

Rapport projet de synthèse NumWeb

Sofiane Hamiti  
Nicolas Reitz

Sommaire

[I. Introduction 3](#_Toc319057501)

[II. L’existant 3](#_Toc319057502)

[III. Accessibilité du web 5](#_Toc319057503)

[IV. Vimperator 6](#_Toc319057504)

[A. Droits 6](#_Toc319057505)

[B. Critique 6](#_Toc319057506)

[V. Notre démarche 8](#_Toc319057507)

[A. Qu’est-on susceptible de faire sur internet ? 8](#_Toc319057508)

[B. Comment résoudre ces différents problèmes ? 9](#_Toc319057509)

[1. La problématique commune à toutes les tâches : le clic 9](#_Toc319057510)

[2. Navigation 9](#_Toc319057511)

[3. Téléchargement 12](#_Toc319057512)

[4. Vidéo et Musique 12](#_Toc319057513)

[5. Copié/Collé 13](#_Toc319057514)

[6. Impression 13](#_Toc319057515)

[C. Technologie pour la mise en œuvre du plug-in 13](#_Toc319057516)

[1. XUL 13](#_Toc319057517)

[2. JavaScript 14](#_Toc319057518)

[3. Ajax 14](#_Toc319057519)

[D. Mise en œuvre 15](#_Toc319057520)

[1. Interface du prototype 15](#_Toc319057521)

[2. Fonctionnement du prototype 22](#_Toc319057522)

[3. Le CGI (Common Gateway Interface) 23](#_Toc319057523)

[4. Relation prototype/CGI 23](#_Toc319057524)

[E. Tests utilisateurs 24](#_Toc319057525)

[1. Protocole des tests 24](#_Toc319057526)

[2. Résultats des tests 26](#_Toc319057527)

[3. Post questionnaire 26](#_Toc319057528)

[4. Conclusion 29](#_Toc319057529)

[F. Conclusion 29](#_Toc319057530)

[Annexe A](#_Toc319057531)

[Ajax : demande de la manipulation souris A](#_Toc319057532)

[CGI utilisé B](#_Toc319057533)

# Introduction

Les utilisateurs handicapés physiques peuvent se heurter à des difficultés pour accéder au contenu du Web. Ces problèmes peuvent toutefois se résoudre grâce à des aides techniques afin de faciliter l'interaction et l'accès au contenu de la page et de permettre aussi de profiter au maximum des outils offerts par le navigateur utilisé.

Notre volonté est de développer une de ces aides techniques afin que les personnes ayant un handicap physique puissent accéder au contenu du Web de façon autonome. Le but est d’associer un clavier à balayage et un capteur de type tout ou rien afin de manipuler la page web.

Malgré la volonté de rendre cet outil accessible au plus grand nombre, à notre stade du développement, le clic reste indispensable. Le prototype reste donc accessible uniquement aux personnes capables de le créer.

Le développement utilise plusieurs technologies. Le développement se base sur le navigateur web Firefox, avec lequel il est possible de connecter des plug-ins codés en XUL. XUL est un langage d’interface graphique dont le dynamisme est apporté via du JavaScript. De plus nous avons utilisé la combinaison d’Ajax et de script CGI afin de réaliser des actions spécifiques telles que la simulation de clic.

# L’existant

Pour des personnes atteintes d’un déficit moteur sévère conjugué à des troubles de la parole, une solution couramment adoptée pour les aider à communiquer avec leur entourage consiste à réaliser un défilement lumineux automatique sur un écran d'ordinateur ou sur un écran autonome. Ce défilement contrôlé par un capteur de type tout ou rien adapté à la personne permet de sélectionner un item (icône, mot, lettre...) parmi la matrice d'items proposée en fonction de l'action (lire, écrire, etc.).

Le principal défaut d'une communication à balayage est sa lenteur d'action : le délai de défilement élémentaire entre deux items peut varier de 200 ms à plusieurs secondes selon les possibilités physiques et cognitives de la personne.

La majeure partie des recherches dans ce domaine traite du problème de la saisie de texte avec l'usage d'un clavier virtuel optimisé. Dans le cas d'un langage de communication alphabétique, de nombreuses études ont cherché à améliorer cet état de fait en diminuant le nombre moyen de sélections nécessaires pour composer une phrase. On peut également chercher à diminuer le temps d'accès à un item en les reconfigurant dynamiquement après chaque validation de façon à présenter les items les plus probables en début de défilement. D'autres travaillent sur le temps d'action que met la personne à valider un item en vue d'adapter automatique le temps de défilement.

Avec les méthodes à balayage, se pose également le problème du nombre d’items sélectionnables qui soient rapidement accessibles. En effet dans le cas de l'utilisation d'un logiciel de navigation sur le Web, on peut se trouver face à un grand nombre de liens ou l'accès par défilement est très lent, incommode, voire même impossible, si le site web ne respecte pas les normes d'accessibilité de pages web.

On trouve des solutions logiciels qui utilisent un clavier virtuel dédié au contrôle du navigateur (clic souris, touche de tabulation, flèches de direction...). Ces solutions émulent généralement les déplacements du curseur de la souris en permettant de contrôler son déplacement afin de faire "comme si l'utilisateur contrôlait sa souris".

On trouve sur le marché des solutions pour piloter les déplacements de la souris via l'usage de caméra (IR, webcam) qui permettent de contrôler le curseur par un suivi de regard ou mouvement de tête. Sans parler des réglages techniques qui peuvent être fastidieux dans certains cas (éclairage de la pièce, sensibilité des déplacements), l'inconvénient majeur à notre sens est que le fait de contrôler les déplacements du curseur ne suffit pas il faut aussi comme sur une souris avoir la possibilité de cliquer. Généralement avec ce type de technologie le clic se fait par une pause à l'endroit où se situe le curseur ce qui demande de la précision et une certaine concentration à l'usage. De plus le curseur suit le regard ou bien la tête de l'utilisateur et de ce fait il n'y a pas vraiment de repos pour la lecture puisque l'arrêt du curseur est compris comme la volonté de cliquer ce qui peut poser problème si l'arrêt se fait sur un item/liens sélectionnable.

L'analyse de l'usage d'un utilisateur face à son écran peut se diviser grossièrement en deux grandes phases : le temps de lecture et le temps d'action sachant que pour certaines personnes le déplacement du curseur est une aide à la lecture et non la volonté de cliquer où se situe le curseur.

C'est dans cette idée de mixer l'usage d'un clavier virtuel contrôlable par accès direct ou bien par défilement, de respecter les phases de lecture et d'action, de permettre d'avoir un grand nombre d'items sélectionnables rapidement que se situe le travail présenté dans ce papier.

On se propose d'introduire le concept de marquage/numérotation à la volée des informations à l'écran qui ouvre la voie à un accès direct avec un clavier virtuel. En effet le fait de numéroter chaque item donne à l'utilisateur la possibilité d'interagir rapidement via son capteur de commande, et ce, de manière localisée quel que soit son type proportionnel ou tout ou rien. Il peut être plus rapide d'écrire "67" sur un clavier numérique que de sélectionner l'item 67 dans une liste, ceci dépendant bien entendu de la technique de balayage utilisée.

Notre objectif est de fournir une méthode générique d'interaction utilisable dans le cadre des interfaces d'aide à la communication basée sur une interaction via un clavier numérique qui permet de contrôler un nombre important d'items en limitant les déplacements du curseur de contrôle de l'interface ou bien les défilements dans le cadre des interfaces à balayage. Le prototype d’une application dédiée à l'utilisation du navigateur Firefox est présenté.

# Accessibilité du web

Actuellement, la majorité des sites internet est accessible uniquement au moyen de navigateurs graphiques traditionnels. Selon l’importance du handicap physique, moteur ou neurophysiologique, leur accès est difficile, voire impossible.

