IFT 3051 – Projet H2007

Présentation : 4 mai 2007

Gulam Moledina
MOLG19068404
moleding @ iro

Directeurs

Yann-Gaël Guéhéneuc
guehene @ iro
www.yann-gael.gueheneuc.net
Professeur adjoint

Houari Sahraoui
sahraouh @ iro
www.iro.umontreal.ca/~sahraouh
Professeur agrégé

Université de Montréal
Département d'Informatique et de Recherche Opérationnelle
Laboratoire de Génie Logiciel

TABLE DES MATIÈRES

I – CONTEXTE

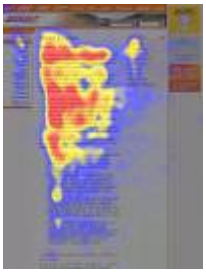
La façon avec laquelle les modèles UML sont conçus et présentés a une influence sur la performance et la complexité des tâches effectuées sur ces modèles. Comprendre ces liens d'influence permet dans un premier temps de choisir les bonnes alternatives de conception et de présentation des modèles en fonction des tâches à effectuer.

Dans un second temps, cette compréhension augmente notre maîtrise des tâches de maintenance et la réduction des coûts de celles-ci. Par ailleurs, de nombreuses innovations telles que les design patterns ont une validité théorique, mais très peu de preuves empiriques ont été produites pour valider leur utilité. Ce projet consiste à mener une série d'expériences en utilisant un environnement de Eye Tracking pour valider/découvrir certaines propriétés d'approches de conception et de présentation de modèles UML.

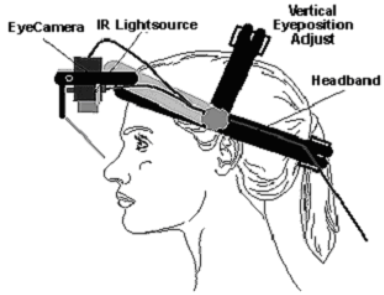
II – INTRODUCTION

Le but de ce projet est de concevoir une expérimentation avec le Eye tracking sur des diagrammes de classes UML afin de pouvoir comparer la facilité d'apprentissage et de compréhension de ces diagrammes utilisant :

- un modèle de patron de conception (layout classique)
- un modèle de patron de conception (layout modifié)
- aucun modèle de patron de conception



Le Eye Tracking (oculométrie) regroupe un ensemble de techniques permettant d'enregistrer les mouvements oculaires et est utilisée comme technique de mesure dans plusieurs domaines tels que la psychologie et la publicité. Les données recueillis permettent par exemple de dresser une carte de chaleur des régions les plus regardés par le sujet. Une étude plus poussée des données permettent de vérifier des hypothèses d'expérimentation scientifique.



Le Eye Tracker est un casque doté de plusieurs caméras qui enregistre les mouvements des yeux pendant tout la phase de l'expérimentation et produit un fichier EDF en sortie qu'il faudra ensuite parser et extraire les variables nécessaires à l'analyse de l'information.

III – TRAVAIL À ACCOMPLIR

Le travail à effectuer comprend les taches suivantes :

- Familiarisation avec l'environnement du Eye Tracking
- Définition des tâches expérimentales
- Rappel du modèle de patron de conception choisit
- Définition des diagrammes et des taches expérimentales
- Extraction (programmation du parseur) des variables jugées utiles qui peuvent être produites par l'environnement
- Supervision du déroulement des tâches expérimentales

La dernière tache a été remplacée durant la session. En effet, n'ayant pas assez de temps pour le déroulement des taches expérimentales, cette tache a été remplacée par :

- Développement du logiciel de gestion des expériences : Taupe

IV – ENVIRONNEMENT DE TRAVAIL ET PRÉALABLE

L'ensemble du travail s'est déroulé sur la plateforme Windows. Mais les logiciels principaux et le langage de programmation utilisé garantissent une grande portabilité. Ce projet a donc requis l'utilisation de plusieurs logiciels :

- Logiciels propre au Eye Tracking (Windows uniquement)
- Plateforme Eclipe pour la programmation Java
- Visio de Microsoft pour la conception des diagrammes UML

De plus, il a fallu apprendre à utiliser le Eye Tracker et le logiciel de suivi.

