Eppur si muove !

Institut Galilée

5 juin 2016

Version 2.0f

Motus

Elève

Kevin GESNOUIN (gesnouin.kevin@gmail.com)
Alexis MALAMAS (malamas.alexis@gmail.com)
Loïc LAFONTAINE (lafontaine.loic@gmail.com)
Maxime LAVASTE (maxime.lavaste@gmail.com)

Encadrant

Nicolas ROLIN (nicolas.rolin@lipn.univ-paris13.fr)

Catherine RECANATI (cathy@lipn.univ-paris13.fr)

L3 Informatique 2015-2016
Université Paris 13

Table des matières

Introduction

Dans le cadre du projet de fin d'études de la licence informatique proposé par l'Institut Galilée, notre but va être d'implémenter le jeu Motus de France 2 sur un support informatique pour qu'il soit jouable seul en mode entraînement. Motus est un jeu ludique dont le but est de trouver un mot mystère en un minimum de tentatives dans un temps imparti entre chaque proposition. Le joueur, dans sa réflexion, doit donc aussi considérer le temps qui lui reste.

Pour pouvoir développer Motus, nous devons dans un premier temps découvrir les particularités de Motus, dans un second temps, décrire les spécificités du logiciel, ainsi que les planifier et enfin choisir l'architecture et les langages informatiques que nous allons utiliser pour développer le jeu. De plus, nous avons convenu de la répartition des fonctionnalités du logiciel.

Le site est accessible ici, et le code sur notre répertoire GitHub.

I. Présentation de Motus

Motus est une adaptation française du jeu télévisé américain Lingo. Le programme est diffusé depuis le 25 juin 1990 sur Antenne 2 puis sur France 2. Le but du jeu est de découvrir des mots d'un nombre de lettres fixé.

Normalement, deux équipes de deux joueurs s'affrontent, mais nous allons uniquement développer un mode solo.

1. Règle du jeu Motus

Le but du jeu est de trouver un mot mystère composé de 6 à 10 lettres. Les tentatives dont le joueur dispose sont définies en fonction de la longueur du mot.

Lors de chaque tentative, le joueur propose un mot appartenant à la langue française dans un délai imparti de 8 secondes. Les lettres bien placées sont alors indiquées par un carré rouge sur la grille du joueur, les lettres mal placées sont entourées d'un cercle jaune. La première lettre du mot à trouver est indiquée au joueur dès le début de la partie. La grille est de dimension taille du mot x nombre de tentatives.

La partie se termine soit lorsque le joueur a trouvé le mot, il est alors déclaré gagnant, soit lorsqu'il n'a pas trouvé le mot au bout de toutes ses tentatives.

II. Description globale de Motus

La proposition de projet de départ demande une simulation du jeu Motus de France 2 pour pouvoir jouer sur une application ou une page web afin que les joueurs puissent s'amuser ou s'entraîner.

1. Profil des utilisateurs

Motus est un jeu qui s'adresse à toute personne lettrée disposant d'un moyen de se connecter à Internet. Nous devons donc concevoir une application utilisable par tout le monde, sans pré requis de connaissances informatiques, pour que n'importe qui puisse jouer à Motus.

2. Environnement

Motus devra être accessible sur un site web. Être accessible sur le web implique d'être disponible et utilisable sur tous types d'écrans, du smartphone au bureau. Nous allons donc devoir développer un site responsive design pour pouvoir répondre à cette problématique. C'est-à-dire que notre contenu va devoir s'adapter à la taille de l'écran pour pouvoir modifier l'affichage ainsi que l'expérience utilisateur pour pouvoir garantir un bon fonctionnement de l'application.

3. Hardware requis

Pour jouer, l'utilisateur devra posséder, au minimum, un écran graphique pour qu'il puisse communiquer avec l'application, d'un clavier tactile, physique, visuel ou vocal pour que l'utilisateur interagisse avec l'application et d'une connexion internet pour pouvoir accéder au site web. De plus, une souris ou un Touchpad est vivement recommandé pour utiliser l'application sur un ordinateur, ainsi qu'une sortie audio pour les effets sonores.

