IFT3051 : Projet d'Informatique

Conception et développement d'un système de soumission de tests pour mesurer l'oreille absolue.

Par :
Mathieu Lemoine
LEMM28068701
mathieu.lemoine@umontreal.ca

Sommaire

Introduction

Dans le cadre des neurosciences, l'étude de différentes aptitudes particulières permettent d'affiner la connaissance de l'organisation du cerveau et de son fonctionnement. Une des particularités étudiées dans le domaine de l'étude musicale est le phénomène de l'oreille absolue. Avoir l'oreille absolue est la faculté de reconnaître rapidement, la note et l'octave d'un son, sans aucune aide extérieure. La détection de cette aptitude se fait en mesurant le temps de reconnaissance d'une série (en ordre aléatoire) de notes et l'exactitude des réponses.

Ces mesures ont deux buts :

Démontrer que l'Oreille Absolue n'est pas une propriété binaire (i.e. qu'il existe différents degrés et différents types de manifestations de cette capacité).

Détecter les personnes jouissant de cette capacité à un certain niveau afin de pouvoir par la suite étudier la présence de singularités divers au niveau du schéma cérébrale et/ou psychique, ce qui permettrait certaines avancées dans l'étude du cerveau humain.

Le laboratoire de Neurosciences Cognitives de l'université de McGill étudie (entre autres) ce phénomène.

Le but de ce projet est de développer une application permettant une mesure du temps de réaction suffisamment précise (idéalement moins de 5ms de décalage) et facilement transportable de machines en machines (afin de pouvoir effectuer les tests sur n'importe quel machine ou presque).

Analyse de la problématique

La problématique se divise en deux points qui seront étudiés séparément :

Précision de l'application

Transportabilité

Précision de l'application

Dans le cas d'une application classique, plusieurs éléments interviennent dans la perception des entrées des utilisateurs par l'application :

1. Le matériel utilisé par l'utilisateur pour interagir avec la machine (clavier et souris par exemple)
2. Les circuits électroniques reliant le matériel d'entrée/sortie (IO) au processeur et au cœur fonctionnel de la machine.
3. Le système d'exploitation et sa gestion de l'IO (ici interviennent notamment les drivers des périphériques d'IO).
4. La communication entre le système d'exploitation et l'application via divers bibliothèques et processus de communication.
5. La gestion de l'exécution de toutes les applications présentes sur le système à un instant donné.

Pour chacun de ces points, une solution est envisageable :

1. Création ou Modification d'un matériel existant spécialisé afin de réduire le temps de réaction matériel.
2. Création d'un circuit et d'une machine spécialisés afin de traiter de manière optimale l'IO et de réagir de manière quasi-instantanée à l'arriver d'un signal d'IO.
3. Pour ce point, deux solutions sont envisageables :
    a. Création d'un système d'exploitation minimal consacré à ces tests, basé sur un noyau Linux par exemple.
    b. Écriture de drivers spécialisés permettant de limiter le temps de réaction et de traitement des signaux.
4. Pour ce point, trois niveaux sont à considérer :
    a. Utilisation d'un langage permettant de se placer très haut dans le chaînage des évènements venant du système d'exploitation.
    b. Utilisation d'une bibliothèque minimisant ses traitements internes d'évènements.
    c. Utilisation de techniques de programmation limitant le traitement intermédiaire et privilégiant le traitement utile.
5. Pour ce point, deux actions peuvent maximiser la réactivité de l'application :
    a. Minimiser le nombre d'applications s'exécutant sur la machine au moment du test.
    b. Maximiser la priorité de l'application par rapport aux autres applications.

Étant donné que le projet doit être effectué en un temps restreint, et que l'optimisation d'un des points entraine plus ou moins directement l'obligation de travailler sur tous les points suivants, le choix s'est porté sur les points 4 et 5, c'est-à-dire, les points purement applicatifs.

L'application devant être graphique, la première recherche a été de chercher un langage et une bibliothèque permettant de minimiser le nombre d'actions intermédiaires à la réception d'une opération d'IO.