Pour répondre à cela, la WAI (Web Accessibility Initiative) du W3C (Web Consortium) propose un ensemble de directives destiné aux créateurs de sites Internet, afin de rendre leurs sites plus accessibles aux personnes handicapées. Ces directives s'appliquent au niveau du code HTML et demandent une réflexion sur la structure et l'organisation des données du site.

Cependant, l'essor des services et applications en ligne reposant sur l'utilisation croissante de technologies hybrides tels que JavaScript ou AJAX complique les choses. La récente Accessible Rich Internet Applications Suite (ARIA) vise à mettre en place le cadre normatif nécessaire à l'accessibilité des applications Web dynamiques.

Dans le cas où le handicap moteur entraîne une imprécision dans les gestes et donc une difficulté à utiliser pleinement certains périphériques externes comme le clavier ou la souris, l’usage de la touche tabulation (ou équivalent) permet de sélectionner séquentiellement les éléments cliquables du document. Ceci est toutefois inopérant si le site ne respecte pas les directives du WAI avec en particulier l’utilisation du JavaScript.

D’autres stratégies existent pour soulager la tâche motrice de l’utilisateur par exemple le clic automatique par survol de la souris. L’utilisateur déplace le curseur souris et le clic est obtenu à la position du curseur après un certain délai sans bouger.

Dans le cas d’un handicap moteur plus important, il est possible d’éviter une bonne partie des tâches motrices avec l’usage de logiciels de reconnaissance vocale. Mais il est plus courant de se porter sur des solutions à base de défilement. Le défilement automatique des liens de la page affichée est le plus courant. CompuWeb-Access propose également un défilement automatique du texte de la page et eSSENTIAL Accessibility™ propose de nombreuses variantes d’accès aux éléments de la page : Clic automatique, Recherche manuelle / Recherche automatique, Souris XY, Souris de direction, Souris radar et Détection de mouvements mains libres à base de caméra web.

Alors que CompuWeb-Access et HandiLog demandent une intégration de leur technologie directement dans le site, eSSENTIAL Accessibility™ s’appuie sur un navigateur propriétaire. Cette solution apparait discutable dans la mesure où ce navigateur reste perfectible et éloigné des standards du marché. De plus, malgré les nombreuses techniques de sélection proposées, l’accès aux éléments de la page reste long et parfois difficile, voire impossible.

De plus le travail proposé pourra être facilement adapté à tous les outils de la fondation Mozilla à l’initiative de Firefox. Pour faciliter l’accès aux éléments de la page, nous avons basé notre travail sur l’extension Vimperator décrite dans la partie suivante.

# Vimperator

## Droits

Vimperator est sous licence MIT, cette licence autorise l'utilisation, la copie, la modification, la fusion, la publication, la distribution, la vente et le changer sa licence. La seule obligation imposée par cette licence est de mettre le nom des auteurs avec la notice de copyright. Cette licence nous permet donc d’utiliser Vimperator avec une grande liberté.

## Critique

Dans la recherche de l’existant, nous avons trouvé deux papiers dont l’aboutissement est fortement ressemblant à Vimperator. Ces papiers exposent notamment un algorithme de fonctionnement général et le second sur la problématique d’optimisation de labélisation des items.

Le but de Vimperator est de pouvoir contrôler Firefox sans utiliser d’autres périphériques que le clavier. Le fonctionnement de Vimperator au niveau de manipulation s’inspire du célèbre éditeur de texte Vim. Vimperator est notamment à l’origine de Pentadactyl, outil plus orienté pour développeur.

Vimperator permet notamment la navigation dans les onglets, dans la page web, la recherche de texte, navigation dans les liens hypertextes. La navigation par lien hypertexte fonctionne via l’association d’un label à un lien.

L’imprime écran ci-dessous montre le site de l’APF (www.apf.fr)



Ci-dessous le même avec la numérotation des liens fait par Vimperator. L’utilisateur a activé cette fonctionnalité à l’aide de la touche « f ». L’accès au lien est alors possible en saisissant le numéro. Un système de filtre permet de réduire le nombre de labels en fonction de ce qui est déjà saisi. Aussi bien lors de la navigation par numéro que par recherche alphanumérique.



Toutefois, certaines limites existent. La numérotation est effectuée par rapport à la page affichée et commence à 1. En cas de défilement, un élément de la page changera donc de numéro. Même si Vimperator fonctionne très bien dans la majorité des cas, certaines difficultés peuvent apparaître avec du code JavaScript ou certaines feuilles de style. Par exemple, sur le site www.apf.fr, il est impossible d’accéder aux menus déroulants : « S’INFORMER », « NOUS AIDER » et « S’IMPLIQUER ». Ils sont numérotés, mais ils nécessitent le pointage avec la souris pour déclencher le script JavaScript. De même, les éléments (à gauche de PACTE APF) sont correctement numérotés de 18 à 22 et donc accessibles directement, mais les numéros n’apparaissent pas correctement à l’écran. Ces défauts rendent donc l’usage de la souris encore nécessaire.

# Notre démarche

Deux de nos objectifs sont de minimiser les modifications sur le plug-in Vimperator, et d’utiliser un minimum le plug-in Vimperator. C’est pourquoi nous avons développé notre propre plug-in avec ses propres fonctionnalités, mais aussi une reprise de fonctionnalités de Vimperator (notamment pour la numérotation des liens).

## Qu’est-on susceptible de faire sur internet ?

Afin de savoir quelles actions est susceptible de faire une personne sur internet, nous avons procédé par association d’idées en nous basant sur les trois sens les plus usités dans l’interaction avec l’ordinateur (Toucher, Ouïe, Vue).

## Comment résoudre ces différents problèmes ?

### La problématique commune à toutes les tâches : le clic

La difficulté même d’une personne handicapée physique est de pouvoir cliquer à un endroit précis. En effet cette modalité permet toutes les actions relatives à l’utilisation d’un ordinateur. Pour pouvoir pallier à ce problème il faut que l’utilisateur soit capable de fournir une entrée binaire, couplée à un système de balayage l’utilisateur est capable d’exécuter une action précise.

Reste à définir comment. L’utilisation d’une souris classique peut se révéler être une bonne approche dans un premier temps. Néanmoins la détection d’un clic peut entrer en conflit avec le système, en effet que ce passe-t-il si l’utilisateur déplace la souris au moment du clic : la perte du focus et/ou une interaction non souhaitée (fermeture de fenêtre, déclenchement d’un lien hypertexte, etc.).

Nous avons donc opté pour l’utilisation d’un Joystick. L'avantage de l’utilisation de ce type de périphérique est de pouvoir exploiter facilement une seule entrée qui plus est binaire.



### Navigation

Nous avons distingué trois moyens de naviguer via :

* des liens préenregistrés : les favoris
* Vimperator, pour naviguer de page en page
* des balayages-écrans et simulations de cliques pour activer les liens hypertextes (nous rentrerons dans les détails dans la partie V.D.3

#### Les liens

Vimperator gère les liens via l’utilisation de la touche « f ». La réflexion que nous avons eue à ce niveau est la portée de numérotation des liens dans la page. Par défaut Vimperator numérote les liens visibles sur l’écran à un instant T. Il est intéressant d’avoir une numérotation sur l’intégralité de la page, en effet de cette manière permet à l’utilisateur de faire des interactions intermédiaires (monter, descendre dans la page) avant de visiter une nouvelle page.



#### Gestion des Favoris et des préférences utilisateurs

L’utilisation des favoris est une fonctionnalité dont une grande majorité des usagers du web se servent. Nous avons trouvé diverses manières quant à ce problème. Dans un premier temps, nous avons pensé à l’utilisation de fichier, grâce à ceci l’utilisateur pourra éditer certaines préférences (vitesse de balayage, etc.), nous l’avons dans un premier temps adapté à la gestion des favoris. Par la suite nous avons trouvé comment manipuler les favoris de Firefox directement.

