

# Simulateur déplacement d'un robot

Le projet consiste à simuler un robot qui se déplace sur une grille 2D en réponse à des commandes utilisateur (par exemple, avancer, tourner à gauche, tourner à droite). La grille représente l'espace de travail du robot et peut contenir des obstacles. L'objectif est d'écrire un programme en C qui permet à l'utilisateur de piloter le robot en donnant des instructions textuelles, tout en gérant les collisions avec des obstacles.

- [Fonctionnalités principales](#)
- [Séance 1 : Mise en place de la grille et déplacement basique](#)
  - [Correction](#)
- [Séance 2 : Optimisation et extension](#)

# Fonctionnalités principales

1) Représentation de la grille : La grille est un tableau 2D (par exemple, 10x10 ou 20x20) où chaque case peut être vide, contenir un obstacle, ou le robot. Le robot est représenté par un symbole (par exemple, 'R'), et les obstacles par un autre symbole (par exemple, 'X'). 2)

Mouvements du robot : Le robot peut se déplacer dans quatre directions (nord, sud, est, ouest) et peut tourner à gauche ou à droite. Les mouvements incluent : Avancer : Le robot avance d'une case dans la direction actuelle. Tourner à gauche/droite : Le robot change sa direction (sans changer de position). Recule (optionnel) : Le robot recule d'une case dans la direction opposée. 3)

Gestion des obstacles : Le robot ne peut pas avancer sur une case qui contient un obstacle. Si une commande le mène vers un obstacle, il doit ignorer cette commande ou signaler une erreur.

Commande utilisateur : L'utilisateur entre des commandes via la console pour piloter le robot (par exemple, 'A' pour avancer, 'G' pour tourner à gauche, 'D' pour tourner à droite). Les mouvements sont ensuite simulés et affichés sur la grille mise à jour après chaque commande.

# Séance 1 : Mise en place de la grille et déplacement basique

Objectif : Implémenter la grille et les mouvements de base du robot. 1) Initialiser la grille 2D : - Créer une matrice (tableau 2D) qui représente la grille. Chaque case peut être un espace vide (' '), un obstacle ('X'), ou le robot ('R'). - Positionner le robot au centre de la grille ou dans un coin spécifique. - Placer des obstacles à des endroits fixes ou les générer aléatoirement. Utiliser un symbole comme 'X' pour représenter un obstacle. 3) Commandes de déplacement : - Implémenter des fonctions pour que le robot se déplace : void haut(), void Gauche(), void Droite(), void bas() - Avant de faire avancer le robot, vérifier si la case vers laquelle il se dirige contient un obstacle. Si un obstacle est détecté, empêcher le robot de se déplacer et afficher un message d'erreur ou ignorer la commande. 4) Affichage de la grille : - Mettre à jour la grille après chaque mouvement pour que l'utilisateur puisse voir la nouvelle position du robot. - Afficher la grille dans la console après chaque commande pour visualiser la progression.

# Correction

```
1 #include <stdio.h>
2 #define ROWS 10 // Nombre de lignes de la grille
3 #define COLS 10 // Nombre de colonnes de la grille
4
5 void initGrille(char grille[ROWS][COLS], int robotX, int robotY) { // Fonction pour initialiser la grille
6     for (int i = 0; i < ROWS; i++) {
7         for (int j = 0; j < COLS; j++) {
8             grille[i][j] = '.'; // Remplir la grille avec des espaces vides
9         }
10    }
11    // Ajouter quelques obstacles
12    grille[1][1] = 'X'; grille[3][5] = 'X'; grille[6][7] = 'X';
13    grille[robotX][robotY] = 'R'; // Placer le robot à la position donnée
14 }
15
16 void afficherGrille(char grille[ROWS][COLS]) { // Fonction pour afficher la grille
17     for (int i = 0; i < ROWS; i++) {
18         for (int j = 0; j < COLS; j++) {
19             printf("%c ", grille[i][j]);
20         }
21         printf("\n");
22     }
23 }
24
25 void deplacerRobot(char grille[ROWS][COLS], int *robotX, int *robotY, int direction) { // Fonction pour déplacer le robot
26     grille[*robotX][*robotY] = '.'; // Retirer l'ancien emplacement du robot
27     int newX = *robotX, newY = *robotY; // Calculer la nouvelle position selon la direction
28
29     switch (direction) {
30         case 'H': newX--; break; // Haut
31         case 'B': newX++; break; // Bas
32         case 'G': newY--; break; // Gauche
33         case 'D': newY++; break; // Droite
34     }
35
36     // Vérifier si le mouvement est possible (pas d'obstacle, ni hors de la grille)
37     if (newX >= 0 && newX < ROWS && newY >= 0 && newY < COLS && grille[newX][newY] != 'X') {
38         *robotX = newX; // Mettre à jour la position du robot
39         *robotY = newY;
40     } else {
41         printf("Mouvement impossible (obstacle ou hors de la grille).\n");
42     }
43
44     grille[*robotX][*robotY] = 'R'; // Remettre le robot dans la nouvelle position
45 }
```

```
46
47 int main() {
48     char grille[ROWS][COLS];
49     int robotX = ROWS / 2, robotY = COLS / 2; // Position initiale du robot au centre de la grille
50
51     initGrille(grille, robotX, robotY); // Initialiser la grille avec le robot
52     afficherGrille(grille); // Afficher la grille initiale
53
54     char commande; // Boucle principale pour les commandes utilisateur
55     while (1) {
56         printf("\nEntrez une commande (H = haut, B = bas, G = gauche, D = droite, Q = quitter) : ");
57         scanf(" %c", &commande); // Lire la commande utilisateur (avec un espace pour ignorer les espaces blancs)
58
59         if (commande == 'H' || commande == 'B' || commande == 'G' || commande == 'D') {
60             deplacerRobot(grille, &robotX, &robotY, commande);
61         } else if (commande == 'Q') {
62             break; // Quitter la boucle
63         } else {
64             printf("Commande non reconnue.\n");
65         }
66         afficherGrille(grille); // Afficher la grille après chaque action
67     }
68     return 0;
69 }
```

# Séance 2 : Optimisation et extension

Objectif : Ajouter des fonctionnalités supplémentaires et améliorer l'expérience utilisateur.

Amélioration de l'interface utilisateur : - Ajouter une fonction pour réinitialiser la position du robot. -

Permettre à l'utilisateur de choisir la taille de la grille au début du programme. - Ajouter une option pour générer aléatoirement des obstacles ou les positionner manuellement.