Dans cette optique, des langages interprétés ou à machine virtuel tels que Flash ou Java on été immédiatement mis de côté. Et C s'est très vite imposé comme le langage permettant la plus grande réactivité, du fait de sa proximité avec le matériel.

C ne disposant pas de bibliothèque standard pour les applications graphiques, il a ensuite fallu choisir un moyen d'afficher les interfaces nécessaires. Or la réactivité d'une bibliothèque graphique n'est pas un point souvent utilisé pour les caractériser et les comparer, de nombreuses recherches n'ont pas permis de trouver des informations très fiables sur les bibliothèques. Finalement, la consultation de diverses personnes travaillant dans les interfaces graphiques (notamment Sébastien Roy, professeur à l'Université de Montréal, http://www.iro.umontreal.ca/~roys/fr_index.shtml) a dégagé le langage de scripts Tcl/Tk comme permettant de s'insérer très haut dans le chaînage des communications avec le Système d'Exploitation.

Ce langage est interprété mais d'une façon très particulière. A son chargement, le script est compilé, pour ensuite être exécuté directement en mémoire, il est donc aussi rapide qu'un langage compilé classique dans le cadre de traitements simples.

De plus ce langage est interfaçable avec C, ce qui permet de développer une partie de l'application en C/C++ et l'autre en Tcl/Tk.

Pour des raisons de simplicité et d'efficacité, les parties non critiques de l'application seront finalement codées en C/C++, tandis que les parties graphiques ou critiques seront faites avec Tcl/Tk.

De plus l'application s'octroiera (grâce à des fonctionnalités fournies par le système d'exploitation) une priorité maximale pour pouvoir réagir au plus vite, et prendre le pas sur les autres applications pouvant s'exécuter en même temps.

Il faudra bien sûr minimiser ces dernières afin d'éviter tout délai supplémentaire.

Transportabilité de l'application

L'application doit pouvoir être facilement transportable (sur une clé USB ou un autre stockage facilement transportable) et pouvoir être utilisée et enlevée du système sans nécessiter d'opérations complexe.

Le but est donc de rassembler tous les fichiers nécessaires dans un dossier complet, permettant ainsi le lancement directement depuis l'espace de stockage.

Il faut également minimiser la taille afin de pouvoir stocker l'application facilement. Ainsi, la limite haute serait de 512 Mo (taille minimale pour une clé USB actuelle), mais beaucoup moins serait largement préférable.

Les langages C et C++ étant compilés vers un seul exécutable natif, il est donc très facile de transporter une application C/C++. Dans un souci de simplicité, l'environnement d'exécution de l'application a été restreint au système d'exploitation Microsoft Windows®, qui est très largement présent dans le parc informatique actuel.

Les langages Tcl et Tk, quand ils sont interfacés avec C sont manipulés via une bibliothèque (API), qui est présente sous la forme de deux DLL.

De plus, il y a la bibliothèque standard Tcl/Tk qui permet certaines manipulations et/ou actions et qui doit également être prise en compte.

Les divers fichiers nécessaires à l'application tels que les images, les sons et les fichiers de configurations sont aussi à inclure dans ce package.

Il faudra s'assurer que l'application puisse s'exécuter sans soucis dans un environnement indéterminé ou faiblement caractérisé.

Approche détaillée des solutions

Approche de l'API Tcl/Tk pour C

L'API Tcl/Tk pour C et plus précisément son emploi et son fonctionnement détaillé sont très peu documentés. Ainsi, certaines fonctionnalités proposées par le langage (et son interpréteur standard) ne sont pas disponibles ou pas fonctionnelles via l'API. Et ces dysfonctionnements ne sont pas présentés dans la documentation de l'API.

Ces détails mis à part, la bibliothèque fait principalement la compilation puis l'interprétation du code Tcl/Tk.

Elle n'est pas mutli-threadés, autrement dit, il n'y a pas de fil d'exécution supplémentaire qui soit lancé par l'API et qui soit chargé de traiter les évènements Tcl/Tk. L'avantage principal est que l'application n'aura pas à jongler entre plusieurs threads, ce qui pourrait dégrader les performances et la réactivité de l'application aux opérations d'IO.

