Programmation web avancée - TD 3 - 3h30

Université Lumière Lyon 2 - 2023/2024

M1 Informatique

Le but de cet exercice est de programmer une variante du célèbre jeu de *Boggle*, dans lequel le joueur doit former des mots en dessinant des chemins dans une grille de lettres tirées au sort (*cf* figure). Dans cette variante, la grille compte 16 lettres réparties sur 4 lignes et 4 colonnes. La première lettre du mot est déterminée par la case pointée lorsque le bouton de la souris est enfoncé. À partir de cette case, et en tenant le bouton enfoncé, le joueur déplace le pointeur de la souris sur l'une des cases voisines (haut, bas, gauche, droite ou en diagonale) pour déterminer la seconde lettre du mot. Puis, en tenant toujours le bouton enfoncé, il continue à déplacer le pointeur de case en case pour former la suite du mot. Lorsque le mot est terminé, le joueur relâche le bouton de la souris. Notez qu'un chemin ne peut pas emprunter deux fois la même case.

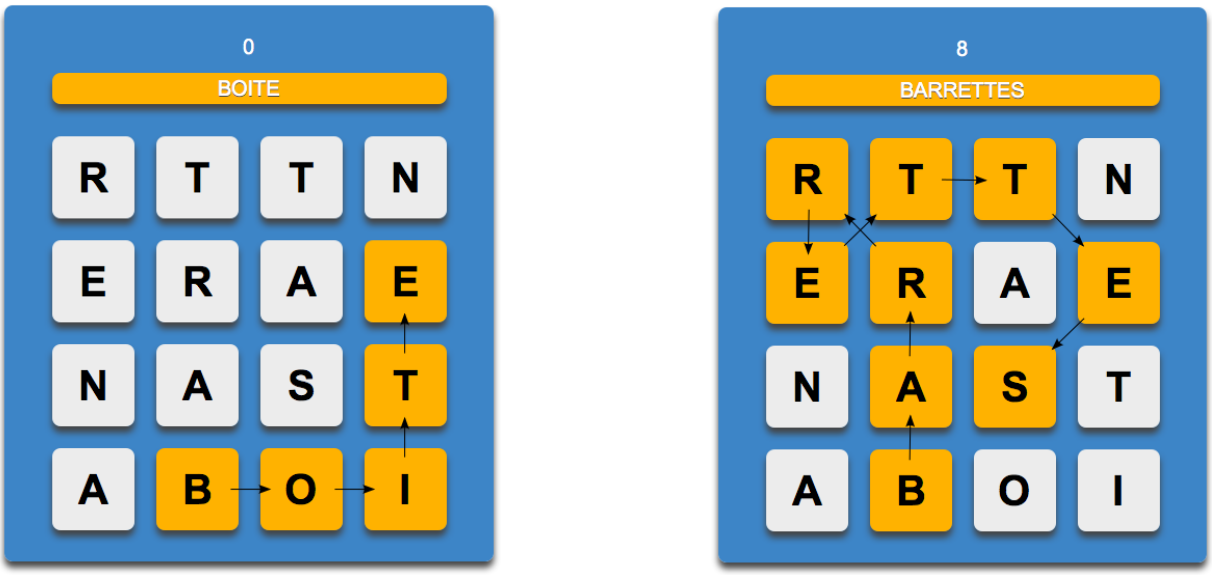

FIGURE 1 – Deux exemples de mots : pour former le premier, *BOÎTE*, le bouton de la souris est enfoncé sur le *B* et relâché sur le *E*, après avoir parcouru 5 cases dans l'ordre indiqué par les flèches ; pour le second, *BARRETTES*, le bouton de la souris est enfoncé sur le *B* et relâché sur le *S*, après avoir parcouru 9 cases dans l'ordre indiqué par les flèches. *N.B.* : les flèches ont été ajoutées à la figure pour matérialiser le déplacement de la souris ; elles ne font pas partie de l'interface graphique du jeu.

0) Préliminaire : Quelle fonction pouvez-vous proposer pour alléger dans tout le reste de l'exercice l'écriture de l'opération qui consiste à récupérer un élément *HTML* à partir de son identifiant ? (1 ligne).

1) Programme principal : Faites en sorte que la fonction `fInitJeu` soit appelée à la fin du chargement de la page. (1 ligne)

La grille de jeu sera représentée par une `<div>` d'identifiant `plateau` contenant seize `<div>` de classe `case` et d'identifiants `case0`, `case1`, ..., `case15` (*cf* code *HTML* en annexe).

Pour le tirage des lettres de la grille, on dispose d'un tableau indicé `tLettres` contenant 100 cases. Chacune des 26 lettres de l'alphabet y est présente en un nombre d'exemplaires proportionnel à la fréquence de la lettre dans la langue française (par exemple, `tLettres` contiendra quinze cases avec un "E", mais une seule case avec un "W").

En outre, on suppose disposer d'une fonction `fNbAlea` qui prend un entier `n` en paramètre et retourne un entier aléatoire compris entre 0 inclus et `n` exclu.

2) Fonction fTirage : Écrivez la fonction `fTirage` qui remplit les cases de la grille avec 16 lettres tirées au hasard parmi les 100 présentes dans `tLettres`. *N.B.* : Vous ferez en sorte qu'une même case ne puisse pas être tirée deux fois (ainsi, si le "W" n'est présent qu'en un exemplaire dans `tLettres`, il ne pourra pas apparaître plus d'une fois dans la grille). (4 lignes)

Pour programmer le jeu, il sera nécessaire de réagir à 3 événements souris : l'appui sur le bouton, le lâché du bouton et le déplacement de la souris.

