ENSSAT

LANNION

École Nationale Supérieure des Sciences Appliquées et de Technologie de Lannion

EyeTracker

Anatole Faugère

Diplôme d'ingénieur spécialité informatique

Rapport Final Mars - 2023

Année universitaire 2022 - 2023

Tuteurs :
Vincent Barreaud
Emmanuel Casseau

1 Résumé

Ce projet a pour objectif de développer un assistant de communication pour les personnes handicapées moteur. L'utilisateur pourra sélectionner par le regard une des réponses proposées sur un écran. L'application développée est basée sur la technologie Tobii-eye tracker permettant de détecter l'endroit où se situe le regard sur une interface graphique.

Cette application permet également à un aidant d'enregistrer un ensemble de questions et leurs réponses associées dans une base de données, pour une utilisation ultérieure. Deux interfaces graphiques ont été réalisées en *Python l'une plus ergonomique qui est l'interface finale, l'autre plus simple qui sert à effectuer les tests de l'application. Suite à la commande tardive de la *Tobii-eye tracker par l'*Enssat, le projet à pris un peu de retard. La *Tobii-eye tracker est en cours d'intégration dans l'application.

Mots-clés : *Tobii-eye tracker, interface graphique, *Python

2 Abstract

This project aims to develop a communication assistant for people with motor disabilities. The user will be able to select by sight one of the answers offered on a screen. The developed application is based on *Tobii-eye tracker technology to detect where the gaze is located on a graphical interface.

This application also allows a caregiver to record a set of questions and their associated answers in a database, for a later use. Two graphical interfaces have been created in *Python : a more ergonomic one which is the final interface, another one, more simple, that is used to perform the tests of the application. Following the late order of the *Tobii-eye tracker by *Enssat, the project took a little delay. The *Tobii-eye tracker is currently being integrated into the application.

Keywords : *Tobii-eye tracker, graphical interface, *Python

3 Remerciement

En premier lieu, je tiens à remercier M.Vincent Barreaud et M. Emmanuel Casseau mes tuteurs, pour leurs sympathies, leurs conseils tout au long du projet.

Je remercie Mme. Nathalie Caradec, ma professeure d'expression-communication, pour son enseignement et ses explications.

Un grand merci mes camarades de classes pour leurs conseils ainsi que leur soutien inconditionnel, et moral.

Table des matières

Table des figures

Glossaire

***API** estune interface de programmation d'application (application programming interface). 11

***Conteneur** est une unité logicielle standard qui regroupe le code et toutes ses dépendances afin que l'application s'exécute rapidement et de manière fiable d'un environnement informatique à un autre (docker.com). 12

***Docker** est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels. (Wikipédia). 12

***Enssat** : École Nationale Supérieure des Sciences Appliquées et de Technologie.. 1, 6, 7, 11, 24

***GitHub** est un service d'hébergement Internet pour le développement de logiciels et le contrôle de version à l'aide de Git. 13

***Jenkins** est un outil d'automatisation utilisé pour créer et tester des projets logiciels. 20

***Python** est un langage de programmation interprété, multiparadigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. (Wikipédia). 1, 11, 12, 14, 17

***QML** (Qt Modeling Language) est un langage de balisage d'interface utilisateur. 17

***SDK** est un ensemble d'outils d'aide à la programmation d'applications mobiles (Software Development Kit). 11

***Tobii-eye tracker** est une technologie de suivi oculaire permettant d'utiliser vos mouvements des yeux comme une entrée supplémentaire dans un système, en complément du clavier et de la souris. 1, 7, 8, 11, 12

***Unity** est un moteur de jeu multiplateforme développé par Unity Technologies. 11

.

4 Introduction

J'ai effectué ce projet de septembre 2022 à mars 2023, dans le cadre du projet long en 3^{eme} année à l'*Enssat. Ce projet a été l'opportunité pour moi d'appréhender toutes les étapes de conception, de developpement et de test d'un logiciel.

Dans un premier temps, j'aborderai la description du projet EyeTracker ainsi que la problématique générale de celui-ci. Puis, j'expliciterai les problématiques techniques du sujet ainsi que ses spécifications. Enfin, je me pencherai sur l'état d'avancement du projet jusqu'à ce jour.

5 Description du sujet

5.1 L'Enssat

L'*Enssat, (École Nationale Supérieure des Sciences Appliquées et de Technologie) est une école d'ingénieurs dans les domaines de l'informatique, des systèmes numériques et de la photonique. L'*Enssat est une école affiliée à l'Institut Mines-Télécom. Mais il s'agit aussi d'un centre de recherche reconnu pour ces projets nationaux et internationaux. De plus ses étudiants participent à de nombreux projets locaux, proposés soit par l'école soit par la communauté locale. Il se trouve que ce projet a été présenté par un habitant dans la région.

5.2 Le contexte

Effectivement ce projet a été proposé par le grand-père de Yoan. Il se trouve que Yoan est une personne avec un fort handicap physique : il ne peut communiquer que par le regard. Le grand-père possède une *Tobii-eye tracker 4C, un appareil qui permet de détecter où se pose le regard sur un écran, et il souhaitait l'utilisé pour développer une application lui permettant de communiquer facilement avec son petit-fils. Ce projet a été soumis M. Casseau qui la transformé en sujet de projet long pour les 3^{eme} année à l'*Enssat.

5.3 Les objectifs et les contraintes

Le grand objectif du projet EyeTracker est de construire un assistant de communication qui a l'aide d'un eye tracker, va capter le regard de la personne sur un écran et l'afficher par un point de couleur. Ainsi sur une interface graphique on affiche différentes propostions de réponses à une question, la personne avec l' handicap peut via son regard sélectionner une réponse.

