MEMORIA PRÁCTICA 2

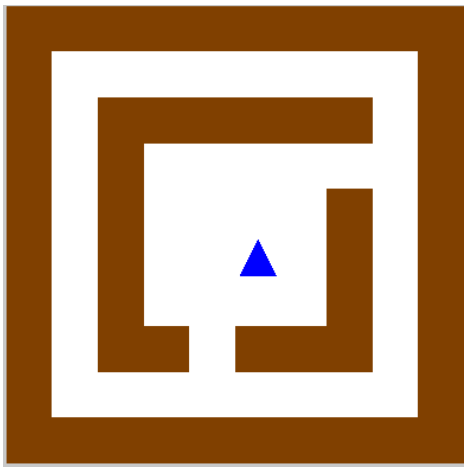
Moreno Picamán Bryan

INDICE

Análisis del problema.

Entorno.-

Al comenzar el análisis del problema lo primero que observamos es el entorno, un espacio cerrado dividido en "cuadrados", los cuales también podrán ser posiciones. Concretamente el tamaño es 10x10, siendo los cuadros de los lados siempre inaccesibles para nuestro robot aspirador, los restantes podrán tener dos estados, "accesible" u "obstáculo".



Agente.-

Se dispone de un agente, o aspirador, el cual será el encargado de la realización de las actividades de esta práctica, que son entre otras, moverse, y aspirar suciedad. Nuestro agente cuenta con una serie de características y funciones las cuales podrán ser usadas para llevar a cabo el desempeño de su función. Estas son las siguientes:

- Sensores:
  - Sensor de suciedad: Indica si la casilla actual tiene suciedad, pero no indica cuanta.
  - Sensor de choque: Ubicado en la punta del agente (el cual tiene forma triangular) indica si este ha chocado.
- Actuadores:
  - actFORWARD: Esta función permite que el robot avance en la dirección en la que apunta.
  - actTURN_L: Esta función hace girar al agente hacia la izquierda 90º.
  - actTURN_R: Girar a la derecha 90º.
  - actSUCK: Activar succión (limpia una unidad de suciedad).
  - actIDLE: No hacer nada.

Con estos sensores y actuadores se podrán realizar las funciones necesarias para la práctica, que consisten en recorrer el entorno y limpiar la suciedad, consiguiendo que la media de 10 recorridos completos sea la menor posible.

Requisitos a cumplir.-

Una vez tenemos toda la información del agente y el entorno y considerando este último aleatorio, se necesita que el agente cumpla una serie de objetivos:

1. Ser capaz de recorrer la habitación completa limpiando la suciedad a su paso
2. Ser capaz de acordarse donde hay obstáculos y no chocar con ellos más de 1 vez.
3. Priorizar las zonas menos visitadas y elegir un camino adecuado para que todas las partes del entorno sean visitadas y limpiadas.

Estos objetivos a su vez conllevan una serie de problemas asociados:

- Recorrer la habitación completa.- Esto sería posible realizarlo por medio de movimientos aleatorios de nuestro agente, pero no aseguraría que se visitaran todas las posiciones.
- Recordar donde hay obstáculos y no chocar repetidamente con ellos.- El problema de nuestro agente es que no dispone de una representación del mundo con la cual poder saber si hay obstáculos
- Priorizar y elegir caminos.- Ya que el sensor de suciedad solo lo tiene nuestro agente, priorizar las zonas menos visitadas por la suciedad que estas contienen es imposible.

Soluciones.-

Para poder llevar a cabo los objetivos de forma correcta, se necesitan de una serie de características que serán implementadas como parte de la solución en nuestro agente:

- Mapa.- Sistema de representación que se ira rellenando conforme el agente recorra el mapa, apuntando caminos, y obstáculos. Para esta misión se usará una matriz de enteros con distintos valores que indicaran que hay en cada posición.
- Ubicación.- Se necesita de un sistema que sitúe al agente y sepa los cambios en su posición cuando este realice acciones. Para esto se hará uso por un lado de dos variables [x] [y] que indicaran la posición del agente con respecto a la inicial, y por otro un sistema de posicionamiento, el cual indicara si se encuentra apuntando al norte, sur, este u oeste, este sistema será calibrado según la posición del agente al inicio de cada recorrido.
- Selector de camino.- Sera el encargado de decir a nuestro agente hacia donde ha de moverse en cada uno de los momentos.
- Sistema de priorización.- Para que nuestro agente priorice las zonas menos visitadas, cada una de las casillas de la matriz tendrá varios valores posibles que ayudaran a elegir el siguiente destino.
    - -1 - Pared/obstáculo
    - 0 - Estado inicial
    - 2 - Visitado en la última acción
    - 2+ - Acciones pasadas desde que se visitó por última vez.

Descripción de la solución planteada.

Para empezar a plantear una solución del problema se necesita primero disponer de las herramientas de posicionamiento, mapeado y cambio de estados.

Mapeado.- Este proceso se realiza por medio del uso de una matriz de tamaño 20 x 20, además de dos variables de posición x e y, que comenzaran en la posición x10y10, esta posición se realiza para que exista libertad de representación de forma independiente a la ubicación inicial de el agente.

