JEAN Aubin – GILQUIN Alexandre

Compte rendu :

Le jeu du pendu

Introduction :

Le pendu est un jeu consistant à trouver un mot en devinant quelles sont les lettres qui le composent. Le jeu se joue la plupart du temps à deux, mais dans le cadre de notre projet nous avons laisser la possibilité à l'utilisateur de jouer seul. Quand le dessin est terminé on voit un bonhomme pendu, si le joueur a perdu
Nous avons choisi de prendre le jeu du pendu car ceci étant notre premier projet d'informatique, nous avons voulu essayer de prendre un sujet accessible pour être sur de pouvoir rendre un programme complet, de plus nous connaissions déjà les règles du jeu. Nous nous sommes lancé dans un projet assez simple mais nous savions que si nous arrivions à le finir rapidement nous pourrions le complexifier, autant dans le programme que dans la partie graphique.

Nous avons essayé de faire une première version du programme remplissant toutes les directives de l'exercice. Nous avons commencer par séparer notre futur programme en plusieurs fonction, nous avons conclu que nous utiliserons quatre fonctions.
Ayant fini avec un peu d'avance nous avons décidé d'améliorer notre partie graphique afin que ce soit plus clair mais surtout plus esthétique.

Algorithme :

Nous avons premièrement fait l'algorithme de base, c'est à dire faire la base de notre programme pour que nous puissions ensuite passé aux parties les plus complexes.

La première phase du pendu va donc être de laisser le choix à l'aide d'une boucle « if » à l'utilisateur si il veut faire une partie à deux joueurs. Dans ce cas là, une personne va pouvoir saisir le mot à trouver pour lancer le jeu et laisser l'autre personne chercher ce mot. Dans le cas contraire, si l'utilisateur est seul, alors le mot à trouver va prendre un mot choisi aux hasard parmi une liste qui a été insérée dans l'algorithme. Le mot à trouver va prendre ce mot à l'aide de la fonction « mot secret »

Ensuite le mot à trouver va rentrer dans une boucle « while » c'est la boucle la plus importante du programme. En effet cette boucle va permettre au programme de demander à l'utilisateur une lettre tant que deux conditions seront réunie. Ces conditions représentent soit la victoire de l'utilisateur soit la défaite, c'est à dire soit le nombre d'erreurs est égale à onze et alors il a perdu soit le nombre de lettres trouvées correspond au nombre de lettres présentes dans le mot et alors il a gagné.

Si ces deux conditions sont remplies le programme va donc demander une lettre à l'utilisateur,

ensuite cette lettre va s'afficher directement sur la partie graphique pour que l'utilisateur soit au courant des lettres qu'il a déjà utilisé pour lui éviter de mettre deux fois la même lettre et donc lui faire perdre une chance.

Ensuite on rentre dans une boucle « if » cette boucle va permettre de faire deux action différentes en fonction d'une condition. Cette condition est tout simplement la présence ou non de la lettre saisie dans le mot. Pour vérifier cette condition nous avons utilisé une fonction déjà présente sur Javascript, c'est la fonction PositionDans, elle nous permet de savoir si la lettre saisie est présente dans le mot car la fonction renvoie une valeur égale à -1 si le caractère n'est pas présent dans la chaîne de caractère.
Si la lettre est présente dans le mot alors on rentre dans la fonction « position_lettre ». Cette fonction à deux rôles, premièrement elle va permettre d'afficher sur la partie graphique la lettre mais cette fonction va aussi permettre de compter le nombre de lettres identiques dans le mot à trouver. Ensuite nous intégrons une nouvelle variable appelée « nb_lettres_trouvees », cette fonction est importante car elle va permettre de stopper la boucle « while » si l'utilisateur à trouver le mot.
Si la condition n'est pas vérifier, on rentre alors dans une nouvelle fonction, la fonction « graphique ». Cette fonction permet de faire la partie graphique en fonction du nombre d'erreurs. En plus de ça, si la condition n'est pas vérifiée, on affiche sur la partie graphique, le nombre de chances qu'il reste à l'utilisateur avant de perdre la partie.