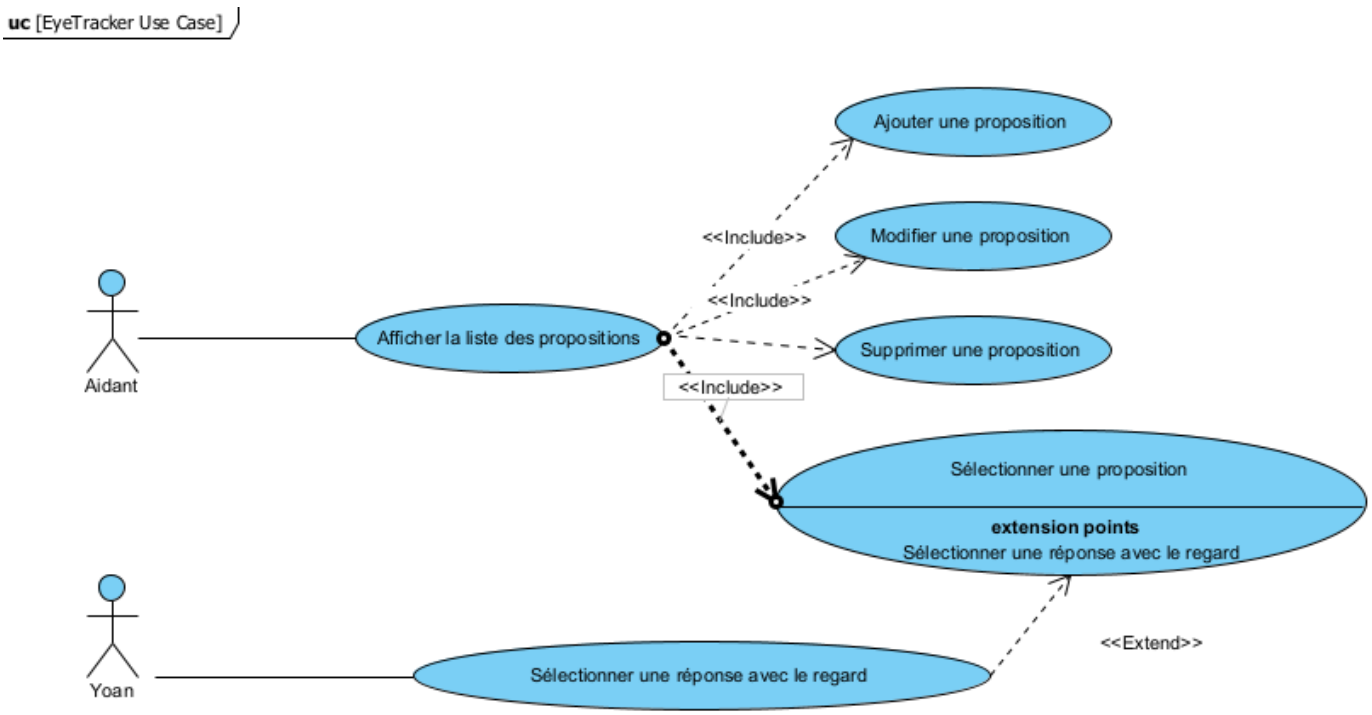

FIGURE 1 – Use Case

A l'aide ce diagramme des cas utilisateurs, on identifie facilement les besoins primaires de l'applications : Afficher les propositions, en ajouter, les modifier ...
Ici on appelle une "proposition" un ensemble contenant une question et 2 à 4 réponses associées.

Les contraintes du projet sont les suivantes :
— Le grand-père de Yoan possède une *Tobii-eye tracker. Notre système devra ré-utiliser cette technologie.
— L'application devra être extremement simple et intuitive lors de son utilisation.

