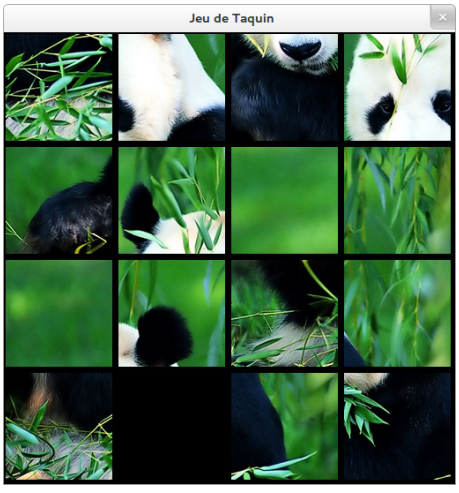
Programmation C avancée
TP 4

L'objectif de ce TP est de programmer un petit jeu de taquin avec interface graphique utilisant la LibMLV.

Pour des questions de simplicité, nous ne manipulerons que des images carrées dont les côtés sont de taille 512 pixels et d'extension `png`, `gif` ou `jpg`.

Soit une telle image avec un splendide panda

Après un découpage 4 par 4 de l'image, l'adjonction d'une case vide et un mélange du jeu, on peut obtenir l'image suivante :



Une fois le jeu fini, vous devriez obtenir pour la même image quelque chose comme :

Attention, on ne peut pas mélanger le jeu comme on le souhaite ! Toutes les positions aléatoires ne donnent pas un plateau de jeu ayant une solution. Vous pouvez remarquer que depuis la position de fin de jeu, seul un nombre pair de mouvements peut donner une combinaison qui fait retrouver la position de fin de jeu. Ainsi, lorsque l'on casse le jeu (imaginez-le en plastique) et qu'on tente de le réparer, ils ne faut pas remettre des pièces n'importe comment (il se passe la même chose pour un Rubik's cube, les cubes ont des positions qui sont liées les unes aux autres). Pour le mélange de début de jeu, une solution relativement raisonnable consiste à exécuter aléatoirement un certain nombre de mouvements acceptables. Les mouvements étant légaux, on a ainsi la garantie d'obtenir un plateau qui possède une fin.

Modélisation du problème :

Dans un premier temps, vous pouvez proposer une version faisant un découpage 4 par 4 de l'image avec la case noire en bas à droite. Ainsi, les deux structures suivantes définissent raisonnablement l'espace de jeu.

```
1  #define NB_COL 4
2  #define NB_LIG 4
3
4  typedef struct carre{
5    int lig; // ligne originale dans l'image
6    int col; // colonne originale dans l'image
7  }Carre;
8
9  typedef struct plateau{
10   Carre bloc[NB_COL][NB_LIG];
11 }Plateau;
```

Chaque petit carré mobile est décrit par sa ligne et sa colonne **dans sa position de base** (celle du plateau quand le jeu est fini). Une fonction initialisant le plateau avant mélange est ainsi

```
1  void InitialisationPlateau(Plateau* P){
2    int i, j;
3
4    for (i=0 ; i < NB_LIG ; i++){
5      for (j=0 ; j < NB_COL ; j++){
6        ((P->bloc)[i][j]).lig = i;
7        ((P->bloc)[i][j]).col = j;
```

```
8         }
9     }
10 }
```

Maintenant, seul le carré identifié par le numéro de colonne et de ligne 3 (le fameux carré noir) a le droit de s'échanger avec des voisins. Pour mélanger un plateau 4 par 4, 120 mouvements aléatoires autorisés mélangent complètement le plateau de jeu (c'est statistique...).

Suggestions d'amélioration : (optionnel, pour ceux qui trouvent les TP trop simples...)

- Faire en sorte que l'utilisateur choisisse la complexité en entrant deux entiers, nécessite une bonne refonte du projet.
- Laisser l'utilisateur choisir son image dans une banque d'image.
- Générer un plateau de jeu avec des entiers (pour un jeu 4 par 4, cela consiste à placer des numéros de 1 à 15 écrits en noir sur un fond blanc), le jeu est fini quand les numéros sont rangés de gauche à droite et de bas en haut.
- Compter les secondes prises par l'utilisateur pour finir la partie. Tenir les high scores dans un fichier dédié, seuls les 10 meilleurs apparaissent dans le tableau des high scores.
- Tout ce qui vous parait sympathique à rajouter...

La LibMLV est la bibliothèque de l'université pour les petits projets graphiques en langage C. Cette bibliothèque interface la SDL et ainsi emballe des fonctions difficiles à utiliser dans des fonctions claires et simples. Cette dernière est disponible sous système Unix ou Windows pour des systèmes 32 ou 64 bits. Le projet est disponible à la page :
`http://www-igm.univ-mlv.fr/~boussica/mlv/api/French/html/index.html`
La documentation des fonctions a été automatiquement générée par `doxygen` et cette dernière est disponible en français à l'adresse :
`http://www-igm.univ-mlv.fr/~boussica/mlv/api/French/html/globals_func.html`

Un module utilisant la LibMLV doit inclure les entêtes de la LibMLV :

```
1 #include <MLV/MLV_all.h>
```

Si vos fonctions n'utilisent pas de type définis dans la LibMLV, vous pouvez confiner cette inclusion dans un fichier de code `.c`, sinon il faudra le mettre dans les headers du module.

Les modules important la LibMLV doivent être compilés avec le flag `-lMLV`.

Voici la liste des fonctions issues de la LibMLV utilisées par votre enseignant pour réaliser un prototype de votre TP :

- MLV_create_window (création d'une fenêtre)
- MLV_free_window (libération de la fenêtre)
- MLV_load_image (ouverture de l'image)
- MLV_actualise_window (demande d'opérer les changements d'affichage à l'écran)

- MLV_draw_text (afficher un texte)
- MLV_draw_filled_rectangle (pour tout effacer, on met un rectangle noir dans la fenêtre)
- MLV_draw_partial_image (affichage d'une sous partie d'une image donnée)
- MLV_draw_pixel (dessiner un pixel particulier sur l'écran)
- MLV_convert_rgba_to_color (fabrication d'une couleur)
- MLV_get_random_integer (fabriquer un entier aléatoire)
- MLV_get_event (récupération des mouvements et clics de la souris)
- MLV_wait_seconds (attendre quelques secondes une fois le jeu fini)

Dernières consignes et remarques :

- Faites du code de qualité (documentation).
- Faites un compte rendu pour ce TP4.
- Modularisez agréablement votre code.
- Faites de la compilation séparée avec des jolis `makefile`.
- Google est capable de rechercher des images de taille exactement 512 par 512 pixels. Choisissez plutôt des images libres de droit.