RAPPORT QuadTree

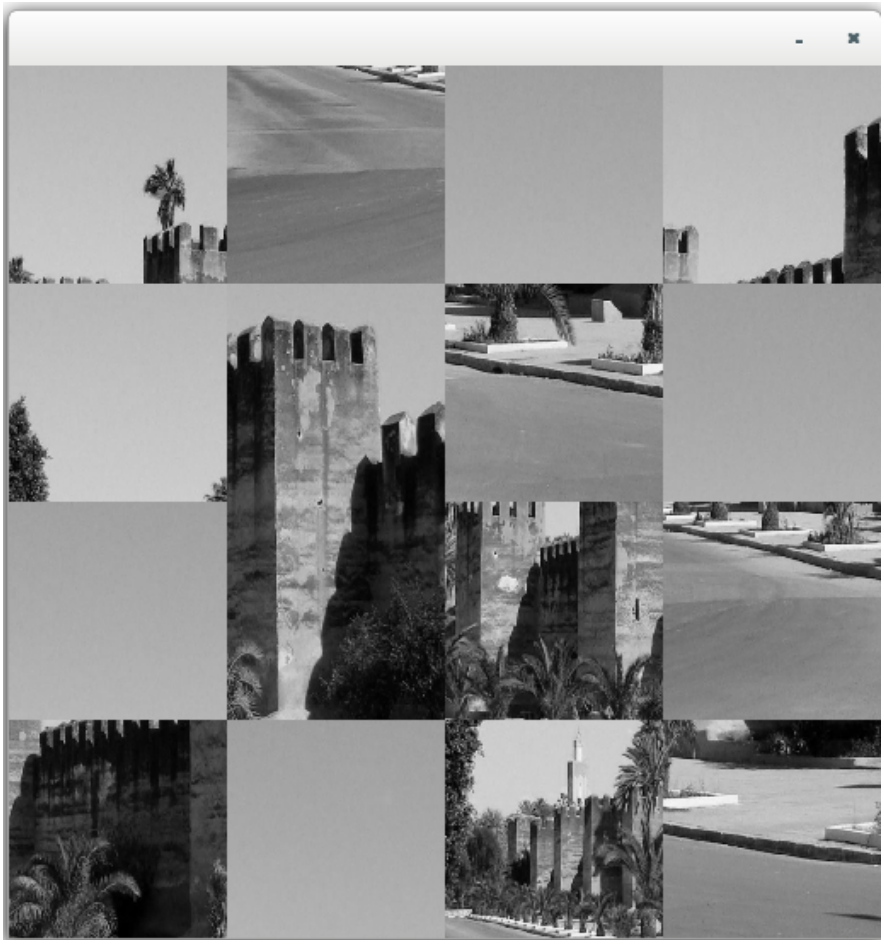
Rapport de Devoir Maison

logins: zsdaf - pravendr

Tables des matières:

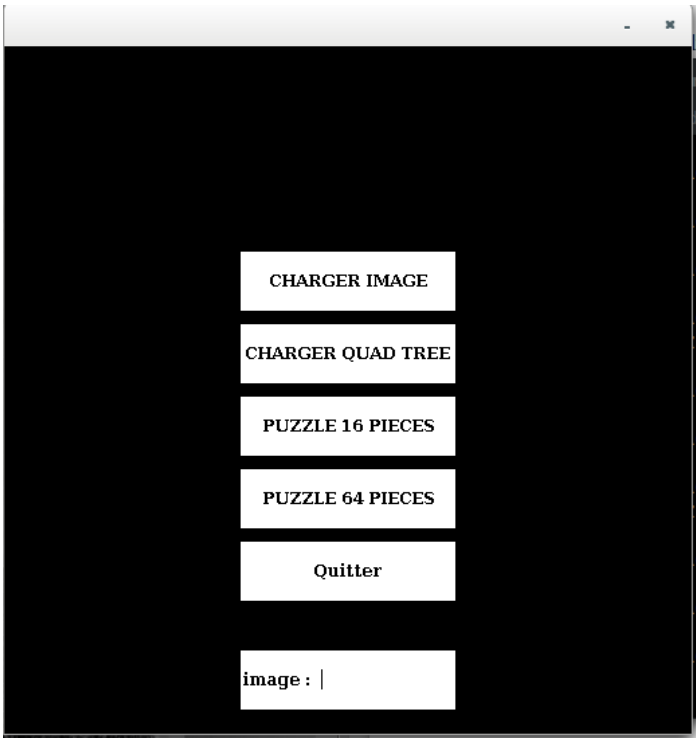
Présentation

L'objectif de ce projet est la réalisation d'un jeu de puzzle résoluble par échange de pièces représentées par un quadtree.
L'objectif principal du projet est l'utilisation d'algorithme de manipulation d'arbre quaternaire.