Uniformisation de la précision

Du fait que l'application doive pouvoir tourner dans des environnements disparates et que chaque être humain n'a pas le même temps de réaction immédiat, les résultats risquent de varier fortement d'une batterie de tests à l'autre et de ne pas reflété correctement le temps de reconnaissance de la note.

De ce fait, un étalonnage de l'application a été prévu avant chaque batterie de test, de même afin d'uniformiser les tests, certaines informations sur le matériel (notamment l'écran) devront être fournies, afin que les images affichées aient toujours la même taille à l'écran.

Présentation de l'application

Documentation de l'application

L'application est documentée en détaille sur son fonctionnement et son utilisation.

Cette documentation est disponible en Annexe.

Utilisation de l'application

L'application a été faite afin d'être le plus simple possible d'utilisation. Et son exécution se déroule en quatre étapes simples :

1. Configuration par l'opérateur
2. Étalonnage par le sujet
3. Soumission des tests
4. Sauvegarde des résultats

L'interface est en anglais car le logiciel est destiné au laboratoire de neuroscience cognitives de l'Université de McGill, qui est anglophone.

La documentation est disponible en Français ou en Anglais.

Discussions et proposition d'extension

L'application possède une précision et une cohérence les plus hautes possibles du fait des langages, techniques et bibliothèques utilisées. De plus, l'ensemble des fichiers ne dépasse pas les 18 Mo.

Cependant, certains points pourraient encore être perfectionnés, et certaines voies qui pourraient peut-être fournir de meilleurs résultats n'ont pas été explorées.

Transportabilité de l'application

Présentement, l'application est très facilement transportable, occupant moins de 20 Mo d'espace de stockage (pouvant être réduit jusqu'à 6 Mo en la compressant) et ne nécessite aucune installation.

Cependant, du fait que la bibliothèque soit liée à l'application de manière dynamique implique la nécessitée d'inclure plusieurs fichiers. Certaines versions de la librairie semble permette une liaison statique ce qui permettrait de se débarrasser de ces fichiers et de diminuer la taille de l'application.

Les scripts Tcl/Tk et la bibliothèque étant interprétés; les sources sont disponibles de manière claire avec l'application, ce qui provoque à la fois une augmentation conséquente du nombre de fichiers nécessaires et le risque de la modification du code Tcl/Tk de l'application.

Il semble exister des solutions (payantes) permettant de compiler le code Tcl/Tk vers du code natif, ce qui diminuerais encore le nombre de fichiers nécessaires, et augmenterais peut-être légèrement la vitesse d'exécution de l'application.

Précision de l'application

L'application présente une précision la plus haute possible du fait des choix effectués, et une grande cohérence des résultats du fait de l'étalonnage.

Cependant, certaines voies non explorées permettraient sûrement d'améliorer encore cette précision en :

- Développant un système d'exploitation (basé par exemple sur le noyau Linux/Unix) et étant spécifiquement optimisé pour ce type d'opération.
- Créant un appareil électronique de test ne comportant qu'un circuit minimaliste pour faire le test et pouvant se connecter à un ordinateur classique (via USB ou port sériel, par exemple).

Portabilité de l'application

L'application est présentement prévue pour s'exécuter sous Microsoft Windows®.

Des modifications mineures seraient nécessaires pour la rendre compatible avec d'autres systèmes d'exploitation.

Par exemple, pour les systèmes de type Unix, ces simples modifications seraient nécessaires :

- Récupération d'une version pour Unix de l'API Tcl/Tk
- Recompilation de l'application pour le système en question
- Modification de l'appel système permettant de modifier la priorité de l'application

Le problème principal réside dans les restrictions au niveau de la priorité de l'application, certains Unix (Linux par exemple) interdisant d'augmenter la priorité d'utilisation pour un utilisateur classique (l'opération est réservée au super-utilisateur/administrateur *root*).