Integrantes:

Rodrigo Martín Díaz Figueroa – 97400

[rodrigomdiaz@live.com](mailto:rodrigomdiaz@live.com)

Federico Germán Agra – 94186

[federico.agra@gmail.com](mailto:federico.agra@gmail.com)

DESARROLLO DE IA PARA AGENTES EN UN ENTORNO VIRTUAL

2do Cuatrimestre – Año 2017

Inteligencia Artificial

Proyecto de Cursado – Etapa 2

Índice

|  |  |
| --- | --- |
| Introducción | 2 |
| Descripción de la Etapa | 2 |
| Correcciones de Etapas Anteriores | 2 |
| Flujo de Ejecución del Agente |  |
| Actualización de Creencias |  |
| Percepción del Entorno |  |
| Mantenimiento de Base de Creencias |  |
| Olvido |  |
| Recuerdo |  |
| Deliberación de Intensiones |  |
| Planeamiento y Ejecución de Acciones |  |
| Realización de Acciones |  |
| Razonamiento y Estrategias |  |
| Especificación de Deseos |  |
| Prioridad Normal |  |
| Prioridad Alta |  |
| Selección de Deseos |  |
| Verificación de cumplimiento de intensiones |  |
| Acciones |  |
| Primitivas |  |
| De Alto Nivel |  |
| Planificación de Acciones de Alto Nivel |  |
|  |  |
|  |  |

introducción

Descripción de la Etapa

La segunda etapa del proyecto consiste en el diseño e implementación en lenguaje Prolog de un agente inteligente basado en una arquitectura BDI *(Beliefs, Desires, Intentions).* En esta etapa se dotó al agente implementado en la etapa anterior de comportamiento racional considerable, respetando el modelo de agente y la arquitectura BDI. Para ello se llevaron a cabo las siguientes tareas:

- Se definieron los deseos considerados por el agente.

- Se implementó la obtención de todos los deseos actuales y deseos de alta prioridad en base a las creencias del agente.

- Se implementó la selección de una intención a partir del conjunto de todos los deseos actuales.

- Se implementó, para cada intención, la verificación de su cumplimiento.

- Se especificaron acciones del agente (siguiendo la representación Strips) para posibilitar la proyección de planes y detectar así su factibilidad.

- Se estableció e implementó el método de planeamiento para cada tipo de acción de alto nivel.

Correcciones de Etapas Anteriores

Actualización de Creencias

Se re-estructuró el modo de actualización de creencias, ya que la implementada en la etapa anterior presentaba inconsistencias en el modo de mantenimiento de los recuerdos y creencias del agente, generando información residuo que imposibilitaba el correcto funcionamiento del agente.

Método de Búsqueda A\*

Flujo de Ejecución del agente

El agente comienza a ejecutarse una vez que se conecta a la aplicación que gestiona el juego. En ejecución el agente mantendrá activa su rutina de ejecución que consta de 4 partes principales bien definidas:

* Actualización de Creencias.
* Deliberación de Intensiones
* Planeamiento y Ejecución de Acciones
* Realización de Acciones.

Actualización de Creencias

## Percepción del Entorno

Se obtiene la percepción del mundo en el instante de tiempo actual, la cual representa una porción del entorno y generada por la propia aplicación. Se identifican en la percepción los siguientes elementos:

* time(T): representa el tiempo de juego en un momento dado, donde T es el tiempo actual.
* node(Id,Pos,Connections): Representa una posición en el mapa del juego, donde Id es el identificador del nodo, Pos es un vector [X,Y,Z] en el espacio y Connections es una lista [[NodoAdy1,Costo1],…,[NodoAdyn,Coston]].

- NodoAdy: Identifica un nodo conectado al nodo Id.

- Costo: Representa el costo de moverse desde el nodo Id hasta NodoAdy.

* at(Entity,Node): Indica que una entidad se encuentra posicionada en un nodo, donde Entity es un identificador de entidad y Node es el identificador del nodo donde se encuentra.
* atPos/2, atPos(Entity,Pos): Indica que una entidad se encuentra sobre el vector posición, donde Entity es un identificador de entidad y Pos es un vector [X,Y,Z] en el espacio.
* has(Entity1, Entity2): Indica que una entidad lleva consigo otra entidad, donde

Entity1 es el identificador de la entidad portadora y Entity2 es el identificador de la entidad que está siendo portada.

* entity\_descr(Entity, Descr): Representa una descripción con ciertas características de la entidad que se consideran visibles para el agente, donde Entity es el identificador de una entidad y Descr es una lista de la forma [[Feature1,Val2],[Feature2,Val2],…,[Featuren,Valn]].

- Feature: Nombre de una característica visible de la entidad.

- Val: Su valor asociado.

- Entity: es una entidad del juego de la forma [<entidad>, <nombre>].

Las entidades del juego reconocidas son:

* inn (Posada): estructura utilizada para que el agente se sitúe dentro y regenere su vida hasta el máximo conforme pasa el tiempo.
* agent (Agente): identifica a un agente dentro del juego. Recolecta objetos y ataca a otros agentes.
* gold (Oro): objeto que puede ser recolectado por un agente. No contiene ninguna utilidad.
* grave (Tumba): estructura que almacena oro.
* potion (Poción): objeto que puede ser recolectado por un agente. Sirve para abrir tumbas y dormir a otros agentes.

## Mantenimiento de Base de Creencias

Una vez obtenida la percepción actual, se procede a modificar la Base de Creencias del agente. Esta tarea es desarrollada en dos partes: Una parte de *Olvido* y otra de *Recuerdo* de los datos provistos por la percepción. La estrategia de actualización de creencias se basa en borrar, en un principio, toda información alojada en las creencias de la que se ha obtenido una nueva versión en la percepción recibida (*olvido*), para luego recordar todo lo percibido (*recuerdo*).

