Documentation et Analyse du Programme Scrabble en C avec SDL2

Table des Matières

---

1. Introduction

Ce programme implémente un jeu de Scrabble simplifié en langage C. Il utilise SDL2 pour l'affichage graphique et SDL2_ttf pour le rendu du texte. Le programme intègre la logique de placement des mots, le calcul des scores, la gestion d'un dictionnaire pour vérifier la validité des mots, ainsi qu'une interface utilisateur interactive qui permet de saisir des mots, de sélectionner des cases sur le plateau, et d'échanger les lettres du chevalet (rack).

2. Structure Générale du Code

Le code se divise en plusieurs sections :

- **Définitions et inclusions :**
  Définition de la version POSIX, inclusion des bibliothèques SDL2, SDL2_ttf et autres bibliothèques standard. Déclaration des constantes pour les dimensions de la fenêtre, des zones de jeu, et des couleurs.
- **Fonctions de logique de jeu :**
  Fonctions qui gèrent le calcul des scores, le tirage aléatoire de lettres, la vérification de la possibilité de placement d’un mot et le placement effectif du mot sur le plateau.
- **Gestion du dictionnaire :**
  Chargement et tri du dictionnaire depuis un fichier, ainsi que des fonctions de recherche (dichotomique) pour valider que les mots saisis existent dans le dictionnaire.
- **Fonctions de rendu graphique :**
  Fonctions qui dessinent les différentes zones de l’interface (plateau, grille, rack, zone de saisie) et affichent le texte (lettres, scores).
- **Boucle principale :**
  La boucle d’événements gère la saisie utilisateur (clique, texte, touches clavier) et met à jour l’état du jeu en appelant les fonctions de rendu.
- **Nettoyage :**
  Libération de toutes les ressources allouées et fermeture propre des sous-systèmes SDL2 et SDL2_ttf.

3. Fonctions de Logique de Jeu

getLetterScore

But : Retourne le score attribué à une lettre en se basant sur les règles du Scrabble.
Fonctionnement :

- Convertit la lettre en majuscule avec `toupper`.
- Utilise `strchr` pour vérifier à quelle catégorie la lettre appartient et retourne le score correspondant.
- Si la lettre n'est pas reconnue, retourne 0.

drawRandomLetter

But : Tire une lettre aléatoire selon la distribution classique du Scrabble.
Fonctionnement :

- Déclare un tableau de structures contenant chaque lettre et son nombre d'occurrences.
- Calcule le total d'occurrences.
- Tire un nombre aléatoire et parcourt la distribution pour retourner la lettre correspondante.

canPlaceWord

But : Vérifier si un mot peut être placé sur le plateau à partir d’une position donnée, dans une direction spécifiée.
Fonctionnement :

- Vérifie que toutes les positions nécessaires sont dans les limites du plateau.
- Pour chaque lettre, si la case est vide, s'assure que le rack contient la lettre ; si la case est occupée, vérifie la correspondance.
- Vérifie qu’en cas de premier coup (score total == 0) le mot passe par la case centrale, et pour les coups suivants, qu'il croise au moins une lettre existante.

placeWord

But : Place le mot sur le plateau et met à jour le rack.
Fonctionnement :

- Parcourt le mot et, pour chaque lettre, calcule la position sur le plateau.
- Si la case est vide, la lettre y est écrite et le rack est mis à jour en remplaçant la lettre utilisée par une nouvelle lettre tirée aléatoirement.

4. Gestion du Dictionnaire

caseInsensitiveCompare

But : Comparer deux chaînes sans tenir compte de la casse.
Fonctionnement : Utilise `strcasecmp` pour effectuer une comparaison insensible à la casse, ce qui permet de trier le dictionnaire.

loadDictionary

But : Charger un dictionnaire de mots depuis un fichier texte et trier les mots.
Fonctionnement :

- Ouvre le fichier et lit chaque ligne (chaque mot).
- Supprime le saut de ligne et duplique la chaîne pour la stocker dans un tableau dynamique.
- Si le tableau devient trop petit, sa capacité est doublée via `realloc`.
- Trie ensuite le tableau avec `qsort` en utilisant `caseInsensitiveCompare`.

binarySearch & isValidWordBinary

But : Rechercher un mot dans le dictionnaire trié.
Fonctionnement :

- `binarySearch` effectue une recherche dichotomique dans le tableau trié.
- `isValidWordBinary` utilise `binarySearch` pour vérifier la validité d'un mot.

validatePlacement

But : Valider le placement d'un mot en simulant son placement sur une copie du plateau et en vérifiant les mots croisés.
Fonctionnement :

- Crée une copie temporaire du plateau.
- Simule le placement du mot et vérifie que toutes les cases sont valides.
- Pour chaque lettre placée, reconstruit le mot perpendiculaire et vérifie sa validité avec le dictionnaire.
- Libère la copie temporaire avant de retourner le résultat.

5. Rendu Graphique avec SDL2

Initialisation de SDL2 et SDL2_ttf

- **SDL_Init(SDL_INIT_VIDEO)** initialise le sous-système vidéo.
- **TTF_Init()** initialise le sous-système pour le rendu des polices.
- **SDL_CreateWindow()** crée la fenêtre principale.
- **SDL_CreateRenderer()** crée le renderer qui sert à dessiner.
- **SDL_SetRenderDrawBlendMode()** configure le mode de blend (transparence).

