HM40

Rapport de projet

Plateforme d'évaluation loi de Fitts et évolution vers l'analyse GOMS/Keytroke

Elaboré par:

Alexandre Viala

Yiwen Jiang

Le 22/04/2022

Sommaire

Rappel du sujet

Notre projet est basé sur ce que nous avons appris en cours sur la loi de fitts et l'analyse GOMS/Keytroke. Nous avions à notre disposition, plusieurs codes sources d'application permettant d'expérimenter la loi de Fitts et nous devions en choisir un et l'améliorer.
Nous avons choisi de prendre le code source de l'application fitts n°6 et l'avons modifié. Dans le processus de réalisation du projet, nous avons utilisé le logiciel qt creator sur linux.
Le projet avait deux objectifs principaux, le premier était de finaliser une solution pour évaluer et tester la loi de fitts et le second était de l'intégrer dans une plateforme logicielle plus générique qui intégrerait les interactions d'évaluation selon le modèle GOMS/keystroke.
Nous n'avons cependant pas réussi à mettre en place une solution pour ce deuxième objectif étant donné que nous avons concentré nos efforts sur l'amélioration de l'interface de l'application Fitts. Nous avons néanmoins réalisé quelques maquettes afin de présenter ce que nous aurions pu réaliser.

La loi de fitts

La loi oi de Fitts est une loi mathématique prédictive. Cette loi porte le nom de son inventeur : Paul Fitts, psychologue de l'Université d'État de l'Ohio. Pour modéliser de manière mathématique, le mouvement humain, il invente cette loi dont voici le principe :
Le temps nécessaire pour aller rapidement d'une position de départ à une zone finale de destination dépend de la taille de la cible et de la distance entre l'utilisateur et celle-ci.

$$T = k \log_2(D/S + 0.5) \text{ avec}$$

k un facteur dépendant de la personne (~ 100 msec),
D la distance entre l'utilisateur et la cible
et S la surface de la cible.

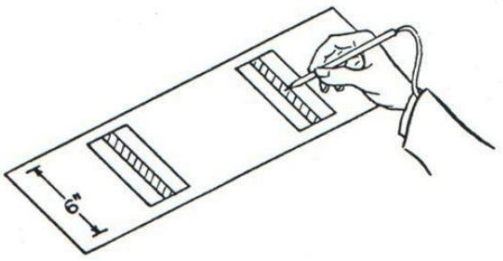
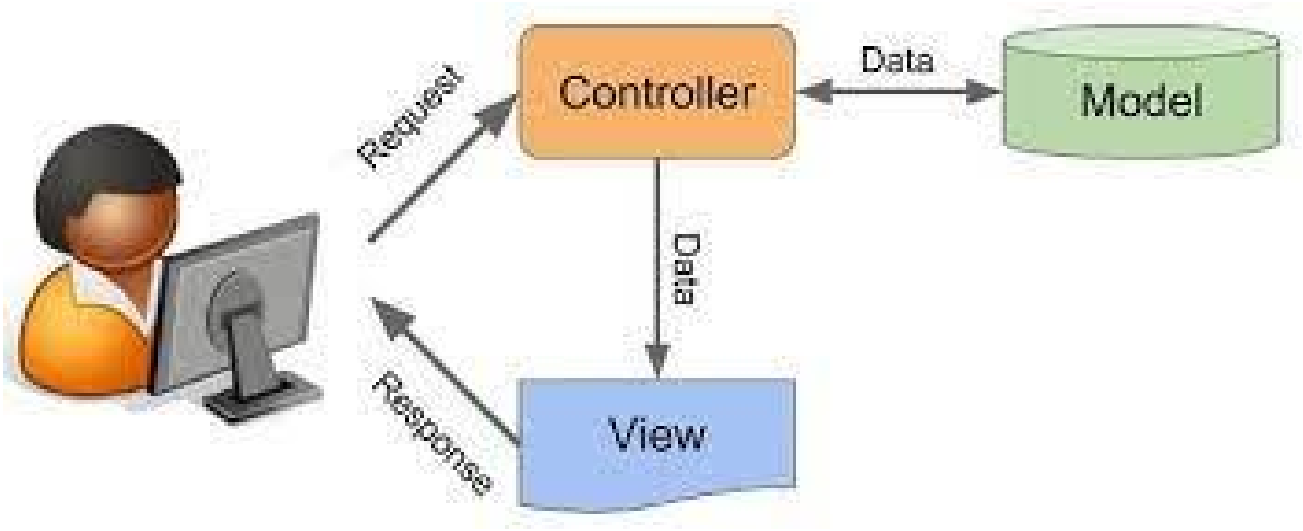


FIG. 1. Reciprocal tapping apparatus. The task was to hit the center plate in each group alternately without touching either side (error) plate.

Le système M-V-C (Model-View-Controller)

Au travers de ce projet, nous avons utilisé le modèle MVC afin d'améliorer ce logiciel. Ce dernier utilisait déjà cette architecture et les fonctions étaient dans l'ensemble bien réparties dans le modèle, la vue et le contrôleur. Nous avons simplement rendu plus cohérent le modèle en répartissant ces méthodes dans le fichier adéquat.



Le motif est composé de trois types de modules ayant trois responsabilités différentes : le modèle, la vue et le contrôleur.

