Manar Nssair & Sarah Nguyen

TP 7: Jeu du Boggle

Mode d'emploi utilisateur:

1) Installation :

Dans le terminal, utiliser la commande `make` pour compiler le projet.

Pour lancer le jeu avec le fichier *Mots* (dictionnaire de support pour le jeu) , écrivez `./main Mots` .

2) Utilisation du jeu :

- Règles :

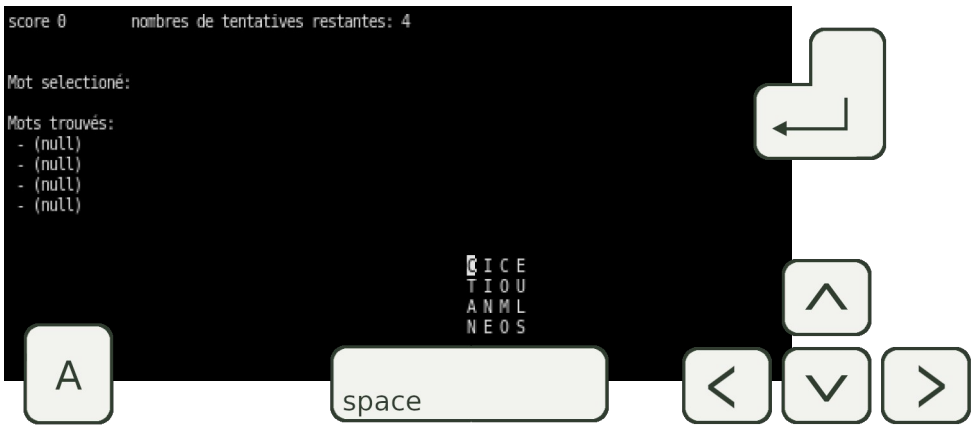
  Vous possédez au départ quatre tentatives et votre score est à 0.

  Sur une grille de lettres, tentez de construire au plus quatre mots qui rapporteront le plus de points possible.

  Pour construire un mot, il faut que les 2 lettres soient voisines : c'est-à-dire si l'une est située immédiatement à gauche, à droite, en haut, en bas ou en diagonale par rapport à l'autre.

  Information supplémentaire : Un mot rapporte des points seulement s'il est composé d'au moins trois lettres.

- Début de partie :



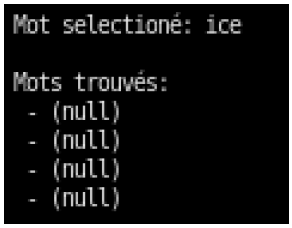
Validez le mot à l'aide de la touche entrée.

Déplacez vous de case en case avec les flèches directionnelles .

Annulez la dernière lettre du mot sélectionné avec la touche 'a'.

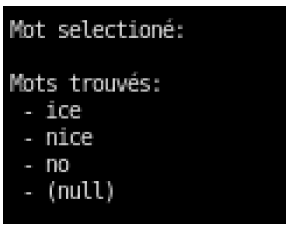
Validez la lettre à l'aide de la touche espace.

- Le mot que vous avez sélectionné est alors affiché :

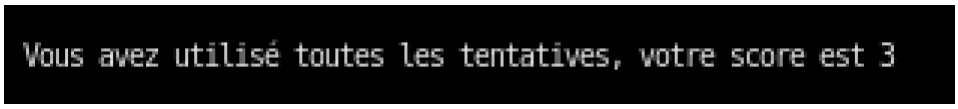


- En validant le mot sélectionné, vous utilisez une tentative.

  Si ce mot existe, il sera alors placé parmi les mots que vous avez trouvés. Dans le cas contraire, ce mot ne sera pas pris en compte.



- Après avoir utiliser vos 4 tentatives, le jeu se termine.



Modules:

Nous avons fait le choix de faire 4 modules :

- Dico : contenant les fonctions qui se chargent du dictionnaire : elles stockent dans un arbre ternaire de recherche tous les mots du fichier Mots.
- Jeu : contenant les fonctions de la grille : initialisation, affichage, etc.
- Joueur : contenant les fonctions qui manipulent la structure joueur, pour remplir la liste des mots trouvées et actualiser son score.
- Graphique : contenant toutes les fonctions de l'affichage graphique.