4. Outils requis

Le projet se réalisera sous la forme d'un site web, il sera accessible sur les navigateurs internet récents. Nous sommes actuellement en train de réfléchir si nous réservons un nom de domaine ainsi qu'un hébergement ou placer le site web sur le serveur de l'université. Un hébergement privé nous permettrait d'avoir plus de sûreté que l'hébergement de l'université qui fait souvent des maintenances. De plus, le réseau de l'université étant accessible que par le réseau interne, il est fort probable que le projet ne soit pas accessible depuis l'extérieur si nous choisissons cette solution.

MOTUS

Pour utiliser la fonctionnalité de traitement automatique, du langage naturel (TALN) afin de savoir si un mot est bien orthographié ne sera pas créé, mais récupéré sur un Framework déjà créé par un organisme extérieur.

III. Les fonctionnalités

1. Du côté utilisateur

- Choisir la longueur des mots à rechercher.
- Choisir le temps entre chaque tentative.
- Choisir le nombre de tentatives.
- Démarrer le jeu.
- Saisir un mot grâce à un clavier.
- Consulter les meilleurs scores.
- Enregistrer son score et son pseudonyme.
- Choisir une langue entre français et anglais.

2. Interface graphique

- Afficher la grille.
- Afficher la première lettre du mot mystère.
- Afficher le temps restant pour chaque proposition.
- Afficher le nombre de tentatives restantes
- Afficher un input.
- Afficher victoire et défaite.
- Mise à jour de la grille en ajoutant le mot.
- Mise en surbrillance des lettres liées à un identificateur.

IV. Les modules

1. Environnement de développement pour les modules

Dans un premier temps, nous avons besoin d'un éditeur de texte ayant au minimum une coloration syntaxique telle que Notepad ou SublimeText pour développer notre application. Ensuite, un navigateur pour pouvoir tester l'affichage de notre site web ainsi que les scripts JavaScript. La console de développement web nous sert de débogueur.

Un interpréteur de script PHP côté serveur sera aussi nécessaire. Lors de la conception, nous utiliserons le logiciel Wamp, qui nous fournit un interpréteur PHP grâce aux modules d'Apache ainsi que MySQL pour la base de données.

2. Hiérarchie des modules

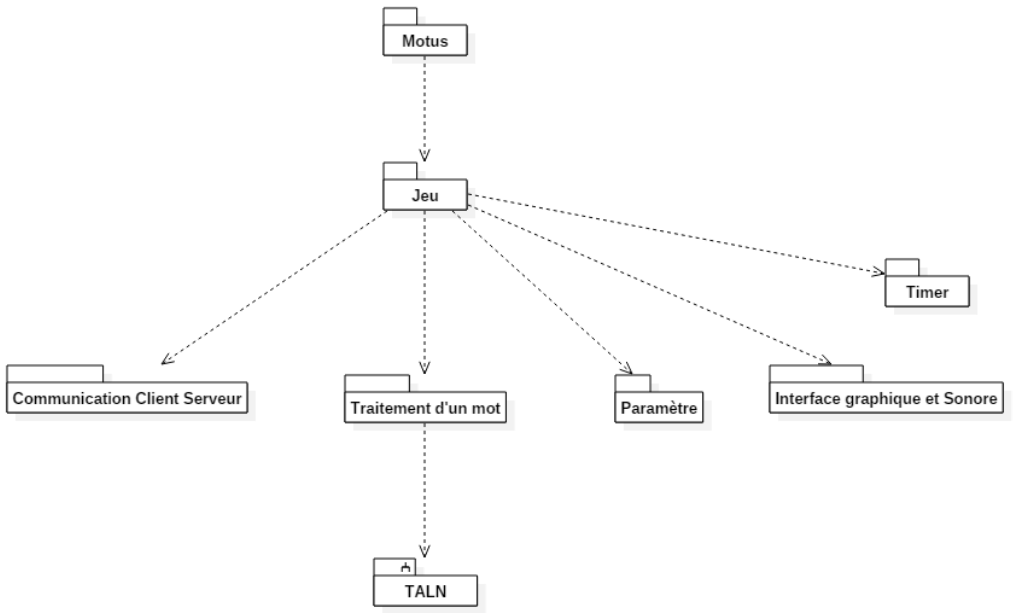

Figure 1 Hiérarchie des modules

3. Les modules

Interface Graphique et Sonore

- Indication du temps restant pour une proposition
- Afficher les propositions du joueur et les concordances avec le mot à chercher
- Afficher le nombre restant de propositions
- Construire et afficher la grille du jeu
- Afficher une victoire ou une défaite
- Présence d'un input contenant la réponse du joueur
- Effets sonores selon évènement

Traitement du mot

- Vérifier regex.
- Les lettres bien placées sont marquées par un carré rouge.
- Les lettres existantes mais mal placées sont marquées par un rond jaune.

