IUT Fontainebleau

Rapport du projet Memory

Star Ouarz Memory

Sylvain THOR
21/12/2018

Table des matières

Introduction

En cette fin d'année 2018, nous a été proposé la réalisation d'un Projet en langage C. Il s'agit d'un memory, l'objectif étant de retourner toutes les paires de carte le plus rapidement possible. Il nous a été mis à notre disposition une bibliothèque graphique (graph.h) afin de pouvoir écrire des applications graphiques sans avoir à maîtriser le fonctionnement et l'API de la couche X11. Ensuite, c'est à nous de faire appel aux connaissances acquises lors du premier module d'APL et de les appliquer.

NB : Veuillez bien m'excuser pour l'offense dans le nom du jeu (Star Ouarz), n'ayant pas les fonds suffisants pour collaborer avec le chef d'œuvre qu'est la saga Star Wars.

Les différentes fonctionnalités du programme

1) Retournement des cartes par 1 clic

Pour se faire, j'ai placé l'ensemble de cette partie du code dans une boucle qui ne s'arrête que lorsque le jeu est terminé (à savoir que toutes les paires sont bien retournées).

A l'intérieur de cela s'additionne une boucle de 2 référençant à quelle carte a-t-on à faire : si c'est la première ou la deuxième.

Ensuite, j'ai mis une boucle qui ne s'arrête pas (elle s'avère être bloquante) tant que le joueur n'a pas cliqué sur aucune des cases. A l'aide de certaines conditions, j'ai fait en sorte que si l'utilisateur ne clique sur aucune des cases (car j'ai superposé les zones cliquables sur les cases respectives), ou si lors du second clic, il clique sur la même case que celle du premier, alors le clic n'aura aucune importance.

Après avoir cliqué sur une case, on détermine grâce à un compteur à quelle valeur du tableau aléatoire on à affaire. Puis on implémente cette valeur dans une chaîne de caractère qui va renvoyer le chemin relatif qui mène à l'image en question. Ainsi on va pouvoir aisément appeler une image à apparaître comme on le souhaite.

Lorsque les 2 cartes retournées sont identiques, alors leurs valeurs correspondent et les deux cartes se voient recevoir une valeur d'état (en l'occurrence : les 2 cartes seront figées) permettant de déterminer si la carte est dévoilable ou pas (dans ce cas elle est figée).

Comme on peut le voir ci-dessus, une fois 2 cartes identiques retournées, alors ces dernières se figent grâce à la valeur d'état ce qui permet en plus de les exclure des cases cliquables (à l'aide d'une simple condition).

2) Timer

De la même manière le timer se situe dans une boucle « infinie » (qui je le rappelle se finit à la réception d'un clic ou de la pression de la touche 't') qui ne cesse de se demander si elle a dépassé l'instant présent (stocké dans une variable) + 1 seconde. En effet, dès que l'instant présent dépasse l'ancien instant présent (qui est, par conséquent, vieux d'1 seconde) + 1 seconde, alors l'ancien instant présent prend la valeur de l'instant présent immédiatement après s'être fait dépasser (cette phrase est on ne plus ambigüe, mais très précise).

Ainsi, il suffit tout simplement d'afficher une chaîne de caractère qui est actualisée toutes les secondes par un compteur pour représenter le timer dans le jeu.

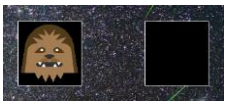
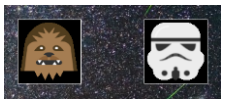
3) Différentes tailles de grille

Dans ma fonction du menu, après avoir cliqué sur le bouton « jouer » il est possible de choisir 3 tailles de grille différentes (4x4, 4x6 ,4x8) comme on peut le voir ci-après :

Intuitivement, j'ai superposé sur ces 3 différents boutons, 3 zones cliquables qui renvoient intimement le nombre de cases de la grille, la taille de la fenêtre correspondante et une variable spécifique en rapport avec la taille de la fenêtre (qui me servira à créer les cases du memory).

4) Cartes découvertes restent affichées 1 seconde

Lorsque les 2 cartes sont différentes alors leurs valeurs respectivent ne constituent pas un doublon donc elles ne sont pas égales et ainsi va s'écouler, par la même technique que pour celle du timer, 1 seconde durant laquelle les 2 images différentes resteront affichées et il sera impossible de cliquer sur aucune des cases. Le point divergent avec le timer, c'est que que cette mini-boucle qui ne dure qu'une seconde, soit qu'1 seul cycle (1 cycle dans le code correspond à 1 seconde) n'attend rien en retour (ni clic, ni touche) et se termine automatiquement au bout d'1 seconde et enfin recouvre ces 2 images.