Le modèle

Le modèle (model) est la partie du programme qui contient l'ensemble des données ainsi que toutes les méthodes permettant l'acquisition et l'écriture de données.
Dans notre projet, le modèle va stocker toutes les informations sur la prise de données en cours comme le nombre de cibles restantes, la position et la taille des cercles mais aussi certains résultats après analyse comme l'erreur-type sur les séries de données ou la différence moyenne.
Nous stockons également les données sauvegardées dans un fichier JSON dans le répertoire caché "**.config**" de l'utilisateur (répertoire standard pour stocker des données d'application). C'est également le modèle qui se charge de lire et d'écrire les données dans ce fichier JSON afin de communiquer avec l'application. Le fcihier JSON se présente comme suit :

```
[
    {
        "a": 0.2,
        "b": 0.1,
```

```
        "cercleCenter": [
            {
                "x": 126,
                "y": 87
            },
            ……
        ],
        "cercleSize": [
            142,
            101,
            …
        ],
        "cibleLeft": 0,
        "clickPoints": [

        ],
        "dateTime": "22/04/2022 - 02:29",
        "diffMoy": 71.64848680481997,
        "ecartType": 55.59132130226261,
        "erreurType": 14.353617439821912,
        "itc95": 28.707234879643824,
        "maxSize": 160,
        "minSize": 50,
        "nbCible": 15,
        "times": [
            715,
            507,
            ……
        ]
    },
      ……
]
```

La vue

La vue (view) est l'interface graphique de l'application. Il s'agit de tout ce avec quoi l'utilisateur va interagir directement. Cela comporte évidemment le visuel, avec les couleurs, les groupements, et le ressenti global. Cependant, la vue prend en compte les boutons ainsi que les possibles intéractions avec ces derniers. la vue n'a néanmoins aucune utilité toute seule car elle n'est pas censé gérer les actions qu'il faudra effectuer, son but est purement esthétique.

Le contrôleur

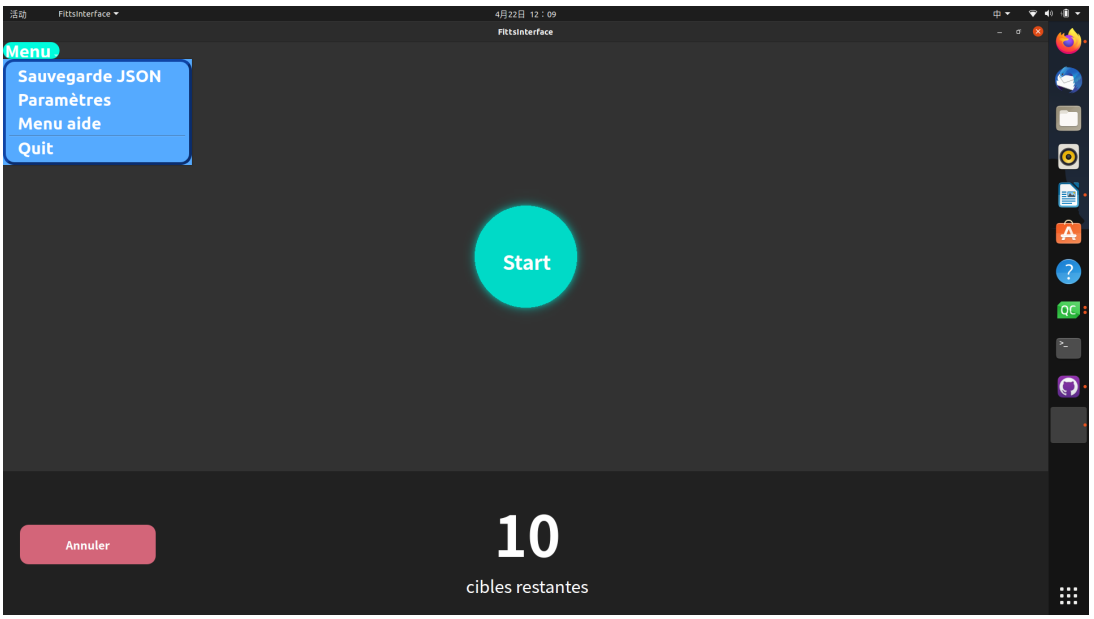
Le contrôleur (controller) est l'élément pivot du système MVC. Il permet de faire le lien entre les vues et les modèles. Il contient toute la logique concernant les actions effectuées par l'utilisateur. Si la vue va être chargée d'émettre l'intégralité des signaux, le contrôleur va, quant à lui, les réceptionner en intégralité et faire fonctionner toute la logique qu'il y a derrière. C'est dans ce composant du système qu'on retrouve tout le calcul qui se base sur les données envoyées par l'utilisateur et les données stockées dans le modèle.
Dans notre projet, le contrôleur va par exemple être chargé de calculer les valeurs des courbes expérimentales et théoriques mais ne va pas les afficher lui même (il fera appel à la vue avec la méthode **displayResults()**).

