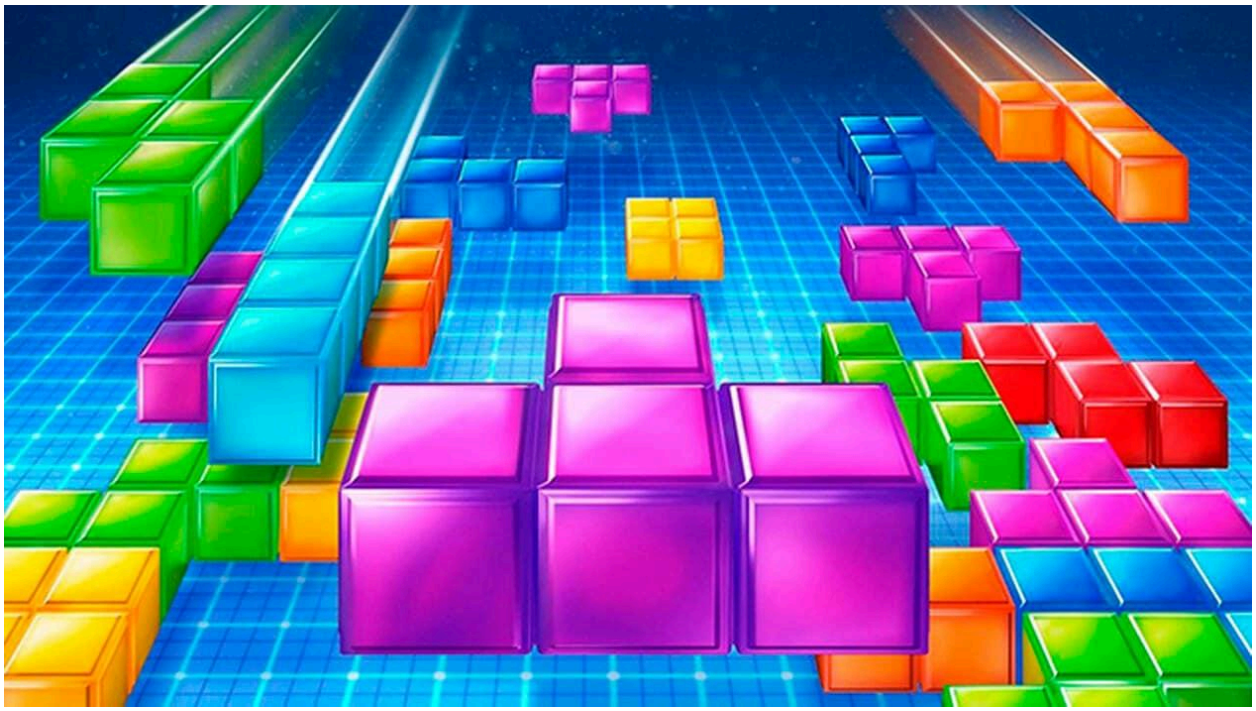
Livre Blanc : Tetris en SDL

Nathan CAVA-LONCHAMP, SIO1

J'ai choisi le Tetris comme jeu à développer. Ce livre blanc va détailler toutes les fonctions utilisées dans le code ainsi que les problèmes rencontrés.



Tetris, un jeu célèbre qui a marqué l'enfance de bon nombre de personnes (moi y compris) et qui m'a motivé à le recréer en SDL.

1/ Génération de blocs

Tout bon Tetris commence par une génération de blocs. Pour les générer, je suis parti du modèle suivant:

-Une structure 'bloc' gérant la couleur et l'état du bloc;

-Une structure 'forme' gérant la forme, la couleur et les positions du bloc ainsi qu'une matrice de booléens.

Ces deux structures ont pour but de gérer les tetriminos en général.

La structure forme est utilisée pour un tableau 'blocs' de 7 éléments pour les 7 blocs différents.

La structure bloc détermine si le bloc est passif ou non, utile pour la génération continue de pièces et le rendu de la pièce inactive sur la grille.

Pour générer ces blocs, une fonction va ensuite parcourir le tableau de booléens de la structure forme; en cas de true, on affiche un bloc avec la couleur du bloc correspondant.

Les positions x et y seront également gérées par la structure.

Difficultés rencontrées :

Cette étape était sans doute la plus facile à remplir. Aucune difficulté particulière au niveau du rendu.

2/ Mouvement des blocs

Tout ça, c'est bien joli, mais maintenant il faut trouver un moyen de déplacer nos blocs. Il faut implémenter quatre options:

- ***Aller vers le bas;***
- ***Aller vers la gauche;***
- ***Aller vers la droite;***
- ***Faire tourner la pièce.***

Pour commencer, voyons comment faire tourner un tetromino. L'algorithme que j'ai utilisé se déroule en deux étapes :

1- Transposer la matrice

2- Inverser l'ordre des lignes.