## Olvido

La parte de olvido de información obsoleta o desactualizada es la más compleja, ya que se descompone en varios ciclos, en los que en cada uno se eliminan datos recordados según un tipo determinado de relación obtenida en la percepción. Estos tipos de relaciones son:

* Por cada nodo percibido que ya había visto antes, olvida el recuerdo del nodo visto. Se eliminan todos los *node* con id igual al de los nodos percibidos que aparezcan en la base de creencias.
* Por cada nodo percibido que ya había visto algo en su ubicación y en su posición (vector), olvida lo que había visto en esa ubicación y lo que había visto en esa posición (vector). Se eliminan todos los *at* y *atPos* con id y posición (vector) igual al de los nodos percibidos que aparezcan en la base de creencias.
* Por cada descripción percibida de una entidad que ya la había recordado antes, olvida su descripción anterior. Se eliminan todas las *entity\_descr* que aparezcan en la base de creencias con Entidad igual a la descripción percibida.
* Por cada entidad en el piso percibida que estaba en posesión de otra entidad, se olvida de esa posesión y de cualquier posición (vector) anterior de esa entidad. Por cada *at(E,\_)*, si existe en la base de creencias un *has(\_,E),* se eliminan todos los *has(\_,E)* y *atPos* de esa entidad E.
* Por cada entidad en el piso percibida que estaba en otro lugar del piso, se olvida de la ubicación anterior y de la posición (vector) anterior. Por cada relación *at* (de una entidad) percibido que aparece en la base de creencias, se eliminan todos esos *at* y su correspondiente *atPos.*
* Por cada entidad percibida en una posición (vector) que estaba en otra posición (vector), se olvida de la ubicación anterior y de la posición (vector) anterior. Por cada relación *atPos* (de una entidad) percibido que aparece en la base de creencias, se eliminan todos esos *atPos* y su correspondiente *at.*
* Por cada entidad percibida que creía que estaba en el piso y en otra posición (vector) pero en la percepción es poseída por otra entidad, olvida que estaba en el piso y su posición (vector) anterior. Por cada *has(\_,E)* encontrado tal que existe en la base de creencias un *at(E,\_)* y un *atPos(E,\_)*, se eliminan todos los esos *at* y *atPos.*
* Por cada una entidad percibida que ya había visto antes la olvida para actualizar su posesión. Por cada *has(E,\_)* percibido que existe en la base de creencias, elimina esos *has(E,\_).*
* Elimina todos los tiempos (*time*) recordados.

## Recuerdo

En esta parte lo único que hace es recordar todo lo percibido. Por cada relación obtenida de la percepción, agrega dicha relación.

Deliberación de Intensiones

…

Planeamiento y Ejecución de Acciones

…

Realización de Acciones

…

Razonamiento y Estrategias

Para que el agente inteligente presente un comportamiento basado en la arquitectura BDI se definieron varias funcionalidades propias de la arquitectura.

Especificación de Deseos

Entre los deseos que un agente puede considerar, se diferencian dos tipos de deseos.

## Prioridad Normal

Son aquellas acciones que el agente puede realizar en un momento determinado. Si no hay una acción de alta prioridad ejecutándose, se selecciona, según alguna condición, un deseo de este tipo para continuar. Estos deseos están implementados mediante el predicado *desire(-Desire,-Explanation),* donde *Desire* es un deseo (actualmente activo de acuerdo a las creencias) y *Explanation* es una explicación que especifica las razones por las cuales *Desire* se considera actualmente un deseo. Estos deseos son los siguientes:

* Obtener Poción: *get([potion, <Nombre>]),* si recuerda que una poción se encuentra en el piso, tener esa tumba es una meta.
* Obtener Tesoro dentro de una Tumba: *get([gold, <Nombre>]),* si recuerda que un tesoro se encuentra dentro de una tumba, tener ese tesoro es una meta.
* Obtener un Tesoro tirado en el piso: *get([gold, <Nombre>]),* si recuerda que un tesoro se encuentra en el piso, tener ese tesoro es una meta.
* Descansar: *rest*, se activa este deseo cuando su vida es inferior a 100 pts. Su meta es llegar a una posada para recuperar puntos de vida.
* Depositar Tesoros: *depositarTesoro([gold,<Nombre>]),* su meta es llegar a su *Home* para depositar un tesoro obtenido.
* Defender el *Home: ir\_a\_casa,* su meta es llegar a su *Home* para defenderlo de posibles ataques de enemigos.
* Mover aleatoriamente: *move\_at\_random,* su meta es moverse aleatoriamente para mantenerse en movimiento siempre.

## Prioridad Alta

Son aquellas acciones de suma urgencia que se desean realizar, a tal punto de interrumpir la acción que se está ejecutando si es que ésta no es de prioridad alta. Estos son modelados por el predicado *high\_priority(-HPDesire, -Explanation),* el cual determina si existe un deseo *HPDesire* de alta prioridad, es decir, tal que implica abandonar la intención actual. En caso de existir tal deseo de alta prioridad, lo retorna junto a una explicación que especifica las razones por las cuales *HPDesire* se considera un deseo que debe adoptarse inmediatamente como intención. Estos deseos son los siguientes:

* Descansar: *rest,* Este deseo se convierte en prioridad cuando la vida del agente no supera los 50 pts. Su meta es llegar a la posada más cercana (que no tenga prohibida su entrada) para recuperar puntos de vida.

Selección de Deseos

En esta etapa se determina, según el contexto en el que el agente se encuentra, que deseo elegir como próxima acción a realizar. Esto se modela mediante el predicado *select\_intention(-Intention, -Explanation, +CurrentDesires),* el cual selecciona uno de los deseos corrientes del agente como intención.