Rapport de projet : Les Blocs Qui Tombent, par Mathias et Barnabé

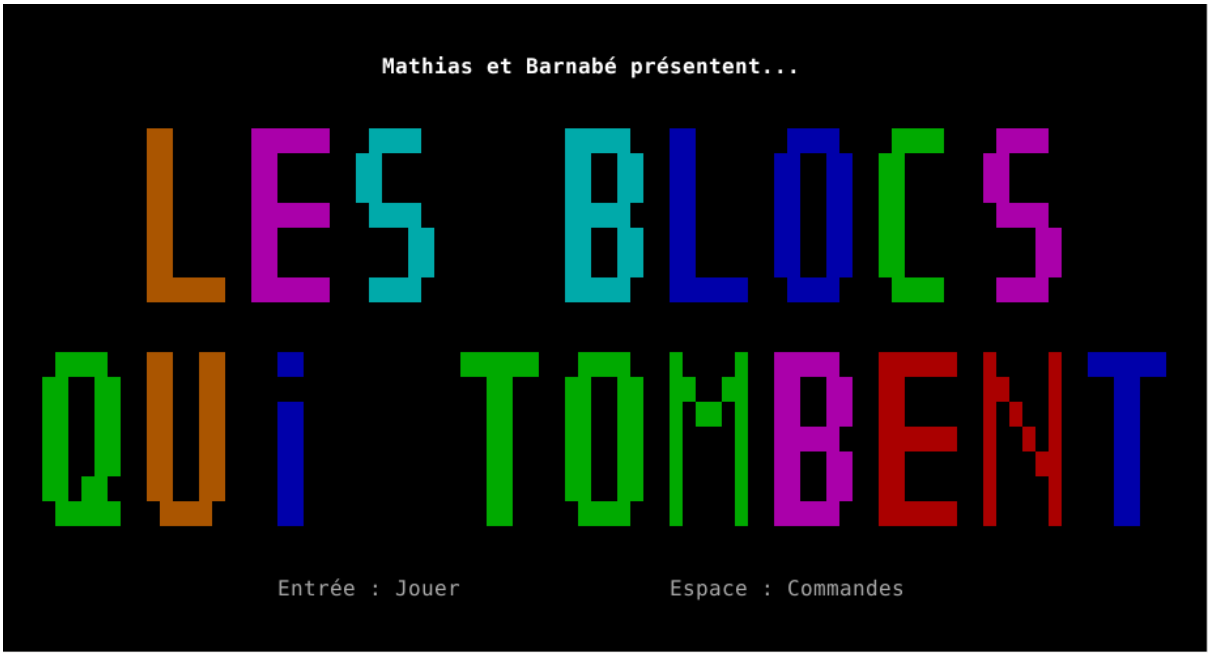
Le but du projet est de créer une version du célèbre jeu Tetris, pouvant se jouer dans le terminal, à l'aide de la librairie ncurses. Le jeu se présente sous la forme d'une grille de 20 lignes et 10 colonnes, dans laquelle diverses formes composées de quatres blocs et appelées "tetrominos" tombent du haut de l'écran et viennent s'amasser en bas de la grille. Lorsque la pile de blocs ainsi formée devient trop haute et atteint le haut de la grille, le joueur perd. Son but est donc de faire disparaître ces blocs, et il doit pour cela compléter des lignes horizontales. Ce faisant, les lignes complétées disparaissent et le joueur gagne des points ; s'il parvient à faire disparaître plusieurs lignes en même temps, il gagne davantage de points. Quoi qu'il en soit, le joueur ne peut pas gagner et son but est donc de faire le score le plus élevé possible.
Plus le joueur a supprimé de lignes, plus son niveau augmente, ce qui a pour effet de rendre le jeu plus difficile en faisant tomber les tetrominos plus vite, mais aussi d'augmenter le nombre de points obtenus lors de la suppression de nouvelles lignes.
Dans notre version du jeu, les 4 prochains tetrominos sont visibles, il est donc possible pour le joueur de savoir à l'avance quels tetrominos il devra placer et donc d'optimiser leur placement. En outre, une réserve a également été implémentée, et elle permet au joueur de stocker un tetromino, qu'il peut échanger à tout moment avec le tetromino en cours de chute.

