Rapport de Projet de POO

Introduction

Pour ce projet, nous devons réaliser une application de reconnaissance de caractères, similaire à Graffiti, fonctionnant sur le système d'exploitation Palm OS. Cette application devrait reconnaître des lettres et des chiffres dessinés par un utilisateur. Nous avons choisi de réaliser cette application avec le langage Javascript pour la partie script, en utilisant également du HTML et du CSS pour l'interface, fonctionnant sur un navigateur web. Le programme peut reconnaître toutes les lettres en majuscule, et les chiffres, écrits de la façon la plus simpliste (similaire à la police Arial). Le projet est sous la forme d'une page web: pour y accéder, il suffit d'ouvrir le fichier **index.html** situé dans le dossier **html** de l'archive.

Présentation du projet

A l'ouverture, vous êtes accueilli avec une page, permettant d'accéder au programme. En appuyant, sur le bouton *Commencer*, vous êtes redirigés vers la page principale de l'application. Vous trouverez sur le côté gauche de cette page, une surface sur laquelle il est possible de dessiner, et sur le côté droit, certains boutons pour:

- *Effacer* entièrement la surface de dessin.
- *Changer le type du caractère* pour passer des lettres aux chiffres, et inversement.
- *Lancer la recherche* pour reconnaître le caractère dessiné, en prenant en compte le choix du type de caractère.

Lors du lancement de la recherche, plusieurs tests vont être effectués pour vérifier d'abord si le dessin est vide, ou excessivement rempli, afin de garantir les résultats les plus fiables. Un tableau s'affiche à la fin de cette recherche, qui indique une valeur d'évaluation de ressemblance entre le caractère dessiné et chaque nom de caractère, du type indiqué précédemment. Le caractère ayant la valeur d'évaluation maximale est considéré comme le caractère dessiné. Les autres caractères ayant une valeur d'évaluation positive sont coloré en vert, pour indiquer tous les caractères ressemblant au dessin. Nous avons décidé d'afficher un tableau pour pouvoir bien vérifier le bon fonctionnement du programme d'abord, mais aussi afficher tous résultats possibles si le programme ne peut pas reconnaître le bon caractère. Si vous appuyez sur le bouton *Confirmer*, ce caractère sera ajouté à une chaîne de caractères initialement vide, qui sera affiché lors du retour sur la page principale. Sinon, en appuyant sur le bouton *Retour*, vous serez seulement redirigé vers la page principale.

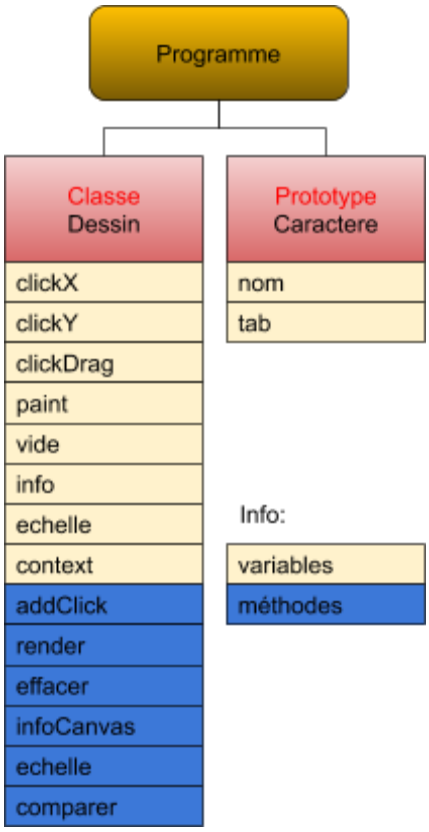
Pour la recherche d'un caractère, nous avons choisi une approche probabiliste: c'est-à-dire que le programme prépare dès son lancement, plusieurs prototypes de tous les caractères possibles, modélisés par des tableaux de taille 9x9 de probabilité de tracé. Ensuite,

si l'utilisateur dessine sur la surface et lance la recherche, le programme va lire cette surface pixel par pixel, reconnaître les limites du tracé du dessin (le point le plus haut du dessin, le plus bas, le plus à gauche et le plus à droite), afin de bien encadrer le caractère dessiné. Ensuite, nous voulons avoir un tableau de taille 9x9 du tracé effectué, pour pouvoir le comparer aux prototypes: puisque les informations de la surface de dessin sont récupérées pixel par pixel, le tableau représentant le tracé de dessin est de grosse taille. On effectue alors une mise à l'échelle de ses informations: on réduit le tout à un tableau de taille 9x9, avec dans chaque case, la densité de pixels par lequel passe le tracé du dessin. Avec ce tableau, nous allons effectuer une comparaison avec tous les tableaux de prototypes de caractères, pour voir quel prototype (donc caractère) est le plus ressemblant au caractère dessiné. Ceci donne en résultat, une valeur, pouvant être positive, si les caractères sont plus ou moins similaires, ou négative si les caractères sont assez différents. Le caractère avec la valeur maximum (donc celle qui possède le plus de similarités) est le caractère que le programme va considerer comme celui dessiné par l'utilisateur.