Manuel

Au lancement du programme avec la commande './quadTree', une fenêtre graphique s'ouvre avec un menu composé de 5 boutons:



- **CHARGER IMAGE** : Ce bouton permet de charger l'image spécifiée dans la boîte de saisie située en bas de l'écran. Une fois l'image chargée, le programme genère un fichier texte ayant le même nom que l'image mais avec l'extension '.quad' qui contient la représentation en quadtree de l'image.

- **CHARGER QUAD TREE** : Ce bouton permet de charger une image à partir d'un fichier .quad (cette fonction n'a pas été implémentée par manque de temps).

- **PUZZLE 16 PIECES** : Permets de lancer une partie de puzzle en découpant l'image chargée en 16 pièces.

- **PUZZLE 64 PIECES** : Permets de lancer une partie de puzzle en découpant l'image chargée en 64 pièces.

- **QUITTER** : Permets de quitter le programme.

Une fois le jeu lancé, le but est de cliquer sur 2 parties de l'image pour les échanger et le jeu se termine lorsque l'image originale est reformée.

Réalisation

On a découpé notre programme en 4 fichiers .c suivi de leur fichiers .h respectifs et pour finir le fichier main.c, on a 2 modules différents :

- ***quad*** composé des fichiers **quad**.h et **quad**.c.

  Ce module regroupe les fonctions et structures en rapport avec la manipulation des arbres quartenaires.

  Pour représenter un arbre quartenaire, nous avons choisis une structure '***Qtree***' et '***Partie***

- ***puzzle*** composé des fichiers **puzzle**.h et **puzzle**.c. Ce module regroupe les fonctions et structures nécessaires pour le jeu du puzzle.;

Points clefs

Les structures principales :

- Structure ***Qtree*** composée d'un champs ***couleur*** (la couleur du noeud ou -1 si c'est un noeud interne), ***level*** (le niveau du noeud dans l'arbre quartenaire), ***partie*** (les coordonnées de la partie de l'image qui correspond au noeud) et 4 pointeurs vers des noeuds qui représentent les 4 sous arbres du quad :

```
typedef struct noeud {
        int couleur;
        int level;
        Partie partie;
        struct noeud* x;
        struct noeud* y;
        struct noeud* z;
        struct noeud* t;
} Noeud, *Qtree;
```

- Structure ***Partie*** composée des positions ***pos_x*** et ***pos_y*** de chaque partie et d'une taille qui sont tous des entiers :

```
typedef struct Partie {
        int pos_x;
        int pos_y;
        int taille;
} Partie;
```

- Structure ***Puzzle*** composée d'un champs image de type ***MLV_Image*** en tant que pointeurs, de deux entiers ***nb_pieces*** et ***level***, puis de trois champs ***quad***, ***tabGame*** et tabFirst de type ***Qtree :***

```
typedef struct {
        MLV_Image* image;
        int nb_pieces;
        int level;
        Qtree quad;
        Qtree tabGame[64];
        Qtree tabFirst[64];
} Puzzle;
```

- Structure ***Rect*** composée de 4 entiers ***x***, ***y***, ***l*** et ***h*** :

```
typedef struct {
        int x;
        int y;
        int l;
        int h;
} Rect;
```

- Structure Event composée de event et state des types respectifs ***MLV_Event*** et ***MLV_Button_state***, puis de deux entiers sx, sy :

```
typedef struct event {
        MLV_Event event;
        MLV_Button_state  state;
        int sx;
        int sy;
} Event;
```

Exécution du programme

L'exécutable **QuadTree** se génère en exécutant le fichier le **Makefile make** mais aussi **./quadTree**.

Conclusion

En conclusion, ce devoir nous a fait travailler sur deux points essentiels : l'utilisation de la bibliothèque graphique MLV, l'utilisation des fichiers .quad, ainsi que sur la modularité. Le fait de découper son projet en plusieurs fichiers le rend plus organisé. De plus certains fichiers peuvent être indépendants et ainsi être utilisés dans de futurs projets.