Projet d'étude Promotion 2005

Projet d'études N°65
Jean-Philippe BIANCHI
Thibaut BRUSSEAUX
Romain CLEDAT
Anna MICHAILOVSKY
Michel SATO

Commanditaire : Laboratoire ICTT, Directeur : Bertrand DAVID
Tuteurs scientifiques : département Mathématiques-Informatique
René CHALON
Bertrand DAVID
Conseiller SHS : Jacqueline VACHERAND-REVEL
Coordonnateur : Alexandre SAIDI

Rapport final – Janvier 2004

TABLE INTERACTIVE POUR LE TRAVAIL COLLABORATIF

(TableGate)

Table des matières

Table des matières détaillée

Introduction

Il y a quinze mois, nous avons choisi de nous intéresser à l'utilisation possible d'une table interactive afin d'améliorer le travail collaboratif. En effet, le travail de l'ingénieur ou du chercheur s'effectue le plus souvent en équipe. Malheureusement, le matériel dont il dispose est très peu adapté à ce genre de tâche : l'ordinateur est un assistant efficace pour rédiger un rapport seul mais peu pratique d'utilisation lorsqu'il s'agit de travailler à plusieurs sur un document affiché à l'écran. Le papier, en revanche, permet d'éviter cette gêne car il est possible de se réunir autour d'une table où sont posés les documents. Cependant, aucune modification n'est possible sur des feuilles et il faut alors repasser par l'informatique pour déplacer des paragraphes, supprimer des zones de texte…

Pourquoi ne pas combiner les avantages de l'informatique (flexibilité) et du papier sur une table (place et vue d'ensemble des documents) ? C'est ce que nous avons voulu faire en créant le logiciel WordTab : un document Word est projeté sur la table tactile et les utilisateurs peuvent le modifier ensemble de manière interactive.

Ce rapport se veut la synthèse de quinze mois de travaux, nous avons donc tenté d'y mettre tout ce que nous avons pu faire et découvert sans pour autant entrer dans des détails techniques qui risqueraient très vite d'être ennuyeux. (Pour ces derniers, se reporter aux annexes ou encore aux autres rapports). Premièrement, nous allons rappeler l'état de l'art existant au début de ce projet ainsi que les objectifs initiaux que nous nous sommes fixés. Ensuite, nous décrirons l'aboutissement de notre travail : le logiciel de traitement de texte WordTab. Ceci nous amènera à décrire les tests que nous avons effectués pour valider son intérêt. Enfin, nous terminerons par une réflexion sur ce qu'a pu nous apporter une telle expérience et sur ce qui s'est passé au sein de l'équipe pendant ces quinze mois.

1 Cahier des charges et état de l'art

1.1 Les commencements du projet

1.1.1 Le dispositif :

Une table interactive est une table physique, horizontale, tactile, reliée à un ordinateur. Un vidéo-projecteur envoie sur la table ce que l'on voit d'habitude apparaître sur l'écran de l'ordinateur. Un utilisateur se sert donc de son doigt ou d'un pointeur quelconque comme il se servirait d'une souris. Il est possible d'augmenter le nombre d'interfaces entre l'utilisateur, la table et l'ordinateur en installant par exemple une caméra vidéo : on peut alors utiliser la reconnaissance de gestes. Pourquoi un tel dispositif est-il intéressant pour notre étude ?

La grande taille de la table la rend particulièrement adaptée au travail collaboratif: plusieurs personnes peuvent voir ce qui se passe en même temps. La mise en réseau de plusieurs tables est également envisageable. Il est possible, sur ce dispositif, de mélanger réel et virtuel, en posant des objets réels sur la table. On peut alors tenter de transférer dans ces objets une signification électronique : les objets seront porteurs d'informations compréhensibles par l'ordinateur : on parle alors de réalité augmentée. L'utilisateur, en présence d'objets de tous les jours (papier, crayons…), est alors dans un environnement plus transparent, plus naturel, pour travailler.

Nous n'avons d'ailleurs pas été les premiers à nous intéresser à ce dispositif, comme le prouvent les recherches que nous avons effectuées sur l'état de l'art.

1.1.2 L'état de l'art

Un dispositif de la table interactive est commercialisé depuis 1999 par Egan Teamboard Inc. sous le nom de TeamBoard, mais des chercheurs ont créé des prototypes similaires depuis 1970. Plusieurs groupes de travail se sont penchés sur l'utilisation de la table. Les articles consultés sur le sujet ont montré que la plupart s'étaient limités aux interfaces décrites en introduction : table tactile munie d'une vidéo-projecteur et, éventuellement, d'une caméra vidéo.

Leurs études ont montré que les utilisateurs s'adaptaient facilement à un tel dispositif, de fonctionnement plus intuitif qu'un couple écran/souris. La projection par le haut a notamment soulevé des inquiétudes quant à d'éventuelles zones d'ombre sous la main d'un utilisateur. En effet, un utilisateur qui passe la main sous le projecteur pour montrer ou pour regarder de plus près quelque chose qui se trouve sur la table va créer une zone noire sur la table, où l'image projetée n'apparaîtra pas. Il ne faudrait pas que cette zone d'ombre cache justement ce que l'utilisateur essaie de montrer! Il n'en est rien car nous sommes tous habitués à travailler sous des lampes de bureau, dont la lumière provient par le haut : un utilisateur de la table interactive ne remarque même pas ce problème. Ces études ont mené à des prototypes tels que des bandes d'informations pour contrôleurs aériensou encore une calculatrice interactive (cf. rapport initial pour plus de détails sur ces applications).

1.2 Cahier des charges :

1.2.1 Logiciel d'édition de texte :

Un tel logiciel a été retenu comme première application à implanter car il exploite deux qualités principales de la table : d'une part l'aide au travail en équipe et d'autre part le mélange d'objets réels et virtuels. Ce logiciel doit être destiné à la structuration de textes longs, surtout lorsque plusieurs personnes doivent y participer. Imaginons qu'une équipe ait produit un rapport d'une dizaine de pages, par exemple sous format Word, dont elle souhaite modifier la structure. Plusieurs problèmes se posent dans le cadre de travail habituel d'un PC. Tout d'abord, il est difficile pour plus de 2 personnes de voir en même temps ce qui se passe sur un écran d'ordinateur. Deuxièmement, on ne peut faire apparaître lisiblement sur un écran qu'une page de texte à la fois, ce qui empêche d'avoir une vue d'ensemble du document.

La table interactive offre des solutions à tous ces problèmes. Il ne suffit pas de projeter sur la table l'ensemble du texte : la précision de la projection n'est pas suffisante. On va mélanger objets réels et virtuels de la façon suivante : le rapport sera tout d'abord imprimé dans l'état où il est, et gardé en mémoire dans l'ordinateur. On va disposer les feuilles imprimées sur la table, les unes à côté des autres, et indiquer à l'ordinateur où on a placé celles-ci. Ainsi, on a l'intégralité du apport sous les yeux à tout moment, et puisqu'il s'agit de feuilles de papier, on peut soulever une feuille, l'approcher de soi et la lire si l'on souhaite connaître le détail de ce qui est écrit, en particulier, on pourra faire circuler ces feuilles à tous les membres de l'équipe présents.

Il suffira ensuite de sélectionner un paragraphe en montrant son emplacement sur la feuille, et de le «déplacer », en indiquant l'emplacement souhaité : une flèche, reliant l'emplacement original au nouvel emplacement, apparaîtra virtuellement sur le document, c'est-à-dire qu'elle sera projetée sur le document réel. Plusieurs modifications (copier, coller, déplacer, supprimer) pourront être ainsi effectuées (avec la possibilité d'en annuler une à tout moment) : lorsque le document sera devenu illisible car saturé d'indications, on imprimera le document modifié et on pourra recommencer.

1.2.2 Utilisation d'une caméra :

L'utilisateur n'aurait plus besoin d'indiquer à l'ordinateur la position des feuilles si une caméra pouvait la repérer automatiquement. Il ne faut pas pour autant penser que puisque l'ordinateur « voit » ce qui se trouve sur la table, une table tactile n'est plus nécessaire : la résolution des caméras est insuffisante pour, par exemple, distinguer une ligne d'une autre dans un texte, et encore plus pour distinguer des caractères. On utiliserait alors le procédé décrit précédemment : la caméra servirait à déterminer une position et non à lire un document. C'est la première application à laquelle nous avons voulu arriver avec la caméra fournie par le laboratoire ICTT.

Pour aller plus loin, nous avons pensé passer à la reconnaissance de geste pour la raison suivante : au début de notre projet, le nombre d'actions réalisables par l'utilisateur (sélectionner, supprimer, etc…) sera raisonnable et il sera possible de les déclencher par le tracé de formes simples sur la table (une croix pour supprimer par exemple). Mais au fur et à mesure de l'avancement de notre travail, le panel des actions de utilisateur pourrait s'étoffer et on ne peut pas trop complexifier les formes à tracer pour les déclencher. Une solution envisageable est l'utilisation d'une

bibliothèque de menu déroulant dans les 8 directions du plan. Le choix d'une direction (par déplacement du stylo) donne accès à 8 sous-menus pouvant eux même se diviser en 7 autres sous menus, et ainsi de suite. Une action se déclenche alors pour une certaine trajectoire du stylo (haut-haut-droite par exemple). L'apprentissage de ces trajectoires par l'utilisateur est simplifié car si le stylo reste immobile, les sous-menus s'affichent lui permettant de choisir la suite de la trajectoire en fonction de l'action à réaliser. Après une période d'apprentissage l'utilisateur trace les trajectoires assez rapidement pour empêcher les menus d'apparaître. La caméra pourrait alors reconnaître directement les gestes de l'utilisateur. Cette implémentation étant complexe, nous avons décidé de nous concentrer sur la reconnaissance de position et éventuellement d'étudier un cas de reconnaissance gestuelle avec un jeu de dame mélangeant pièces réelles et pièces virtuelles. (cf. rapport initial pour plus de détails)

1.2.3 Réflexion sur une mise en réseau possible de plusieurs tables :

Nous avons choisi ici de nous limiter à une réflexion puisqu'une seule table est à notre disposition. Déjà, de nombreuses questions se posent. On comprend aisément l'intérêt pour le travail collaboratif d'une telle mise en réseau : les différentes personnes travaillant sur le projet n'auraient plus besoin d'être dans la même pièce, ni même dans le même pays! Mais mélanger objets réels et virtuels devient plus compliqué : les objets réels posés sur une table devront être reproduits virtuellement sur les autres. Se pose également le problème des modifications simultanées sur différentes tables. Ces questions sont à développer pour une réflexion ultérieure.

Durant nos quinze mois de travail, nous avons tenté de répondre à ces différents objectifs, ce qui nous a amené à la création de notre logiciel WordTab.

2 Aboutissement de nos travaux : WordTab

2.1 L'automation de Word

2.1.1 Qu'est-ce que l'automation ?

Le logiciel Microsoft Word, comme du reste toute la suite Microsoft Office, possède la possibilité d'insérer des macros dans les documents qu'il crée. Ces macros sont des petits bouts de code Visual Basic qui permettent d'effectuer des actions automatiquement sur le document associé. Toute fonctionnalité offerte à l'utilisateur humain de Word, peut être réalisée à l'aide d'une macro en Visual Basic. Ces macros permettent donc de piloter Word.
L'automation de Word consiste à utiliser ce type de code pour piloter Word depuis l'extérieur de celui-ci, en particulier depuis un programme réalisé dans le langage Visual Basic.

Avec Visual Basic il est en effet très simple de créer un objet Word qui donne accès à presque toutes les fonctions élémentaires de Word. On accède à ces fonctions en faisant « MonWd.nom_de_la_fonction » (si on nomme MonWd l'objet Word ainsi crée par simple ajout de la référence msword.olb). Ainsi « MonWd.Documents.open "test.doc" » ouvrira le fichier test.doc dans une fenêtre word. Si on a affecté true à la propriété « MonWd.visible », Word apparaîtra dans sa fenêtre habituelle ; dans le cas contraire Word est totalement invisible et on peut ainsi travailler sur un fichier .doc de manière transparente pour l'utilisateur.

Une des raisons qui nous ont fait nous limiter au paragraphe pour la plus petite sélection possible (en dehors de l'esprit de WordTab qui est de se limiter à l'édition macroscopique du texte) a été que le paragraphe est un objet reconnu par Word et donc plus facile à manipuler.

Ainsi on peut donner l'ordre à Word de sélectionner le nième paragraphe du texte en cours par « MonWd.ActiveDocument.Paragraphs(n).Range.Select ». Remarquons qu'en plus de l'objet Word, il se crée aussi un objet à part entière dans le programme : l'objet sélection qui représente le texte sélectionné dans l'objet word. Cet objet sélection donne accès à toute sorte d'informations indispensables par le biais de la propriété « Selection.informations(Type_info) » où type_info est un paramètre qui désigne comme son nom l'indique le type d'information que l'on souhaite obtenir.
Par exemple, nous on été utile :
- « wdActiveEndPageNumber » qui désigne le numéro de la page sur laquelle se trouve la dernière ligne de la sélection.
- « wdVerticalPositionRelativeToPage » qui désigne la distance entre le haut de la sélection et le haut de la page.
- etc. …

La partie automation de Word a donc été réalisée en Visual Basic, sous la forme d'un contrôle ActiveX nommé « word_ocx ».

2.1.2 L'ActiveX « word_ocx »

Le module automation de Word devant être réalisé en Visual Basic, et le reste de WordTab étant codé en C++, il s'est posé le problème de la communication entre ces deux entités. Le choix du contrôle ActiveX pour sa simplicité de portage n'a pas tenu ses promesses, puisqu'il a été impossible de facilement l'insérer dans notre code VC++. Nous avons développé ensuite une solution plus lourde, mais non moins efficace, qui nous a permis de ne pas perdre le travail effectué sur l'ActiveX. Nous verrons plus loin en quoi elle consiste.
Word_ocx, comme tout ActiveX est conçu comme une classe, et d'ailleurs utilisé comme tel en VC++. On peut le décrire par son schéma UML.

2.1.2.1 Schema UML de word_ocx

word_ocx
- m_fichier_word : Variant - m_vsbl : Booléenn - m_nbr_par : Entier - paragraphe.page(500) : Tableau d'entiers - paragraphe.ymin(500) : Tableau de réels
+ setfichier_word(Variant) + getfichier_word() + setvsbl(Booléen) + getvsbl() : Booléen + nbr_par() : Entier + par_ymin(Variant) : Variant + page_parmax(Variant) : Variant + page_parmin(Variant) : Variant + par_page(Variant) : Variant + affiche_par(Variant,Variant) + kompile(Variant) + compilation() : Variant + ferme() + imprime() + ouvre()

2.1.2.2 Brève description des méthodes

Depuis le conteneur, la seule façon de communiquer avec word_ocx est d'utiliser l'une de ses méthodes.

- "setfichier_word(chemin_d'accés)" : stocke le chemin d'accès du fichier Word à ouvrir dans l'attribut m_fichier_word.
- "getfichier_word()" : renvoie le chemin d'accès du fichier traité.

- "setvsbl(b)" : stocke dans l'attribut m_vsgl si Word sera visible ou pas lors de l'exécution des méthodes "compilation()" et "ouvre()" (b=True : Visible, b=False : Invisible).
- "getvsbl()" : renvoie le contenu de m_vsbl (True : Word Visible, False : Word Invisible)
- "nbr_par()" : renvoie le contenu de l'attribut m_nbr_par qui spécifie le nombre de paragraphes dans le fichier Word ouvert avec "ouvre()".
- "par_ymin(n)" : renvoie la distance en pour cent-mille entre le haut de la feuille et le haut du paragraphe numéro n.
- "page_parmin(n)" (resp "page_parmax(n)") : renvoie le numéro du premier (resp dernier) paragraphe de la page n.
- "par_page(n)" : renvoie le numéro de la page où commence le paragraphe n.
- "affiche_par(par1,par2) : affiche dans une boite de dialogue le contenu du texte ouvert avec ouvre(), du paragraphe par1 au paragraphe par2 (fenêtre de débugage à la base).
- "kompile(n)" : s'appelle de manière itérative, rajoute le paragraphe n à la suite de la liste ordonnée des paragraphes du texte formaté.
- "compilation()" : effectue réellement le formatage, enregistre le fichier modifié et renvoie "OK" en cas de réussite.
- "ferme()" : ferme le document, ainsi que l'objet word.
- "imprime()" : imprime le document en cours.
- "ouvre()" : lit les informations utiles dans le fichier spécifié avec "setfichier_word".

2.1.2.3 Remarque sur le langage d'implémentation

Cet ActiveX ne se comportera pas de la même manière selon qu'il sera utilisé en VC++ ou en Visual Basic.
- En VB : il n'y a pas de méthode différente (set et get) pour écrire dans un attribut. Une méthode qui réalise ces deux opérations est appelée propriété. Même si concrètement, dans le code de l'ActiveX, il s'agit de 2 procédures différentes qui doivent être prévues, vu de l'extérieur, depuis un programme Visual Basic, l'appel de la propriété se fera avec la même instruction. Par exemple
c1 = nom_activeX.vsbl() stockera la valeur de l'attribut m_vsbl dans la variable c1.
nom_activeX.vsbl() = c2 stockera la valeur de la variable c2 dans l'attribut m_vsbl.
Les méthodes « pures » s'appellent de la manière habituelle. Par exemple
nom_activeX.ouvre() lance la méthode ouvre().

- En VC++ : Il n'y a pas de notion de propriété comme méthode spéciale. On écrit dans un attribut avec la méthode commençant par set, et on lit avec la méthode commençant par get.
Ainsi les mêmes opérations que précédemment s'écrivent
c1 = nom_activeX.getvsbl() ;
nom_activeX.setvsbl(c2) ;
ce qui complique l'écriture du code, mais on voit mieux ce qui se passe dans l'ActiveX…

2.1.3 Les 2 exécutables « structure.exe » et « kompil.exe »

Devant les difficultés rencontrées lors de l'utilisation directe de l'activeX word_ocx (qui effectue l'interface entre le programme principal et Word) en Visual C++, nous avons décidé d'adopter une autre stratégie, provisoirement définitive.

Cela a consisté en l'ajout d'une couche entre WordTab et word_ocx, sous forme d'exe indépendants qui lancent les méthodes de l'ActiveX, et utilisent comme variables des fichiers binaires. Ces exe passerelles sont au nombre de deux :

« structure.exe » ouvre le fichier word, sélectionné dans une boite de dialogue qu'il fournit. Il lit les informations utiles à WordTab pour effectuer l'équivalence coordonnées du click sur la feuille et numéro du paragraphe pointé (au nombre de quatre, pour chaque paragraphe : la page de son 1er mot, la distance séparant son 1er mot du haut de sa page, la page de son dernier mot et la distance séparant son dernier mot du haut de sa page). Ces informations sont stockées dans un fichier binaire nommé « structure.txt », dans le même répertoire que « structure.exe ». Il interface WordTab avec les méthodes « fichier_word », « ouvre », « ferme », « par_page » et « par_ymin » de word_ocx.

« kompil.exe » réordonne les paragraphes du fichier word dont le chemin d'accès est passé en paramètre de la ligne de commande, selon l'ordre spécifié dans un fichier binaire « kompil.txt » situé dans le même répertoire que « kompil.exe ». Il interface WordTab avec les méthodes « fichier_word », « kompil » et « compilation » de word_ocx.

A noter que ces deux exécutables sont deux modules indépendants pouvant resservir à toute application travaillant avec Word (pas forcément WordTab). On pourrait aussi les remplacer par des fichiers de même nom qui effectuent les mêmes opérations pour un traitement de texte différent.

Ainsi on y a perdu en compacité et en rapidité du code, mais au final le résultat est le même (la rapidité n'est pas un facteur essentiel pour WordTab), et cela nous a permis de nous concentrer sur les problèmes plus fondamentaux, d'avancer dans les fonctionnalités de notre application.

2.1.4 Conclusion

L'ensemble s'organise donc sur quatre couches, modulables car indépendantes :

Application WordTab
Programmes annexes « structure.exe » et « kompil.exe »
ActiveX word_ocx
Logiciel Microsoft Word

2.2 L'interface graphique

2.2.1 Une première version qui n'a pas abouti :

Bien qu'elle n'ait pas été retenue, la première version de l'interface graphique présentait l'intérêt d'être programmée en utilisant les MFC et les messages de souris gérés par Windows. Son implémentation a donc été beaucoup plus compliquée du fait de notre manque d'expérience dans ce domaine mais a permis d'avoir un code plus léger et un programme moins gourmand en ressources. (pas d'utilisation de bitmap mais plutôt de formes géométriques simples, rafraîchissement économique

de l'affichage…). Elle assurait les fonctions de copie, de déplacement et de suppression mais avec moins de souplesse que la dernière version WordTab. Ayant choisi de faire prévaloir l'esthétique et la fonctionnalité à la simplicité du code, nous n'avons donc pas continué à travailler sur cette ancienne version.

Néanmoins, celle-ci nous aura permis de mettre en forme nos réflexions, de nous orienter vers des choix meilleurs et surtout de coder tout le reste du logiciel (partie formateur de texte et automation de Word qui n'ont presque pas changé) pour fournir un livrable opérationnel. (Pour une description plus technique de cette ancienne version, se référer aux précédents rapports).

2.2.2 La dernière version de l'interface graphique, à la fois esthétique et intuitive :

WordTab est un logiciel qui permet l'édition macroscopique d'un texte au format *.doc*. L'interprétation de ce format repose sur l'automation de Microsoft Word par l'intermédiaire de Visual Basic. Le cœur du programme ainsi que l'interface ont été programmés en C++. L'interface graphique utilise une librairie multimédia multi-plate-forme gratuite nommée SDL (Simple DirectMedia Layer). C'est par l'intermédiaire de cette bibliothèque que le programme gère l'affichage des graphismes et l'environnement sonore.

La présente partie s'attache à la description du fonctionnement de l'interface du point de vue de l'utilisateur. Pour de plus amples détails concernant l'implémentation du code proprement dite, veuillez vous reporter à l'annexe technique correspondante.