Jeu

- Gestion partie

Communication Client/serveur

- Gestion des meilleurs de score

Timer

- Gérer le temps

Module TALN

- Vérification de l'existence du mot en français ou en anglais (selon la langue choisie)

Paramètre

- Modification de la taille du mot
- Modification nombre de tentatives
- Modification temps entre chaque tentative
- Modification de la langue

4. Descriptions détaillées des modules

Module « Paramètre »

Le joueur a le choix de pouvoir régler la longueur des mots, le temps entre chaque tentative, le nombre de tentatives par mot et la langue entre le français et l'anglais. Lors de l'implémentation, nous avons le choix de l'utilisation de superglobales sur le serveur (utilisation des cookies et des sessions) ou conserver ces valeurs dans des variables côté navigateur.

Module « Traitement d'un mot »

Il s'assurera de réagir à l'évènement « L'utilisateur entre un mot au clavier », puis mettra à jour l'interface graphique afin d'informer le joueur du résultat de sa tentative.

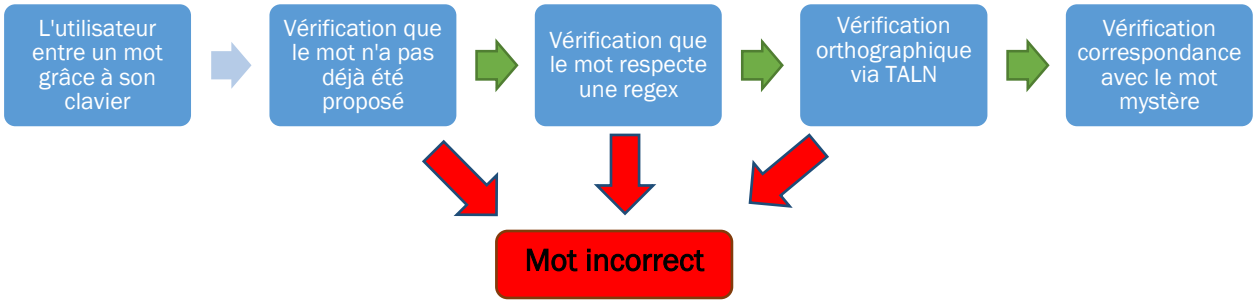

Figure 2 Processus de traitement d'un mot

Le mot entré devra respecter plusieurs contraintes pour passer à la vérification orthographique évaluée par une regex.

- Même longueur de mot que la taille demandée lors du réglage.
- Uniquement des caractères autorisés (26 lettres de l'alphabet).
- Le mot entré doit posséder la même première lettre que celle du mot mystère.

Concernant la vérification des correspondances avec le mot mystère, nous avons deux choix. Soit, lorsque l'on cherche un mot, on le conserve ensuite dans une variable chez le client, soit on le conserve côté serveur. Chaque méthode à ses défauts, la première étant qu'un joueur pourrait facilement tricher en regardant quel est le mot mystère dans la console. Pour la seconde, c'est que pour chaque demande de

vérification, nous allons devoir utiliser une requête Ajax pour vérifier les concordances nécessitant des connexions récurrentes du joueur à notre serveur.

Module « TALN »

Le TALN servira à vérifier l'orthographe des mots. Ce module sera externe et nécessitera uniquement une implémentation dans notre projet. Nous avons utilisé un framework JavaScript appelé Typo.js créé par Cris Finke fonctionnant grâce aux dictionnaires Hunspell sous la Modified BSD License.

Module « Communication Client/serveur »

Ce module sera chargé de la gestion des highscore. En effet, le module permet à un utilisateur de sauvegarder le nombre de mots consécutifs trouvés et de l'enregistrer. De plus, il peut consulter la liste des meilleurs scores.

Il utilisera des requêtes Ajax pour ne pas obliger le rechargement de pages afin de dynamiser notre site web.

Module « Interface Graphique et sonore »

Le site web hébergeant le jeu sera mis en forme grâce au langage CSS. L'interface visible par l'utilisateur sera un formulaire comportant des inputs pour modifier la taille des mots, le nombre de tentatives possibles, le temps maximum entre chaque proposition ainsi qu'un radiobox pour choisir la langue.