La partie principale est donc terminée. Nous avons rajouté une dernière boucle « if » à la fin. Cette boucle a comme condition la victoire ou la défaite de l'utilisateur. Si il a gagné, alors le nombre d'erreurs est différent de onze et alors on fait afficher le message « Gagné ! » sur la partie graphique. Si le nombre d'erreurs est égal à onze, alors l'utilisateur a perdu et un message s'affiche sur la partie graphique disant : « Perdu... » En plus de ça, si l'utilisateur a perdu alors on affiche en dessous du premier message quel était le mot que l'utilisateur n'a pas réussi à trouver.

Fonctions :

Pour simplifier notre algorithme final, nous avons décidé de le séparer en différentes fonctions. Cela va nous permettre d'avoir un algorithme final simple composé de quatre fonctions.

Fonction « mot_secret »:

```
function mot_secret() {

  var tableau_de_donnees = ['lapin', 'arbre', 'cheveux', 'trousse', 'crayon', 'informatique', 'cravate', 'zebre', 'ordinateur', 'clavier', 'docteur', 'impression',
                            'systeme', 'supprimer', 'feuille', 'ingenieur', 'ciseaux', 'gomme', 'papier', 'enveloppe', 'timbre', 'antilope', 'mondialisation',
                            'programmation', 'programme', 'souris', 'elephant', 'coccinelle', 'araignee', 'temperature', 'monochromatique']


  writeFile('mot_hasard', tableau_de_donnees[Hasard(Taille(tableau_de_donnees))])
  return (readFile('mot_hasard'))
}

```

La fonction « mot_secret » est une fonction qui est utile si l'utilisateur veut jouer seul. Dans ce cas, dans les consignes il était demandé d'utiliser un fichier.Pour cela, nous avons donc premièrement rentré dans un tableau une multitude de mot qui pourront être susceptibles d'être le mot à trouver. Ensuite on créé un fichier à l'aide de writeFile(), nous l'appelons « mot_hasard » et lui donnons une valeur aux hasard des mots présents dans le tableau. Le fichier va donc avoir un mot au hasard. Ensuite, on retourne de la fonction ce qui est présent dans le fichier à l'aide de la fonction readFile.

Fonction « tirets » :

```
4  function tirets(mot_a_trouver) {
5    var mottiret = ''
6    var j
7    for (j = 0; j < Taille(mot_a_trouver); j++) {
8      mottiret = mottiret + '_'
9
10   }
11   setCanvasFont('times', '80px', 'normal')
12   Texte(40, 680, mottiret, 'black');
13 }
```

Cette seconde fonction va permettre de représenter le mot sous forme de tiret. On lui envoie donc le mot_à_trouver, puis on ajoute une variable que nous avons appelé «mottiret » C'est donc le mot sous forme de tiret. On fait donc une boucle « for », c'est boucle va continuer pour une variable j commençant à zéro jusqu'à ce que sa valeur soit égale au nombre de lettre dans le mot à trouver. A chaque fois que j prendra une nouvelle valeur, alors une nouvelle variable appelée mottiret va elle prendre son ancienne valeur plus un tiret. Un fois la boucle finie, on affiche mottiret sur la partie graphique.

Fonction « position_lettre » :

```
18
19 function position_lettre(mot_a_trouver, lettre, nb_lettres_identiques) {
20   var h;
21
22   for (h = 0; h <= Longueur(mot_a_trouver); h++) {
23
24     if (CaractereEn(mot_a_trouver, h) == lettre) {
25       setCanvasFont('times', '50px', 'normal')
26       Texte(40 + h * 60, 675, lettre,'black')
27       nb_lettres_identiques = nb_lettres_identiques + 1
28     }
29   }
30   return (nb_lettres_identiques)
31 }
```

Ceci est donc la troisième fonction. Cette fonction à deux rôles. Premièrement une partie graphique qui va permettre d'afficher les lettres trouvées au bons endroits par dessus les tirets. Pour cela, nous entrons dans une boucle « for » qui, comme précédemment fonctionne jusqu'à ce que la variable h

soit inférieur au nombre de lettre dans le mot à trouver. Grâce à la boucle pour et à cette condition, on va pouvoir vérifier si la lettre saisie est présente dans le mot et on va pouvoir vérifier pour toutes les lettres présentes dans le mot et donc pouvoir prendre en compte la possibilité qu'une lettre soit présente plusieurs fois dans le même mot. On va donc pouvoir, à chaque fois afficher la lettre saisie si elle est égale à la lettre du mot à la position h. Pour l'emplacement de la lettre, c'est la variable h qui va nous la donner. Ensuite dans cette fonction, on introduit la variable « nb_lettres_identiques » qui elle va compter le nombre de lettres identiques dans un mot et faire ressortir cette valeur car elle va nous permettre de savoir quand l'utilisateur aura trouvé le mot.