Les fonctionnalités ajoutées

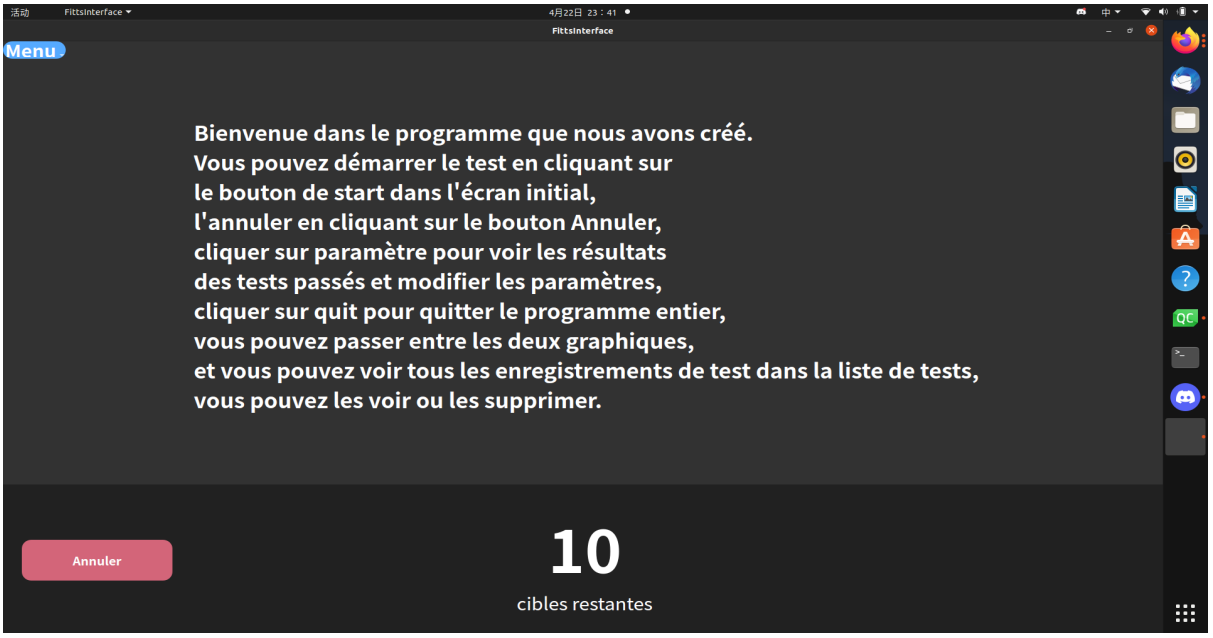
Ajout de menu

Nous avons ajouté un menu à l'interface et les avons conçus de manière à ce que lorsque nous cliquons sur un menu, celui-ci se déroule pour nous donner les options disponibles.

Lorsque nous cliquons sur le bouton Paramètres, le programme passe à l'écran des paramètres, où nous pouvons modifier les différentes options que nous voulons changer.
Lorsque nous cliquons sur le bouton Quit, le programme se termine.
Le bouton de sauvegarde permet de sauvegarder le graph actuellement affiché sur la vue.



Ajout d’un menu aide



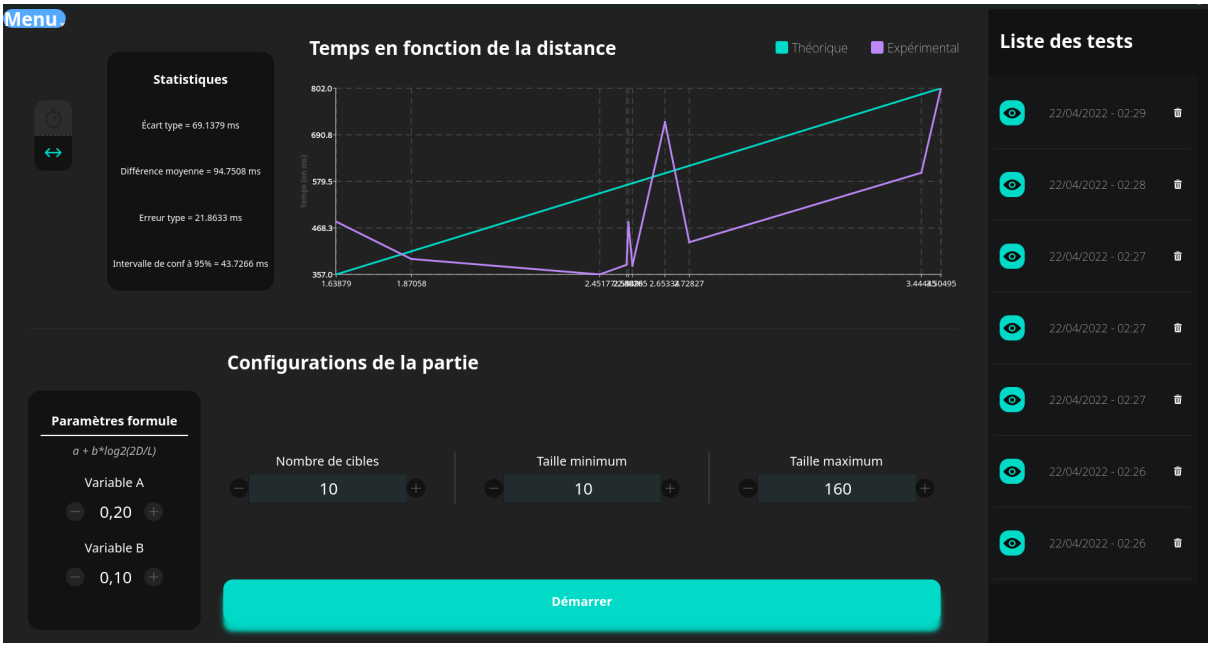
Lorsque nous cliquons sur le bouton Aide, une fenêtre contenant une description générale du programme s'ouvre. Cette fenêtre ne s’affiche que sur la page de test car nous avons considéré qu’elle n’était pas nécessaire sur la page d’exploitation des données. Le but de cette page d’aide et surtout de comprendre comment effectuer le test.

Ajout de commentaires détaillés

Dans notre code, nous avons ajouté de nombreux commentaires pour aider le lecteur à mieux comprendre le code et faciliter sa maintenabilité. Nous avons notamment ajouté des commentaires au-dessus des blocs de code destinés aux différents boutons et widgets afin de plus facilement scinder les fonctionnalités à l'intérieur du programme.