Voici un aperçu des objectifs que nous nous sommes fixé :

Fonctions	**Critères**	**Niveaux**
Jouabilité	Descente instantanée	Oui
	Descente rapide	Oui
	Réserve	Utilisable une seule fois par tetromino
	Ecran de fin proposant de rejouer	Oui
	Lectures des commandes fréquente	Environ 60 fois par secondes
Graphisme	Couleurs	Semblables au jeux de base
	Esthétisme	Les blocs ne doivent pas être des lettres Les interfaces graphiques doivent être compréhensibles
	Présence d'un écran titre	Oui
Fonctionnalité	Difficulté progressive	évolue lorsque l'utilisateur supprime des lignes
	Score	Augmentation dépendant du niveau Augmentation plus importante si plusieurs lignes sont détruite en une seule fois
	Commandes personnalisables	Oui
Interface de jeu	Score visible	Oui, 6 chiffres
	Nombre de ligne détruite visible	Oui, 3 chiffres
	Réserve visible	Oui
	Niveau de jeu visible	Oui, 2 chiffres
	Visibilité des tetrominos suivants	4 tetrominos

Dans notre jeu, la grille est représentée par un tableau à deux dimensions. Ce tableau contient des entiers qui représentent les différents blocs composant les tetrominos. Cet entier code à la fois la couleur du bloc et la différence entre un bloc “occupé” et un espace “vide”.
Les tetrominos sont représentés par des tableaux de 4 par 4 entiers qui représentent la position des blocs. Plus précisément, une structure “tetromino” contient toutes les informations relatives à chaque tetromino, c’est-à-dire les coordonnées où il doit apparaître, le nombre de rotations possibles, les tableaux contenant l’arrangement des blocs qui le composent à chaque rotation… Lorsqu’ils doivent être affichés, ils sont directement "copiés" dans la grille à la place qu’ils doivent occuper
Une fonction boucle_jeu exécute 60 fois par seconde une boucle qui fait descendre le tetromino et effectue les actions correspondant aux entrées du joueur.

Écran titre :
L’écran titre comporte simplement le nom du jeu et les premières commandes. Le nom du jeu témoigne de notre manque d’inspiration pour les tâches littéraires.



Les lettres du titre changent de couleur aléatoirement et indépendamment les unes des autres, au rythme de 5 changements par seconde. Elles ont été dessinées à la main, faute d’avoir trouvé un moyen d’afficher des “grandes lettres” avec ncurses, et le peu de place disponible pour chaque lettre (8x9 caractères) explique leur aspect parfois irrégulier et un peu tordu.
Leur changement de couleur aléatoire est réalisé avec un tableau de 22 entiers contenant les couleurs des emplacements des 22 caractères du titre (ce qui inclut les espaces). 5 fois par seconde, un de ces entiers est modifié aléatoirement, ce qui

a pour effet de modifier la couleur d'une lettre si cet entier correspond à une lettre. S'il correspond à un espace, aucun changement n'est visible à l'écran.

Menus et commandes

Pour rendre le jeu plus jouable, nous avons créé un interface permettant à l'utilisateur de modifier les commandes qu'il utilise dans le jeu. Cette interface est accessible depuis l'écran titre. Les réglages qui sont choisis par l'utilisateur sont sauvegardés sous la forme d'un fichier keybind.bin. Il a donc été nécessaire de créer une interface pour permettre à l'utilisateur de modifier les commandes. Pour ce faire deux nouvelles structure ont été créées :

