N ° d'ordre: 688.

THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

pour obtenir le titre de

DOCTEUR

En Productique: Automatique et Informatique Industrielle.

par

TREHOU Guy

Maître ès Sciences

CONCEPTION ASSISTEE DE COMMUNICATEURS POUR HANDICAPES A L'AIDE D'UN LANGAGE ORIENTE OBJET: ACCES DESORDONNE A UN DICTIONNAIRE PAR RECHERCHE BINAIRE

Soutenue le 5 février 1991 devant la Commission d'Examen.

JURY:	Président:	Pierre Vidal	Professeur à Lille 1
	Rapporteurs:	Pierre Vidal	Professeur à Lille 1
		Jacques Bremont	Professeur à Nancy 1
	Membres:	Brigitte Baudel-Cantegrit	Maître de Conférences à Lille 1, codirecteur du travail
		Jean Marc Toulotte	Professeur à Lille 1, codirecteur du travail
		Alain Chapoton	Professeur à Lille 1
		Noël Malvache	Professeur à Valenciennes
	Membre invité:	Jean-Claude Moreau	Directeur du développement, Handisoft International

AVANT PROPOS

La réalisation d'un générateur de communicateur permettant d'adapter les communicateurs à tout type d'handicap, dont cette thèse fait l'objet, est le résultat d'un travail d'équipe effectué en collaboration avec monsieur TICHON Jacques, sous la supervision, les conseils et les directives du professeur TOULOTTE et de madame BAUDEL-CANTEGRIT.

Sauf en ce qui concerne le chapitre III relatif à la méthode d'accès à un dictionnaire par matrice binaire, qui est entièrement personnel, il est difficile de préciser la part prise par chaque membre de l'équipe dans le produit final.

Cette thèse a le mérite de rassembler dans un tout cohérent le travail de chacun.

Nous remercions, en conséquence, le professeur TOULOTTE et madame BAUDEL-CANTEGRIT, dont la compétence et le savoir nous ont été très utiles dans la conduite de ce travail.

TABLES DES MATIÈRES

INTRODUCTION GÉNÉRALE

CHAPITRE I

NOTIONS GENERALES SUR L'AIDE A LA COMMUNICATION POUR HANDICAPES

CHAPITRE II

CONCEPTION ASSISTÉE D'UN COMMUNICATEUR POUR HANDICAPÉS A L'AIDE D'UN LANGAGE OBJET.

CHAPITRE III

COMMUNICATEUR BLISS UTILISANT LA MÉTHODE BINAIRE COMME MÉTHODE D'ACCÈS À UN DICTIONNAIRE

ANNEXE

BIBLIOGRAPHIE

INTRODUCTION GENERALE

Nous vivons parmi les handicapés. Il faut en effet savoir qu'une personne sur douze est handicapée à des degrés divers. En Europe, on estime à un million le nombre de traumatisés crâniens suite à des accidents de la route ou des accidents de travail au sens large. Les uns se rétablissent. D'autres, par contre, ne recouvrent jamais plus leurs capacités. Pour ces handicapés, l'aide de l'informatique peut se révéler très intéressante.

L'aide à la communication, pour un grand nombre d'handicapés, et en particulier pour ceux qui sont privés de la parole, est une nécessité essentielle:

- soit pour pouvoir simplement exprimer des besoins élémentaires,
- soit pour atteindre un niveau convenable d'interaction avec un environnement humain ou matériel,
- soit enfin pour pouvoir accéder aux outils puissants et peu onéreux que sont les micro-ordinateurs et ainsi s'insérer dans le milieu éducatif et même professionnel.

Depuis de nombreuses années des chercheurs, sous la direction du professeur Toulotte, étudient des systèmes de communication pour handicapés.

Ces études ont commencé par la recherche d'une structure de communicateur qui réponde aux besoins et ensuite par la vérification de cette structure grâce à la réalisation pratique de communicateurs spécifiques.

Il est ensuite apparu la nécessité de développer un système de génération automatique de communicateurs adapté à tous les types d'handicap.

Un premier générateur a été développé au sein du centre d'automatique de l'université (projet GENEPIC). Ce générateur nous a permis de nous rendre compte que pour être effectivement intéressant, il faut absolument qu'il se présente sous la forme d'un système ouvert. Cette structure permet, en effet, de faire évoluer ou de modifier le générateur en fonction de nouveaux besoins.

Nous avons donc réalisé une plate-forme de prototypage de communicateurs telle que son évolution soit aisée. Nous avons, pour cela, choisi de développer cette plate-forme suivant la méthodologie objet.

Nous avons vérifié la possibilité d'évolution du générateur en lui ajoutant un module qui permet la transformation d'un message écrit dans un langage adapté aux handicapés (Bliss) en français correct.

Le système présenté, très convivial permet à une personne, non spécialisée en informatique (l'opérateur), de réaliser très rapidement le prototypage d'un communicateur. Ce prototype utilisé par l'handicapé (l'utilisateur) est alors évalué, modifié afin d'obtenir une adaptation parfaite du communicateur aux possibilités de l'utilisateur.

L'opérateur dispose d'un micro-ordinateur courant. Il initialise, de manière interactive, un ensemble de variables, qui caractérisent le communicateur à construire. Le logiciel de génération du système de communication produit le communicateur adapté, ainsi que la description du matériel nécessaire.

L'exposé de notre solution comporte trois parties.

Dans un premier temps, il est fait un exposé général sur l'aide à la communication des handicapés.

Dans un deuxième temps, nous décrivons le fonctionnement du générateur.

Enfin, dans un troisième temps, nous étudions la façon de transformer les éléments d'information fournis par l'handicapé en un français correct.

CHAPITRE I

NOTIONS GENERALES SUR L'AIDE A LA COMMUNICATION POUR HANDICAPES

I.1.INTRODUCTION.

Dans ce chapitre, nous décrivons d'une manière générale un communicateur pour handicapés afin d'en déduire les critères auxquels doit nécessairement obéir un système de communication.

I.2. DESCRIPTION D'UN COMMUNICATEUR.

I.2.1. Description générale

Vu de l'extérieur, un système d'aide à la communication pour handicapés sert d'intermédiaire de communication entre l'utilisateur et l'environnement. Il a pour mission de "dialoguer" d'une part avec l'utilisateur (interaction handicapé-communicateur) et d'autre part avec l'environnement (interaction communicateur-environnement), comme le montre la figure I.1.

Par l'intermédiaire du communicateur, l'utilisateur construit son message à partir d'éléments informatifs (mot, syllabe, caractère Bliss...), grâce à une interface d'accès et grâce à une méthode de sélection de ces éléments parmi un ensemble de base prédéfini.

Dans le même temps, le communicateur visualise sur un dispositif d'affichage le message en cours de construction et facilite à l'utilisateur l'accès aux éléments les plus probables. Le communicateur peut éventuellement transmettre à l'utilisateur des messages envoyés par l'environnement.

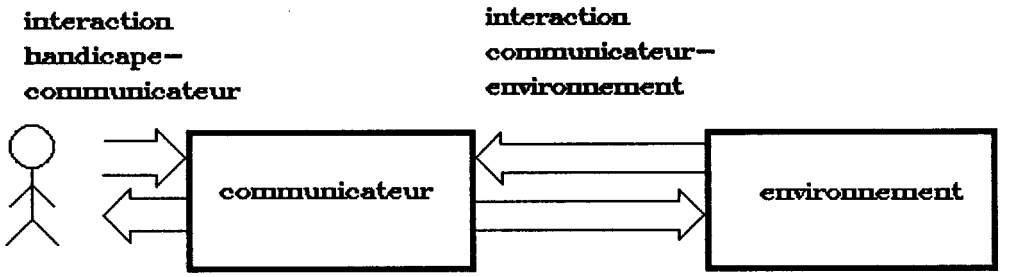


figure I.1

I.2.2. Composition d'un communicateur [TOU.87].

Le concept général retenu comporte quatre niveaux (figure I.2):

- un niveau organes d'accès et méthodes d'accès de l'handicapé à l'ordinateur. Il permet l'acquisition et le codage des interactions,
- un niveau de réalisation d'un message abstrait primaire. On y mémorise les messages à partir d'éléments de base à sélectionner et diverses informations. On accélère la composition des messages à partir des éléments de base,
- un niveau de réalisation d'un message abstrait secondaire. On transforme les messages de la forme primaire en une forme compréhensible par l'environnement,
- un niveau organe de sortie où l'on émet les messages transformés vers l'environnement.

