CFPT – École d'informatique

Horloge qlocktwo

M306 – I.IN-P4B

SEEMULLER Julien
21/04/2015

Table des matières

Introduction

Dans le cadre du module 306, je développerais une réplique d'horloge « qlocktwo » en HTML5 & JavaScript. La particularité de cette dernière est que le temps n'est pas représenté à l'aide d'aiguilles ou de nombres, mais à l'aide de mots disposés dans un ordre précis. J'utiliserais comme modèle la version française de la « qlocktwo ». J'ai pour objectif personnel de rendre la réplique la plus fidèle possible au design original de l'horloge.

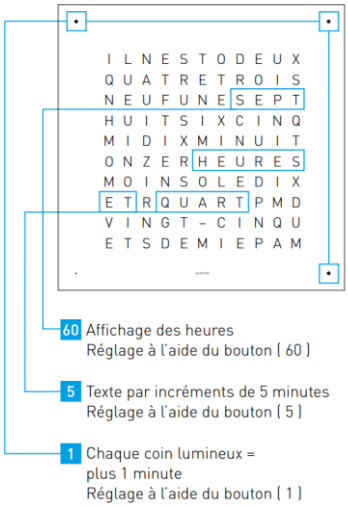
Analyse fonctionnelle

Description du fonctionnement

Le fonctionnement d'une horloge « qlocktwo » est plutôt intuitif, L'affichage de l'heure en toutes lettres change toutes les cinq minutes. Les quatre spots lumineux aux coins affichent les minutes en temps exact, par exemple : « Il est neuf heures et quart » & deux coins sont illuminés = 09 :17.

L'heure de l'horloge sera synchronisée automatiquement avec l'horloge interne de l'ordinateur de l'utilisateur pour assurer une heure exacte, l'heure ne pourra donc pas être réglée manuellement via l'application comme l'indique le schéma original ci-joint.

L'horloge murale originale utilise des LEDs pour mettre en valeurs les mots à afficher. Dans mon application, un changement de couleur de la police de caractère symbolisera cette fonction.



Fonctionnalités de l'horloge

Voici la liste des fonctionnalités qu'aura l'horloge :

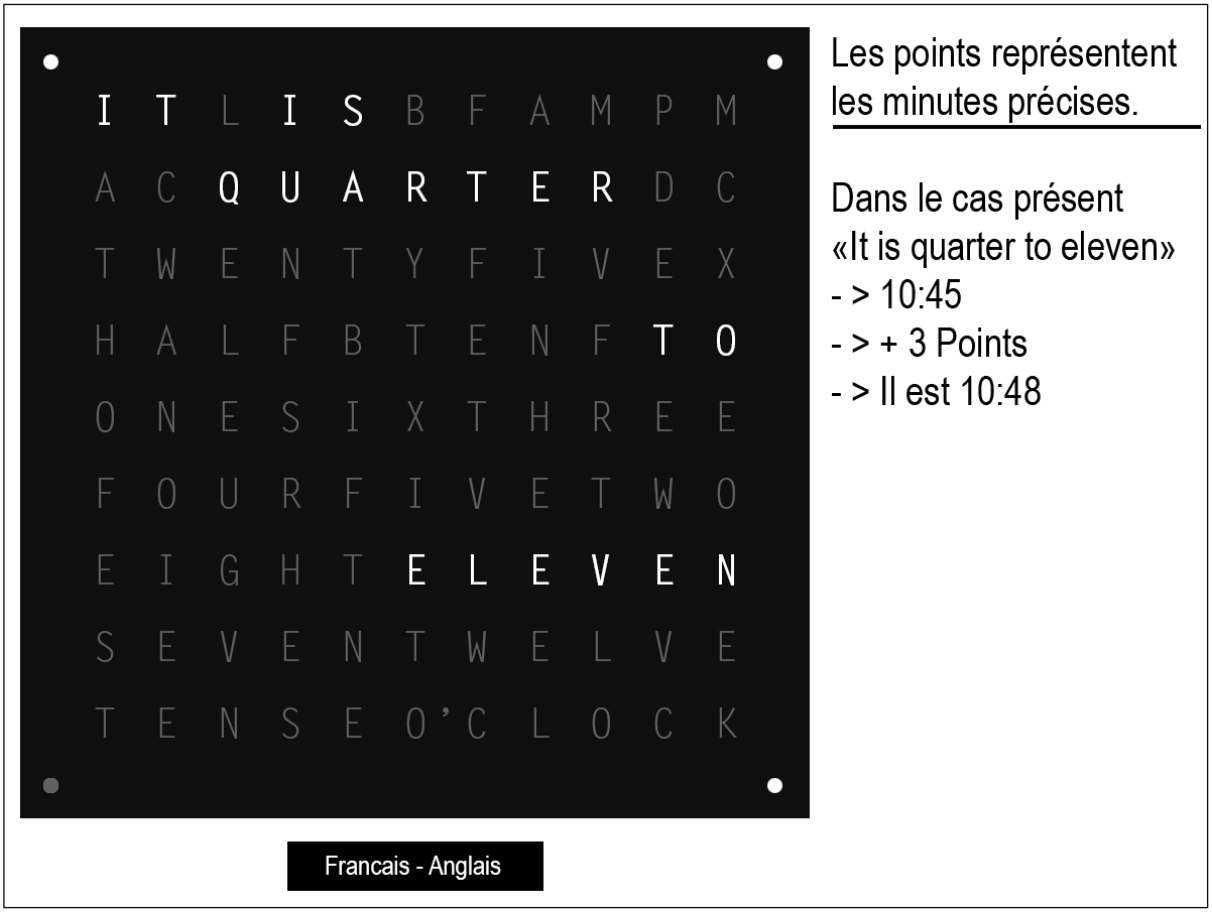
1. L'heure se met à jour automatiquement.
2. Deux langages sont disponibles à l'utilisateur.
3. Le modèle doit être adaptable à d'autres langages.
4. Les points disposés dans chaque coin indiquent les minutes précises à l'utilisateur.
5. Le design est fidèle à l'horloge originale.
6. Des transitions et des animations rendent l'interface plus agréable à l'œil.