3) Fonction fEcouteSouris : Écrivez la fonction `fEcouteSouris` qui associe les fonctions `fEnfonceBouton`, `fRelacheBouton` et `fDeplaceSouris` aux événements adéquats. (3 lignes)

Pour marquer les cases utilisées par le joueur pour former le mot, on utilisera un style *CSS* associé à une classe `active` modifiant la couleur de fond par défaut de la case (passant par exemple du blanc à l'orange, comme sur la figure). Vous aurez donc besoin d'ajouter ou de retirer la classe `active` aux éléments concernés. D'autre part, pour pouvoir interdire au joueur de passer deux fois par la même case, vous aurez besoin de savoir si une case fait déjà partie du chemin, c'est-à-dire possède déjà la classe `active`. Pour toutes ces manipulations de classe, vous pourrez utiliser le sous-objet `classList` disponible dans tout élément et disposant des méthode `add`, `remove` et `contains`.

Une fois qu'un mot a été formé par le joueur et traité par le programme (vérification, modification du score, *etc.*), il est nécessaire de préparer la grille pour le mot suivant en rétablissant la couleur de fond par défaut de toutes les cases.

4) Fonction fInitGrille : Écrivez la fonction `fInitGrille` qui rétablit la couleur de fond par défaut des cases de la grille. (3 lignes)

Afin que le joueur ne puisse pas marquer des points en utilisant plusieurs fois le même mot, on mémorisera les mots déjà formés dans un tableau associatif global `tMotsFaits` (qui associera par exemple la valeur `true` à chaque mot formé par le joueur).

5) Fonction fInitJeu : Vous pouvez maintenant écrire la fonction `fInitJeu` qui pose les écouteurs de souris, met le score à 0 dans la `<div>` d'identifiant `score`, réalise le tirage des lettres, et initialise le tableau `tMotsFaits` ainsi que la variable (globale) `mot` stockant le mot formé par le joueur. (5 lignes)

Pour que le joueur ait le droit d'utiliser une case de la grille dans la composition de son mot, il faut que cette case soit voisine de la case précédemment utilisée (ou bien que ce soit la première du chemin). Vous aurez donc besoin d'une fonction `fVoisines` prenant en paramètres deux numéros de case `c1` et `c2` entre 0 et 15, et retournant `true` si `c1` et `c2` sont voisines (c'est-à-dire se touchent par un côté ou un coin) et `false` sinon.

6) Fonction fVoisines : Écrivez la fonction `fVoisines`. (10 lignes)

Vous êtes maintenant en mesure d'écrire les 3 écouteurs de souris `fEnfonceBouton`, `fRelacheBouton` et `fDeplaceSouris` qui gèreront la création d'un mot par le joueur et son traitement par le programme (vérification du mot et gestion du score). Pour vérifier la validité d'un mot, on supposera disposer d'un lexique des mots acceptés sous la forme d'un tableau associatif global `tMots`. Sur le même principe que le tableau `tMotsFaits` décrit à la question 5, `tMots` associera la valeur `true` à chaque mot accepté. En ce qui concerne le décompte des points, le score sera augmenté d'une valeur proportionnelle à la longueur du mot formé (vous avez le choix de la formule exacte).

7) Fonction fEnfonceBouton : Écrivez la fonction `fEnfonceBouton`, appelée lorsque le bouton de la souris est enfoncé. (1 ligne)

8) Fonction fRelacheBouton : Écrivez la fonction `fRelacheBouton`, appelée lorsque le bouton de la souris est relâché. (5 lignes)

9) Fonction fDeplaceSouris : Écrivez la fonction `fDeplaceSouris`, appelée lorsque le pointeur de la souris est déplacé. (15 lignes)

10) Améliorations : a) Adapter le jeu aux écrans tactiles (événements `ontouchXXXX` en lieu et place (ou en plus) des événements souris utilisés) ; b) Ajouter du son (mot correct, mot incorrect, fin de partie, *etc.*) ; c) Ajouter un temps limité.

Annexe : extrait du code *HTML*

```
<body>
  <div id="jeu">
    <div id="score"></div>
    <div id="mot"></div>
    <div id="plateau">
      <div id="case0" class="case"></div>  <!-- 1ère case de la 1ère ligne (en haut à gauche) -->
      <div id="case1" class="case"></div>  <!-- 2e case de la 1ère ligne -->
      <div id="case2" class="case"></div>  <!-- 3e case de la 1ère ligne -->
      <div id="case3" class="case"></div>  <!-- 4e case de la 1ère ligne -->
      <div id="case4" class="case"></div>  <!-- 1ère case de la 2e ligne -->
      <div id="case5" class="case"></div>  <!-- ... -->
      <div id="case6" class="case"></div>
      <div id="case7" class="case"></div>
      <div id="case8" class="case"></div>
      <div id="case9" class="case"></div>
      <div id="case10" class="case"></div>
      <div id="case11" class="case"></div>
      <div id="case12" class="case"></div>
      <div id="case13" class="case"></div>
      <div id="case14" class="case"></div>
      <div id="case15" class="case"></div> <!-- 4e case de la 4e ligne (en bas à droite) -->
    </div>
  </div>
</body>
```