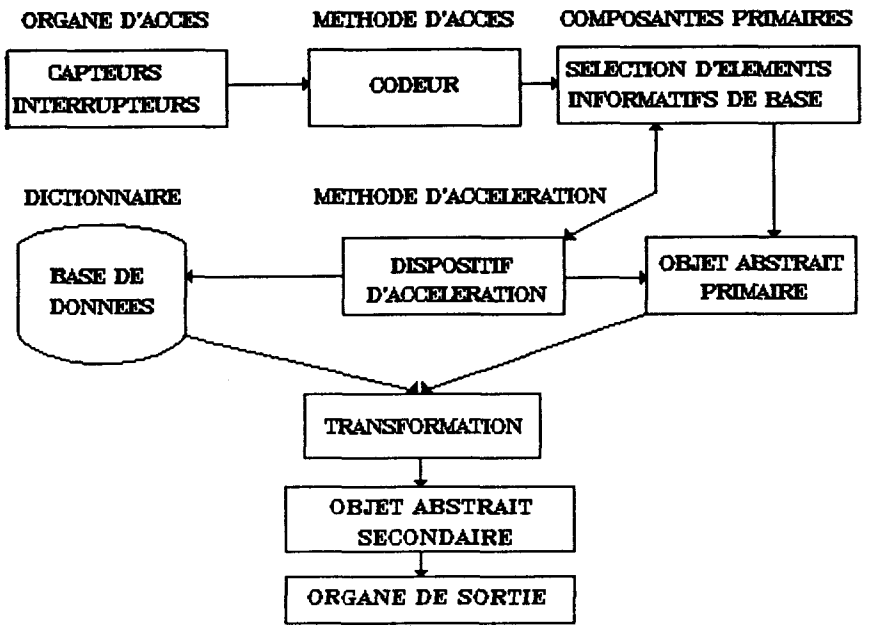


figure I.2

I.2.2.1. Niveau organes et méthodes d'accès.

Dans l'aide à la communication, l'interaction entre le communicateur et l'handicapé se réalise par visualisation sur un dispositif d'affichage d'informations qui pourront être sélectionnées par l'handicapé grâce aux organes et méthodes d'accès.

On utilise habituellement, dans le cas des handicapés voyants, un seul écran à tube cathodique.

Le niveau "organes et méthodes d'accès" comprend différents types de capteurs, ainsi que des moyens de repérage et éventuellement de codage de l'information.

Les organes d'accès doivent être parfaitement adaptés à chaque type d'handicap. Il faut pouvoir transformer toutes les grandeurs physiques contrôlables par l'handicapé en signaux de commande du communicateur [MYE.82][ROS.72].

Parmi les organes d'accès, on distingue:

- le tout ou rien unique,
- le multi tout ou rien,
- la commande positionnelle.

I.2.2.1.1. Les organes d'accès

Le tout ou rien unique

C' est soit un interrupteur, soit un capteur associé à un circuit à seuil. La commande est exercée par un geste simple (avec un doigt, la main, le coude, le pied, la contraction d'un muscle...).

Le multi tout ou rien

Il comporte, soit toutes les variétés de claviers du type AZERTY, soit un capteur associé à plusieurs seuils, qui réagit à différents niveaux de la grandeur physique mesurée.

La commande positionnelle

Elle est constituée d'un capteur associé à un circuit convertisseur analogique digital, qui fournit l'information directement proportionnelle à la grandeur physique mesurée. Les grandeurs physiques mesurées par les capteurs sont diverses: la lumière, le déplacement d'une partie fixe par rapport à une partie mobile, la pression, l'humidité, le bruit. La souris est de plus en plus utilisée. Elle ne peut toutefois être utilisée que par les handicapés capable de maîtriser son déplacement.

Les organes et méthodes d'accès ont pour mission de choisir des composantes primaires par une sélection d'éléments informatifs de base. La visualisation d'un élément est appelée "pavé", car elle est généralement de forme carrée ou rectangulaire.

I.2.2.1.2 Les méthodes d'accès [TOU.83][ROS.82][MAN.85][THO.81]

Les méthodes d'accès à l'information sont des méthodes de sélection permettant le choix d'un élément informatif de base (pavé) parmi un ensemble de pavés regroupés en menus.

Il existe diverses méthodes de sélection. On peut les classer dans les catégories suivantes:

- sélection directe sans codage (sélection positionnelle),
- sélection avec codage,
- sélection par balayage.

1)La sélection directe sans codage

Ce type de sélection ne demande aucun codage intermédiaire. L'utilisateur sélectionne un pavé en le pointant avec un joy-stick ou une tige buccale par exemple.

Cette méthode permet la sélection d'un pavé à chaque action sur l'interface d'accès. Le temps de sélection est très court, mais nécessite de la part de l'handicapé un effort physique et mental important, ainsi qu'une grande maîtrise de ses mouvements.

2)La sélection avec codage

L'utilisateur choisit un pavé en composant le code correspondant. Ainsi:

- un mono-interrupteur effectue un codage direct (morse).

- un multi-interrupteur deux touches permet:

  - la commande de balayage et la validation (arrêt sur l'élément choisi).

  - un codage direct: morse à deux codes (point, barre).

- un multi-interrupteur quatre touches permet la commande des mouvements de deux curseurs (un curseur ligne et un curseur colonne) par la mise en oeuvre de deux touches, une troisième touche provoque l'arrêt du balayage sur l'élément choisi, une quatrième touche ramène les curseurs à leur position initiale.

- un multi-interrupteur cinq touches permet la commande du déplacement curseur suivant quatre directions(-->,<--,↑,↓), la cinquième touche autorise le choix de l'élément.

- un multi-interrupteur à huit touches permet le codage direct en utilisant, par exemple, le code ETRAN.

La représentation vidéo de ce code est présentée sur la figure I.3.

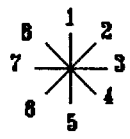
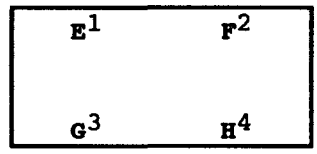
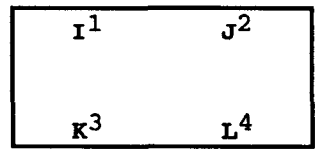
.1 ?2 ,3 :4	A1 B2 C3 D4	E1 F2 G3 H4
I1 J2 K3 L4		M1 N2 O3 P4
Q1 R2 S3 T4	U1 V2 W3 X4	Y1 Z2 (3)4

figure I.3

La case centrale représente le clavier à huit touches. Dans les autres cases sont reprises les lettres de l'alphabet quatre par quatre, ainsi que les signes de ponctuation. La première frappe permet le choix d'un nombre dans la case centrale. Par exemple, la frappe de l'interrupteur 2 oriente le choix vers la case supérieure droite.



Une deuxième frappe de 1 à 4 permet le choix d'une des lettres ci-dessus. De même une première frappe de l'interrupteur 7 oriente le choix vers la case:



I.2.2.1.3. Sélection par balayage

La méthode de sélection par balayage est utilisée par les handicapés ayant une capacité motrice très réduite. L'ensemble des pavés est initialement divisé en groupes et sous-groupes. Le balayage se fait par groupe à l'aide d'un moyen visuel de désignation (inversion vidéo par exemple). Sa vitesse (la durée d'un pas) peut être ajustée par l'utilisateur au moyen d'un élément de commande. Un appui sur l'organe d'accès (généralement un simple ou double interrupteur adapté) conduit à la sélection du groupe désigné. Si le groupe contient des sous-groupes, on reprend le balayage à l'intérieur d'un sous-groupe. Et ainsi de suite, jusqu'à la sélection d'un pavé.

On distingue, en fonction du groupement des pavés à sélectionner:

- le balayage série

C'est un balayage pavé par pavé de façon séquentielle, et suivant l'ordre naturel des pavés, par exemple ligne par ligne (ou colonne par colonne).

Une seule frappe de la touche de sélection suffit pour la sélection d'un pavé. Le temps de sélection dépend de la position du pavé dans l'ensemble. La sélection du $n^{ième}$ pavé dans l'ensemble demande au moins (n-1) pas de balayage.

Ainsi, par exemple, un curseur se déplace sur un écran, l'action sur un mono-interrupteur arrête son déplacement sur le pavé sélectionné.

- le balayage ligne-colonne (ou colonne-ligne) [TIC.85].

C'est un groupement à deux niveaux. Les pavés à balayer sont groupés en lignes (ou colonnes).

Dans un premier temps, le menu est balayé de ligne en ligne (ou de colonne en colonne). Suite à une action sur l'organe d'accès, une des lignes (ou colonne) est sélectionnée. Il se poursuit alors un balayage pavé par pavé à l'intérieur de cette ligne (ou colonne). Ce balayage aboutit à la sélection d'un pavé.

La sélection d'un pavé nécessite donc deux actions sur l'organe d'accès. L'effort physique est donc plus important que dans le balayage série. Le temps nécessaire à la sélection est considérablement réduit. Pour qu'il soit minimum, il y a intérêt à:

- ranger les pavés dans une matrice la plus carrée possible.

- disposer les pavés dans une matrice de manière telle que les pavés les plus utilisés soient proches du coin supérieur gauche.