Description de l'interface Homme-Machine

L'interface graphique de l'application sera intuitive. Une face noire sera affichée avec des mots grisés, après le démarrage de l'horloge les mots représentant l'heure actuelle seront illuminés. Le schéma ci-dessus nous permet d'observer l'ordre précis des lettres de la version française de la qlocktwo. Grace au design de l'horloge, il est envisageable de créer des versions possédant de différentes langues.

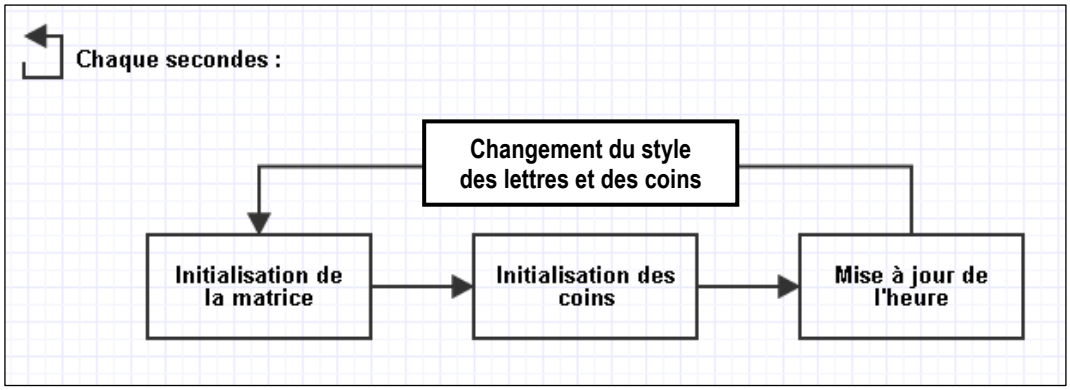
Maquette du site web

Voici une première version de la maquette de mon site web :



Il sera possible d'afficher l'horloge en deux langues à l'aide du bouton inférieur à l'horloge. Sur le schéma ci-dessus, l'heure est affichée en anglais.

Schéma de fonctionnement



L'horloge est mise à jour chaque secondes pour permettre un affichage de l'heure précis. On réinitialise la matrice chaque seconde pour éviter des bugs graphiques tels que l'affichage de lettres précédemment allumées mais qui ne sont actuellement plus pertinentes. Grâce à cette méthode, il est envisageable de créer un mode affichant les secondes actuelles, comme présent dans l'horloge « qlocktwo » originale.

Analyse organique

Généralités

Dans cette section, je décrirais les composants principaux de mon programme sous forme de pseudocode et sous forme de schéma. Les fonctionnalités avancées du programme seront aussi décrites en détail ici.