##### Via l’utilisation de fichier

Nous proposons plusieurs solutions à ce niveau. Nous avons regardé plusieurs méthodes afin d’avoir un aperçu des différentes possibilités que nous offrent les technologies du web et de Firefox.

###### Avec Firefox

En utilisant cette méthode, on peut lire, écrire, manipuler des fichiers à travers un objet spécifique. Quelques problèmes de compatibilité entre différentes versions de Firefox ont été soulevés suite à son utilisation.

###### Avec FileReader et FileWriter

Une autre solution est de passer via ces objets qui permettent respectivement la lecture et l’écriture directement en JavaScript sans passer par Firefox. Ces objets sont issus de HTML5. Dans l’écriture de code, cette méthode est plus lisible et plus intuitive que via Firefox.

###### Avec Ajax

Dans la mesure où nous utilisons un serveur local pour la lecture d’un script servant au balayage, nous pouvons aussi utiliser l’Ajax pour récupérer des données de préférence. Cela nécessite bien sûr d’avoir les fichiers sur le serveur local.

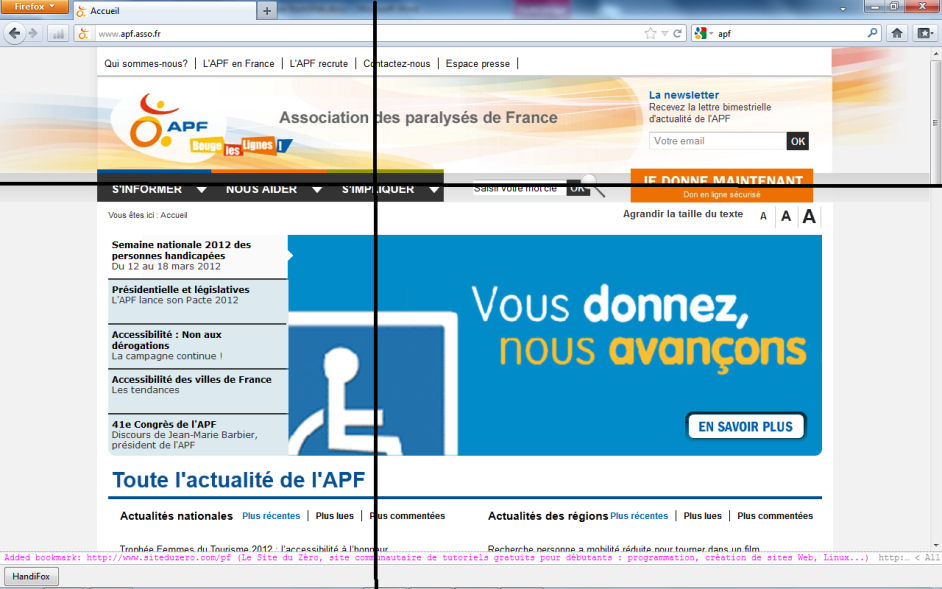
##### Base de données SQLite de Firefox pour la gestion des favoris

Firefox utilise une base de données SQLite comme base de données interne. Il est possible de manipuler les favoris à travers deux objets. Le premier servant pour la lecture, écriture, modification des informations sur un lien donné. Le second pour récupérer l’ensemble des liens suivant certains critères (répertoire, nom, etc.).

#### Le balayage

Nous exploitons le balayage de l’écran dans le cas ou Vimperator n’est plus suffisant. Comme cité auparavant dans le cas des menus déroulants, lorsqu’une fenêtre modale s’ouvre, ou simplement si l’utilisateur souhaite déclencher une action qui n’est ni associé à un élément formulaire ou lien.

##### Écran



Ce type de balayage est utile dans le cas où l’utilisateur aurait à manipuler l’interface de Firefox par exemple. Les traits noirs sont les curseurs utilisés pour le balayage.

##### Fenêtre



Le trait signalant balayage en y

Ce type de balayage est plus adapté au contenu web, en effet ce balayage est spécifique au contenu ce qui représente un gain de temps par rapport à un balayage-écran.

### Téléchargement

La navigation jusqu’au téléchargement du contenu ne pose pas de problème en effet Vimperator rempli très bien ce rôle. Pour ce qui est du choix de répertoire de destination, on pourrait utiliser un balayage-écran.

### Vidéo et Musique

Il existe divers moyens d’insérer une vidéo dans une page par exemple avec une balise non standard : <embed> ou la balise standardisée <object> qui permet d’incorporer du contenu dans une page HTML. Ce contenu pouvant être du HTML, ou de la vidéo. Avec l’arrivée du HTML5, une nouvelle méthode se développe avec l’utilisation de la balise <video>, cette balise permet une manipulation plus aisée de la vidéo.

Du fait que certaines balises peuvent ne pas être interprétées, on peut se retrouver dans le cas d’une imbrication de ces balises afin que le navigateur soit capable de la lire. Ce type d’imbrication, pouvant devenir vite compliqué, ne simplifie pas le cas de la manipulation de vidéo. Par contre, avec le système de balayage expliqué dans une partie précédente il est facile de lancer une vidéo, qui nécessite une « activation » manuelle. En effet il est fréquent que les vidéos se lancent automatiquement sous des sites comme YouTube.

### Copié/Collé

L’utilisation d’un objet propre à Firefox devait permettre cette manipulation. Nous avons testé un code disponible sur le site [www.developer.mozilla.org](http://www.developer.mozilla.org) qui semble ne pas fonctionner. Après de plus amples recherches, il semble que Firefox bloque cette fonctionnalité dans les dernières des raisons pour cause de sécurité.

### Impression

Il est possible de faire l’impression d’une page via JavaScript ou encore via le balayage-écran.

## Technologie pour la mise en œuvre du plug-in

### XUL

XUL est un langage permettant la mise en œuvre d’une interface graphique, au même titre que le HTML. A titre de comparaison.

XUL

<window xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

<box>

<description>Hello world!</description>

</box>

</window>

HTML

<html>

<body>

<p>

Hello World!

</p>

</body>

</html>

### JavaScript

L’utilisation de JavaScript permet d’amener le dynamisme à XUL au même titre qu’il le fait pour le HTML.

### Ajax

Dans le cadre de ce projet, l’Ajax est employé afin d’exécuter des scripts côté serveur. Afin d’éviter une mise en application délicate d’Ajax, nous utilisons la librairie JQuery.

## Mise en œuvre

Le temps étant assez court pour la réalisation du projet, l’ergonomie du clavier n’a pas été étudiée en profondeur pendant le développement puisque ce n’était pas notre objectif premier. Nous voulions délivrer un prototype avec le maximum de fonctionnalités indispensables à la navigation (liens, favoris, navigation dans la page et dans les onglets, permettre de fermer une fenêtre pop-up). Ces fonctionnalités primordiales sont présentées dans la présentation de l’interface qui suit.