Structures :

- ATR : arbre ternaire de recherche ayant 3 pointeurs fg, fm, fd (module dico).
- Lettre : possédant une lettre et une probabilité d'apparaître dans la grille (module jeu).
- Jeu : possédant une grille : un tableau statique à deux dimensions (de taille 4*4) (module jeu).
- Joueur : avec un nombre de tentatives qui part de 4, un mot qu'il sélectionne, la liste des mots qu'il a trouvée et son score (module joueur).

Fonctions :

Module dico :

- `ATR creer_ATR_vide();`
  → Initialise un arbre ternaire de recherche vide.

- `void liberer_ATR(ATR * A);`
  → Libère l'arbre ternaire de recherche entré en paramètre.

- `int ajoute_branche(ATR * A, char * mot);`
  → Ajoute une nouvelle branche à l'arbre contenant le nouveau mot.

- `int inserer_dans_ATR(ATR *A, char * mot);`
  → Insère dans l'arbre un nouveau mot.

- `void affiche_aux(ATR A, char buffer[], int i);`
  → Affiche les différents mots présents dans l'arbre .

- `void afficher_ATR(ATR A);`
  → Affiche l'arbre.

- `FILE *ouvre_fichier(const char *chemin);`
  → Ouvre le ficher avec le mode d'accès choisi.

- `int remplir_ATR(FILE * df, ATR * A);`
  → Rempli l'arbre avec les mots lus dans le fichier .

- `int appartient_dico(ATR t, char * mot);`
  → Teste si un mot appartient à un arbre ternaire de recherche : renvoie 1 si c'est le cas, et renvoie 0 sinon.

Module jeu :

- `void appel_init_grille(Jeu * j);`
  → Elle déclare un tableau de Lettre de longueur 26, pour y mettre toutes les lettres de l'alphabet avec leur probabilité d'apparaître dans la grille selon l'ordre dans lequel elles sont misent dans le tableau.
  Par exemple `{'E', 0}, {'T', 11}, {'A', 19}` sont les 3 premiers éléments du tableau. On commence par la lettre 'E', celle-ci prendra donc la valeur 0. La seconde lettre ('T') prendra alors comme valeur la somme entre la probabilité de la lettre précédente (ici 'E', soit 11 %) et sa valeur (0), et ainsi de suite (la valeur de 'A' sera alors la valeur de la lettre 'T' (soit 11) qu'on additionne à la probabilité de 'T' (8), on obtient alors 19).

- `char generer_lettre_aleatoire(Lettre Tab[26]);`
  → Génère une lettre aléatoire à partir du tableau.

- `void init_grille(Lettre Tab[26], Jeu * j);`
  → Initialise la grille contenant les lettres aléatoires .

- `void afficher_grille(Jeu * jeu);`
  → Affiche la grille.

Module joueur:

- `Joueur * allouer_joueur();`
  → Alloue un joueur et l'initialise.

- `void calcul_score(Joueur * j);`
  → Calcul le score du joueur .

- `void ajouter_c_mot_selec(Joueur * joueur, char c);`
  → Ajoute un caractère au mot sélectionné.

- `void supprimer_fin_chaine(Joueur * joueur);`
  → Supprime le dernier caractère du mot sélectionné.

- `void ajouter_mots_trouves(Joueur * j, char * mot);`
  → Ajoute un mot à la liste des mots trouvés du joueur.

- `int valider_mot_selec(Joueur * j, ATR t);`
  → Ajoute le mot sélectionné au mots trouvés si le mot appartient à l'arbre ATR et si le mot n'est pas déjà dans les mots trouvés.(le joueur ne l'a pas déjà trouvé). Elle revoie 1 si le mot a été ajouté et 0 sinon.

Module graphic:

- `void afficher_grille_graph(Jeu jeu, int i_lettre, int j_lettre);`
  → Affiche la grille du jeu en prenant en compte la surbrillance de la lettre aux coordonnées i et j.

- `int cases_voisines(int col, int li, int x, int y);`
  → Renvoie 1 si x et y sont les coordonnées d'une des 6 cases autour de la case de coordonné (col, li), et 0 sinon.

- `char * selectioner_mot(Jeu jeu, Joueur * joueur, ATR A);`
  → Cette fonction permet de jouer au jeu : Tant que le nombre de tentatives du joueur n'est pas fini, on sélectionne un mot avec les touches correspondantes et on l'ajoute au mots trouvés s'il est bien dans le dictionnaire (l'arbre).

Conclusion :

Dans l'ensemble, nous étions parfois bloqué mais cela se résolvait assez rapidement. Nous avons cependant rencontrés des difficultés lors de la mise en place du jeu et de l'utilisation de la bibliothèque graphique en même temps.