Stockage et affichage des différents langages

Le langage actuellement utilisé sera stocké dans un tableau nommé « arrayLanguage », ce tableau multidimensionnel contient les lettres qui seront affichées par la suite. Le tableau pourrait ressembler à ceci :

```
//On crée un tableau pour la langue française.
arrayLetters = [
        ["I", "L", "N", "E", "S", "T", "O", "U", "N", "E", "R"],
        ["D", "E", "U", "X", "N", "U", "T", "R", "O", "I", "S"],
        ["Q", "U", "A", "T", "R", "E", "D", "O", "U", "Z", "E"],
        ["C", "I", "N", "Q", "S", "I", "X", "S", "E", "P", "T"],
        ["H", "U", "I", "T", "N", "E", "U", "F", "D", "I", "X"],
        ["O", "N", "Z", "E", "R", "H", "E", "U", "R", "E", "S"],
        ["M", "O", "I", "N", "S", "O", "L", "E", "D", "I", "X"],
        ["E", "T", "R", "Q", "U", "A", "R", "T", "R", "E", "D"],
        ["V", "I", "N", "G", "T", "-", "C", "I", "N", "Q", "U"],
        ["E", "T", "S", "D", "E", "M", "I", "E", "P", "A", "N"]
    ];
```

On peut constater que la disposition des lettres dans le tableau est identique à celles présentes sur la tuile de couleur.

On va par la suite parcourir ce tableau et ajouter au conteneur principal le contenu du tableau. Les lettres sont toutes insérées à l'intérieur de balise « div » pour faciliter leur positionnement, on leur assigne un style « letterBox » et une classe formée de la combinaison de deux chiffres permettant de les identifier par la suite. Ceci nous permettra d'illuminer individuellement les lettres et d'avoir un design plus robuste.

Voici un code envisageable pour l'affichage des lettres à l'intérieur du conteneur :

```
    //On parcours le tableau de lettre et on crée un div de 50px/50px pour chaque lettre
    for (i = 0; i < arrayLetters.length; i++) {
      for (j = 0; j < arrayLetters[i].length; j++) {
        $("#matrix").append("<div class=\"letterBox"+" "+ i + j +"\">"+ arrayLetters[i][j] +"</div>");
      }
    }
```

La fonction « append » permet d'ajouter un élément au DOM, dans notre cas, on ajoute chaque « div » à l'intérieur d'un conteneur possédant un id (#matrix). Le résultat ressemble peut ressembler à l'image ci-contre.

```
<div class="letterBox 93">D</div>
<div class="letterBox 94">E</div>
<div class="letterBox 95">M</div>
<div class="letterBox 96">I</div>
<div class="letterBox 97">E</div>
<div class="letterBox 98">P</div>
<div class="letterBox 99">A</div>
<div class="letterBox 910">N</div>
```

Comme on peut l'observer, la classe « letterBox » est appliquée ainsi qu'une classe regroupant deux nombres concaténés. Ces nombres représentent la position verticale et horizontale de la lettre à l'intérieur du tableau. Par exemple :

`<div class="letterBox 910"></div>` -> La lettre est à la position verticale 9 (y) et horizontale 10 (x). (Il faut aussi noter que le tableau commence à 0.)

Stockage des positions des mots à afficher

Les positions des mots à illuminer sont stockées à l'intérieur d'un objet « sentence » permettant de former des phrases. Le fonctionnement général est plutôt simple, on stocke les positions à illuminer pour chaque heures et minutes (1h – 12h & 0m – 55m), puis on récupère ces dernières lors de l'affichage. Un enregistrement de position est composé d'un tableau de trois nombres entiers et se présente comme ceci :

- Le premier nombre indique la position verticale du mot
  - Ex : « 8 » pour la 9ème ligne en partant du haut du tableau.
- Le deuxième indique la position horizontale de la première lettre du mot à afficher.
  - Ex : « 5 » pour le 6ème caractère en partant de la gauche du tableau.
- Le troisième indique la position horizontale de la dernière lettre du mot à afficher.
  - Ex : « 9 » pour le 10ème caractère en partant de la gauche du tableau.

Ces informations nous permettent de savoir la position de chaque lettres à illuminer, nous utiliserons par la suite des boucles pour parcourir ces informations.