### Interface du prototype

#### Page d’accueil

La page web se présente de la façon suivante :

* A gauche de la page se trouve le clavier de navigation et à droite la page web concernée
* Le balayage des boutons s’effectue de gauche à droite et le clic devra être effectué dès lors que le bouton associé à la fonction souhaitée sera rouge

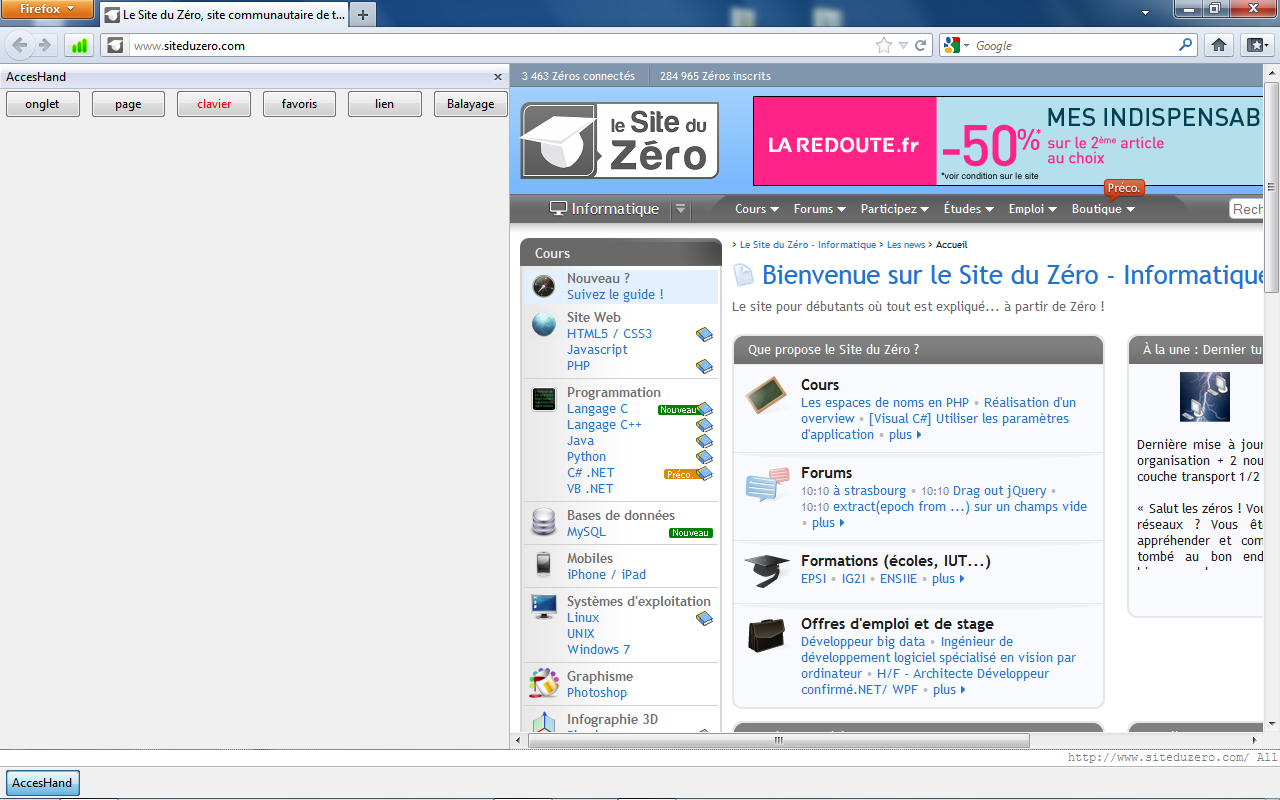


Figure 1 : impression-écran de la page d'accueil du plug-in

#### Les onglets

Le bouton onglet va nous permettre de naviguer dans les onglets de la fenêtre courante.



Figure 2 : impression-écran de la fonctionnalité onglet

Les boutons ‘gauche’ et ‘droit’ sont respectivement pour naviguer vers l’onglet de gauche ou de droite, lorsque l’onglet à l’extrémité droite est rencontré, l’onglet suivant sera le premier et vice versa. Il est également possible d’ouvrir ou de fermer un onglet via les boutons ‘ouvrir’ ou ‘fermer’. Par défaut, un nouvel onglet s’ouvre à la suite des onglets déjà ouverts. Il faudra, malgré l’ouverture, naviguer vers ce dernier. Cette nouvelle page sera le moteur de recherche Google.

#### Page

Le bouton page va permettre de naviguer dans la page courante.



Figure 3 : impression-écran de la fonctionnalité page

Les boutons ‘monter’ et ‘descendre’ permettent de faire défiler l’écran et ceci sur une page. il est également possible, à l’aide des boutons ‘précédent’ et ‘suivant’ de naviguer dans l’historique des pages visitées.

#### Le clavier

Le bouton ‘clavier’ va comme son nom l’indique nous présenter le clavier du plug-in. Celui-ci est une simple image de clavier AZERTY sur laquelle un numéro est associé à chaque caractère. Le clavier numérique en dessous de l’image permettra à l’utilisateur de sélectionner le caractère souhaité. Par exemple pour sélectionner le caractère ‘n’, il suffira de sélectionner le numéro 14. Ce clavier numérique se base sur un double balayage afin de permettre une sélection la plus rapide possible. En effet il faudra d’abord sélectionner la colonne où se trouve le numéro 1 dans notre exemple puis un second balayage de cette colonne se déclenche et il faudra donc sélectionner 1 au moment où le bouton est bleu. Nous avons distingué le balayage vertical et horizontal afin d’avoir une meilleure vision du numéro à sélectionner. Une fois les deux numéros sélectionnés, le caractère souhaité s’inscrit automatiquement dans la zone de texte active.

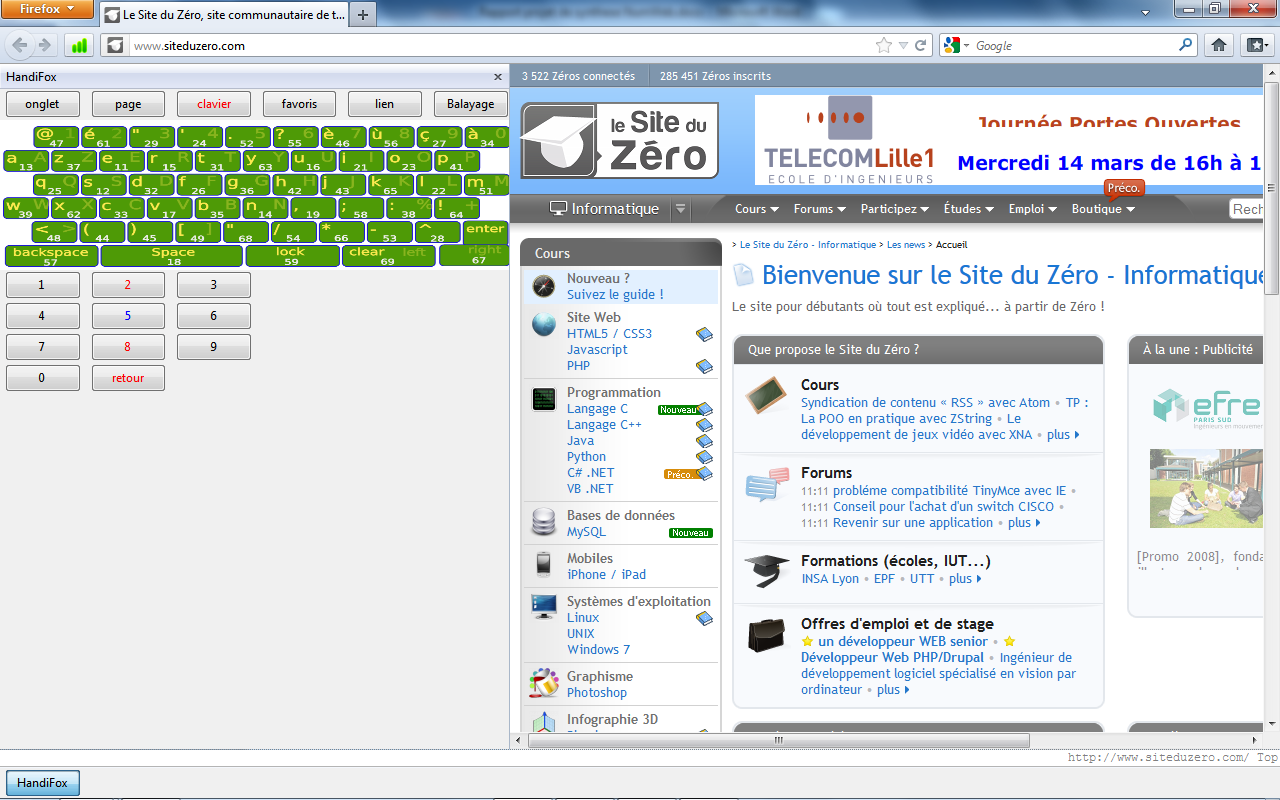


Figure 4 : Impression-écran de la fonctionnalité clavier du plug-in

#### Les favoris

Le bouton ‘favoris’ permettra bien évidemment d’avoir accès aux boutons correspondant aux pages sélectionnées comme favoris par l’utilisateur. Le bouton aura comme libellé le nom du site et le clic aura pour action d’ouvrir le site dans un nouvel onglet. Il est aussi possible de rajouter ou de supprimer un favori de la liste et du fichier par la même occasion ; dès lors que la page à rajouter ou à supprimer est active. Comme dit précédemment, les favoris seront stockés dans la base de données SQLite utilisées par Firefox.