- le balayage dichotomique

Il consiste à grouper les 2^{2M} pavés à sélectionner en M niveaux. A chaque niveau, le groupe de pavés est décomposé en quatre sous-groupes. On sélectionne, à chaque niveau, l'un des quatre sous-groupes jusqu'à sélection du pavé désiré. Le choix d'un pavé parmi n, ou $n = 2^{2M}$, nécessite donc M niveaux de balayage. Le balayage dichotomique ne peut se faire que sur un nombre de 2^{2M} pavés.

La sélection par balayage dichotomique exige de la part de l'utilisateur un effort physique plus important que dans les méthodes précédentes. Le nombre d'actions sur l'organe d'accès est plus important. Par contre, le gain en temps d'accès est considérable.

Considérons, à titre d'exemple, un menu présenté sous la forme d'une matrice 8X8. Elle est donc constituée de 64 pavés, le choix d'un pavé nécessitera donc 3 niveaux de balayage ($M = 1/2 \log_2 n$).

Au premier niveau, l'opérateur choisit un des quatre groupes de pavés présentés, par exemple le premier.

a	b	c	d	e	f	g	h
i	j	k	l	m	n	o	p
q	r	s	t	u	v	w	x
y	z	A	B	C	D	E	F
G	H	I	J	K	L	M	N
O	P	Q	R	S	T	U	V
W	X	Y	Z	1	2	3	4
5	6	7	8	9	0	é	è

Au deuxième niveau, le système présente à l'opérateur le groupe de pavés présélectionnés. Il choisit un des quatre groupes. Par exemple le premier.

a	b	c	d
i	j	k	l
q	r	s	t
y	z	A	B

Au troisième niveau, l'opérateur choisit un des quatre derniers pavés sélectionnables. Le premier, par exemple.

a	b
i	j

- le balayage dichotomique mixte

Il est utilisé dans le cas où le nombre de pavés sélectionnables est une puissance de deux: $n=2^M$ avec n égal au nombre de pavés et M un entier positif. L'ensemble des n pavés est regroupé en groupes et sous-groupes sur M niveaux. La sélection est faite de manière à sélectionner, à chaque fois, la moitié des pavés du groupe. Chaque étape du balayage consiste à balayer les deux sous-groupes du groupe en cours.

I.2.2.1.4. Méthodes de désignation

Le ou les pavés sélectionnables peuvent être désignés par:

- inversion vidéo,
- clignotement,
- soulignage,
- déplacement d'un curseur,
- encadrement,
- changement de couleur.

I.2.2.1.5. Les éléments informatifs de base [HE.87]

Au cours de la communication, les éléments informatifs de base sont présentés à l'utilisateur sur un dispositif d'affichage pour être sélectionnés à l'aide d'un organe d'accès et une méthode de sélection.

Ces éléments sont utilisés soit pour:

- composer des messages,
- commander une action.

On distingue donc deux catégories d'éléments: les éléments informatifs (appelés aussi pavés affichables) et les éléments de commande (appelés aussi pavés exécutables).

Ces éléments sont logiquement classés en groupes (menus) et sous-groupes suivant différents critères.

Les éléments informatifs sont appelés pavés affichables du fait que tout élément informatif sélectionné correspond à un affichage dans un message.

Dans les communicateurs, on rencontre des éléments informatifs très divers. On peut citer, de façon non exhaustive, les catégories suivantes:

- les lettres d'une langue naturelle,
- les chiffres,
- des caractères spéciaux (signes de ponctuation, opérateurs mathématiques, et divers symboles),
- des mots,
- des phrases,
- des phonèmes,
- les mots réservés à un langage informatique de programmation
- des primitives pictographiques (point, carré, cercle, couleur, etc),
- les composants des symboles BLISS,
- les éléments du PAR-LE-SI-LA-B,
- les symboles de notation musicale, ...

Un élément informatif peut avoir éventuellement un comportement particulier. On distingue:

- le pavé générateur

C'est un pavé affichable dont la sélection entraîne celle d'autres pavés affichables, il "génère" ces nouveaux pavés.
Ainsi, le pavé "boire" engendre le message "je voudrais boire quelque chose".

- le pavé modulateur

C'est un pavé affichable dont l'existence dans un message détermine celle de certains pavés qui le suivent. La sélection de ce pavé entraîne, dans le menu, la suppression de ces derniers.

Par exemple, si le début d'un message est constitué des deux pavés "vouloir" et "boire", il est logique que les pavés "tartine", "viande"... soient supprimés.

- le pavé modulateur-générateur

C' est un pavé affichable qui peut générer des pavés et en supprimer d'autres.

- le pavé affichable-exécutable

C' est un pavé affichable dont la sélection déclenche une commande exécutable. Ainsi la sélection du pavé "boire" par l'utilisateur peut entraîner, afin de prévenir son entourage, l'émission immédiate d'un message vers le synthétiseur vocal ("je voudrais boire quelque chose).

- le pavé vide

Il représente un espace blanc de dimension fixe ou initialement non déterminée. Il sert de séparateur dans la représentation des messages primaires.

Les éléments de commande se distinguent des éléments informatifs par le fait qu'ils n'ont pas pour mission d'être des composants de messages. La sélection d'un tel élément déclenche immédiatement l'exécution de la commande correspondante.

On peut classer les éléments de commande d'un communicateur en deux groupes:

- les éléments de contrôle d'édition des messages,
- les éléments de contrôle de l'environnement.

a) Les éléments de contrôle d'édition des messages:

Ce sont les commandes utilisées pour faciliter le processus d'édition des messages primaires ou bien pour manipuler le message en élaboration. Elles sont analogues aux commandes d'un éditeur de texte. Les commandes peuvent exister pour:

- le déplacement du curseur dans le tampon de message en cours,
- l'insertion, la suppression et l'interversion de pavés sélectionnés,
- le changement de tampon de message,
- le changement de fenêtre de visualisation de message,
- le changement de menu,
- l'ajustement de la vitesse de balayage,
- le changement de mode de travail.

Ces commandes n'ont pas d'interaction avec l'environnement de la communication.

b) Les commandes de contrôle de l'environnement

Il s'agit ici des commandes, qui contrôlent les dispositifs de l'environnement par l'intermédiaire des interfaces de sortie. Les commandes sont très diverses.

On trouve par exemple: l'allumage d'un téléviseur,la sortie du message d'un tampon vers un synthétiseur vocal ou une imprimante, ...

I.2.2.2. Réalisation de l'objet abstrait primaire.

Le deuxième niveau du communicateur est celui de la réalisation de l'objet abstrait primaire ou en d'autres termes du message abstrait primaire.

A partir des composantes primaires et en s'appuyant éventuellement sur un dictionnaire et des méthodes d'accélération, l'handicapé élabore l'objet abstrait primaire. Abstrait car, à ce stade, il peut se présenter sous une forme incompréhensible pour l'environnement humain ou matériel.

La classification des composantes primaires en menus et sous-menus se fait suivant divers critères tels que:

- les mots usuels rangés par ordre alphabétique,
- la classe grammaticale,
- le nombre de caractères,
- la classe sémantique
- les bigrammes, trigrammes...
- les bigrammes débuts de mot etc ...

I.2.2.2.1. Visualisation des menus.

Les systèmes d'aide à la communication pour handicapés voyants utilisent, dans la plupart des cas, un écran à tube cathodique pour visualiser les informations mentionnées ci-dessus.

L'écran est partagé en fenêtres: une ou plusieurs fenêtres pour les menus, une ou plusieurs fenêtres pour les messages et éventuellement une fenêtre de renseignement.

La figure I.4. présente quelques configurations possibles de l'écran.

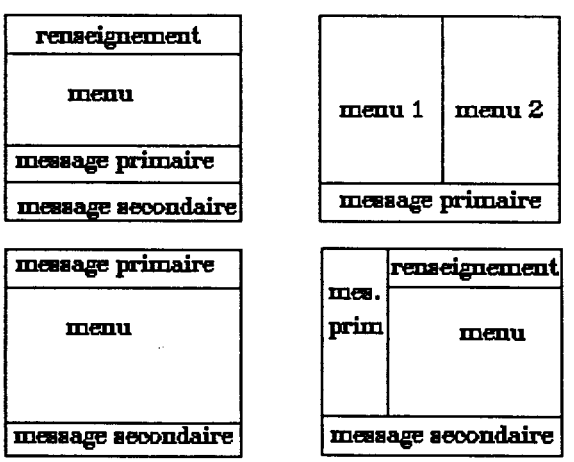


figure I.4

Le nombre, la dimension et la disposition de chacune des fenêtres dépendent de la taille des menus, ainsi que de la nature des messages à construire.

Le nombre et la dimension des pavés d'un menu dépendent principalement des aptitudes physiques et mentales de l'utilisateur. Il doit être facilement possible de modifier ces paramètres en fonction des progrès effectués par l'utilisateur dans le maniement du système d'aide à la communication.

