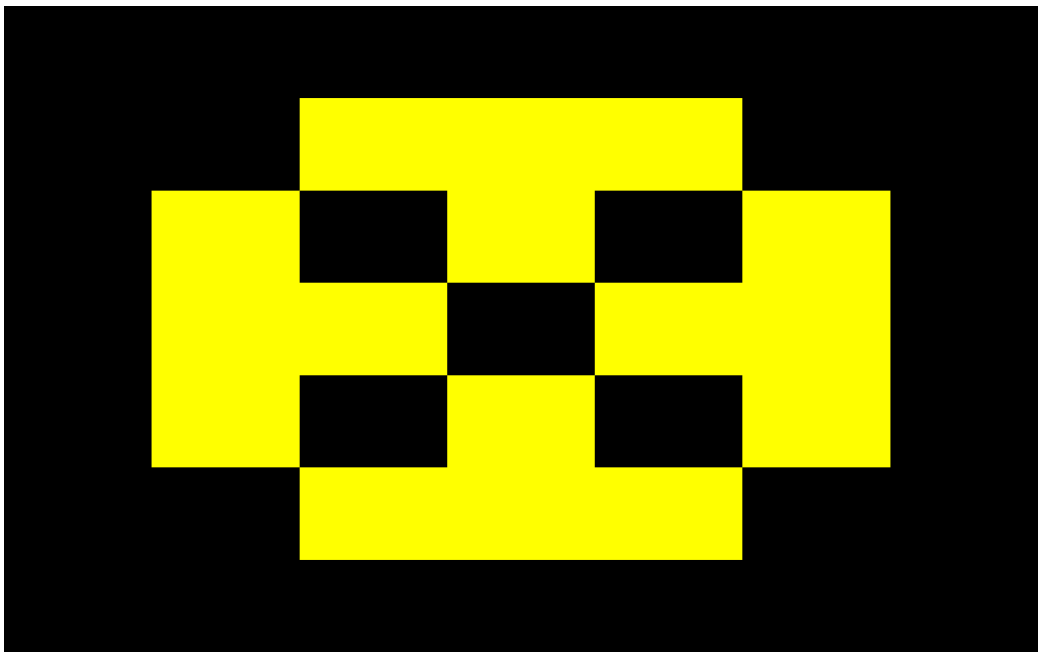
GAME OF LIFE



01/11/2016

LOPES Marco, ISELI Cyril, RINGOT Gaëtan

Travail pratique du cours de Programmation Concurrente,
sur le jeu de la vie.

Game of life

LOPES MARCO, ISELI CYRIL, RINGOT GAËTAN

Table des matières

I. GENERALITES :

Le but principal de ce travail pratique est de recréer une version du jeu de la vie, et ce en utilisant toutes les notions apprises durant le cours.

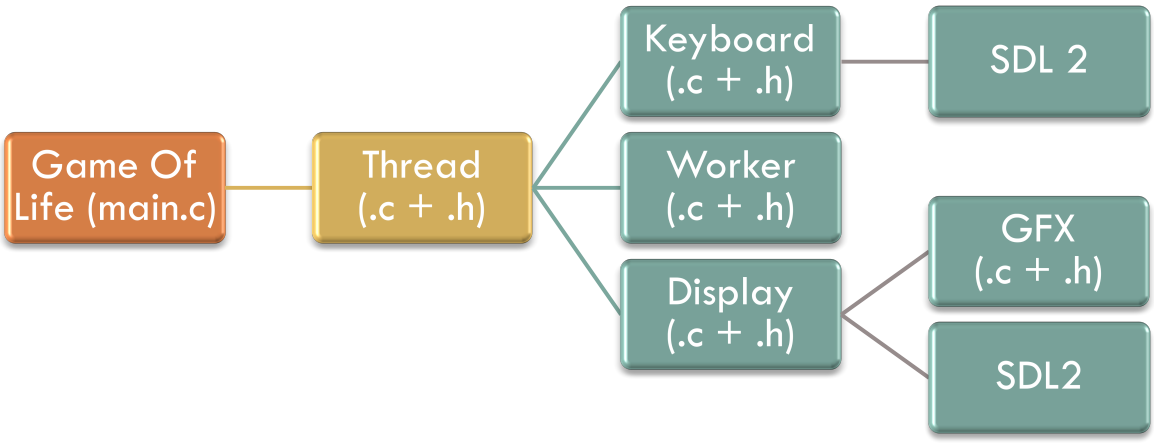
Règles du jeu de la vie :

Grâce à la figure ci à gauche, les règles sont par la suite plus simples à exprimées :

- Une cellule morte possédant exactement trois voisines vivantes devient vivante.
- Une cellule vivante possédant deux ou trois voisines vivantes reste vivante, sinon elle meurt.

II. LA DECOMPOSITION DU PROGRAMME ET DES TACHES :

1. Dépendances mutuelles pour le programme :



2. Répartition du travail :

❖ Tâches principales :

- Marco : keyboard + display.
- Cyril : workers.
- Gaëtan : main + makefile.