5.4 Planning prévisionnel

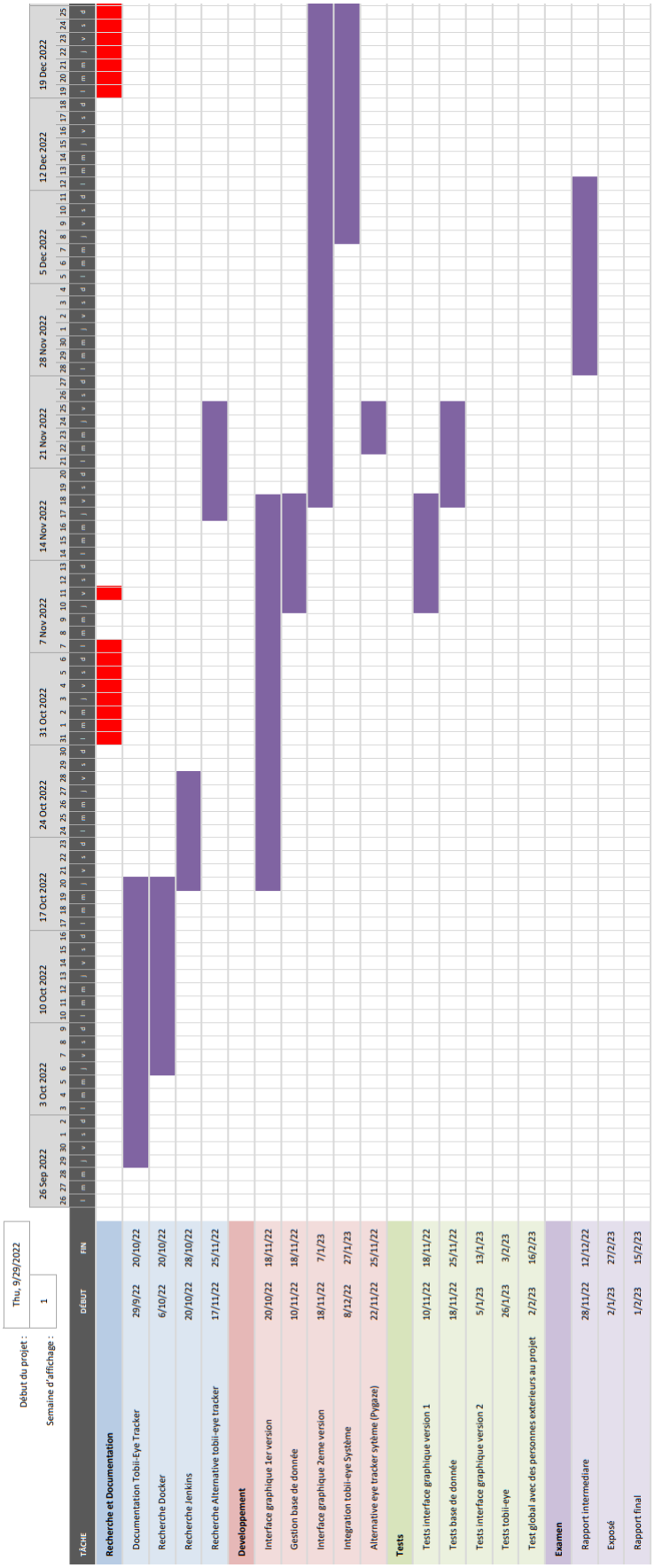

FIGURE 2 – Diagramme de gantt partie-1

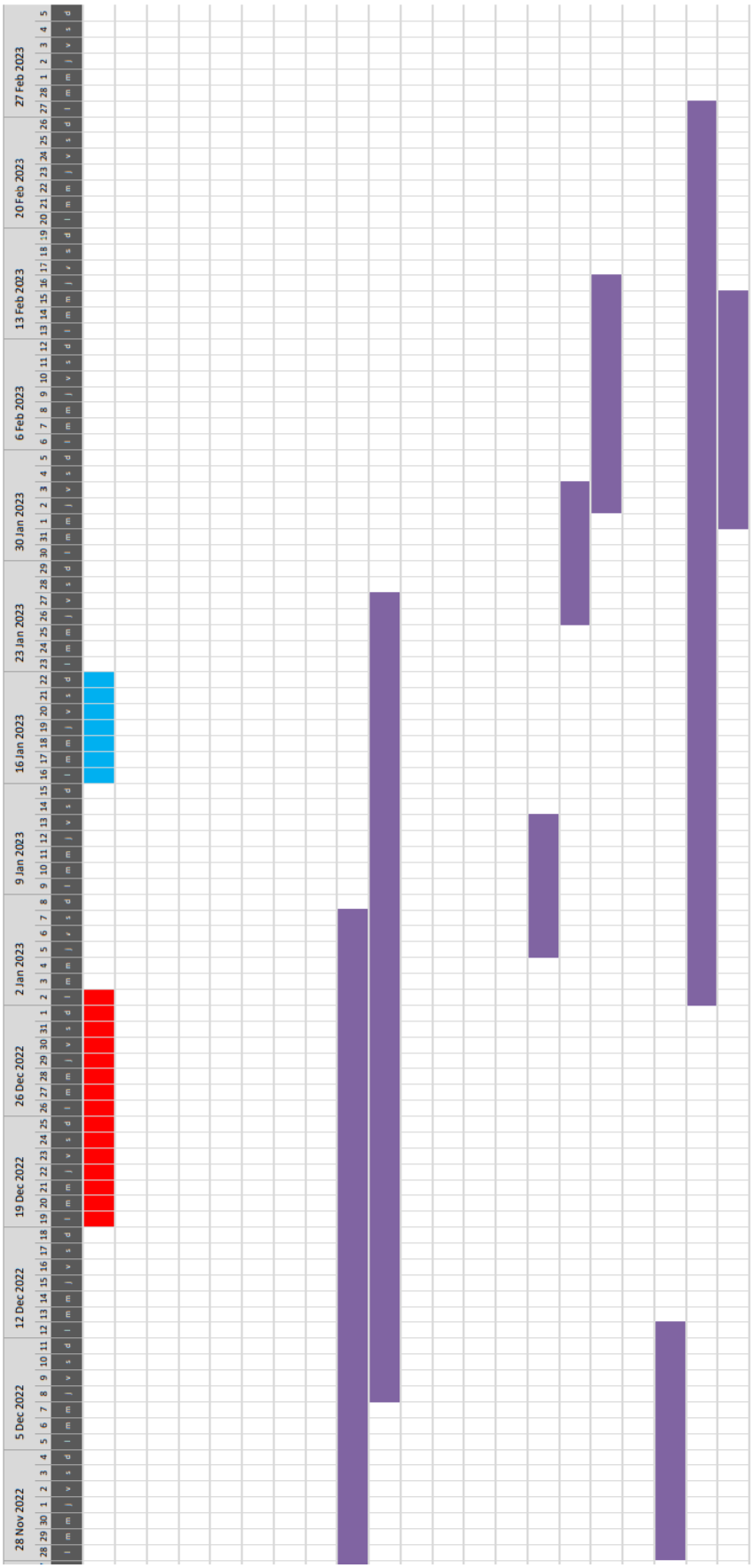

FIGURE 3 – Diagramme de gantt partie-2

6 Problématique et spécifications du projet

6.1 La Tobii-eye tracker

La *Tobii-eye tracker est une technologie de suivi oculaire permettant d'utiliser le mouvement des yeux comme une entrée supplémentaire dans un système, en complément du clavier et de la souris.[4]

Le grand-père de Yoan possède une *Tobii-eye tracker 4C, mais cette version n'est plus en vente actuellement : on est maintenant à la 5^{eme} géneration. J'ai donc fait une étude compartive entre la version 4C et la version 5.

	Tobii Eye 4C	Tobii Eye 5
Driver and software	À installer manuellement	Déjà installé sur la machine
Logiciel	Tobii Core	Tobii Expérience
Taille	335 mm	285 mm
Capteur	IS4 avec capteur NIR standard	IS5 avec capteur Tobii NIR personnalisé
Champ de vue	38*29 degrés	40*40 degrés
Taille de l'écran supporté	27" 16 :9 or 30" 21 :9 *	27" 16 :9 or 30" 21 :9 *
Taux d'échantillonnage de l'image et fréquence du regard	90Hz	133 Hz
Récupération du regard	Sélecteur de mode pour récupérer le regard	Récupération continue

[5]

Après analyse, on remarque que toutes les fonctions de l'*API de la Tobii-eye que l'on utilisera seront les même pour les version 4C et 5.

De plus la *Tobii-eye tracker, peut être utilisée sur les systèmes Windows, MacOS et Linux. Elle peut être programmée en *Python, Matlab, Octave, C, ou en C# avec *Unity [3]. Me spécialisant plus en *Python et C#, j'ai privilegié le *Python afin qu'un autre étudiant puisse poursuivre le projet et le compléter dans les années futures.
(le C# et *Unity n'étant pas enseigné à l'*Enssat).

J'ai utilisé la licence : "Getting Started development license" [2]. Cette licence est destinée à un usage non commercial et à la non distribution. Il s'agit d'une licence limitée qui fournit des droits de développement avec les *SDK Tobii et les logiciels et l'*API [1] liés à l'oculométrie.

6.2 L'environnement

6.2.1 Docker

J'ai conteneurisé mon application, car les *Conteneurs permettent l'isolation des applications entre elles et du système sous-jacent. Ils permettent aussi la portabilité, puisque les applications n'ont pas à être liées au système d'exploitation hôte. Pour cela j'ai utilisé le système de gestion de *Conteneur *Docker. Mon *Conteneur possède une image Debian plus une image *Python3.8 ainsi que différents packages nécessaires au fonctionnement de la *Tobii-eye tracker. Ce contenaire possède 2 ports vers l'extérieur : le premier pour l'affichage graphique de l'application, et le second pour se connecter à la Tobii.

Pour la partie Docker de mon projet, j'ai mis en place une connexion entre l'interface graphique de ma machine hôte et le container Docker en installant et configurant XLaunch. Pour télécharger XLaunch, j'ai suivi les étapes suivantes :

1. Je me suis rendu sur le site officiel de Xming à l'adresse suivante : `https://sourceforge.net/projects/xming/files/latest/download`
2. J'ai téléchargé le programme d'installation de XLaunch en cliquant sur le lien de téléchargement proposé sur la page.
3. J'ai exécuté le programme d'installation en double-cliquant dessus et j'ai suivi les instructions à l'écran pour installer XLaunch sur ma machine.

Une fois l'installation terminée, j'ai lancé XLaunch et j'ai configuré les paramètres de connexion en suivant ces étapes :

1. J'ai sélectionné l'option "Multiple Windows" pour permettre l'affichage de plusieurs fenêtres graphiques dans le container Docker.
2. J'ai choisi le paramètre "Start no client" pour lancer XLaunch sans l'ouverture d'une fenêtre graphique.
3. J'ai activé l'option "No Access Control" pour permettre à mon container Docker d'accéder à l'interface graphique de ma machine hôte.

J'ai utilisé deux fichiers Dockerfile : Dockerfile.1 et Dockerfile.2, correspondant aux interfaces 1 et 2.
Pour les exécuter, j'ai utilisé les commandes suivantes :

```
#!/bin/bash
docker build -f Dockerfile.1 -t eyetracker_python_test .
docker run -d -it --name eyetracker -e DISPLAY
   =172.29.208.1:0 -v C:\Users\anato:\anato --privileged
   -v /dev/bus/usb/:/dev/bus/usb eyetracker_python_test
```

La première commande construit l'image Docker à partir du fichier Dockerfile.1 et la nomme "eyetracker_python_test".