Figure 5 : impression-écran de la fonctionnalité « favoris » du plug-in

#### Les liens

Le bouton lien va permettre la coloration et la numérotation de tous les liens visibles ou non de la page ainsi que l’apparition d’un clavier numérique qui permettra la sélection du lien souhaité. Le choix ici a été de reprendre la numérotation des liens proposées par Vimperator mais de l’étendre à la totalité de la page. Ainsi lorsqu’un utilisateur revient sur un site régulièrement, la numérotation des liens ne changera pas, ou du moins pour les informations importantes (menu), et il n’aura plus qu’à sélectionner ce numéro qu’il aura sûrement retenu de ses anciennes visites. Cela lui permettra un gain de temps puisque la recherche du numéro d’un lien en bas de page ne sera plus à faire.

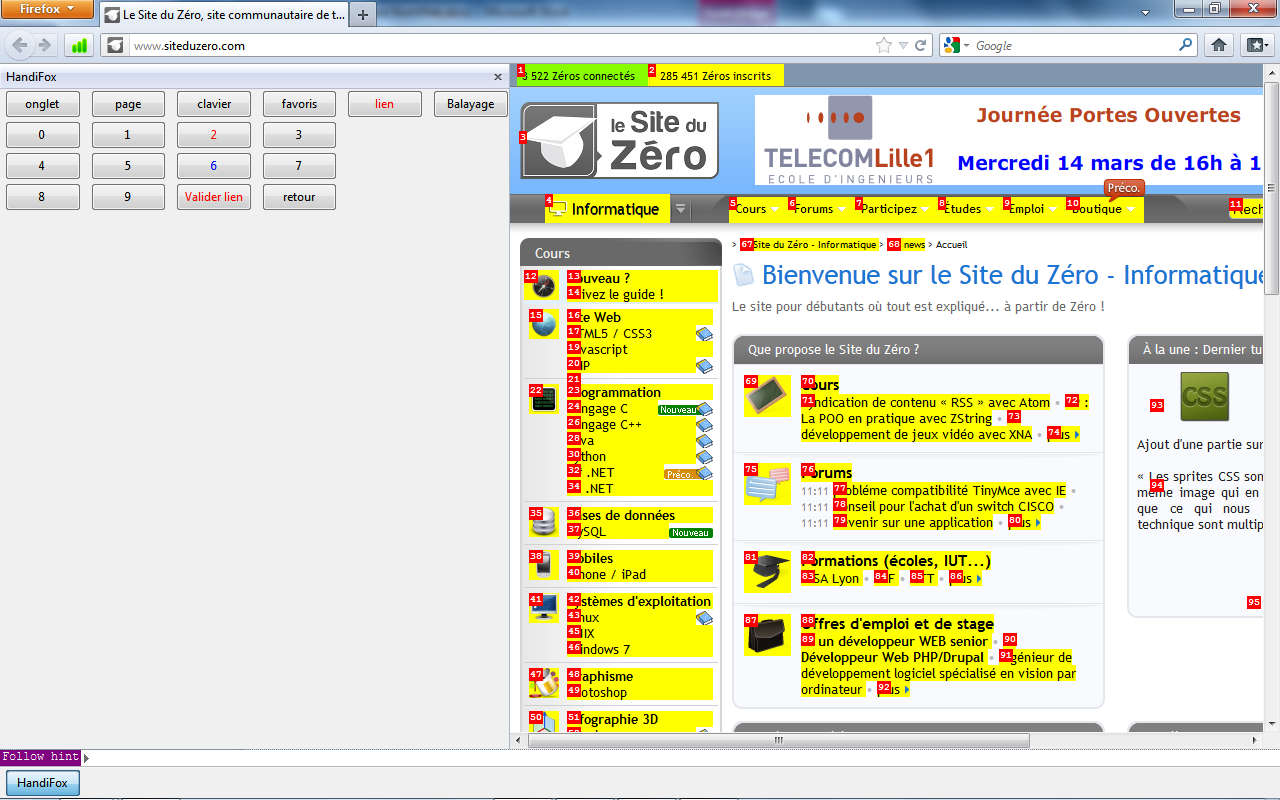


Figure 6 : Impression-écran de la fonctionnalité liens du plug-in

#### Le balayage

Le bouton ‘balayage ‘ est utile dans le cas où l’utilisateur souhaiterait cliquer à un endroit précis de la page. Cette fonctionnalité a été développée afin d’avoir accès aux informations non disponibles par les liens de façon directe. Par exemple elle sera utile pour la manipulation de vidéo, sur des sites tels Dailymotion ou YouTube. La fonctionnalité est déclinée pour permettre, un clic sur la page web directement comme expliquer précédemment. Mais notre volonté est également de pouvoir interagir avec des fenêtres systèmes, qui s’ouvriraient lors de la navigation. Dès lors, le plug-in perdra la main sur la page. Pour cela, un balayage de la fenêtre entière sera pris en compte. Cette fonctionnalité n’est pas encore intégrée dans cette version du plug-in.



Figure 7 : Impression-écran de la fonctionnalité balayage du plug-in

### Fonctionnement du prototype

#### Architecture

Comme expliqué précédemment, le prototype est basé sur l’utilisation du clavier à balayage, qui se trouve à gauche dans la page, à l’aide d’une entrée binaire équivalente à celle du bouton 1 de la manette. L’architecture globale du plug-in est la suivante.

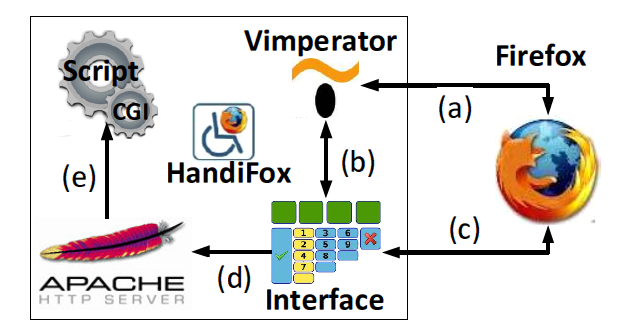


Figure 7 : Architecture d'HandiFox

Comme le schéma le montre, une première connexion est faite avec Vimperator. Cette connexion est simplement l’utilisation de la numérotation des liens, proposée par ce dernier avec le changement de la numérotation sur toute la page. Une deuxième connexion existe directement avec Firefox pour interagir directement avec ce dernier pour ce qui concerne toutes les autres fonctionnalités de HandiFox. Enfin la dernière connexion se fait avec un serveur Apache, celle-ci est indispensable en ce qui concerne le balayage. En effet, afin d’interagir avec les éléments JavaScript d’une page ou d’une fenêtre, non-fille, qui s’ouvrirait et qui nous ferait perdre la main, nous utilisons un serveur Apache couplé à un script CGI (Common Gateway Interface - Interface de Passerelle Commune).