Après validation du formulaire, on voit une grille de jeu l'affichage d'un timer ainsi qu'un champ de formulaire afin de pouvoir rentrer un mot.

Lorsque l'utilisateur rentre un mot, ce mot est affiché dans la grille de jeu avec le code couleur des bonnes lettres, ainsi que le son correspondant. Dans la ligne suivante, on réécrit les lettres trouvées à la bonne place.

Lors d'une fin de partie, l'interface graphique propose au joueur de sauvegarder son score via un input, puis montre les meilleurs scores des joueurs.

Module « Timer »

Le Timer est la partie de l'application qui gère le temps. Il sera utilisé lors de la phase de réflexion entre chaque tentative du joueur pour savoir si le joueur dépasse la durée maximum pour proposer une proposition.

Pour l'implémenter, nous utiliserons les fonctions liées à la gestion de temps de JavaScript.

Module « Jeu »

Le module jeu permet de rassembler tous les modules et de les faire communiquer entre eux. Donc, il gère le déroulement de la partie. Pour la création de ces modules, nous nous baserons sur les différents langages du web tel que HTML, PHP, JavaScript/Jquery.

V. Conception détaillée

Le module paramètre est constitué d'un formulaire dont chaque *input* définit un paramétrage de la partie. A la validation de ces paramètres un objet Motus est créé en fonction de ceux-ci.

Avant de démarrer une partie, un mot est créé aléatoirement en parcourant un dictionnaire créé par un utilisateur du site OpenClassrooms. Le mot est alors affiché dans un tableau affiché dynamiquement en fonctions des paramètres (*taille du mot* x *nombre de tentatives*) et la première lettre apparaît. Si le mot comporte plusieurs fois cette lettre, elle est affichée à chaque occurrence.

Un input permet ensuite au joueur de rentrer son mot. Grâce à un écouteur JavaScript, le joueur ne peut rentrer ni accent ni nombre. Il peut uniquement écrire des lettres de l'alphabet.

Le mot entré est changé en majuscule, puis il est testé dans cette fonction.

```
ajouterMotTableau(mot_propose){
        var tmp = this.mot.mot_a_trouver;
        var audio;
        var delay=225;
        for(var i = 0; i < this.taille; i++){
                if(this.mot.mot_a_trouver.charAt(i) === mot_propose.charAt(i)){
                        tmp = tmp.replace(mot_propose.charAt(i));
                }
        }
        for(var i = 0; i < this.taille; i++ ){

                var present_dans_le_mot = tmp.indexOf(mot_propose.charAt(i));
                var caseARemplir=$('table tr:nth-child(' + (this.tentative + 1) + ')  td:nth-child('+ (i + 1) +')')
                var classLettre ='lettreNormal';

                $('table tr:nth-child(' + (this.tentative + 1) + ')  td:nth-child('+ (i + 1) +')').html(mot_propose
                if(this.mot.mot_a_trouver.charAt(i) === mot_propose.charAt(i))
                {
                        audio = playerAudio.bienPlacee;
                        classLettre='lettreCorrect';
                        this.ajouterIndiceLettreFind(i);
                }
                else if(present_dans_le_mot !== -1){
                        console.log(" 2. " + this.mot.mot_a_trouver.charAt(i) + " 20. "+  mot_propose.charAt(i))
                        audio = playerAudio.malPlacee;
                        classLettre='lettreMalPlacee';
                        tmp = tmp.replace(mot_propose.charAt(i));
                }
                else
                {
                        classLettre='';
                        audio = playerAudio.mauvaiseLettre;
                }

                delay=220*(i+1);
                (function(delayTime,myCase,myClass,audioToPlay){
                setTimeout( function(){pauseAudio();myCase.addClass(myClass);audioToPlay.play();}, delayTime);
        })(delay,caseARemplir,classLettre,audio);
```

Figure 3 Fonction ajouterMotTableau

Il est dans un premier temps recopié dans une variable temporaire tmp. Tmp va nous permettre de pouvoir connaitre le nombre d'occurrence d'une lettre mal placée. Par exemple, si on cherche le mot ARBRE, et que l'on rentre AVANT. On doit obtenir le résultat suivant AVANT et non AVANT.