Posicionamiento y cambio de estados.- Para esto disponemos principalmente de 4 herramientas de representación: *aumenta, disminuye, cambiaposicion, cambiaposicionr y posición.*

- *Aumenta: Cambia el valor de las variables de posición x e y dependiendo de la posición en la que se encuentre el agente, a una posición más (simula un fordward).*
- *Disminuye: Es la función complementaria a la anterior.*
- *Cambiaposicion: En este caso cambia representación interna de la posición del agente (simula a girar a la izquierda).*
- *Cambiaposicionr: Es la función complementaria a la anterior*
- *Posición: Es una variable de 4 valores, de 0 a 3, siendo 0 "posición inicial".*

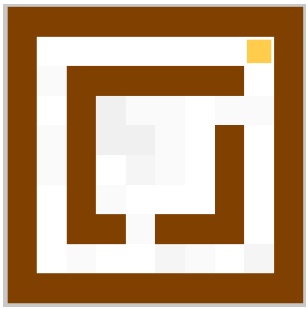
Una vez se dispone de un mapa, una posición inicial y un sistema que indica hacia qué posición está dirigido el agente, se definen las normas de funcionamiento.

Para hablar de la solución, nos centraremos en las reglas que se describen en el Agent.cpp de la práctica, siendo una de estas reglas aplicable siempre, sea cual sea el caso, esta regla es la denominada MasValor(), que aumenta en cada iteración el valor de todos los elementos del mapa en -1 (menos los ya marcados como obstáculo). Las normas se dividen en:

- Si choca
    - Marca la posición del mapa como obstáculo.
    - Cambia la representación que tenemos de la posición actual.
    - Marca la nueva posición (anterior al choque) como limpia.
    - Si está sucio.
        - Limpia y Marca posición como limpia
    - Si no.
        - Elige nueva posición
        - Cambia orientación
        - Marca que existe un camino a seguir
- Si no
    - Si esta socio.
        - Limpia y Marca posición como limpia
    - Si no.
        - Marca mapa como limpio.
        - Si "cambio" es verdadero.
            - Termina de hacer un giro y pone cambio a "false"
        - Si tenemos camino y no hay obstáculo.
            - Cambia su posición y se mueve
        - Si todas las posiciones tienen obstáculo.
            - Gira y pone cambio a "true"
        - En otro caso
            - Elige nueva posición y cambia la orientación
            - Marca que existe un camino a seguir
- Aumenta el valor de la matriz con Agent::MasValor()

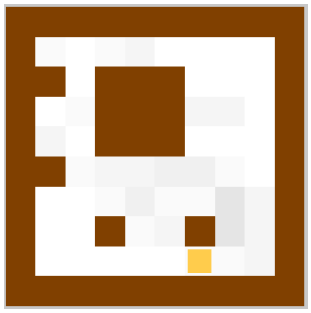
Resultados.

Agent.map



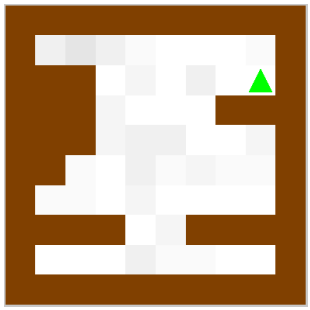
Average dirty degree: 65.064,900

Mapa1.map



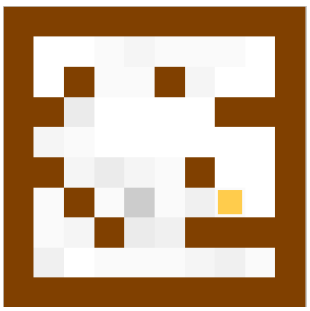
Average dirty degree: 106.013,500

Mapa2.map



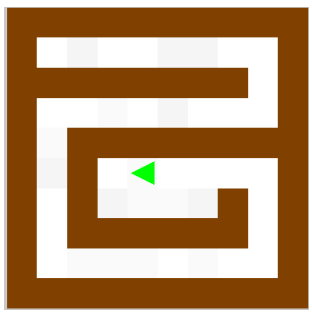
Average dirty degree: 95.683,700

Mapa3.map

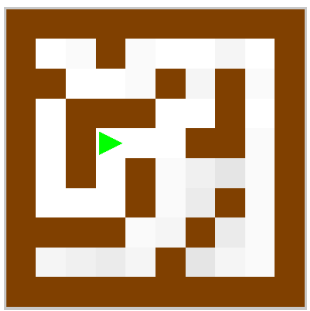


Average dirty degree: 123.580,500

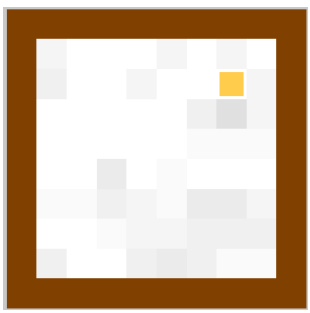
Mapas Extra.-



ADD: 58.788,800



ADD: 84.076,400



ADD: 163.604,700