|  |
| --- |
|  |
| DM Programmation C |
| Mots-mêlés |

|  |
| --- |
| Clément GAUDET / Maxime JAILLARD  19/11/2020 |

|  |
| --- |
| L2 – ProgC |
| Projet Programmation C |
| Threetogo |

Clément GAUDET / Maxime JAILLARD

16/01/2021

Sommaire

[Manuel utilisateur 3](#_Toc61604034)

[Listing des fonctions du programme 3](#_Toc61604035)

[token.c 3](#_Toc61604036)

[moteur.c 3](#_Toc61604037)

[graphique.c 4](#_Toc61604038)

[threetogo.c 4](#_Toc61604039)

[Architecture du site 5](#_Toc61604040)

[Arborescence des pages 5](#_Toc61604041)

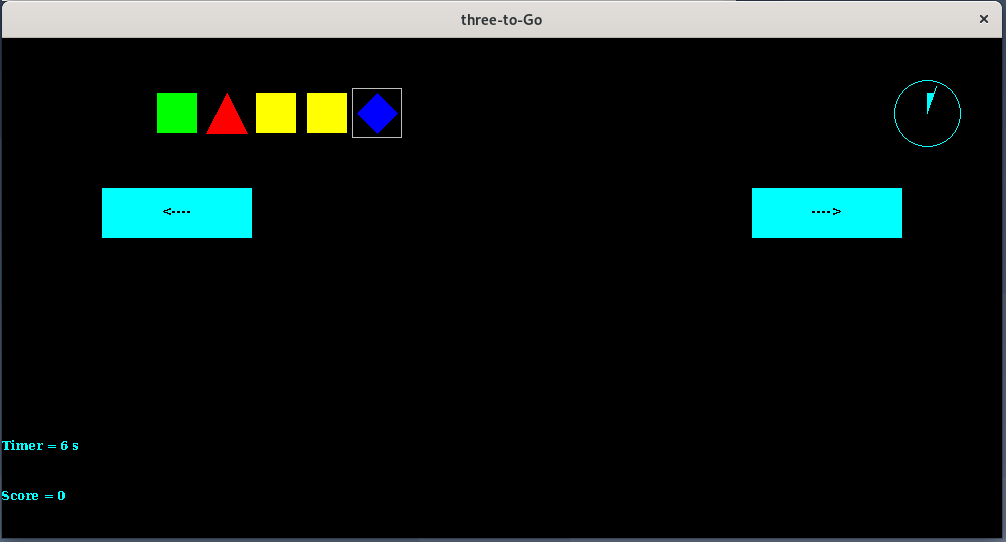
[Requêtes SQL majeures 5](#_Toc61604042)

[Pistes d’amélioration 5](#_Toc61604043)

# Manuel utilisateur

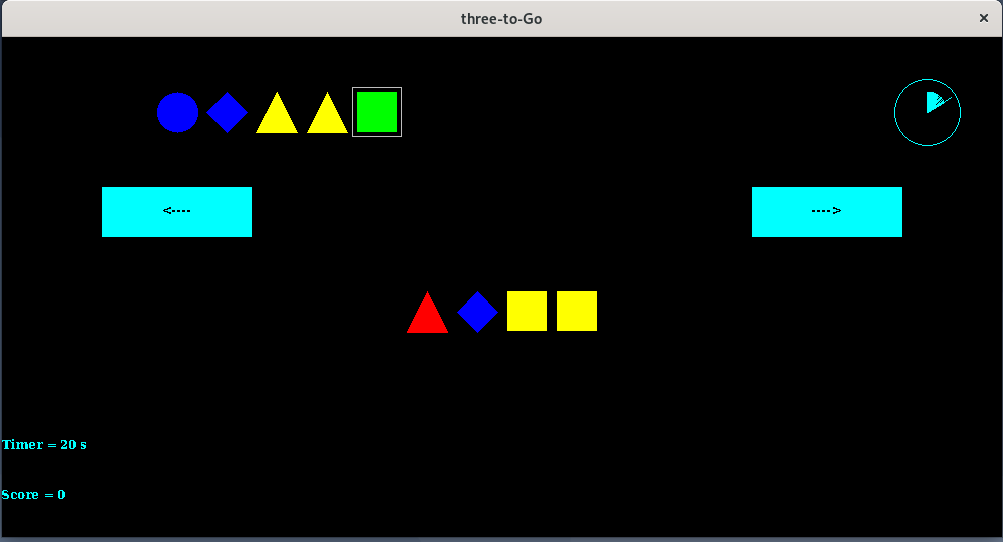
Pour compiler le programme, on se place dans le dossier racine du jeu threetogo et on utilise la commande « make » dans le shell.

Une fois le programme exécuté, la partie commence immédiatement et le timer se met en marche.



En haut à gauche de l’écran, on retrouve la « **queue** » de jetons dont un jeton encadré qui représente le prochain jeton à ajouter.