Je ne continue pas le projet d'un autre étudiant. Je n'avais donc accès à aucun résultat ou projet antérieur dont je devais m'inspirer. Sauf pour le développement du logiciel Taupe (détails plus tard).

Le site web du projet est http://www-etud.iro.umontreal.ca/~moleding/projet/index.htm

V – DÉFINITION DES TÂCHES EXPERIMENTALES

Il s'agit donc en premier temps de concevoir 3 versions différentes de 3 applications utilisant le modèle de patron de conception choisis, soit le patron de conception Visitor.

Soient l'ensemble Ai : les applications choisies :

A1 : JHotDraw
A2 : JRefactory (remplace DPL qui avait été choisi en premier lieu)
A3 : PADL

Soient l'ensemble Ti : les taches à accomplir par les sujets afin de s'assurer de leur compréhension des diagrammes :

T1 : apporter des modifications dans une région du patron du diagramme de classe
T2 : apporter des modifications dans une autre région

Soient l'ensemble Di : les versions des diagrammes de classes des projets choisis :

D1 : aucun modèle de patron de conception
D2 : avec patron de conception, layout classique
D3 : avec patron de conception, layout modifié

Soient l'ensemble Si : groupe de 10 sujets selon leur connaissance :

S1 et S2 : connaissent les patrons de conception
S3 et S4 : ne connaissent pas les patrons de conception

Voici alors le scenario du déroulement de l'expérimentation :

A1				A2				A3			
D1	D2		D3	D1	D2		D3	D1	D2		D3
	T1	T2			T1	T2			T1	T2	
S1	S2	S3	S4	S2	S3	S4	S1	S3	S4	S1	S2

Les hypothèses de l'expérimentation sont :

1-/ Ceux qui connaissent les patrons de conception exécutent une modification plus vite que les autres :

- comparer (S1 et S2) VS (S3 et S4)
- pas d'importance pour les taches choisies : T1 et/ou T2
- sur les diagrammes D2 et D3

2-/ Ceux qui connaissent le patron de conception exécutent une modification plus vite si la tache touche au patron en question :

- comparer les résultats des taches T1 et T2
- sujets S1 et S2 uniquement
- sur les diagrammes D1 et D2

3-/ Ceux qui connaissent les patrons de conception regardent plus les classes du patron que les autres si la tache touche le patron en question, et inversement :

- comparer (S1 et S2) VS (S3 et S4)
- comparer les résultats des taches T1 et T2
- sur les diagrammes D2 et D3