Ce programme fonctionne assez bien et peut reconnaître le bon caractère, si le tracé du dessin est plus ou moins correcte. En cas de doute pour le modèle de chaque caractère, vous pouvez les trouver dans les fichiers **lettres.js** et **chiffres.js** dans le dossier **javascript**. La reconnaissance est d'autant plus efficace aussi lorsqu'il y a plus de dessin à traiter. Cependant, il pourrait toujours être amélioré. Tout d'abord, pour les tableaux des prototypes, nous n'utilisons que des probabilités de 0 et de 1: nous avons essayé des tableaux avec des valeurs intermédiaires, dans l'espoir de rendre la reconnaissance plus efficace, mais cela donnait des résultats un peu plus improbables. Nous sommes donc restés sur des tableaux simples. Il est aussi possible de faire des tableaux plus grands: cela pourrait améliorer les résultats mais puisque nous mettons à l'échelle les données du dessin, l'amélioration ne serait pas significative. La taille 9x9 est déjà une façon assez simple et fonctionnelle pour représenter des caractères. Ensuite, à propos de l'évaluation de la comparaison du dessin et de chaque caractère, nous n'utilisons pas de probabilités (résultats en pourcentages) mais plutôt des valeurs décimales: pour avoir un système de pourcentages, il aurait fallu faire plusieurs changements sur la structure du programme, donc nous avons gardé un système de "valeur maximum" qui reste tout de même fonctionnel. Enfin, ce qui pourrait également être amélioré est l'équité entre les tableaux de prototypes: certains tableaux de probabilités de prototypes, possèdent plus de valeurs à 1 que d'autres, ce qui les rend plus favorables à avoir une valeur d'évaluation élevée, lors de la comparaison. Pour régler cela, nous avons développé un système de punition, lors du calcul de l'évaluation: si les cases dans les deux tableaux ne se ressemblent pas du tout, alors on enlève un point à l'évaluation. C'est pourquoi, il existe des valeurs positives et négatives pour l'évaluation des caractères. Donc même si nous avons trouvé une méthode qui fonctionne assez bien, le programme, pourrait toujours être amélioré.

Nous aurions voulu aussi créer un système pour ajouter des caractères via l'interface, mais n'avons pas pu, par manque de temps. Cela n'aurait pas amélioré la reconnaissance, mais juste proposé plus de choix de caractères.

Diagrammes d'héritage



Dans cette application nous avons utilisé plusieurs structures objets déjà implémentées par Javascript, mais nous avons également créer les structures suivantes:

La classe *Dessin* permet de créer est de traiter toute la surface de dessin. Le prototype *Caractere*, permet de donner une base pour la création des prototypes de chaque caractère.

Algorithmes principaux

D'abord, il existe la fonction *Caractère* qui va simplement servir de base pour créer chaque prototype de caractère, composé d'un nom et d'un tableau de probabilité. Ensuite, nous avons écrit un ensemble de méthodes dans la classe *Dessin* qui permettent de dessiner sur une surface (balise canvas en HTML) et également la traiter.

Lors de la création d'un type *Dessin*, cette surface est donc créée. Dans cette classe, il existe plusieurs méthodes qui forment la grande majorité des algorithmes:

- *addClick* permet d'enregistrer les coordonnées du tracé de l'utilisateur sur la surface de dessin, dans deux tableaux.
- *render* permet d'afficher sur cette surface, le tracé effectué, en utilisant les deux tableaux ayant enregistré le tracé de l'utilisateur.
- *effacer* va entièrement effacer la surface de dessin, et aussi vider les tableaux ayant enregistrés les coordonnées du tracé.
- *infoCanvas* permet de détecter tous les pixels de la surface de dessin, qui ne sont pas blancs (donc les pixels par lequel passe le tracé), va encadrer le caractère dessiné et retourner un tableau avec un tableau avec toutes les coordonnées des pixels du tracé, la colonne du pixel du tracé le plus à gauche, la ligne du pixel du tracé le plus en haut, la colonne du pixel du tracé le plus à droite et la ligne du pixel du tracé le plus en bas.
- *echelle* qui va récupérer toutes les informations retournées par *infoCanvas* et créer un tableau de densité des pixels du tracé, de même taille que les prototypes, pour pouvoir comparer ce tableau avec les prototypes facilement.
- *comparer* est la méthode principale de la recherche des caractères. La méthode va récupérer un tableau mis à l'échelle et un tableau de tous les prototypes de caractères à comparer. Ensuite, elle va créer une table pour afficher les résultats, et la remplir au fur

et à mesure de la comparaison du tableau mis à l'échelle avec chaque prototype. Dans la table seront créées alors des rangées avec les noms des caractères comparés et une évaluation indiquant une valeur de ressemblance avec le prototype et le dessin mis à l'échelle. Une boucle va parcourir les deux tableaux (mis à l'échelle et prototype) en même temps et va incrémenter l'évaluation, initialement nulle, avec la multiplication des deux valeurs aux emplacements appropriés des tableaux. Si à un emplacement d'un des tableaux, la valeur est nulle mais pas dans l'autre, alors l'évaluation est décrémentée de 1. A la fin de cette méthode, on a donc un table avec une évaluation pour chaque caractère comparé.

Répartition du travail

Pour réaliser ce projet, nous avons séparé les tâches de cette façon: Mika se concentre sur la partie de la création des prototypes et de l'algorithme de comparaison des caractères, alors que Adonis, s'occupe de créer toute la structure nécessaire pour la création de dessin. Ensuite, nous avons regroupé nos idées et au fur et à mesure, nous avons tout les deux amélioré le travail de l'autre, jusqu'à arriver au résultat final. Pour tout ce qui concerne l'interface, nous avons tous les deux travaillé dessus. Le travail, en général, a été réparti équitablement.

Difficultés rencontrées

Lors de la réalisation de cette application, nous n'avons pas eu de difficulté majeure. Grâce aux outils déjà à disposition avec Javascript et HTML, il a été assez facile de créer l'interface et le traitement d'image. A propos du script pour la reconnaissance d'un caractère, l'aspect mathématique de la résolution du problème prenait parfois un peu de temps de réflexion, mais rien de significatif tout de même. Ce projet a été assez intéressant à réaliser.