IUT de Fontainebleau
Département informatique
22/12/2017

MICROPROJET

Le jeu du Taquin (Version informatique)

Jedrocha Benjamin
Leblanc Kévin

Rapport Projet Taquin

Introduction ____ 3
Description des fonctionnalités ____ 4
Menu ____ 4
Quitter ____ 4
Règles ____ 4
Jouer ____ 5
Structuration interne du programme ____ 7
Explication des données représentant l'état d'une partie en cours ____ 9
Compteur ____ 10
Permutation des tuiles ____ 10
Vérification ____ 11
Explication du mélange ____ 12
Avis ____ 13
Jedrocha Benjamin : ____ 13
Leblanc Kévin : ____ 13

Introduction

Le microprojet consiste en la réalisation du jeu du Taquin dans un langage informatique, le C89. Le taquin traditionnel se joue seul et est composé de 15 tuiles numérotées de 1 à 15 qui glissent dans un cadre prévu pour 16. Le but est de déplacer les tuiles pour revenir à la configuration initiale.

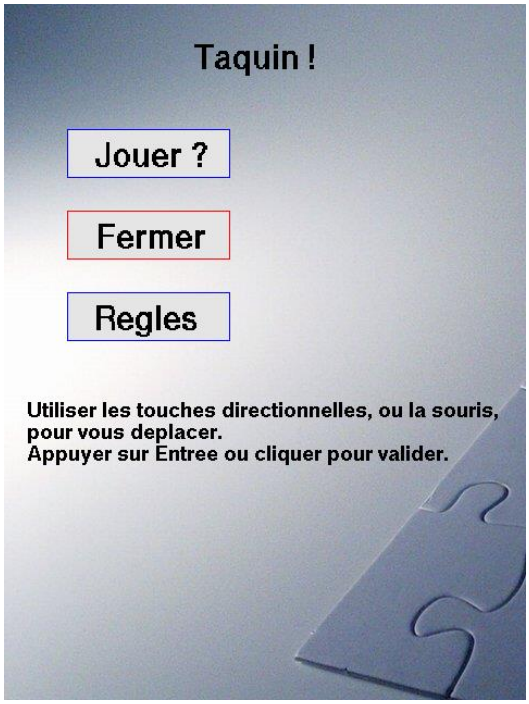
Le sujet nous impose de ne pas utiliser des chiffres mais une image découpée en morceaux mélangés. L'utilisateur aura le choix d'au moins trois images de dimensions différentes et du nombre de colonnes et de lignes indépendamment (entre 3 et 8 pour chacun des deux) pour enfin reconstituer l'image d'origine en déplaçant ces morceaux. Les morceaux d'image doivent être mélangés de façon à ce que la configuration initiale de l'image doit être réalisable en glissant les images une par une dans la case vide.

Le Taquin doit être réalisable aussi bien au clavier qu'à la souris. Il devra proposer une image de référence de la configuration initiale qui sera disponible lorsque l'utilisateur jouera. Un compteur indiquera en permanence le nombre de coups joués.
Lorsque la configuration initiale sera retrouvée, le programme demandera à l'utilisateur s'il souhaite quitter ou recommencer une nouvelle partie.

