Projet de programmation C : Three to go !!!

Licence 2e année - 2017/2018

MALEK Akram
GEOFFREY Lesergent

Presentation

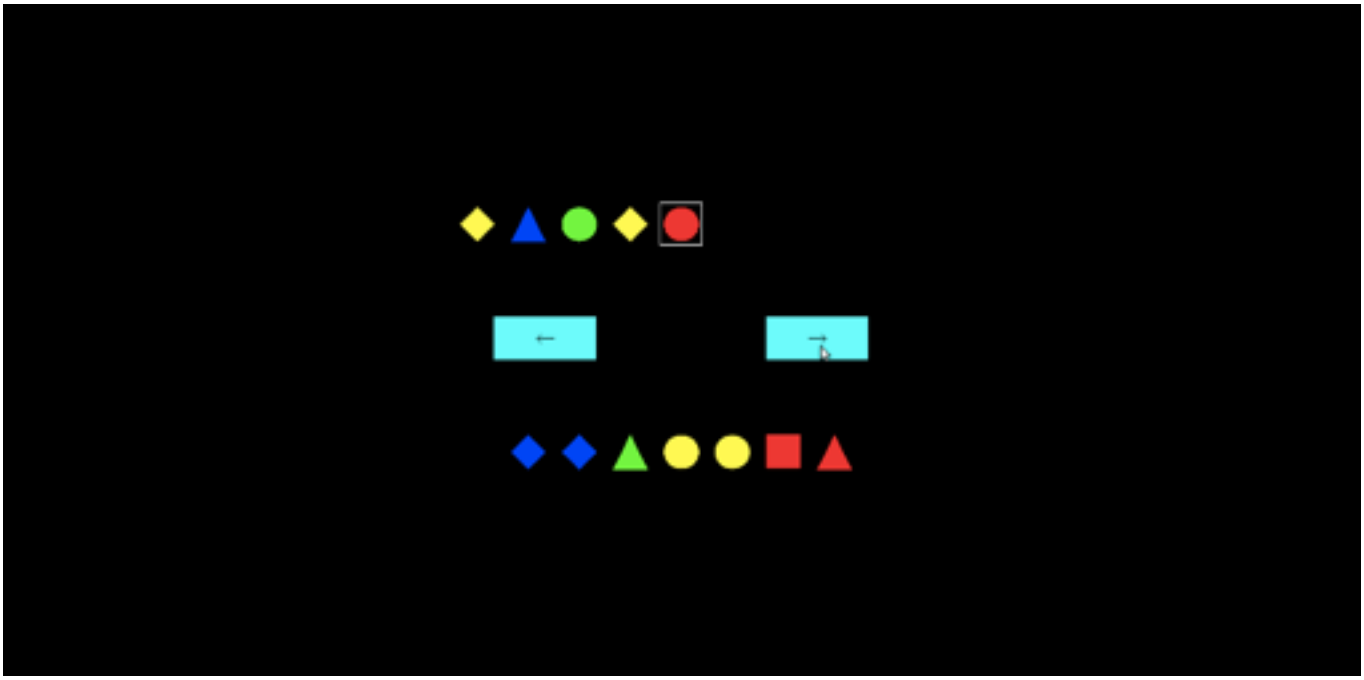
L'objectif de ce projet est de développer une application graphique d'un jeu ou l'utilisateur doit produire des motifs répétés avec des petits tokens caractérisés par une forme et une couleur.

Jeu :

- Les formes sont au nombre de 4 : Carré, rond, triangle, diamant.

- Les couleurs sont au nombre de 4 : Rouge, bleu, vert, jaune.

Au début du jeu, un certain nombre de tokens (fixé à la compilation, par défaut 5) est affiché dans une liste en haut de l'écran. On peut, à l'aide de deux boutons, faire descendre le dernier token de cette liste et l'ajouter à gauche ou à droite de la liste du bas. A chaque fois qu'un token descend, la liste du haut est poussée vers la droite et un nouveau token y est ajouté.



Si trois tokens ou plus ont une forme où une couleur en commun, ils disparaissent. On peut aussi cliquer sur un token du bas pour afficher un menu permettant de faire des décalages. Le bouton du haut décale circulairement les tokens de couleur similaire vers la gauche, tandis que le bouton du bas décale circulairement vers la gauche les tokens de forme similaire. Ces actions peuvent aussi produire des enchaînements de tokens.