Fonctions de Rendu

drawGrid

But : Effacer l'écran et dessiner la grille du plateau.
Fonctionnement :

- Efface l'écran avec la couleur de fond.
- Calcule la taille d'une case en fonction des dimensions du plateau.
- Dessine des lignes verticales et horizontales pour former la grille.

drawBoard

But : Dessiner le plateau de jeu complet.
Fonctionnement :

- Parcourt chaque case du plateau et définit un rectangle correspondant.
- Colore la case selon le bonus (ou si c'est la case centrale).
- Si une lettre est placée, dessine un overlay beige et rend la lettre ainsi que sa valeur.

drawRack

But : Dessiner la zone du chevalet (rack) et le bouton « Echanger ».
Fonctionnement :

- Dessine un rectangle de fond pour la zone du rack.
- Pour chaque jeton du rack, calcule sa position et dessine la case, la lettre et la valeur.

- Dessine ensuite le bouton "Echanger" à côté du rack.

drawInputArea

But : Dessiner la zone de saisie en bas de la fenêtre et afficher le message d'invite correspondant à l'état de saisie.
Fonctionnement :

- Détermine la zone de saisie et la remplit d'une couleur de fond.
- En fonction de l'état (`STATE_IDLE`, `STATE_INPUT_TEXT` ou `STATE_INPUT_DIRECTION`), prépare un message d'invite.
- Rendu du texte via TTF_RenderText_Blended puis affichage sur la zone.

6. Boucle Principale et Gestion des Événements

La fonction `main()` orchestre le programme. Voici son fonctionnement détaillé :

- **Initialisation**
    - Génère une graine aléatoire avec `srand(time(NULL))`.
    - Charge le dictionnaire via `loadDictionary()`.
    - Alloue dynamiquement le plateau de jeu et l'initialise (15x15, avec chaque case contenant un espace pour signifier une case vide).
    - Initialise un tableau bonus pour appliquer les multiplicateurs de score selon les règles du Scrabble.
- **Initialisation SDL**
    - Appelle `SDL_Init()` et `TTF_Init()` pour préparer l'affichage graphique et le rendu du texte.
    - Crée la fenêtre et le renderer.
    - Charge les polices nécessaires depuis le système.
- **Gestion du Rack**
    - Le rack est initialisé avec 7 lettres tirées aléatoirement via `drawRandomLetter()`.
- **Gestion de la Saisie Utilisateur**
    - Le programme utilise un système d'états (`InputState`) pour gérer les différentes phases de saisie :
        - `STATE_IDLE` : Le programme attend un clic pour démarrer la saisie.
        - `STATE_INPUT_TEXT` : L'utilisateur saisit un mot (les caractères sont récupérés avec SDL_TEXTINPUT et convertis en majuscules).
        - `STATE_INPUT_DIRECTION` : Si le mot contient plus d'une lettre, l'utilisateur doit spécifier la direction (h ou v).
    - Les événements gérés comprennent :
        - **SDL_MOUSEBUTTONDOWN** : Pour sélectionner une case du plateau ou cliquer sur le bouton "Echanger".
        - **SDL_TEXTINPUT** et **SDL_KEYDOWN** : Pour la saisie de texte, gestion du retour arrière (Backspace) et validation avec Entrée (Return).

        - **SDL_QUIT** : Pour fermer la fenêtre et quitter le programme.
- **Rendu Graphique**
    - Après la gestion des événements, le programme appelle successivement les fonctions de rendu pour afficher :
        - Le plateau (grille, bonus, lettres, overlays).
        - Le rack et le bouton "Echanger".
        - La zone de saisie avec le message d’invite.
        - Le score du dernier mot et le total des points.
    - `SDL_RenderPresent()` met à jour l'affichage à l'écran.
    - `SDL_Delay(16)` limite le rafraîchissement à environ 60 images par seconde.

7. Nettoyage et Libération des Ressources

Avant de quitter, le programme procède à un nettoyage complet :

- Libération de la mémoire allouée pour le plateau et le dictionnaire.
- Fermeture des polices chargées via `TTF_CloseFont()`.
- Destruction du renderer et de la fenêtre.
- Appels à `TTF_Quit()` et `SDL_Quit()` pour fermer proprement les sous-systèmes.

8. Conclusion et Perspectives d'Amélioration

Ce programme combine la logique d'un jeu de Scrabble et une interface graphique interactive grâce à SDL2. Pour quelqu'un qui souhaite comprendre et améliorer le code, voici quelques pistes d'amélioration :

- **Optimisation du rendu graphique :**
  Regrouper certains appels de rendu pour éviter la redondance (par exemple, le redessin final des lettres qui est actuellement dupliqué dans `drawBoard` et à la fin de la boucle principale).
- **Gestion de la saisie :**
  Actuellement, la saisie se fait via SDL_TEXTINPUT et SDL_KEYDOWN. Il serait intéressant d'ajouter une interface plus riche pour la saisie, par exemple, en affichant visuellement le texte saisi dans la zone de saisie et en gérant d'autres actions clavier.
- **Validation du placement :**
  La fonction `validatePlacement()` peut être améliorée pour mieux gérer les cas particuliers des mots croisés et intégrer des règles supplémentaires du Scrabble.
- **Modularisation :**
  Le code peut être séparé en plusieurs fichiers (par exemple, un fichier pour la gestion du plateau, un autre pour la logique du jeu et un autre pour le rendu graphique) pour faciliter la maintenance et l'évolution.
- **Interface utilisateur :**
  Ajouter des menus, des notifications et d'autres éléments visuels pour rendre l'expérience utilisateur plus agréable.