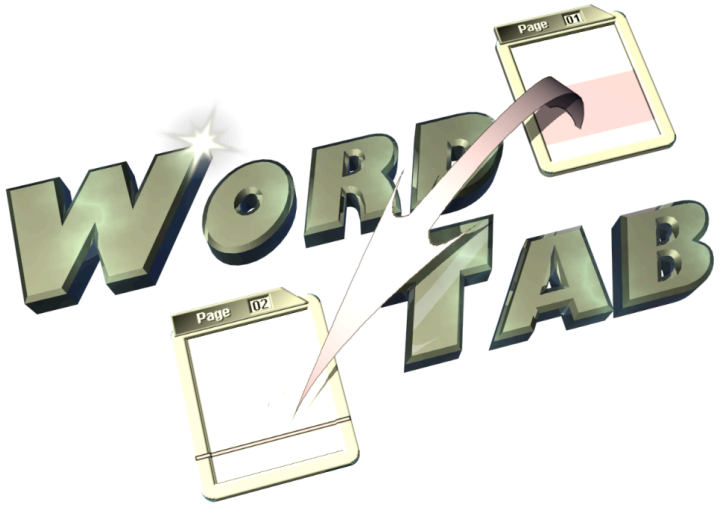

Figure 1 : Bienvenue dans Word Tab

2.2.2.1 Options de base

Les fonctions permises par l'interface sont représentées par un jeu d'icônes situés tout autour de l'écran. Certains de ces icônes sont affichés de manière permanente, d'autres apparaissent ou disparaissent suivant les situations.

2.2.2.1.1 Ouverture du fichier texte

Avant toute chose, il faut ouvrir le fichier *.doc* sur lequel on désire travailler. Il faut cependant s'assurer que ce dernier ne comporte pas de mises en forme trop sophistiquées telle que des tableaux, des colonnes ou des objets superposés au texte… En effet, gardons à l'esprit que WordTab n'est capable de travailler que sur des paragraphes et considère qu'ils s'étalent sur toute la largeur de la feuille.

Figure 2 : Icône d'ouverture de fichier

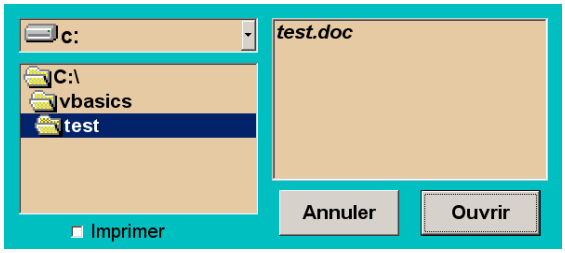

Figure 3 : Ouverture d'un fichier *.doc*

Ainsi, après avoir cliqué sur l'icône « Ouverture de fichier », une boîte de dialogue apparaît et liste tous les fichiers comportant une extension *.doc* présents dans le répertoire sélectionné. Une fois le fichier ouvert, l'icône disparaît et laisse place aux icônes « Enregistrement » et « Impression ». Il faut donc noter que dans cette version, il n'est possible d'ouvrir qu'un seul fichier par session.

A l'ouverture, il est possible de décider d'imprimer le fichier en cochant l'onglet correspondant dans la boîte de dialogue.

2.2.2.1.2 Enregistrement du résultat

Une fois les modifications apportées, il est possible d'enregistrer le résultat. Lorsque l'on actionne l'icône « Enregistrement », le logiciel ré-agence les paragraphes suivant les modifications apportées par l'utilisateur et enregistre le résultat dans un fichier Word portant le nom du fichier initial auquel il rajoute le suffixe *_tag*.

Figure 4 : Icône d'enregistrement

2.2.2.1.3 Impression du résultat

Figure 5 : Icône d'Impression

L'icône « Impression » a le même effet que l'icône « Enregistrement ». Cependant, il lance l'impression du nouveau fichier après sa création, directement depuis l'interface. Ceci évite d'avoir recours à Microsoft Word pour imprimer le résultat des manipulations.

2.2.2.1.4 Verrouillage de la table

Lors de la manipulation d'objets sur la table, il est possible de verrouiller son accès afin d'éviter toute pression malencontreuse : il suffit d'actionner l'icône « Verrouillage de la table ». Une fois actionné, il reste enfoncé et devient l'unique partie sensible de la table. Une seconde pression aura pour effet de déverrouiller la table.

Figure 6 : Icône de verrouillage de la table

2.2.2.1.5 Sortie du programme

Figure 7 : Icône de sortie de programme

La sortie du programme ce fait par l'icône « Sortie de programme ». Attention, il faut noter qu'aucune structure de sauvegarde de l'état du logiciel ou d'avertissement de sortie n'est prévue. Ainsi, si l'utilisateur presse cet icône sans avoir enregistré préalablement son travail en cliquant sur les icônes « Enregistrement » ou « Impression », les données concernant les manipulations effectuées seront irrémédiablement perdues.

2.2.2.1.6 Basculement entre les deux modes d'édition

Une fois le fichier texte chargé, son édition avec WordTab s'effectue en deux temps. Tout d'abord, il faut préparer le plan de travail : disposer les feuilles sur la table tactile et indiquer à l'application leur position. Ensuite, l'édition proprement dite peut débuter : l'utilisateur peut commencer à manipuler les sélections.

Ainsi, l'interface peut être mise dans deux états : le mode « Edition des feuilles » et le mode « Edition des sélections ». Les deux modes sont accessibles à tout moment, ce qui permet d'ajouter ou de retirer des pages en cours de manipulation, toute en concervant les données relatives aux sélections déjà effectuées. Il faut noter que certaines actions ne seront possibles que dans l'un ou l'autre de ces modes. L'état dans lequel se trouve l'interface est systématiquement indiqué en haut à gauche de l'écran. Le basculement entre les deux modes se fait à l'aide de deux icônes situés en bas de l'écran.

Figure 8 : Icônes de basculement entre les modes « Edition des feuilles » (à gauche) et « Edition des sélections » (à droite)

2.2.2.2 Mode « Edition des feuilles »

Par défaut, c'est le mode d'édition des feuilles qui est actif à l'ouverture du logiciel. Ceci permet de commencer directement à entrer les positions et dimensions des feuilles une fois celles-ci déposées et le fichier Word chargé. Cette étape s'effectue très simplement pour peu que les feuilles aient bien été disposées sur la table (c'est-à-dire bien à plat, parallèlement aux bords de la table). Pour faciliter cette étape, une grille de positionnement est présente dans ce mode.

Le programme considère que les feuilles réelles occupent exactement l'intérieur noir des fenêtres, c'est pourquoi cette phase d'édition est primordiale pour une bonne précision des données projetées.

Figure 9 : L'interface en mode édition des feuilles

2.2.2.2.1 Création des fenêtres

Une simple pression sur la table permet d'obtenir une fenêtre de taille minimale. Cette fenêtre représente la page dont le numéro est affiché dans son entête.

Le pointeur défini en fait le coin supérieur gauche de la feuille directement extérieur au rectangle intérieur noir (croix rouge sur la figure, cette croix n'apparaît pas dans le logiciel). Tant que la pression est maintenue, il est possible de déplacer ce point et donc la fenêtre pour s'approcher au mieux du coin de la feuille réelle.

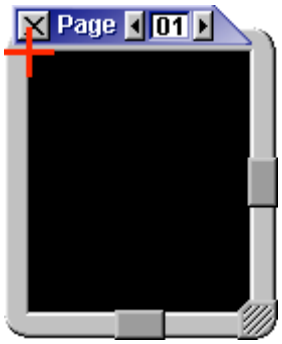

Figure 10 : Fenêtre d'encadrement d'une page

Après avoir relâché la pression, il est de toute façon encore possible de modifier la position de la fenêtre.

Une même page n'est représentée qu'une seule fois. De ce fait, l'ajout de fenêtres supplémentaires reste possible tant que le nombre total de pages contenues dans le texte n'a pas été atteint. De plus, il faut tenir compte du fait qu'il n'est pas permis de superposer plusieurs fenêtres, ni de faire sortir leur cadre de l'écran. Ainsi, si la place est insuffisante ou si le nombre total de pages est atteint, la fenêtre ne sera pas créée.

2.2.2.2.2 Edition des fenêtres

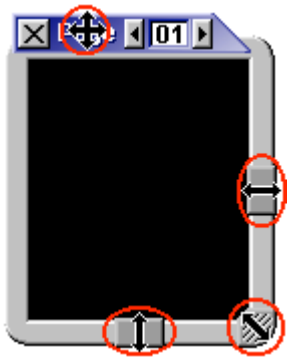

Figure 11 : Jeu de poignées de contrôle associées à leur curseur

Les fenêtres créées sont de taille minimale par défaut, il faut donc pouvoir régler leur encombrement afin qu'elles représentent le plus précisément possible les feuilles réelles.

Les 3 poignées de contrôle permettent de modifier à souhait les dimensions de la fenêtre en pointant et glissant. Le programme impose cependant les règles suivantes : les fenêtres ont une taille minimale, leurs cadres ne peuvent pas sortir de l'écran et il est impossible de les superposer.

En pointant sur le mot *page* de l'entête de la fenêtre, il est possible de la déplacer, toujours dans les mêmes limites décrites précédemment. Par contre, il reste envisageable de passer au-dessus d'une autre fenêtre en faisant glisser le pointeur au-delà et si la place à l'arrivée est suffisante.

2.2.2.2.3 Entête des fenêtres

L'entête des fenêtres présente l'icône classique de fermeture qui permet de détruire la fenêtre considérée. Il est également possible de régler précisément le numéro de la page prise en compte dans la fenêtre en le faisant défiler grâce aux flèches de contrôle. Ainsi, l'utilisateur n'est pas limité par la capacité maximale de la table dans le nombre de pages du texte à éditer : il est possible de poser des pages non-consécutives sur la table. Cependant, la limite d'affichage du compteur est fixée à 99 pages. Ceci ne nous semble pas du tout réducteur car l'utilisateur avisée prendra soin de découper son texte en plus petites unités. De plus, il peut s'avérer assez laborieux de faire défiler plus d'une trentaine de page dans le compteur.

Figure 12 : Entête des fenêtres

Ce compteur prend en compte les pages affichées dans les autres fenêtres de telle sorte qu'il est impossible de voir s'afficher une même page dans deux fenêtres différentes.

2.2.2.2.4 Reconnaissance automatique de la position des feuilles

Afin d'assister l'utilisateur dans cette phase laborieuse, une fonction de reconnaissance automatique de la postion des feuilles sur la table a été intégrée au logiciel. Elle est accessible, une fois le fichier texte chargé, par l'intermédiaire de l'icône « Caméra ». Cette reconnaissance repose sur la capture de l'état de la table par la caméra située au-dessus de celle-ci, puis sur l'analyse de l'image obtenue dans le but d'en faire ressortir la position des cadres des pages.

Figure 13 : Icône de positionnement automatique assisté par la caméra

Il faut donc prendre plusieurs précautions afin d'augmenter la précision du dispositif. Tout d'abord, il faut veiller à imprimer le texte en activant le cadre de chacune des pages, car c'est bien ce cadre qui est reconnu par le logiciel, et non pas le bord de la feuille. Ensuite, il faut éviter de traverser le champ de la caméra lors des captures et écarter tout objet inutile.

Le dipositif n'est pas capable de reconnaître le numéro des pages sur les feuilles. Par défaut, le logiciel affecte systématiquement le numéro 1 à la fenêtre située en haut à gauche, puis l'incrémente de gauche à droite, et de haut en bas suivant le sens conventionnel de lecture.

Il ne faut pas oublier qu'il est toujours possible de modifier le résultat de la capture le cas échéant en modifiant les numéros de page, redimensionnant, supprimant ou ajoutant des fenêtres.

2.2.2.3 Mode « Edition des sélections »

Maintenant que les feuilles sont positionnées et les fenêtres dimensionnées, nous pouvons nous occuper de l'édition des sélections. Pour cela, il suffit de pointer sur l'icône adéquate pour basculer dans le mode « Edition des sélections ». Remarquez que le mode précédent est toujours disponible.

Figure 14 : L'interface en mode édition des sélections

Il est important de souligner que les modifications de page, de position ou de dimensions des fenêtres ne peuvent se faire que dans le mode d'édition des feuilles. En effet, dans le mode d'édition des sélections, les poignées de contrôle des dimensions et les flèches du compteur disparaissent. De plus, l'entête n'est plus sensible au pointeur, bloquant le déplacement de la fenêtre. Cet état est représenté par une entête grisée.

Figure 15 : Fenêtre bloquée en mode édition des sélections

On limite ainsi les pointages intempestifs qui pourrait dérégler le positionnement des fenêtres. De plus, il est légitime de penser qu'une fois les réglages faits il est peu probable d'avoir à les modifier souvent.

Rappelons que la vocation de l'application est l'édition macroscopique d'un texte au moyen de manipulations basiques (copier, déplacer, supprimer). L'unité macroscopique retenue est le paragraphe. Le logiciel permet de sélectionner différents groupes de paragraphes. Ces groupes sont destinés à être supprimés, déplacés ou copiés vers d'autres endroits du texte. Comme on dispose du support physique immuable que sont les feuilles imprimées, ces opérations ne sont pas effectuées en temps réel. C'est un jeu de représentations graphiques qui nous en donne l'illusion : le logiciel surligne par projection les groupes de paragraphes sélectionnés, puis il affiche un fléchage explicite pour représenter les transformations effectuées.

2.2.2.3.1 Création de nouvelles sélections

Figure 16 : Icône de création de nouvelles sélections

Le basculement en mode « Edition des sélections » fait apparaître un nouvel icône. Il s'agit de l'icône « Nouvelle sélection ». C'est celui-ci qui permet de définir de nouvelles sélections. Une fois une nouvelle sélection créée, il est possible de lui affecter autant de paragraphes que souhaité en cliquant sur ceux-ci. Cependant, il faut se souvenir qu'un paragraphe ne peut faire partie que d'une seule sélection à la fois. Si la sélection éditée ne possède encore aucun paragraphe, cet icône n'a aucun effet, ce qui évite de multiplier les sélections vides.

2.2.2.3.2 Sélection des paragraphes

Une sélection regroupe un ensemble de paragraphes. S'il s'agit de la sélection courante, il est possible de lui affecter des paragraphes ou, au contraire, de lui en retirer. A sa création, une sélection devient systématiquement la sélection courante. Pour éditer une sélection ayant perdu le focus, il suffit de cliquer sur un paragraphe lui appartenant, ou encore de la choisir dans la liste des sélections présentée par le *Browser des sélections* (dont le fonctionnement sera décrit dans une section ultérieure).

Pour ajouter un paragraphe à la sélection courante, il suffit donc de cliquer dessus, dans la mesure où celui-ci n'appartient pas déjà à une autre sélection. Si la pression est maintenue, tous les paragraphes survolés n'appartenant à aucune autre sélection seront affectés à la sélection courante, et ce sans changer de sélection courante si des paragraphes appartenant à d'autres sélections sont survolés.

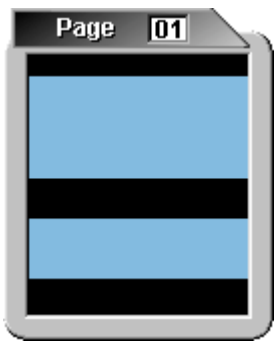

Figure 17 : Représentation d'une sélection

Pour retirer un paragraphe de la sélection courante, il suffit alors de recliquer dessus. De la même manière, si la pression est maintenue, tous les paragraphes survolés appartenant à la sélection courante seront déselectionnés, le fait de survoler un paragraphe appartenant à une autre sélection n'affectant pas le focus.

2.2.2.3.3 Affectation des actions

Une fois la sélection créée, il est possible de lui affecter une des trois actions disponibles : *déplacer*, *supprimer*, ou *copier* le groupe de paragraphes sélectionné. Il est également possible de revenir à l'état par défaut qui considère qu'aucune action n'est affectée à la sélection. Ces actions sont accessibles grâce aux quatre icônes correspondants qui apparaissent en bas à droite de l'écran dès qu'une sélection est éditée, ou encore grâce au menu en croix qui apparaît dès que la pression est maintenue en dehors d'une zone de texte.

L'icône « Aucune action » ne désigne pas une action à proprement parler. Il s'agit en fait d'une absence d'action. C'est l'état par défaut des nouvelles sélections. Il permet de mettre en évidence un groupe de paragraphes plus que d'éditer le texte. L'icône permet de revenir à un état par défaut dans le cas d'hésitations.

Figure 18 : Aucune action

Figure 19 : Déplacer

L'icône « Déplacer » permet comme son nom l'indique de déplacer le groupe de paragraphes sélectionnés. Il s'agit du condensé du couper-coller des traitements de textes classiques.

L'icône « Suppression » permet de supprimer le groupe de paragraphes sélectionnés. Il faut cependant se souvenir que si la sélection supprimée contient une ligne d'insertion d'une sélection qui lui est antérieure, les paragraphes déplacés par cette dernière seront également supprimés !!! On peut s'interroger sur l'utilité d'une telle action. Si le programme la prévoit, on peut s'imaginer qu'elle ne présente aucun intérêt pour l'utilisateur avisé.

Figure 20 : Suppression

Figure 21 : Copier

La copie sera assurément moins utilisée que le déplacement ou la suppression, mais elle est prévue. C'est à l'instar du déplacement un condensé du copier-coller des traitements de textes usuels.

Le menu en croix permet également d'accéder à ces quatre actions. Il se déroule lorsque une sélection est éditée et que la pression est maintenue en dehors d'une zone de texte, d'un icône ou d'un menu. Un fois complètement déroulé, le fait de lâcher le pointeur dans l'un de ses icônes active l'action correspondante. Lorsqu'il s'agit d'un *déplacement* ou d'une *copie*, l'icône correspondant en bas à droite de l'écran s'enfonce et la démarche à adopter est identique. L'accès aux différentes actions est donc plus rapide par le biais de ce menu. Celui affecte à chacune une direction dont il suffit de se souvenir.

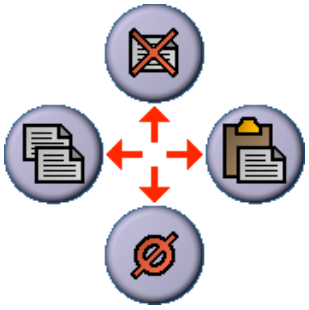

Figure 22 : Actions associées à leur direction respective

Les actions *copier* et *déplacer* demandent la désignation d'une ligne d'insertion. Celle-ci ne peut être placée qu'au début ou à la fin d'un paragraphe. Lorsque l'on désire manipuler une telle ligne, il faut cliquer sur l'icône de l'action correpondante. Celui-ci reste enfoncé tant que l'emplacement de la ligne n'a pas était indiqué. En cliquant sur la première moitié d'un paragraphe, la ligne d'insertion sera placée au début de ce paragraphe. Si la seconde moitié est pointée, l'insertion se fera à la fin du paragraphe. Le fait de maintenir la pression du pointeur permet alors de faire glisser la ligne d'insertion ainsi créée afin de faciliter son positionnement. Il reste possible d'annuler l'édition de la ligne d'insertion en cliquant en dehors d'une zone de texte ou à nouveau sur l'icône. Dans ce cas, le programme considère que l'action précédente reste valable. Comme aucune nouvelle ligne d'insertion n'a été désignée, l'ancienne ligne est conservée le cas échéant. Lorsque l'on fait glisser la ligne d'insertion, il est possible de se déplacer d'une fenêtre à l'autre sens annuler l'édition de la ligne. La ligne est alors définitivement positionnée lorsque le pression est relâchée, et ce quelque soit l'endroit où se trouve le pointeur à cet instant.

Lors de l'insertion d'un groupe de paragraphes non consécutifs, le logiciel les place les uns à la suite des autres à l'endroit indiqué et par ordre d'apparition dans le texte initial.

2.2.2.3.4 Modification de la couleur d'une sélection

Figure 23 : Icône de la palette des couleurs

Lors de leur création, le programme affecte par défaut une couleur de surlignage aux sélections. Celle de la sélection courante peut être modifiée à tout moment grâce aux icônes de la palette des couleurs. Ces derniers apparaissent quand l'icône « Palette des couleurs » est enfoncé et présent à l'écran. En effet, l'icône « Palette des couleurs » n'est accessible que lorsqu'une sélection est éditée.

Il existe 8 pots de couleurs. Pour affecter à la sélection courante la couleur contenue dans un pot, il suffit de le sélectionner. Les 7 premiers en partant du bas de l'écran contiennent des couleurs prédéfinies imposées par le logiciel. Le huitième pot contient une couleur pouvant être redéfinie par l'utilisateur grâce au menu situé juste au-dessus. Celui-ci indique les taux de rouge, de vert et de bleu contenus dans cette encre et donne la possibilité de les modifier. La couleur du huitième pot est propre à une sélection donnée et les réglages sont indépendants d'une sélection à l'autre. Les valeurs initiales des taux de la couleur contenue dans ce huitième pot sont fixées aléatoirement à la création de la sélection, de telle sorte qu'il y soit très peu probable de disposer de deux couleurs identiques pour deux sélections différentes, à moins d'avoir modifié manuellement les valeurs.

A la création d'une nouvelle sélection, le logiciel tente de lui affecter une couleur prédéfinie qui ne soit pas déjà utilisée par une autre sélection. Dans le cas contraire, il lui affectera la couleur du huitième pot.

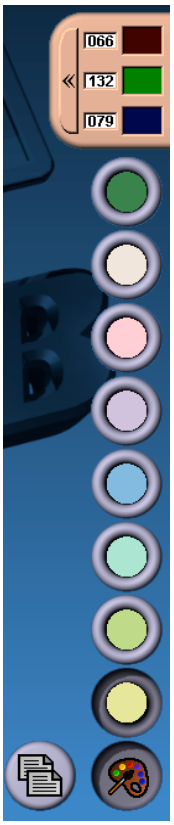

Figure 24 : Palette des couleurs

Le menu de réglage de la couleur du huitième pot peut être agrandi ou rétréci en cliquant sur le bouton longeant son bord gauche. Il permet de contrôler le triplet RGB (Rouge, Vert, Bleu) définissant la couleur de l'encre grâce aux curseurs qu'il est possible de dépacer sur les jauges (dans la version pleine de la fenêtre). Les chiffres indiquent la valeur des coefficients (compris entre 0 et 255), le petit cadre à droite de chaque compteur donne un aperçu de la couleur pure. Il n'y a aucune restriction sur le choix de la couleur, si ce n'est qu'il est conseillé à l'utilisateur d'éviter les teintes trop sombres, sans quoi la sélection pourrait être difficilement discernable sur le fond noir des fenêtres représentant les pages.

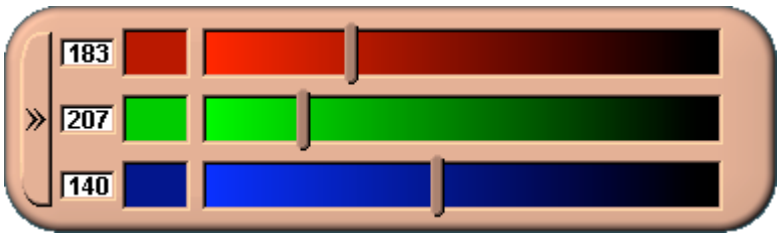

Figure 25 : Menu de réglage de la couleur du huitième pot

2.2.2.3.5 Browser des sélections

Le *browser* des sélections dresse l'inventaire des sélections. Il s'agit du menu en bas, au centre de l'écran. Il se déploit ou se range grâce au bouton situé le long de son bord supérieur.

