

Assignment 2 - RMI

Diogo Mendes (88801)
Raquel Pinto (92948)

A dark blue diagonal gradient bar that starts from the bottom left corner and extends towards the top right corner, covering the lower half of the slide.

Recap Challenge 2

Visualizamos o mapa como se estivesse dividido em quadrados.

O objetivo é o agente estar no centro de um quadrado para mapear as 4 direções.

Desenvolveu-se uma classe vetor.

Criamos um vetor “next” e trabalhamos com ele como o objetivo.

- Máquina de estados: GA, RL, RR, INV, END
 - Se TargetReached() então chama mappingDecode (escolher novo objetivo);
 - Se “caminho” não estiver vazio então chama a runCaminho();
 - Caso nenhuma das situações de cima se confirmem então faz as funções de movimento;
 - Estado END - writeMap(), termina simulador e exit do java.
- Função Estados que define o estado consoante os valores do compass e do compass_goal.

Recap Challenge 2 – mappingDecode()

```
void mappingDecode() {  
    //guardar o valor das coordenadas atuais na lista coordsAntigas  
    coordsAntigas.add(vetor atual);  
  
    if(target){  
        //inserir o target na posicao do ground do array  
        objetivos[ground] = vetor atual;  
    }  
    //mapear a vizinhança com os sensores  
    coord[x vetor atual][y vetor atual] = "X" || "|" || "-"; //conforme a situação  
  
    //verificar posicoes visitaveis e adicionais  
    //lista com todas as posicoes visitaveis que ainda nao foram visitadas  
    visitaveis.add(vetorposicaovizitavel);  
    //lista so com as coordenadas possiveis a volta do robo  
    localViz.add(vetorposicaovizitavel);  
  
    if(localViz.isEmpty()){ //se o robo estiver num beco  
        aStar(); //calcular o melhor caminho para a proxima posicao visitavel  
        runCaminho(); //correr o caminho dado pelo aStar()  
    }  
    atualizar a proxima posicao a ser atingida pelo robo  
    atualizar o compass_goal do robo  
    writeMap();  
}
```

Recap Challenge 2 – setCaminho() e runCaminho()

setCaminho()

- Escolhe as próximas coordenadas do next como objetivo (visitaveis.next());
- Usando a posição atual como partida;
- Utiliza o A* para calcular o caminho do objetivo para a partida.

runCaminho()

- define o primeiro vetor da LinkedList “caminho” como o next;
- Ajusta o compass_goal para este next;
- E vai percorrendo esse caminho.

Recap Challenge 2 – A*

Código fonte retirado de: <https://stackabuse.com/graphs-in-java-a-star-algorithm/>

Principais adaptações:

- Adição de um valor vetor aos nós;
- Reformulação de algumas partes para aceitarem os novos nós;
- Adição de campo “filhos” à classe vetor para guardar os vizinhos com movimentos possíveis.

Challenge 2 – Outras funções

- WriteMap() - Escreve o String [][] num ficheiro txt;
- AddToMap() - Adiciona ao String[][];
- OnMap() - verificar se coordenadas já foram preenchidas com algo diferente de " ";
- FillMap() - encher mapa com " ";
- Funções para obter coordenadas nas 4 direções, para k+1 e k+2;
- Funções para arredondar a bússola e a bússola objetivo;
- Verificar se é parede nas 4 direções.

Recap Challenge 3

- Em vez de ter a função WriteMap temos a WritePath que escreve o path obtido num txt;
- Adição de um array vetor[getNumBeacons()] para guardar as coordenadas dos targets;
- Adição de uma função para calcular o caminho para os targets - usa o A* para calcular o caminho do (0->1->2->...->0), soma tudo na mesma LinkedList para ser escrito.

Challenge 4

- Utilizamos a função WriteMap e a WritePath;
- Novo estado: Collision - para corrigir o posicionamento quando ocorre uma colisão;
- Novo estado: Wait - para verificar quando começa a execução da simulação;
- Máquina de estados com 7 estados.