La seconde commande exécute cette image dans un conteneur Docker nommé "eyetracker", en spécifiant l'adresse IP de l'interface graphique de ma machine hôte (172.29.208.1 :0) pour l'affichage des fenêtres graphiques, le répertoire partagé entre ma machine hôte et le conteneur Docker (`C:\Users\anato`) et les privilèges nécessaires pour accéder aux périphériques USB (`/dev/bus/usb/`).

J'ai suivi la même procédure pour exécuter le Dockerfile.2 que pour le Dockerfile.1, en utilisant les mêmes options de la commande "docker run". Les fichiers

Dockerfile.1 et Dockerfile.2 sont disponibles en annexes.[A]

6.2.2 Git

Enfin je stocke mes codes sources sur *GitHub : https://github.com/FAtole/EyeTracker.

Les interfaces se trouvent respectivement dans les dossiers "projecteyetracker" et "ProjectEyeTracker2". Ces dossiers contiennent un fichier "assets" qui contient les images, les données, la licence et les pages de QML pour la seconde interface.

Le dossier "src" contient les scripts Python de l'application. Les fichiers sont organisés de la manière suivante :

"main.py" : lance l'application
"reader_csv.py" : gère la base de données des questions et des réponses

Pour lancer la premier il faut appeler la commande :

```
python projecteyetracker/src/main.py
```

Pour la seconde :

```
python ProjectEyeTracker2/src/main.py
```

7 L'état d'avancement

7.1 Développement de l'interface

Afin de répondre aux exigences du projet, il m'a paru indispensable d'avoir une interface graphique utilisateur. En *Python il existe differents libraries pour construire une application de bureau avec une interface graphique : tkinter, PyQt, WxPython, Kivy ...

La première version de l'interface a été effectuée avec tkinter [6] car c'est framework très simple qui permet d'avoir un résultat trés rapidement en *Python.
Cette interface possède l'intégralité des fonctionnalités attendues.

Pour installer tkinter, vous pouvez utiliser la commande :

```
pip install tkinter
```

La classe Window est une sous-classe de Tk, la classe principale de la bibliothèque tkinter. Elle hérite ainsi de toutes les propriétés et méthodes de Tk.
On y initialise la base de données des propositions. On définit également la fenêtre en elle-même, sa taille, son titre, ses polices de caractères, ses images d'icônes, etc.

La classe Window dispose également d'une méthode $show\_frame()$, qui permet de basculer entre différentes frames (pages) de l'application en cachant les frames actuellement affichées et en affichant celle passée en argument.

Attention : Configuration requise pour la pleine utilisation de l'application

Si vous utilisez Windows ou Mac, veuillez utiliser la commande suivante :

```
self.state('zoomed')
```

En revanche, si vous utilisez Linux, vous devez utiliser la commande pour activer le mode plein écran.

```
self.attributes('-fullscreen', True)
```

Dans le code suivant, il y a quatre classes qui représentent différentes pages de l'application :

1. `page_accueil` : Cette pages liste les propositions.
2. `page_reponses` : Cette page affiche les réponses et permet à l'utilisateur d'en sélectionner une.
3. `page_add_proposition` : Cette page permet à l'utilisateur d'ajouter une nouvelle proposition.

4. `page_modifie_proposition` : Cette page permet à l'utilisateur de modifier une proposition existante.