Description des fonctionnalités

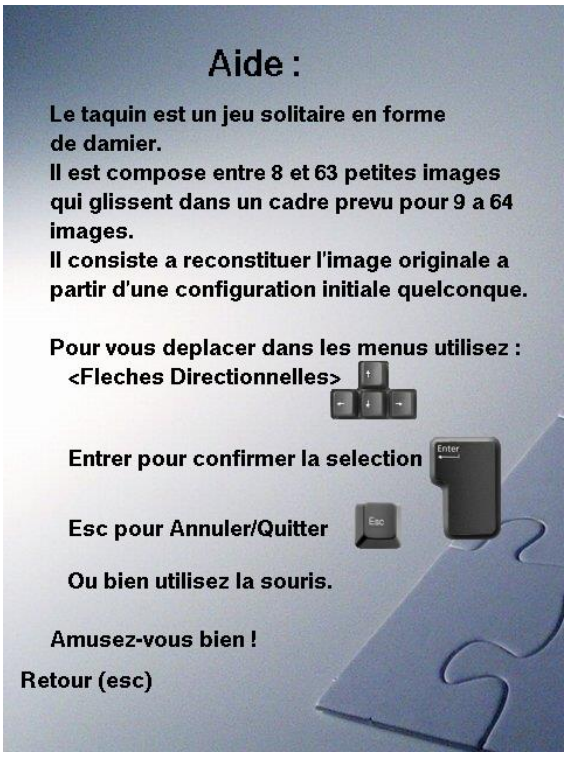
Menu

Au lancement du programme, l'utilisateur voit un premier menu. Il a le choix entre Quitter, Règles, et Jouer. Il peut se déplacer dans le menu avec les flèches directionnelles, Entrée ou bien sélectionner avec la souris.



Quitter

Comme son nom l'indique, ferme la fenêtre et le programme.

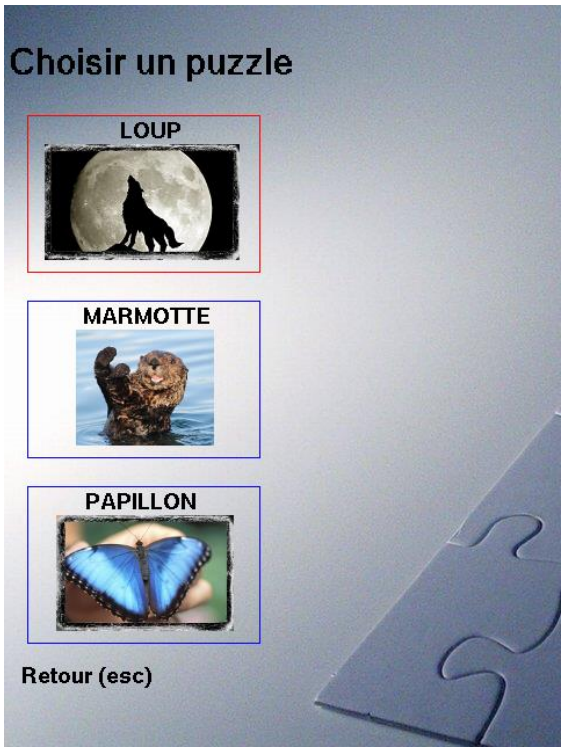


Règles

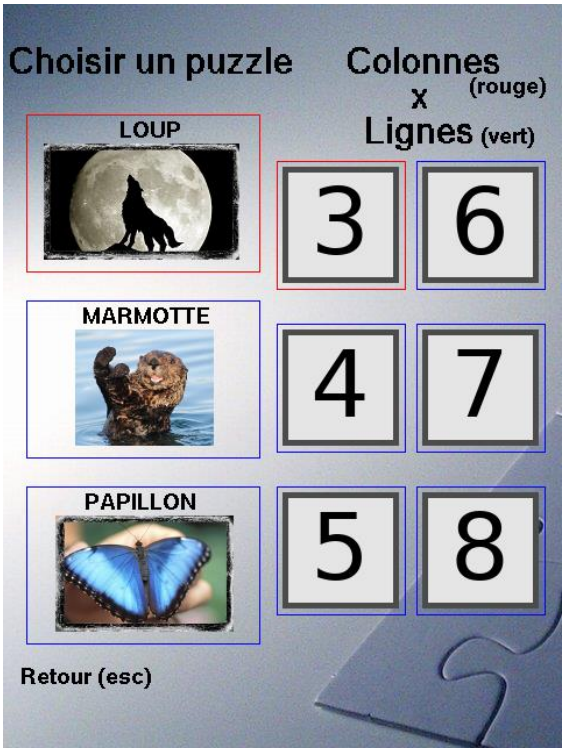
Si vous avez cliqué sur Règles, une rapide description des règles du Taquin modifié est affiché, ainsi qu'une aide pour jouer, se déplacer dans les menus, à l'aide du clavier et de la souris.

