CERDAN Baptiste
BLESCH Eve
L2 CMI

Mai 2018

Projet de Programmation Orientée Objet 2
Graffiti

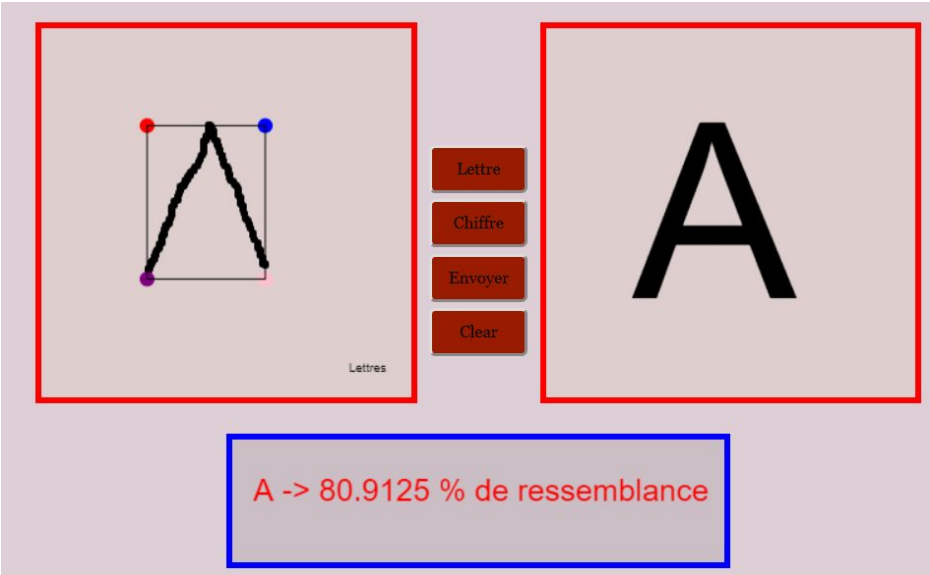
Ambiguïtés du sujet

De part le manque de précision sur ce qui était obligatoire ou non d'implémenter, nous avons interprété le fait de devoir faire une lettre d'un seul trait comme un limite de l'application. Nous avons donc essayé de contrer ce problème afin de rendre possible l'écriture des lettres en plusieurs traits (voir E, H, K, T) . Nous avons remarqué que tout le monde n'écrit pas ses lettres de la même manière, nous avons donc rendu l'application fonctionnelle quelque soit le sens d'écriture. Nous n'avons pas implémenté de caractères spéciaux qui peuvent être présents sur l'application Graffiti comme ce n'était pas précisé dans le sujet.

Choix de modélisation

Voici l'alphabet que nous avons implémenté. Certaines lettres diffèrent un peu de celles données dans le sujet pour montrer la possibilité d'écrire les lettres en plusieurs traits.

Fonctionnement de l'interface



Il faut lancer le fichier affichage.html pour lancer le programme de préférence avec Google Chrome. Pour faire fonctionner l'application, il faut cliquer sur les boutons "**Lettre**" ou "**Chiffre**" en fonction de ce qu'on veut écrire. Si aucun choix n'est fait, le mode "**Lettre**" est le mode de base. Il faut ensuite écrire dans le cadre de gauche la lettre souhaitée. Il est possible de faire autant de traits que l'on veut mais il faut garder à l'esprit que plus la lettre est dessinée simplement, mieux sera le résultat. Une fois le dessin fini, il faut cliquer sur le bouton "**Envoyer**", ce qui va afficher dans le cadre de droite la lettre correspondante. Dans le cadre inférieur sera affiché le pourcentage de ressemblance avec la lettre renvoyée. Le bouton "**Clear**" permet d'effacer tous les cadres afin d'écrire une autre lettre.

Choix de modélisation et explications de l'implémentation

Nous avons décidé de concevoir ce projet en Javascript. Pour faire les cadres de saisie nous utilisons des Canvas de 300*300 qui sont déjà implémentés en JS.