On supprime donc dans tmp les lettres du mot entré bien placé. Ensuite, pour chaque lettre du mot entré, on test si la lettre est présente dans tmp, si elle est présente cela veut dire que cette lettre est présente au moins une fois dans le mot, et mal placé.

Si la lettre est bien placée alors on ajoute une classe « lettreCorrect ». Si la lettre est mal placée alors on ajoute une classe « lettreMalPlacee ». Sinon on n'ajoute pas de classe. Ensuite dans le code CSS, on met les couleurs correspondantes suivant les classes.

On vérifie donc dans un premier temps, si la lettre est bien placée, si elle est bien placée on mémorise la classe à insérer ainsi que le son à émettre. Sinon, on regarde si la lettre est présente dans le mot, mais pas au bon emplacement on mémorise la classe « lettreMalPlacee » et le son correspondant et on supprime cette lettre de tmp, pour conserver le nombre d'occurrences d'une lettre. Dans le dernier cas, cela veut dire que c'est une mauvaise lettre, on charge uniquement le son d'une mauvaise lettre.

Le timer est un objet en JavaScript possédant 2 champs : *horloge* et *temps*. Le champ *temps* est fixe et défini en fonction des paramètres rentrés par le joueur. L'*horloge* est une variable dont la valeur initiale est le champ *temps* et qui décrémente de 1 toutes les secondes. Pour effectuer cette décrémentation, on utilise la fonction *setInterval*. Un événement est créé lorsque le timer arrive à 0 : on effectue les mêmes actions (et donc fonctions) qui découlent d'un appui sur le bouton valider.

Lorsqu'un mot est rentré, le timer s'arrête et attend la fin de la vérification des lettres avant de sa réinitialisation. L'attente est simulée avec *setTimeout.*

Les sons sont gérés par JavaScript via un objet qui charge au préalable tous les fichiers sons dont on a besoin. Ce préchargement permet un accès rapide aux fichiers lors de l'utilisation d'effets sonores. En effet, le chargement n'a donc lieu qu'une fois.

Pour manipuler les sons, on utilise l'objet Audio fourni par JavaScript. L'utilisation des sons nécessite de retarder l'exécution du code suivant, pour attendre la fin de la

lecture d'un son. Nous les jouons selon la lettre entrée et leur concordance avec le mot à chercher. Encore une fois nous avons utilisé *setTimeout*.

Quand le joueur a perdu alors le jeu propose de rentrer son pseudonyme afin de sauvegarder son score. Lorsque le joueur opte pour sauvegarder son score alors grâce à une requête AJAX, on fait appel au fichier "ajax.php" qui permet d'insérer le pseudonyme, le score et la longueur du mot dans la table score de notre base de données. Ensuite, un deuxième appel AJAX permet de récupérer les meilleurs scores des parties effectuées présentes dans notre table score et un bouton "Rejouer" est généré afin que l'utilisateur puisse faire une nouvelle partie. Si le joueur ne souhaite pas sauvegarder son score alors aucune insertion ni génération de score ne sera effectué ensuite il sera directement redirigé vers la page de personnalisation.