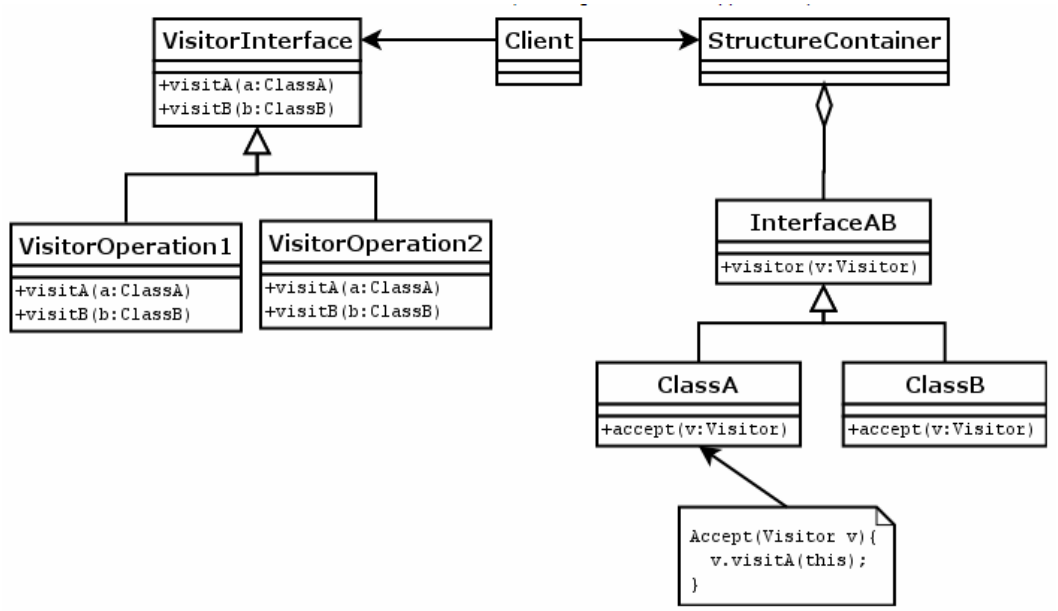
VI – PATRON DE CONCEPTION VISITOR

Source : site Wikipédia et livre Design Patterns de Gamma

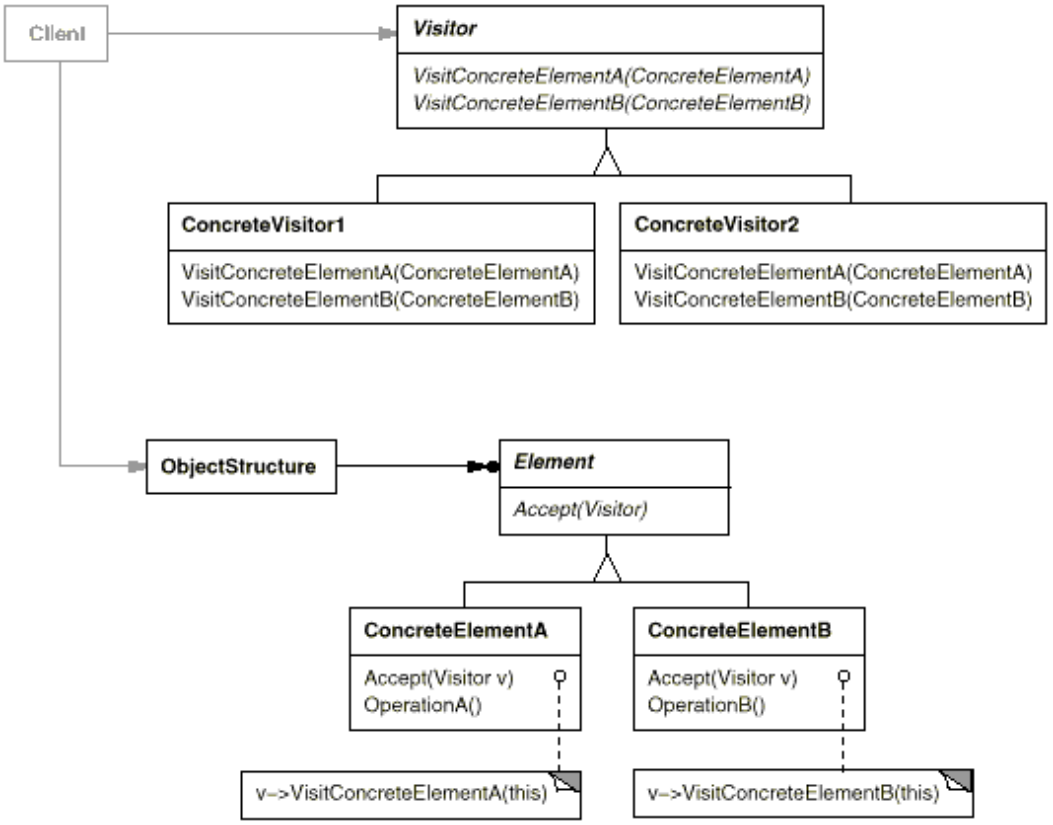
Le motif de conception visiteur est une manière de séparer un algorithme de la structure d'un objet. Ce modèle de conception permet à une classe externe d'accéder aux variables internes d'autres classes.