Il permet de parcourir la liste des sélections grâce à l'ascenseur situé à gauche de la fenêtre de visualisation. Dans cette liste, chaque sélection apparaît avec la couleur de surlignage qui lui est affectée et l'action appliquée au groupe de paragraphes qu'elle représente. La sélection courante y est encadrée en bleu. Si elle ne contient aucun paragraphe, ses attibuts sont indiqués en bleu clair, sinon, ils sont affichés en blanc. Pour choisir la sélection courante, il suffit de cliquer sur la sélection souhaitée dans la liste.

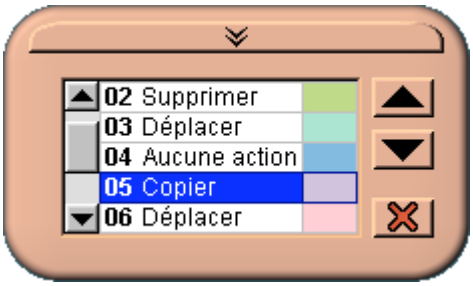

Figure 26 : Le *browser* des sélections

Il est important de remarquer que la liste des sélections présente un certain ordre. En effet, celui-ci joue un rôle lors d'insertions de paragraphes d'une sélection entre deux paragraphes consécutifs d'une autre sélection ou encore lorsque plusieurs sélections sont destinées à être insérées au même endroit (de plus amples détails seront donnés à ce sujet dans la section suivante). Les deux flêches à droite de la fenêtre de visualisation permettent de déplacer la sélection courante dans la liste. Une nouvelle sélection viendra systématiquement s'ajouter en fin de liste, de telle sorte que si aucune modification n'est apportée à la liste des sélections, celles-ci apparaitront dans l'ordre où elles ont été créées.

Enfin, il est possible de supprimer complétement une sélection de la liste grâce à la croix rouge située sous les flêches de déplacement. Attention, il n'est pas possible de récupérer les données concernant une sélection supprimée.

2.2.2.3.6 Ordre des sélections

Comme nous venons de le souligner, l'ordre des sélections à son importance. En effet, le logiciel traite les manipulations effectuées sur les sélections une à une, dans l'ordre où elles apparaissent dans la liste des sélections. Ainsi, si une première sélection insère des paragraphes entre deux paragraphes consécutifs appartenant à une sélection apparaissant plus tard dans la liste, les paragraphes insérés seront considérés comme appartenant à cette dernière sélection, et subiront donc le traitement réservé aux paragraphes de cette sélection. Dans le cas où les paragraphes sont insérés entre deux paragraphes consécutifs d'une sélection ayant déjà été traitée, ils ne subiront aucun nouveau traitement. De plus, il est permis de définir plusieurs lignes d'insertion au même endroit. Dans ce cas, les sélections seront insérées les unes à la suite des autres dans l'ordre indiqué dans la liste des sélections.

Tentons de mieux comprendre la logique d'actions imbriquées par un exemple simple. Il s'agit de considérer deux sélections, une verte et une bleue, la verte devant être déplacée et la bleue supprimée. La ligne d'insertion de la sélection verte se situe entre deux paragraphes consécutifs de la sélection bleue. Les paragraphes déplacés

seront traités de manière différente suivant la relation d'antériorité entre les deux sélections.

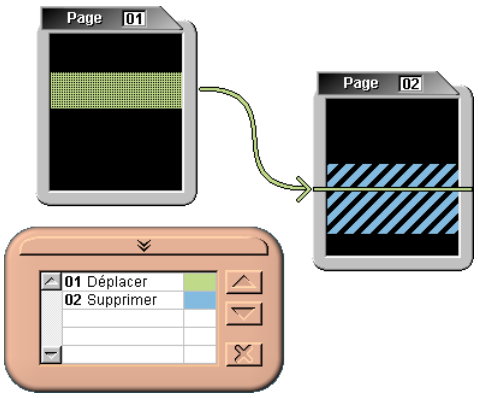

Figure 27 : Première situation d'antériorité

Considérons tout d'abord que la sélection verte soit traitée avant la sélection bleue. Dans ce cas, les paragraphes de la sélection verte vont d'abord être déplacés et insérés entre les deux paragraphes de la sélection bleue. Le programme considérera alors qu'ils font partie de la sélection bleue au moment de la traiter et ils seront alors effacés avec les autres paragraphes de la sélection bleue. Notez que la ligne d'insertion est pleine, ce qui signifie bien que la sélection bleue englobe les paragraphes insérés par la sélection verte.

Dans le cas inverse où la sélection bleue est traitée en premier, les paragraphes surlignés en bleu seront effacés d'abord, ensuite seulement les paragraphes de la sélection verte seront insérés à l'endroit occupé précédemment par les paragraphes supprimés. La ligne d'insertion est coupée par le surlignage bleu, indiquant que celui-ci ne la prend pas en compte.

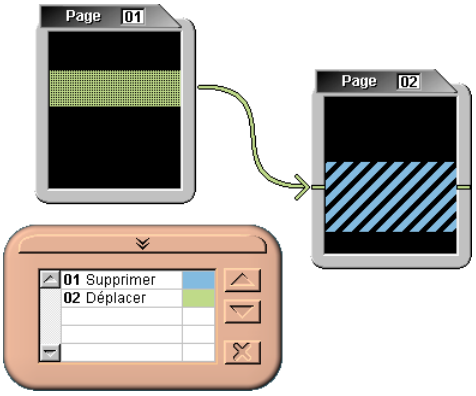

Figure 28 : Seconde situation d'antériorité

On peut remarquer que dans le premier cas, il aurait été plus efficace et pertinent de demander au logiciel d'effacer directement la sélection verte. Dans le second cas, il est possible d'éviter toute confusion en indiquant directement au programme d'insérer les paragraphes de la sélection verte avant ou après le groupe de paragraphes consécutifs considéré. Il ne s'agissait en fait ici que d'un simple exemple pour comprendre les mécanismes d'imbrication. Car ces règles d'antériorité sont valables pour toute action et notamment les actions *déplacer* et *copier*. Dans ces deux cas, certes plus complexes, on comprend mieux comment le logiciel réagira au vu de cet exemple.

2.2.2.3.7 Représentation des actions

Outre le fait d'être répertoriées dans le *browser* des sélections, les actions sont également représentées graphiquement par un motif de surlignage spécifique. De plus, les actions *déplacer* et *copier* sont agrémentées d'une flêche qui relie le groupe de paragraphes sélectionnés à la ligne d'insertion correspondante, lorsque les pages contenant ces éléments sont représentées dans une fenêtre.

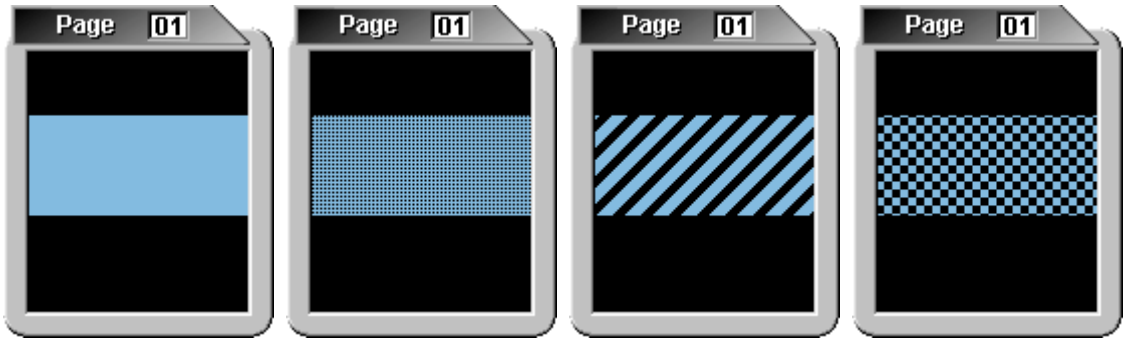

Figure 29 : Motifs de surlignage (de gauche à droite : *aucune action*, *déplacer*, *supprimer*, *copier*)

Ainsi les paragraphes sélectionnés ne subissant pas de modification sont surlignés de manière uniforme. Les paragraphes déplacés sont grisés. Ceux qui sont supprimés sont hachurés. Enfin, ceux qui sont copiés sont recouverts d'un damier.

2.3 Le formateur de texte

2.3.1 Principe : utilisation de classes abstraites

Dès le début, nous avions décidé d'utiliser des classes abstraites pour représenter les pages, les sélections et autres actions qu'un utilisateur pouvait faire. Ceci permettait donc de séparer l'interface et les classes réutilisables du niveau abstrait. Ainsi, nous avons par exemple aujourd'hui deux interfaces différentes qui peuvent cependant tourner sur les mêmes classes abstraites.

Nous avons ainsi élaboré deux niveaux d'abstraction:

- les classes abstraites en elles mêmes
- des fonctions globales d'accès à ces classes

Bien entendu, un programme peut directement accéder aux classes abstraites mais cela enlève un niveau d'encapsulation qui est prévu pour simplifier la vie de l'utilisateur. Nous allons brièvement décrire dans cette partie les différents composants de ces classes abstraites et de la façon dont elles ont été construites. Le rapport technique pourra fournir de plus amples informations ainsi que les annexes C et D. Cependant, les classes abstraites ont légèrement évolué par rapport au rapport technique pour intégrer de nouvelles contraintes et pour corriger certains bugs.

2.3.2 Les trois piliers de l'abstraction

L'abstraction est construite autour de trois classes: Cun_paragraph, Cune_selection, Cune_page. Ces trois classes représentent respectivement:

- la plus petite unité adressable par le programme (ici c'est un paragraphe mais rien n'empêche l'utilisation de ces classes en pouvant adresser un mot directement)
- un ensemble de Cun_paragraph sur lequel on effectue une action donnée
- un ensemble de Cun_paragraph qui permet leur localisation dans le document final. La classe Cune_page sert surtout à faire le lien entre la représentation réelle du document sous forme de page et sa représentation abstraite sous forme de suite de Cun_paragraph.

2.3.2.1 Principes communs

Les classes évoquées ci-dessus étant, par construction, abstraites, ne peuvent pas regrouper toutes les informations qu'on voudrait qu'elles regroupent. En effet, nous pourrions par exemple vouloir définir la couleur d'une sélection (c'est donc un aspect visuel qui n'a rien à faire dans la couche abstraite d'une sélection mais qu'il est cependant commode de stocker directement dans la selection). Pour remédier à ce probleme, nous avons construit chacune des classes comme un modèle (template). Ainsi, il faut spécifier le type (qui peut donc etre une classe ou une structure que nous avons construite) du plugin qu'on rajoute à ces classes. Ainsi, la classe Cune_selection aura par exemple une structure qui contient la couleur de la sélection tandis que la classe Cun_paragraph n'aura rien de plus (type void).

Les différentes classes se contiennent les unes et les autres. Ainsi, par exemple, une page est composée de paragraphes et des paragraphes composent aussi les selections. Cependant, les paragraphes peuvent aussi appartenir à plusieurs selections (cette possibilité n'est pas autorisée dans l'interface graphique mais elle est prévue dans les classes abstraites). Chaque classe contient donc des listes de pointeurs vers les objets contenus mais aussi vers les contenants si cela s'applique pour permettre un accès plus rapide dans les deux sens. Nous avons utilisé le type set de la STL que nous avons amélioré en réécrivant une fonction erase qui prend en argument ce qu'on veut effacer et pas un itérateur de ce qu'on efface. L'utilisation d'un set permet d'avoir un ordre sur les listes. Ceci permet ainsi par exemple de ne désigner qu'une seule flèche pour l'insertion de paragraphes consécutifs.

2.3.2.2 La classe Cune_page

La classe Cune_page contient une liste des paragraphes qui composent la page ainsi qu'une liste des selections dont le point d'insertion est dans la page. La fonction principale de cette page est de faire le lien entre les coordonnées relatives d'un paragraphe ou d'un point d'insertion par rapport à la page et les coordonnées absolues. De plus lors d'un re-dimmensionnement des pages, cette classe s'occupe de mettre à jour tous les paragraphes et les points d'insertion pour que les coordonnées relatives et absolues soient bien en concordance. Cette classe est donc une classe un peu à cheval entre les représentations abstraites et le monde réel qu'on représente.

2.3.2.2.1 La notion de virtualité

Un point important à noter est l'existence de pages virtuelles. Une page virtuelle est une page qui existe mais qui n'est pas affichée. Cette notion est très importante car de part l'implémentation un paragraphe ne peut pas exister sans la (ou les pages) qui le contiennent. En effet, un paragraphe a besoin de la page pour pouvoir calculer ses coordonnées relatives (ou absolues) en fonction de la position de la page. Une page virtuelle est donc une page qui n'est plus affichée (soit parce que l'utilisateur ne l'a pas encore créée visuellement soit parce qu'il l'avait créée mais supprimée ensuite. La notion de page virtuelle s'étend ensuite aux paragraphes et aux sélections. La virtualité d'un paragraphe est déterminée par la virtualité des pages contenant les deux points extrêmes du paragraphe et la virtualité d'une sélection est déterminée par la virtualité de la page contenant son eventuel point d'insertion. La notion de virtualité permet de savoir quelles sont les coordonnées auxquelles on peut faire confiance. Par exemple, pour un paragraphe virtuel, les coordonnées relatives (c'est à dire les pourcentages par rapport à la taille totale de la page) sont les données intéressantes; les valeurs des coordonnées absolues ne sont pas fiables car la page contenant le paragraphe n'est pas dessinée a l'écran et n'existe donc pas vraiment.

La notion de virtualité permet aussi de comprendre ce qu'est une destruction de page au niveau abstrait. Lorsqu'on décide d'enlever une page du champ visuel dans la deuxième interface, on ne va pas détruire cette feuille mais on va la rendre virtuelle. Ceci ne libère donc pas l'espace mémoire réservé pour la page mais permet de continuer à considérer la page comme existante. Ainsi, toutes les selections de cette page seront conservées et si on la refait apparaître, elles seront intactes. Une page abstraite ne signifie donc pas forcément une page visible à l'écran. En fait, lors du chargement d'un fichier Word au démarrage du programme, toutes les pages sont créées en mémoire ainsi que tous les paragraphes; ils sont tous virtuels.

2.3.2.2.2 Le plugin pour Cune_page

Dans la nouvelle interface. la classe Cune_page se dote d'un plugin qui permet de savoir si les différents boutons ont été appuyés (le bouton fermeture par exemple ou le bouton augmentation du numéro de page). L'ancienne interface n'a pas besoin de toutes ces données supplémentaires et le plugin est donc un void.

2.3.2.3 La classe Cune_selection

Une sélection est représentée de manière abstraite comme:

- un ensemble quelconque de Cun_paragraph (ils peuvent être contigus ou pas)
- une action (qu'est ce qu'on fait avec les paragraphes)
- un point supplémentaire éventuel (par exemple, si on fait une insertion, le point d'insertion)

Ceci laisse ainsi beaucoup de possibilités pour une sélection. Les actions d'une sélection sont extensibles à l'infini (il suffit de définir d'autres actions dans l'énumération des actions). Cependant, l'implémentation de cette classe n'est pas encore parfaite et pourrait être améliorée. En effet, le paramètre supplémentaire (à savoir le point d'insertion) n'est présent que pour les actions COPY et DEPLACEMENT. Or ceci est codé en dur dans le code de la classe. Il faudrait pouvoir rajouter des actions demandant un paramètre supplémentaire sans avoir à modifier le code de la classe. C'est une des améliorations possibles du code.

La liste des sélections est la base de tout le programme. En effet, cette liste (qui est ordonnée) va permettre de produire le fichier final en analysant les sélections et en interprétant leur contenu. Il est important de noter que la liste est ordonnée ce qui permet donc une hiérarchisation des actions. En effet, l'ordre est très important car il permet de savoir comment les actions vont se faire. Ceci est traité lors de la description de l'interface. (voir Ordre des sélections)

2.3.2.4 La classe Cun_paragraph

Cette classe symbolise la plus petite entité considérée par le programme. Son nom, Cun_paragraph vient du fait que nous considérons l'analyse du document au niveau du paragraphe mais il pourrait très bien s'agir d'un mot. Toute l'implémentation est en place pour traiter un objet de taille quelconque en deux dimensions. Il suffit pour cela qu'il y ait un ordre (qu'on puisse donc compter les objets) et qu'ils soient en deux dimensions. Le programme s'adapterait parfaitement. Ceci démontre bien que l'abstraction des objets réels utilisés peut être utile dans un cas pareil: l'objet réel change (en l'occurence on passe d'un paragraphe à un mot) mais sa fonction ne changeant pas, sa virtualisation peut ne pas changer aussi.

2.3.3 Conclusion

En conclusion, nous pouvons dire que la séparation en modules et l'abstraction des structures a permis de greffer deux différentes interfaces au même programme. La deuxième interface, certes plus complexe et demandant donc plus de fonctionnalitées aux classes abstraites, a certes demandé certaines modifications. Elles auraient pu être évitées si une réflexion plus approfondie avait été menée au début sur les fonctions souhaitées du programme. Par exemple, avec la première interface, une sélection était toujours une suite continue de paragraphes. Ceci a dû être modifié pour la deuxième interface pour prendre en compte des sélections de paragraphes non consécutifs. Cependant, les appels de fonctions de la première interface sont restés globalement fonctionnels et à l'heure actuelle, les deux interfaces fonctionnent parfaitement sur les mêmes classes abstraites.

Cependant, plusieurs améliorations peuvent encore être apportées au programme et particulièrement aux classes abstraites. Par exemple, la destruction et la création des instances de chaque classe laisse à désirer. Il n'est en effet pas encore tout à fait établi qui doit detruire quoi et qu'est ce qui peut subsister sans quoi. Pour le moment, la solution que nous avons retenue est de tout détruire à la fin du programme en parcourant les entités contenant les pointeurs vers toutes les instances créées. Cependant, les destructeurs de chacune des classes abstraites laissent aussi à désirer. Que doivent-ils detruire exactement? Pour le moment, seuls les plugins sont détruits. Ceci semble être la meilleure solution car c'est la seule chose créée directement par la classe. Cependant, est-ce qu'une page peut survivre si elle n'a plus d'utilité. Le langage C n'est pas très bien construit pour ce genre de considérations. En effet, il ne prend pas en compte le nombre d'objets qui font référence à un autre. On aurait pu implémenter un système dans ce sens où une page serait détruite si elle n'a plus d'utilité. Les analyses de fuites de mémoire laissent cependant à supposer que toute la mémoire réservée est bien libérée à la fin du programme.

2.4 La caméra

2.4.1 But et description générale du système associé à la caméra

L'interface que nous avons implémentée ne présente d'intérêt que si elle est suffisamment pratique d'utilisation. Or, comme nous l'avons décrit précédemment, le pointage des feuilles est une étape assez fastidieuse car l'utilisateur est obligé de redimensionner chaque fenêtre de feuille. C'est l'opération qui prend le plus de temps. Pour pallier cet inconvénient, nous avons ajouté une caméra au dispositif pour assurer un premier positionnement automatique des feuilles quitte à permettre ensuite à l'utilisateur d'apporter des corrections légères si nécessaire.
Une caméra directionnelle a donc été fixée au-dessus de la table, de manière à avoir la plus grande surface de la table dans son champ de vision. Ces réglages «faits maison» ont posé d'ailleurs certains problèmes de précision dont nous reparlerons par la suite.
Avant toute description de l'usage de la caméra, il faut rappeler un point important qui est la nécessité de la calibrer. En effet, la caméra, de par son installation physique, ne peut visualiser la totalité de la zone affichée par le projecteur. Lors d'une capture, un point n'aura donc pas les mêmes coordonnées relativement à l'image enregistrée et à la fenêtre de l'application, comme le montre le schéma suivant :

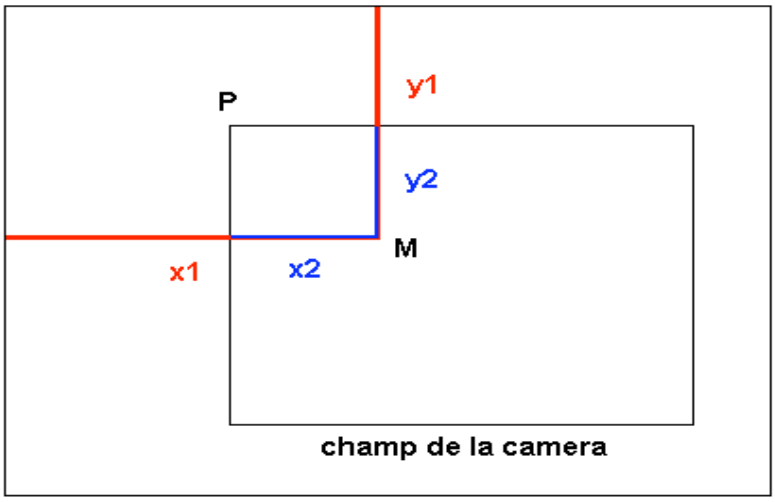

Figure 30

(x1, y1) sont les coordonnées de M par rapport à la fenêtre de l'application.
(x2, y2) sont les coordonnées de M par rapport à l'image capturée par la caméra.

L'utilisation de la caméra se fait donc en deux étapes.
Tout d'abord, au début du programme et avant que l'interface ne soit affichée, on effectue le calibrage de la caméra. Pour cela, un contour rectangulaire noir est affiché sur un fond rouge et la caméra effectue une capture (cette capture se fait par un exécutable qui utilise un ActiveX permettant de manipuler un flot d'images). L'image ainsi enregistrée est alors soumise à l'algorithme de reconnaissance de rectangle (voir Romain pour les détails techniques). Les coordonnées du coin supérieur gauche sont calculées et comparées aux coordonnées du même point mais relativement à la fenêtre de l'application (qui sont celles imposées dans le code pour afficher le contour). Par différence, nous obtenons ainsi la correction à apporter en x et y entre ce que la caméra voit et ce qui est réellement placé sur la table. (en fait les coordonnées du point P sur le schéma précédent).

L'interface est alors affichée et le programme est utilisable comme dans sa version précédente. Cependant, en mode d'édition de feuille, une pression sur le bouton « appareil photo » provoque l'affichage d'un fond rouge (l'interface est alors momentanément masquée et l'utilisateur ne doit surtout pas masquer une partie de la table !). La caméra effectue une nouvelle capture et les feuilles (imprimées avec une bordure noire de 6 points au préalable) sont alors repérées sur l'image obtenue. En tenant compte de la correction calculée au début du programme, les fenêtres de feuille peuvent être affichées toutes en même temps et à la bonne place. L'utilisateur peut toujours modifier ces fenêtres.