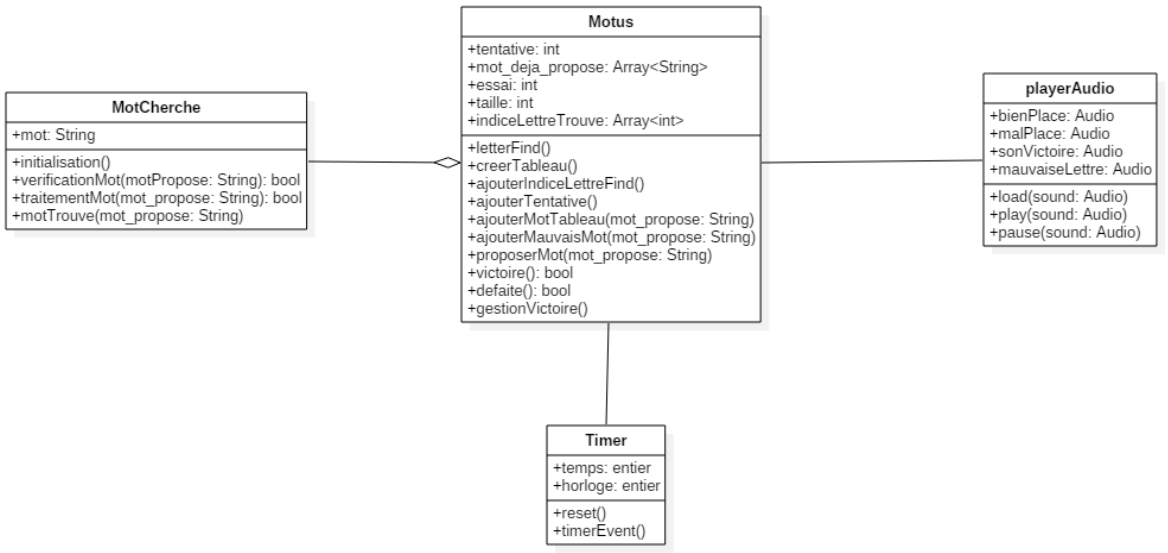

Figure 4 Diagramme de classe du jeu Motus

1. Planification du projet

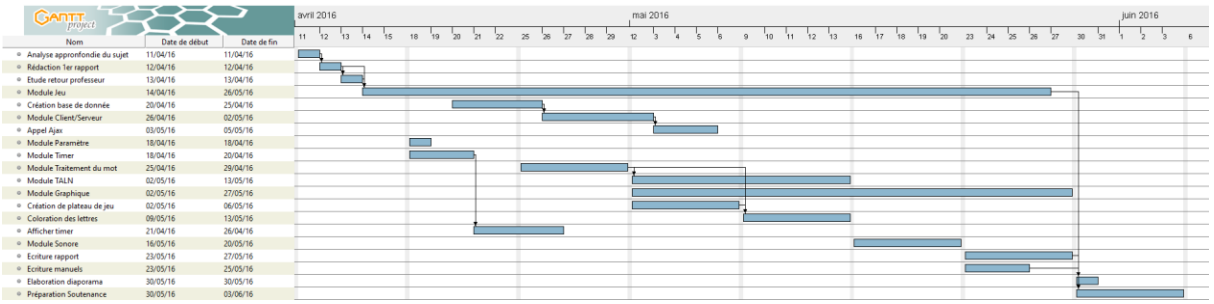

Figure 5 Diagramme de Gantt prévisionnel

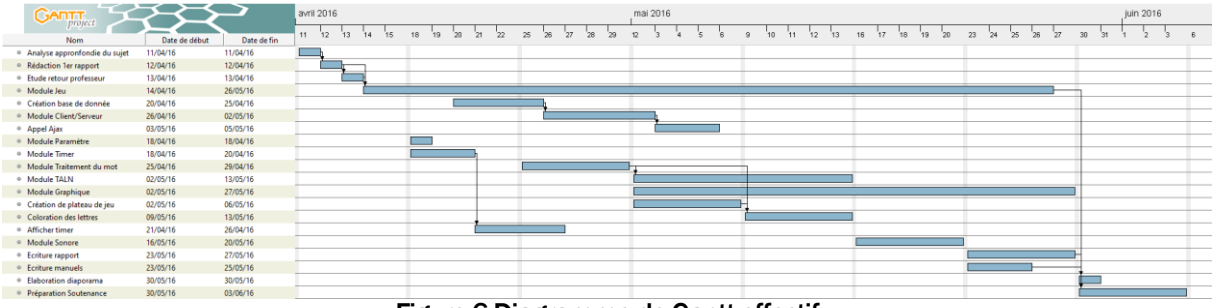

Figure 6 Diagramme de Gantt effectif

2. Améliorations possibles du projet

AMELIORATION POSSIBLE	DÉVELOPPÉ
MULTIJOUEURS	Non développé
MODE RANKING	Développé
MULTILINGUES	Développé
ACCESSIBILITES POUR LES MALVOYANTS	Non développé

Manuel d'utilisateur

1. Manuel d'installation

Avant toute chose l'utilisateur doit posséder un appareil connecté possédant un écran et permettant une connexion Internet. À cela s'ajoute qu'il faut qu'il utilise un navigateur web de préférence Chrome et posséder la dernière version pour une utilisation.

Nous utilisons les dernières spécificités de JavaScript ECMAScript 6 dont les classes, implémentées dans les dernières versions des navigateurs.