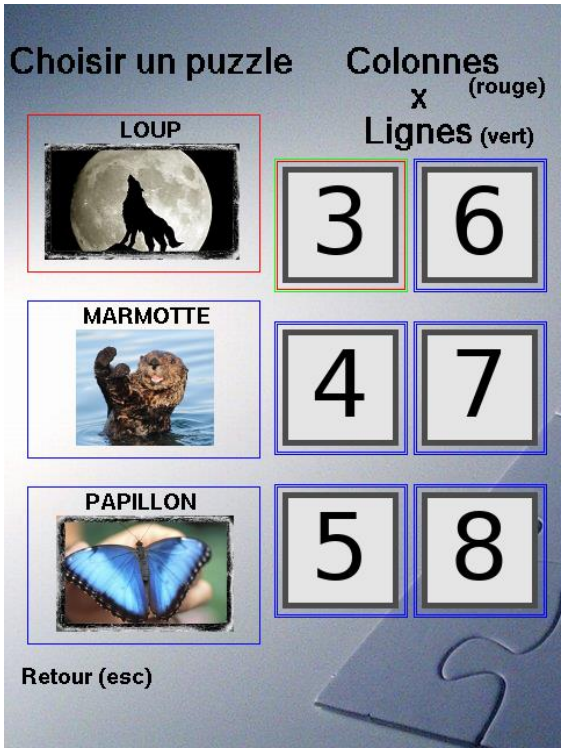
Jouer

Un menu proposant 3 images apparaît, vous pouvez les sélectionner en vous déplaçant grâce au clavier et en la sélectionnant avec Entrée ou à la souris en cliquant sur l'image (un indicateur rouge permet de savoir quelle image est sélectionnée). Vous pouvez bien évidemment faire retour au menu principal en appuyant sur Echap ou en cliquant sur Retour (esc).

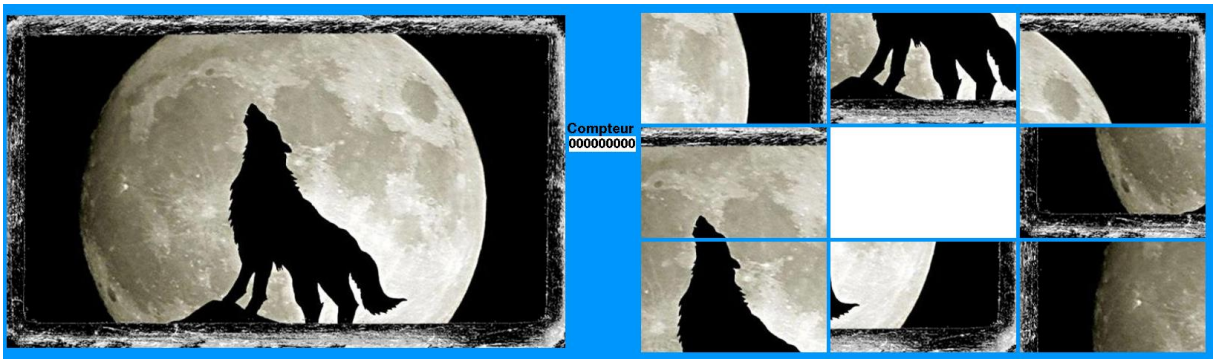


Si vous sélectionner une des 3 images, de nouveaux choix apparaissent. En premier le nombre de colonnes identifié par un indicateur rouge, puis le nombre de ligne avec l'aide d'un indicateur vert.