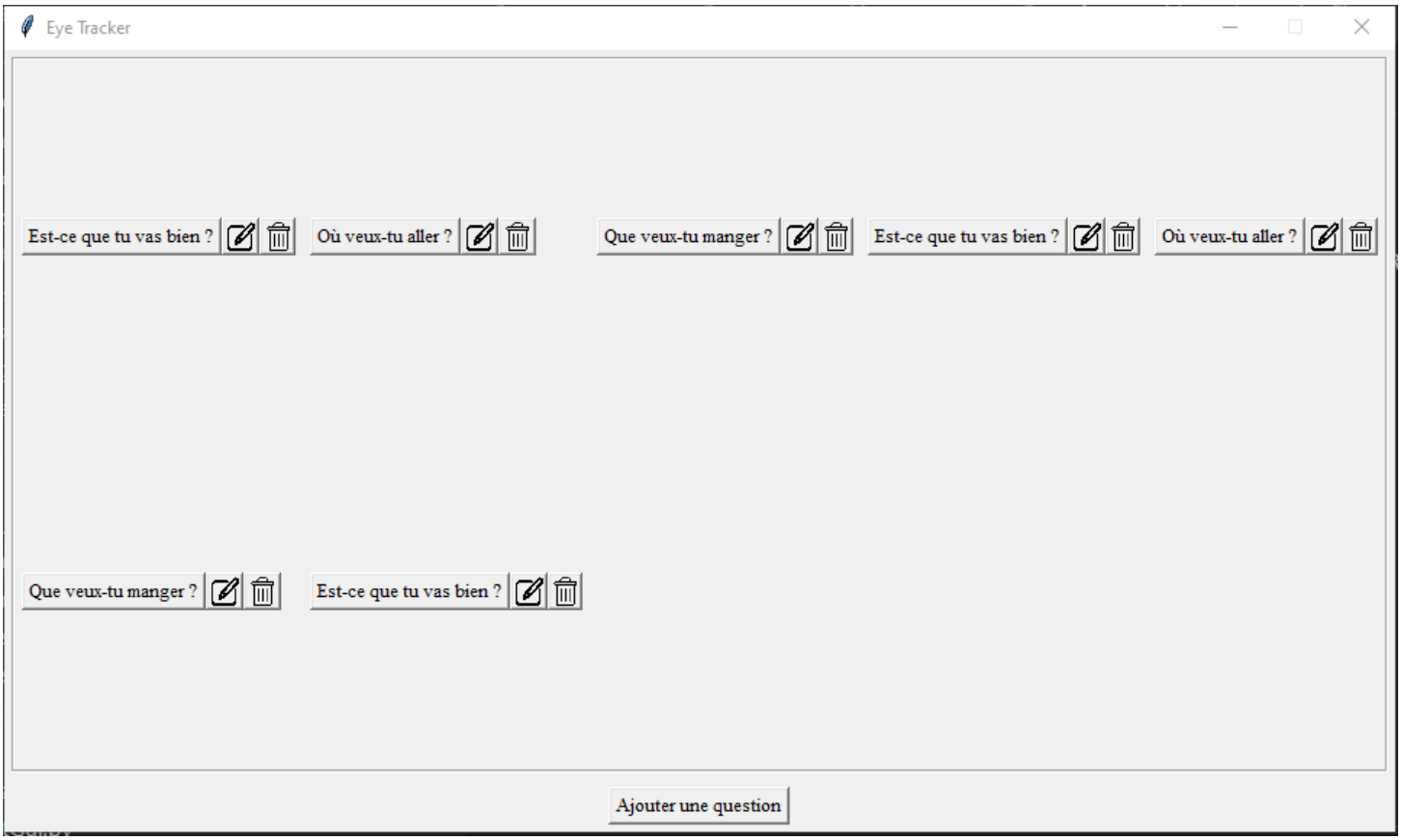

FIGURE 4 – 1^{er} Interface liste des propositions

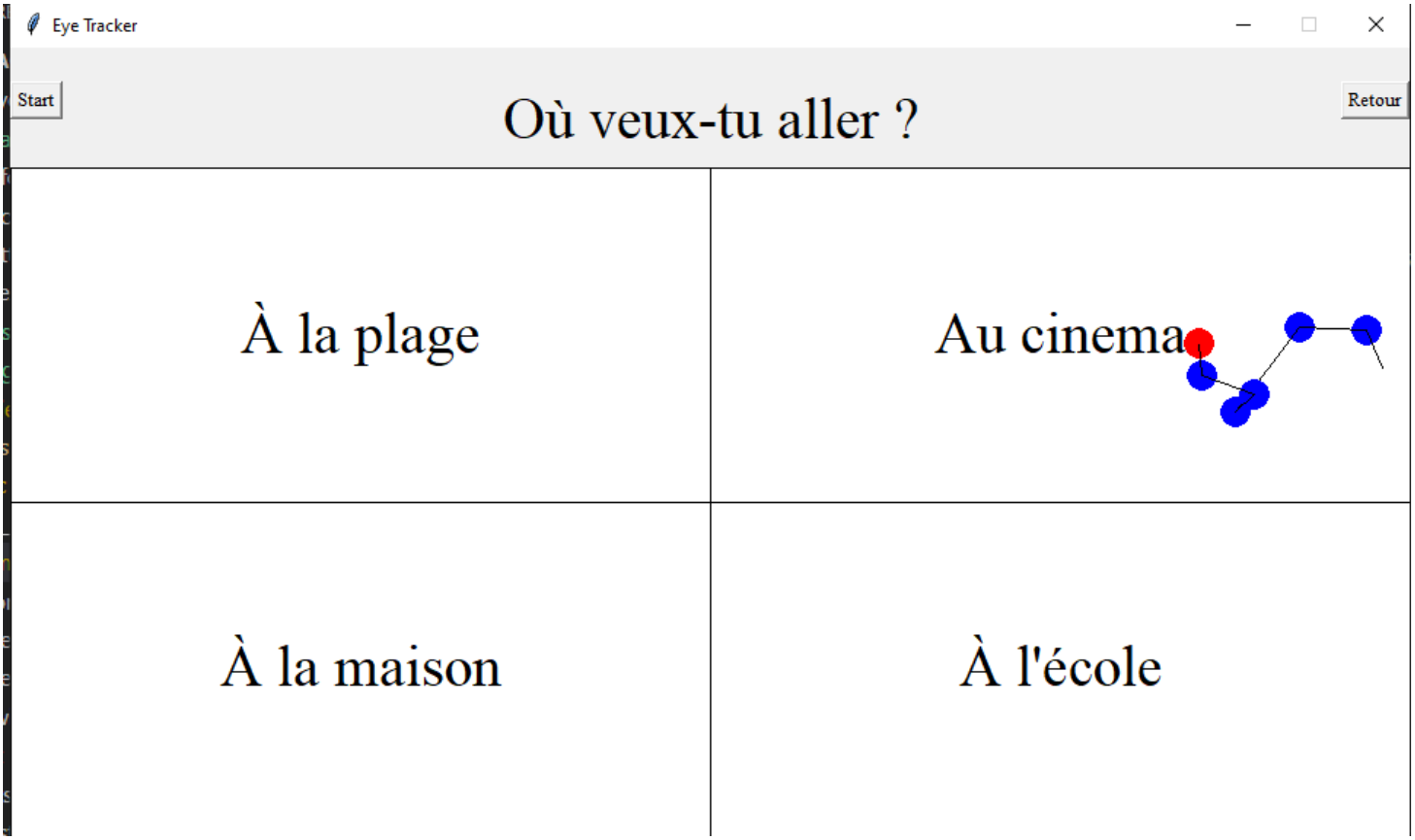

FIGURE 5 – 1^{er} Interface exemple d'eye-tracker

J'ai d'abord fait des croquis sur un logiciel de dessin, que j'ai validé avec mes tuteur, avant de les implémenter.
Cette interface est simple et fonctionnelle. Mais j'ai décidé d'en refaire une autre plus ergonomique.

La 2^{eme} version est basée sur les librairies PyQt et PySide [7]. J'utilise le logiciel Qt Design Studio pour génerer des pages en *QML, qui sont ensuite appelé en *Python.

Pour télécharger les librairies PySide2 et PyQt5 avec pip, il suffit d'ouvrir une fenêtre de commande (ou un terminal) et de taper les commandes suivantes :

```
% Telechargement de PySide2
pip install PySide2

% Telechargement de PyQt5
pip install PyQt5
```

Pour la deuxième interface, j'ai utilisé des pages en QML que j'ai reliées au backend Python. J'ai également utilisé l'application Qt Design Studio pour m'aider à concevoir l'interface graphique grâce à ses outils visuels.

Pour installer Qt Design Studio, rendez-vous sur le site officiel de Qt :
https://www.qt.io/download. Dans l'installeur, cochez la case "Qt Design Studio".

Au niveau de l'architecture, le dossier "`asset/qml`" contient le fichier principal "`main.qml`" qui définit la structure de l'application. Les différentes pages sont organisées dans le dossier "`pages`" sous forme de cadres distincts qui peuvent être navigués grâce à un StackView. Enfin, les objets personnalisés tels que les boutons et les tableaux sont stockés dans le dossier "`components`".

Le StackView est un élément de QML qui permet de gérer la navigation entre les différentes pages de l'interface utilisateur. Il empile les pages sous forme de pile, de sorte que seule la page en haut de la pile est visible à tout moment. En naviguant vers une nouvelle page, le StackView ajoute la nouvelle page en haut de la pile et la page précédente est cachée. La navigation entre les pages peut être gérée en utilisant les méthodes fournies par le StackView, telles que `push()`, `pop()`, `replace()` ou `clear()`.

La classe MainWindow hérite de la classe QObject de PySide2, qui fournit un mécanisme de signaux et de slots pour la communication entre Python et QML. Les signaux et les slots sont utilisés pour déclencher des événements dans l'interface utilisateur QML et pour traiter ces événements dans le code Python.

Le fichier QML (main.qml) contient la partie frontale de l'interface utilisateur. Les objets QML sont créés à l'aide de balises QML, qui sont ensuite liées à des propriétés et des méthodes Python à l'aide de l'objet rootContext() de PySide2.

L'objet backend de la classe MainWindow est passé à l'interface utilisateur QML en tant que contexte de propriété, ce qui permet aux éléments QML d'interagir avec

les propriétés et les méthodes de la classe Python.

