PROJET PICROSS

KAJAK Rémi
KINZY Érick
MAROUF Taous
NOUVELIÈRE Benjamin

4 avril 2018

Table des matières

I. Introduction **3**

II. Organisation du travail **4**

A Fonctions principales **4**
A.1 Sous-fonctions 4

B Outils de partage **5**

III. Analyse et conception **6**

IV. Codage, méthode et outils **7**

A SDL **7**

B Programmation modulaire **7**

V. Résultats et conclusion **8**

I. Introduction

L'informatisation est le phénomène le plus important de notre époque. Elle s'immisce dans la plupart des objets de la vie courante, que ce soit dans l'objet proprement dit ou bien dans le processus de conception ou de fabrication de cet objet.

L'objet de notre travail est de programmer une version simplifiée du jeu « Picross », qui est un jeu de réflexion solitaire. Le but consiste à découvrir un dessin - ou un motif - sur une grille en noircissant des cases. Les valeurs données sur le côté gauche et en haut de la grille donnent des indices : ils indiquent la taille des groupes de cases noires de la ligne ou de la colonne sur laquelle ils se trouvent. La difficulté est progressive, et au fur et à mesure de son avancée dans le jeu, le joueur doit résoudre des grilles comportant plus de cases et plus de groupes par rangée (ligne/colonne).

Notre objectif était de réaliser l'analyse et la conception d'un Picross en utilisant le langage C. Nous nous sommes servi des outils de base de la programmation, de la récursivité et avons manipulé des fichiers.

Dans ce rapport, nous allons tout d'abord exposer l'analyse de ce jeu, puis le codage et les méthodes utilisées.

II. Organisation du travail

Pour mener à bien la création du jeu avec le langage C, il a fallu trouver des structures et un moyen efficace pour stocker les données des puzzles à résoudre. Étant un jeu à grille, la définition et l'initialisation de matrices furent immédiatement envisagées. Quant au stockage des données, des fichiers texte possédant un format spécifique ont été créés pour répondre au besoin de la génération des données pour les matrices.

A Découpe du problème en fonctions principales

Pour ce faire, il a fallu définir quatre matrices principales avec une taille fixe :

- ☐ une matrice solution où, pour un niveau donnée, le motif final serait présent ;
- ☐ deux matrices périphériques qui contiendraient les nombres des groupes, nécessaires à la résolution du puzzle par le joueur ;
- ☐ une matrice de jeu, vide par défaut, qui serait la grille de jeu pour l'utilisateur.

Différentes types de matrices ont été codées et testées - entre autre, une structure contenant une variable de type énumération -, mais au final, seules les matrices de type entier (utilisées pendant les tests unitaires) ont été préservées. La fonction la plus à même de remplir ce rôle est **init_matrice([1 argument])**, où N est une constante pour la taille par défaut.

Les matrices ne suffisant pas pour le jeu, des fichiers texte ont été conçus avec un format par défaut :

- ☐ puzzles_binaires.txt ;
- ☐ nombres_puzzle.txt.

La fonction **lecture_fic([5 arguments])** permet de lire ces fichiers et d'interpréter les données présentes. Le premier fichier texte contient des rangées de 0 et de 1 en clair qui sont directement assignées à la matrice solution ; les nombres des matrices périphériques sont générées ensuite avec la lecture de la première matrice. Quant au second fichier, il s'agit de l'opération inverse : les nombres des matrices périphériques sont données, et il faut les utiliser pour générer la matrice solution.

Ces deux types de fichier sont nécessaires pour répondre aux deux parties demandées par la liste des sujets montrée au début du semestre :

1. Réaliser une première version du jeu « Picross » (terminal ou graphique) ;
2. Mettre au point un solveur => l'ordinateur doit pouvoir résoudre une grille de Picross sans regarder la matrice solution.

Enfin, une fois les générations des trois premières matrices effectuées, les fonctions gérant les parties (**ChangerEtat([1 argument])** et **verifierGrille([2 arguments])**) permettent au joueur d'interagir avec la grille de jeu.

A.1 Sous-fonctions nécessaires

Partie à rédiger moi-même

B Outils de partage

Pour mener à bien le projet, et comme ce fut demandé par les professeurs encadrants, un dépôt distant GitHub a été créé et mis à disposition à tous les membres du groupe. Il est disponible à l'adresse suivante :

http://github.com/PicrossDevTeam/S4

Voici la liste des comptes ayant un libre accès au dépôt distant et pouvant le modifier à souhait, y pousser de nouveaux fichiers ou dossiers (push), ou bien les récupérer (pull) :

- ☐ L2info041 (KAJAK Rémi) ;
- ☐ taous06 (MAROUF Taous) ;
- ☐ nouveliere-benjamin (NOUVELIÈRE Benjamin) ;
- ☐ erick022 (KINZI Érick).

Néanmoins, quelques problèmes ont été relevés peu de temps après la création du dépôt distant. Outre l'initiation des néophytes à la création d'un dépôt local (git init) et à sa configuration pour pouvoir mettre en ligne les fichiers ajoutés à ce dépôt, le dépôt distant n'acceptait que les mises en ligne du compte propriétaire ayant servi à sa création. Le problème a été réglé quelque temps après - environ deux semaines depuis le début du projet - en attribuant aux membres simples un droit d'écriture au lieu du droit de lecture défini par défaut.

De plus, peu de temps après la résolution de ce problème, l'historique des différents commits de ces membres n'ont pas tout de suite été pris en compte par GitHub. Seul ceux du propriétaire du dépôt (L2info041) s'affichaient après chaque nouvelle mise en ligne. Le problème s'est réglé de lui-même sans raison apparente (erreur interne possible). Entre-temps, le propriétaire du dépôt récupérait tous les fichiers sur son propre dépôt local et les mettait en ligne pour ne pas transgresser les règles imposés dans le cadre du projet.

En-dehors de GitHub et des sessions TP de conduite de projet, le service e-mail de l'ENT et une conversation de groupe Facebook ont été utilisés pour communiquer sur l'avancée du projet. Bien que ce dernier outil ne soit guère recommandé par la pédagogie, il était plus instinctif pour nous de l'utiliser plutôt que de créer un serveur sur Slack. L'idée avait été trouvé peu après le début du semestre 4, mais elle a été oubliée par la suite.

III. Analyse et conception

Partie rédigée par Benjamin

IV. Codage, méthode et outils

A SDL

Partie rédigée par Benjamin

B Programmation modulaire

Partie rédigée par Érick

V. Résultats et conclusion

Dans ce rapport, nous avons expliqué d'une manière générale les étapes de développement du jeu. Nous avons spécifié les besoins puis nous avons proposé une solution.

Voici une liste des différentes problèmes que nous avons pu détecter sans pouvoir leur trouver une solution :

- la fonction **lecture_fic([5 arguments])** ne gère pas toujours les sauts de lignes entre chaque ligne de nombres, ce qui restreint le format trouvé.

Ensuite, voici quelques améliorations possibles :

- une interface graphique convenable ;
- un affichage dynamique en fonction de la complétion de chaque rangée (grisage automatique) ;

Ce projet nous a permis de nous familiariser avec certains outils informatiques vu en cours. Enfin, nous aurions pu réaliser toutes les améliorations citées ci-dessus, mais le temps nous a manqué.