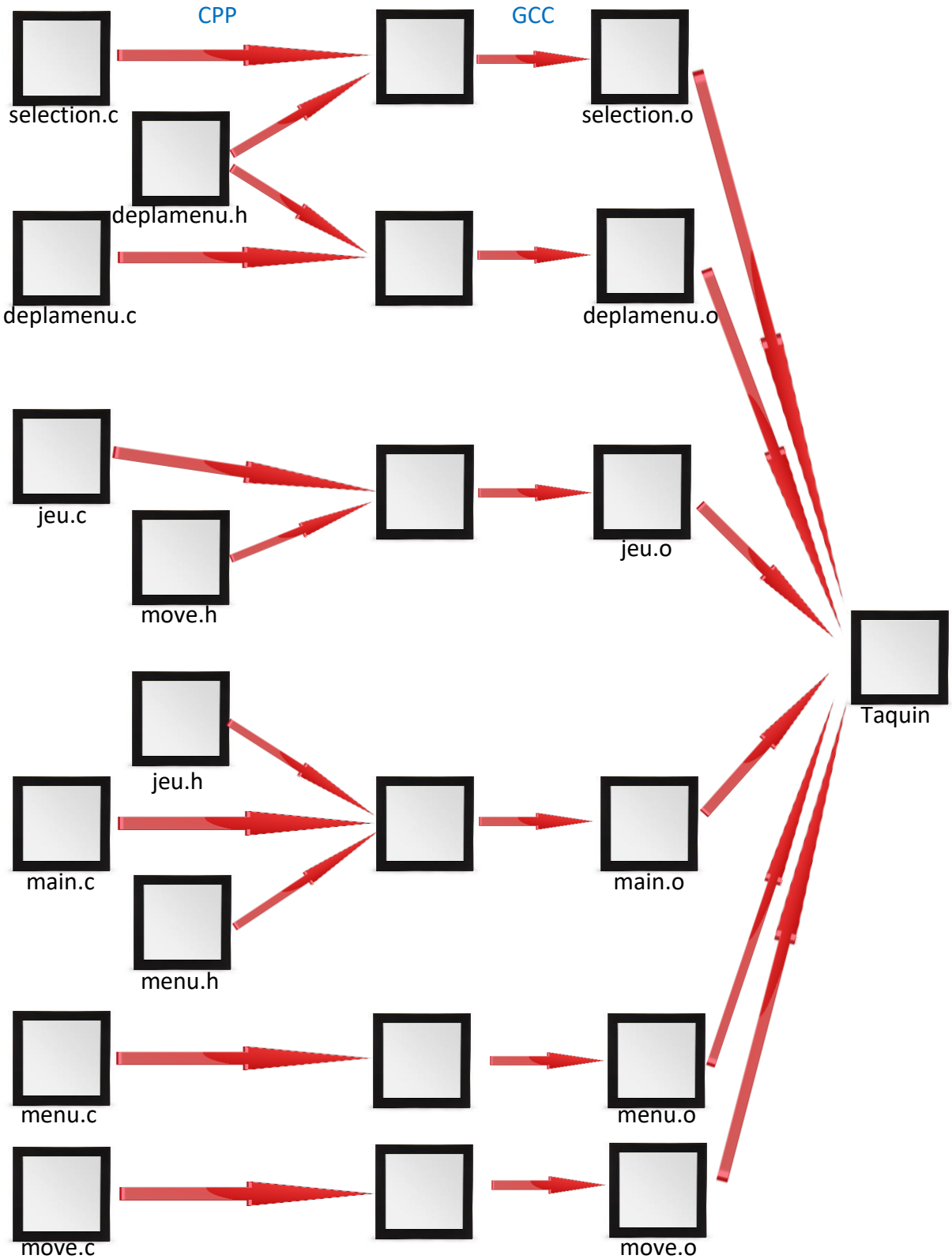




Une fois tout sélectionné, l'utilisateur se trouve enfin en jeu avec une image à gauche, celle de référence pour retrouver la configuration de l'image découpée et mélangée à droite. Entre l'image de référence et celle découpée, se trouve le compteur où le nombre de coups est actualisé à chaque fois que des morceaux de l'image permutent. Une fois la configuration initiale retrouvée, une autre interface se crée pour y afficher le nombre de coups pour retrouver l'image ainsi qu'un bouton quitté et un autre pour rejouer et revenir au menu des choix des images.



Structuration interne du programme



Notre programme contient les fichiers :

-main.c qui est dépendant de menu.h et de jeu.h

-selection.c qui est dépendant de deplamenu.h et ne contient qu'une seule fonction qui va initialiser les données pour le déplacement dans les menus.

-jeu.c qui est dépendant de move.h qui contient les fonctions mélange, jeu et vérification. Nous les avons regroupées ainsi car cela permet de savoir que nous sommes dans la partie en cours.

-menu.c qui contient delw, displayReplay, puzzles, lines, help et menu. Ces fonctions sont regroupées car elles ne font qu'afficher du texte ou des images.