Si déporter des opérations contenues dans une classe vers une autre peut sembler mauvais au sens POO, il y a de bonnes raisons pour le faire. En effet, si ces opérations sont identiques pour chaque classe au lieu de dupliquer cette méthode il est préférable de mettre ces opérations dans un visiteur (centralisation de l'opération). Le visiteur utilisera ensuite les données internes de chaque objet pour effectuer l'opération demandée.

Fonctionnement technique : chaque classe pouvant être « visitée » doit mettre à disposition une méthode publique « accepter » prenant comme argument un objet du type « visiteur ». La méthode « accept » appelera la méthode « visit » de l'objet du type « visiteur » avec pour argument l'objet visité. De cette manière, un objet visiteur pourra connaître la référence de l'objet visité et appeler ses méthodes publiques pour obtenir les données nécessaires au traitement à effectuer.



Ou le layout habituel (classique) :



VII – LES DIAGRAMMES DE CLASSES

VIII – LES DONNÉES DU EYE TRACKER

Le Eye Tracker enregistre les mouvements des yeux pendant toute la phase de l'expérimentation (les « samples ») et produit ensuite un fichier EDF en sortie.

1-/ Les « samples »

Les 'Samples" sont des informations sur la position des yeux, la grosseur de la pupille et l'état des boutons, plus précisément :

- le temps du 'sample' (timestamp) en milliseconde
- un ou plusieurs type des positions des yeux (ex:
    - pupil position data (x,y) from the eye camera
    - HREF (head-reference) position data (eye rotation angles relative to the head)
    - Gaze position : the actual (x,y) coordinates of the subject's gaze on the display
- la grosseur (size) des pupilles
- résolution angulaire des informations sur la position des yeux
- l'état (state bits) des boutons ou des ports

Les « samples » sont enregistrés par le EyeTracker à une fréquence pouvant atteindre les 500 samples/seconde. On peut bien sur diminuer cette fréquence ou même choisir d'enregistrer un sample uniquement lorsqu'un évènement significatif sur la position des yeux a lieu.

2-/ Mouvement des yeux (après avoir parsé les samples)

Les événements lies au mouvement des yeux sont générés en pair : un évènement au début du mouvement des yeux (selon une/des condition(s) prédéterminés) et un évènement à la fin. Ces évènements sont associés à l'œil qui les a déclenché. Si on est en mode "binocular", un 'debut' et 'fin' d'événement est enregistré pour chacun des yeux séparément.

Le parseur de EyeLink II analyse les informations (l'ensemble des « samples » enregistrés) sur la position des yeux afin d'identifier les saccades et les fixations et génèrent les informations suivantes :

Fixations :
- le début et la fin du 'sample' de la fixation (temps)
- l'œil qui a généré l'événement
- HREF au début, a la fin et en moyenne, ainsi que la position du 'gaze' (regard?)
- la taille du pupille au début, a la fin et en moyenne
- la vitesse du mouvement des yeux au début, a la fin, en moyenne et le maximum
- début et fin de la résolution angulaire du 'gaze-data'

Fixation update :
On peut subdiviser l'événement "Fixation". En effet, on peut déclencher l'évènement 'FIXUPDATE' à un intervalle de temps régulier durant une fixation. Très utile pour le contrôle en temps réel, non nécessaire dans notre cas.