En théorie, l'utilisateur n'a donc plus besoin de pointer les feuilles ce qui signifie un gain de temps et une économie d'énergie musculaire non négligeable. Malheureusement, ce n'est pas le cas pour une simple raison : la qualité de la

caméra et donc de l'image issue de la capture. En effet, le code est parfaitement opérationnel comme nous avons pu le vérifier avec une analyse de BMP contenant des contours de feuilles simulés par des rectangles sur un fond unicolore. Avec ce genre d'image idéale (tant en netteté qu'en luminosité), toutes les feuilles sont reconnues et l'algorithme ne fait qu'une erreur de deux ou pixels sur la position des coins ce qui est convenable quand on pense que la résolution de la table ne permet de descendre en dessous de cette précision.

Le problème est que les images capturées par la caméra ne sont pas d'une telle qualité bien au contraire : les variations de luminosité, la réflexion sur la table blanche ne permettent pas toujours de faire ressortir les bordures noires du fond rouge. La solution serait alors d'épaissir encore plus ces dernières mais Word ne le permet pas et on comprend pourquoi : l'utilisateur doit pouvoir écrire quelques mots entre ces bordures ! Un bon positionnement de la caméra permettrait d'éviter la distorsion de l'image mais le dispositif n'étant pas fixe, il nous a été impossible de la placer de manière définitive : chaque utilisation a donc entraîné des écarts assez aléatoires et des images de plus ou moins bonne qualité!

Une deuxième raison qui pourrait empêcher la reconnaissance des feuilles par l'algorithme est due au code lui-même mais peut être évitée : nous avons défini une taille de feuille en pixels sous forme de constantes dans le code. Hors ces valeurs dépendent du zoom du projecteur et donc de l'installation. Nous n'avons pas eu le temps de créer un fichier de configurations (.ini) modifiable par l'utilisateur. C'est la solution la plus rapide à nos yeux pour adapter les paramètres du programme au dispositif physique. Toujours est-il que notre dispositif ne fonctionne pas toujours, c'est d'ailleurs aussi pour cela que nous avons préféré ne pas supprimer le pointage manuel. Voyons de manière plus précise comment cette reconnaissance des feuilles par la caméra est possible.

2.4.2 Le code qui se cache derrière la caméra

La reconnaissance de formes de la caméra se base sur la librairie OpenCV. Cependant, cette librairie ne nous permet pas de détecter les pages de façon correcte car plusieurs rectangles vont être détectés pour une même page. Il faut donc ensuite les filtrer pour pouvoir mettre les rectangles qui correspondent à la même page ensemble. Dans le principe, nous allons donc effectuer une moyenne sur les différents rectangles correspondants à une même page pour déterminer les coordonnées de la page en question. L'algorithme utilisé est assez basique et n'est aucunement optimisé pour une grande vitesse. Ceci n'a pas trop d'importance car le temps d'acquisition de la caméra est déjà relativement long; la différence en terme d'efficacité et de gain de temps pour l'utilisateur serait ainsi négligeable. De plus, cette procédure n'est appelée qu'une seule fois ce qui réduit l'intérêt de son optimisation. Il est en effet préférable d'optimiser des procédures fréquentes pour avoir le plus grand gain de temps possible.

2.4.2.1 Quel est son principe ?

Pour reconnaître si des rectangles détectés par la librairie OpenCV représentent une même feuille, on détecte tout simplement si les rectangles sont proches en calculant la distance entre leurs sommets respectifs. Le critère de proximité des rectangles peut être réglé. Il est ainsi possible de classer les rectangles trouvés.

On va en fait parcourir un à un les rectangles. Lorsqu'on trouve un rectangle non déjà traité, on va ensuite rechercher tous les autres rectangles qui sont "proches" (au sens défini ci-dessus) et les traiter - c'est à dire qu'on va faire une moyenne, pour chaque point, en prenant chaque rectangle. Il apparaît, après des tests, qu'en moyenne, la librairie trouve une cinquantaine de rectangles pour une page. Ce chiffre varie beaucoup en fonction de la définition de l'image, de sa netteté et de l'épaisseur du trait contour de la feuille.

Ainsi, pour résumer le fonctionnement de l'algorithme, on va:

1) Parcourir la liste des rectangles trouvés pour prendre un rectangle non traité;

2) Rechercher ensuite tous les rectangles proches et calculer pour chaque point du rectangle une moyenne en prenant en compte tous les rectangles trouvés;

3) On a donc trouvé une page qu'on stocke. On efface ensuite de la liste des rectangles tous les rectangles traités car les recherches ultérieures seront plus rapides;

4) On re-boucle sur l'étape 1 jusqu'à ce qu'on ait traité tous les rectangles.

2.4.2.2 Les difficultés rencontrées

L'algorithme en lui même est donc très simple, son explication est d'ailleurs très courte. Cependant plusieurs problèmes sont apparus et il a fallu les traiter avec soin.

2.4.2.2.1 L'ordre des points

La librairie OpenCV ne retourne pas une liste de rectangles (après avoir crée par exemple une structure représentant les rectangles) mais plutôt une liste de points. Un rectangle est donc formé de quatre points consécutifs dans la liste. Ceci pose plusieurs problèmes évidents:

- Il faut faire attention à où on se situe dans le tableau et ne pas prendre des points de deux rectangles;
- Le sens de parcours des rectangles n'est pas toujours le même;
- Le premier des quatre points représentant un rectangle dans la liste ne correspond pas toujours au même point physique (à savoir par exemple le coin en haut à gauche).

Ces différents problèmes nous ont donc conduits à une complexité accrue de l'algorithme car il fallait traiter tous les différents cas. Veuillez vous reportez en annexe pour plus de détails sur l'algorithme (annexe C)

2.4.2.2.2 La validation des pages

Une fois qu'on pense avoir trouvé une page, le programme effectue quand même une vérification ensuite pour s'assurer que la page trouvée est bien une page et pas le cadre projeté ou une autre boite de dialogue qui se serait inopinément glissé sur l'écran au moment de la capture. Ceci se fait tout simplement par une vérification des dimensions de la page trouvée . Il suffit pour cela de régler des

paramètres définissant la taille d'une page normale. Le programme vérifiera ensuite que l'objet trouvé a une taille s'approchant bien de la taille théorique.

2.4.2.2.3 Ordonnancement des pages

Un autre gros problème qu'il a fallut surmonter est l'ordonnancement des pages. En effet, il n'y aucune raison pour que les rectangles sortis par OpenCV soient dans un ordre de lecture habituel. L'ordre des pages en sortie sera donc complètement aléatoire ce qui ne va pas du tout. Pour ordonnancer les pages, nous avons choisi d'utiliser une liste triée en surchargeant la classe less<structure de page>. Ceci nous permet ainsi de classer automatiquement (la liste le fait directement dès l'ajout d'un élément) les pages trouvées en fonction de leur position sur l'écran. Cette solution permet donc d'avoir une disposition habituelle des pages à l'exécution de la reconnaissance des pages.

2.4.2.3 Conclusion

En conclusion, l'algorithme de détection des pages à partir des rectangles fournis par OpenCV se fait de manière fonctionnelle mais non optimisée. A la sortie des appels des fonctions (voir annexe C pour plus d'informations sur les fonctions spécifiques), un tableau contient des structures pages trouvées. Il suffit ensuite de re-injecter ces pages dans le programme principal. Pour cela, on va appeler la fonction faire_operation (cf. annexe C) en lui demander de créer des nouvelles pages. Les représentations abstraites de ces pages seront donc rentrées en mémoire et l'interface graphique va pouvoir se baser dessus pour les dessiner.

3 Les tests (expérience de validation)

3.1 Objectifs

Une fois que nous avons eu une version utilisable de notre logiciel, dans laquelle toutes les fonctionnalités souhaitées avaient été implémentées, avec une interface graphique améliorée, il nous a paru nécessaire d'organiser une procédure de tests sur des volontaires extérieurs au projet. En effet, on sait qu'un code testé et retesté par ses développeurs montre souvent des bugs lors d'une utilisation par un utilisateur extérieur : proposer à de telles personnes de travailler sur notre logiciel est une manière de s'assurer qu'il est utilisable. Mais avant tout, notre projet visait à améliorer le travail collaboratif en proposant un dispositif de traitement de texte intuitif et facile d'utilisation : nous espérions vérifier que notre objectif avait été atteint.

Pour cela, nous avons construit un protocole de test mettant les volontaires, élèves de l'Ecole Centrale, dans une situation de travail collaboratif, et nous avons observé leur utilisation du dispositif pour déterminer si celle-ci était agréable, et si elle améliorait le travail collaboratif par rapport au travail sur un simple ordinateur.

3.2 Protocole

Les tests, d'une durée d'une quarantaire de minutes, se déroulent au laboratoire ICTT, en présence d'une équipe de trois volontaires et d'un membre observateur du projet TableGate. Un protocole de test, qui a été modifié depuis la première version qui a été présentée dans le Rapport Intermédiaire de septembre 2003, a été rédigé à l'intention de ce membre observateur, afin de s'assurer que chaque équipe de volontaires ait reçu les mêmes instructions (voir celles-ci en annexe). Un test se compose de 4 grandes parties :

- Une présentation orale du dispositif : projecteur, table tactile, introduction de la notion de réalité augmentée ; suivie d'une présentation rapide du principe de fonctionnement du logiciel : impression d'un texte Word présent dans l'ordinateur, logiciel de traitement de texte où l'unité de travail est le paragraphe. A noter qu'à la fin du test, de nombreuses équipes ont demandé des renseignements supplémentaires (voir l'analyse des résultats), d'où un retour fréquent à ces explications en fin de test.

- Une démonstration du travail avec le logiciel : après avoir expliqué les étapes à suivre lors de l'utilisation du logiciel (placement des feuilles, sélection des zones à modifier, opérations, compilation, réimpression), le membre du projet présent utilise un texte de type JP pour faire une démonstration de ce travail. Pour rappel, un texte de type JP (voir le rapport intermédiaire) est un texte contenant des paragraphes dans le désordre et des parties superflues, mais tel que l'ordre d'origine soit évident. Nous avons utilisé des paragraphes comportant l'un des nombres de un à douze, ou un mot ressemblant à l'un de ces chiffres, répété sur quelques lignes. Ainsi, le travail sur un texte de ce type n'entraîne aucune réflexion quant au texte lui-même, mais permet d'utiliser le logiciel pour une édition de paragraphes. Après cette démonstration, les volontaires sont invités à tester les différentes opérations (placement, sélection, opérations) et fonctionnalités (menu déroulant, browser) par eux-même : ils peuvent alors poser toutes sortes de questions techniques au

membre du projet. C'est la phase pendant laquelle ils apprennent à manipuler le logiciel, pendant 10 à 15 minutes.

- Le test proprement dit, où les volontaires doivent effectuer un travail seuls, et sont observés. Pour se mettre dans une situation de travail collaboratif, nous avons conçu un texte de 4 pages où alternent des morceaux d'un sketch de Bigard découpé en paragraphes et mélangés, et des parties « parasites », des textes sur la réalité augmentée ou des passages d'une nouvelle de Maupassant par exemple. On donne à lire à chaque volontaire une page et demie de ce texte en leur précisant dans une consigne écrite le type et le thème du texte d'origine, ainsi que le travail que l'on attend d'eux : retrouver le texte initial. Ils se trouvent ainsi dans une situation proche de l'édition commune d'un rapport écrit à plusieurs : chacun a écrit une partie et n'a pas lue celle des autres, mais en connaît tout de même le thème. Lors de cette étape, le membre du projet est présent pour répondre à toute question technique sur la manipulation, il note également ses remarques dans une fiche d'observation.
Utiliser une fiche d'observations plutôt que noter des remarques sur une page blanche a plusieurs avantages. D'une part, elle facilite la prise de notes et permet de ne rien oublier ; d'autre part, en étant assez précise (chaque point à noter ne requiert qu'un mot ou un symbole), elle assure que plusieurs membres observateurs pensent bien à la même chose lorsqu'ils notent une remarque, ce qui permet ensuite de dépouiller ces fiches d'observations comme un ensemble et d'en extraire des statisques (élémentaires au vu du petit nombre d'observations).

Nous avons partagé notre fiche en plusieurs parties correspondant à plusieurs aspects du travail sur le dispositif que nous souhaitions observer. Tout d'abord, nous y notons les caractéristiques de l'équipe pouvant avoir une influence sur leur travail : leur expérience en informatique, leur expérience du travail collaboratif et le fait qu'ils se connaissent ou non. Ensuite, nous notons le temps qui leur a été nécessaire pour effectuer le travail demandé, et si le texte s'est bien retrouvé dans l'ordre voulu à l'issue de la manipulation. Des remarques sur l'utilisation du dispositif de la table interactive (positionnement autour de la table, mobilité, gêne/accès aux fonctions, manipulation ou annotations sur les feuilles de papier réelles) et sur l'utilisation du logiciel WordTab (manipulation aisée des différentes fonctions, erreurs fréquentes) ont également été prévues. Enfin, nous avons voulu nous intéresser plus précisement au travail collaboratif dans cette situation, en notant la méthodologie employée pour le travail (actions effectuées en premier, en dernier) et surtout des remarques sur le partage des tâches (immédiat, maintenu, source de conflits ou non), la communication dans l'équipe, les accidents de manipulation dûs à des opérations simultanées et les éventuels énervements ou autres problèmes survenus au sein du groupe.

- Enfin, un questionnaire à remplir par les volontaires : celui-ci se compose de huit questions, sous forme de cases à cocher ou de lignes de réponses. Les premières questions portent sur l'utilisation du logiciel, le temps estimé d'apprentissage et le temps d'amortissement. D'autres questions plus ouvertes demandent au volontaire l'intérêt qu'il a vu au logiciel, les utilisations pour lesquelles on pourrait l'adapter, ou tout autre commentaire ou suggestion.

Parallèlement à ces tests sur la table, le protocole prévoit que quelques équipes effectuent le travail de la phase trois (remettre dans l'ordre le texte de 4 pages) munis uniquement d'un ordinateur : le texte est le même, les utilisateurs ont

également une connaissance partielle du texte. Pour ces tests, une observation systématique n'est pas prévue, car cette situation de travail nous est bien connue, et étudier le travail sur ordinateur n'est pas l'objectif de nos tests. On note en revanche le temps mis pour remettre le texte dans l'ordre, que l'on compare au temps mis par les volontaires passant sur le dispositif TableGate, ainsi qu'au temps qu'ils estiment avoir gagné en utilisant notre logiciel (cette question leur est posée dans le questionnaire).

3.3 Déroulement des tests

Nous avions initialement prévu de faire passer entre 5 et 10 équipes de 3 personnes sur le dispositif. Une première équipe « test » à inauguré la procédure à la mi-octobre, à la suite de quoi nous avons précisé notre questionnaire et notre fiche d'observation. En effet, nous avons alors eu une idée plus précise de ce qu'il serait pertinent d'observer ; d'autre part le questionnaire (sous sa forme initiale) qu'ils ont rempli et les questions qu'ils nous ont posées nous ont montré sous quelle forme notre questionnaire nous apporterait des éclairages intéressants sur l'impression qu'ont eue nos sujets de WordTab : ainsi, c'est par exemple à la suite de ce premier test que nous avons ajouté les questions concernant l'intérêt d'utiliser des feuilles papier ou sur les applications possibles du logiciel.

Nous avons recruté nos volontaires parmi les élèves de l'Ecole, en essayant de former des équipes de types distincts : des équipes ayant une grande expérience de l'informatique (membres d'Eclair, élèves ayant un projet d'étude en informatique), d'autres en ayant un peu moins (élèves de première année sans intérêt particulier pour l'informatique – en gardant à l'esprit le fait que trouver quelqu'un n'ayant jamais manipulé un traitement de texte parmi cette population est difficile), des équipes ayant déjà rédigé des rapports ensemble, des équipes dont les membres se connaissent, d'autres non... Ainsi, cinq équipes se sont succédées (pour un total de 13 élèves au lieu de 15 en raison de problèmes d'organisation), et puisque toutes ont mis des temps comparables et fait des remarques similaires, quelle que soit de leur expérience, ce chiffre nous a semblé suffisant.

3.4 Analyse des résultats

Au niveau du temps, les équipes ont mis entre 2 et 3 minutes pour placer les feuilles, entre 9 et 13 minutes pour effectuer les modifications sur le texte. On a pu remarquer que l'équipe ayant mis le plus de temps est celle qui était le plus expérimentée à la fois en informatique et en travail collaboratif, et que l'équipe la plus rapide était composée d'élèves qui n'avaient jamais travaillé en commun sur un texte, ne se connaissaient pas avant le test, et semblaient assez timides. Puisqu'il ne s'agit que de quelques personnes, on se gardera d'en déduire que l'utilisation de WordTab est la plus facile pour les personnes ne maîtrisant pas les traitements de textes habituels, mais on en retiendra déjà que notre logiciel est assez simple d'utilisation pour que beaucoup de personnes, plus ou moins expérimentées, puissent s'en servir.

Travail d'une "équipe type" :

Tous les tests ayant donné des résultats similaires, on peut dégager de ces tests les caractéristiques du travail d'une « équipe type ». Aucune équipe n'a eu

exactement le comportement décrit ci-dessous, mais tous les tests ont fortement ressemblé à ce qui suit. Lors du positionnement des feuilles, les sujets placent sur la table la ou les pages qu'ils ont eues à lire, puis se positionnent devant. (Ce positionnement - que nous avons noté « propriétaire » dans la fiche d'observation - semble naturel, mais on aurait pu imaginer qu'afin d'être moins serrés, l'élève ayant lu la première partie du texte ce soit par exemple placé sur le côté gauche de la table). Les volontaires n'ont pas de problèmes majeurs pour accéder aux fonctions, s'aidant à appuyer sur tel ou tel bouton ; ni pour lire le texte, opérant des translations d'un côté à l'autre de la table au cours du travail. L'équipe n'utilise pas les avantages de la réalité augmentée (annotations sur les feuilles, déplacement de celles-ci), et ses réponses au questionnaire ont montré qu'elle n'en avait pas bien compris le principe : pourquoi on utilisait du papier plutôt qu'une projection directe du texte sur la table, et les avantages que l'on pouvait en tirer. La méthode de travail employée est de supprimer en premier lieu les paragraphes superflus, puis de rechercher dans l'ordre le premier paragraphe, puis le deuxième... suivant une rédaction complètement linéaire. Au cours du travail, les volontaires se font de très brefs résumés de ce qu'ils ont lu, mais sont tout de même amenés à survoler des paragraphes n'appartenant pas au texte qu'ils ont eu à lire, ce qui prend du temps, mais sans les lire entièrement. En ce sens, le transfert d'informations pertinentes dans l'équipe est un peu faible, en revanche la communication est importante. Un bon équilibre et un partage des tâches est trouvé dès le départ, les accidents de manipulation sont peu nombreux, il n'y a pas de conflits dans l'équipe. Le travail se déroule dans une bonne ambiance.

Remarques issues des observations :

En ce qui concerne le positionnement, les mouvements, l'accès aux fonctions, l'utilisation du principe de la réalité augmentée, les sujets ont tous suivi ce qui a été décrit pour l'équipe type: aucun n'a effectué de permutation avec l'un de ses voisins, aucun n'a soulevé une feuille, aucun n'a eu de difficultés particulières à accéder à un bouton, aucun n'a écrit sur une feuille (alors que les consignes qui leur ont été distribuées précisaient qu'ils pouvaient faire ce qu'ils voulaient avec leur copie du texte, et que des stylos leur étaient fournis).

L'utilisation du logiciel a été plus diversifiée selon les sujets : certains ont utilisé quasi-exclusivement les icônes, d'autres ont préféré le menu déroulant. Tous ont compris le rôle du browser, mais toutes, sauf une, ont fait appel au membre observateur de TableGate pour des précisions quant à son utilisation lorsqu'ils ont voulu s'en servir. Les sélections dites "avancées", c'est-à-dire le fait d'effectuer la même opération (par exemple la suppression) sur plusieurs paragraphes à la fois, ont été utilisées au moins une fois par équipe. Les seules moments où les équipes ont fait appel au membre observateur, en plus de l'utilisation du browser, sont les occasions où plusieurs flèches se sont retrouvées à pointer vers le même endroit : il est vrai qu'il faut alors se rappeler quelle opération a été faite en premier (passer par le browser est également possible), ce qui est très différent des situations habituelles où les modifications sont faites en temps réel, permettant une vérification immédiate du travail effectué. Cependant, le fait de ne pas avoir sous les yeux le texte final, à l'issue du travail, n'a pas gêné les sujets : aucune équipe n'a oublié de partie de texte à cause de cela et les seules erreurs dans les textes modifiés ont été dues à des erreurs de compréhension.

Une seule équipe n'a pas commencé son travail en supprimant les paragraphes parasites, mais a quand même effectué un travail entièrement linéaire (comme toutes les autres), faisant toutes les suppressions en dernier. Il est intéressant de noter cet attachement à une rédaction linéaire alors que WordTab permet justement d'autres méthodologies peut-être plus efficaces : ayant tout le texte sous les yeux, on peut tout à fait choisir de regrouper d'abord des paragraphes qui semblent se suivre, ou vouloir commencer par la fin. Lors de la présentation du logiciel, le membre de TableGate prenait pourtant le soin de montrer que c'était possible, mais cet aspect du logiciel n'a pas été assimilé par les sujets. On a même vu quelques équipes reconnaître très vite le paragraphe final (ce qui était assez simple puique le texte retenu, un texte comique, comprenait une chute facilement identifiable), mais choisir de ne le déplacer vers la fin du texte qu'en dernier, après toutes les autres modifications. Une équipe a mis à profit dans cette situation la fonction "aucune action", qui permet de surligner une partie de texte et de n'effectuer la modification que plus tard ; les autres équipes n'ont jamais utilisé cette fonction.

En ce qui concerne le fait de travailler à plusieurs sur un texte partiellement connu, les équipes ont souvent commencé le travail en survolant rapidement des yeux les parties qu'ils n'avaient pas eues à lire. Dans une équipe, les membres ont commencé par lire presque intégralement les parties des autres : ceci leur a pris du temps mais au final, ils ont été les plus rapides à effectuer le travail. Il en serait sans doute autrement avec un texte encore plus long. Les quelques équipes que nous avons fait passer sur un ordinateur ont utilisé la même stratégie : avant de commencer les modifications, elles ont fait défiler à l'écran le texte entier. Notre dispositif est plus avantageux pour cette étape puisque chaque personne ne relit pas sa propre partie, mais seulement celles des autres. Pour le travail sur le texte, dans chaque équipe, chaque personne a manipulé le logiciel. Le partage des tâches s'est organisé de façon naturelle dès le début du travail et n'a pas causé de problèmes, sauf pour quelques équipes dont les membres, voulant aller vite, ont appuyé en même temps sur la table.