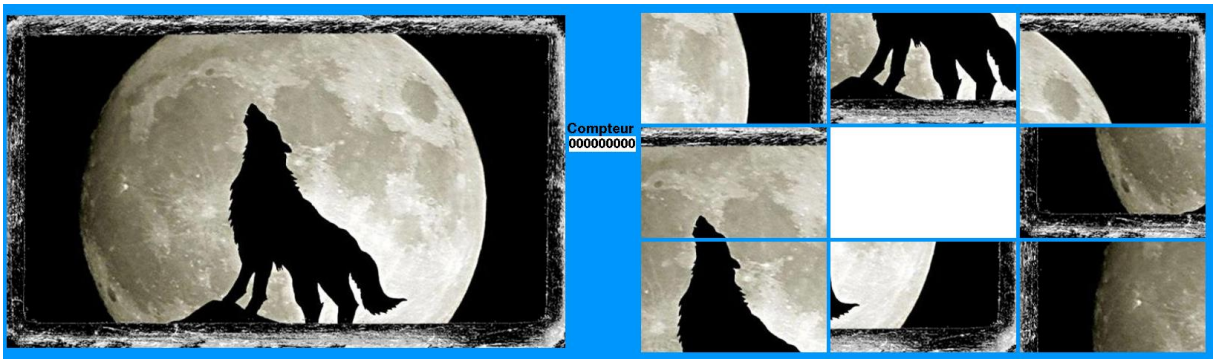
-move.c qui contient les fonctions displayCompteur, clavier et souris puisqu'il s'agit du déplacement des tuiles dans la partie en cours.

-deplamenu.c qui est dépendant de deplamenu.h et qui contient les fonctions selclavier et selsouris puisqu'il s'agit du déplacement dans les menus.

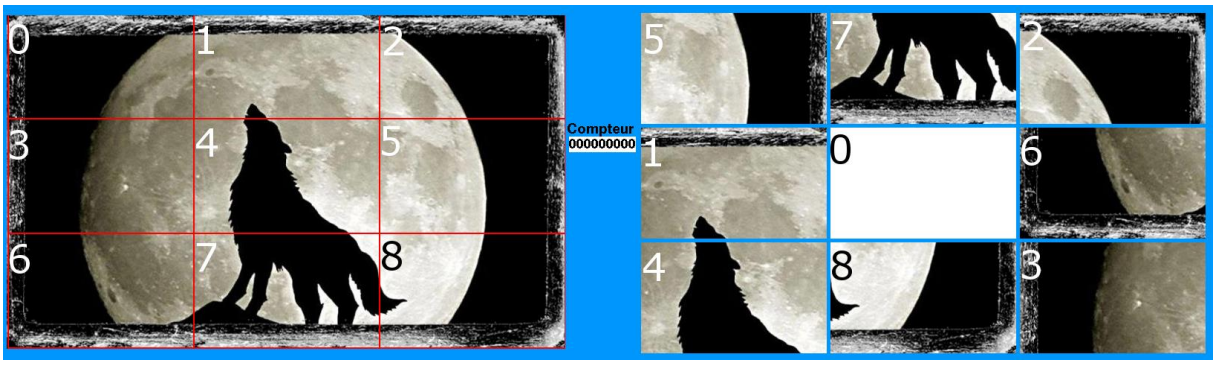
Le fichier main.c va préparer presque toutes les données nécessaires et va appeler quasiment chaque fonction qui vont préparer les données restantes nécessaire puis appeler les fonctions importantes.

Explication des données représentant l'état d'une partie en cours

Lorsqu'une partie est en cours, l'image de référence se situe à gauche et l'image découpée à droite, déjà mélangée et que l'on peut modifier, avec le compteur initialisé à zéro.