Le jeu ne peut être gagné. En effet le but est de faire le score le plus grand, ce qui devient de plus en plus difficile à mesure que la partie s'éternise. Toutefois, une limite de deux minutes (fixée à la compilation) entre chaque clic est imposée afin d'éviter les parties trop longues.

Documentation technique :

Quand le programme se lance, il entre dans une boucle. Cette boucle n'est quittée qu'en cas d'erreur d'allocation ou lorsque la partie se termine. Au début de celle ci, on regarde s'il manque des tokens à la liste du haut, et on les crée si besoin est. Ensuite, on teste les tokens successifs, d'abord à gauche puis à droite. On les supprime le cas échéant, et on ajoute des points au joueur. Enfin, le programme dessine les listes et le menu (les deux boutons fléchés, le score et un carré autour du dernier token en haut). La partie interactive commence maintenant. Le programme attend au plus deux minutes pour un clic, puis détermine sur quoi ce clic a été fait. Si c'est sur un bouton, on fait descendre le token puis on passe au tour de boucle suivant. Si c'est sur un token du bas, il faut afficher le menu (c'est à dire un token de même couleur mais de forme différente au dessus, un token de même forme mais de couleur différente en bas), puis attendre à nouveau un clic. Si ce dernier clic est sur une des deux formes dessinées tantôt, on fait l'action correspondante (le décalage) et on passe au tour de boucle suivant. Si un des clics est invalide, on l'ignore simplement.

Les tokens du bas sont dessinés avec une taille dynamique : Si leur nombre le permet, ils sont agrandis. S'il y en a trop, ils sont rétrécis.

Les fonctions :

- Pour manipuler la liste des tokens, plusieurs fonctions ont été écrites :

libere_token_debut et *libere_token_fin* font sortir le premier et le dernier token d'une liste, respectivement, et en renvoient le pointeur. Pour ce faire, on doit calculer le token précédent et le token suivant celui qui va être supprimé. On modifie les pointeurs du précédent afin de pointer directement sur le suivant. Idem pour les formes et les couleurs.

Insere_token_debut et *insere_token_fin* font l'opération inverse. Pour les formes et les couleurs, il faut alors trouver le token précédent de la même couleur, et le token suivant de la même couleur, pour actualiser le chaînage.

Comme la liste ne possède pas de pointeurs vers le token précédent, une fonction le fait à la place. Elle itère sur les tokens suivants jusqu'à ce que le prochain soit le token de départ. Nombre_tokens est trivialement implémentée.

decaler_gauche_forme3 et *decaler_gauche_couleurs3* sont similaires. On passe au token suivant de même forme / couleur et on lui donne la valeur de son suivant. Par souci de simplicité, on modifie directement les valeurs. Ceci a l'avantage de ne pas poser de problèmes de pointeurs à actualiser, notamment si on décale le pointeur de liste. Cependant, il faut alors refaire le chaînage opposé : si on décale les couleurs, on doit refaire le chaînage des formes, et vice-versa. Les fonctions refait_chainage_couleurs et refait_chainage_formes s'en occupent. Elles itèrent sur tous les tokens successivement, et regardent tous les autres tokens via le pointeur suivant. Si la forme / couleur est identique, on actualise les pointeurs de couleurs / formes.

Liberer_liste est trivialement implémentée.

Bugs et fonctionnalités manquantes :

Les tokens consécutifs au milieu de la liste ne sont pas supprimés. C'est du au fait que les fonctions de libération des tokens ne fonctionnent qu'au début et à la fin de la liste. Une piste serait de créer une liste à part et d'en libérer la fin, puis de fusionner ces listes.

Valgrind reporte une utilisation incorrecte de la mémoire dans la version graphique. Cependant, même avec un programme vide (la seule ligne du main est return 0), importer la bibliothèque MLV semble provoquer des allocations qu'elle ne libère pas. (4 allocations et 2 free pour un main vide, si on multiplie par le nombre de lignes du programme, on obtient beaucoup de mémoire perdue).