Projet : Tectris

I. Répartition des tâches et rôles

Riyane

- Chargement des pièces à partir d'un fichier texte
- Création des fonctions de rotation des pièces
- Affichage de la grille et génération de la grille initiale
- Développement des fonctions de gestion des scores (enregistrement, chargement, tri)
- Conception et intégration du menu principal
- Intégration du timer avec précision à la milliseconde
- Modularisation du code en plusieurs fichiers .c

Salim

- Implémentation de la suppression des lignes pleines
- Décalage dynamique des lignes supérieures
- Synchronisation de la grille après modification
- Correction d'un bug d'affichage appelé "pièces flottantes"

Mehdi

- Conception et débogage de la fonction poser_piece
- Gestion des entrées erronées utilisateurs
- Mise en place de la fonction viderBuffer()
- Détection et gestion des dépassements verticaux des pièces
- Rassemblement de fonction pour ensuite crée Game qui est le jeu

II. Problèmes rencontrés et solutions apportées

Riyane : Gestion des pièces, scores et timer

Riyane a dû surmonter plusieurs difficultés techniques, notamment liées à la gestion des pièces entre tableau dynamique et statique. Intégrer les pièces dans des fonctions adaptables a nécessité de refaire la structure, en choisissant une représentation plus simple et fiable.

L'implémentation du tri des scores a également été complexe. Le tri fusion, choisi au départ, posait des problèmes en allocation dynamique. Riyane a temporairement utilisé le tri rapide, qui a causé un bug critique : une boucle infinie dans le menu score. Après analyse, la solution a été de revenir au tri fusion avec allocation dynamique, ce qui a permis de résoudre le problème.

Concernant le timer, l'obtention d'une mesure précise en millisecondes n'était pas possible avec “time()”. Riyane a trouvé une solution dans les spécifications du C11

Enfin, certaines pièces dépassaient la grille à cause de leur format 5x5. Il a créé deux fonctions : l'une pour “réduire la taille” des matrices des pièces, l'autre pour “calculer dynamiquement” leur hauteur et largeur, évitant tout débordement.

Salim : Suppression de lignes et gestion de grille

La principale responsabilité de Salim était de gérer les lignes pleines. La suppression et le décalage des lignes supérieures présentait des pièges logiques, en particulier pour maintenir la cohérence de la grille. Un souci en particulier a été de portée au calcul correct des index après chaque suppression.

Il a également dû résoudre un bug important : certaines pièces restaient "suspendues" dans la grille, comme figées. Ce problème d'affichage venait d'un manque de synchronisation entre les données de la grille et ensuite l’affichage après des tentatives pour essayer de régler le soucis nous avons laissé de côté.

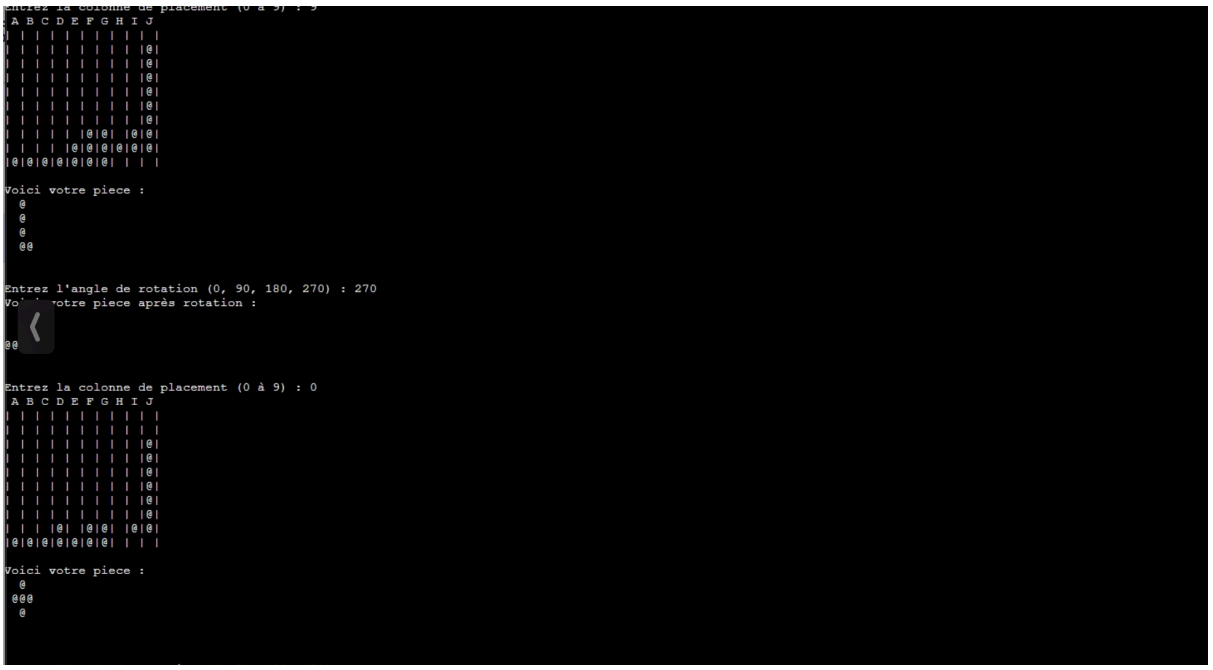
Mehdi :Placement des pièces et robustesse

J'ai dû faire face à plusieurs problèmes sur la fonction poser_piece. Premier soucis, les caractères @ se positionnaient mal, apparaissant sur les bords de la grille. Le souci venait d'une mauvaise gestion des coordonnées de départ, corrigée par un ajustement précis du calule des colonnes.

Un deuxième bug concernait la détection de dépassement horizontal: certaines pièces enlever directement un bout de pièce sans alerte et ça seulement à droite de la grille(normal comme on calculer le placement en partant de la gauche (sur le coup je trouvais pas ça normal et ne comprenait pas pourquoi)). J’ai eu beaucoup de mal a régler ce problème car pour moi ma fonction était bonne et même en décalant soit je crée un bug avec la colonne 9 un jouable ou sinon je retombais sur la même choses car je ne voyais pas d’autre possiblité pour moi. Ce problème, difficile à identifier, a été résolu grâce à l'aide de Riyane, qui a eu une vision extérieur et qui m’a expliqué d’où venait mon soucis et avec son aide on a résoulu le problème en quelques testes seulement.
Enfin, j’ai également du travailler sur une meilleure robustesse du programme. Lorsqu'un utilisateur saisissait un caractère au lieu d'un entier, le programme entrait en boucle infini. La création de la fonction viderBuffer() a permis de gérer ces cas de façon propre, évitant tout comportement imprévisible.

Problème:

-Si nous avons un sol pas complet suivi d’une ligne que nous essayons de compléter et qu’on la supprime alors nous avons une pièce qui “flotte” qui ne descend pas jusqu'au bout (voir image).



On peut voir que lorsqu'on supprime la ligne 2 car pleine la ligne 3 descend bien mais du coup on se retrouves avec le L “flottant qui n’est pas descendu.