Les propriétés et les slots sont deux mécanismes clés de la communication entre Python et QML. Les propriétés sont des variables qui peuvent être utilisées dans le code QML, tandis que les slots sont des fonctions qui peuvent être appelées à partir du code QML.

La classe MainWindow possède plusieurs propriétés et slots définis. Les propriétés sont définies à l'aide des décorateurs "@Property". Par exemple, la propriété "model" est définie comme une liste de QVariant en utilisant le décorateur "@Property("QVariantList", notify=ModelChanged)". Cette propriété peut être utilisée dans le code QML pour afficher une liste d'éléments. La propriété "currentItem" est définie comme une instance de la classe PropositionModel, qui représente un élément de la liste.

Les slots sont définis à l'aide du décorateur "@Slot". Par exemple, le slot "AddProp" permet d'ajouter un nouvel élément à la liste. Les slots peuvent être appelés à partir du code QML en utilisant l'objet "backend" qui représente une instance de la classe MainWindow.

Il est recommandé de créer une classe dérivée de QVariantList pour transmettre des listes de données complexes entre Python et QML.

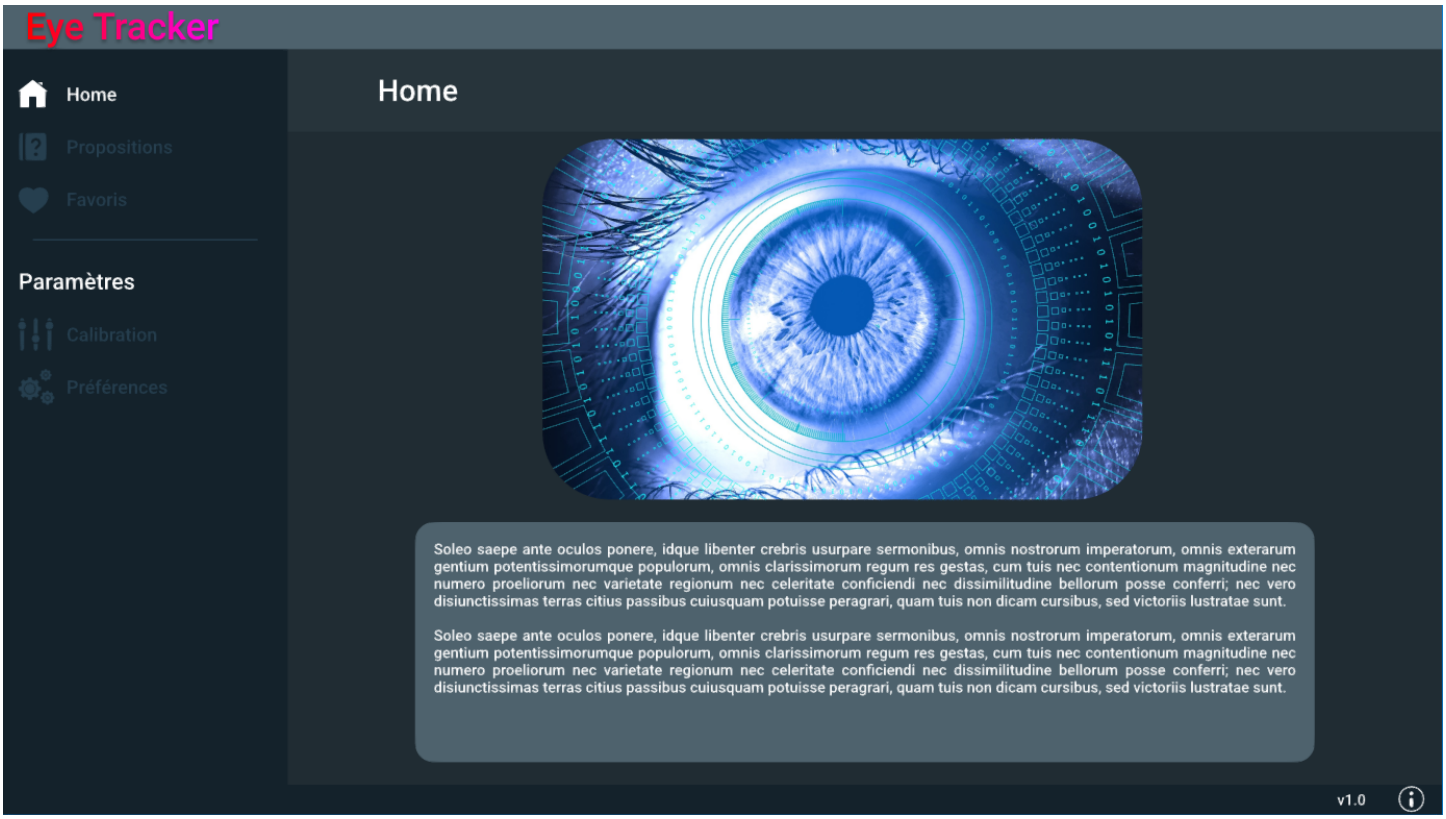

FIGURE 6 – 2^{eme} Interface page Home

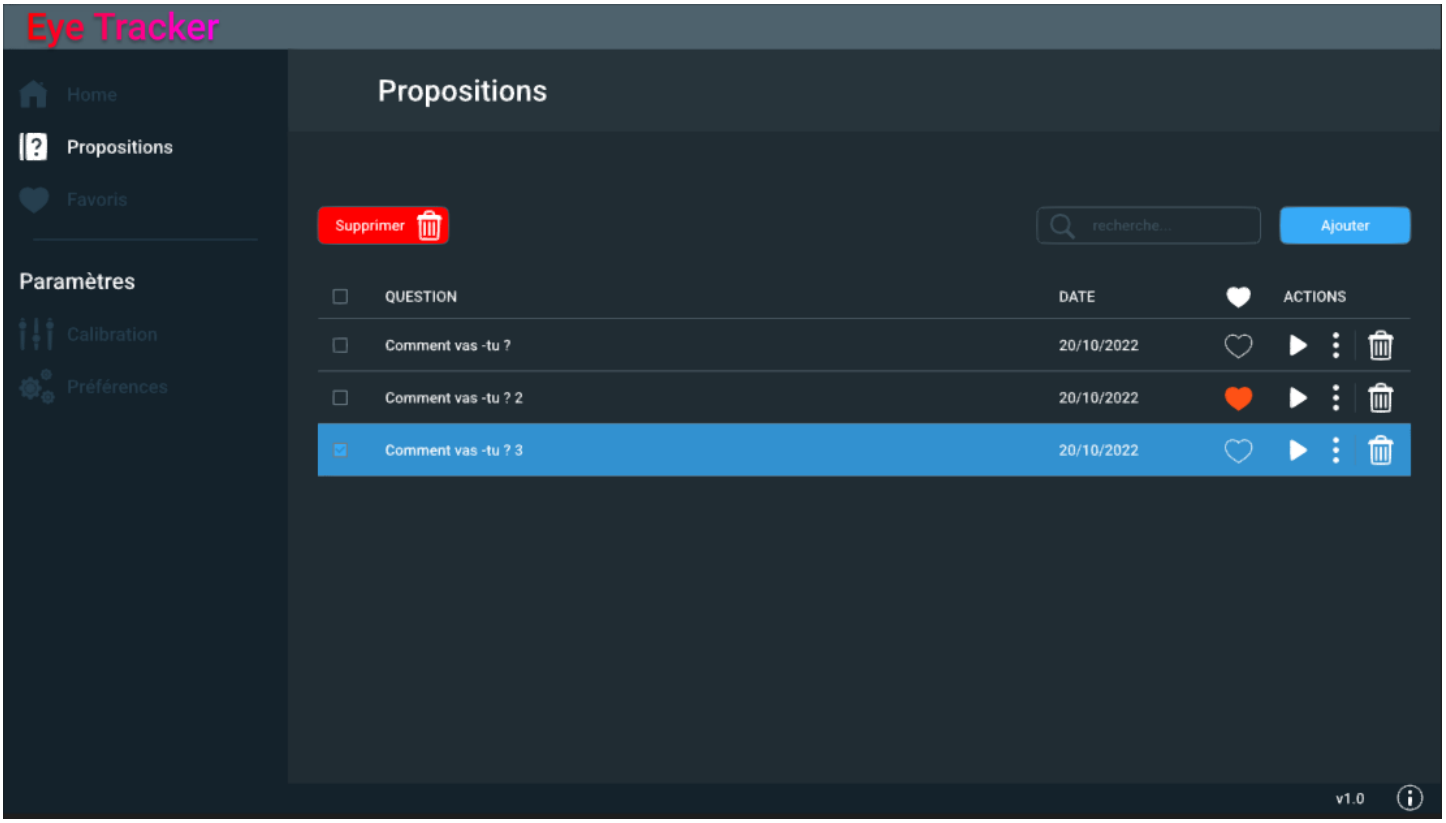

FIGURE 7 – 2^{eme} Interface page Propositions

J'ai amélioré l'interface en ajoutant une page d'accueil conviviale, une section "Propositions" intuitive permettant de soumettre, modifier et supprimer des propositions.
Nous avons également ajouté la possibilité de mettre une proposition en favoris et de stocker la date de la dernière mise à jour pour une expérience utilisateur améliorée.

7.2 Gestion de la base de données

Les données sous forme de questions / réponses sont stockées en format csv. On utilise la librairie csv pour lire et écrire dans ce fichier.

La classe `Loader_CSV` [C] permet de lire un fichier CSV contenant des questions et leurs réponses, et de les stocker dans une liste d'objets `Proposition`.

La méthode `__init__` ouvre le fichier CSV spécifié, le lit avec le module `csv`, et crée une instance de `Proposition` pour chaque ligne (à l'exception de la première ligne qui contient les noms de colonnes). Les instances de `Proposition` sont stockées dans une liste `Propositions`, et le nombre de questions est enregistré dans `nbr_questions`.

Les autres méthodes de la classe permettent d'ajouter, supprimer ou modifier des propositions dans la liste `Propositions`, ainsi que de sauvegarder les propositions dans le fichier CSV.

En bref, cette classe fournit une façon pratique de charger, manipuler et sauvegarder des données de questions-réponses dans un format facilement lisible et modifiable.

Les fonctions de lecture, de modifications, d'écriture, et de suppression de propositions ont été implementées et testées automatiquement avec *Jenkins(tests unitaire, tests fonctionnels, tests d'intégration).

7.3 Création d'un exécutable

7.3.1 Étape 1 : Utilisation de auto-py-to-exe pour transformer le script en exécutable