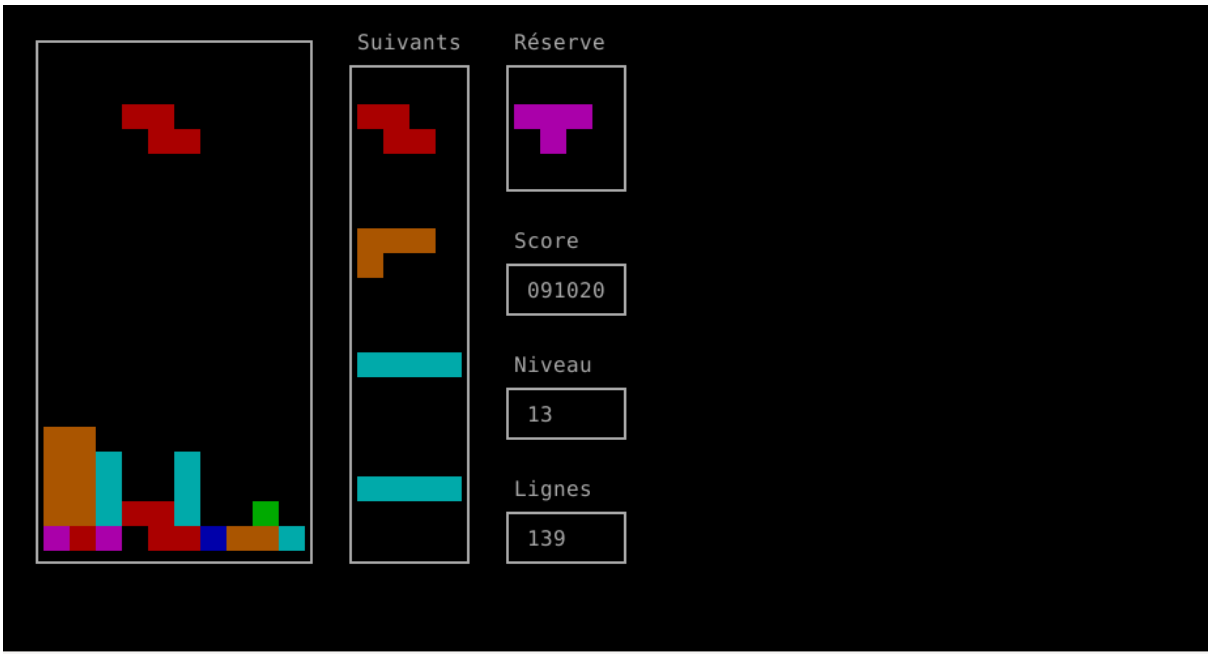
- Une structure "menu" : Elle contient toutes les informations relatives au menu : une chaîne de caractères qui est son titre, le nombre de boutons, et de texte qu'il contient la liste des boutons et des texte qu'il contient, lequel des boutons est sélectionné etc. Certains de ses champs ont des tailles variables et sont donc alloués dynamiquement dans la mémoire fonctions comme "addbutton" ont donc dû être créées pour une gestion propre de la mémoire.

- Une structure : "button" : Cette structure est une représentation informatique d'un bouton, elle contient notamment le texte qu'il affiche, sa position, son identifiant, ainsi qu'un pointeur vers la fonction qui doit être appelée lorsque le bouton est cliqué. Cette structure est créée par la fonction "addbutton".

- Une structure "textbox" : La textbox est un champ de texte. Il est très similaire au bouton, les seules différences sont que la "textbox" ne peut pas être sélectionnée par l'utilisateur.

Quand l'un des boutons du menu est cliqué par le joueur, il lui est demandé d'appuyer sur une touche qui sera la nouvelle commande. Pour ce faire, on utilise un pointeur vers une fonction, en l'occurrence vers la fonction onclick qui permet d'effectuer des actions quand le bouton est cliqué. La fonction onclick prend en paramètre un entier et un pointeur vers le menu ce dernier permet à la fonction d'avoir le contexte dans lequel elle est appelée, le premier lui sert simplement à faire passer l'identifiant du bouton cliqué.
Lorsque le menu est lancé une fonction "menuloop" est lancée elle permet de capter les interactions avec le menu. Lorsque cette fonction se termine, la mémoire allouée dynamiquement est nettoyée pour éviter toute fuite mémoire. La fonction display quant à elle se contente d'afficher le menu que l'on lui donne en argument.

Graphismes :
Les graphismes du jeu ont été réalisés à l'aide de la librairie ncurses, qui permet de créer simplement des graphismes en 2D dans le terminal.



