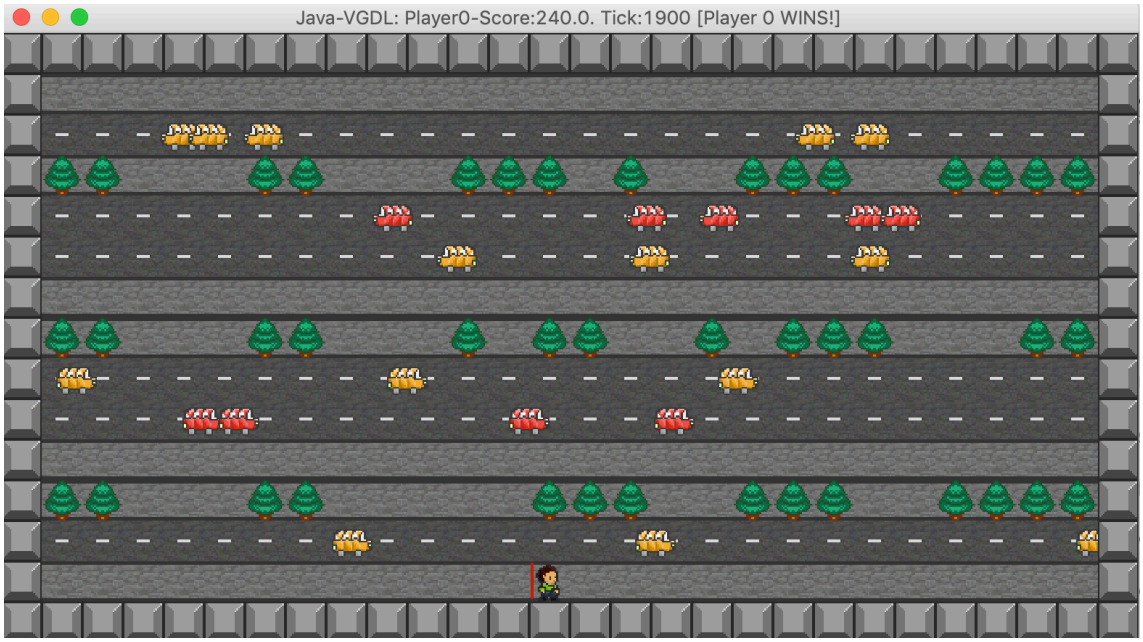

```
public class Agent extends AbstractPlayer {
    protected ArrayList<Types.ACTIONS> actions;
    public Agent(StateObservation so, ElapsedCpuTimer elapsedTimer)
    {
        //otras variables que puedas necesitar
        actions = so.getAvailableActions();
    }
 public Types.ACTIONS act(StateObservation stateObs, ElapsedCpuTimer elapsedTimer) {
//1) Percibir mediante el analisis de stateObs
//2) calcular camino mínimo
//3) actuar con la acción elegida según camino y otros (posición ticks, etc…)
       return actions.get(index);
    }
```

La observación del estado proporciona información sobre el estado:

- Paso de tiempo actual, puntaje, estado de victoria (victoria, pérdida o en curso) y puntos de salud del jugador.
- La lista de acciones disponibles para el jugador.
- Una lista de observaciones. A cada uno de los sprites del juego se le asigna un único tipo entero y una categoría atendiendo a su comportamiento (estáticos, no estáticos, personajes no jugadores, coleccionables y portales). La observación de un sprite, a menos que esté oculto, contiene información sobre su categoría, tipo de identificación y posición. Estas observaciones se proporcionan en una lista, agrupadas por categoría (primero) y tipo de sprite (segundo).
- Una cuadrícula de observación con todas las observaciones visibles en formato de cuadrícula. Aquí, cada observación se ubica en una matriz bidimensional que corresponde a las posiciones de los sprites en el nivel.
- Historia de eventos avatar. Un evento de avatar es una interacción entre el avatar (o un sprite producido por el avatar) y cualquier otro sprite en el juego. El historial de eventos de avatar se proporciona en una lista ordenada, ordenada por paso del juego, y cada instancia proporciona información sobre el lugar donde ocurrió la interacción, el tipo y la categoría de cada evento.

Al terminar el juego (1900 ticks) se muestra la puntuación en la parte de arriba de la pantalla y en la consola:

```
Result (1->win; 0->lose): Player0:1, Player0-Score:230.0, timesteps:1900
```



Estrategia a Implementar

Debe proponerse una estrategia de mapeo de los objetos móviles e inmutables y de optimización de la búsqueda que minimice el numero de veces que se ejecuta la búsqueda y el numero total de acciones que ejecuta el avatar para llegar a la meta.