Les pavés peuvent être disposés de différentes manières dans un menu. On utilise, soit des menus à disposition rectangulaire, soit des menus à disposition triangulaire (figure I.5).

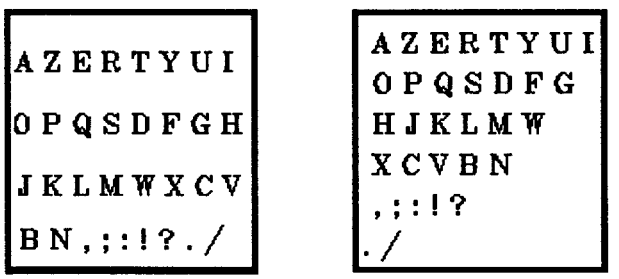


figure I.5

Dans les menus triangulaires, le nombre de pas de déplacement maximum du curseur est constant et égal au nombre de colonnes plus un. Dans le cas des menus rectangulaires, le nombre maximum de pas de déplacement est égal à la somme du nombre de lignes et de colonnes du menu.

I.2.2.2.2. Les méthodes d'accélération

[TIC.85][OUA.86][HEC.83].

Les méthodes d'accélération sont destinées à améliorer le rapport vitesse de production effort nécessaire à la production. Elles peuvent être classées en cinq méthodes:

- accélération fréquentielle statique,

- accélération fréquentielle dynamique,

- accélération par expansion,

- accélération syntaxique,

- accélération sémantique.

a) Accélération fréquentielle statique.

Les éléments informatifs de base (pavés) sont classés et présentés à l'handicapé en fonction de leur fréquence d'utilisation (fréquentielle statique).

Une analyse des mots utilisés par l'handicapé permet d'établir une table donnant la probabilité d'utilisation des lettres de l'alphabet. On les présente dans l'ordre décroissant de leur fréquence d'utilisation:

E A S I T N R U L O D C M P V Q G F B H J X Y Z K W

Un système utilisant ce type d'accélération est généralement utilisé par des handicapés physiques profonds. Le système est, par exemple, constitué d'un mono-interrupteur et d'un écran vidéo. Les caractères sont présentés sur l'écran dans un menu triangulaire en fonction de la table de probabilité. Ceux qui présentent la plus grande probabilité d'utilisation sont disposés le plus près possible du coin supérieur gauche de l'écran. Un caractère repère se déplace verticalement. Une action sur le mono-interrupteur permet le choix de la ligne qui correspond à la position du repère. Le choix de la colonne se fait par la même technique.

On peut faire l'analyse de la fréquence d'utilisation des éléments en cours d'utilisation. Le communicateur enregistre sur une certaine période les fréquences réelles d'utilisation des éléments. Une mise à jour de la table des éléments informatifs permet une adaptivité du système.

b) Accélération fréquentielle dynamique [WAL90].

Les éléments informatifs de base sont présentés à l'handicapé en fonction de leur probabilité d'apparition (fréquentielle dynamique). Le système tient compte des éléments déjà choisis. Après chaque choix, il présente à l'utilisateur des nouveaux menus dont les éléments sont disposés en fonction de leur nouvelle probabilité d'utilisation. Certains systèmes ne tiennent compte que de la probabilité d'utilisation sur deux (bigrammes) ou trois caractères (trigrammes).

A titre d'exemple, la liste des bigrammes de la langue française, établie à partir d'un texte choisi par nous, pour les lettres A et B sont les suivantes:

A---I N T R U L S C V M B G P Y D Q F X

B---L O I R A J E S U T B

La lettre A est suivie le plus souvent de la lettre I et au deuxième rang par la lettre N et ainsi de suite. La lettre B est suivie le plus souvent de la lettre L et au deuxième rang par la lettre O et ainsi de suite.

c) Accélération par expansion

Cette méthode suppose l'existence d'un dictionnaire. Il s'agit, à partir d'un nombre minimum d'informations d'accéder à l'élément recherché du dictionnaire. Elle permet, par exemple, un accès:

- lettre-mot,
- lettre-phrase,
- mot-phrase,
- mnémonique-phrase,
- symbole d'un élément-élément,
- partie d'élément-élément,
- bigramme ou trigramme début de mot.

Dans le cas, par exemple, du lettre-mot, l'handicapé choisit une lettre et il accède à tous les mots du dictionnaire commençant par cette lettre, lesquels sont visualisés.

Une analyse du contexte de la communication est souvent utile pour limiter le nombre d'éléments présentés à la sélection. Ainsi, la sélection de la lettre "G" conduit à proposer tous les mots possibles commençant par "G". Mais, si l'on se limite au contexte de la programmation en BASIC, le nombre de mots possibles est réduit à 2 (GOTO et GOSUB).

Une analyse de ce type d'accélération nous a permis de montrer que l'accélération de la communication peut être très importante [TIC.85] . Dans cette étude, les éléments informatifs sont rassemblés par catégorie grammaticale (Noms, Verbes, Adjectifs et Divers) et par bigrammes début de mot. L'utilisateur désirant composer un message choisit successivement:

- la catégorie grammaticale du mot à écrire,
- la première lettre du mot,
- la seconde lettre du mot.

Le système de communication recherche et visualise tous les mots du dictionnaire qui répondent à ces trois éléments informatifs de base.

d) Accélération syntaxique.

Cette méthode consiste à présenter à l'utilisateur uniquement les éléments corrects au point de vue grammatical en tenant compte des éléments déjà sélectionnés.

Par exemple, la sélection de l'article déterminant "le" ne donne accès qu'aux mots masculins singuliers du dictionnaire.

On simule l'ensemble des messages primaires que l'utilisateur souhaite construire. On regroupe les éléments primaires et les mots du dictionnaire en classes ou catégories syntaxiques, puis on établit un ensemble de règles, qui assurent la liaison entre ces classes.

Comme il est impossible de prévoir toutes les structures possibles, on limite généralement l'analyse syntaxique aux "phrases" couramment utilisées par l'utilisateur.

Il a été développé dans notre centre d'automatique un éditeur de Dessins à utiliser par des handicapés qui utilise cette méthode d'accélération. Le langage logiciel utilisé est le Logo. Le choix d'une instruction implique le respect de sa syntaxe. Le logiciel propose à l'utilisateur le choix des seuls éléments qui sont conformes à cette syntaxe. Ainsi, si l'utilisateur choisit l'instruction "avance", le logiciel propose "nombre de pas".

e) Accélération sémantique.

L'analyse syntaxique est complétée par une analyse sémantique. Le système présente à l'utilisateur uniquement les éléments utiles du point de vue sémantique.

Ainsi, le verbe manger entraîne la visualisation des éléments représentant le nom des aliments.

Cette méthode n'est en pratique réalisable que pour un vocabulaire restreint.

I.2.2.2.3. Visualisation des messages primaires.

Durant leur élaboration, les messages primaires sont mémorisés dans une mémoire tampon ("buffer") et visualisés sur l'écran vidéo dans une fenêtre de message.

Le nombre et la dimension des fenêtres de message peuvent varier. Une fenêtre peut être liée à un ou plusieurs menus.

Le contenu du tampon de messages doit pouvoir être modifié, effacé, copié, conservé. A l'aide des éléments de commande contenus dans les menus, l'utilisateur contrôle constamment l'élaboration de ses messages et peut à tout moment:

- changer de tampon en cours,
- initialiser un tampon,
- sauvegarder un tampon,
- faire appel à un tampon sauvegardé,
- chaîner des tampons,...

I.2.2.2.4. Visualisation de renseignements divers.

On peut, en dehors des menus et fenêtres d'affichage des messages primaires, visualiser des informations utiles telle que:

- heure et date,
- température intérieure et extérieure,
- programme de radio et télévision,
- mémento de l'utilisateur,
- liste et heure des soins,
- numéro du tampon de message affiché, ...

I.2.2.3. Réalisation de l'objet abstrait secondaire.

Venons-en maintenant au troisième niveau de notre système de communication: la réalisation de l'objet abstrait secondaire. Jusqu'ici, l'objet abstrait primaire était retenu dans une mémoire tampon et réalisé dans un langage compris par l'utilisateur du communicateur. Il peut ne pas être compréhensible par son destinataire (humain ou machine). Ainsi, le langage idéographique Bliss utilisé par de nombreux handicapés n'est pas compréhensible par un non initié. Il faut lui faire subir des transformations d'ordre syntaxique et sémantique. Les principales transformations utilisées sont des traductions:

- Français - phonèmes,
- Bliss - Français,
- Français simplifié - Français,
- Par-Le-Si-Lab - Français, ...

I.2.2.4. Organe de sortie.

Le message parvient alors à l'organe de sortie, qui est le quatrième niveau de notre système. Le message peut se présenter sous les formes suivantes:

- langue usuelle:

  écran de visualisation, imprimante, voix synthétique, afficheur braille...