Chaque tuile correspond à un numéro, ces numéros permettent de trouver quelle est la portion d'image qu'il représente dans l'image entière (ici, celle de gauche) comme dans l'exemple ci-dessous :

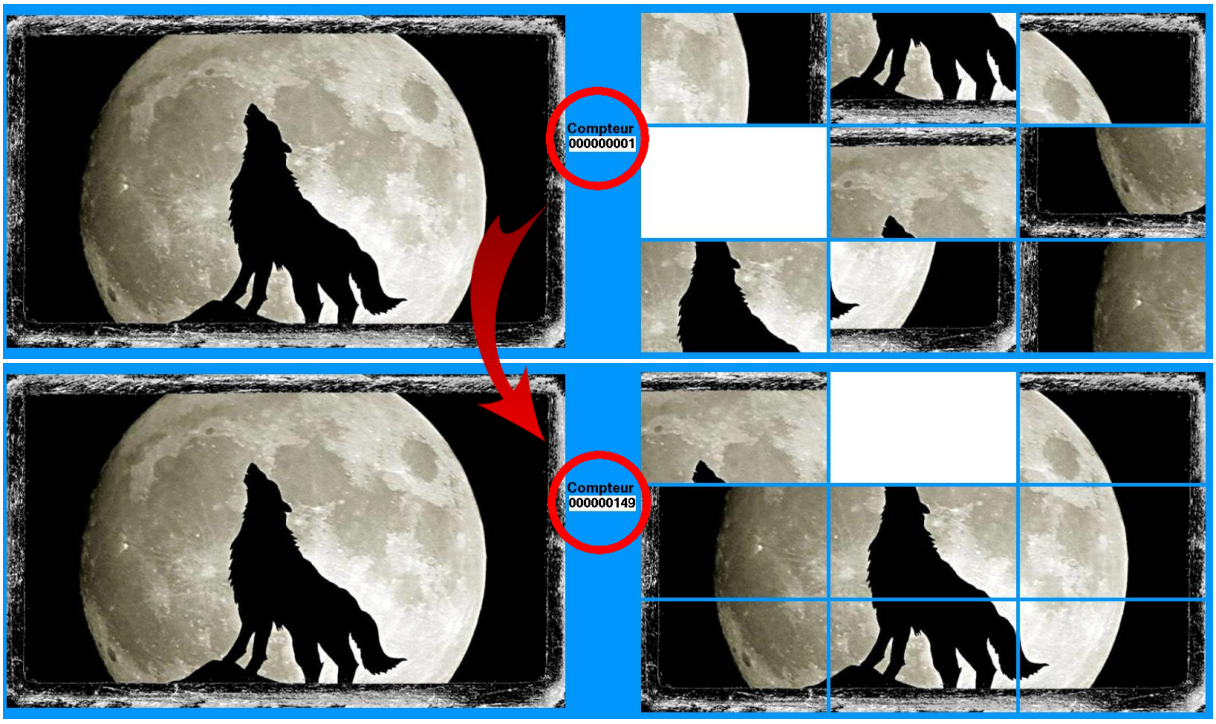


Dans l'endroit où l'on déplace les tuiles, le numéro 0 représente la case vide.

En divisant ce numéro par le nombre de colonnes, on peut trouver quelle coordonnée Y correspond la portion d'image dans l'image entière. Si on utilise le modulo sur ce même numéro, on obtient la position en X de cette portion. La position de ce numéro dans le tableau permet de savoir sur quelles coordonnées afficher cette portion d'image dans le jeu. Par exemple dans l'exemple ci-dessus, la tuile vide 0 a pour coordonnées dans le tableau (1 ;1) si on part du fait que la tuile 5 a pour coordonnées (0 ;0) ceci nous permettra de faire la vérification que nous verrons plus tard.