La estrategia de búsqueda para encontrar el camino más corto es la búsqueda A* (proponer un heurístico según lo disponible en la literatura). Para determinar cuales son los nodos utilizables se debe procesar el mapa de observación proponiendo una estrategia sobre la meta, objetos movibles y no movibles. Esta "percepción inteligente", generará un mapa donde para una posición concreta de los objetos y el jugador tenga un conjunto de cuadriculas "pisables" por las que pueda planificar la búsqueda, y será el tablero/grafo/mapa que reciba el algoritmo de búsqueda.

Algunas consideraciones sobre las posibles estrategias de percepción:

- Los coches tienen una velocidad y una posición relativa al jugador, por lo que el mapeo debería tener en cuanta "donde estará el vehículo" , mas que donde está ahora.
- Los coches aparecen por la izquierda y derecha según el sentido del carril, no podemos saber que va a aparecer por un extremo, por lo que esa posición no es predecible, excepto "probabilísticamente".
- Si la búsqueda no encuentra solución, podemos esperar a un mejor momento, pero también puede ser que estemos marcando demasiadas casillas y no encuentre una forma de llegar a la meta.

En posteriores prácticas se modificará la forma de actuación, por lo que es conveniente que la estructura de observación y el conjunto de pasos resultado del camino sea una clase bien organizada según el paradigma de percepción , análisis y actuación.

Entrega

El objetivo de la práctica es que el alumno implemente la estrategia de percepción (mapeo siguiendo una estrategia), análisis (camino mínimo) y acción e agentes reactivos, por lo que deberá exponer en el documento la estructura general de estas tres características.

En los casos en los que se haya tomado alguna decisión para realizar el código que deba remarcarse (elección del heurístico, forma de implementar las estructuras generales, estructura del estado, operadores disponibles, etc…) se debe explicar brevemente tanto en el código (comentando las líneas que realizan cada parte) como en el texto (no incluyendo código, sólo breve explicación de las estructuras y clases).
Para dar por válida esta práctica debe deben obtener al menos 190 puntos , resultado de haber ejecutado sobre el número total de tics todas las ejecuciones .
Se valorará la claridad de implementación separando las funciones del agente y la explicación especifica del algoritmo A. Se puede utilizar código de librerías para el A* preexistentes o utilizar código propio , pero en ambos casos deberá explicarse la configuración del problema como un problema de búsqueda A*: heurístico, estrategia de repetidos, generación sucesores, definición de estado, acciones disponibles, origen y validación de meta alcanzada)

El texto debe ocupar entre 1 y 4 páginas con al menos los siguientes apartados:

Desarrollo: Descripción de las estructuras, estados, operadores, etc…

Análisis: Detalle del comportamiento basado en la propuesta de mapeo , percepción y actuación.

El alumno debe entregar en Moodle el código fuente y el documento explicativo en un único fichero .zip.

Asimismo se debe utilizar la herramienta https://gitlab.uhu.es, para entregar el paquete desarrollado dentro de un grupo con el nombre SI_2020, donde habrá creado un subgrupo con el nombre SI_2020_nommbredeusuario **sin puntos**. En este subgrupo deberéis crear un repositorio con el nombre **exacto** *practica1,* donde debe estar todo lo necesario para que el agente funcione.

Con posterioridad se comparará el funcionamiento de los agentes en una herramienta externa que permitirá medir el comportamiento comparado de todas las entregas de los alumnos entre si mostrando los puntos totales alcanzados por cada uno.

Notas:

- La fecha de entrega se entiende como fecha límite. Se puede entregar con anterioridad a esa fecha y la corrección de errores y re-entrega en plazo no acarreará consecuencias en las calificaciones.
- Se penalizarán las prácticas que no tengan su código **comentado** ni **tabulado**.
- La **detección de copia de código específico** de otros compañeros o de otras fuentes será motivo de suspensión de las prácticas. Sin embargo se promueve que se reutilice código de uso genérico (de fuentes externas) con la obligación de citar la fuente. Por ejemplo si se quiere utilizar la biblioteca de Inteligencia artificial AIMA, o se quiere utilizar el código de una práctica de gestión de grafos del año pasado o una librería genérica de grafos o gestión de colas. De otra forma puede detectarse que un porcentaje de la práctica es parecido al de otra práctica y ser rechadazas cuando se trata de un uso lícito de código no específico.

Fecha Límite de entrega: 23 de Marzo.