#### Clavier à balayage

|  |  |
| --- | --- |
| Sur une ligne | Sur 3 lignes |
| C:\Users\Sarah\Downloads\screenshot HandiFox\screenshot HandiFox\Principale.png |  |

Le balayage des différents claviers présentés s’effectue à l’aide de Timer et d’appels récursifs. Comme le montrent les différentes impressions-écrans, le clavier peut s’étendre sur une ou plusieurs lignes, selon un paramétrage manuel. En fonction du nombre de lignes, indiqué pour l’affichage du clavier, un balayage une dimension ou deux dimensions est utilisés.

Lors de la construction de ces claviers l’action appropriée et associée directement au bouton, ainsi il suffit de récupérer le bouton choisi lors du balayage et d’appeler sa méthode « click ». Ce mécanisme permet de conserver l’interaction avec la souris. Dans ces impressions-écrans les boutons sont classiques, mais on peut aussi leur associer une image pour un affichage plus agréable.

### Le CGI (Common Gateway Interface)

Un CGI est un script ou un binaire exécutable qui permet de communiquer avec un serveur HTTP. Il écrit dans notre cas en C++, mais qui peut aussi l’être en Java ou Perl, et est exécuté sur un serveur local. Il permet d’exécuter des actions à partir des données transmises par le navigateur lors d’une action spécifique de l’utilisateur comme des coordonnées-écrans dans notre cas. Il peut aussi se contenter d’afficher une page HTML. En annexe le code utilisé. Nous avons remarqué que la dernière chose devant être effectuée par le programme est la génération de la page HTML, si besoin. En effet, il semble que le script soit mal interprété si ce n’est pas le cas (l’action est exécutée, mais un message d’erreur est retourné).

Le programme CGI se compile de la même manière qu’un programme en mode console. La précaution à prendre est de modifier l’extension de l’exécutable généré en « .cgi ». Ceci pour qu’il puisse être reconnu par le serveur. Il est aussi possible de modifier la configuration de serveur pour qu’il accepte une autre extension.

### Relation prototype/CGI

Après avoir vu ce qu’est un CGI, nous allons décrire la relation entre ce CGI et HandiFox. L’utilisation de la souris reste indispensable pour le plug-in. En effet, les éléments JavaScript d’une page restent toujours délicats à traiter. Un problème est remonté à la surface au moment de trouver une solution pour manipuler le pointeur souris, afin de simuler un clic à un endroit donné par le. Effectivement, pour des raisons de sécurité, il est impossible de déplacer le curseur avec un simple script écrit en JavaScript avec Firefox. Nous avons donc dû utiliser un script CGI associé à un serveur Apache : XAMPP afin de permettre ce déplacement. Lorsqu’un déplacement de souris est nécessaire, l’interface envoie une requête au serveur local afin d’invoquer le script CGI avec les coordonnées désirées. Cette requête est envoyée en Ajax via JQuery.

## Tests utilisateurs

Afin d’avoir un avis sur la mise en œuvre du plug-in, nous avons fait passer quelques tests utilisateurs. Cette partie sera donc consacrée à ces tests et aux résultats que nous avons eus.

### Protocole des tests

#### Objectif

L’objectif de ces tests est bien évidemment d’avoir un retour utilisateur concernant le plug-in HandiFox. Ces tests ne seront donc pas dirigés avec une volonté de comparer l’efficacité du moyen de navigation traditionnel (clavier + souris) avec celle du plug-in, puisque qu’il est évident, niveau performance, que le plug-in ne permet pas une rapidité d’exécution des tâches, étant donné l’utilisation d’une entrée binaire. Les tests auront donc pour but d’avoir un avis sur l’implémentation du plug-in et sur les choix pris tout au long du développement des fonctionnalités.

#### Environnement

En ce qui concerne l’environnement d’utilisation, les tests ont été effectués sous Mac avec la version Windows 7 édition familiale premium installé avec bootcamp, avec un processeur Intel Core 2 Duo P8600 et une mémoire Ram 4.0Go.

Les utilisateurs ont été mis dans des conditions similaires pour les tests. Un petit tutoriel du plug-in est présenté avant de passer aux scénarii à réaliser. Ce tutoriel a pour but de présenter en quelques minutes le plug-in et les fonctionnalités disponibles pour la navigation. La présentation reprend chaque point du menu du plug-in en expliquant sa fonction et comment l’utiliser. Puis l’utilisateur renseigne un premier formulaire, et effectue les scénarii. Pendant ce temps, nous restons à sa disposition afin de le guider s’il a un problème lors de la navigation puisque le but, encore une fois, n’est pas de montrer la performance du plug-in, mais plutôt d’avoir un retour de satisfaction. Enfin le testeur répond à un questionnaire qui nous permet de tirer des conclusions quant à son avis à propos de HandiFox et de certains points en particulier.

#### Utilisateurs

Le temps ne nous a pas permis de faire des tests sur les vrais futurs utilisateurs de HandiFox, nous avons donc décidé de tester l’application sur des personnes valides. Des tests sur des personnes ayant un handicap physique seront faits ultérieurement. Cinq personnes âgées de 16 à plus de 50 ans ont testé l’interface, elles ont des niveaux en informatique variés. Le tableau suivant est issu du préquestionnaire effectué :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Utilisateur** | **Sexe** | **Age** | **Temps passé sur internet quotidiennement (par heure)** | **Navigateur utilisé** | **Utilisation d’un curseur à balayage** | **Déjà naviguer sans la souris** |
| **U1** | Homme | 21-30 | Supérieur à 4 | Firefox, Chrome, Safari | Non | Oui |
| **U2** | Homme | 21-30 | Supérieur à 4 | Chrome | Ne connais pas | Oui |
| **U3** | Homme | 21-30 | De 2 à 4 | Firefox | Non | Oui |
| **U4** | Femme | 15-20 | De 2 à 4 | Firefox | Ne connais pas | Non |
| **U5** | homme | 50 + | Inférieur à 2 | Firefox | Non | Non |

Nous avons donc choisi de faire passer les tests sur des personnes de tout âge, ayant des connaissances en informatiques variées afin d’avoir des avis venant de personnes les plus différentes possible et donc avoir un maximum d’informations sur les choix de développement. Les testeurs ont été prévenus au préalable qu’il ne s’agissait que d’un prototype et que l’ergonomie n’était pas définitive. Le seul point qu’ils devaient prendre en compte est l’intuitivité des menus et des choix de navigation.

#### Scénarii

Chaque utilisateur a eu trois scénarii à effectuer. Chaque scénario a été effectué au préalable afin de ne pas rencontrer de problèmes pendant les tests. Donc chaque tâche à effectuer est bien réalisable à l’aide du plug-in. Lors de la présentation de chaque scénario, une indication est donnée sur les fonctionnalités à utiliser, afin de réaliser la tâche. Les indications données guident le testeur à travers le scénario. Les scénarii sont les suivants :

* Rechercher « Metz » avec le moteur de recherche Google. (Indication : liens et clavier)
* Aller sur l’onglet comportant la page du site DailyMotion . Mettez la première vidéo de la page en marche. (Indication : onglet et balayage)
* Ouvrez la page du site « Laposte » qui est un favori et allez sur la page concernée. Puis sélectionnez le lien « Particulier », qu'elles sont les 5 outils et services les plus utilisés ? (information en bas de page) (Indication : favoris, onglet, liens et page)

Les scénarii ont été choisis de sorte que toutes les fonctionnalités du plug-in soient utilisées au moins une fois.