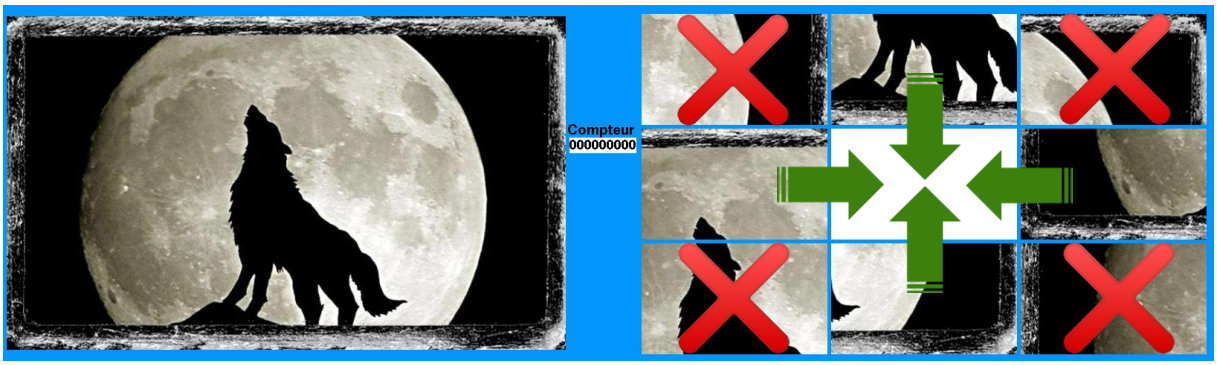
Compteur

Il s'actualise lorsque que deux tuiles sont permutées. L'ancien nombre affiché à l'écran est effacé lorsque les cases permutent sinon les nombres d'après se superposent. Nous avons utilisé un rectangle plein qui permet d'effacer le nombre avant que le nouveau s'affiche.



Permutation des tuiles

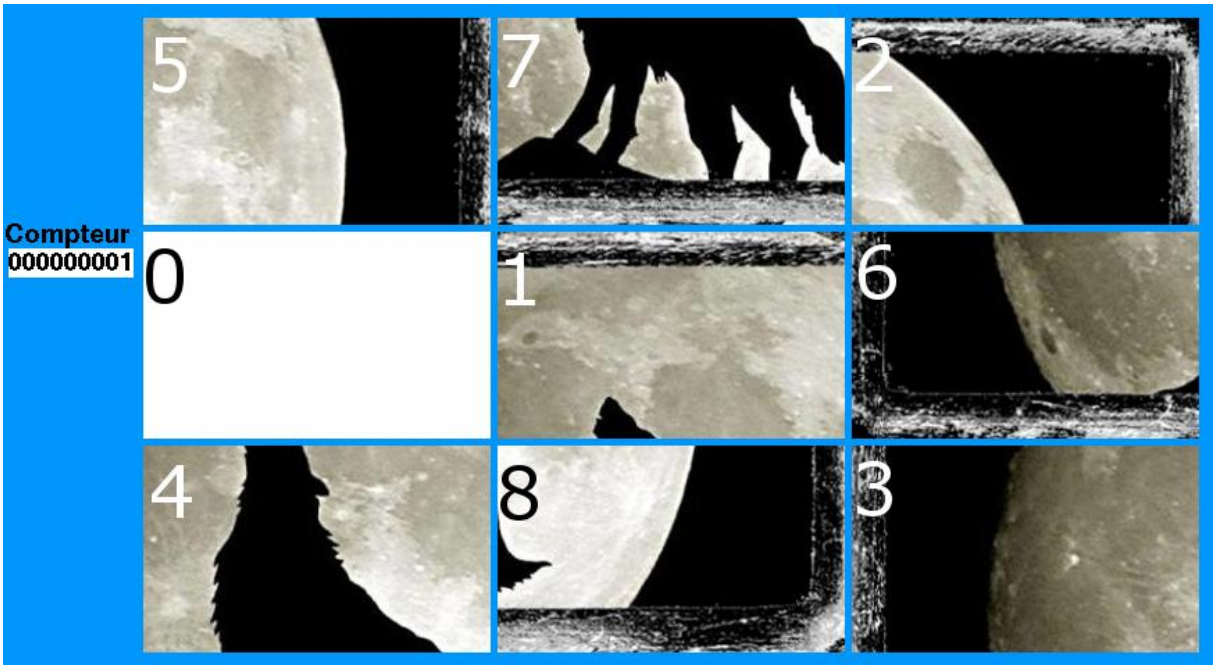
Lorsque vous sélectionnez une tuile à déplacer qui est obligatoirement adjacent à la tuile blanche, leur numéro permute dans le tableau et ainsi comme vu plus haut, grâce à leur position dans le tableau et à leur valeur, on met un carré blanc là où se trouvait la tuile sélectionnée et à l'autre position on y affiche la portion d'image. Ainsi, il n'y a qu'un rafraichissement sur les 2 tuiles et non sur toutes les tuiles ce qui permet de jouer rapidement.



