# Perfectionnement à la programmation en C
## Fiche de TP 7

L2 Informatique 2020-2021

*Jeu du* Boggle

Ce TP se déroule en **une seule séance** et est à faire par binômes. Le travail réalisé doit être envoyé au plus tard exactement **une semaine après** le début de la séance de TP. Il sera disposé sur la plate-forme prévue à cet effet et constitué des programmes répondant aux questions et des éventuels fichiers annexes qui peuvent être demandés.

L'objectif de ce TP est d'implanter le jeu du *Boggle*[1], ou plus exactement une variante dont nous préciserons les règles. Nous utilisons et approfondissons ici les notions de

— lecture d'une spécification ;

— découpage d'un projet en modules ;

— lecture efficace dans un fichier texte ;

— interface NCURSES.

Ce TP est beaucoup moins directif que les précédents. Il ressemble plus à une véritable spécification fournie par un client. Il existe ainsi plusieurs façons de mener à bien le travail demandé, mais il faut faire attention à répondre exactement à la spécification et à utiliser **tout** ce qui a été vu pour le moment dans les cours et TD/TP précédents (comme entre autres, la mise en place de pré-assertions, les fonctions à gestion d'erreurs, l'écriture de documentation, la bonne façon de pratiquer la modularisation et la mise en place de tests).

Une attention particulière sera également apportée à la partie *documentation pour l'utilisateur* du rapport, partie qui rappelons-le, explique comment installer l'application et comment l'utiliser. Il doit s'agir ici de rédiger un mode d'emploi complet recensant toutes les fonctionnalités de l'application, illustrées possiblement d'exemples et de figures.

## 1 Spécification

La spécification du projet est divisée en deux parties : nous commençons par donner un aperçu global du comportement de l'application, puis nous nous focaliserons sur des points

1. Voir https://fr.wikipedia.org/wiki/Boggle.

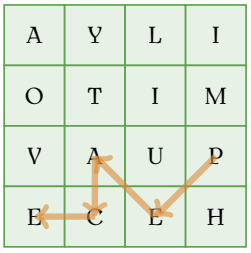
plus précis. Ces explications se terminent sur des ouvertures possibles donnant quelques idées de variantes et d'améliorations.

## 1.1 Explications générales

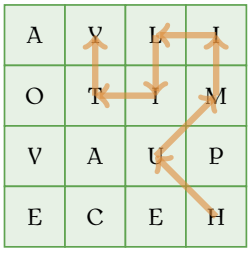
Il s'agit de programmer un jeu à un joueur qui présente une grille de dimensions $4 \times 4$ de lettres générée aléatoirement. Par exemple,

| | | | |
|---|---|---|---|
| A | Y | L | I |
| O | T | I | M |
| V | A | U | P |
| E | C | E | H |

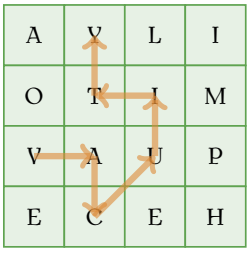
est une grille possible. L'utilisateur (nommé « joueur » dans la suite) doit désigner une suite de cases distinctes dans la grille, deux à deux voisines. Deux cases sont voisines si l'une est située immédiatement à gauche, à droite, en haut, en bas ou en diagonale par rapport à l'autre. Voici quelques mots que le joueur est autorisé à construire :



Le mot `PEACE`.



Le mot `HUMILITY`.



Le mot `VACUITY`.

Dans la forme initiale de l'application, le joueur doit tenter de construire de cette façon au plus quatre mots qui lui apporteront des points relativement à leur longueur.

## 1.2 Détails obligatoires

### 1.2.1 Langue

Comme nous avons pu le remarquer, la langue du jeu est l'anglais. Cela nous est utile pour éviter de manipuler des accents et de rester dans l'alphabet ASCII. De plus, par convention, la casse n'est pas significative.

### 1.2.2 Génération de la grille

À chaque lancement d'une nouvelle partie, la grille est générée aléatoirement, mais selon une distribution précise. En l'occurrence, chaque lettre à une probabilité d'apparaître selon la table

| Lettres | Probabilité |
|---|---|
| E | 0.11 |
| T | 0.08 |
| A, I, N, O, S | 0.07 |
| R | 0.06 |
| H | 0.05 |
| D, L | 0.04 |
| C, M, U | 0.03 |
| B, F, G, P, W, Y | 0.02 |
| J, K, Q, V, X, Z | 0.01 |

Le contenu de chaque case de la grille est ainsi généré en respectant cette distribution et de manière indépendante relativement aux autres contenus des cases.