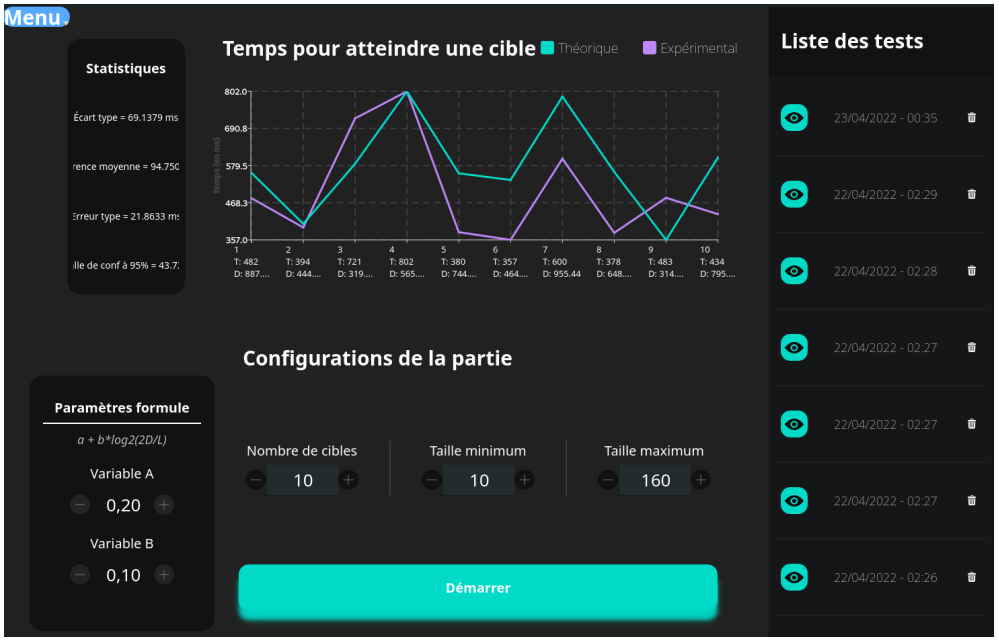
Interface « responsive »

Nous avons essayé au cours de ce projet de rendre l'interface davantage "responsive". Une interface dite "responsive" est une interface qui est ajustable à la taille de l'écran. Il est normalement relativement aisé de la redimensionner comme bon nous semble pour s'adapter à l'écran de chacun mais également être disponible en mode fenêtré.
Nous avons réussi à rendre certaines parties de l'interface davantage"responsive" mais nous avons cependant été limité sur de nombreux points. En effet, l'interface que nous avons choisi d'adapter présentait ses données d'une manière relativement claire sur un seul écran qui fait également office d'écran de paramétrage (cf. image ci-dessous). La constitution des groupes sur cet écran de paramètre et leur positionnement nous a paru très cohérent : il est relativement simple de repérer les différents paramètres à rechercher et l'espacement entre les différents groupes facilite grandement la lecture.



Bien que cette interface soit relativement simple à lire, elle présente un défaut : elle est difficilement ajustable sans faire disparaître des informations.
Nous avons donc fait le choix de limiter l'ajustement de la fenêtre pour éviter que la compression rende illisible notre interface.

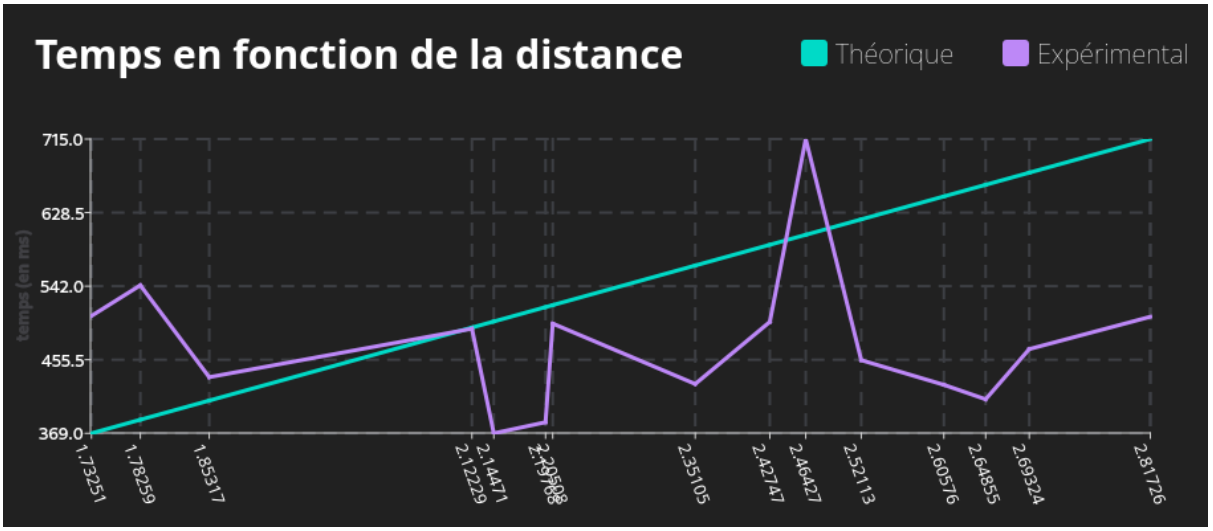
Résultat après compression :



Changements apportés sur les graphes

Facilitation de la lecture des données sur le graphe T = f(log(2D/L))