La première étape consiste à utiliser un outil appelé *auto-py-to-exe* pour transformer le script Python en exécutable. Cet outil est disponible gratuitement sur la plateforme Python et peut être utilisé pour transformer des scripts Python en exécutables sous Windows, Mac et Linux. Voici les étapes à suivre :

1. Ouvrir une fenêtre de commande et installer *auto-py-to-exe* en utilisant la commande.

```
pip install auto-py-to-exe
```

2. Une fois que *auto-py-to-exe* est installé, lancer l'outil en utilisant la commande.

```
auto-py-to-exe
```

3. Sélectionner le script Python que vous souhaitez transformer en exécutable en utilisant le bouton "Browse".
4. Choisir les options de compilation et de packaging souhaitées, telles que l'ajout de fichiers supplémentaires ou l'optimisation de l'exécutable pour la taille ou les performances.
5. Cliquer sur le bouton "CONVERT" pour lancer le processus de transformation en exécutable.
6. Une fois que la conversion est terminée, un fichier exécutable sera généré dans le même dossier que votre script Python.

7.3.2 Étape 2 : Utilisation d'Inno Setup pour créer un installateur

La deuxième étape consiste à utiliser un outil appelé *Inno Setup* pour créer un installateur pour votre exécutable. *Inno Setup* est un outil gratuit et open-source qui permet de créer des installateurs pour Windows. Voici les étapes à suivre :

1. Télécharger et installer *Inno Setup* à partir du site web officiel.
   https://jrsoftware.org/isinfo.php
2. Lancer *Inno Setup* et créer un nouveau projet.
3. Ajouter les fichiers que vous souhaitez inclure dans l'installation, y compris le fichier exécutable généré par *auto-py-to-exe*.
4. Ajouter des informations sur votre application, telles que le nom, la version et les auteurs.
5. Personnaliser les options d'installation, telles que le répertoire d'installation et les raccourcis du menu Démarrer.
6. Compiler l'installateur en utilisant la fonction "Build" d'*Inno Setup*.
7. Une fois que la compilation est terminée, vous obtenez un installateur que vous pouvez distribuer aux utilisateurs.

Lien tuto vidéo : https://www.youtube.com/watch?v=ormsdIk_Uhw

8 Guide d'installation

8.1 Étape 1 : Brancher la Tobii Eye Tracker

Connectez la Tobii Eye Tracker 4C ou la Tobii Eye Tracker 5 à un port USB sur votre ordinateur.

8.2 Étape 2 : Télécharger du logiciel de la Tobii

Si vous avez la Tobii 4C il faudra installer le Tobii Eye Tracking Core Software, si c'est une Tobii 5 il faut le logiciel Tobii Experience !

Attention normalement pour la Tobii 5 le logiciel s'intalle automatiquement lors du branchement de la Tobii, mais ça ne marche pas toujours.

Accédez au site web de Tobii : https ://gaming.tobii.com/getstarted/ ?bundle=tobii-core.
Dans "Select the hardware", sélectionnez "Tobii EyeTracking".
Dans "Select Tobii device", sélectionnez "Tobii Eye Tracker 4C" ou "Tobii Eye Tracker 5.
Dans "Select application", sélectionnez "Tobii Eye Tracking Core Software" ou "Tobii Experience Driver".
Cliquez sur "Télécharger" pour commencer le téléchargement du logiciel.