- commande de l'environnement immédiat:

  éclairage, radio et télévision, position du lit, conditionnement d'air, fauteuil roulant, bras-robot

On prévoit habituellement de permettre à l'utilisateur de pouvoir, en cas de nécessité, répéter l'émission du message par une simple commande.

I.3. CRITERES DE QUALITE.

Les critères de qualité d'un système d'aide à la communication pour handicapés doivent, à notre avis, être les suivants:

- adaptabilité aux handicapés,
- optimalité dans le rapport effort / vitesse,
- non spécificité, mais possibilité de relation à la fois avec des systèmes spécifiques et universels,
- multi-contexte,
- de faible coût,
- portable et transportable,
- autonome mais facilement connectable,
- fiable et maintenable.

La qualité d'adaptivité est primordiale, puisque chaque utilisateur peut être considéré comme un cas particulier, dont il importe d'évaluer précisément le nombre de degrés de liberté lui permettant d'exprimer une certaine quantité d'informations via des éléments d'entrées physiquement adaptés.

L'adaptivité doit aussi aller dans le sens de l'optimisation du rapport effort/vitesse. Pour chaque individu existe une quantité d'informations initiale exprimable. Celle-ci rend possible un codage d'accès d'un ensemble plus ou moins grand de constituants élémentaires dits composantes primaires du message élaboré. En tenant compte du niveau informatif du message, un compromis doit être trouvé entre le nombre de ces constituants et le nombre d'interventions de l'utilisateur pour élaborer un message.

Un système de communication doit être non spécifique dans le sens où il doit, à la fois, permettre l'accès à des dispositifs spécifiques tels que des contrôleurs d'environnement, pilote de fauteuil roulant ou de micromanipulateurs et à des systèmes universels (vidéotexte, micro-ordinateurs avec tous les logiciels et didacticiels).

Une procédure multi-contexte doit être établie afin de permettre l'interruption d'un travail en cours, pour effectuer une autre activité plus urgente, puis revenir au travail antérieur sans perte d'informations.

Le système ne doit pas être un monstre, uniquement utilisable dans un centre spécialisé, car l'utilisateur, hors de ces sites privilégiés, serait alors privé de communication. Il faut donc un système facilement transportable et autonome, mais connectable à un grand nombre de dispositifs soit eux-mêmes portatifs, soit assurant des conditions de vie meilleure dans un cadre spécifique.

Enfin la mise en oeuvre du système doit être portable, c'est-à-dire qu'elle doit pouvoir suivre, sans remise en cause fondamentale, l'évolution de la technologie. Elle doit aussi être fiable et réalisée à partir d'éléments standards permettant un coût faible et une maintenabilité accrue.

I.4. CONCLUSION.

Un communicateur doit être parfaitement adapté aux possibilités de l'utilisateur et il doit pouvoir évoluer en même temps que croissent ses possibilités.

Au départ, on utilisera des communicateurs très simples, qui seront complétés au fur et à mesure des progrès de l'utilisateur. Pour ce faire, les handicapés devront être suivis par des spécialistes qui se chargeront de l'apprentissage et qui adapteront cet apprentissage à l'évolution des performances de l'utilisateur.

Les spécialistes en informatique devront donc fournir aux non spécialistes en informatique la possibilité **de générer** un communicateur de base parfaitement adapté aux besoins du moment mais susceptible d'évoluer ou d'être modifié très rapidement.

C'est de ce générateur, dont il sera traité dans le chapitre suivant.

CHAPITRE II

CONCEPTION ASSISTEE D'UN COMMUNICATEUR POUR HANDICAPES A L'AIDE D'UN LANGAGE OBJET.

II.1. INTRODUCTION

Nous nous proposons, dans ce chapitre, de présenter la conception générale d'un système de génération automatique de logiciels de communication pour handicapés.

Le système génère des communicateurs basés sur les concepts, qui ont été décrits dans le chapitre précédent.

Le générateur est conçu suivant le concept de la programmation orientée objet. Générer, dans ce cas, consiste à créer des objets spécifiques au communicateur et à les placer dans une base de données.

Le communicateur est un logiciel, qui gère la communication entre objets. Ce logiciel est tout à fait général. Seule la base de données est spécifique, elle contient l'ensemble des objets qui caractérisent un communicateur.

II.2. GÉNÉRATION AUTOMATIQUE DE COMMUNICATEURS SPÉCIFIQUES.

Les logiciels mis en oeuvre traditionnellement sont souvent onéreux, mal documentés, peu modifiables et peu réutilisables. Un effort important a donc été mené ces dernières années, pour produire automatiquement des logiciels moins coûteux et parfaitement adaptés aux besoins spécifiques de l'utilisateur.

On distingue les systèmes suivants:

- progiciels individualisables,
- programmation évolutive,
- programmation par l'exemple,
- programmation en langage orienté objet,
- systèmes experts et langages orientés objets.

L'utilisation simultanée de plusieurs systèmes est possible.

Le processus de génération d'un produit logiciel comprend une phase de spécification des besoins, une phase de production du programme et une phase éventuelle de production de la documentation du produit.

II.2.1. Progiciels individualisables [VAN.82].

L'idée de base, dans le progiciel individualisé, repose sur un ensemble de modules de programme fortement paramètrés et des valeurs des paramètres correspondantes. Au cours de la phase spécification des besoins, un programme est élaboré grâce à un questionnaire préétabli.

L'opérateur (1) reçoit le questionnaire et répond en fonction de l'application qui lui est propre. Ceci se passe le plus souvent par un système de questions et réponses.

---

(1) Pour rappel, l'opérateur est la personne, non spécialisée en informatique, chargée de générer le communicateur pour l'handicapé.

Au fur et à mesure de l'introduction de la spécification, le système construit, à partir des réponses, le modèle particulier du logiciel à produire. Ce modèle détermine d'une part le sous-ensemble de modules à sélectionner parmi l'ensemble des modules préprogrammés et les valeurs des paramètres, d'autre part l'ensemble des données propres à l'application. Il est utilisé pour la production du programme et la production de la documentation.

Pendant la phase de production du programme, suivant le modèle particulier, le système sélectionne le sous-ensemble de modules, substitue les valeurs des paramètres de ces modules, génère les structures de données de l'application, assemble ces modules et ces données pour obtenir un programme dont les comportements peuvent satisfaire les besoins spécifiques de l'handicapé.

Les opérations dans cette phase sont automatisées, l'opérateur n'a pas à intervenir.

La phase de documentation permet, à la demande de l'opérateur, de produire la documentation du produit généré, grâce au modèle particulier du logiciel. La documentation peut se présenter sous différentes formes : mode d'emploi du logiciel pour l'opérateur, rapport informatique pour l'opérateur du système de génération.

II.2.2. Programmation évolutive [VAN.82].

La programmation évolutive utilise un ensemble de programmes de base (ou "plans"), qui réalisent les fonctions standards (ou "types") du domaine d'application et un ensemble de règles de transformation de programmes.

Les programmes générés sont obtenus par transformation des programmes de base. Ces programmes sont obtenus par essai, modification et généralisation.

Les programmes sont rédigés dans un langage de très haut niveau. L'opérateur et le système coopèrent à la mise au point dans un environnement interactif.

II.2.3. Programmation par l'exemple [BIE.76].

La programmation "par l'exemple" revient à fournir des programmes à partir de spécifications, qui décrivent quelques exemples typiques des résultats que l'opérateur désire obtenir.

Le système, dans le domaine d'application, analyse les relations d'entrée-sortie, les généralise, en dégage une loi générale de correspondance, et fournit un logiciel adéquat à partir de cette loi. La généralisation de la loi de correspondance est parfois rendue possible et facilitée grâce à des interactions supplémentaires avec l'opérateur.

La façon de spécifier un programme paraît naturelle et aisée à l'utilisateur.

II.2.4. Systèmes à base de connaissances [HEW.75][LAU.82].

Cette approche utilise des techniques relevant du domaine de l'Intelligence Artificielle.

Un système expert de production de logiciels fait usage d'une base de connaissances dans laquelle on distingue:

- des connaissances de la programmation,
- des connaissances du domaine d'application,
- des connaissances sur la mise au point,
- des connaissances du comportement et de l'interface opérateur-système.

Le moteur d'inférence est le programme, qui se charge du raisonnement. Il utilise des connaissances pour résoudre le problème particulier de l'utilisateur. Les spécifications de l'application se trouvent dans la base de faits.

La production d'un programme se fait en plusieurs phases:

- spécification de l'application,
- transformation en procédures,
- vérification du produit (modèle),
- programmation automatique.

L'opérateur spécifie l'application à l'aide d'une interface interactive, en langage naturel par exemple. On définit ainsi un modèle particulier, qui est ensuite traité à l'aide des connaissances propres au domaine d'application. Le modèle traité est vérifié par l'opérateur, afin d'assurer la conformité du modèle à ses besoins et de permettre d'éventuelles modifications du modèle. L'application de règles de production et d'optimisation au modèle permet de produire un programme parfaitement adapté.