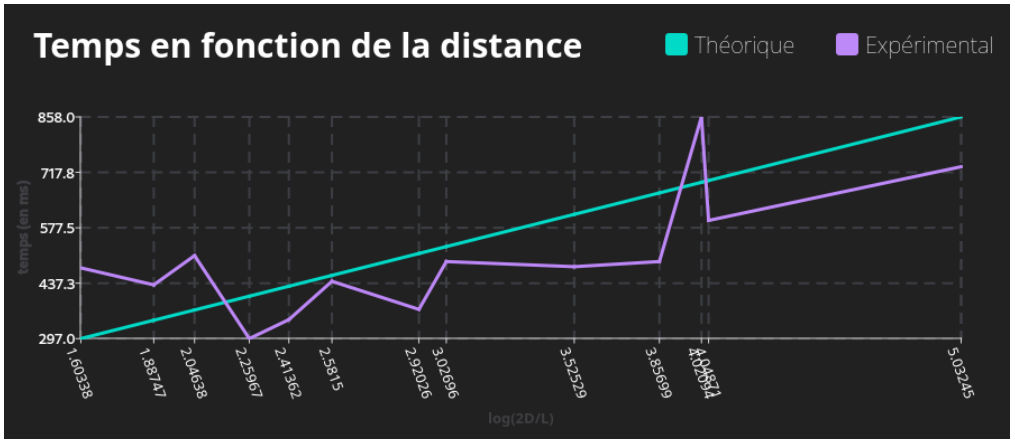
Nous avons ajouté une modification très simple sur le graphe affichant le temps en fonction de la distance. Sur ce graphe, le principal problème était qu'il n'était pas possible de voir correctement les données car ces dernières se chevauchaient souvent. Afin de remédier au problème, nous avons simplement mis un angle appliqué à ces données afin de les rendre plus lisible.



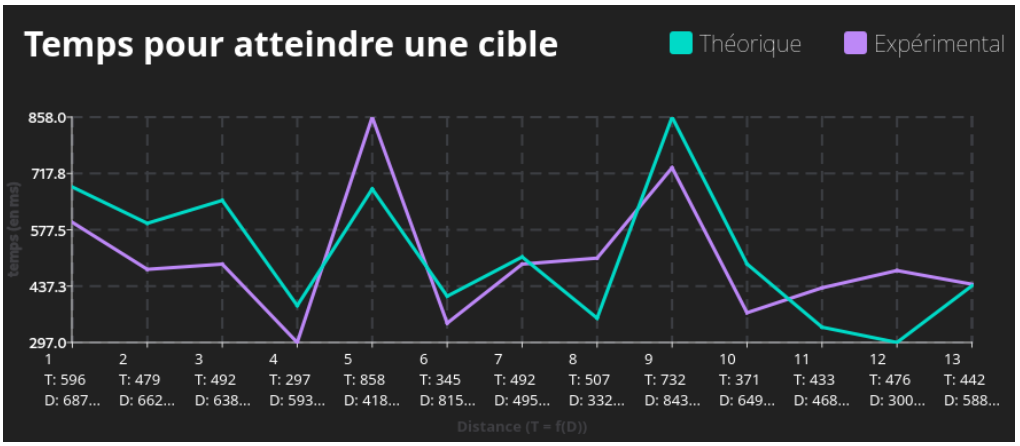
On remarquera que certaines valeurs se chevauchent encore car ces données sont bien trop proches.

Facilitation de la lecture des axes des graphes

La loi de Fitts n'est pas une loi évidente à comprendre. Néanmoins avant nos modifications, il était impossible de comprendre correctement quelles valeurs étaient affichées en abscisse. Nous avons donc décidé d'ajouter un un titre sur ces graphes afin de mieux comprendre à quelle loi nous sommes confrontés.



Graphe T = f(log(2D/L))



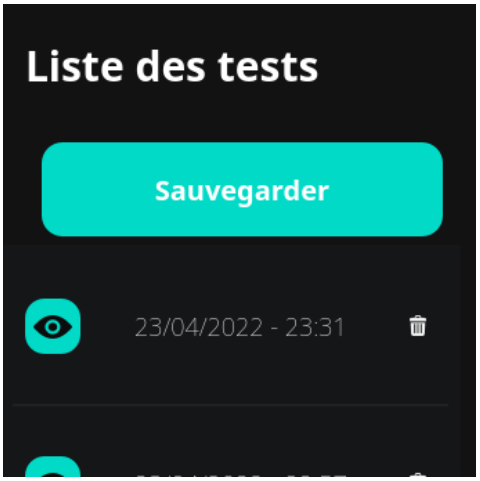
Graphe T = f(D)

Ajout de sauvegarde au format JSON

Dans le projet dont nous disposions, certaines données étaient déjà stockées au format JSON mais les techniques utilisées présentaient quelques défauts:

- Les temps mesurés lorsque l'utilisateur clique sur une cible n'étaient pas mesurés.
- L'interface graphique ne permettait que de visualiser la présence de sauvegarde sans pouvoir visualiser leur contenu ou les supprimer.
- Les sauvegardes étaient enregistrées automatiquement à la fermeture du logiciel.

Afin de résoudre ces différents points faibles, nous avons donc reliés les boutons pour chaque sauvegarde à des signaux et des slots personnalisés afin de pouvoir gérer l'index de la sauvegarde dans le fichier JSON.
Nous avons ensuite rajouté les données manquantes dans le fichier JSON par l'intermédiaire du modèle.
Enfin, nous avons désactivé la sauvegarde automatique des données et nous avons rajouté deux possibilités pour laisser l'utilisateur sauvegarder : par l'intermédiaire du menu ou par l'utilisation d'un bouton de sauvegarde. Si nous avons choisi de permettre cette interaction par deux moyens, c'est pour permettre à l'utilisateur de plus facilement trouver la fonctionnalité qu'il recherche.