Saccades :
Le parseur détecte les saccades avec la vitesse et l'accélération du mouvement des yeux. L'information sur les saccades est constitue des mêmes champs que pour les fixations, à savoir :
- le début et la fin du 'sample' de la saccade (temps)
- l'œil qui a généré l'événement
- HREF au début, à la fin et en moyenne, ainsi que la position du 'gaze' (regard?)
- la taille du pupille au début, a la fin et en moyenne
- la vitesse du mouvement des yeux au début, a la fin, en moyenne et le maximum
- début et fin de la résolution angulaire du 'gaze-data'

Blinks (clignement des yeux):
Événement lorsque la taille du pupille est très petite ou que la pupille n'est pas visible à la camera.
Seuls les données temporelles sont enregistrés (début-fin)

On peut avoir d'autres événements (non lies directement au mouvement des yeux), à savoir :
- les messages : du texte ou du binaire que l'on peut faire écrire dans le fichier de sortie lors d'un autre événement important de notre choix
- les boutons : possibilité d'avoir au plus 8 boutons

IX – LOGICIEL TAUPE

À partir du fichier EDF fournit par le parseur du Eye Tracker, nous pouvons concevoir des multitudes de logiciel afin d'analyser les données recueillis durant l'expérimentation.

Mon objectif initial était de programmer un parseur afin de générer un fichier csv (« Comma Separated Values ») qui permettrait alors l'analyse des informations sur un tableur (comme Excel de Microsoft par exemple).

Mais finalement, j'ai travaillé dans le développement de Taupe, un logiciel de gestion des expériences, qui permet une multitude de fonctions supplémentaires.

1-/ Le parseur EDF → CSV

Dans le fichier EDF, les informations sur les fixations et les saccades sont enregistrés ainsi :

MOT CLÉ <info1> <info2> <info3> …. <infoN>

Plus précisément :

- pour les fixations :

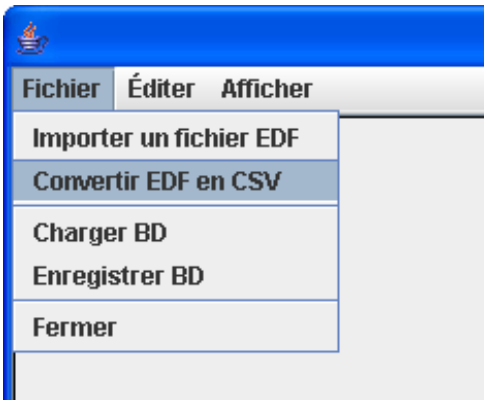
EFIX <eye> <stime> <etime> <dur> <axp> <ayp> <aps>

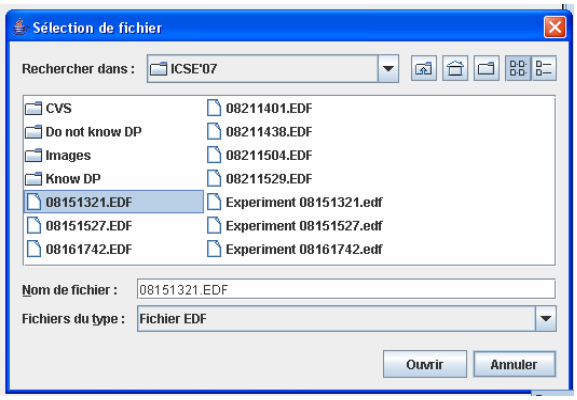
- pour les saccades :

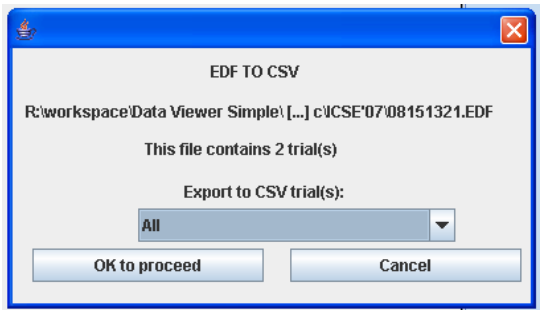
ESACC <eye> <stime> <etime> <dur> <sxp> <syp> <exp> <eyp> <ampl> <pv>