Remarques issues des questionnaires :

Les sujets ont exprimé dans les questionnaires que les menus déroulants étaient moins intuitifs et moins faciles d'utilisation que le browser, alors que les observations semblaient montrer qu'ils étaient beaucoup moins à l'aise avec le browser. Le plus gros reproche fait aux menus déroulants est le fait qu'il faille rester appuyé avant de choisir l'action à effectuer, et certains ont proposé que le menu déroulant reste ouvert après une pression sur la table et jusqu'à la pression suivante. Nous n'avons pas voulu faire cette modification dans WordTab car elle s'oppose à notre objectif suivant d'utiliser la reconnaissance de gestes : on se retrouverait à cliquer comme sur un ordinateur alors que nous voulons aller dans le sens de gestes intuitifs. Cette difficulté avec le menu déroulant provient sans doute d'une expérience trop courte sur le dispositif. Plus de la moitié des sujets ont répondu qu'ils étaient "à l'aise pour travailler de façon autonome" sur le dispositif après les 10 minutes de présentation et d'essais du logiciel : l'expérience a prouvé le contraire puisque tous ont demandé des renseignements techniques par la suite ! Il faut donc sans doute compter plutôt 15 à 20 minutes pour le familiariser au dispositif.

Seul un sujet n'a pas pensé avoir gagné de temps par rapport à une équipe qui n'aurait travaillé qu'avec un ordinateur, les autres ont estimé avoir gagné entre 5 et 10 minutes. Environ la moitié pensent que le temps de familiarisation au dispositif est amorti au bout d'une seule utilisation : à la suite de ces tests, nous sommes un peu moins optimistes qu'eux.

Tous pensent que WordTab facilite le travail à plusieurs. En ce qui concerne le travail sur les textes longs, les réponses sont plus mitigées : certains (plus particulièrement ceux qui sont les plus expérimentés en informatique) regrettent que les modifications ne soient pas faites en temps réel. En revanche, un sujet a remarqué que garder le texte dans l'ordre jusqu'à la fin permettait de "se rappeler où on a lu les paragraphes", ce qui peut être un grand avantage lors du travail collaboratif, et auquel nous n'avions pas pensé.

Les réponses sur l'intérêt du logiciel ont rejoint les commentaires et suggestions sur WordTab. Certaines de ces réponses ont montré une compréhension incomplète des objectifs et avantages du dispositif, ou étaient imporssibles techniquement : désir de voir les modifications en temps réel, modifications du texte à petite échelle (orthographe par exemple), suppression des feuilles de papier... D'autres propositions étaient des choses auxquelles nous avions pensé, même implémentées puis remplacées par des fonctions plus performantes : il s'agissait de numéroter les flèches pour se souvenir de l'ordre des actions, d'accéder à une fonction en traçant un symbole sur la table, d'écrire le nom des fonctions à côté des icônes correspondantes... Nous pensons toujours que les nouvelles version du logiciel sont les meilleures, mais elles demandent peut-être un temps d'adaptation supplémentaire, d'où ces suggestions. Enfin, certaines suggestions nous ont paru intéressantes et ont été introduites dans une nouvelle version de WordTab. Nous avons d'abord corrigé certains problèmes (problème de luminosité des projections lors des suppressions, bugs lors du déplacement en début ou en fin de texte). Ensuite, suite à une remarque d'un sujet, un quadrillage s'affiche maintenant sur la table dans la période de sélection afin de faciliter le placement des feuilles.

Les questions sur les utilisations possibles du dispositif n'ont rien apporté : les sujets n'y ont vu que l'édition de texte et des applications graphiques. Il aurait peut-être fallu poser la question ("Ce dispositif vous semble-t-il utile ? Dans quelles type d'applications ?") d'une autre manière, en insistant que la question portait sur le dispositif de table interactive et non seulement sur le logiciel WordTab.

3.5 Conclusion

Cette procédure de tests nous a tout d'abord permi de constater que nous avions développé un logiciel qui marche, sans bugs depuis les quelques modifications apportées à la suite des tests. Il est apparu que WordTab était utilisable par un utilisateur quelconque au prix d'environ quinze minutes de formation. Son fonctionnement est intuitif et les utilisateurs n'ont pas de problèmes pour le manier, mais les concepts qu'il utilise sont assez nouveaux : si l'on se trouvait un jour dans une situation industrielle, il faudrait expliquer en détail ses avantages pour convaincre quelqu'un de son intérêt.

Les quelques équipes auxquelles nous avons fait faire le même travail sur un ordinateur ont mis les mêmes temps en moyenne, mais avec un écart-type plus important. On peut penser qu'il y aurait eu une différence de temps en faveur de WordTab si le texte retenu avait été plus long, ou si les sujets avaient eu plus de temps pour se familiariser avec le dispositif, mais nos tests ne sont pas suffisants pour affirmer que WordTab fasse gagner du temps. La recherche d'efficacité n'était en aucun cas l'objectif premier de WordTab, et il est en revanche apparu de façon nette que le dispositif encourageait chacun à participer et à manipuler, ce qui représente bien une amélioration du travail collaboratif.

4 Réflexion sur le processus de projet

Ce rapport final est l'occasion pour nous de dresser un bilan sur le processus de projet et sa dimension humaine. Depuis le début du projet, il y a maintenant un peu plus d'un an, notre vision du travail en équipe a évolué. Les mécanismes de communication au sein de notre groupe, alors en rodage lors de la rédaction de notre premier rapport, semblent maintenant correctement établis et fonctionnels, même s'ils s 'enraillent parfois encore, lors de la confrontation de points de vue divergents. La distribution des rôles a subi quelques modifications et présente désormais une configuration stabilisée. Il est alors intéressant de dresser ce bilan en miroir à la partie similaire du rapport initial, afin de mieux cerner l'évolution du groupe.

Il paraît judicieux dans un premier temps de faire quelques petits rappels sur la formation et les motivations de l'équipe afin de fixer les idées. Ensuite, nous présenterons la nouvelle distribution des rôles au sein du groupe en nous attachant particulièrement aux modifications effectuées. Dans le but de mieux comprendre cette nouvelle répartition des tâches, nous analyserons l'évolution de l'organisation du travail de groupe au cours de la réalisation de ce projet. Enfin, nous dresserons un bilan final sur les avantages et les difficultés rencontrés lors de notre expérience de travail en équipe.

4.1 La formation et les motivations du groupe

Dans un premier temps donc, nous tenions à rappeler les circonstances de la formation du groupe et les attentes de chacun dans la conduite de ce projet.

Ainsi, chacun des membres présentait déjà un vif intérêt pour l'informatique, ce qui nous a conduit tout naturellement à choisir un projet d'étude dans cette discipline. Celui-ci était particulièrement attrayant par la liberté qu'il offrait dans le choix du logiciel à développer compte-tenu de la diversité des applications envisageables pour une telle table interactive. Son côté innovant était, et reste, séduisant.

L'occasion d'approfondir ou de mettre en œuvre ses connaissances en informatique dans le cadre d'une réalisation concrète était à saisir, notamment en ce qui concerne le langage C++. Cette expérience apparaissait d'ailleurs comme un bon complément à l'enseignement prodigué par l'Ecole.

De plus, le projet ne se limitait pas uniquement à la programmation : il offrait également une véritable réflexion sur l'interactivité, la réalité augmentée et le travail collaboratif. En effet, il s'agissait de définir des besoins afin d'améliorer le travail collaboratif en utilisant le concept de réalité augmentée, et de valider à terme le logiciel suivant cette approche. De surcroît, il présentait une seconde dimension puisqu'il devait lui-même être le résultat d'un travail collaboratif.

L'équipe s'est formée au départ sur la base d'un noyau dur de trois élèves ayant suivi leurs études préparatoires au sein de la même classe et présentant les mêmes intérêts. Les deux autres membres se sont raccrochés ensuite au groupe par affinité avec le sujet.

Au moment de clôturer le projet, nous pouvons estimer que la réponse aux attentes de chacun est globalement positive. De plus chacun a eu l'occasion de contribuer à la réflexion portant sur la réalité augmentée et sur son apport dans le

cadre du travail collaboratif lors de la phase de conception, puis de validation du logiciel.

4.2 La répartition des rôles au sein de l'équipe

Depuis le premier rapport, les rôles des différents membres de l'équipe ont quelque peu évolué. Certains aléas de parcours ont conduit à une nouvelle distribution des tâches au sein du groupe. Il est donc nécessaire de procéder à une nouvelle description de l'équipe. Par ordre alphabétique donc…

Jean-Philippe Bianchi se charge essentiellement de l'automation de Microsoft Word par l'intermédiaire de Visual Basic. Cette tâche lui a été naturellement assignée au début du projet puisqu'il avait déjà une certaine expérience dans le domaine et possédait la documentation adéquate. Bien que son rôle n'ait pas fondamentalement évolué, sa tâche n'a pas été de tout repos : il lui a fallu s'habituer continuellement aux nouvelles exigences et solutions adoptées par le programme. De plus, son expérience dans l'automation grandissant, il a su améliorer constamment son code, tant et si bien qu'il a fourni plus d'une dizaine de versions de son module. Nous rappelons que celui-ci se charge d'interpréter le format *.doc* des fichiers textes Microsoft Word pour son utilisation dans WordTab, puis de la mise en forme dans ce même format du résultat des modifications apportées au texte. Il est aussi à l'origine du module de capture du flux vidéo provenant de la caméra. Enfin, il est également en charge de la trésorerie du projet.

Bien que s'étant penché initialement sur l'interprétation des modifications apportées au texte et sur sa restructuration, Thibaut Brusseaux s'est consacré à sa vocation première : la réalisation graphique. Ainsi, il est chargé du design de l'interface et du site Internet du projet. Après la découverte fortuite d'une bibliothèque graphique simple d'emploi, il a présenté à l'équipe un nouveau type d'interface. Celle-ci regroupait l'ensemble des souhaits émis par l'équipe qui n'étaient malheureusement pas facile de réaliser par les MFC de Windows. Michel, qui s'occupait alors de la première version de l'interface lui a laissé la main. Anna l'a également chargé d'ajouter une touche graphique au site web, d'autant plus qu'il a suivit l'approfondissement Technologies du Web en fin de première année.

Romain Clédat est le membre du groupe qui maîtrise le mieux le langage C++. C'est donc tout naturellement qu'il a été désigné pour rédiger le cœur du programme. Ainsi, il a mis au point une structure abstraite gérant les paragraphes du texte et les sélections, la plus générale possible, capable de s'affranchir de l'interface graphique et du logiciel de traitement de texte utilisé. La portabilité du noyau vers d'autres plates-formes voire d'autres applications est donc envisageable. Cette partie du programme a d'ailleurs pu être rapidement adaptée aux nouvelles fonctionnalités offertes par la nouvelle interface, certaines de ces fonctionnalités ayant même déjà été prévues. Romain a par ailleurs repris les travaux de Thibaut concernant le formateur de texte, puisqu'il s'agissait d'une partie en étroite relation avec son module. De plus, il se charge d'adapter un algorithme existant de reconnaissance de formes géométriques afin de permettre la détermination automatique des positions de feuilles sur la table par capture vidéo.

Après s'être consacrée à la rédaction du site Internet et à la valorisation du travail de groupe, Anna Michailovsky s'est chargée de la phase de validation du logiciel. Ainsi, à l'issu de la réflexion du groupe sur les tests à mettre en œuvre, elle

en a proposé une description claire ainsi qu'une grille d'analyse fonctionnelle. Elle s'est alors occupée de l'organisation logistique de la phase de tests qu'elle a entièrement supervisée. Enfin, elle s'est chargée de la synthèse des résultats de cette étape finale.

Michel Sato assume la lourde responsabilité de chef de projet. Il assure une bonne communication au sein du groupe, et motive l'équipe afin de mieux respecter les délais et objectifs fixés. Sa position centrale l'a amené à toucher à tous les modules du logiciel. Ainsi, il a été chargé dans un premier temps de l'interface, dont il a programmé la première version. Il a travaillé alors conjointement avec Jean-Philippe et Romain afin de conserver la compatibilité entre les différents modules. Une fois la nouvelle interface en cours de programmation, il a conservé ce rôle de supervision et s'est attaché à regrouper les différents modules. Il a lancé en parallèle les travaux sur l'implémentation de la caméra, dont il a assuré la programmation du module de calibrage.

4.3 L'évolution de l'organisation du travail

L'organisation du travail au sein de l'équipe a donc subi quelques modifications au cours de l'avancement du projet. La restructuration de la répartition des tâches a en fait été la réponse du groupe à une nécessité de redistribuer les rôles en fonction des compétences de chacun et dans le but de faire progresser le projet. Nous pouvons reprendre chronologiquement les grandes étapes du projet afin de mieux comprendre quand, comment et pourquoi l'organisation du travail a dû évoluer.

La première phase du projet, phase de recherche et de documentation, ne semble pas significative en soit. En effet, aucune tâche particulière n'a été assignée puisqu'il s'agissait de se renseigner sur les travaux existants exploitant une table interactive dans le cadre du travail collaboratif et de la réalité augmentée. Cependant, cette phase d'appropriation du sujet a son importance puisqu'elle a conduit au choix de l'application à développer et à une première répartition des tâches. Il a fallut de plus désigner un chef de projet afin de coordonner nos travaux.

Très vite, la division du programme en différents modules nous est apparue nécessaire. A cela plusieurs raisons. Tout d'abord, il s'agissait de satisfaire la volonté de chacun de programmer. Ensuite, il paraissait judicieux de répartir la charge de travail. De plus, nous voulions rendre le programme évolutif et en faciliter la maintenance en distinguant des modules spécifiques. Enfin, nous étions confrontés à une contrainte supplémentaire : l'automation de Microsoft Word ne semblait facilement envisageable qu'à partir de Visual Basic et il ne semblait pas envisageable de développer l'application dans son intégralité à partir de ce langage. Les rôles se sont alors répartis naturellement. Jean-Philippe ayant déjà essayé de piloter Word à partir de Visual Basic a été désigné pour assurer l'automation du traitement de texte. Romain ayant déjà une bonne maîtrise du langage C++ semblait la seule personne pouvant fournir au logiciel un noyau stable et efficace. Michel ayant déjà commencé à s'intéresser aux MFC de Windows se proposa pour fournir au programme un support graphique. Thibaut réfléchissait déjà au moyen de recombiner le texte en interprétant les modifications apportées et aux représentations graphiques intuitives dont qui pourraient agrémenter l'interface. Anna enfin, puisqu'il n'y avait aucun autre module à programmer, devait se charger de la valorisation du

travail par la rédaction du site web, pour laquelle elle se formait de manière autodidacte à HTML.

Cependant, le groupe commençait à rencontrer des difficultés d'ordre techniques. Tout d'abord, la communication entre le code de l'interface rédigé en C++ et le module d'automation écrit en visual Basic était problématique. La seule solution envisageable et propre était la mise au point d'un ActiveX. Mais il ne s'agit pas d'une tâche aisée, et elle a conduit aux premiers retards. Du côté de l'interface, les MFC se sont avérées fastidieuses à utiliser, d'autant plus que Michel s'y formait de manière autodidacte en parallèle. Les possibilités offertes par l'interface avaient dû être revues à la baisse afin de répondre à ces contraintes. Finalement, la première version du logiciel a pu voir le jour. Malgré quelques bugs, elle présentait le premier résultat concret pour le projet, et permettait de tester à la fois le module d'automation et le cœur du programme.

A la suite des vacances d'été, la découverte fortuite d'une bibliothèque graphique très simple d'emploi a conduit à une restructuration de l'organisation du travail. Elle permettait d'envisager de mettre en place toutes les fonctionnalités attendues dans l'interface. Ainsi, Thibaut a pu présenter une première esquisse d'une nouvelle version d'interface basée sur l'exploitation de cette nouvelle bibliothèque. Michel lui a donc laissé la main. L'incorporation directe de l'ActiveX au code C++ a été mise de côté laissant place à une solution moins propre mais de mise en œuvre plus aisée : l'utilisation d'exécutables indépendants. Cependant, l'ActiveX est toujours utilisé dans le module d'automation, qui utilise ses fonctions dans les exécutables qu'il contient. Jean-Philippe a donc dû s'attacher à adapter son module à la nouvelle structure adoptée. Romain a démontré le caractère général de sa structure en la rendant compatible avec les nouvelles fonctions disponibles depuis la nouvelle version de l'interface. Il a par ailleurs pris à sa charge le module de formation du texte. Michel a pu lancer parallèlement les recherches sur l'implémentation de la caméra. La réflexion sur la procédure de test a débuté, encadrée par Anna qui s'est chargée alors d'en superviser l'organisation. Thibaut a été désigné pour mettre en page le site web au regard sa formation de première année en Technologies du Web, afin de permettre à Anna de se consacrer à sa tâche.

Une fois la première version de la nouvelle interface disponible, les tests ont pu commencer. Pendant ce temps, Thibaut, Jean-Philippe et Romain se sont attachés à la relecture et à l'amélioration de leur code. Parallèlement Jean-Philippe a fourni un exécutable capable de récupérer le flux vidéo de la caméra, Michel s'est penché sur les problèmes de calibrage et Romain de la reconnaissance des cadres des feuilles.

Ainsi, l'équipe a su restructurer son organisation en fonction des difficultés rencontrées et des améliorations proposées afin d'assurer l'avancement du projet.

4.3.1 Les avantages et les inconvénients du travail en équipe

Comme nous l'avions déjà souligné à l'occasion du premier rapport, ce projet constitue pour la plupart d'entre nous une première expérience de travail en équipe s'inscrivant sur une longue période. Cette expérience nous a semblé essentielle dans le cadre de notre formation d'ingénieur. En effet, il est possible d'en tirer plusieurs enseignements tant sur le plan des avantages que des difficultés que

présente le travail en groupe. Les situations rencontrées dans le cadre de cet exercice nous permettront de mieux appréhender des situations analogues auxquelles nous pourrions être confrontés dans notre profession future.

Le premier avantage remarquable qu'apporte le travail en équipe est incontestablement la répartition de la charge de travail. En effet, un projet d'une telle ampleur ne saurait être supporté par un individu seul dans un temps raisonnable. De plus, la diversité des domaines abordés multiplie les exigences en termes de compétences requises. Afin d'obtenir un résultat de qualité, il apparaît donc nécessaire de réunir des personnes d'horizons variés et complémentaires.

Cependant, cette juxtaposition de compétences ne sert à rien s'il est impossible de les faire travailler ensemble. Pour cela, la figure de chef de projet nous semble particulièrement importante. Elle permet une répartition et une coordination des tâches sans perdre de vue les objectifs principaux du projet. Elle peut aussi jouer un rôle dans la modération et la régulation de la communication au sein du groupe. Il est également nécessaire de pouvoir et savoir faire confiance à ses coéquipiers, ce qui n'est pas toujours évident lorsque l'on a été habitué à travailler individuellement.

Le travail d'équipe permet également la multiplication des points de vue. En cela il est enrichissant sur un plan principalement personnel. De plus, il implique la confrontation d'idées, et ouvre donc la voie à des solutions auxquelles on aurait pu ne pas penser seul. Il en résulte une émulation profitable au projet.

Cependant, il ne faut pas tomber dans l'extrême. En effet, l'apparition de différences d'opinions peut engendrer la discorde. Dans ce cas, le processus de projet peut se trouver bloqué, et la perte de temps occasionnée peut s'avérer considérable. Le chef de projet peut alors intervenir, dans le cas évidemment où il n'est pas lui-même impliqué. La meilleure façon de remédier à ce genre de situation reste de tenter de se contrôler soi-même. Il faut apprendre à écouter ses coéquipiers et à considérer leurs idées. Ceci peut être la tâche la plus difficile, d'autant plus, si l'on a l'habitude de travailler seul.

Le fait de programmer en modules peut aussi poser problème. En effet, chacun doit programmer une petite partie à la fois indépendante et dépendante des autres : c'est un morceau de code formant une entité, mais qui devra bien sûr communiquer avec les autres parties du programme. Il peut alors se faire que plusieurs personnes, programmant simultanément, prennent des orientations ou fassent des choix de programmation légèrement différentes : il faut alors veiller à la compatibilité des différentes parties. Lors de ce projet, chacun a pris soin d'informer systématiquement par e-mail le reste de l'équipe des modifications dans ses travaux et d'en mettre la nouvelle version sur notre serveur CVS. Par contre, avant de continuer à travailler sur sa partie, nous n'avons pas toujours pensé à vérifier ce qu'avaient fait les autres, et il a parfois fallu que des personnes adaptent leur partie après-coup. Ce problème reste inférieur aux avantages en terme d'efficacité et de possibilités de réutilisation qu'offre le programmation en modules.

L'organisation de réunions est un exercice difficile auquel nous avons dû nous habituer au cours de ce projet. Très vite, nous avons compris la nécessité de préparer les réunions à l'avance, en déterminant un plan de réunion et des durées approximatives à consacrer à chaque point du plan. Ce travail nécessite de faire une synthèse des résultats de chacun et oriente le travail à venir, en cela il revient naturellement au chef de projet. Cette préparation nous a permis d'être plus efficaces, que ce soit lors des réunions hebdomadaires internes au groupe ou lors des rencontres avec nos tuteurs. En revanche, le groupe a mis plus de temps à savoir determiner quels membres de l'équipe devaient participer à une réunion

donnée : il en a résulté des réunions où tout le groupe était présent, mais seuls quelques uns assez compétants pour participer réellement au débat. Une réflexion sur la composition de la réunion fait maintenant partie de son organisation, et ce n'est plus un problème.

Le travail en équipe peut également présenter un côté agréable. On peut y trouver une ambiance de travail conviviale pour peu que l'entente règne au sein du groupe. Ceci peut éventuellement apparaître comme le défaut de l'exercice. En effet, le groupe réunit une population d'individus relativement hétérogène sur le plan de l'âge et des études suivies. De plus, il n'existe aucun lien hiérarchique, et les membres du groupe peuvent avoir éventuellement noué des liens d'amitié. Ainsi, la situation est légèrement faussée. Cependant, on peut facilement s'imaginer que si des difficultés ont pu apparaître même dans ce cadre « protégé », elles pourraient être bien plus complexes dans une situation réelle de la vie active. Il est bon de se poser ces questions lors de notre formation, afin d'être capables de trouver le recul nécessaire le temps voulu.

Ainsi, cette expérience en équipe nous est apparue comme enrichissante. Outre le fait de nous préparer à une situation courante de la vie active, elle nous a permis de bénéficier de l'expérience des autres dans des domaines où nous n'étions pas ou peu compétents. Si les avantages du travail en groupe sont évidents, les difficultés qu'il présente sont elles aussi bien réelles. Cependant, nous avons eu la chance de ne pas en avoir trop souffert dans le cadre du déroulement de notre projet.