Vérification

A chaque fois qu'une tuile est déplacée, on vérifie le nombre de tuiles qui sont bien placés en comparant la position dans le tableau des valeurs associées à chaque tuile à un autre tableau où les valeurs sont dans l'ordre et si ce nombre est égale au nombre de cases que compose le jeu (qui est différent selon le découpage choisi) alors le jeu s'arrête pour aller dans le menu de fin pour choisir entre quitter et rejouer.

Dans l'exemple ci-dessous, il n'y a pas de tuile à la bonne position donc le nombre de tuile à la bonne position est 0 et n'est pas égale aux nombres de cases du jeu donc le jeu continue.



Explication du mélange

Tout d'abord on initialise la fonction srand() sur le temps actuel.

Ensuite on remplit un tableau de nombres suivis (0 jusqu'au nombre de tuiles -1) .

Puis on mélange ce tableau grâce à rand() et à une variable temporaire (tmp).

On cherche la parité de la tuile blanche dans le tableau ainsi que celle de la permutation afin de vérifier la condition suivante : « si la permutation est impaire alors que la case vide est paire, la résolution est impossible » (wikipédia). Ainsi ils doivent avoir la même parité pour que le mélange soit soluble.

Avis

Jedrocha Benjamin :

Pour conclure, j'ai assez bien aimé faire ce projet, surtout en groupe. Avec ce projet j'ai pu m'améliorer et mieux compris certaine notion comme le Makefile les structures et les allocations dynamiques.

On a pu se partager les tâches facilement et ainsi gagner du temps. Je me suis principalement occupé des sélections dans les menus et du jeu avec les découpages, la taille des fenêtres, la permutation des tuiles … De plus on pouvait s'entraider et soit corriger ou améliorer le code de l'un et l'autre, nous étions bien organisés.

Nous avions aussi utilisé le site GitHub pour organiser les ressources. Un dossier servait de dossier référence où tout fonctionnait et un autre où l'on faisait les modifications. C'est lorsque ces modifications étaient terminées que nous les remplaçons dans le fichier référence et ainsi de suite, ceci nous permettait de rester à jour.

Nous avons aussi corrigé des bugs comme par exemple dans le menu des Règles, si l'on cliquait les touches étaient mise en tampon et dès que l'on revenait au menu ça pouvait sélectionner pour aller dans les autres menus.

Ce projet m'a montré aussi le temps qu'il faut pour la réalisation d'un tout petit jeu, ce qui permet de mieux appréhender le temps et les compétences nécessaire pour réaliser autre chose et que l'organisation est très importante.

On a essayé également d'utiliser le plus possible des mêmes fonctions, celles qui font par exemple la sélection, avec les flèches directionnelles, les rectangles bleus avec en rouge celui qui montre lequel est sélectionné, nous les avons utilisés sur tous les menus.

Merci pour la lecture.

Leblanc Kévin :

J'ai beaucoup aimé travailler sur ce projet avec mon coéquipier. L'avantage est qu'on a pu se partager les tâches pour gagner du temps, s'entraider, et voir que son binôme n'a pas forcément les mêmes idées et qu'il y a plusieurs chemins pour parvenir au même résultat. Ce projet m'a permis de m'entraîner et de progresser sur des notions plus ou moins bien comprises.

Lors de la répartition des tâches je me suis centré sur la partie logique du programme, les différents déplacements selon les menus, la solvabilité du taquin et quelques parties graphiques.

J'ai aussi appris que la notion du temps est vraiment très importante, au fur et à mesure que le programme se construisait/s'organisait, on souhaitait ajouter des fonctionnalités ou des facilités d'utilisation et pourquoi pas rendre les graphismes meilleurs mais on a été contraint de faire certains choix, comme par exemples quelques morceaux de codes non adaptables et une fenêtre de jeu adaptative à l'image choisie, et parfois le programme ne fonctionnait plus normalement, il fallait trouver l'erreur entre le Makefile incompatible entre certains ordinateurs et des morceaux de codes qui ne fonctionnent plus.

On peut donc dire qu'on a eu un petit aperçu de ce que pouvait donner la réalisation d'un programme qui paraît simple mais demande pourtant beaucoup de travail si l'on souhaite un résultat fini et non simplement opérationnel.

Merci de votre lecture et de votre attention.