### Résultats des tests

Toutes les tâches ont été effectuées avec succès et les indications se sont avérées inutiles pour la plupart des personnes. Les différents scénarii se sont effectués dans un temps raisonnable et quasi égal pour tous les utilisateurs. Quelques remarques ont été émises pendant les tests concernant l’ergonomie du plug-in, mais ce n’était pas le but ici. Une remarque récurrente concernant la vitesse de balayage du clavier, qui est trop rapide pour certains, a été émise.

### Post questionnaire

A la fin des tests, un postquestionnaire a été rempli par les utilisateurs afin de connaitre la satisfaction globale de l’utilisation du plug-in. Une discussion avec chaque testeur à propos du plug-in nous a également permis de faire un point sur leur satisfaction. Les résultats sont les suivants :

**Question 1 : Le balayage vous semble-t-il facile à utiliser ?**

Il s’agit du balayage du clavier dans ce cas. Le graphique suivant montre les résultats obtenus pour cette question.

Il s’avère donc que quatre personnes sur cinq ont été satisfaites par l’utilisation du clavier à balayage. Aucun problème majeur n’a été relevé lors des tests concernant le clavier mis à part le temps trop long pour écrire la requête lors du scénario 1. Mais ce problème est un sujet de recherche effectué par d’autres étudiants. Malgré la vitesse de balayage, les personnes ont trouvé l’idée excellente du fait de la difficulté de prendre en compte l’entrée binaire et aucune autre solution n’a été proposée.

**Question 2 : Pensez-vous que HandiFox nécessite une formation ?**

Avant chaque test, l’utilisateur a eu droit à un petit tutoriel afin de comprendre en détail les fonctionnalités de HandiFox ainsi que la méthode d’interaction.

D’après le graphique, ce tutoriel s’est avéré utile pour 60% des testeurs qui n’auraient pas forcément su utiliser le plug-in aussi facilement sans cette petite formation. Sachant que les 40% des testeurs qui pensent que HandiFox ne nécessite pas de formation sont des étudiants en informatique qui avaient déjà entendu parler du plug-in. Ce résultat nous conforte dans l’idée de proposer un tutoriel d’utilisation aux futurs acquéreurs du plug-in.

**Question 3 : Les points à améliorer**

L’ergonomie du plug-in est clairement un point à améliorer. Mais pour notre prototype, les points à améliorer selon les utilisateurs sont les suivants :

* Mise en évidence du bouton courant, ou sélectionné, plus visible
* Ajout de la première ligne de balayage lors du balayage
* Amélioration de la sélection des liens

**Question 4 : Avez-vous eu des difficultés à vous adapté à HandiFox ?**

D’après le graphique suivant, les utilisateurs ont eu du mal à s’adapter à HandiFox à hauteur de 40%. Ceci démontre qu’il y a encore un effort à effectuer concernant l’utilisabilité du plug-in. Mais ce résultat est toujours délicat à prendre en compte puisque les testeurs ne sont pas le public cible du plug-in et ces difficultés peuvent ne pas en être pour des personnes handicapées qui n’ont pas l’habitude de naviguer sur Internet de façon autonome.

**Question 5 : Si oui quelle(s) difficulté(s) avez-vous rencontrée(s) ?**

Les difficultés exprimées par les utilisateurs concernent essentiellement la vitesse d’utilisation ainsi que l’utilisation du clavier AZERTY avec les numéros pour chaque caractère. Etant donné qu’il ne s’agit que d’un prototype et que le clavier est un point à améliorer obligatoirement, nous avons décidé de ne pas prendre en compte cette difficulté qui sera résolue dans les prochaines versions du plug-in.

**Question 6 :** **Quelle option de navigation préférez-vous ?**

Concernant les options de navigation entre liens et balayage, le graphique précédent nous indique que les utilisateurs ont préféré le balayage. La facilité d’atteindre un lien avec le balayage a plu, mais c’est surtout le temps mis pour sélectionner un lien avec la seconde méthode qui a été décisive. Une amélioration dans la mise en œuvre du suivi de liens doit être faite afin de permettre aux utilisateurs de sélectionner un lien plus rapidement.

### Conclusion

Pour conclure à propos des tests utilisateurs, le retour global concernant le plug-in est plutôt positif. En effet, outre l’interface prototype, les utilisateurs ont aimé la volonté de proposer l’accès au Web à tout le monde. Les fonctionnalités développées pour le moment ont été considérées comme suffisantes pour une utilisation complète, mais simple du Web.

## Conclusion

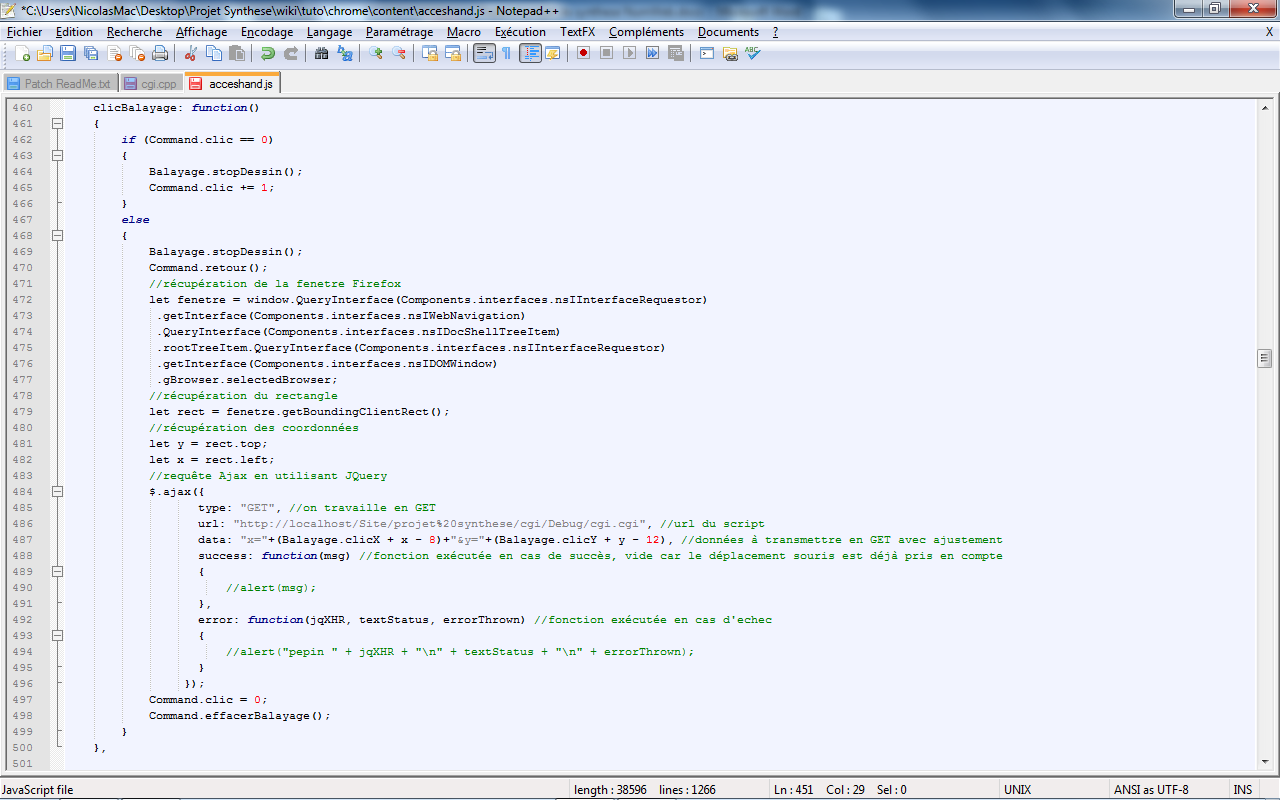
En conclusion, le sujet de ce projet de synthèse nous porter vraiment à cœur. Malgré les difficultés et les problèmes rencontrés tout au long du développement, nous avons réussi à créer un prototype assez complet qui permettrait une utilisation du Web de façon autonome pour les personnes handicapées.