4.3.2 Conclusion

Nous avons réussi, au bout de quinze mois, à répondre à la plupart des objectifs que nous nous étions fixés en début de projet : notre livrable (sous forme de CD) contient, en plus du logiciel WorTab, toutes les sources que nous avons programmées pour arriver à sa version finale, les librairies utilisées, la documentation technique ainsi que les résultats des expériences de validation menées pendant un mois. Ce livrable, et plus précisément le logiciel, répond au problème initial qui était d'améliorer le travail collaboratif grâce à une table interactive : tout en conservant les fonctions très pratiques de Word (copie, suppression…), l'utilisateur peut désormais effectuer ces manipulations avec une vision d'ensemble du document et surtout, plusieurs personnes peuvent travailler ensemble sur le dispositif sans se gêner. Nos résultats n'auraient rien signifié si nous n'avions pas procédé à une série de tests de validation : cette étape primordiale dans le projet, même si elle n'a pas été assez longue pour apporter toutes les réponses suffisantes a notre problème, a bien été assurée! Elle nous a même permis de soulever des points ayant conduit à l'amélioration du logiciel.

A cause de certains retards dus à des problèmes techniques, nous n'avons malheureusement pas pu étudier la mise en réseau de plusieurs tables. Cependant, nous avons réussi, avec l'aide de M. Chalon, à recruter une nouvelle équipe qui prendra le relais et s'attaquera à cet aspect important du travail collaboratif : le travail à distance.

Au delà de la satisfaction d'avoir mené un projet jusqu'au bout, avec toutes les difficultés que cela implique, TABLEGATE aura été une expérience passionnante pour chacun d'entre nous en nous faisant découvrir le travail en équipe. Nos rôles ayant changé au cours de ces quinze mois, nous nous sommes tous investis dans

plusieurs domaines (programmation, tests, site Web) ce qui nous a permis de profiter pleinement de l'opportunité que nous a offert ce projet d'étude.

5 Remerciements

L'équipe du projet TABLEGATE tient à remercier tout particulièrement M. René Chalon pour toute l'aide qu'il a pu lui apporter au cours de ces quinze mois de travail, ainsi que M. Bertrand David, Jacqueline Vacherand-Revel et M. Alexander Saïdi.

Nous tenons aussi à remercier :

- Toutes les personnes qui ont accepté de réaliser les tests sur la Table
- Isabelle San Jose, secrétaire à l'ICTT pour son aide lors de ces tests
- Tous les membres de l'ICTT pour avoir supporté le bruit

6 Bibliographie

Ivor Horton, **Visual C++ 6.0**, éditions Eyrolles, 2003

B. Stroustrup, **C++**, Troisième Edition, Campus Press

Guy de Maupassant, **Contes fantastiques**, éditions Marabout, 2003

Bigard, sketch " **La chauve-souris** "

http://www.intel.com/research/mrl/research/opencv/ pour la librairie Intel

http://www.libsdl.org/index.php pour la librairie SDL

J. L. Crowley, F. Berard et J. Coutaz , **Finger Tracking as an Input Device for Augmented Reality**, International Workshop on Face and Gesture Recognition, 1995

A. Demeure et G. Calvary, **Jeu et Réalité Mixte : Retours d'expérience**, IHM, 2002

W. E. Mackay, **Augmented Reality: Linking real and virtual worlds**, ACM Press, 1998

P. Wellner, **Interacting with Paper on the DigitalDesk**, ACM Press, 1993

7 Annexes

Annexe A : Qu'est-ce qu'un contrôle ActiveX ?

Un contrôle ActiveX correspond à du code compilé utilisable en théorie dans tout autre langage de programmation Windows courant. Il se compose de 3 parties : les méthodes, les propriétés et les évènements.

Les méthodes sont des procédures ou des fonctions appelées depuis le conteneur éventuellement avec des paramètres, qui s'exécutent dans l'activeX. Elles sont appelées exactement comme si c'était des procédures ou fonctions du programme conteneur : ActiveX.nom_de_méthode param1, param2,… lancera la procédure nom_de_méthode de l'activeX, qui admet param1, param2,… comme paramètres.

Les propriétés sont des méthodes particulières qui permettent de lire dans des variable locales de l'ActiveX (les attributs), accessibles depuis le conteneur en lecture et/ou en écriture par leur intermédiaire. En fait au moment de la création de l'activeX, cela se programme comme une classe : dans le cas d'une propriété accessible en écriture on a une variable locale et il faut prévoir une fonction property let nom_de_propriété qui se charge de transmettre le contenu de la variable depuis le conteneur jusqu'à cette variable locale. De la même manière, pour une propriété accessible en lecture il faut prévoir une fonction property get nom_de_propriété qui lit dans la variable locale et transmet son contenu au conteneur. Cependant vu du conteneur Visual Basic, cette propriété présente l'avantage d'être utilisable comme une variable. Ainsi « nom_ActiveX.nom_de_propriété = 8 » affectera la valeur 8 à la propriété nom_de_propriété de l'ActiveX.
En Visual C++, la propriété se décompose en 2 méthodes set et get, utilisées respectivement pour écrire et lire dans un attribut.

Un évènement est une procédure interne à l'activeX qui se déclenche automatiquement lors d'un certain évènement extérieur. Par exemple, dans le cas d'un activeX avec interface graphique, cet évènement extérieur peut être un survol de la souris, un click avec le bouton gauche, etc. …

Annexe B : Comment créer un contrôle ActiveX en Visual Basic ?

Toutes les procédures internes qui ne doivent pas être appelées depuis l'extérieur sont déclarées de la manière habituelle en Visual Basic :
Private Sub nom_procédure_interne(param1,parm2,…)
…
End sub

En revanche une procédure interne qui devra pouvoir être appelée depuis l'extérieur, et donc constituer une méthode de l'ActiveX, sera déclarée de même, en remplaçant Private par Public :
Public Sub nom_de_méthode(param1,parm2,…)
…
End sub

Les propriétés doivent permettre de lire et d'écrire dans les attributs.
Les attributs sont des variables locales à l'ActiveX. On les déclare comme n'importe quelle autre variable en Visual Basic, avec pour convention de précéder son nom par « m_ » (ainsi l'attribut vsbl sera une variable locale nommée m_vsbl).
La propriété porte par convention (pas obligatoire) le même nom que l'attribut associé. On prévoit 2 procédures, pour écrire et pour lire dedans.
Celle permettant d'écrire dans l'attribut nom_attribut s'écrira :
Public Property let nom_attribut(new_val as type_new_val)
…
m_nom_attribut=new_val
End Property
Celle permettant de lire dans l'attribut nom_attribut s'écrira :
Public Property get nom_attribut() as type_m_nom_attribut
…
nom_attribut= m_nom_attribut
End Property
On peut aussi ajouter du code à la place des … dans les 2 cas. Il ne faut pas limiter l'ActiveX à une simple interface entre des variables et un conteneur, c'est aussi un programme à part entière.

Les événements se programment de manière analogue à des événement Visual Basic. Il n'est cependant pas conseillé de les utiliser dans un ActiveX destiné à VC++ car l'utilisation en sera grandement compliquée.
Si l'ActiveX est destiné à être inséré dans un programme VC++ il est conseillé de travailler avec des attributs et des paramètres de type Variant, qui sont assez universels (alors que integer de VB différent de int de VC++ par exemple…).
Enfin avec VB6, et sans doute avec les versions suivantes, se trouve un assistant de création d'interface pour les contrôles ActiveX, qui facilite grandement la tache.
On y accède dans projet/ajout de contrôle utilisateur.
Sélectionner « Assistant d'interface de controle ActiveX ».
La 2eme page de l'assistant permet d'ajouter des méthodes, propriétés (et évènements) à l'interface.
La 4eme page permet de définir

-pour les méthodes : le type renvoyé si elles renvoient quelque chose (Empty sinon), la listes des paramètres sous forme param1 as type param1, param2 as type_param2…
-pour les propriétés : à quel type d'attribut elle est associée, et si elle est disponible en lecture et/ou écriture…

A la fin, l'assistant fournit une ossature du code, avec les bons prototypes pour chaque méthode/propriété, ainsi que les bonnes déclarations des attributs. Cela constitue la partie interface de l'ActiveX. Reste à écrire tout le code (ce que fait réellement le programme ActiveX, qui remplit les attributs avec les bonnes valeurs).

Annexe C : L'algorithme de reconnaissance des pages

- ***Principes***

Pour nous aider à détecter la présence de feuilles, nous nous sommes servis de la librairie libre d'Intel OpenCV. Cette librairie détecte en particulier les rectangles présents sur une image. Nous nous sommes servis de cette fonction pour déterminer les rectangles présents sur une photo prise de la table. En effet, sur chaque feuille, un bord noir a été dessiné ce qui permet un repérage facile. Cette méthode reste tout de même très dépendante des conditions de lumière de la piece. Nous avons tenté de remédier à ce problème en projetant un fond bleu (ou ROUGE A VOIR) sur la table au moment de la prise de vue.

Une fois les rectangles trouvés, ils sont stockés sous forme de liste de points. Quatre points forment un rectangle. Cette liste de points va ensuite être interprétée par notre algorithme pour sortir les rectangles intéressants. En effet, le défaut des méthodes de la librairie est de sortir de nombreux rectangles s'approchant de plus en plus du véritable rectangle. De plus, l'épaisseur du bord des feuilles fait qu'il y a au moins deux rectangles qui sont trouvés (celui à l'intérieur du bord et celui à l'extérieur du bord). Il s'agit donc de regrouper tous les rectangles qui vont ensemble pour ne sortir qu'une seule feuille à la fin.

- ***Etapes***

L'analyse des points s'effectue en plusieurs étapes:

1)
On prend les quatre premiers points de la liste qui doivent donc former un rectangle. On vérifie que la taille du rectangle formé correspond à peu prêt à la taille d'une feuille de papier. En effet, la librairie peut par exemple déceler le rectangle représentant la projection entière du rétroprojecteur. Il faut donc éliminer cette "feuille".

2)
On recherche ensuite parmi tous les autres points dans la liste un autre point proche du premier point de la liste. Pour trouver un point proche, on calcule tout bêtement la distance entre deux points et on compare cette valeur à une constante fixée qui déterminer la tolérance sur la proximité de deux points.

3)
Il s'agit ensuite de reconstituer l'autre rectangle à partir de ce point. La principale difficulté ici est que les points ne sont pas ordonnés. Ainsi, le point trouvé peut très bien être au milieu (dans la liste) des quatre points qui constituent le rectangle. Ainsi, si nous trouvons un point en position n, le point qui correspondra au point 2 de notre triangle initial peut se trouver en position n-1 ou n+1.

Pour déterminer la bonne combinaison de points, on compare les différentes possibilités en prenant celle qui minimise la distance globale entre tous les points.

4)
On calcule ensuite une moyenne de position sur chacun des points pour essayer d'obtenir la meilleure approximation du point réel.

5)
On efface de la liste les quatre points traités et on recherche ensuite dans la suite de la liste un autre point correspondant a l'étape 2. On reboucle jusqu'à ne plus avoir de points à traiter

6)
On efface les quatre premiers points de la liste.

7)
On stocke la page trouvée et on retourne à l'étape 1.

- ***Difficulté majeure***

La difficulté majeure a consisté à reconstruire les rectangles à partir de la liste des points. Il etait en effet primordial de trouver tous les rectangles correspondants à une même feuille pour en pas avoir des doublons à la sortie. Le problème de l'ordre des points et du sens de rotation dans les rectangles a posé de nombreux problemes qui ont finalement été résolus par l'analyse soigneuse de tous les cas possibles. Les formules trouvées pour référencer les points (avec des 'ET' bit à bit et des opérations de modulo) sont très certainement simplifiables mais elles ont le mérite de marcher. Nous allons donc pour l'instant les conserver car le temps gagné lors de l'exécution avec des formules simplifiées serait sûrement dérisoire.

Annexe D : Les classes abstraites

Il serait fastidieux de décrire en détail le fonctionnement de toutes les classes abstraites. Pour une compréhension complète du fonctionnement de ces classes, le mieux est de se référer aux fichiers entêtes. Le principal s'appelle selection.h; il contient les déclarations de toutes les fonctions wrappers ainsi que les déclarations et les définitions de tous les modèles qui forment les classes abstraites (il faut en effet que les déclarations et les définitions soient dans un même fichier pour que les modèles puissent fonctionner). Nous allons brièvement décrire ici les principes sous-jacents ces classes.

Il est important de noter que ces classes ont mûries au cours de l'évolution du projet et ne sont donc pas parfaites. Les continuels changements qu'on y a effectués laissent parfois entrevoir des irrégularités et des illogismes. Des utilisateurs futurs pourront modifier ces classes sous les termes de la licence GNU (http://www.gnu.org/licenses/licenses.html#GPL)

Les templates

Toutes les classes abstraites, comme précisées précédemment, sont implémentées en temps que Template. Ceci permet de spécifier les modules 'plugins' qu'on veut rajouter pour chaque classe. Bien entendu, vu l'interdépendance des classes, la template nécessite trois paramètres, à savoir les trois types de 'plugins' pour les trois classes.

Ceci permet donc une grande évolution possible pour ces classes. En effet, on peut stocker absolument toutes sortes de données dans le 'plugin' ce qui permet d'associer n'importe quelle interface graphique au cœur du programme.

Le repérage dans l'espace

Le programme repère des feuilles, des sélections et des paragraphes dans un espace bi-dimensionnel; à ce titre deux coordonnées sont nécessaires pour repérer un point. Le programme ne considère aussi que des formes rectangulaires en repérant le point supérieur gauche et le point inférieur droit de chaque rectangle. Il est donc impossible d'avoir des paragraphes alambiqués ou des pages de formes non conventionnelle.

Une classe échappe à ce principe; la classe Cune_selection regroupe un ensemble de Cun_paragraph. Ainsi, une instance de Cune_selection n'a pas de lieu rectangulaire dans l'espace mais plutôt l'union de lieux rectangulaires. Pour une une sélection, seul le point d'insertion est donc repéré avec des coordonnées. Nous avons en fait considéré une ligne d'insertion horizontale et donc trois coordonnées sont considérés pour le point d'insertion à savoir: sa coordonnée en y et deux coordonnées en x. Ceci permet en particulier de bien calculer les extrémités des flèches dans la seconde interface graphique.

Deux coordonnées sont données pour chaque point: des coordonnées absolues par rapport au point (0,0) de l'ordinateur et des points en pour millième par rapport au point (0,0) de la feuille. Ces deuxièmes coordonnées sont importantes car Word raisonne dans ces coordonnées. En effet, ce sont les seules coordonnées valables lorsque la feuille en elle même n'a pas de coordonnée réelle. Ainsi, il est important de toujours garder ces deux coordonnées synchronisées. Un point est virtuel si les coordonnées absolues ne peuvent pas être connues. Il faut bien noter que si les coordonnées relatives ne sont pas disponibles et qu'on efface le paragraphe ou une page contenant un point d'insertion d'une sélection, on risque de perdre des informations. Une exception est soulevée dans un cas pareil. Les coordonnées privilégiées sont les coordonnées relatives. On s'en sert donc souvent par exemple pour obtenir les coordonnées absolues lors d'un redimensionnement d'une feuille.

Communication par exceptions

Beaucoup de fonctions soulèvent des exceptions. Ceci ne veut pas dire qu'il y a eu un problème particulier qui doit arrêter le programme mais plutôt qu'une condition particulière a été découverte et doit être traitée à l'endroit approprié. Par exemple, un cas classique est l'exception Virtual_Page qui est produite à chaque fois qu'une opération est tentée sur une page virtuelle. Cette exception permet aux fonctions appelantes de savoir que la page est virtuelle et de la traiter en conséquence. Les exceptions sont un mécanisme ultra puissant pour communiquer entre les fonctions et permettre une bonne remontée des messages le long de la chaîne d'appel des fonctions.

Annexe E : Description technique de l'interface

L'interface de WordTab est programmée en C++. Elle utilise une bibliothèque multimédia multi-plate-formes gratuite : SDL (Simple DirectMedia Layer). Celle-ci gère l'affichage des graphismes, l'environnement sonore, ainsi que les événements de la souris qui traduisent les actions effectuées sur la table tactile.

Pour bénéficier des fonctions de la SDL, il suffit d'inclure les fichiers suivant en en-tête du programme :

- SDL.h
  Manipulations graphiques élémentaires, gestion des événements de la souris et du clavier, émulation sonore basique mais difficile d'emploi, ainsi que d'autres fonctionnalités non exploitées par le programme...
- SDL_mixer.h
  Gestion simplifiée de l'émulation sonore.
- SDL_ttf.h
  Gestion des polices au format « true type ».

Figure 31 : La bibliothèque SDL

Il est également nécessaire d'indiquer les bibliothèques correspondantes à l'éditeur des liens : SDLmain.lib, SDL.lib, SDL_mixer.lib et SDL_ttf.lib. Enfin, il suffit de fournir les dll adéquates pour faire fonctionner l'exécutable : SDL.dll, SDL_mixer.dll et SDL_mixer.dll. Plusieurs sites Internet décrivant la démarche à suivre pour créer le projet suivant les environnements de programmation existent.

SDL s'initialise par la ligne de commande suivante qui indique que nous désirons utiliser les fonctions vidéos et sonores :

```
SDL_Init(SDL_INIT_AUDIO|SDL_INIT_VIDEO);
```

Les modules vidéo, audio et de gestion des polices « true type » s'initialisent ensuite individuellement. De plus amples détails sont donnés dans la suite de cet exposé. Les fonctions d'initialisation sont appelées par la procédure `Init()` du programme de l'interface.

Les éléments graphiques

SDL manipule des « surfaces » (rectangulaires) qui peuvent être vue concrètement comme des pointeurs sur la mémoire vidéo. Lors de l'initialisation du mode vidéo, il faut désigner une surface principale qui correspond à la surface qui

sera affichée sur la sortie vidéo. Il est également possible de définir des surfaces hors écran de différentes tailles, ce qui permet le stockage de sprites ou la mise en place d'un double-buffering.

SDL dispose donc de sa propre structure pour définir une surface :

```
SDL_Surface *maSurface;
```

L'initialisation du mode vidéo se fait donc en désignant une surface principale (`fenetre` dans le cadre de notre programme) :

```
fenetre = SDL_SetVideoMode(1024, 768, 32, SDL_HWSURFACE);
```

Ici, on indique la résolution de l'écran (1024*768), la profondeur d'affichage des couleurs (32 bits) et l'utilisation de l'accélération matérielle (attribut `SDL_HWSURFACE`) si elle est disponible.

Cependant, on ne dessinera pas directement dans cette surface. Afin d'obtenir un rafraîchissement acceptable de l'écran, on utilise une surface tampon de travail de mêmes dimensions que la surface principale (la surface `fond`). Une fois tous les éléments graphiques dessinés sur cette surface, il suffit de la copier entièrement dans la surface principale (commande `SDL_BlitSurface()`) puis d'actualiser l'affichage (commande `SDL_UpdateRect()`) :

```
SDL_BlitSurface(fond, 0, fenetre, 0);
SDL_UpdateRect(fenetre, 0, 0, 0, 0);
```

En fait la commande `SDL_BlitSurface()` est bien plus puissante. Elle permet de copier une partie d'une surface à un endroit donné dans une autre surface. Afin de définir ces zones, il faut utiliser la structure `SDL_Rect`. En effet, elle permet de définir un rectangle par les coordonnées de son sommet supérieur gauche (attributs `x` et `y`) et par ses dimensions (sa hauteur `h` et sa largeur `w`). Ainsi, pour copier une partie d'une surface dans une zone donnée sur une surface cible, il suffira d'écrire :

```
SDL_Surface *surfaceSource;
SDL_Surface *surfaceCible;
SDL_Rect zoneSource;
SDL_Rect zoneCible;
... // Définition des zones et des surfaces
SDL_BlitSurface(surfaceSource, &zoneSource, surfaceCible, &zoneCible);
```

Pour dessiner sur les surfaces, il est possible d'écrire directement dessus pixel par pixel, ou alors d'y dessiner des rectangles élémentaires. Pour un accès direct aux pixels, il suffit d'utiliser directement le pointeur "pixels" associé à la surface (`int *pointeur = (int *)maSurface->pixels;`). Celui-ci fournit un accès total à la mémoire associée. Attention, si la surface est allouée en mémoire vidéo, il faudra veiller à encadrer l'accès direct par des appels aux fonctions `SDL_LockSurface()` et `SDL_UnlockSurface()` (Se reporter à une documentation sur l'utilisation de SDL pour de plus amples informations à ce sujet). Le remplissage de rectangles est possible sans se soucier de ces détails. Dans ce cas, il suffit de définir une zone grâce à une structure `SDL_Rect`, et d'utiliser la procédure `SDL_FillRect()`.

```
SDL_Surface *maSurface;
SDL_Rect monRectangle;
int rouge, vert, bleu;
...
int maCouleur = (rouge<<16) + (vert<<8) + bleu;
SDL_FillRect(maSurface, & monRectangle, maCouleur);
```

Il est également possible de charger directement dans une surface une image au format bitmap (*.bmp*). Cette opération se fait à l'aide de la commande `SDL_LoadBMP()`.

```
SDL_Surface *maSurface;
maSurface = SDL_LoadBMP("image.bmp");
```

Enfin, il est permis de définir pour chaque surface une encre transparente. Ainsi, lors de la copie d'une partie de cette surface sur une autre surface, les pixels de cette couleur ne seront tout simplement pas copiés, et laisseront visibles les pixels de l'image de destination. Cet effet et largement utilisé dans le programme, notamment pour l'affichage des icônes circulaires. La fonction utilisée se nomme `SDL_SetColorKey()`.

```
SDL_Surface *maSurface;
int rouge, vert, bleu;
...
int couleurTransparente = (rouge<<16) + (vert<<8) + bleu;
SDL_SetColorKey(maSurface, SDL_SRCCOLORKEY, couleurTransparente);
```

Il ne faut pas oublier de libérer la mémoire allouée aux surfaces en sortie de programme. Pour cela un simple appel à la procédure `SDL_FreeSurface(maSurface)` suffit. La procédure `Quitter()` du programme se charge de libérer toutes les surfaces utilisées par l'interface.

Il s'agit ici des principales fonctions graphiques utilisées pour réaliser l'interface. L'affichage des icônes, des éléments graphiques des menus et du curseur utilise la copie de surfaces. En effet, ces éléments sont stockés dans des surfaces hors écran. Pour les fenêtres, dont on peut modifier les dimensions, la représentation graphique est effectuée au moyen de rectangles pour les bords et le cadre intérieur, l'entête et les coins du cadre sont quant à eux stockés dans des surfaces.