❖ Tâches communes :

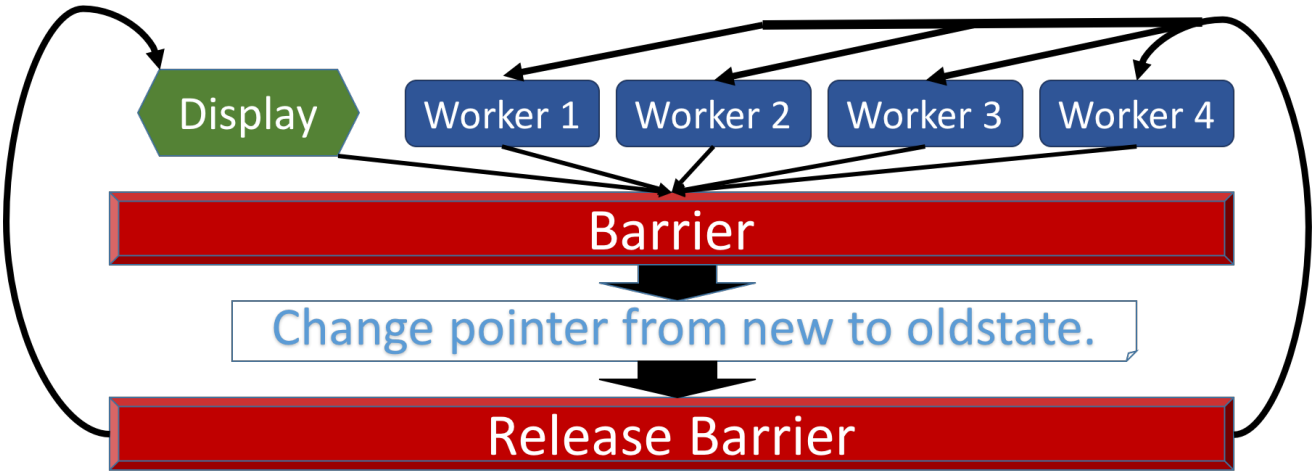
- Aide sur les problèmes rencontrés.
- Commentaires sur le code.
- Rédaction du rapport.

III. LE PROGRAMME PRINCIPAL ET PRINCIPE DE BARRIERES :

1. Programme principal :

Nous avons décidé de faire une gestion des arguments rentrés en ligne de commande et en vérifier la bonne syntaxe, un affichage final du temps, ainsi que la création des threads. Enfin, la première grille qui est un tableau a 2 dimensions, est créé grâce aux paramètres reçus.

2. Barrières au sein du programme :



Afin d'éviter la lecture du clavier avant l'initialisation de la SDL, nous avons utilisé une barrière, qui attend sur la fin de l'initialisation de celle-ci, avant de lire le clavier. Cela permet, d'éviter une erreur.

IV. LE CLAVIER ET GESTIONS DES ENTREES CLAVIER :

La fréquence de vérification de l'appui clavier est effectué grâce à un DEFINE, celle-ci est faite de cette façon car elle est imposée dans le cahier des charges.

Pour la fonction en question, celle-ci ne varie que peu de celle donné lors de l'exemple. En effet, la différence importante est le fait qu'une boucle sur tous les événements présents dans SDL_PollEvent, doit être effectuée, afin d'éviter une congestion des différents événements pendant la lecture, par exemple lors du déplacement de la souris.

V. LA PARTIE AFFICHAGE :

Celui-ci est simple, l'affichage s'effectue sans risque (les travailleurs modifient le nouvel état dans une autre grille). Pour cela, le thread parcourt chaque cellule, et la colorie par la couleur définie si celle-ci est vivante. Enfin, une barrière permet d'attendre que les travailleurs est finis le calcul de l'état futur.

VI. LA PARTIE TRAVAILLEURS (WORKERS) :

1. Paramètres :

Dans un premier temps, nous avions pris la décision de faire des INT pour les paramètres (comme la largeur (width), la hauteur (height), le nombre de Thread, etc …), cependant après relecture du cahier des charges, des UINT sont plus appropriés car ces valeurs sont obligatoirement strictement positives.

- bool *end : Permet de savoir que la touche Esc a été pressée.
- bool *quit : Evite le deadlock lors de la fermeture du programme (après pression sur Esc).
- bool ***oldState et bool ***actualState : permettant de faire une inversion des tableaux grâce au pointeurs, et non pas réécrire le nouvel état dans l'ancien.

2. Fonctionnement :

Chaque travailleur à une plage d'action qui est bien définie. En effet, afin de répartir équitablement la charge de travail, chaque thread travailleur, prend les valeurs avec un saut défini (par le nombre de threads).

Exemple, si notre grille fait 8 cellules et qu'il y a 2 travailleurs, le premier va faire la cellule 1,3,5 et 7, et l'autre travailleur fait 2,4,6 et 8. Ils ont donc bien une charge équitable.

Chaque travailleur a pour mission de calculer le nombre de voisins que la cellule possède. Par la suite, suivant l'état précédent, il en décide de l'état futur et le place dans la nouvelle grille (état futur).

VII. LE TEMPS DANS LES DIVERSES PARTIES :

Afin de faciliter la gestion des attentes grâce à la fonction usleep(), chaque valeur est ramenée dans cette unité, par exemple, le temps donné par la structure « timespec » qui est en micro secondes doit être divisé par 1000 pour finir en nano secondes.

Enfin lors de la fin du programme, un affichage du temps d'exécution est effectué dans la ligne de commande.

VIII. CONCLUSION :

Nous avons pu observer que la librairie SDL, n'est pas entièrement prévue pour être en multithread et doit par conséquent être adapté, ce qui n'est pas toujours flagrant au premier abord. En effet, suivant les machines, des erreurs peuvent apparaître ou non (cas des machines de l'école).

Le principe du programme étant le même que celui du password cracker, la programmation a été plus simple, et l'ajout des barrières permettant la synchronisation des threads, par attente passive.