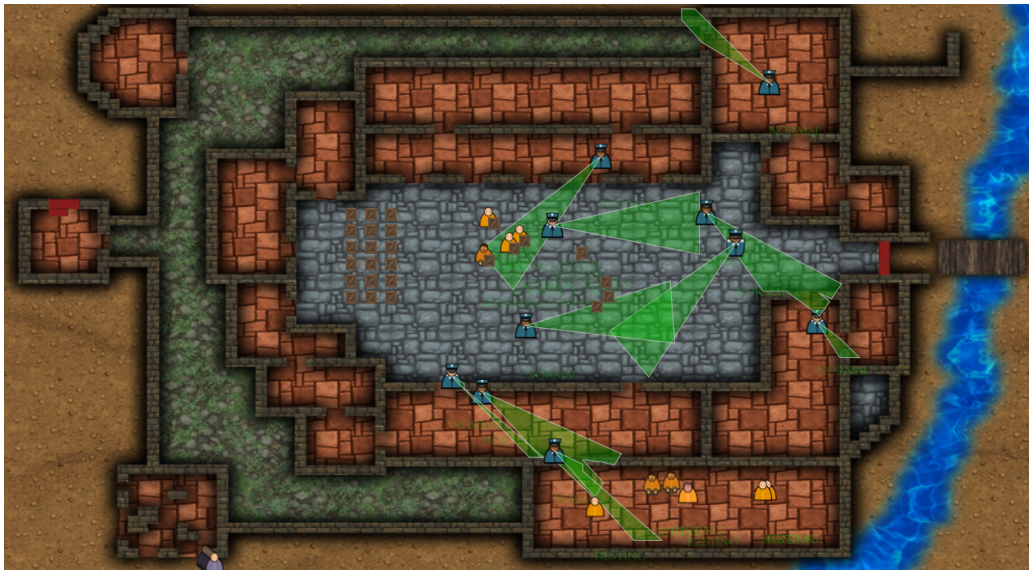
IA 2017 – Prison Escape



Marcos Vázquez Rey @ IA 2017

Índice

1. Descripción del proyecto

Esta demo implementa una serie de agentes que actuan de forma independiente e interactúan unos con otros. El cuerpo de éstos agentes viene representado por diferentes sprites provenientes del videojuego *Prison Architect.*

Se implementan varios tipos de agentes con objetivos y comportamientos diversos, así como capacidades para interactuar con el entorno. Éstos se detallarán en profundidad en el siguiente apartado.

Se ha implementado un límite al tiempo de cómputo de la IA por frame, de modo que si el cálculo de la IA se hace demasiado pesado, o no resta tiempo en un frame debido a otros cáculos previos, el programa guardará la posición del último agente actualizado y continuará actualizándolos desde ahí en el siguiente frame.

Para una documentación más exhaustiva sobre la implementación de los algoritmos utilizados puede consultarse el código fuente. La descripción de las funciones y atributos se encontrará comunmente en las cabeceras, y los detalles de implementación concreto en las propias implementaciones de las clases y métodos.

2. Agentes

Los agentes implementados en esta demo muestran patrones de movimiento y comportamiento basándose en máquinas de estado finitas. Éstos se mueven utilizando algortimos de búsqueda de caminos (A*), así como otros patrones de movimiento determinista y *tracking*.

Cada uno de ellos posee un conjunto de diferentes estados y unas condiciones específicas para pasar de uno a otro.

El movimiento a través del mapa se realiza combinando el algoritmo A* con una serie de *waypoints* para trayectos comunes.

2.1 Guardia 

El guardia patrulla las zonas interiores de la prisión utilizando *waypoints* para trayectos comunes y A* para distancias más largas, complejas o no cacheadas.

El guardia posee un cono de visión que colisiona con las partes intransitables definidas en el mapa de costes, impidiéndole ver a través de los muros, y le permite detectar puertas y agentes de otros tipos (prisioneros y soldados).

En estado normal el guardia patrulla entre las zonas interiores de la prisión, y es capaz de detectar puertas abiertas y soldados dentro de su cono de visión.

Si el guardia ve un soldado, pasará a modo sospecha y le perseguirá indefinidamente utilizando *tracking*. El hecho de que el cono de visión no atraviese muros permite que el guardia no atraviese paredes al intentar perseguir un posible enemigo al otro lado del muro.

Si el guardia detecta una puerta abierta, entra en estado de alerta, se dirige hacia dicha puerta y da la alarma. Nótese que una puerta se cerrará cuando el guardia esté a una distancia menor que su radio de acción, y siempre que no haya soldados cerca.