Suivez les instructions sur l'écran pour installer le logiciel sur votre ordinateur.
Remarque : Assurez-vous d'avoir les autorisations nécessaires sur votre ordinateur pour installer le logiciel.

8.3 Étape 3 : Vérifier le bon fonctionnement de la Tobii Eye Tracker

Une fois que logiciel est installé, démarrez-le :

Pour le "Tobii Eye Tracking Core Software" :

Céer un profil puis effectuez le calibrage de la Tobii Eye Tracker en suivant les instructions.

Puis vérifier que l'application est l'application est sur "Oui".

Ensuite, dans l'application, cliquez sur "Gaze Tracker" pour vérifier que le suivi oculaire fonctionne correctement. Vous devriez voir un pointeur suivre le mouvement de vos yeux autour de l'écran.

Ensuite, cliquez à nouveau pour le désactiver.

Pour le "Tobii Experience Driver" :

Céer un profil puis dans les parmètres effectuez le calibrage de la Tobii Eye Tracker en suivant les instructions.

Puis vérifier que le button "Tobii Experience" est sur “Oui”.

Ensuite, dans l'application, cliquez sur "Apercu du regard " pour vérifier que le suivi oculaire fonctionne correctement. Vous devriez voir un pointeur suivre le mouvement de vos yeux autour de l'écran.

Ensuite, cliquez à nouveau pour le désactiver.

8.4 Étape 4 : Téléchargement de Tobii Ghost

Tobii Ghost est un logiciel de suivi de regard, il permet aux utilisateurs de visualiser leur propre regard en temps réel et d'analyser leur comportement visuel lors de l'utilisation de leur ordinateur.

Rendez-vous sur https ://gaming.tobii.com/getstarted/ et téléchargez Tobii Ghost.

8.5 Étape 5 : Configuration de Tobii Ghost

Ouvrez Tobii Ghost et cliquez sur "Settings" pour accéder aux options de configuration. Sélectionnez "ON" pour "Embed Ghost into content", "ON" pour "Preview", et "Bubble" pour la forme.

Puis, fermez la fenêtre.

8.6 Étape 6 : Installation de l'application EyeTracker

Dans le dossier "build" télécharger "EyeTrackerInstallation", suivez les instructions sur l'écran pour installer le logiciel sur votre ordinateur. Le logiciel se lance automatiquement à la fin.

9 Ce qui reste a faire

Il serait envisageable d'intégrer le système de pointeur du regard directement dans l'interface utilisateur plutôt que d'utiliser le logiciel Tobii Ghost. En outre, il serait pratique d'avoir l'option de calibrer la Tobii directement dans l'interface en cliquant sur l'onglet "calibration".

Par ailleurs, il faudrait implémenter la barre de recherche dans la page "propositions" et de compléter une page dédiée aux favoris.

10 Conclusion

Pour conclure, ce projet est représentatif d'un problème complexe tels ceux qu'on pourrait rencontrer en entreprise et permet de mettre en perspective les notions vues durant ma formation à l'*Enssat. Cela m'a donné un aperçu du travail d'ingénieur, surtout au niveau de l'acquisition de nouvelles compétences.

Je pense qu'il serai intéréssant d'être au moins 2 sur ce projet.

11 Bibliographies

Références

[1] API Tobii Python https://developer.tobiipro.com/python/python-sdk-reference-guide.html (2022).

[2] Licence Tobii https://developer.tobii.com/wp-content/uploads/2021/01/Tobii-Tech-Getting-Started-SDLA-29-Sept-2020FINAL.pdf (29 Septembre 2020).

[3] Tobii Developer Zone Unity https://developer.tobii.com/pc-gaming/unity-sdk/api-overview/ (2022).

[4] Tobii eye-trackerhttps://www.tobii.com/products/eye-trackers (2022).

[5] What's the difference between Tobii Eye Tracker 4C and 5 ? https://help.tobii.com/hc/en-us/articles/360008539058-What-s-the-difference-between-Tobii-Eye-Tracker-4C-and-5- (1 decembre 2022).

[6] Documentation Tkinter https://docs.python.org/fr/3/library/tk.html (7 decembre 2022).

[7] Documentation Qt https://doc.qt.io/ (2022).

[8] Documentation PyGaze http://www.pygaze.org/documentation/ (2014).

A Annexe : Dockerfile.1

```
FROM python:3.8

RUN pip install -U pip setuptools
# Install Tobii
RUN pip install tobii_research

# Download Package Information
RUN apt-get update -y
# Install Tkinter
RUN apt-get install tk -y

WORKDIR /usr/app/src

COPY  projecteyetracker ./projecteyetracker

CMD python ./projecteyetracker/src/main.py
```

B Annexe : Dockerfile.2

```
FROM python:3.8

RUN pip install -U pip setuptools
# Download Package Information
RUN apt-get update && apt-get install ffmpeg libsm6
   libxext6  -y

RUN pip uninstall opencv-python
RUN pip install opencv-python-headless
# Install  et Pyside
RUN pip install PyQt5==5.15
RUN pip install PySide2

WORKDIR /usr/app/src

COPY  ProjectEyeTracker2 ./ProjectEyeTracker2

CMD python ./ProjectEyeTracker2/src/main.py
```

C Annexe : Loader_CSV

```
from proposition import Proposition
import csv

# Class pour lire les csv


class Loader_CSV:
    def __init__(self):
        self.emplacement_fichier_csv = "
           ProjectEyeTracker2/asset/data/
           question_reponses.csv"
        self.Propositions = []
        self.nbr_questions = 0
        with open(self.emplacement_fichier_csv, 'r',
           encoding='utf-8') as f:
            reader = csv.reader(f)
            for row in reader:
                if self.nbr_questions != 0:  # je ne
                   prends pas la premiere ligne du csv
                    self.Propositions.append(Proposition(
                       row))
                self.nbr_questions += 1

    def Get_Proposition(self, i) -> Proposition:
        # 0<= i <= len(self.Proposition)
        return self.Propositions[i]

    def Get_Question(self) -> list:
        questions = []
        for prop in self.Propositions:
            questions.append(prop.question)
        return questions

    def Add_Proposition(self,proposition):
        self.Propositions.insert(0,proposition)

    def Remove_proposition(self,index):
        self.Propositions.pop(index)

    def Modify_proposition(self,index,proposition):
        self.Propositions[index] = proposition

    def Save(self, model):
        with open(self.emplacement_fichier_csv, 'w',
           newline='',encoding='utf-8') as csvfile:
```

```
            fieldnames = ['Favoris','Date','Format','
               Question','Reponse1','Reponse2','Reponse3'
               ,'Reponse4']
            writer = csv.writer(csvfile)
            writer.writerow(fieldnames)
            for prop  in model:
                row =[str(prop.favoris), prop.date, prop.
                   format, prop.question]+prop.reponses
                writer.writerow(row)
```