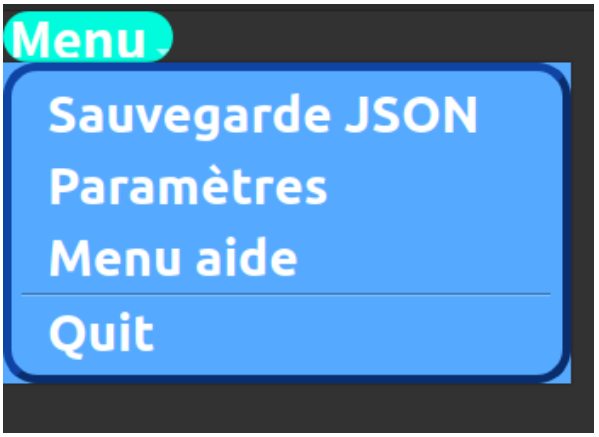
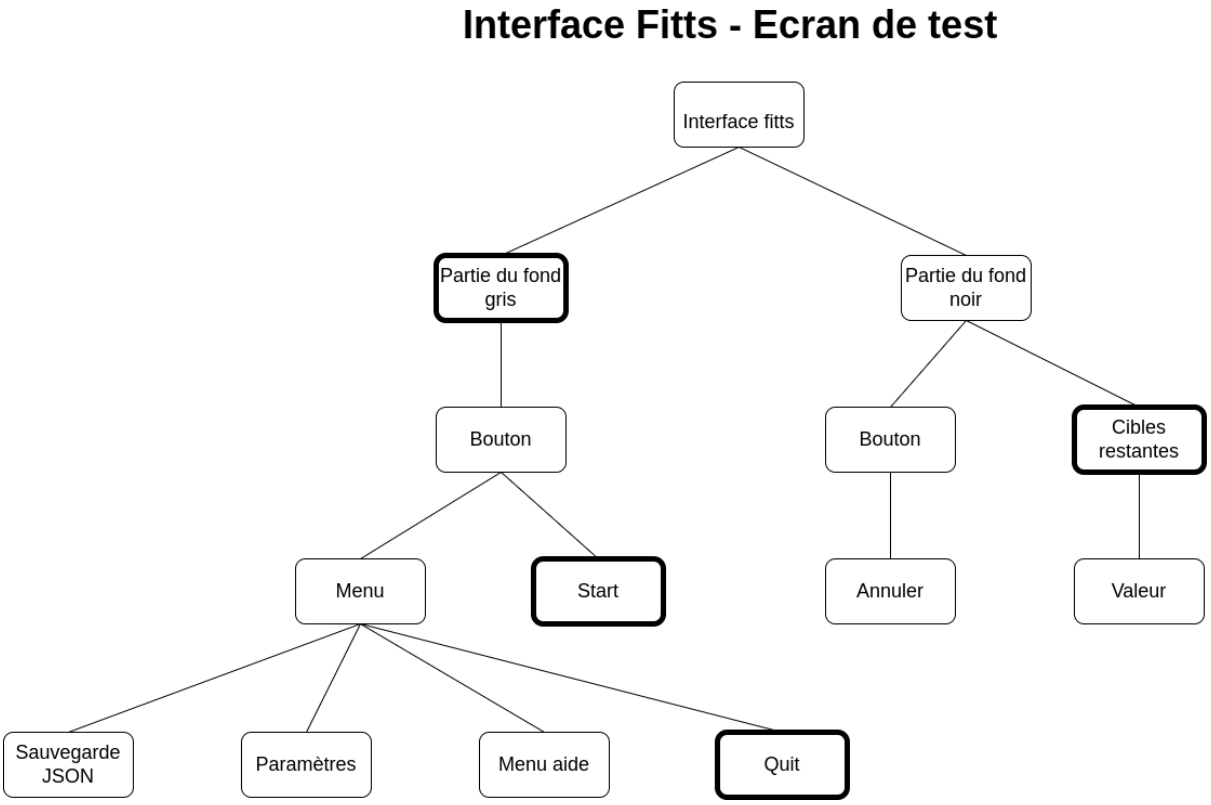