II.2.5. Programmation en langage orienté objet [GOL.83][MEV.87].

La programmation objet est fondamentalement différente de la programmation des langages dits classiques.

La caractéristique essentielle de ce type de langage est l'économie de concept. L'ensemble repose, en effet, sur quatre notions:

- l'objet,
- la classe,
- le message,
- l'héritage.

Un objet est une entité informatique regroupant un certain nombre d'informations. Il comprend une collection de variables locales, qui sont ses propriétés ou attributs, ainsi qu'un ensemble de procédures (ou méthodes) pouvant traiter ces variables ou faire des actions . Ainsi, dans un éditeur de dessins, la plume peut être considérée comme un objet formé d'une collection de variables qui définissent son épaisseur, sa couleur,... capable de dessiner à main levée, des cercles, des droites ...

Les étapes, qui permettent de définir un objet sont:

- l'identification des données statiques,
- l'identification des données dynamiques,
- l'établissement des liens entre objets.

L'identification des données statiques établit la nature, l'identité et la composition de l'objet.

L'identification des données dynamiques a pour but de préciser les opérations, qui peuvent être exécutées par l'objet sur d'autres objets ou qui peuvent être subies par l'objet de la part d'autres objets.

L'établissement des liens entre objets se fait de la manière suivante: l'environnement manipule un objet en le sélectionnant et en lui disant ce qu'il doit faire. L'objet décide comment exécuter l'injonction en identifiant le message à mettre en oeuvre dans la table de sélecteurs que sa classe comporte.

Ces objets peuvent être regroupés sous forme d'ensembles appelés "classes". Une classe donne tous les attributs communs à un ensemble d'objets particuliers. C'est, en fait, un modèle d'objet qui servira à en construire d'autres. Ainsi la classe des "Menus" sera un objet décrivant les propriétés d'un objet "menu". La notion de classe permet donc de créer un grand nombre d'objets similaires. Ceux-ci disposent tous des mêmes attributs, mais avec des valeurs différentes. Un objet appartenant à une classe est dit "instance" de celle-ci.

Dans la plupart des cas, il n'est pas nécessaire de dire au programme quels paramètres doivent être pris par défaut, car il recherchera de lui-même les bons paramètres lors de l'exécution d'une méthode.

Quant au message, il est constitué d'un ensemble d'expressions: le receveur, le sélecteur et les arguments. Le receveur est l'objet auquel est destiné le message, tandis que le sélecteur dirige les instructions destinées au receveur. Quant aux arguments, il s'agit simplement des paramètres utilisés par le sélecteur. Trois types de messages peuvent être envoyés:

- des messages unaires privés d'arguments,
- des messages binaires ayant un argument unique et un receveur,
- des messages-clés composés d'un sélecteur et d'un ou plusieurs arguments. C'est à l'aide de ces divers messages que sont composés les programmes Smalltalk.

En Smalltalk, la structure de base admet qu'une classe soit décrite indépendamment des autres. Il faut pourtant tenir compte de la possibilité d'héritage. Avec elle, une classe peut être décrite comme une modification d'une autre classe, qui devient alors sa super-classe. Toute classe, qui modifie une classe particulière est appelée une sous-classe. Une sous-classe hérite des variables d'instance et des méthodes de sa super-classe. Pour permettre une description d'une classe, il est donc nécessaire de regarder ses super-classes et cela jusque la classe "Object". La classe "Object" est la seule, qui ne comporte pas de super-classe. Toutes les classes héritent donc des méthodes de la classe "Object".

Ce type de langage est basé sur le principe de la modularité forte, qui veut qu'aucune partie du système ne dépende des détails de fonctionnement des autres parties. C'est ce qui le rend particulièrement intéressant au point de vue évolution du logiciel. Le programmeur peut le faire évoluer sans avoir à étudier la structure des objets existants. Seule la connaissance du comportement externe des objets est nécessaire.

Les avantages des langages objet sont multiples:

- toutes les communications ont la même forme et il suffit de connaître la forme des messages pour pouvoir programmer. Le travail du programmeur est ainsi réduit dans une large mesure et sa productivité est augmentée. Il en résulte donc une diminution des coûts.

- grâce à la modularisation et au fait que les données ne sont visibles qu'à partir de l'intérieur du module, la mise en oeuvre du logiciel résiste à de mauvaises manipulations des données.

- la modularité du système permet des modifications et des extensions bien plus aisées à réaliser que dans les systèmes de programmation classiques.

- l'héritage permet la création de nouvelles classes à partir d'autres moins spécialisées. Ces classes sont créées en héritant de toutes les variables et procédures définies dans une classe primitive, en lui adjoignant des variables et procédures spécifiques.

II.2.6. Systèmes experts et langages orientés objets.

Le rôle des systèmes experts est de reproduire le plus fidèlement possible des raisonnements jusqu'alors dévolus à des spécialistes. Les premiers systèmes fonctionnaient à l'aide de règles du type

"SI" PRÉMICES "ALORS" CONCLUSION "ET" ACTION.

Le pari des systèmes experts consistait à penser que ces morceaux de connaissances (shunk of knowledge) puissent, en s'agençant, constituer un raisonnement complet. Ces systèmes sont surtout cantonnés dans des domaines où l'heuristique et l'expérience acquise prédominent. Plutôt que de décomposer un problème informatique en modules et procédures, on peut construire des programmes bâtis directement autour d'objets existant réellement.

Le système se compose d'objets et de règles.

Dans un système expert, il faut pouvoir accéder aux règles de manière associative, c'est-à-dire par leur contenu ou par leurs caractéristiques. Ainsi, par exemple, appliquer toutes les règles relatives à l'accord des verbes en "er".

II.2.7. Conclusions

De toutes ces approches seule l'approche "progiciels individualisables" est vraiment opérationnelle. Elle permet de produire automatiquement des logiciels d'application "taillés sur mesure" dans un domaine d'application bien circonscrit [HE.87b]. Le défaut majeur d'un progiciel individualisable est le manque de flexibilité vis à vis d'un utilisateur particulier. Ce défaut, inhérent à l'approche, est dû au fait que les différents types d'informations qui concernent l'application et la programmation sont "figés" en une seule représentation: les modules paramétrés.

Nous avons voulu concevoir un système qui évite ces problèmes. La programmation orientée objet présente l'avantage de permettre l'écriture d'un logiciel sans remettre en cause la structure du logiciel existant. L'organisation du système en objets permet une évolution aisée du communicateur. Pour faire évoluer le logiciel de génération afin qu'il réponde à de nouveaux besoins, il suffit d'ajouter de nouvelles classes et méthodes à celles qui existent déjà. Il n'est pas nécessaire pour le programmeur d'étudier la structure des objets existants. Seule la connaissance du comportement externe des objets est nécessaire.

II.3. CONCEPTION ASSISTÉE DE COMMUNICATEURS A PARTIR DE LANGAGES OBJETS (PROJET CACHALOT).

II.3.1 Projet CACHALOT [TIC.87]

Notre projet consiste en la conception et la réalisation d'un système qui permet la **C**onception **A**ssistée de **C**ommunicateurs pour **H**andicapés à l'**A**ide d'un **L**angage Obje**T** (CACHALOT).

Ce système doit aider un opérateur, non spécialisé en électronique, de concevoir des prototypes de communicateurs adaptés à toute personne handicapée.

Il doit être d'une grande facilité d'utilisation et de présentation moderne (utilisation de: fenêtres, menus déroulants, utilisation de la souris ...).

Il doit pouvoir être implanté dans un micro-ordinateur.

Le logiciel doit être écrit suivant la méthodologie objet.

Nous avons choisi SMALLTALK/V comme langage de programmation. Il permet de réaliser un système qui répond à l'ensemble de nos critères.

L'utilisation du logiciel de génération du système CACHALOT est décrit en détail dans l'annexe 1.

II.3.2. Réalisation de prototypes

Le système de communication est une prothèse qui doit pouvoir être acceptée par l'handicapé. Pour que celui-ci ne le rejette pas, il est indispensable que le système réponde immédiatement à son attente.

Le communicateur doit pouvoir être généré "sur site". En collaboration avec l'handicapé. Le personnel chargé de générer le communicateur doit donc pouvoir disposer d'un outil qui lui permet la génération rapide d'un communicateur, son évaluation et des modifications aisées.

II.3.3. Conclusion

Le système construit en langage d'objet est très prometteur, car d'une grande souplesse de mise en oeuvre. La possibilité de modifier le logiciel est aisée et la réalisation rapide.