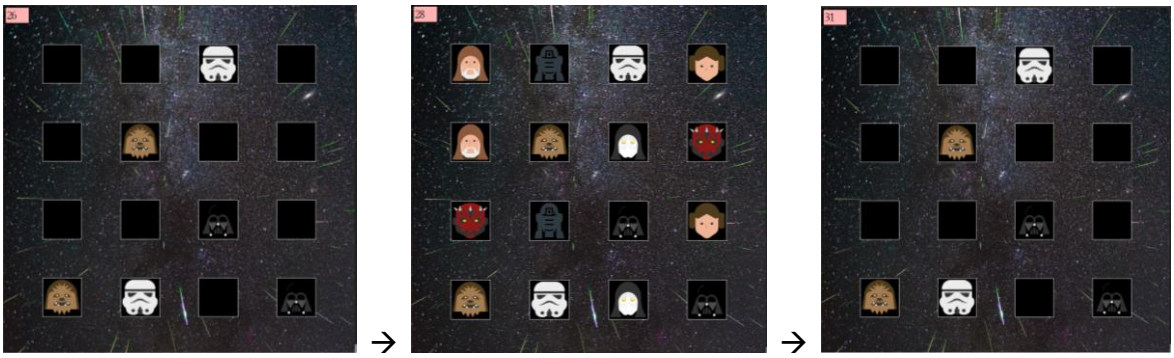
 →  (1sec s'écoule) → 

5) Mode de triche

Enfin, tout comme avec le clic, l'activation du mode de triche impose la réception d'une action, dans ce cas il s'agit de la pression de la touche « t ».

J'ai le mode de triche dans une fonction à part entière, elle s'occupe elle-même d'analyser si la touche reçue par le tampon s'identifie bien à la touche « t », sinon, elle sort de la fonction. Lorsque le mode sera activé, toutes les images seront appelées à leur place et seront affichées. Cependant, aucune valeur d'aucun tableau n'aura été modifié,

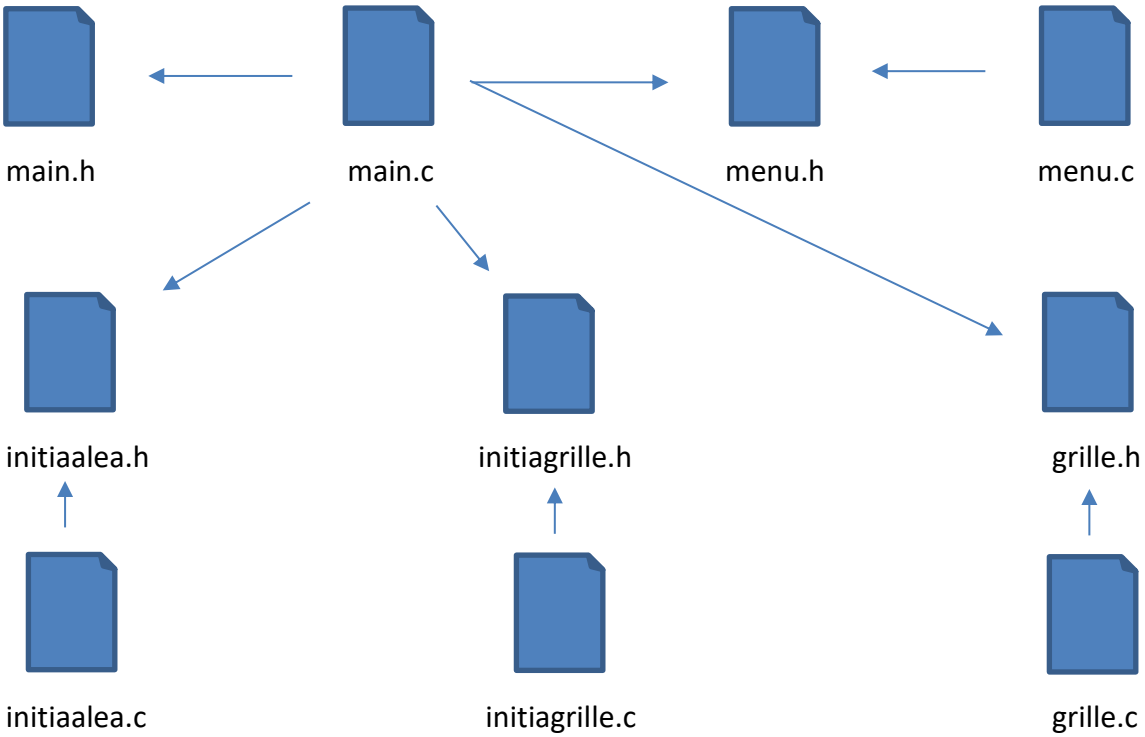
le jeu sera en quelque sorte sauvegardé à l'état où il était avant de lancer le mode de triche.



On constate que la partie qui était en cours a bien été sauvegardée.