La fonction `Chargements()` du programme charge les bitmap contenues dans le dossier *bitmap* dans les surfaces de stockage correspondantes. Afin de faciliter la correspondance entre les fichiers et les surfaces, les surfaces portent le même nom que le fichier associé.

Barre.bmp	Barre d'état inférieure. Celle-ci a les mêmes dimensions que la barre d'état de Windows, de telle sorte que lorsque la barre de Windows est visible, cette dernière recouvre cette partie de la surface principale du programme.
Browser.bmp	Eléments graphiques du *browser*.
Curseurs.bmp	Les différentes formes que peut prendre le curseur.
Elements.bmp	Eléments graphiques de l'en-tête des fenêtres.
Fond.bmp	Image de fond. Elle peut donc être modifiée par l'utilisateur. Il faut cependant respecter la résolution : 1024*768. Elle est stockée dans la surface `image_fond` et sert à renouveler l'affichage dans la surface `fond`.
Icones.bmp	Le jeu d'icône dans les états sortant et enfoncé.
Jauges.bmp	Eléments graphiques du menu de réglage de la couleur définie par l'utilisateur.
Mask_copier.bmp	Masque du motif de surlignage des sélections destinées à être copiées.
Mask_deplacer.bmp	Masque du motif de surlignage des sélections destinées à être déplacées.
Mask_supprimer.bmp	Masque du motif de surlignage des sélections destinées à être supprimées.
Word_tab.bmp	Image de présentation au démarrage de l'application. Cette image n'apparaissant qu'une fois, au lancement de l'application, elle n'est stockée que temporairement.

Chaque partie graphique de l'interface dispose de sa procédure d'affichage. Celle-ci se charge de tracer sur la surface `fond` les éléments correspondants en fonction de l'état de l'objet.

`Affiche_grille()`	Affiche la grille de positionnement en mode « édition des feuilles »
`Affichage_feuilles()`	Affiche les fenêtres. Prend en compte le mode d'édition actif, car les fenêtres n'ont pas exactement le même aspect dans les deux modes.
`Affichage_paragraphes()` `Affichage_lignes_insertion()`	Affiche le surlignage des paragraphes sélectionnés et les lignes d'insertion en mode « édition des sélections ».

`Affichage_icones()`
`Affiche_icone`
`(int type, int x, int y, bool enfoncé)`
	Affiche les icônes. La procédure `Affichage_icones()` boucle sur toutes les icônes de l'interface et appelle la procédure `Affiche_icone(...)` qui place l'icône de type `type` à la position (`x`, `y`) dans l'état enfoncé ou non suivant la valeur du booléen `enfonce`.

`Affichage_menu_1()`
`Affichage_menu_2()`
`Affichage_menu_3()`
	Affichent les menus. Le menu 1 représente le menu de réglage de la couleur définie par l'utilisateur. Le menu 2 correspond au *browser* des sélections. Le menu 3 représente le menu en croix.

`Affichage_barre()`	Affiche la barre d'état inférieure.
`Affichage_curseur()`	Affiche le curseur.

Chacune de ces procédures est appelée par la fonction principale `main()` dans un ordre déterminé afin de respecter certaines règles de superposition. Tout d'abord, l'image de fond est copiée dans la surface tampon, ce qui a pour effet d'effacer les anciennes données affichées. Ensuite, la grille est tracée si l'interface est en mode « édition des feuilles ». Puis le programme affiche les fenêtres, ainsi que les surlignages et lignes d'insertion si le mode « édition des sélections » est actif. Le fléchage a un statut spécial. En effet, les flèches sont des courbes de Bézier. Leur calcul et leur affichage sont gourmands en temps d'exécution. Ainsi, elles ne sont recalculées que dans certains cas précis susceptibles d'avoir modifié leur configuration (définition ou modification de lignes d'insertion, basculement entre les deux modes d'édition ...). Le calcul est effectué par la procédure `Trace_fleches()` et l'ordre de rafraîchissement est contrôlé par le booléen `rafraichissement_fleches`. En fait, lors de leur calcul, les flèches sont tracées dans une surface transparente annexe `fleches`. Celle-ci conserve donc le tracé des flèches et est systématiquement superposée à la surface de fond. Ensuite, il s'agit d'afficher les menus 1 et 2 puis les icônes. Le menu 3, ou menu en croix, n'est affiché que s'il est actif. Enfin, le curseur est superposé au tout, et la barre d'état inférieure est ajoutée. L'affichage complet est rafraîchi à chaque itération de la boucle principale, ce qui permet de limiter la complexité du programme, mais reste extrêmement gourmand en ressources.

L'environnement sonore

Pour bénéficier de la puissance de SDL_mixer, il faut tout d'abord l'initialiser :

```
Mix_OpenAudio(22050, AUDIO_S16SYS, 2, 2048);
```

Les réglages ci-présents sont ceux conseillés pour une utilisation standard de la bibliothèque. La bibliothèque met à la disposition de l'utilisateur une structure spécifique pour stocker les échantillons sonores :

```
Mix_Chunk *monSon;
```

Il suffit ensuite de charger les sons grâce à la procédure `Mix_LoadWAV()`. Les formats reconnus sont assez nombreux, mais ici nous nous sommes contentés du format *.wav*.

```
monSon = Mix_LoadWAV("sound.wav");
```

La lecture se fait alors par un simple appel de la procédure `Mix_PlayChannel()` :

```
Mix_PlayChannel(0, monSon, 0);
```

Le premier argument indique le canal sur lequel sera joué l'échantillon. Par défaut, huit canaux sont disponibles, ce qui permet de jouer huit sons en même temps. Le troisième argument désigne le nombre de fois que sera répété l'échantillon. Ici, on ne désire pas répéter l'échantillon.

En sortie de programme, il est nécessaire de libérer la mémoire allouée aux échantillons sonores :

```
Mix_FreeChunk(monSon);
```

L'environnement sonore de l'interface n'est pas très riche mais a le mérite d'exister. Les variables stockant les échantillons sonores portent les mêmes noms que les fichiers *.wav* correspondants, auxquels a été ajouté le préfixe de « s_ ». Les fichiers sons sont enregistrés dans le répertoire *sound*. Les échantillons sonores utilisés par l'interface sont chargés par la procédure `Chargements()`. Ils sont libérés dans la procédure `Quitter()`.

Gliss.wav	Déplacement des menus.
New.wav	Création d'une nouvelle sélection.
Press.wav	Clic sur des icônes ou des boutons de contrôle.

Les polices « true type »

Cette partie de la bibliothèque nécessite également une initialisation (toujours dans la procédure `Init()` de notre programme) :

```
TTF_Init();
```

Les polices « true type » sont stockées dans une structure réservée : `TTF_Font`. Elles sont chargées par l'appel de la commande `TTF_OpenFont()` dans laquelle il faut spécifier la taille des caractères désirée (ici 12) :

```
TTF_Font *maPolice;
maPolice = TTF_OpenFont("ARIAL.TTF", 12);
```

L'interface n'utilise que deux polices de caractères de taille 12 : ARIAL.TTF et ARIALBD.TFF (police grasse). Elles sont enregistrées dans le répertoire *fon*t. Elles sont chargées lors de l'appel de la procédure `Chargements()` dans les variables `font_normal` et `font_gras` et libérées en sortie de programme par la procédure `Quitter()` (commande `TTF_CloseFont(maPolice)`).

Nous avons défini une procédure d'écriture : `Ecrire(char *message, int x, int y, int r, int g, int b, bool gras)`. Celle-ci utilise la

commande TTF_RenderText_Solid() fournie par la bibliothèque et qui écrit dans une surface donnée le message passé en argument, dans la police spécifiée et avec la couleur choisie.

```
SDL_Color couleur;  // Structure de couleur fournie par la bibliothèque
SDL_Surface *text = NULL;
couleur.r = rouge;
couleur.g = vert;
couleur.b = bleu;
text = TTF_RenderText_Solid(maPolice, message, couleur);
```

La procédure Ecrire(...) inscrit alors le texte contenu dans message aux coordonnées x, y dans la surface tampon fond, avec la couleur décrite par son triplet RGB (r, g, b) et dans une police grasse ou non suivant la valeur du booléen gras.

Les événements de la souris

SDL permet une gestion aisée des événements de la souris puisqu'il est possible de connaître à chaque instant l'état de cette dernière. Dans notre programme, la fonction qui se charge de reconnaître et d'interpréter l'état de la souris se nomme Gestion_pointeur().

Celle-ci utilise la commande SDL_GetMouseState(&x, &y) pour stocker dans les variables x et y la position absolue de la souris. Pour connaître l'état du bouton gauche de la souris, il suffit dévaluer SDL_GetMouseState(NULL, NULL)&SDL_BUTTON(1).

En fait l'interface utilise la variable pointeur du type donnees_pointeur pour enregistrer ces données. Elle y stocke les coordonnées du pointeur, l'état du bouton (pressé ou non), s'il vient d'être pressé ou relâché, et à quel moment il a été pressé ou relâché (par l'utilisation de SDL_GetTicks() qui renvoie le nombre de millisecondes écoulées depuis l'initialisation de SDL). La structure ne se contente pas uniquement d'enregistrer l'état de la souris, elle factorise en fait un ensemble de données relatives au pointeur : le mode d'édition dans lequel se trouve l'interface, le type de curseur qui représente également certaines actions effectuées par l'utilisateur, l'icône ou le menu éventuellement survolé, quelle moitié de paragraphe est survolé le cas échéant, si l'utilisateur est en train de modifier les dimensions d'une fenêtre ou le numéro d'une page, s'il s'agit de sélectionner ou de désélectionner les paragraphes survolés lorsque l'on laisse glisser le pointeur, les coordonnées relatives du pointeur lorsque l'on édite une fenêtre... Ces données sont mises à jour par d'autres parties du programme que la fonction Gestion_pointeur() et notamment dans les autres procédures possédant le préfixe « Gestion_ ».

La variable pointeur est initialisée par la procédure Initialisation_pointeur().

Les icônes

Les icônes sont représentées par la structure donnees_icones et stockées dans un vector. Elles sont identifiées par leur indice dans le vecteur. Afin de connaître la correspondance, il suffit de se reporter à la procédure

`Initialisation_icones()` qui se charge de créer les icônes et d'en fixer les valeurs initiales.

Une icône dispose de deux positions : une position « caché » qui en interdit l'accès et une position « visible » qui permet de l'utiliser. Le basculement entre ces deux positions est agrémenté d'une petite animation d'apparition ou de disparition qui consiste en fait en une simple translation. Pendant cette phase de transition et lorsqu'elle est cachée, l'icône n'est pas sensible aux actions de l'utilisateur (attribut `ok` à zéro). La vitesse de translation est en fait déterminée par le temps d'apparition/disparition qui détermine la durée de basculement entre les deux positions. Cette durée est calculée à partir du moment où l'ordre d'apparaître ou de disparaître à été donné à l'icône. L'icône est enfoncée si l'utilisateur presse le pointeur dessus ou s'il est indiqué comme étant actif. La structure `donnes_icones` est abondamment commentée à ce sujet.

La procédure `Gestion_icones()` se charge dans un premier temps de déterminer la position de l'icône en fonction de l'ordre qui lui a été donné et du temps. Dans un second temps, et s'il est en position visible, elle teste les actions de l'utilisateur sur celui-ci et agit en conséquence (basculement de modes, création de nouvelle sélection, désignation d'actions, ordre d'apparition d'autres icônes ...). Si le pointeur survole une icône, son indice est enregistré dans la variable `pointeur.sur_quel_icone`, ce qui permet de gérer les priorités lorsque le curseur survole plusieurs objets actifs superposés.

L'affichage des icônes est géré par les procédures `Affichage_icones()` et `Affiche_icone(...)` comme vu précédemment.

Les menus 1 et 2

Les menus 1 (édition de la couleur définie par l'utilisateur) et 2 (*browser* des sélections) sont gérés de manière quasi-identique. Chacun dispose de sa propre structure cependant, car ils diffèrent dans les options qu'ils proposent. Il existe trois positions différentes : une position cachée, une position visible fermée et une position visible ouverte. Le basculement entre ces différentes positions se fait par translation lorsque l'ordre en est donné, comme pour les icônes. Les vitesses de déplacement sont réglées à partir de l'instant où l'ordre est donné par les temps d'apparition/disparition, d'ouverture et de fermeture. Le menu n'est accessible qu'en position ouverte (booléen `ok` à 1). Chacune des structures propose des booléens d'état des différents boutons présents dans le menu correspondant (enfoncé ou non). Lorsque le pointeur est appuyé sur un bouton, celui-ci est enfoncé. L'action correspondante s'active en général lorsque l'on relâche la pression. Ensuite, chacun des menus dispose de propriétés qui lui sont propres : le *browser* conserve le numéro de la première sélection apparaissant dans sa fenêtre de visualisation, l'épaisseur et la position du curseur de l'ascenseur ; le menu de réglage de la couleur définie par l'utilisateur retient les valeurs des taux RGB. Une fois de plus, les structures correspondantes sont abondamment commentées dans le code source : `donnees_menu_1` et `donnees_menu_2`.

Chacun des menus est géré par une procédure : `Gestion_menu_1()` et `Gestion_menu_2()`. Ces procédures sont organisées en trois phases. La première donne les ordres d'apparition ou de disparition suivant l'état de l'interface. La seconde gère les actions de l'utilisateur sur le menu si celui-ci est en position ouverte. Enfin, la troisième anime le menu si celui-ci est en phase de transition. Si le

pointeur survole l'un de ces menus, le numéro du menu est enregistré dans la variable `pointeur.sur_quel_menu`, ce qui permet, au même titre que la variable `pointeur.sur_quel_icone` , de gérer les priorités lorsque le curseur survole plusieurs objets actifs superposés.

L'affichage des menus est assuré par les procédures `Affichage_menu_1()` et `Affichage_menu_2()` comme cela a déjà été précisé. Les valeurs initiales sont établies dans la procédure `Initialisation_menus()`.

Le menu 3

Il s'agit du menu en croix. Celui-ci dispose d'un statut particulier. Il est activé lorsque l'utilisateur édite une sélection et qu'il clique en dehors d'une zone de texte (dans la procédure `Gestion_mode_edition_selection_compils()`). Dans ce cas, la variable `pointeur.sur_quel_menu` est fixée à 3, et le menu est traité prioritairement en ce qui concerne les actions de l'utilisateur, tant que la pression du pointeur reste maintenue. Les icônes qu'il contient se développent en spirale à partir de la position initiale du curseur et jusqu'à ce qu'ils aient atteint leur position maximale. La vitesse de rotation est déterminée par la durée d'apparition, une fois de plus.

Le menu est géré par la procédure `Gestion_menu_3()` qui n'est appelée que lorsque le menu est sollicité par l'utilisateur. Celle-ci s'occupe dans un premier temps de calculer la position de la première icône en fonction du temps d'apparition (celles des autres étant déduites par rotations de 90°), puis, si la position maximale est atteinte, la procédure teste les actions effectuées par l'utilisateur. Elle se contente alors de simuler le clic sur les icônes d'action principales en bas à droite de l'écran.

L'affichage des icônes du menu est assuré par la procédure `Affichage_menu_3()` qui fait alors appel à la procédure `Affiche_icône(...)` déjà mentionnée. Le menu est initialisé avec les autres menus dans la procédure `Initialisation_menus()`.

Le mode « édition des feuilles »

Celui-ci est géré par la procédure `Gestion_mode_edition_feuilles()`. Une fois de plus, le code source est largement commenté. Cependant, nous pouvons présenter rapidement les grandes phases de la procédure afin d'en faciliter la lecture.

Tout d'abord, il s'agit de tester les actions de l'utilisateur sur les entêtes des fenêtres (modification du numéro de la page représentée, suppression de la fenêtre). Ensuite, il faut déterminer si l'utilisateur clique sur une zone sensible de déplacement ou de redimensionnement dans le cas où il n'est pas déjà en train de modifier une fenêtre (booléen `pointeur.edite_feuille` à zéro). Dans ce cas, le curseur prend la forme correspondant à l'action entreprise par l'utilisateur (variable `pointeur.curseur`). Il faut noter que si l'utilisateur est déjà en train de modifier une fenêtre, le curseur conserve sa forme jusqu'à ce que la pression soit relâchée. En effet, c'est la forme du curseur qui est testée dans la suite du programme pour savoir quelle modification est apportée au cadre de la fenêtre. La fenêtre modifiée est retenue grâce à la variable `page_courante`. La procédure `Teste_collision_feuilles(int i, int j)` permet de tester si la nouvelle forme du cadre est compatible avec l'interdiction de superposer les fenêtres. Le

cadre de la fenêtre est alors repositionné en conséquence, de telle sorte qu'il reste tangent aux cadres des fenêtres voisines dans le cas où on tenterait de les traverser. La dernière phase concerne la création d'une nouvelle fenêtre si on clique en dehors d'une zone sensible.

Il est important de se reporter à la description de la structure abstraite pour mieux comprendre le lien de celle-ci avec cette procédure. En effet, l'interface gère ce qui lui paraît comme étant des fenêtres, alors que la structure manipule des pages virtuelles ou réelles.

Le mode « édition des sélections »

La procédure concernée se nomme `Gestion_mode_selection_compils()`.

Tout d'abord, elle détermine quel paragraphe est survolé. Elle considère également la moitié de paragraphe dont il s'agit. Ensuite, il faut prendre en compte deux cas : soit il est question de sélectionner/désélectionner des paragraphes, soit l'utilisateur est en train de définir une ligne d'insertion (ce qui est déterminé par les états des icônes de déplacement et de copie). Dans le premier cas, la première pression détermine si on sélectionne (il y a une sélection courante et le paragraphe pointé n'appartient à aucune sélection), si on désélectionne (il y a une sélection courante et le paragraphe pointé appartient à cette sélection), ou si on change de sélection courante (le paragraphe pointé appartient à une sélection qui n'est pas la sélection courante). Si la pression est maintenue, l'action dépend de l'état du curseur (booléen `pointeur.gomme`) au moment de la première pression : soit il continue à sélectionner, soit il continue à désélectionner. Dans le cas où l'utilisateur est en train de définir une ligne d'insertion, on met à jour sa position tant que le pointeur reste enfoncé.

Dans le cas ou aucun paragraphe n'est pointé, c'est-à-dire que l'on pointe en dehors d'une zone de texte, le menu 3 est déclenché.

Cette procédure est également le lieu privilégié d'échanges entre l'interface et la structure abstraite, du point de vue des sélections cette fois-ci.

Le lien avec le module d'automation de Microsoft Word

L'interface fait appel aux exécutables *structure.exe* et *kompil.exe* au moyen de la commande de gestion de processus `spawnl` (fichier include : *process.h*).

```
spawnl (P_WAIT, "structure.exe", NULL);
spawnl (P_WAIT, "kompil.exe", path.c_str(), NULL);
```

La valeur `P_WAIT` du premier argument indique au processus appelant (l'interface) d'attendre la fin de l'exécution du processus fils (l'exécutable *structure.exe* ou *kompil.exe)* comme un appel classique de sous-programme. La commande permet de plus de transmettre des arguments au processus fils : *kompil.exe* nécessite d'être appelé avec le chemin d'accès complet et le nom du fichier Word édité.

Lors d'une pression sur l'icône d'ouverture de fichier, la procédure `Charge_fichierWord()` est appelée. Celle-ci cède alors la main à l'exécutable *structure.exe* qui présente une fenêtre permettant la recherche du fichier *.doc*. Une fois sélectionné, l'exécutable interprète sa structure par automation de Word, et inscrit le résultat dans le fichier *structure.txt*, sans omettre de préciser en entête le chemin d'accès complet et le nom du fichier étudié. Ainsi, l'interface ayant repris la

main étudie le fichier *.txt* qu'elle est capable d'interpréter : elle stocke le chemin d'accès pour pouvoir le fournir à *kompil.exe* et met à jour les données dans la structure abstraite.

Lors de l'enregistrement du résultat, la formation du texte est lancée, et l'ordre de réagencement est stocké dans le fichier *kompil.txt*. Puis l'exécutable correspondant est appelé avec les bons paramètres et se charge d'interpréter le contenu de *kompil.txt* pour créer le nouveau fichier *.doc*. La première ligne du fichier *kompil.txt* indique à l'exécutable s'il doit imprimer (1) ou non (0) le fichier résultant. Elle est éditée en conséquence selon que l'utilisateur clique sur l'icône d'impression ou d'enregistrement simple.

La page de présentation

La procédure `Page_ouverture()` est appelée au démarrage afin d'afficher la page d'accueil. La procédure `Fondu(int i)` est utilisée lors de la transition entre cette page et l'image de fond du logiciel. Elle grossit les pixels de l'image initiale en appliquant un filtre de couleur bleu. La grosseur des carrés obtenus est réglée par le paramètre *i*. Il ne s'agit que d'un artifice afin de rendre plus attrayante l'entrée dans le logiciel.

Annexe F : Consignes des tests pour l'expérimentateur

Consignes pour TableGate

Données oralement aux participants par l'observateur de TableGate.

Attention : Lors de l'explication des opérations (icônes, menus) : on peut expliquer comme on le souhaite, il faut que chacun ait bien compris. Pour les explications sur les objectifs du logiciel et les consignes pour le test M, il faut s'en tenir à ces consignes (ne pas dire par exemple : "c'est pratique parce qu'on peut soulever les feuilles pour les voir de plus près et les faire passer")

1) Explication rapide du dispositif

Projecteur, table tactile qui fonctionne comme une souris. Realité augmentée.

2) Présentation rapide du logiciel

- Logiciel de traitement de texte : remaniement de la structure d'un texte.
- Texte sous forme .txt dans l'ordinateur, imprimé.
- Les feuilles imprimées sont posées sur la table.
- On travaille sur les feuilles papier posées sur la table, l'ordinateur fait les modifications sur le fichier Word.
- Unité d'édition : le paragraphe.

3) Etapes du travail sur un texte

- disposer les feuilles : dire à l'ordinateur où sont les feuilles
- sélectionner un paragraphe : dire ce sur quoi on veut travailler
- effectuer une modification
- compilation : effectue les changements dans le texte Word

4) Explication des opérations à partir de l'exemple de type JP

a) Disposition des feuilles (ne pas appuyer trop vite, possibilité de verrouillage)
b) Selection (réappuyer à chaque fois)
c) Opérations : description des icônes, du menu déroulant, du browser. Insister sur la manière d'annuler une sélection ou une opération en cas d'erreur.
d) Compilation

5) Consignes pour le test type M

Un petit texte est distribué aux participants en même temps que la partie qu'ils doivent lire. On les fait s'asseoir devant une table pour le lire, des stylos sont disponibles.