La structure des langages de type objet est telle que, implantée dans des micro-ordinateur de type PC-compatible, le dispositif fonctionne relativement lentement. En effet, au niveau du langage de programmation, le logiciel exécute des méthodes constituées d'un ensemble de méthodes. L'exécution d'une de celles-ci nécessite sa décomposition en primitives exécutables. Par ailleurs, dans un communicateur utilisant un langage pictographique, les éléments informatifs mémorisés en mémoire de masse sont codés. Il faut les décoder avant affichage, le temps nécessaire à la reconstitution d'un pavé codé est de l'ordre de 50 millisecondes. Toutes ces lenteurs ne constituent toutefois pas un problème, car l'handicapé ne fait en moyenne que quatre sélections par minute.

II.4. PRINCIPE DE LA GÉNÉRATION

L'opérateur dispose du logiciel de génération qui permet de créer, d'une manière interactive, des objets spécifiques à un communicateur.

Ce logiciel permet de générer des objets pour toutes les variantes prévisibles des fonctions de base du domaine d'application; il est donc censé couvrir la totalité du domaine.

Ce logiciel est rédigé dans un langage de haut niveau qui facilite la maintenance et la possibilité d'extension.
La présentation du générateur se fait sous la forme de fenêtres et de menus. L'opérateur définit étape par étape les différents constituants de son communicateur.

II.5. PHASES DE GÉNÉRATION D'UN COMMUNICATEUR

La génération d'un communicateur comprend une phase de spécification des besoins, une phase de production du communicateur et une éventuelle phase de documentation du produit, d'archivage et de réutilisation.

II.5.1. Spécification des besoins

La spécification des besoins du communicateur à générer se fait en définissant, dans un environnement interactif, chacun des constituants répondant au cahier des charges de l'application.

Le générateur analyse les réponses de l'opérateur, détecte les incohérences et les incompatibilités dans la constitution d'un objet. La hiérarchie et l'ordre de définition des objets sont prédéterminés, cependant il est possible de les modifier.

Le système génère ainsi un prototype qui peut être évalué à tout moment. L'opérateur peut ainsi contrôler, à chaque étape, le comportement du communicateur en élaboration et le modifier si nécessaire.

II.5.2. Production du communicateur

Après avoir testé le prototype de communicateur, on génère la base de données propre à ce communicateur.

La base de données est un ensemble de chaînes de caractères représentant les informations nécessaires pour reconstituer les objets du communicateur. Cette manière de sauvegarder les objets, quel que soit leur classe, permet de reconstituer n'importe quel objet à partir d'une seule méthode (à savoir: "evaluate:aStream"). Dans le cas du stockage de formes, la dimension des fichiers peut devenir très importante. Il est dès lors nécessaire de compresser ces fichiers. La récupération des fichiers, puis la réinstallation des instances qui y sont stockées demandent un temps qui peut sembler important. Toutefois cette installation se fait au lancement du communicateur et est sans influence pendant son fonctionnement. Nous avons créé des méthodes spécifiques à chaque fichier de façon à reconstituer les objets et à les réimplanter dans le communicateur.

Ainsi un communicateur se présente sous la forme d'une série de fichiers qui contiennent les caractéristiques (les instances) des:

- fenêtres: "instance.fen",
- mémoires de message ("buffers"): "instance.buf",
- menus: "instance.men",
- paves: "instance.pav".

Ces fichiers contiennent toutes les instances nécessaires à la reconstitution des objets, qui caractérisent le communicateur.

II.5.3. Phase de documentation, archivage et réutilisation.

En dehors du logiciel de communication, le système de génération peut produire automatiquement des documents sur le logiciel généré. L'opérateur, qui utilise le logiciel de communication dispose de renseignements tels que le mode d'emploi, la date, la liste des pavés, la liste des menus, le graphe de relation entre les menus, etc.... Le système de génération archive le communicateur généré de façon à ce que les informations contenues dans le communicateur puissent être utilisées ultérieurement non seulement pour la maintenance et l'extension du communicateur, mais aussi dans d'autres communicateurs.

II.6. GÉNÉRATION DE COMMUNICATEURS PAR PROGRAMMATION ORIENTÉE OBJET.

II.6.1. Introduction.

Un communicateur pour handicapés est composé d'une partie visuelle et d'une partie fonctionnelle.

La partie visuelle est composée d'objets présentés sur un écran vidéo: les pavés. Ces pavés font partie de menus, qui sont associés à des buffers.

On voit tout naturellement dans un communicateur une classe "Pavé", une classe "Menus" et une classe "Buffer".

La partie fonctionnelle est constituée de la classe "Communicateur", dont la mission essentielle est d'initialiser le communicateur et de la classe "Transformation" qui permet de transformer le message primaire en un message compréhensible par le récepteur.

Un communicateur est réalisé grâce à l'utilisation d'un générateur de communicateurs. Le générateur est un communicateur particulier qui possède toute une série d'"outils" (des utilitaires) qui permettent de créer ou de modifier l'ensemble des objets qui constituent un communicateur. La classe "Générateur" a donc été créée. Associée aux classes "InspectCommunicateur", "InspectPavés" et "InspectMots", elles permettent d'agir sur les variables, qui caractérisent un communicateur.

Afin d'augmenter le rapport vitesse de production d'un message / effort nécessaire à la production de ce message, nous avons intégré au système une méthode d'accélération à l'accès à un dictionnaire par recherche binaire.

II.6.2. Découpage d'un communicateur en objets.

II.6.2.1. Le communicateur.

Smalltalk n'est pas seulement un langage de programmation, c'est aussi une méthodologie de conception. Cela induit une façon de travailler différente de celle utilisée jusqu'à présent en informatique. Nous pouvons analyser notre système suivant:

- le concept de l'analyse descendante,

- la notion de spécialisation.

Dans le premier cas, on constitue l'arbre du communicateur. On regroupe ensuite les objets par famille en essayant de profiter de l'héritage.

Dans le second cas, on décrit un objet général que l'on spécialise en affinissant sa définition.

Dans notre système, nous avons décomposé un communicateur en entités (objets) qui appartiennent à des classes de base.

La figure II.1 présente la décomposition générale du communicateur en objets.

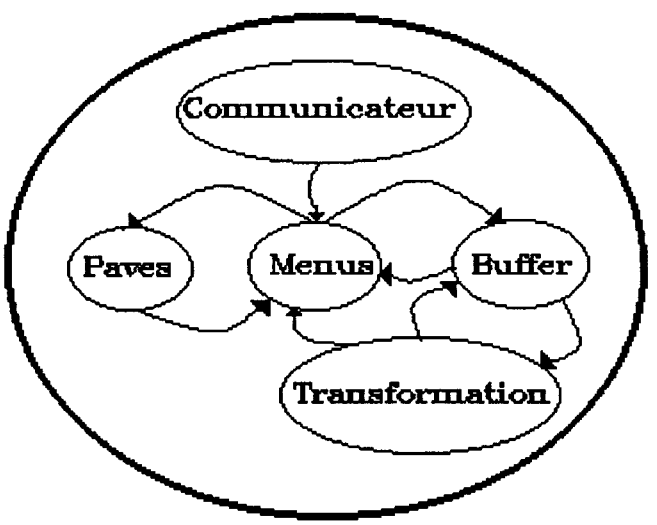


Figure II.1

Communiquer, c'est choisir dans un menu (classe Menus) des éléments informatifs de base (classe Pavé) afin de créer un message (classe Buffer) qui doit être compréhensible par le monde extérieur (classe Transformation). Il faut disposer de méthodes d'accélération qui donnent accès, d'une manière désordonnée, à des dictionnaires (classe BaseDeDonnées).

Pour générer et mettre au point un communicateur, il faut disposer d'un outil. Les classes InspectCommunicateur, InspectPavés et InspectMots nous fournissent cet outil.

II.6.2.1.1. La classe "Pavé".

La classe "Pavé" est une entité physique qui permet de visualiser les éléments informatifs de base sur un écran vidéo (figure II.2). Elle est la classe pivot du communicateur. C'est par l'intermédiaire des objets "pavés" que l'handicapé dialogue avec le monde extérieur et avec le logiciel.

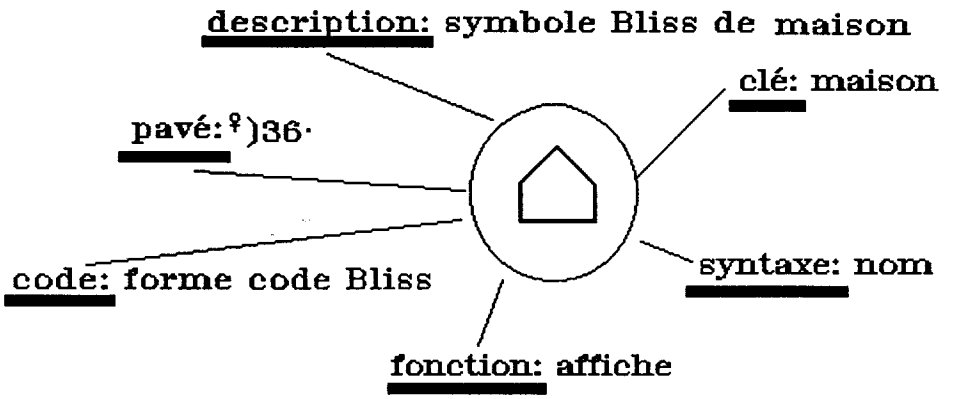


figure II.2

Un pavé contient tous les renseignements nécessaires à sa visualisation. Ce sont ses variables d'instance: code, pavé, clé, fonction, syntaxe et description.

