M5 - TD2 - LE PENDU (algorithme/JS)

vendredi 3 juillet 2020
16:00

JEU DU PENDU

1 Rappel des règles

Le jeu du pendu consiste à trouver un mot caché, dont on connaît initialement le nombre de lettres.
Lors de chaque coup joué, le joueur propose une lettre :

- Si celle-ci figure dans le mot caché, elle y apparaît à sa (ou ses) place(s) ;
- Sinon, un élément d'une potence (avec un pendu) est dessiné.

Le jeu se termine lorsque le joueur a trouvé le mot (il a alors gagné) ou lorsque la potence avec le pendu est entièrement dessinée (le joueur a alors perdu).

2 Un premier algorithme

Avant toutes choses précisons que :

- Le choix d'un mot à faire deviner est considéré ici comme une étape « élémentaire » de l'algorithme ;
- Il en va de même de la construction pas à pas de la potence, qui se résumera à l'instruction « on trace un trait de plus sur la potence » ;
- Le nombre d'erreurs autorisé pouvant varier d'un dessin de potence à un autre, sa valeur sera « logée » dans une variable nommée `erreurs_autorisees`
- Un premier algorithme respectant les règles du jeu précédemment énoncées pourrait être : JEU DU PENDU

JEU DU PENDU

- En entrée : Un mot à trouver `mot_a_trouver` et un nombre d'erreurs autorisées `erreurs_autorisees`;
- En sortie : Le résultat du jeu ;

- On précise (affectation) le nombre d'erreur autorisées
  `erreurs_autorisees`←10

- On choisit un mot `mot_a_trouver` que le joueur doit deviner
- On calcule (affectation) la longueur du mot à trouver
  `longueur`← `longueur(mot_a_trouver)`
- On initialise (affectation) le nombre d'erreur commises par le joueur
  `erreurs_commises`←0
- On initialise (affectation) le mot trouvé par le joueur
  `mot_trouve`← – – ⋯ –
  `longueur`
- tant que (`mot_trouve != mot_a_trouver`) et (`erreurs_commises < erreurs_autorisees`) faire

  On demande au joueur une lettre `lettre`

  Si
  | `lettre` est dans `mot_a_trouver`

  Alors
  | On « ajoute » dans `mot_trouve` la `lettre` là où elle apparaît

  Sinon
  | `erreurs_commises`←`erreurs_commises`+1
  | On trace un trait de plus sur la potence

  Fin si
  | On affiche `mot_trouve`
  | On affiche la potence

  Fin tant que

- Si `mot_trouve = mot_a_trouver`

  Alors

  | On écrit : « Gagné ! »

  Sinon

  | On écrit : « Perdu ! »

  Fin si

3 Quelques passages délicats

Plusieurs étapes de cet algorithme méritent d'être encore détaillées pour être considérées comme intelligibles dans le cadre d'un programme écrit en JS :

- **Choix du mot à trouver**

Une liste de mots à faire trouver dans le jeu figure dans le fichier `mots_a_trouver.js`

- **Constructions successives du mot trouvé**

La variable `mot_trouve` est initialisée sous la forme d'une chaîne de caractères contenant uniquement des traits d'union (le nombre de ces traits étant la valeur de la variable `longueur`) : cette construction mérite d'être un peu détaillée...

Dans la boucle tant que, l'une des instructions est « on « ajoute » dans `mot_trouve` la `lettre` là où elle apparaît »

.

Pour effectuer une telle instruction, on aura recours à une fonction `lettres_placees`, ayant pour paramètres un mot `mot_complet` et une chaîne de caractères `lettres_trouvees`

.

Par exemple

On veut que `lettres_placees("elevage","ela")` renvoie : `ele-a-e`

.

4 Programmation

1) Ecrire un programme `pendu.js` permettant de jouer au pendu, en suivant l'algorithme précédent.

2) Modifier le programme précédent pour que le joueur soit invité à rejouer chaque fois qu'il propose une lettre déjà tentée.

3) Modifier le programme précédent pour que, en fin de partie, on précise au joueur le nombre d'erreurs qui lui étaient autorisées.

4) Modifier le programme précédent pour que, à la fin de chaque partie, on propose au joueur de rejouer, jusqu'à ce qu'il décide d'arrêter. A l'issue de toutes les parties, le programme afficher le « score » moyen du joueur (le score pouvant être le nombre d'erreurs autorisées en fin de partie), son meilleur score et son moins bon score.