Ordinateur Mobile

Fonctionnalité	Chrome	Firefox (Gecko)	Edge	Internet Explorer	Opera	Safari
Support simple	42.0[1] 49.0	45	13	Pas de support	Pas de support	9.0

Figure 7 Implémentation des classes sur les versions desktop des navigateurs

Ordinateur **Mobile**

Fonctionnalité	Android	Android Webview	Firefox Mobile (Gecko)	IE Mobile	Opera Mobile	Safari Mobile	Chrome for Android
Support simple	Pas de support	42.0	45	?	?	9	42.0[1] 49.0

Figure 8 Implémentation des classes sur les versions mobiles des navigateurs

Lorsque l'utilisateur voudra jouer au jeu MOTUS, il faudra qu'il aille sur son navigateur web et rentre l'URL suivant : http://touiteur.esy.es/motus/.

Donc il n'a pas besoin d'installation particulière pour pouvoir jouer, c'est un client léger.

2. Manuel de jeu

Personnalisation du jeu

L'utilisateur voit sur son écran 4 paramètres qu'il peut effectuer. Il peut modifier le nombre d'essais pour trouver un mot de 6 à 10. Ensuite, la taille du mot compris entre 6 et 10. De même, il peut modifier le timer, c'est-à-dire le temps de réponse entre chaque réponse de 8 à 30 secondes. De plus, il peut choisir sa langue de jeu entre l'anglais et le français. Les mots proposés et acceptés changeront selon la langue choisie.

Lorsqu'il a fini de faire ses modifications, il lui suffit de cliquer sur « Jouer » pour lancer une partie.

La partie de jeu

L'utilisateur voit sur son écran apparaître un timer, une grille de jeu et un champ pour rentrer son mot. Dès que la page est chargée, la partie commence et l'utilisateur doit rentrer un mot avant le temps imparti. Plusieurs cas sont possibles lors de la saisie du mot.

- L'utilisateur n'a pas eu le temps de rentrer un mot alors la ligne de la grille est remplacée par des «-» et le joueur perd une tentative.

- L'utilisateur a rentré un mot trop long ou trop court ou qui n'existe pas. Alors la ligne de la grille ne donne aucune indication au joueur et il perd une tentative.

- L'utilisateur a rentré un mot qui existe, mais qui n'est pas le mot mystère. Le jeu lui indique si une lettre est bien placée par un carré rouge autour de la lettre, une lettre mal placée par un rond jaune autour de la lettre ou reste inchangé pour une lettre non présente dans le mot. De plus, des sons sont ajoutés suivant selon le résultat du traitement de la lettre.

- L'utilisateur a rentré le mot à trouver. Alors toutes les lettres du mot sont mises en rouges pour lui indiquer qu'il a trouvé le mot et un son d'applaudissement se fait retentir.

La fin de partie

- Lorsque l'utilisateur a gagné, le jeu affiche une nouvelle grille avec un nouveau mot à trouver. De plus, le champ mot trouvé est actualisé.

- Lorsque l'utilisateur a perdu, le jeu lui propose de rentrer un pseudonyme afin de sauvegarder son nombre de mots trouvé à la suite ou de passer l'étape. De plus, il peut voir les meilleurs scores effectués. Enfin, il est redirigé à la page de personnalisation.

Conclusion

Suite au projet Motus, nous avons pu approfondir nos connaissances en HTML, CSS, PHP, JavaScript et Ajax. En effet, nous avons manipulé des timer et des sons au sein de notre programme. De plus, nous avons appris à manipuler les classes en JavaScript et la lecture de fichier. Nous avons dû faire des recherches sur la manipulation de setTimeOut en JavaScript. En ce qui concerne le CSS, nous avons utilisé Bootstrap. Afin de mettre en ligne notre site en ligne, nous avons dû faire des recherches sur le FTP.

Pour la réalisation du projet, nous avons décidé d'utiliser GitHub. En effet, cela permet aux personnes du groupe de travailler individuellement sur sa partie et de le partager aux autres membres aisément sans conflits.

D'un point de vue humain, le Motus nous a montré que notre formation nous a donné de bonnes bases de compétences et connaissances afin de faire ce projet. De plus, le travail d'équipe c'est relativement bien déroulé car nous avons déjà travaillé ensemble et nous nous connaissions.

Bibliographie

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Classes

http://getbootstrap.com/css/

https://openclassrooms.com/forum/sujet/titre-a-modifier-ressource-pour-le-pendu-33934

https://github.com/cfinke/Typo.js

https://jquery.com/