Programmation C Avancée - TP 6

Irwin Madet

7 novembre 2020

Résumé

Ce projet consiste à concevoir et programmer un *Jeu de Taquin*. Le but du jeu est de reconstituer une image, découpée en morceaux, qui a été mélangée. Le programme doit être capable de détecter la victoire du joueur, compter le nombre de coups et être configurable facilement.

1 Introduction

1.1 Compilation

Le programme repose sur des librairies standard et la librairie `MLV`. Il faut installer cette dernière pour compiler le programme. La librairie `MLV` peut être trouvée à l'adresse suivante :

`http://www-igm.univ-mlv.fr/~boussica/mlv/`

Une fois la librairie installée, il suffit de compiler le programme avec la commande :

```
$ make
```

1.2 Utilisation

Le programme se lance en ligne de commande et nécessite au minimum, un chemin vers une image qui servira pour le jeu. Il est possible de renseigner en argument le nombre de cases en hauteur, le nombre de cases en largeur, ainsi que le nombre de mouvements à effectuer au mélange initial. La syntaxe est la suivante :

```
$ ./taquin <Chemin image> [--height x] [--width y] [--scramble z]
```

Le jeu se joue en cliquant sur une case adjacente à la case vide pour les échanger. Une fois l'image reconstituée, le jeu se termine et le programme attend un clic pour terminer.

2 Description des modules

2.1 Module `Board`

Ce module sert à modéliser un plateau de jeu, lui-même composé de carrés. Chaque structure de carré comporte les coordonnées du carré dans le plateau. Deux variables globales `COL` et `ROW` représentent le nombre de lignes et le nombre de colonnes. Ces dernières sont modifiées dans `main.c` par les arguments.

Le module inclut les fonctions suivantes pour interagir avec le plateau :

- `int initBoard(Board *b, int row, int col)`
  Initialise et alloue dynamiquement un plateau de dimensions `COL` × `ROW`.
- `void scramble(Board *b, int moves)`
  Mélange le plateau en effectuant `moves` mouvements aléatoires.
- `int swapSquare(Board *b, int col, int row)`
  Échange le carré de coordonnées (`col`;`row`) avec le carré vide si ce dernier est adjacent.
- `void getSquare(Board *b, int col, int row, Square **square)`
  Cherche le carré aux coordonnées (`col`;`row`) et range son adresse dans `square` si ce dernier a été trouvé.
- `int win(Board b)`
  Détermine si le plateau a été résolu ou non.
- `void freeBoard(Board b)`
  Libère la mémoire réservée par le plateau.

2.2 Module `Renderer`

Ce module se charge de gérer la fenêtre de jeu et dessine les différents éléments sur la fenêtre. Il contient diverses macros constantes représentant certaines mesures de la fenêtre.

Le module inclut les fonctions suivantes pour interagir avec la fenêtre :

- `void initWindow(Board b)`
  Initialise la fenêtre à des dimensions dépendantes de la taille de l'environnement de bureau dans lequel est exécuté le jeu.
- `void drawBoard(Board b, MLV_Image *image)`
  Dessine le plateau de jeu avec l'image passée en paramètre.
- `void drawMoves(Board b, int moves)`
  Dessine le texte affichant le nombre de coups `moves` effectués.
- `void drawVictory(Board b)`
  Dessine le message de victoire.

3 Difficultés rencontrées

Les principales difficultés rencontrées résident dans la partie graphique et le positionnement des différents éléments de l'interface. Une autre source d'erreurs a été le système d'échange de carrés. En effet, lors des premiers tests, il était fréquent que le programme plante, que les carrés ne bougent pas ou que de la mémoire fuie.

4 Démonstration

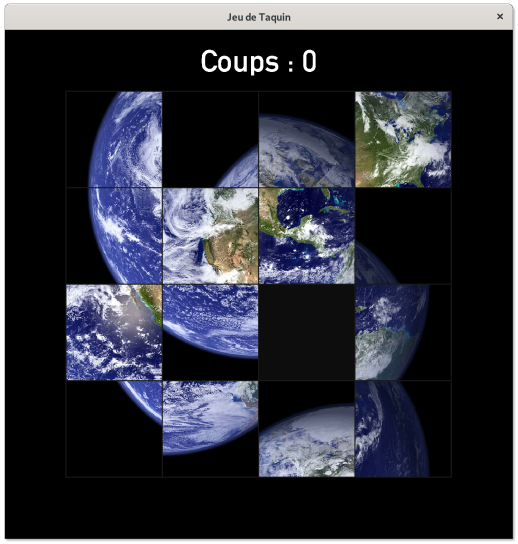

FIGURE 1 – Situation initiale

FIGURE 2 – Jeu terminé