En dessous, on retrouve le bouton d’**ajout à gauche** et celui d’**ajout à droite**. En cliquant sur un de ces boutons, le jeton encadré dans la **queue** sera ajouté sur le côté choisi dans la ligne du dessous.



En cliquant sur un des jetons de la ligne du bas, on peut procéder à des décalages de formes ou de couleurs dans l’alignement suivant les règles du threetogo.

En bas à droite de la fenêtre, on retrouve le **timer numérique** (en secondes) et le **score** du joueur. Pour savoir combien de temps il reste au joueur, une **horloge** est affichée en haut à droite de la fenêtre et avance en même temps que le timer.

Le but du jeu est de former des combinaisons de 3 jetons ou plus de même couleur ou de même forme pour gagner des points (100 points par jetons dans la combinaison) pendant le temps imparti pour obtenir le plus haut score possible.

# Fonctionnement d’une boucle

## Rafraichissement de l’affichage

On ne rafraîchit l’affichage qu’une seule fois par occurrence de la boucle. Toutes les fonctionnalités d’affichage sont regroupées dans la fonction **refresh\_screen** du fichier ‘graphique.c’. Cette fonction commence par effacer tous les dessins de la fenêtre et applique la couleur noire à la totalité de la fenêtre. Elle redessine ensuite dans l’ordre :

* La queue et son cadre de sélection
* Les boutons d’ajout
* La liste des enchaînements de jetons
* Le timer et le score du joueur
* L’horloge

Seulement une fois tous ces dessins effectués, on fait appel à la fonction **MLV\_actualise\_window**. En pratique, cela évite que l’interface et les dessins ne « flashent » lorsque l’on tente de faire fonctionner le jeu à un taux d’affichage par seconde agréable (de l’ordre de 50-60 images par secondes).

La fonction **refresh\_screen** tient également compte de l’état du jeu en utilisant les coordonnées du dernier clic enregistré et le nombre de jetons dans la ligne du dessous. Si le clic a eu lieu sur l’un des jetons de la liste, celui-ci sera mis en évidence lors du rafraîchissement par l’ajout des jetons de démonstration au-dessus et en dessous pour représenter les décalages. Si le nombre de jetons dans la liste atteint le nombre maximum déterminé par la constante **MAX\_TOKENS**, les boutons d’ajout ne seront pas dessinés.

## Gestion et interprétation des événements

/\*\* A COMPLETER \*\*/

## Mise à jour des informations de la partie

Pendant l’exécution de la boucle, on met à jour la composition de la queue de jeton ainsi que celle de la liste des ajouts. Lorsqu’un clic est enregistré sur l’un des boutons, si le nombre de jetons alignés est strictement inférieur à **MAX\_TOKENS**, on appelle la fonction adéquate **add\_left** / **add\_right** pour rattacher le jeton à la liste. On incrémente le nombre de jeton dans la liste de 1 et on génère un nouveau jeton pour la queue. Lorsqu’un clic est enregistré sur un des jetons de démonstration pour produire un décalage, on fait appel à la fonction **shift\_commoncolor\_left** / **shift\_commonshape\_left** pour mettre à jour la composition de la liste en accord avec le décalage demandé.

En fin de boucle, on vérifie si des combinaisons de 3 jetons ou plus de la même couleur/forme sont présentes dans la liste avec **check\_combinations** et, si c’est le cas, la fonction les extrait de la liste et renvoie le nombre de points gagnés. Ce nombre est ajouté au paramètre score de la partie et le nombre de jetons dans l’alignement est mis à jour en conséquence.

# Limites du programme

## Fonctionnalités implémentées

/\*\* A COMPLETER \*\*/

## Problèmes non-pris en charge

/\*\* A COMPLETER \*\*/

# Répartition du travail

Maxime Jaillard :

Je me suis concentré sur l’interface graphique du jeu (dessin des jetons, des boutons, du timer, de l’horloge et affichage du score). On réutilise des fonctions logiques utilisées dans la version à affichage ascii du jeu pour contrôler l’état des différents paramètres de la partie.

La première étape du travail a été de créer plusieurs fonctions et structures élémentaires. Découper la fenêtre en cases semblait plus évident pour travailler de manière plus lisible sur l’affichage des dessins. J’ai donc commencé par créer une structure **Case** et la fonction **mouse\_to\_square** pour transformer tous les clics dans la fenêtre en clics de zone. Une fois que la gestion des coordonnées était terminée, la partie suivante était centrée sur l’interprétation de ces coordonnées et répercuter les actions de l’utilisateur sur la queue et la liste de jetons ajoutés.

Il m’a fallu beaucoup de temps et l’aide de Clément pour la partie gestion des événements. Trouver le moyen de gérer sans la moindre latence chaque occurrence de la boucle après chaque clic de l’utilisateur a été moins évident que je le pensais. Il a fallu explorer la librairie MLV pour pouvoir gérer le framerate et comprendre le format et l’utilité des différents arguments de ces fonctions.