### 1.2.3 Calcul du score

Le calcul du score $s$ marqué par le joueur à la fin de la partie ayant trouvé un ensemble $M$ de mots se réalise selon la formule

$$s = \sum_{\substack{u \in M \\ |u| \geqslant 3}} 2^{|u|-3},$$

où $|u|$ désigne la longueur du mot $u$. Par exemple, si le joueur devine un mot de 3 lettres, deux mots de 5 lettres et un mot de 8 lettres, il totalise

$$2^{3-3} + 2^{5-3} + 2^{5-3} + 2^{8-3} = 1 + 4 + 4 + 32 = 41$$

points. Remarquons que, d'après la formule, les mots de longueurs 1 ou 2 ne valent aucun point.

### 1.2.4 Saisie de mots

Parlons à présent de l'ergonomie de l'application. Concernant son aspect graphique, outre l'utilisation obligatoire de NCURSES, rien n'est imposé. En ce qui concerne l'interaction entre le joueur et la machine pendant la partie, voici les recommandations à suivre. À chaque instant, une case de la grille est sélectionnée (initialement la case en haut à gauche) et apparaît en surbrillance. Le joueur peut sélectionner une case voisine au moyen des flèches directionnelles. Pour saisir un mot, le joueur sélectionne et valide une à une et dans l'ordre les cases formant les lettres du mot visualisé. Une case validée doit se distinguer graphiquement des cases qui ne le sont pas. La validation de chaque case se fait au moyen de la touche espace et la validation finale du mot au moyen de la touche entrée. La touche `'a'` sert à annuler les cases éventuellement précédemment validées.

### 1.2.5 Validité de la saisie

Une fois que le joueur a proposé un mot de cette façon, l'application va vérifier si le mot existe bien en prenant comme support un dictionnaire. Celui-ci est fourni sous la forme d'une archive `Mots.zip` qui contient un fichier texte `Mots` renfermant une collection de 370103 mots, un par ligne, et triés lexicographiquement. Il s'agit d'utiliser ce fichier de manière efficace : lors du lancement du programme, le fichier est lu pour être transféré dans une structure adéquate et efficace pour les besoins des recherches. Rien n'est imposé pour cette structure mais une recherche en complexité temporelle logarithmique par rapport à la taille du dictionnaire est souhaitable (mais pas absolument obligatoire). Une fois le jeu initialisé et lancé, aucune lecture du fichier n'est autorisée.

### 1.2.6 Suite et fin du jeu

Lorsque le joueur propose un mot existant dans le dictionnaire, son score partiel est affiché, totalisant le score du nouveau mot et du score marqué précédemment. À l'inverse, si le joueur propose un mot qui n'existe pas, il a utilisé une tentative qui fait qu'il ne pourra proposer plus que trois mots et ainsi de suite. Une fois quatre tentatives faites, le jeu s'arrête avec mention du score.

## 1.3 Variantes et améliorations optionnelles

Un respect de toutes les consignes précédentes (*y* compris celles figurant dans l'en-tête de cette fiche) vallent la note maximale si tout est parfaitement mis en œuvre. Cependant, pour ceux qui veulent aller plus loin, il est possible d'inclure un ou plusieurs points parmi les suivants pour améliorer l'application :

— *(Facile.)* proposer des grilles de dimensions arbitraires comme $5 \times 5$, $6 \times 6$, *etc.* et aussi rectangulaires $4 \times 5$, $3 \times 6$, *etc.* ;

— *(Moyen.)* au lieu de terminer la partie après quatre tentatives réalisées par le joueur, proposer un système qui permet au joueur d'en faire autant qu'il le souhaite (tentatives erronnées comprises) pendant un temps imparti. Utiliser pour cela les mécanismes de mesure du temps étudiés dans le TP du *Chronomètre* ;

— *(Difficile.)* proposer un algorithme qui, connaissant le dictionnaire, va jouer de manière la plus optimale possible sur une grille donnée.

# 2 Écriture du programme

Utiliser à présent toutes les techniques de réflexion, d'analyse et de gestion de projet étudiées dans les séances précédentes pour mener à bien la tâche qui vous est confiée.