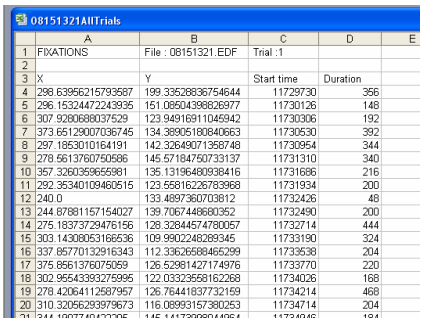
Avec :
<eye> : « L » ou « R » (Left ou Right : l'œil qui est a l'origine de l'événement)
<stime> : Start Time en Timestamp
<etime> : End Time en Timestamp
<dur> : Duration. Attention! Duration = End Time – Start Time + Δ (erreur de precision)
<axp> et <ayp>: Average X and Y of eye position
<sxp> et <syp> : Start X and Y of eye position
<exp> et <eyp> : End of X and Y of eye position
<aps> : Average pupil size
<ampl> : saccadic amplitude (degrees)
<pv> : peak velocity

J'ai donc programmé un parseur (qui est maintenant intégré au logiciel Taupe) qui génère un fichier CSV avec les informations sur les fixations et les saccades à partir du fichier EDF en entrée.









2-/ Le logiciel de gestion des expériences

Il s'agissait d'un très grand défi car je devais continuer le développement d'un logiciel qui avait été commencé par une autre personne et :

- il n'y avait aucun document justificatif ou explicatif de la conception et des conventions
- le code ne contenait aucun commentaire
- aucune garantit du code fournit : je ne pouvais pas faire confiance à 100% à tout le code qui m'avait été transmis car le logiciel était inachevé. Mais finalement le code était assez fiable.

J'ai donc du lire et comprendre la totalité du code et deviner l'intention du programmeur (la programmation est un art, tout le monde le fait à sa manière) afin de continuer la programmation dans la même optique tout en adaptant selon ma compréhension des enjeux et des spécifications supplémentaires de Mr Yann-Gaël Guéhéneuc.

a) Les packages

Les principaux packages :

- *laigle.data.export* : permet d'exporter un fichier edf en csv, utilise le parseur TrialDataHandler
- laigle.data.viewer.data : gère les « area of interest », contient les classes Fixation et Saccade et le parseur TrialDataHandler
- laigle.data.viewer.database : contient les structures de données des expériences
- laigle.data.viewr.ui : contient la classe qui lance le programme, Launch
- laigle.data.viewer.ui.filter : les filtres pour l'ouverture des fichiers
- laigle.data.ui.panel : tous les JPanel de l'interface
- liagle.data.ui.splash : toutes les fenetres d'option qui apparaissent en splash

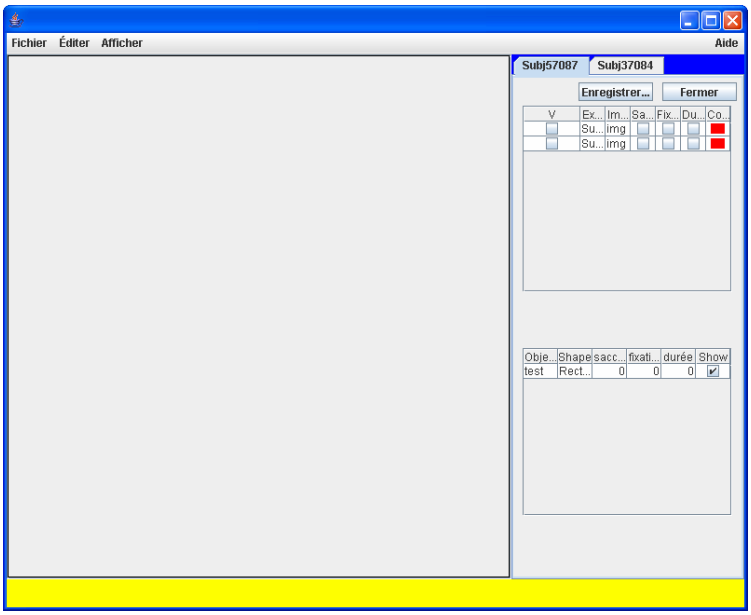
b) Les structures de données

TaupeDataTrial : idTrial, image path, list of subjects
TaupeDataSubject : idSubject, comments, list of experiences
TaupeDataExperience : idExp, idTrial, trial