Fonction « graphique » :

La fonction est un peu longue, voici le début, sinon elle continue jusqu'à nberreur = 11

```
33 function graphique(nberreurs) {
34   if (nberreurs == 1) {
35     RectanglePlein(86, 586.5, 989, 11.5,'black')
36   }
37   if (nberreurs == 2) {
38
39     RectanglePlein(86, 92, 21.5, 494.5,'black')
40   }
41   if (nberreurs == 3) {
42
43     RectanglePlein(86, 92, 559, 11.5,'black')
44   }
45   if (nberreurs == 4) {
46
47     Ligne(107.5, 230, 344, 103.5,'black')
48   }
```

La fonction graphique est-elle composée d'une multitude de boucle « if » , en effet il y a une condition pour chaque valeur de la variable nberreurs, c'est à dire qu'à chaque nouvelle valeurs, la fonction va rajouter une étape sur la partie graphique du pendu.

Difficultés rencontrées :

La première difficulté rencontrée a été pour la troisième fonction « position_lettre ». Nous avons eu du mal à afficher les lettres au bon endroit et à envisager le fait qu'il puisse y avoir plusieurs fois la même lettre dans le mot. Nous avions premièrement pensé utiliser la fonction déjà présente appelée PositionDans qui nous renvoie la position d'un caractère dans une chaîne cependant, dans le cas où lettre est présente plusieurs fois dans le mot, cela ne fonctionne pas. Nous avions aussi essayé de rentrer toutes les positions des lettres dans un tableau, mais c'était après assez complexe de ressortir ces valeurs puis d'afficher les lettres au bonnes position. Mais finalement nous avons eu l'idée d'inclure la partie graphique dans la boucle pour, ce qui rend cette partie beaucoup plus simple et courte.

Ensuite nous avons eu du mal à utiliser un fichier pour stocker nos mots puis pour en ressortir un au hasard. Nous avions premièrement essayé de faire un fichier texte à part pour ensuite le charger dans le programme. Nous avons donc eu l'idée de tout simplement créer notre fichier dans le programme, chose beaucoup plus simple à faire et qui a fonctionné directement.

Nous avons aussi eu quelques problèmes pour afficher le nombre de chances restantes à l'utilisateur pour qu'il puisse réussir le jeu. La difficulté était dans le fait que notre fonction Texte était dans une boucle pour, et le nombre de chances restantes changeant à chaque fois, sur la partie graphiques elles s'affichaient les unes sur les autres :

Pour y remédier, nous avons eu l'idée d'intégrer dans la boucle pour,sur la partie graphique, un petit carré blanc avant le message. Ceci va permettre de cacher le nombre de chances restantes précédentes.

Tentative amélioration de la partie graphique :

Ayant fini notre programme et voyant qu'il fonctionne bien, nous avons décidé d'essayer d'améliorer la partie graphique. Nous avons donc eu l'idée de faire quelque chose de beaucoup plus sofistiquer, incluant MouseClick. Nous avons réussi la partie MouseClick, cependant nous n'avons pas réussi à l'insérer dans notre programme final. Nous vous envoyons donc les deux versions même si la deuxième ne fonctionne pas.

Conclusion :

Ce projet nous a donc permis d'apprendre beaucoup de nouvelles notions en programmation. En effet le fait de travailler en autonomie nous a permis d'apprendre par nous même. Nous n'avions, tous les deux aucune connaissance en programmation autre que que celles apportées en travaux pratique et ce projet nous a permis de connaître beaucoup de nouvelles techniques et fonctions et de mieux comprendre les vraies difficultés.
C'est un enseignement aussi très pédagogique et ludique que d'apprendre en faisant un jeu de la sorte, ce genre de projet nous a motivé dès le départ et cela nous a permis de faire notre maximum sur ce projet.