Si se ha dado la alarma, todos los guardias pasarán inmediatamente a estado de alerta y permancerán en este estado hasta que la alarma se apague. La alarma se apagará cuando ningún guardia haya avistado una puerta abierta o un soldado en un tiempo mayor que T.

En estado de alerta, el guardia patrullará con normalidad la prisión y perseguirá a cualquier soldado o prisionero indefinidamente, hasta que la alarma se apague o éste haya huído por la parte inferior del mapa.

FSM

Lista de transiciones de estado:

Normal – Suspect

El guardia ha visto a un soldado.

Suspect – Normal

El guardia no ha visto a ningún soldado en T tiempo.

Normal – Alert

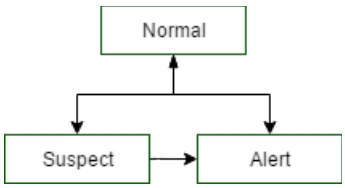
El guardia ha cerrado una puerta.

Alert – Normal

El guardia no ha visto a ningún soldado ni puerta abierta en T tiempo.

Suspect – Alert

El guardia ha cerrado una puerta mientras perseguía un soldado.



2.2 Prisionero 

El prisionero realiza una serie de movimientos cíclicos entre la zona de trabajo y la de descanso. Para éstos se utilizan en su mayoría *waypoints* y movimientos deterministas de muy corta longitud.

Los prisioneros transportan cajas de la zona de carga a la de descarga. Éstas cajas son entidades propias y persistentes en la demo.

Al principio de la demo, al prisionero se le asigna un turno de descanso o trabajo según si su turno asignado coincide con el actual en la prisión.

El prisionero se mueve siguiendo *waypoints* preestablecidos hacia su zona asignada, y al llegar cambia de estado.

Un prisionero descansando pasea dentro de la estancia de descanso, moviéndose a puntos fijos aleatorios dentro de dicha estancia (asignado como movimiento determinista con un solo punto).

Un prisionero trabajando pasa de *Loaded* a *Unloaded* y viceversa según se mueve de la zona de carga a la de descarga y transporta cajas en el proceso.

Cuando el turno de trabajo cambie, los prisioneros descansando o yendo a descansar, pasarán a estar yendo a trabajar. Del mismo modo, los prisoneros trabajando o yendo a trabajar pasarán a estar yendo a descansar.

Si se ha dado la alarma en la prisión, el prisionero pasará de cualquiera de los otros estados a *Escaping*. En este estado irán de una puerta a otra hasta que encuentren una abierta. Si ésto ocurre,

se calculará un camino hasta la base de huída situada en la parte inferior del mapa, y desaparecerá al llegar allí.

Si un prisionero se encuentra escapando y la alarma se apaga, éste terminará su ruta de movimiento actual y después volverá al trabajo o descanso según su turno de trabajo asignado.

FSM

Lista de transiciones de estado:

Idle – GoingToRest/GoingToWork

Establecidos al inicio de la simulación según el turno de trabajo actual corresponda al prisionero. Aquellos prisioneros cuyo turno asignado sea el activo se dirigirán a la zona de trabajo, mientras que el resto lo harán hacia la zona de descanso.

GoingToRest – Resting

El prisionero ha llegado a la zona de descanso.

GoingToRest/Resting – GoingToWork

Se ha activado el turno de trabajo correspondiente al prisionero.

GoingToWork – WorkingUnloaded

El prisionero a llegado a la zona de carga.

WorkingUnloaded - WorkingLoaded

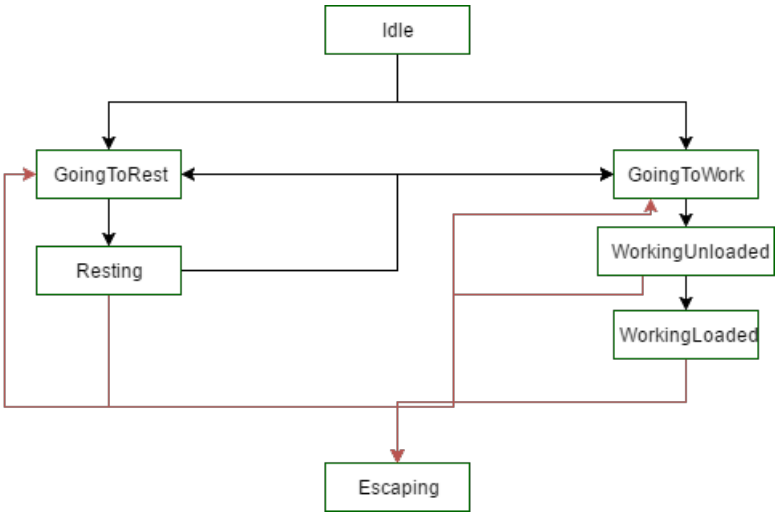
El prisionero ha llegado a la zona de descarga.