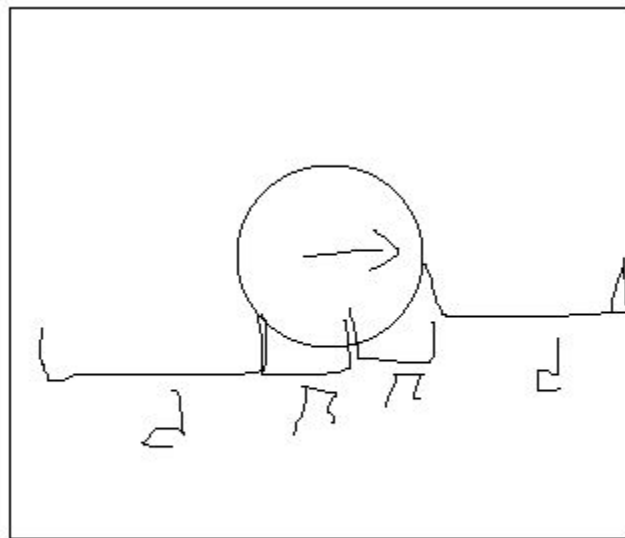
$$out_t = \frac{in_i + out_{t-1}}{2}$$

$$x_t = x_{t-1} + lin * cos(\theta_{t-1})$$

$$y_t = y_{t-1} + lin * sen(\theta_{t-1})$$

$$lin = \frac{out_t^l + out_t^r}{2}$$

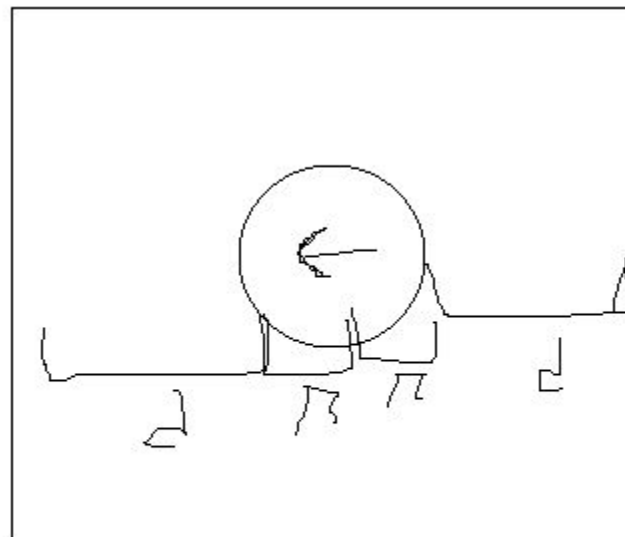
Challenge 4 – localização



A

B

C



-C

-B

-A

Challenge 4- localização (2)

- Criaram-se funções auxiliares para de acordo com o xy calculados pelos motores devolverem a parede da frente, trás, direita e esquerda do robô.
- Sintetizou-se nas seguintes funções o cálculo do x e y:
 - $xy = paredeF - ((dist + 0.5) * nivel);$
 - $xy = paredeT + ((dist + 0.5) * nivel);$
- Foram criados e testados 5 modelos diferentes:
 - Com correção de xy usando todos os sensores (4 lados diferentes);
 - Em todos os ciclos;
 - Apenas quando (acha que) chega ao centro de uma célula;
 - Com correção de xy usando apenas 2 sensores (frente e trás, 0° e 180°);
 - Em todos os ciclos;
 - Apenas quando (acha que) chega ao centro de uma célula;
 - Sem correção de xy usando valores dos sensores.

Resultados

No caso de ocorrer a correção com todos os sensores apenas quando se encontra no centro de uma célula.

- Tempo médio para a primeira colisão: 1400 ciclos;
- Número médio de colisões: 100 colisões;
- Tempo para terminar o mapeamento todo: NaN - nunca termina.

```
mapa.txt
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

```

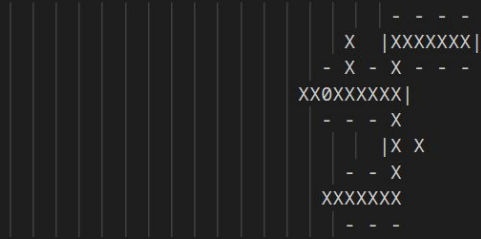
      | - - - - -
      |xxxxx|xxxxxxxxx|      x
      | - - X - X - - - X - - - X
      |xxxx0xxxxxx|xxxxx|x| |xxxxxxx|
      |X - - - X X - - - - X - - X
      |xxx| |xxx|xxx|xxx|xxx|3xx
      |X X      X - X - X X - X - -
      |X      |X|xxxxx|1|xxxxx|xxx|
      | - - - - X -      X X X X - X
      |xxxxx|xxxxxxxxx|x| |xxxxx|xxxxx|
      |X - X - - - X      X - - X - X
      |X X|X|xxxxxxxx| X|xxxxxxxxx|
      | - X X X - - X - - X - - X
      |xxxxxxxxxxx|xxxxxxxxxxxxxxxxx|
      |X - - - - X - - - - -
```

Resultados (2)

No caso de ocorrer sempre a correção
(com todos os sensores).

- Tempo médio para a primeira colisão:
400 ciclos;
- Número médio de colisões: 160
colisões;
- Tempo para terminar o mapeamento
todo: NaN - nunca termina.

```
mapa.txt
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```



Resultados (3)

Conclusões

- Contrariamente ao expectado o modelo sem correção de xy usando os valores dos sensores foi o que não só obteve melhores resultados como um melhor mapa e foi o mais estável, ou seja, conseguia terminar com sucesso mais vezes o mapeamento.