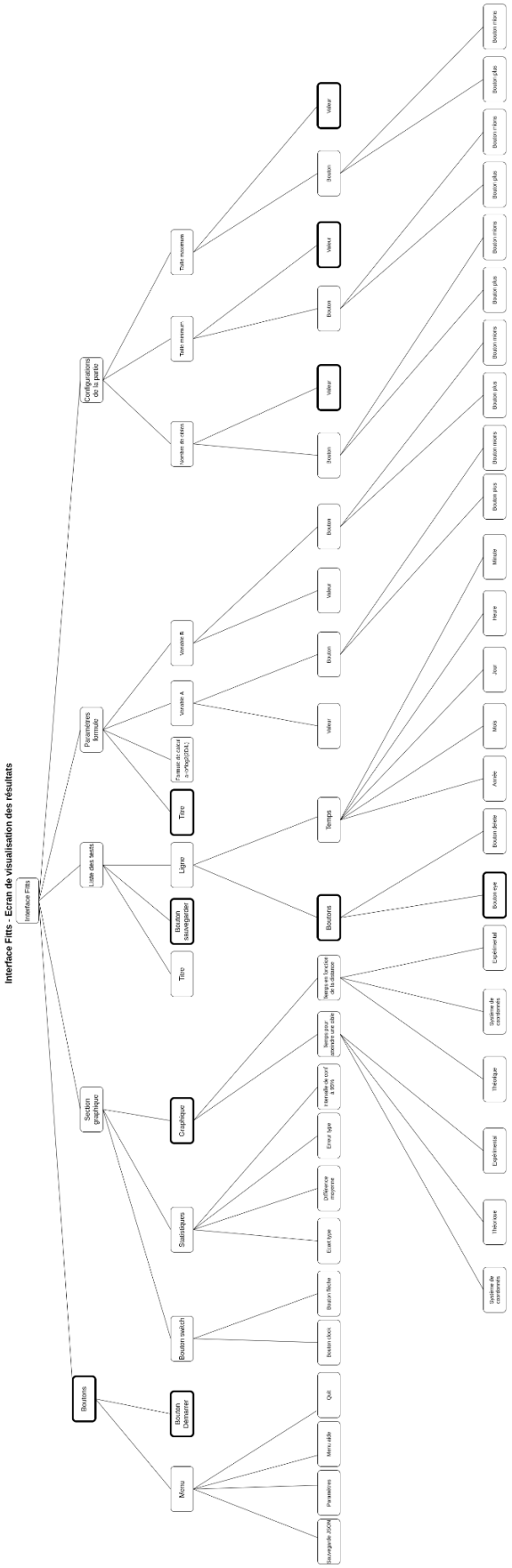
Diagramme de structure de l'interface proposée

Nous avons utilisé le logiciel draw.io pour dessiner les diagrammes de structure de l'interface proposée. Nous avons réalisé un diagramme de structure pour chaque écran de notre application. Nous avons représenté les sujets pragmatiques en gras afin de mieux les différencier.

Interface Fitts - Ecran de test:



Interface Fitts - Ecran de visualisation des résultats:



Vers l'ajout d'une évaluation de la loi de GOMS/Keystroke

Au cours de ce projet nous étions censés ajouter une interface permettant d'évaluer la loi de GOMS/Keystroke. Cependant, la tâche s'est avérée plus ardue que nous le pensions. Nous n'avons donc pas pu implémenter cette interface sur notre logiciel.
Néanmoins, nous avons réfléchi à une manière d'évaluer la loi de GOMS/Keystroke. C'est au travers de maquettes que nous allons vous présenter ce que nous pensions mettre en place avec davantage de temps.

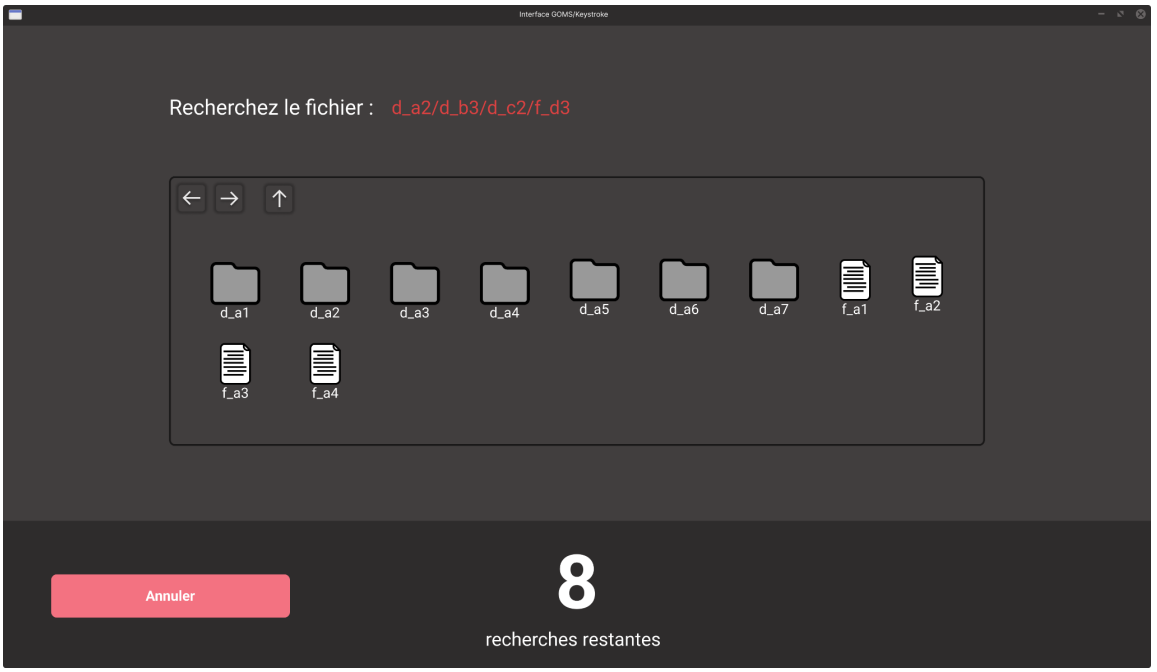
Menu pour l'intégration dans une application commune

Tout d'abord, nous devions trouver un moyen de lier l'application GOMS/Keystroke, c'est pourquoi nous avons opté pour l'ajout d'un menu au démarrage de l'application qui nous permet de choisir quelle partie de l'application nous voulons démarrer.

