Katia SIFESSALAM groupe 7
Kilian TROUET groupe 7

# DM2 Algo

## I Introduction

Dans ce DM, nous avions pour objectif de créer un QuadTree à l'aide de nos connaissances sur les arbres vus en cours ainsi que de la libMLV pour l'affichage graphique.

## II Structures

1) Position:

Il est préférable d'initialiser une structure position étant donné que nous manipulons des coordonnées, pour l'emplacement des points et des positions noeuds (carré) à afficher.

```
typedef struct Position {
    int x;
    int y;
} Position;
```

x correspond à l'axe des abscisses et y à l'axe des ordonnées.

2) Quadtree:

Notre Quadtree est un tableau de noeud. Chaque nœud possède les indices des nœuds fils, les positions de début et de fin d'un nœud (carré) sur la fenêtre, l'indice du premier point ajouté dans ce nœud et le nombre de points placé dans ce nœud et nœud enfant.

```
typedef struct Noeud {
    int fa;
    int fb;
    int fc;
    int fd;
    Position positionFirst;
    Position positionSecond;
    int plist;
    int nbp;
}Noeud, * QuadTree;
```

fa, fb, fc, fd sont les indices des nœuds fils.
fa -> indice * 4 + 1
fb -> indice * 4 + 2
fc -> indice * 4 + 3
fd -> indice * 4 + 4

plist -> premier élément ajouté dans ce nœud. On le met à -1 si c'est un nœud parent.

nbp -> nombre de points présent dans les nœuds fils et ce nœud.

3) Points:

On matérialise un point par une position et une couleur (dans un objectif purement esthétique).

```
typedef struct  {
    Position pos;
    int color;
} Point;
```

pos -> coordonnées du points dans la fenêtre.
color -> entier correspondant à une couleur.

Nous avons choisi d'utiliser deux tableaux pour nos points. Le premier, tab_points, contient nos différents points et le deuxième, tab_peres, contient pour chaque indice, correspondant à l'indice de notre point dans le premier tableau, le point qui est le premier de la liste chaînée dans la cellule où se trouve le point auquel on s'intéresse.
Nous avons décidé d'utiliser cette technique pour avoir seulement une autre allocation de plus à faire afin de savoir quels points sont ensemble dans le QuadTree.

4) Option:

Dans un objectif d'amélioration nous avons décidé d'initialiser une structure contenant les informations d'exécution du programme.

A partir de ces options nous pouvons:
- afficher/masquer les points: 'a'
- activer le déplacement des points: 'z'
-mettre fin au programme: 'e'

```
typedef struct Option {
    int affiche_point;
    int deplacement;
    int play;
    Deplacement type_deplacement;
}Option;
```

5) Type de déplacement:

Quatre types de déplacement sont possibles pour les points:

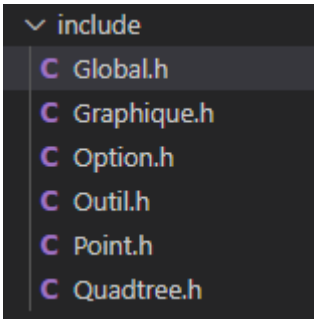
-**Aléatoire centrique**: Les points se déplacent de manière aléatoire autour de sa position initiale.

-**Spiral**: Un point invisible se déplace de manière circulaire dans la fenêtre. Ce point est suivi par tous les points initialisés.

-**Centre**: Les points se déplacent vers le centre de la fenêtre.

-**Aléatoire:** Les points se dirigent vers une position qui leur est attribuée en fonction de leur indice. Lorsque le point atteint la position ciblée. Une nouvelle en est générée.

## III Modularisation




La modularisation de ce projet est réalisée en fonction du “type” des fonctions  pour Point.h et Quadtree.h

Le module **Global** permet de passer en variable global les paramètres d'exécution du programme, comme la taille de la fenêtre, taille du tableau de point, taille maximum du tableau de point, taille minimum des cellules, nombre max de point par cellule. Mais également le tableau des pères (point présent dans un même nœud) et le tableau de point, car ce sont des tableaux qui sont initialisés avec une taille fixe qui n’est pas vouée à changer jusqu’à la fin de l’exécution du programme.

**Point** contient les fonctions allocation, d’initialisation, d’ajout et de déplacement de point. **Quadtree** possède les fonctions d’initialisation

**Graphique** comporte les fonctions utilisant les éléments de la lib MLV pour afficher les points et le quadtree. Mais également les fonctions récupérant les event de la souris et du clavier qui sont utilisés pour ajouter un point, activer des options, comme le déplacement, afficher/masquer les points, changer le type de déplacement.

**Outil** contient les fonctions qui n’ont pas été placées dans d’autres modules et **Option** la fonction d’initialisation des options pour l’exécution du programme.

## IV Implémentation et amélioration

Pour définir le nombre limité par nœud, la taille minimum d’un nœud, la taille de la fenêtre et le nombre de points initialisés au lancement du programme, il faut modifier les macros situés dans le main.

```
#define SIZE_WINDOW 512
#define NBP 7
#define MIN_SIZE_CELL 4
#define POINTS 250
```

Pour matérialiser les fils d'un nœud du tas, nous avons opté pour l'utilisation de l'indice du fils au lieu de l'adresse. Ce qui nous permet de parcourir de manière plus simple le tas mais également car nous trouvions plus cohérent d'utiliser des indices pour tout (comme pour plist).

Nous avons décidé d'apporter quelques améliorations au projet initialement demandé comme:

-Afficher/Masquer les points.
-Mouvement des points.
-Différent type de déplacement pour les points.

## V Difficultés rencontrées

Quelques difficultés ont été rencontrées lors de la compréhension de l'énoncé. Au départ nous avons commencé à utiliser une structure d'arbre basique et voulions ajouter les fils d'un nœud seulement si le nombre de point limite était dépassé. Puis nous avons compris que nous devions simplement allouer toute la mémoire du tas nécessaire au début.

La seule erreur rencontrée en utilisant valgrind est liée à la lib MLV.

## VI Instruction de compilation

| | |
|---|---|
| Compilation du programme | make main |

## VII Instruction d'utilisation

| | |
|---|---|
| exécution | ./main |

## VIII Mode d'emploi

**Ajout d'un point**: Clic gauche
**Activer/Masquer les points**: 'a'
**Activer le déplacement des points**: 'z'
**Déplacement aléatoire des points**: 'q'
**Déplacement en spiral** : 's'
**Déplacement aléatoire autour du point initial**: 'd'
**Déplacement vers le centre** : 'f'
**Quitter le programme**: 'e'