Pour afficher les blocs, une procédure affiche_grille récupère des entiers dans le tableau à 2 dimensions "grille", ces entiers peuvent correspondent soit à la couleur d'un bloc ou à l'absence de bloc. La même technique a été utilisée pour afficher les 4 blocs suivants et la réserve, en récupérant les informations dans d'autres tableaux. Lors du jeu et en fin de partie, le score du joueur, ainsi que son nombre de lignes et le niveau atteint, sont affichés, avec un certain nombre de chiffres, récupérés dans des variables entières du "gamestate". Ce nombre de chiffres est toujours le même, même si la valeur est faible, et cela peut causer un mauvais affichage du nombre si sa valeur dépasse 10^n. Ce cas peut arriver si le score atteint un million ou si le nombre de ligne atteint 1000, mais c'est dans la pratique quasiment impossible.

Les mouvements et déplacements du tetromino, et leur validité :
Lors du jeu, le tetromino courant descend à chaque fois qu'un certain nombre de ticks (c'est à dire d'itérations de la boucle de jeu) est atteint. Le nombre de ticks entre chaque descente dépend du niveau dans lequel se trouve le joueur. Avant chaque descente, on vérifie que la nouvelle position est valide, c'est-à-dire que le tetromino ne rentre en collision avec rien. Lorsque la descente est impossible, le tetromino est figé dans sa position actuelle, on vérifie si des lignes ont été complétées, puis on fait apparaître un nouveau tetromino.

Pour déplacer un tetromino, que ce soit lors de sa descente automatique ou suite à une action du joueur, il a été nécessaire de vérifier si le déplacement est possible, c'est à dire que le déplacement du tetromino n'est pas bloqué par les bords de la grille ou par des blocs déjà contenus dans celle-ci. Pour effectuer cette vérification,

un état du jeu temporaire est généré, puis vérifié par la procédure mouvement_valide. Si le déplacement est possible, il est effectué en remplaçant le précédent état du jeu par l'état temporaire venant d'être vérifié.

Les différentes positions des blocs de chaque tetromino lors de ses rotations sont enregistrées sous la forme d'un tableau d'entiers. Les rotations ne font ainsi pas réellement "tourner" le tetromino, elles changent le tableau dans lequel celui-ci est lu. Il est à noter que pour les tetrominos n'ayant pas un aspect différent à chaque rotation (par exemple le carré de 2x2, qui ne change pas par rotation), un seul tableau est utilisé sur les 4 possibles.

Le statut de jeu (ou "gamestate") :
C'est une structure comprenant toutes les informations relatives au jeu comme la vitesse de descente des tetrominos, la position du tetromino du joueur, la rotation du tetromino, les blocs de la réserve, le score…
Cet objet est le cœur du jeu, il est utilisé par la plupart des fonctions, qui ont ainsi accès à toutes les variables du jeu. De fait, quand une modification du gamestate est à faire, on utilise un gamestate temporaire dont on vérifie la validité avant de remplacer celui du jeu en cours.

Suppression et comptage des lignes :
Lorsque le joueur réalise des lignes complètes, il est nécessaire de les supprimer afin que la partie puisse continuer, mais également de les compter afin de pouvoir augmenter la vitesse du jeu. La fonction nettoie_lignes s'occupe de ces 2 tâches, elle est appelée juste avant l'apparition de chaque nouveau tetromino, et elle vérifie si des lignes de la grille sont pleines. Lorsqu'une ligne est pleine, elle la supprime et fait descendre d'un rang toutes les lignes supérieures. Après avoir vérifié l'intégralité des 20 lignes de la grille, elle renvoie un entier correspondant au nombre de lignes supprimées lors de cet appel, qui est ensuite ajouté au compte total de lignes situé dans l'état du jeu.

La vérification d'une ligne de 10 blocs se fait en vérifiant d'abord le premier bloc, puis en vérifiant le suivant uniquement si le précédent est plein, jusqu'au dixième. Ainsi, pour une ligne entièrement vide, l'algorithme ne perd pas de temps à vérifier si les 10 blocs de la ligne sont pleins, puisqu'il arrête les vérifications après avoir testé le 1er bloc, qui est vide.

Conclusion :
Notre projet correspond aux exigences que nous nous étions fixées et le jeu est jouable. Pour améliorer le projet, nous avions pensé à rajouter des animations lors de la suppression des lignes, ainsi qu'une "ombre" en dessous du tetromino courant pour faciliter l'utilisation de la descente instantanée. Nous aurions également aimé rajouter des effets sonores, cependant nous avons été limités par les possibilités offertes par les librairies standard de C.