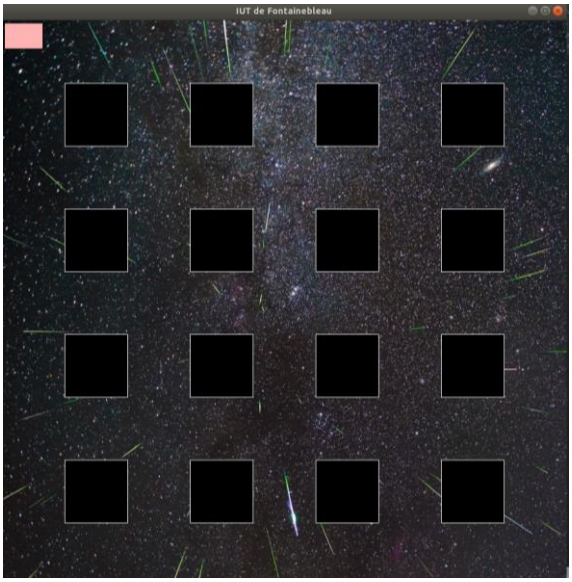
Pour ce qui est du timer, le temps ne s'écoule que lorsqu'on est dans la boucle « infinie » (je rappelle qu'elle ne l'est pas vraiment) ou lors de la seconde de comparaison de 2 images différentes. Quand le mode de triche est activé on sort de cette boucle et le temps ne s'écoule plus du tout jusqu'à la désactivation du mode de triche.

Présentation de la structure du programme

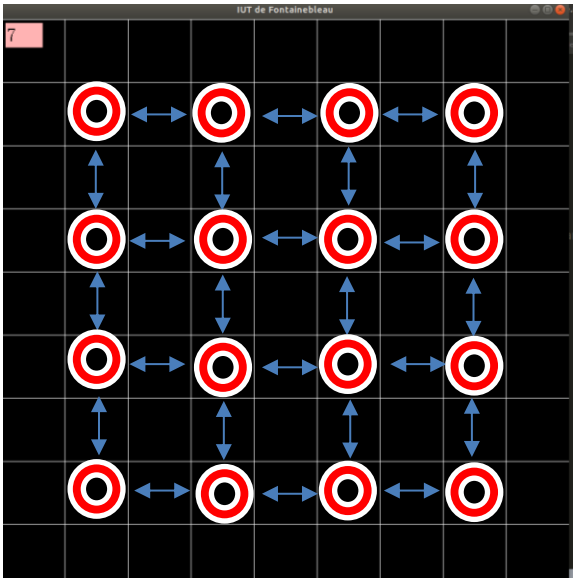


Explication de la forme des données qui représentent la grille de jeu

C'est à l'aide d'une sorte de boucle imbriquée (sans tableau multidimensionnel) que je divise la fenêtre de jeu en carrés (grâce à la variable div dont j'ai parlé précédemment). Evidemment, cette action n'a pas les même valeurs en fonction de la taille de la grille et par conséquent de la fenêtre. Puis je vais me contenter d'afficher seulement un carré sur 2 :



Remarque : j'ai réalisé un schéma qui montre bien que tous les carrés/cases (paradoxalement représentés par des ronds dans cet exemple) sont équidistants les uns des autres comme on peut le voir ci-après :



Exposition de l'algorithme qui remplit cette grille en début de partie

Dans cette partie, j'ai fait en sorte d'effectuer un tirage aléatoire sur un intervalle donné (qui dépend de la taille de la grille), si la taille de la grille s'élève à 16 cases, alors je cherche un tirage de 16 valeurs, dont 8 doublons (le tout dans un tableau).

L'algorithme stocke dans une variable la première valeur aléatoire, ensuite, à l'aide d'une boucle et d'un compteur temporaire, il va tester toutes les valeurs du tableau et incrémenter le compteur temporaire lorsqu'il tombe sur une case du tableau qui est égale à la variable aléatoire. Tant que le compteur temporaire est inférieur à 2 (car on ne veut pas plus que des paires/doublons) on assigne la valeur de la variable aléatoire à la position courante du tableau, et quand c'est le cas on incrémente la boucle principale afin de réiterer l'opération avec une autre valeur aléatoire stockée dans la variable.

Conclusion personnelle

Cette expérience représente l'aboutissement du premier projet de programmation auquel j'ai été sujet. Au début, insouciant, je commence les choses comme je le peux, je découvre la bibliothèque graphique, j'ai mes premières idées jusqu'à ce que je me rende compte qu'être en quelque sorte lâché dans la nature sans aucune expérience préalable s'avère être plus compliqué que prévu.

Cette sensation de ne pas savoir nager dans un océan bien vaste m'a poussé à aller de l'avant en m'intéressant de fond en comble sur cette bibliothèque graphique. Le premier module de programmation que j'affectionne particulièrement en dépit de mes résultats s'est révélé détenteur de surprises auxquelles je n'étais pas préparé. Effectivement, confronté aux problèmes concrets rencontrés lors de la compilation, la réflexion ou encore la réalisation, on finit bien par se résoudre à devoir revoir certains points abordés en TP/TD.

Je suis très fier d'avoir pu revoir différentes notions et de les avoir redécouvertes n'ayant pas en moi ce recul qui permet de se rendre compte instantanément des erreurs à ne pas faire comme par exemple (pour ma part) : négliger le cours sur l'allocation dynamique en me demandant bien quand est-ce que j'aurais besoin de ce genre de chose lors de mes futurs codes. Ce projet a réelement provoqué pour moi une prise de conscience de ce qu'est la programmation, de pourquoi apprends-t-on autant une notion bien qu'elle ne nous parle peu.