Pas très clair à vue d'œil… Cependant, pas besoin de développer car une image vaut mille mots:

Matrice originelle:

1 2 3

a [0 1 0]

b [0 1 0]

c [1 1 0]

(Correspond à un bloc L)

Après transposition:

a b c

1 [0 0 1]

2 [1 1 1]

3 [0 0 0]

Après inversement:

a b c

1 [0 0 0]

2 [1 1 1]
3 [0 0 1]

Et voilà le travail! Nos pièces peuvent être tournées (pas avant leur avoir associé une touche vai SDL, bien sûr, mais le principe est prêt)

Pour ce qui est du déplacement latéral, on augmente ou diminue l'axe x de la pièce en fonction de la touche pressée (gauche ou droite).

Pour se déplacer en bas, on incrémente le rang y du tetromino lorsque l'on presse la touche du bas.

Difficultés rencontrées :
Le plus dur n'était pas d'implémenter l'algorithme mais de le comprendre. Même avec visualisation, cela paraissait compliqué et il est difficile de créer quelque chose que l'on ne comprend pas.

3/ Collisions avec la grille

Il faut maintenant s'assurer que la pièce ne sorte pas de la zone de jeu. Pour se faire, une fonction de vérification est mise en place.

Pour faire simple, la fonction prend la position x & y de la pièce et vérifie les prochaines coordonnées après mouvement de la pièce, avant de la faire vraiment bouger.

Par exemple, admettons que l'on soit au bord de la zone jouable et que l'on souhaite se déplacer encore plus à droite. La fonction 'peut_deplacer' calcule les prochaines coordonnées du déplacement avant, et détectera un dépassement de limite; il n'y aura pas de déplacement.

Il est évident que le système prend également en compte le cas où le tetromino touche le sol.

Difficultés rencontrées :
Le système que j'utilisais auparavant ne se servait pas des coordonnées suivantes mais des coordonnées actuelles, ce qui posait certains problèmes au niveau de la gestion.

Cela a été réglé, mais je n'ai malheureusement pas réussi à corriger un bug qui permettait aux pièces de dépasser les bordures latérales non pas par déplacement mais par rotation.

4/ Collisions entre blocs

Nos blocs peuvent maintenant être confinés dans leur zone, mais reste encore à gérer les collisions entre blocs. Pour ce faire, il faut d'abord générer de nouveaux blocs continuellement lorsqu'on atteint le fond avec un bloc. (*if peut_deplacer*)

Tout d'abord, j'ai ajouté une fonction permettant de faire naturellement tomber le bloc toutes les demi-secondes grâce à SDL_GetTicks, très utile pour les taux de rafraîchissement et autres.

Si le bloc atteint le bas (*else*), on fait appel à une autre fonction, placer_bloc, qui va "fixer" le bloc dans la grille et invoquer encore une autre fonction, generer_bloc. Cette fonction va aléatoirement choisir l'une des 7 pièces disponibles et l'initialiser en haut et au milieu grâce à x et à y.

Maintenant que nos blocs sont générés en continu, on gère les collisions entre blocs. L'IA m'a permis de lier un tableau de booléens (lié à la structure bloc) à la grille, ce qui permet en quelque sorte "d'enregistrer" toutes les pièces tombées dans la grille. On bricole alors la fonction 'placer_bloc' pour pouvoir gérer les collisons entre blocs.

Difficultés rencontrées :
Cette étape était de loin la plus ardue pour moi. Il m'est difficile de parvenir à comprendre les systèmes de collision utilisés mais au moins tout est un peu plus clair pour moi.

5/ Effacement des lignes pleines

Pour effacer les lignes pleines, on doit parcourir la grille pour vérifier s'il y a une ligne remplie de blocs passifs.

Pour chaque axe y, on regarde une ligne x entière; si un seul des blocs est vide et non pas un tetromino passif, il n'y pas de ligne pleine.

S'il y en a une, on retire l'état passif aux éléments de la ligne, les faisant disparaître de la grille. On décrémente l'axe y de tous les éléments au-dessus de la ligne effacée pour les faire retomber au bon niveau.

Difficultés rencontrées :
Une étape ni trop compliquée, ni trop simple.

Note personnelle

Ce projet a été particulièrement compliqué à réaliser en sachant que mes compétences en SDL et en C étaient relativement limitées; de plus, le seul tutoriel que j'avais choisi pour m'aider était incomplet, il ne contenait pas de guide pour l'effacement des lignes, la génération de la grille, les collisions, etc. Cependant, j'ai appris plein de concepts intéressants et ma méthode de réflexion a été mise à l'épreuve face aux méthodes proposées par l'IA. Il est donc opportun de dire que ce projet a été un pas en avant pour moi.