ANY – Escaping

Se ha dado la alarma en la prisión. Este estado puede ser alcanzado desde cualquier otro.

Escaping – GoingToRest/GoingToWork

Se ha apagado la alarma en la base y el prisionero ha terminado su ruta de movimiento actual (hacia una puerta o hacia la huída).



2.3 Soldado 

Los soldados parten de la base inferior izquierda del mapa e inmediatamente se dirigen a la puerta derecha de la prisión.

Los soldados abrirán cualquier puerta que se encuentre dentro de su radio de acción, y dicha puerta quedará bloqueada durante un breve periodo de tiempo (ningún guardia podrá cerrarla).

En estado normal, el soldado va de una puerta de la prisión a otra alternativamente, abriéndolas al llegar a ellas.

Si se ha dado la alarma en la base, el soldado terminará su ruta de movimiento actual y después huirá dirigiéndose a la base inferior del mapa.

FSM

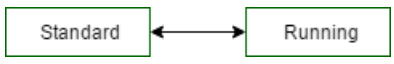
Lista de transiciones de estado:

Standard – Running

Se ha dado la alarma en la base.

Running – Standard

Se ha apagado la alarma y el soldado ha terminado su ruta de movimiento actual.



2.4 Pathfinder

El agente *Pathfinder* posee una cola de peticiones de cálculo A*, que el resto de los agentes rellenan y éste calcula en orden estricto de llegada y un máximo de una por frame.

Este agente debe actualizarse en primer lugar, antes que el resto de agentes. Dado que realiza operaciones costosas, ésto permite una mejor estimación del tiempo disponible en el frame y el número restante de agentes que pueden ser actualizados en el mismo.

2.5 Messager

El sistema de mensajería entre agentes no se ha implementado en esta demo, se incluye un agente gestor de los mensajes, al que el resto de los agentes envíarían mensajes para que este los redistribuya.

3. Pathfinding

Esta demo utiliza diversos sistemas de pathfinding, siendo el más potente y preciso el A*, pero también y por mucho el más caro computacionalmente.

Varias clases están involucradas en reducir el coste de los algoritmos de búsqueda de caminos o especificar las secciones transitables y costes asociados al movimiento de los agentes.

Muchas de estas mejoras son específicas del mapa en el que se desarrolla la demo y están especificadas manualmente en el código.

3.1 A*

El algoritmo A* calcula un camino dado un punto inicial y final sobre un mapa de costes.

3.2 Mapa de costes

El mapa de costes se carga desde una imagen de diferentes resoluciones y especifica las secciones transitables e intransitables. También puede ser usado para especificar el coste de movimiento de diferentes zonas.

3.3 Camino

El camino es un simple vector de puntos que indica el movimiento determinista a seguir para llegar del punto A al punto B.

3.4 PrisonMap

Ésta clase contiene optimizaciones de movimiento específicas del presente mapa. Sus valores deberían ser adaptados para utilizarlo con un diseño diferente de prisión.

Habitaciones

Se han guardado diversas habitaciones con sus respectivas esquinas. Ésto es útil cuando se quiere calcular caminos entre habitaciones sin importar el punto concreto de la habitación destino.

También se utiliza para hallar si un punto se encuentra dentro de una habitación.

Puertas

Listado de puertas presentes en la prisión.

Waypoints

Puntos intermedios para facilitar movimientos comunes. Se establecen diversas rutas precalculadas entre pares de habitaciones.

4. Otras clases

Algunas otras clases incluídas a tener en consideración que no caen bajo categorías anteriores.

4.1 Puertas

Las puertas actúan como entidades independientes situadas en el escenario que modifican el mapa de costes en su posición correspondiente.

Una puerta puede ser abierta por un soldado, permitiendo el cálculo de caminos a través de su posición. Cuando un soldado abre una puerta, ésta no puede ser cerrada en un tiempo menor que T.

De igual modo, una puerta puede ser cerrada por un guardia, marcando su posición como intransitable en el mapa de costes.

4.2 GameStatus

Ésta clase singleton contiene variables globales que deben ser accedidas desde diversos puntos del programa, así como otras referentes a la ejecución general del mismo y que no caen bajo ninguna categoría concreta (e.g. tiempo de simulación transcurrido).