Voici à quoi ressemble l'objet sentence après avoir été complété :

```
sentence = {
        //Debut de phrase (Il est ...)
        pre: {
            all: [[0, 0, 1], [0, 3, 5]]
        },
        //Heures
        hour: {
            1: [[0, 7, 9]],
            2: [[1, 0, 3]],
            3: [[1, 6, 10]],
            4: [[2, 0, 5]],
            5: [[3, 0, 3]],
            6: [[3, 4, 6]],
            7: [[3, 7, 10]],
            8: [[4, 0, 3]],
            9: [[4, 4, 7]],
            10: [[4, 8, 10]],
            11: [[5, 0, 3]],
            12: [[2, 6, 10]]
        },
        //Minutes
        minute: {
            0: [[5, 5, 10]],
            5: [[8, 6, 9], [5, 5, 10]],
            10: [[6, 8, 10], [5, 5, 10]],
            15: [[7, 0, 1], [7, 3, 7], [5, 5, 10]],
            20: [[8, 0, 4], [5, 5, 10]],
            25: [[8, 0, 9], [5, 5, 10]],
            30: [[9, 0, 1], [9, 3, 6], [5, 5, 10]],
            35: [[6, 0, 4], [8, 0, 9], [5, 5, 10]],
            40: [[6, 0, 4], [8, 0, 4], [5, 5, 10]],
            45: [[6, 0, 4], [7, 3, 7], [5, 5, 10]],
            50: [[6, 0, 4], [6, 8, 10], [5, 5, 10]],
            55: [[6, 0, 4], [8, 6, 9], [5, 5, 10]]
        }
    };
```

Utilisation de l'objet « sentence » pour illuminer les lettres

Après avoir rempli l'objet sentence avec les valeurs désirées, on parcourt chaque catégories de l'objet (pre, heures, minutes) et on affiche la phrase finale.

Voici le code permettant d'afficher une phrase :

```
//On affiche les infos permanantes (Il est ...)
sentence.pre.all.forEach(function (y) {
  for (i = y[1]; i <= y[2]; i++) {
    $("." + y[0] + i).addClass("lightLetter");
  }
});
```

En résumé, on parcourt l'objet **`sentence.pre.all`** et pour chaque élément récupéré (y), on va ajouter la classe « lightLetter » (chaque lettre individuelle). On utilise la fonction « addClass » sur l'identifiant formé précédemment avec la position verticale et horizontale de la lettre.

De cette façon, il est très facile d'adapter le modèle à d'autres langages dû à la flexibilité du code

Récupération de l'heure actuelle au format 24 heures

Pour récupérer l'heure actuelle, j'ai développé la fonction « updateTime() » suivante :

```
function updateTime() {
    var d = new Date();
    var time = {hours: d.getHours(), minutes: d.getMinutes(), seconds: d.getSeconds()};
    return (time);
}
```

Cette dernière renvoie un objet contenant l'heure actuelle de la machine de l'utilisateur au format 24 heures, ce qui devra être changé par la suite dans le programme.

Correction de bugs d'affichages

En illuminant les lettres, j'ai pu constater divers problèmes d'affichage. Voici comment j'ai procédé pour les résoudre :

```
//Si les minutes sont au dessus de 30, on ajoute 1 a l'heure
//Ex : il est CINQ heures MOINS vingt-cinq
if (newMinutes > 30) {
    newHours += 1;
}

//On limite les heures à un format 12 heures
if (newHours > 12) {
    newHours -= 12;
}
```

Le premier problème étais le problème des « moins », si l'heure est « 10 : 40 », le programme affichait « Il est dix heures moins vingt » au lieu d'afficher « Il est onze heures moins vingt ». Pour résoudre ce problème, on ajoute une heure à la variable des heures à chaque fois que les minutes sont supérieures à trente.

Le deuxième problème était que l'heure récupérée par la fonction « updateTime() » était au format 24 heures, pour corriger ceci, on retire douze heures à la variable si l'heure actuelle dépasse les douze heures. (Exemple : Pour 14 :00, on retire 12 heures -> 14 – 12 = 2, Donc il est deux heures).

Changement de la couleur de l'horloge

Le changement du thème de l'horloge est effectué à l'aide de deux fonctions, la première « updateColor() » sert à mettre à jour la couleur des différents éléments de l'interface. On donne en paramètre une couleur, puis on applique cette dernière aux différents éléments de l'interface grâce à la fonction « .css » de jQuery.

Pour utiliser cette fonction, différents rectangles de couleurs sont disposés en bas de l'application, permettant à l'utilisateur de choisir intuitivement la couleur de l'horloge. Voici le pseudo code du changement de thème de l'horloge.

```
//Fonction mettant à jour la couleur de l'horloge
function updateColor(color)
{
    $("#matrix").css("background-color", color);
    $(".button-container").css("background-color", color);
    $(".button-color-container").css("background-color", color);
    $(".btnLanguage").css("background-color", color);
}

//Changement du thème
$(".btnColor").click(function () {
    updateColor($(this).css("background-color"));
});
```

Quand on clique sur l'un des différents boutons de couleurs, on récupère la propriété « background-color » du bouton, et on applique cette couleur aux différents éléments de l'interface grâce à la fonction « updateColor() ».

Tests

Conclusion

Bilan, améliorations envisageables

Comparaison analyse et réalisation

(Mes satisfactions, ce que j'ai appris)