Plusieurs tests utilisateurs devront être encore effectués sur des personnes correspondantes au public cible afin de mettre en évidence des problèmes que nous n’aurions pas pu percevoir.

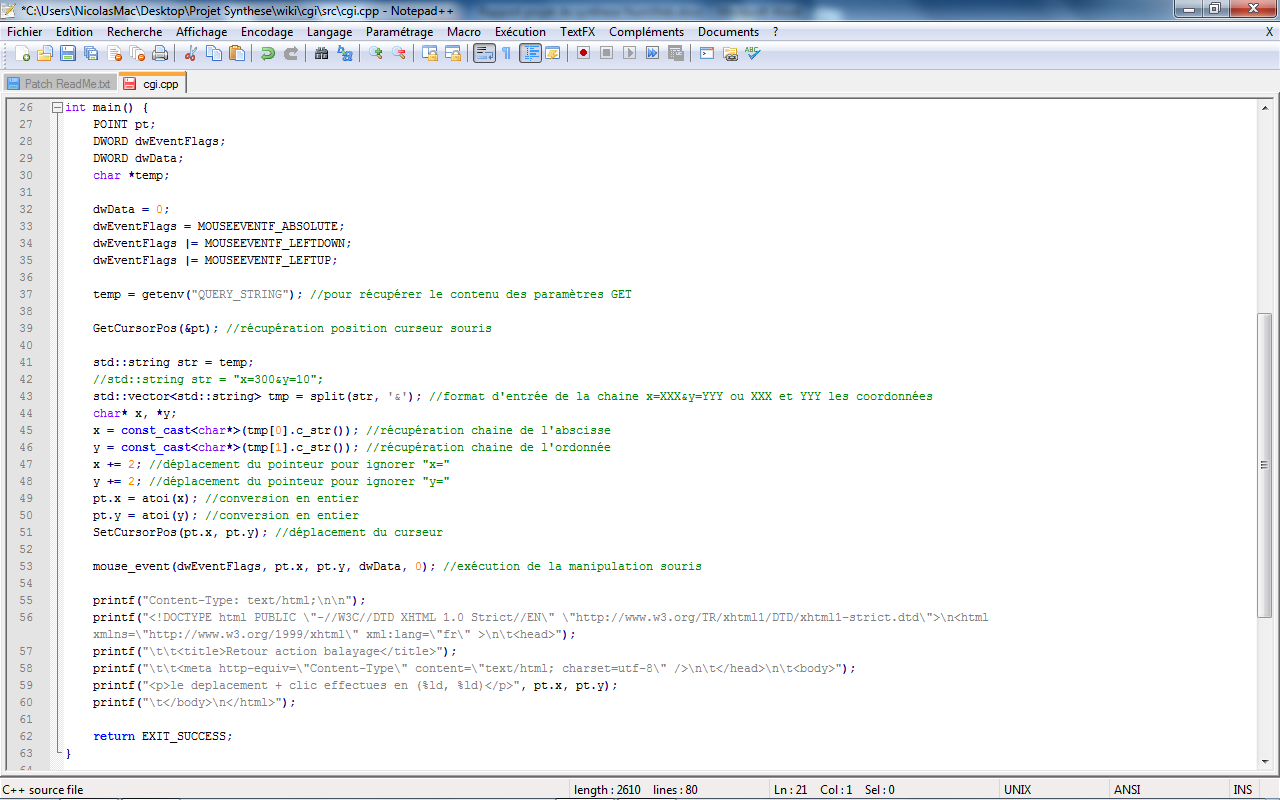
Les prochaines versions du plug-in devront bien évidemment prendre en compte une ergonomie mieux adaptée au sujet et remplir d’autres fonctionnalités plus avancées.

# Annexe

## Ajax : demande de la manipulation souris



## CGI utilisé



Les lignes 33 à 35 permettent de configuration de l’action souris, ici nous demandons la manipulation en coordonnée absolue, puis on demande l’appuie sur le bouton gauche, et enfin la validation du clic par le biais de MOUSEEVENTF\_LEFTUP.

Une ligne importante de ce code est la 37. Cette ligne permet de récupérer les données transmises en Ajax. Nous envoyons les données en GET, nous devons donc les récupérer par le biais de « getenv ». Si nous avions décidé de passé en POST, la récupération des données se serait faite en lisant le flux stdin.

Tables des matières

[I. Introduction 3](#_Toc319056279)

[II. L’existant 3](#_Toc319056280)

[III. Accessibilité du web 5](#_Toc319056281)

[IV. Vimperator 6](#_Toc319056282)

[A. Droits 6](#_Toc319056283)

[B. Critique 6](#_Toc319056284)

[V. Notre démarche 8](#_Toc319056285)

[A. Qu’est-on susceptible de faire sur internet ? 8](#_Toc319056286)

[B. Comment résoudre ces différents problèmes ? 9](#_Toc319056287)

[1. La problématique commune à toutes les tâches : le clic 9](#_Toc319056288)

[2. Navigation 9](#_Toc319056289)

[a) Les liens 9](#_Toc319056290)

[b) Gestion des Favoris et des préférences utilisateurs 10](#_Toc319056291)

[(1) Via l’utilisation de fichier 10](#_Toc319056292)

[(a) Avec Firefox 10](#_Toc319056293)

[(b) Avec FileReader et FileWriter 10](#_Toc319056294)

[(c) Avec Ajax 11](#_Toc319056295)

[(2) Base de données SQLite de Firefox pour la gestion des favoris 11](#_Toc319056296)

[c) Le balayage 11](#_Toc319056297)

[(1) Écran 11](#_Toc319056298)

[(2) Fenêtre 12](#_Toc319056299)

[3. Téléchargement 12](#_Toc319056300)

[4. Vidéo et Musique 12](#_Toc319056301)

[5. Copié/Collé 13](#_Toc319056302)

[6. Impression 13](#_Toc319056303)

[C. Technologie pour la mise en œuvre du plug-in 13](#_Toc319056304)

[1. XUL 13](#_Toc319056305)

[2. JavaScript 14](#_Toc319056306)

[3. Ajax 14](#_Toc319056307)

[D. Mise en œuvre 15](#_Toc319056308)

[1. Interface du prototype 15](#_Toc319056309)

[a) Page d’accueil 15](#_Toc319056310)

[b) Les onglets 16](#_Toc319056311)

[c) Le clavier 17](#_Toc319056312)

[d) Les favoris 18](#_Toc319056313)

[e) Les liens 19](#_Toc319056314)

[f) Le balayage 20](#_Toc319056315)

[2. Fonctionnement du prototype 21](#_Toc319056316)

[a) Architecture 21](#_Toc319056317)

[b) Clavier à balayage 22](#_Toc319056318)

[3. Le CGI (Common Gateway Interface) 22](#_Toc319056319)

[4. Relation prototype/CGI 22](#_Toc319056320)

[E. Tests utilisateurs 23](#_Toc319056321)

[1. Protocole des tests 23](#_Toc319056322)

[a) Objectif 23](#_Toc319056323)

[b) Environnement 23](#_Toc319056324)

[c) Utilisateurs 24](#_Toc319056325)

[d) Scénarii 24](#_Toc319056326)

[2. Résultats des tests 25](#_Toc319056327)

[3. Post questionnaire 25](#_Toc319056328)

[4. Conclusion 28](#_Toc319056329)

[F. Conclusion 28](#_Toc319056330)

[Annexe A](#_Toc319056331)

[Ajax : demande de la manipulation souris A](#_Toc319056332)

[CGI utilisé B](#_Toc319056333)