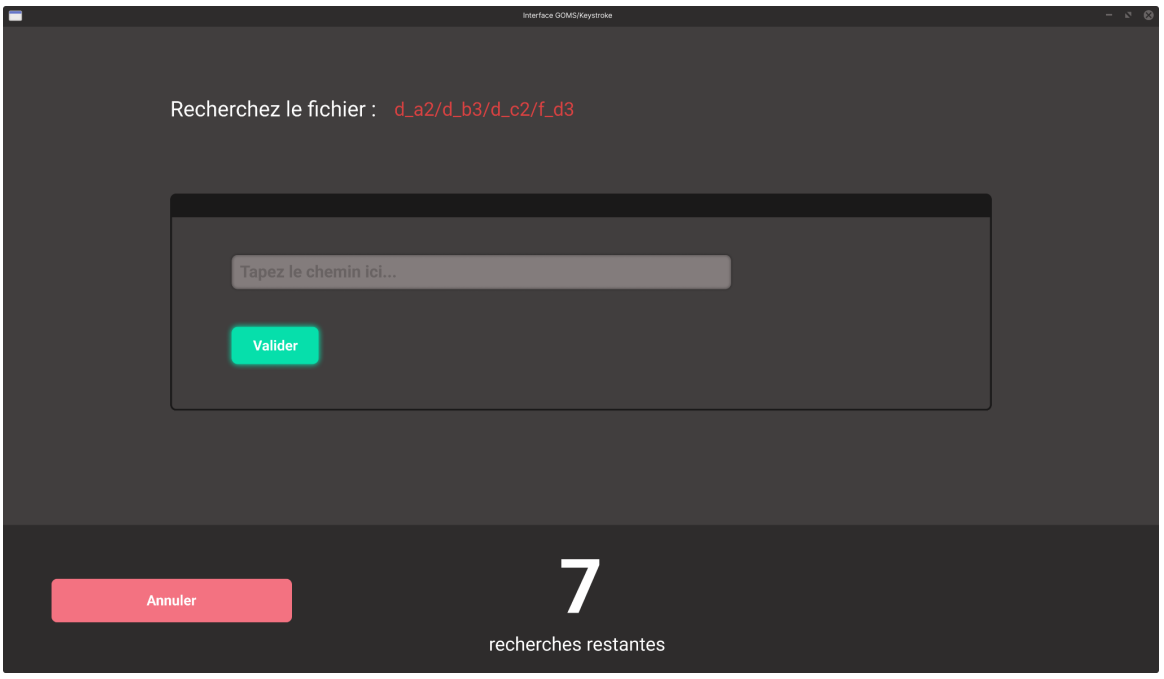
Méthode d'évaluation

Une fois que nous avions réfléchi à une manière de relier nos deux applications, nous devions mettre en place un protocole permettant d'évaluer la loi de GOMS/Keystroke.
Nous nous sommes donc inspirés du travail que nous avions réalisé aux cours de nos travaux dirigés où nous devions calculer le temps nécessaire pour la recherche d'un répertoire ou d'un fichier dans une arborescence de fichier à l'aide de la méthode GOMS/Keystroke.

Nous avions donc prévu de mettre en place une simili-interface d'explorateur de fichiers pour permettre à l'utilisateur d'évaluer par la méthode graphique, la loi de GOMS/Keystroke. Nous avons ensuite rajouté un champ de texte pour laisser l'utilisateur taper le nom du répertoire dans un champ afin d'évaluer la méthode textuelle.
Nous avons fait le choix de donner une taille fixe à chaque dossier ou répertoire afin de faciliter le calcul de la loi de GOMS/Keystroke.
L'interface reprend beaucoup d'éléments graphiques de l'interface de l'évaluation FItts sur laquelle nous avions travaillé afin de proposer une unicité visuelle.



Interface pour évaluer la méthode visuelle



Interface pour évaluer la méthode textuelle

Visualisation des données

Afin de visualiser les données, nous pensions adopter un affichage similaire à celui que nous avions pour la visualisation de Fitts. Néanmoins, étant donné que la loi de GOMS/Keystroke n'est qu'une somme de somme de durées, elle ne se prête pas à un affichage sous forme de droite affine.
Nous aurions pu opter pour un diagramme à barre mais nous aurions perdu en unicité avec l'interface testant la loi de Fitts. Nous avons choisi de n'afficher qu'une seule courbe pour qu'elle soit plus visible.
Sur cet écran, nous pensions également ajouter la gestion des paramètres en permettant de modifier les temps qui rentrent en ligne de compte dans la loi de GOMS/Keystroke. Ces paramètres peuvent dépendre de l'utilisateur et il paraissait donc évident de faire en sorte que l'utilisateur puisse les faire varier.

L'évaluation de l'interface

Nous pensons que notre interface est facile à utiliser et à comprendre car elle est simple et comporte des instructions très claires (texte ou graphiques) sur la manière de l'utiliser, de sorte que même un utilisateur novice peut rapidement s'y retrouver. Notre interface est simple à utiliser, elle est donc efficace et facile à retenir pour les utilisateurs.
L'interface tient également compte des éventuelles erreurs de l'utilisateur, de sorte que si celui-ci clique accidentellement sur le bouton de démarrage, il peut l'annuler en appuyant sur le bouton d'annulation, ou supprimer le résultat d'expérience s'il n'est pas satisfait des données qu'il a obtenues.
Nous pensons que notre interface est assez agréable d'un point de vue esthétique et nous avons stylisé les différents boutons, les images, etc. afin que l'utilisateur soit satisfait de l'interface lorsqu'il l'utilise. Nous pensons que l'interface est également flexible, car l'utilisateur peut ajuster les paramètres pour obtenir l'effet qu'il souhaite.

Bibliographie

- Définition de la loi de Fitts (et illustrations) : https://www.usabilis.com/definition-loi-de-fitts/
- Explication du motif d'architecture MVC : https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur
- Illustration du motif d'architecture MVC : https://helloacm.com/model-view-controller-explained-in-c/