Pour modéliser les zones de lettres et de chiffres, nous avons décidé d'utiliser un seul Canva au lieu de deux. Pour cela, nous avons fait une fonction de choix (**`choixCanvas`**) qui prend en argument un entier (1 ou 2). Le bouton lettre mettra le choix en 1, et le bouton chiffre mettra le choix en 2 grâce à un événement Onclick qui appelle la fonction de choix.

- **Création des bases de données**

Nous avons créé deux fichiers de base de données (**bdd_chiffres** et **bdd_lettres**) comportant respectivement les lettres et les chiffres. Nous avons ainsi préparé pour chaque lettre 10 tableaux comme ceci avec des lettres plus ou moins bien tracées afin d'augmenter la précision de l'application. Nous utilisons la fonction **`afficher_tableau20LETTRE()`** du fichier Dessin.js qui crée le tableau (voir figure 1 qui représente un A).

```
tabA[0] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0];
tabA[1] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0];
tabA[2] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0];
tabA[3] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0];
tabA[4] = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0];
tabA[5] = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0];
tabA[6] = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0];
tabA[7] = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0];
tabA[8] = [0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0];
tabA[9] = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0];
tabA[10] = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0];
tabA[11] = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0];
tabA[12] = [0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0];
tabA[13] = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0];
tabA[14] = [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0];
tabA[15] = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0];
tabA[16] = [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1];
tabA[17] = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1];
tabA[18] = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1];
tabA[19] = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1];
```

Figure 1

- **Transformation de la lettre/chiffre en un tableau**

Dans un premier temps, nous récupérons les points de la lettre dessinée grâce aux évènements liés à la souris. Nous rangeons les points du dessin dans un tableau de 300*300 (taille de la zone du traitement). Ainsi, chaque point est retranscrit dans ce tableau à la bonne position avec des 1 quand il y a un point et des 0 autrement (voir figure 2 simplifiée). Nous avons encadré la lettre par un cadre de 4 points, ce qui permet d'avoir l'intervalle entre le point le plus à gauche/droite et l'intervalle entre le point le plus haut/bas. Cela permet de dessiner une lettre de n'importe quelle taille. Ensuite une des phases principales de notre code sera de passer d'un tableau de 300*300 à un tableau de 20*20 (rastérisation, voir figure 2) pour permettre de comparer ce tableau avec les lettres enregistrées dans les bases de données.

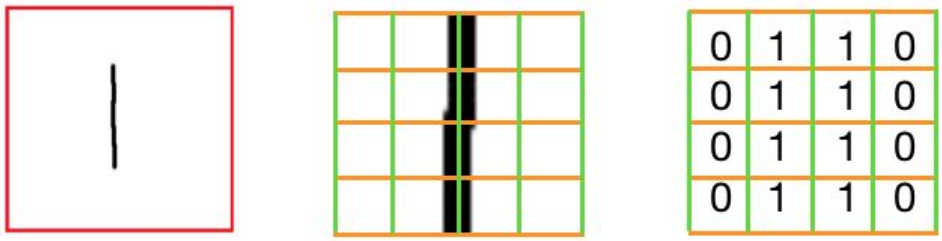


Figure 2

- **Pourcentage de similarité**

Dans le fichier **Dessin.js**, nous avons une fonction **comparer_tab** qui compare deux tableaux : le tableau de la lettre dessinée et un tableau prédéfini. Nous comparons ainsi le nouveau tableau avec tous les tableaux de la BDD. Au départ, le pourcentage est de 100%, et il diminue à chaque fois qu'il y a une différence entre chaque case du tableau pour à la fin donner un pourcentage de similarité. Nous avons également fait une fonction pour chaque lettre (`estDansA()` par exemple) qui calcule le pourcentage de ressemblance avec la lettre dessinée. Cette fonction utilise la fonction `comparer_tab` précédente. Dans le fichier **main.js**, nous récupérons les pourcentages de ressemblance avec toutes les lettres et récupérons le pourcentage maximum qui devrait correspondre à la lettre écrite.

Difficultées :

- Nous avons eu du mal à trouver l'algorithme de départ afin que ça reste tout de même de la programmation orientée objet.
- Nous avons eu des difficultés avec certaines lettres qui avaient pas mal de points en communs avec d'autres et donc elles tombaient tout le temps.

Diagramme d'héritage et de classes/prototype