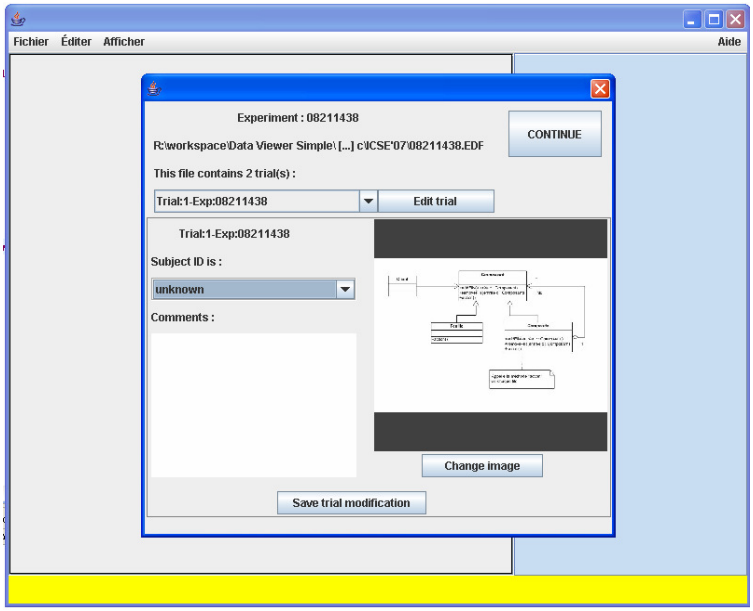
Dans TaupeDataBase :
HashMap<idTrial, TaupeDataTrial>
HashMap<idSubject + image path, TaupeDataExp>
HashMap<idSubject, TaupeDataSubject>

HashMap<idSubject + image path, TaupeDataExp> permet de retrouver facilement toutes les expériences qui on été faites pas un sujet X ou toutes les expériences basées sur la même image.

c) L'interface graphique



Panneau principal



Ajout d'une nouvelle expérience

X – DISCUSSION

1-/ Difficultés rencontrées

Durant tout le long du projet, j'ai du surmonter de nombreux défis (normal). Malgré l'aide de mes deux directeurs, certaines phases du projet étaient assez difficiles :

- conception des diagrammes de classe à partir de gros projet existant

- conception des diagrammes de classe « sans patron de conception »
- compréhension du code de Taupe sans aucune documentation

2- Logiciel Taupe

Il s'agit d'un logiciel qui promet beaucoup! On pourrait ne jamais s'arrêter de développer et d'ajouter des nouvelles fonctionnalités à un tel logiciel.

3- Profits personnels

Ce projet joue un rôle assez important dans ma formation d'informaticien, surtout dans le domaine du génie logiciel. De plus, j'ai eu un exemple pratique de ce que pourrait signifier être programmeur dans une entreprise (importance des plateformes de développement, CVS, …). J'ai également eu l'occasion de revoir pratiquement des notions théoriques vues en cours.

De plus, J'ai eu l'occasion d'utiliser mon expérience d'assistant de recherche au laboratoire de psychologie cognitive. En effet, la conception de l'expérimentation fut facilitée grâce au fait que j'ai déjà eu à encadrer des expériences de psychologies assistées par ordinateur maintes fois.

4- Critiques

La plus grande critique que je peux me faire est le manque d'organisation et le manque de constance dans l'effort fourni durant la session. Mais cela n'est pas du à une négligence mais plutôt au débordement de travail que j'ai eu a subir tout au long de la session.

5- Remerciements

Je tiens sincèrement à remercier mes deux directeurs de projet, Mr Yann-Gaël Guéhéneuc et Mr Houari Sahraoui, pour leur disponibilité et leur support durant tout le long de la session.