Le membre du projet est présent pour toutes questions d'ordre techniques uniquement, et ce, seulement en cas de blocage. (pas de conseils sur la façon de se placer autour de la table , ect...). Mentionne la possibilité d'une réimpression.

Annexe G : Consignes des tests pour les sujets

Consignes à l'intention des participants

Organisation du test :

Ce test se déroule par équipes de trois. Chacun lit une partie d'un texte (5 pages au total), qui est un sketch de Bigard commençant par "J'ai entendu à la radio c'matin..." et qui a pour sujet une chauve-souris enragée. Malheureusement, ce texte a été mélangé et des parties "parasites" ont été ajoutées.

Le texte qui vous a été distribué est à vous, faites-en ce que vous voulez.

Objectif :

A l'aide du dispostif TableGate, retrouver le texte initial !

Durée :

Le test devrait prendre une vingtaine de minutes. Ce n'est pas important d'aller vite, on ne vous arrêtera que si ça s'éternise.

Aide :

Un membre du projet sera présent pour répondre à toutes les questions d'ordre technique.

Au travail !

Annexe H : Fiche d'observation utilisée pendant les tests

Fiche d'observation

Date : Equipe : Noms :

1) Equipe :
- a) expérience en informatique
- b) expérience du travail - à plusieurs
  - ensemble

2) Temps :
- disposer les feuilles ($2^{ème}$)
- début du travail ($2^{ème}$)
- première compilation achevé ? ($2^{ème}$)

3) Utilisation du dispositif :
- a) positionnement
- b) statique
- c) gêne / accès facile aux fonctions
- d) réalité augmentée - annotations
  - déplacement de feuilles

4) Utilisation du logiciel :

a) manipulation :
- icônes
- menu déroulant
- browser
- membre PE / mémo

b) fonctionnalités :
- sélections « avancées »
- « aucune action »
- annulation

c) erreurs fréquentes

5) Méthodologie :
- a) première action effectuée
- b) rédaction linéaire

6) Travail collaboratif :
- a) communication - active
  - de tous
- b) énervements
- c) parties étrangères - survolées
  - lues
  - résumés

d) manipulation partagée - dès le départ ?
- équilibre
e) accidents de manipulation simultanée

Annexe I : Questionnaire rempli à l'issue de l'expérience

Petit questionnaire sur TableGate

1) L'utilisation du logiciel est facile et intuitive :

pour les icônes en bas de fenêtre ? _ oui _ non

pour les menus déroulants ? _ oui _ non

pour le browser ? _ oui _ non

2) Les couleurs et icônes forment-ils un ensemble lisible et intuitif ?

_ oui _ non

3) Au bout de combien de temps est-on à l'aise pour travailler de façon autonome sur le dispositif ?

_ après les explications sur le premier type de texte

_ après avoir travaillé sur les deux textes

_ il faudrait plus de temps

4) Combien de temps pensez-vous avoir gagné par rapport à une équipe qui n'aurait eu qu'un ordinateur ?

Le temps de se familiariser au dispositif est-il amorti au bout d'une seule utilisation ? _ oui _ non

5) Le logiciel facilite-t-il :

le travail à plusieurs ? _ oui _ non

le travail sur des textes longs ? _ oui _ non

6) Voyez-vous un intérêt au fait d'utiliser des feuilles papier ?

7) Ce dispositif vous semble-t-il utile ? Dans quel type d'applications ?

8) Avez-vous d'autres commentaires sur le travail sur ce dispositif, ou des suggestions pour améliorer le logiciel ?

Annexe J : Textes utilisés pour les expériences

Texte pour s’habituer au dispositif :

un un
trois trois
huit huit
quatre quatre
cinq cinq
six six
sept sept
neuf neuf
deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux

deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux deux
dix dix
bouze bouze
teuf teuf

Texte principal à remettre dans l’ordre

Sketch de Bigard : la chauve-souris enragée
Il s’imagine comment, dans le pire des cas, il pourrait risquer de se faire attaquer chez lui par une chauve-souris enragée

J'ai entendu à la radio c' matin :"On a une chance sur dix millions de se faire mordre par une chauve-souris enragée". Faut quand même savoir qu'il y a un gars, qui s'est sûrement cassé le cul pendant des mois et des mois (avec du matériel et tout !) pour arriver à la conclusion suivante : "On a une chance sur dix millions de se faire mordre par une chauve-souris enragée". Alors moi, j'suis pas curieux, mais j'aurai quand même bien voulu savoir comment il a fait, le gars, pour déterminer qu'on avait TOUS - en moyenne - une chance sur dix millions de se faire mordre.
Je ne conterai point notre histoire. L’amour n’en a qu’une, toujours la même. Je l’avais rencontrée et aimée. Voilà tout. Et j’avais vécu pendant un an dans sa tendresse, dans ses bras, dans sa caresse, dans son regard, dans ses robes, dans sa parole, enveloppé, lié, emprisonné dans tout ce qui venait d’elle, d’une façon si complète que je ne savais plus s’il faisait jour ou nuit, si j’étais mort ou vivant, sur la vieille terre ou ailleurs.

Bon, mais, admettons, admettons, elle pousse la porte. Ben, vous allez quand même pas m' dire qu'elle va monter les cinq étages à pied, comme ça (grimace de la chauve-souris enragée), sans se faire repérer par quelqu'un, NON ??? J'y crois pas, j'y crois pas ! Honnêtement, j'y crois pas...

Ben oui, parce que bon, moi, j' vois, j'habite dans le dixième. Bon, déjà, le dixième, c'est un quartier qui question chauve-souris est assez tranquille, il faut le reconnaître...

Si, si, si, c'est tranquille. En plus, j'habite au cinquième sans ascenseur, j' vais t' dire, j'ai un code à la porte en bas et un interphone. Ha, ha ! Alors, j' vais t' dire, la chauve-souris, elle peut toujours se pointer, tu vois ! Bon, mais admettons, admettons.

Quand la nuit fut noire, très noire, je quittai mon refuge et me mis à marcher doucement, à pas lents, à pas sourds, sur cette terre pleine de morts. J’errai longtemps, longtemps, longtemps. Je ne la retrouvais pas. Les bras étendus, les yeux ouverts, heurtant des tombes avec mes mains, avec mes pieds, avec mes genoux, avec ma poitrine, avec ma tête elle-même, j’allai sans la trouver. Je touchais, je palpais comme un aveugle qui cherche sa route, je palpais des pierres, des croix, des grilles de fer, des couronnes de verre, des couronnes de fleurs fanées ! Je lisais les noms avec mes doigts, en les promenant sur les lettres. Quelle nuit ! quelle nuit ! Je ne la retrouvai pas !
Arrive une chauve-souris. Enragée. Elle est là, tu vois, elle bave, elle est enragée. Elle est là : "Argglbrrll..." Bon... Hop ! Elle s'fout à trifouiller mon code. Elle est là, elle trifouille. Elle cherche, elle cherche.
Si, si, Monsieur, quand elles sont enragées, elles cherchent quand même ! Elle est là, elle cherche, elle cherche...

J'y tire dessus, même si j'la loupe, eh ben, elle a peur et puis elle s'envole par la fenêtre. Alors... Quand j' pense à L'AUTRE C.., là, qui fout les j'tons à tout l' monde avec ses chauve-souris... En plus il est dangereux c' mec là, hein ? Imagine : j'ai un copain qui arrive à la maison déguisé en chauve-souris pour me faire rigoler. Eh ben, J'Y TIRE DESSUS !!! Parfaitement !!! J'ai des copains qui viennent chez moi, à la maison, déguisés en chauve-souris... Pour me faire rigoler. Alors, hé, qu'il arrête, L'AUTRE LA, avec ses chauve-souris, hein... J'ai PEUR des chauve-souris...

Un dispositif utilisant l'idée de « réalité augmentée » permet de mélanger objet réels (appartenant au monde physique) et virtuels (créés numériquement). A la différence de la « réalité virtuelle », l'utilisateur n'est pas plongé dans un monde complètement numérique. Il est entouré des objets dont il a l'habitude, par exemple du papier, des crayons, mais ceux-ci sont enrichis d'une composante électronique, ils peuvent communiquer avec un ordinateur et utiliser les avantages de la technologie et du travail sur ordinateur. L'avantage principal est que l'utilisateur n'est pas déstabilisé par un monde inconnu, mais peut travailler comme il l'a toujours fait, et avec des outils dont le maniement est intuitif. Le dispositif de table interactive installé à l'ICTT et dont nous nous servons pour le projet Tablegate est adapté à des applications utilisant ce concept. En effet, la grande taille et la position horizontale de la table permettent d'y poser des objets bi- ou tridimensionnels. D'autre part la table est munie d'interfaces lui permettant de communiquer avec un ordinateur : un projecteur pour l'entrée d'informations ; une surface tactile et bientôt une caméra pour la sortie.

Ben oui, parce que bon... Admettons qu' le gars soit très copain avec une bande de chauve-souris enragées, il les connaît depuis longtemps, il les a déjà dépannées quand elles étaient dans la merde et tout... mon vieux ! elles lui doivent tout, elles lui cachent rien, ils ont un degré d'intimité ensemble, mais bon...

Le fleuve était parfaitement tranquille, mais je me sentis ému par le silence extraordinaire qui m'entourait. Toutes les bêtes, grenouilles et crapauds, ces chanteurs nocturnes des marécages, se taisaient. Soudain, à ma droite, contre moi, une grenouille coassa. Je tressaillis : elle se tut ; je n'entendis plus rien, et je résolus de fumer un peu pour me distraire. Cependant, quoique je fusse un culotteur de pipes renommé, je ne pus pas ; dès la seconde bouffée, le cœur me tourna et je cessai. Je me mis à chantonner ; le son de ma voix m'était pénible ; alors, je m'étendis au fond du bateau et je regardai le ciel. Pendant quelque temps, je demeurai tranquille, mais bientôt les légers mouvements de la barque m'inquiétèrent. Il me sembla qu'elle faisait des embardées gigantesques, touchant tour à tour les deux berges du fleuve ; puis je crus qu'un être ou qu'une force invisible l'attirait doucement au fond de l'eau et la soulevait ensuite pour la laisser retomber. J'étais ballotté comme au milieu d'une tempête ; j'entendis des bruits autour de moi ; je me dressai d'un bond : l'eau brillait, tout était calme.

D' t'tes façons, j'ai fait mettre un judas, moi, à ma porte. Alors... Bon ! Elle arrive sur le pallier... Qu'est ce qui prouve qu'elle va venir frapper à MA porte ? Y'a quatre appartements sur le pallier...

Pour établir le cahier des charges correspondant à un tel logiciel, nous avons en premier lieu réfléchi sur le travail d'édition de texte, et plus particulièrement dans un contexte de travail collaboratif. Nous avons conclu de cette réflexion que l'unité minimale d'édition dans une telle situation était le paragraphe, puisque les utilisateurs se trouveraient typiquement dans une situation où ils souhaitent remanier le texte dans ses grandes lignes, la correction de l'orthographe ou le travail de style étant des activités plutôt individuelles, ne nécessitant pas de toute façon d'avoir une vue d'ensemble du texte. D'autre part, des considérations techniques sur la précision du matériel à notre disposition nous ont fait nous limiter au paragraphe. Nous avons choisi pour notre logiciel les actions habituelles des traitements de texte, à savoir la suppression, la copie et le déplacement. Le traitement de texte retenu a été Word, parcequ'il est le plus couramment utilisé.

Qu'est ce qui prouve qu'elle va me sauter tout de suite à la gorge ??? Dis donc ?! Elle s'rait pas un p'tit peu fatiguée cette chauve-souris, depuis tout c' qu'elle a fait tout à l'heure ? Elle a p't être soif ? Bon, ben moi j' l'installe dans la banquette. Je fais mine d'aller y chercher un verre d'eau dans la cuisine. Qu'est ce qui m'empêche de revenir avec un FUSIL ?!!! Ben alors ! Parcequ'elle se méfie pas... C'est quand même qu'une bête !!! Alors ?

« Heu, dis donc machin ! ... j'ai rêvé ou t'as bousculé ma femme, là? Laisse Madeleine. Qu'est-ce t'as, t'es bigleux, tu veux mes lunettes?... Dans la gueule? Elle est trop petite, t'as pas fait gaffe, t'as marché dessus? Comment? Elle est largement assez grosse? Ma femme est grosse? Laisse Madeleine. Alors pour marcher dessus on est sympa, mais après pour dire pardon on ferme sa gueule. Comment? Excusez-moi madame. Ah! ah! ah! ... mais il répond l'effronté! Mais tu joues avec ta santé, là. C'est ta vie que t'as entre tes doigts, t'as vu combien ça pèse ça ! avais t'l'envoyer à travers la gueule, ma parole, si j'te rate rien qu'avec le vent j't'enrhume ! Allez casse-toi casse-toi hein. Moi un mec y m'aurait dit la moitié d'ça, j'lui aurais foutu mon poing dans la gueule, le pif comme ça, t'aurais été obligé d'te faire greffer une brouette, là. Laisse Madeleine.

Bon ! Elle frappe à ma porte. Ben j'ai pas d' pot jusqu'à maintenant, hein ? Parce que là, manifestement, c'est moi qu'elle veut mordre. Là, elle m'a choisi dans l'immeuble, elle veut plus mordre personne d'autre : c'est moi ! Bon ! J' vais jusqu'au bout : J'ouvre !

Le brouillard qui, deux heures auparavant, flottait sur l'eau, s'était peu à peu retiré et ramassé sur les rives. Laissant le fleuve absolument libre, il avait formé sur chaque berge une colline ininterrompue, haute de six ou sept mètres, qui brillait sous la lune avec l'éclat superbe des neiges. De sorte qu'on ne voyait rien autre chose que cette rivière lamée de feu entre ces deux montagnes blanches ; et là-haut, sur ma tête, s'étalait, pleine et large, une grande lune illuminante au milieu d'un ciel bleuâtre et laiteux.

Moi j' dis qu' c'est un peu facile de foutre les jetons à tout l'monde, avec, finalement, la SEULE chance qu'on a de s'faire mordre, en occultant volontairement - excusez moi du peu - les 9 millions 999 mille 999 autres chances qu'on a de PAS se faire mordre. Donc, de pas mourir dans d'atroces souffrances...

J'étais là debout, frémissant, les yeux fixés sur le verre, sur le verre plat, profond, vide, mais qui l'avait contenue tout entière, possédée autant que moi, autant que mon regard passionné. Il me sembla que j'aimais cette glace, -je la touchais, -elle était froide ! Oh ! le souvenir ! le souvenir ! miroir douloureux, miroir brûlant, miroir vivant, miroir horrible, qui fait souffrir toutes les tortures ! Heureux les hommes dont le cœur, comme une glace où glissent et s'effacent les reflets, oublie tout ce qu'il a contenu, tout ce qui a passé devant lui, tout ce qui s'est contemplé, miré, dans son affection, dans son amour ! Comme je souffre !

Pan ! Elle tombe sur mon code ! Faut encore qu'elle imite la voix d'un gars que j' connais !!! Sinon, j'ouvre pas, moi ! Bon... J'étais un peu bourré la veille, j'ai pas bien dormi, c'est pas la question. J'ouvre ! Faut encore qu'elle "pousse" la porte... C'est grand comme ça une chauve-souris...

Annexe K : Notice d'utilisation du programme "capture.exe"

IMPORTANT:
Ce programme nécessite pour son exécution:
-La présence du fichier ezvidc60.ocx, dans le même répertoire.
-Un périphérique d'acquisition vidéo (webcam...)

PRINCIPE:
Le but de ce programme est de capturer une image du périphérique vidéo, et de la sauvegarder.

L'utilisation est simple. "capture.exe" se lance sans paramètre, capture l'image courante du périphérique vidéo par défaut et la sauvegarde au format bmp sous le nom « cap.bmp » dans le même répertoire.

REMARQUE:
Si le répertoire contient déjà un fichier « cap.bmp » lors de l'exécution de « capture.exe », celui-ci sera écrasé.
Si il n'ya pas de périphérique vidéo installé, cela ne fonctionnera pas (renvoie un message d'erreur et ferme).

Ce programme est basé sur l'excellent ActiveX « ezvidcap.ocx » (qui permet aussi de faire de la capture vidéo) :

```
'*********************************************************************
'*  ezVidC60.ocx for VB5 / VB6
'*
'*  A full featured video capture control for VisualBasic
'*  By Ray Mercer <raymer@shrinkwrapvb.com>
'*
'*  This component is based on the VBVidCap.exe project which is
'*  freely downloadable from http://www.shrinkwrapvb.com
'*  This component was written entirely in Visual Basic 5.0 SP3.
'*
'*  This version of the ezVidCap.ocx was compiled in VB6 (no Service Pack)
'*  and named ezVidC60.ocx.  The
'*
'*  ezVidCap should be compatible with any Video For Windows
'*  capture device.  Comments, suggestions, and bug reports can be
'*  sent to raymer@shrinkwrapvb.com
'*
'*  Dependencies:
'*  Visual Basic 6.0 runtime
'*
'*  Copyright (c) 1998-2000, Ray Mercer.  All rights reserved.
'*********************************************************************
```

Consulter le fichier ezvidc60.ocx.txt pour plus d'info à ce sujet.

Annexe L : Notice d'utilisation du programme "kompil.exe"

IMPORTANT:
Ce programme nécessite pour son exécution:
-La présence du fichier word_ocx.ocx, dans le même répertoire (ou au pire dans le sur-répertoire).
-Une Version de Word compatible avec Word 2000.
-Que Word soit fermé !

PRINCIPE:
Le but de ce programme est de mettre dans l'ordre souhaité les paragraphes d'un document word, en les dupliquant éventuellement.

Le document à mettre en ordre se spécifie dans une chaîne de caractères passée en paramètre de la ligne de commande lors de l'appel de "kompil.exe".
Le nouvel ordre se lit dans un fichier binaire nommé "kompil.txt" qui se trouve dans le même répertoire que le fichier "kompil.exe". Celui-ci doit contenir la liste ordonnée des paragraphes dans l'ordre du nouveau document (1 numéro de paragraphe par ligne, voir exemple). La 1ere ligne indique au programme si il faut imprimer le document compilé (0->pas d'impression, 1->impression).
Si le même paragraphe apparaît plusieurs fois il sera dupliqué.
L'exécution correcte de "kompil.exe" va créer un nouveau fichier dont le nom est celui du fichier d'origine prolongé de "_tag.doc" et qui se trouvera dans le même répertoire que ce dernier.

EXEMPLE:
Par exemple, si le fichier "c:\docs\document.doc" contient 10 paragraphes, que l'on souhaite inverser leur ordre, dupliquer le 1er à la fin et imprimer le résultat :
"Structure.txt" doit contenir :

1
10
9
8
7
6
5
4
3
2
1
1

Et on doit appeler kompile.exe ainsi

kompile.exe c:\docs\document.doc

Le résultat est un fichier c:\docs\document_tag.doc, dont l'ordre des paragraphes est l'inverse de "c:\docs\document.doc", et dont le dernier paragraphe (anciennement le 1er) apparaît 2 fois à la suite.

REMARQUES:
-Il n'est pas obligatoire d'utiliser tous les paragraphes du fichier Word d'origine. Un fichier "kompil.txt" contenant
0
5
5
créera un fichier de 2 paragraphes identiques (le 5ème paragraphe du fichier d'origine), et ne l’imprimera pas.

-Si on indique un numéro de paragraphe n'existant pas dans le fichier d'origine le programme plantera.

-En cas de plantage, Word qui est ouvert en arrière plan ne se fermera pas. Comme il est en mode invisible, la solution est de le fermer en terminant le processus, à l'aide d'un bon vieux Ctrl-Alt-Suppr.

-Si on choisit d'imprimer le résultat, c'est l'imprimante par défaut du système qui s'en chargera dans le cas où plusieurs imprimantes sont installées. Si aucune imprimante n'est installée, le programme plantera.

Annexe M : Notice d'utilisation du programme "structure.exe"

IMPORTANT:
Ce programme nécessite pour son exécution:
-La présence du fichier word_ocx.ocx, dans le même répertoire (ou au pire dans le sur-répertoire).
-Une Version de Word compatible avec Word 2000.
-Que Word soit fermé !

PRINCIPE:
Le but de ce programme est d'obtenir les informations suivantes sur les paragraphes d'un document word:
-page de son 1er mot
-distance séparant son 1er mot du haut de sa page, en millième de page A4
-page de son dernier mot
-distance séparant son dernier mot du haut de sa page, en millième de page A4

Le document à analyser se choisit par la boite de dialogue qui s'affiche.
Les informations utiles se stockent dans un fichier binaire nommé "structure.txt", qui se trouve dans le même répertoire que le fichier "structure.exe" et dont la structure est la suivante:

Première ligne : chemin d'accès complet du fichier Word

Lignes suivantes : pour chaque paragraphe, sur une ligne
-une chaîne de caractères sans espace, contenant "ocx_power" (à l'heure actuelle, ceci ne sert à rien sauf de montrer l'enthousiasme du programmeur, mais permet de laisser un champ disponible en cas d'évolution future du programme).
-page du premier mot.
-distance de ce mot au haut de la page.
-page du dernier mot.
-distance de ce mot au haut de la page.

EXEMPLE :
Exemple de fichier "structure.txt"

C:\tablegate\test.doc
ocx_power 1 83 1 100
ocx_power 1 100 1 116
ocx_power 1 116 1 132
ocx_power 1 132 1 150
ocx_power 1 150 1 166
ocx_power 1 166 1 182
ocx_power 1 182 1 199
ocx_power 1 199 1 215
ocx_power 1 215 1 231
ocx_power 1 231 1 248
ocx_power 1 248 1 264

ocx_power 1 264 1 280
ocx_power 1 280 1 297
ocx_power 1 297 1 313
ocx_power 1 313 1 329
ocx_power 1 329 1 346
ocx_power 1 346 1 363
ocx_power 1 363 1 379
ocx_power 1 379 1 395
ocx_power 1 395 1 412
ocx_power 1 412 1 428
ocx_power 1 428 1 444
ocx_power 1 444 1 461
ocx_power 1 461 1 477
ocx_power 1 477 1 493
ocx_power 1 493 1 510
ocx_power 1 510 1 526
ocx_power 1 526 1 543
ocx_power 1 543 1 559
ocx_power 1 559 1 576
ocx_power 1 576 1 592
ocx_power 1 592 1 608
ocx_power 1 608 1 625

REMARQUES:
-En cas de plantage, Word qui est ouvert en arrière plan ne se fermera pas. Comme il est en mode invisible, la solution est de le fermer en terminant le processus, à l'aide d'un bon vieux Ctrl-Alt-Suppr.
-Si dans le même répertoire que "structure.exe" se trouve déja un fichier "structure.txt", ce dernier sera écrasé.