La variable d'instance "code" indique la forme sous laquelle la représentation du pavé est codée. Le choix du codage influe sur la place mémoire occupée mais aussi sur le temps de reconstitution du pavé (décodage). Un compromis adéquat entre la vitesse de décodage et l'espace mémoire utilisé doit être trouvé. Ainsi un pavé peut être codé en tant que:

- forme (bitmap),

- chaîne de caractères (string: mots, bigrammes..),

- code de Freeman,

- code Bliss...

La variable d'instance "pavé" contient les informations qui concernent la représentation visuelle du pavé.

La variable d'instance "clé" contient le nom du pavé. Cette variable permet la recherche du pavé dans le dictionnaire.

La variable "fonction" décrit le comportement du pavé lors de sa sélection. Rappelons qu'un pavé sélectionné peut être soit affichable dans la mémoire de message (buffer) soit exécutable. Dans ce cas, sa sélection déclenche une action.

La variable "syntaxe" fournit la catégorie syntaxique du pavé. Cette information est utilisée pour transformer le message abstrait primaire en un message secondaire écrit en français correct lors de la phase de transformation.

La variable "description" est accessible à l'opérateur chargé de la génération d'un communicateur. Elle fournit les renseignements qui concernent le pavé.

Le fonctionnement dynamique du pavé est défini par ses méthodes. Elles sont principalement utilisées, au niveau du communicateur, pour fournir aux classes "Menus" et "Buffers" le pavé décodé sous sa forme affichable ainsi que sa fonction.

II.6.2.1.2. La classe "Menus".

Un menu visualise sur écran vidéo (figure II.3) un ensemble de pavés. Il contient un ensemble d'informations qui, sous la forme de ses variables d'instance, permettent de connaître:

- le type de représentation retenu du menu sur l'écran (menus triangulaires, rectangulaires, menus (menu dont on choisi le nombre de lignes et de colonnes)),

- le mode de balayage des pavés (ligne, ligne-colonne, dichotomique...),

- le mode de désignation (inversion vidéo, cadre, modification de la couleur du pavé

sélectionnable...),

- le délai de balayage, par défaut le délai est égal à la vitesse de balayage définie dans le communicateur,

- la fonction du pavé: passage au menu suivant, effacement d'un pavé dans le menu, changement de fenêtre (scrolling), initialisation...

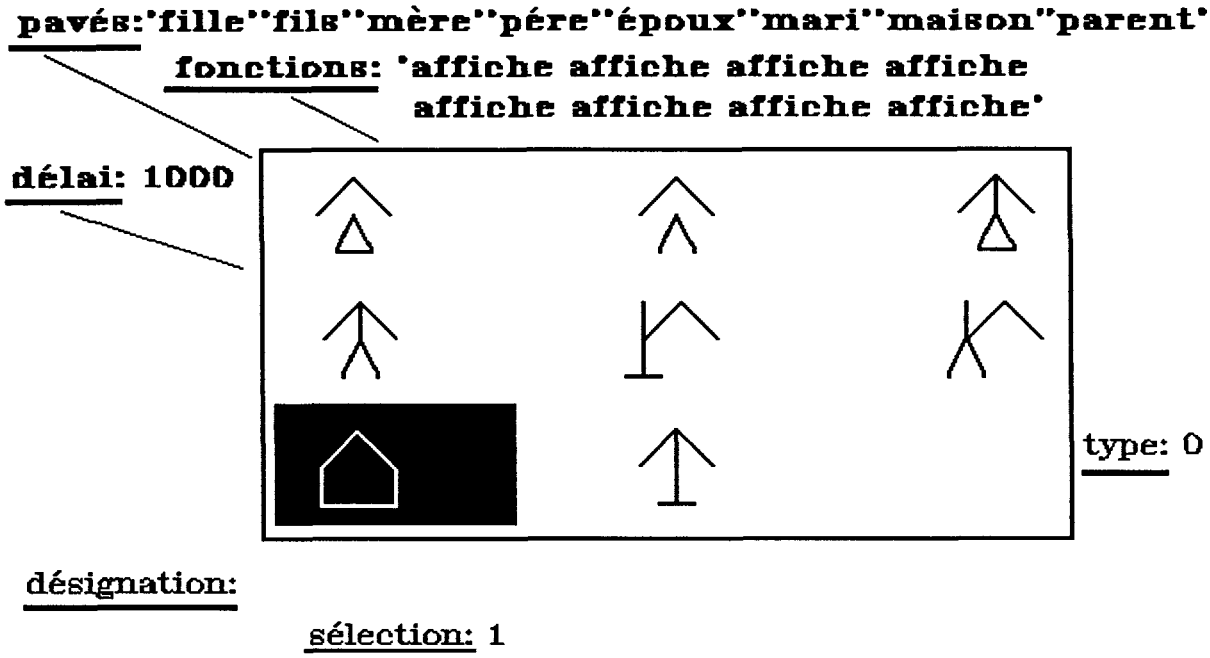


Figure II.3

La comportement d'un menu est très spécifique, très différent de celui des diverses fenêtres, qui existent dans le système Smalltalk de base (GraphPane, ListPane et TextPane). Nous avons donc bâti une fenêtre spécialisée adaptée aux menus.

Rappelons qu'une fenêtre délimite le lieu de l'écran où se déroule l'interaction entre une application et l'utilisateur.

Avec Smalltalk, toutes les applications interactives sont bâties suivant le même schéma appelé MVC (Modèle - Vue - Contrôleur)[Mév.87].

La démarche repose sur trois entités qui sont:

- l'objet sur lequel on veut travailler (Modèle),
- une interface de sortie, qui sert à présenter cet objet à l'utilisateur (Vue),
- une interface d'entrée qui permet d'interagir avec l'objet ou avec sa représentation (Contrôleur).

Le menu est l'objet sur lequel on va travailler. On l'appelle "Modèle" dans le système Smalltalk.

La Vue, représentation de l'objet sur l'écran, est l'interface de sortie. Elle s'appelle la "MenuPane" dans le communicateur.

Le Contrôleur est l'interface d'entrée, "MenuSélecteur" dans le communicateur.

Les relations entre un menu (modèle), sa MenuPane (vue) et son MenuSélecteur (contrôleur) sont présentées dans la figure II.4.

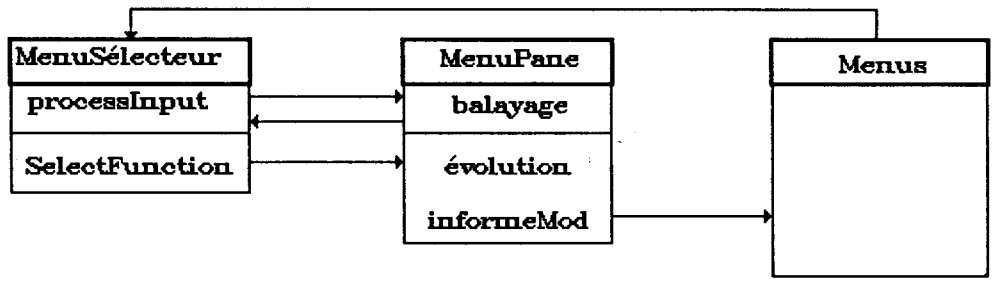


Figure II.4

La méthode "menus: unMenu" initialise les variables d'instance de la fenêtre, ce sont: balayage, délai, liste (liste des pavés affichés) et menu.

Pour pouvoir initialiser ses variables, la classe MenuPane questionne les classes Menus et Pavés. Celles-ci lui fournissent respectivement les informations, qui concernent le mode de balayage du menu et les pavés qui constituent le menu.

La méthode "methodMenu" questionne l'objet menus pour connaître le type du menu à visualiser (menu rectangulaire ou triangulaire). Enfin la méthode "showWindow" dessine les bords de la fenêtre et la méthode "afficheMenu" affiche les pavés qui constituent le menu.

Une fois le menu affiché, le contrôleur ("MenuSélecteur") fait évoluer le balayage et informe la menuPane et le modèle de l'état de l'organe d'accès du communicateur.

Si l'organe d'accès est activé, un message donnant le numéro du pavé sélectionné est envoyé dans la classe "Communicateur" à la méthode "évoluer: unIndex". Cette méthode demande au menu la fonction du pavé et applique la méthode associée à ce pavé. Ainsi, si le pavé est un pavé affichable, la méthode "affichGraph: aPavé" de la classe "Buffer" est activée. Cette méthode positionne le pavé dans le buffer et renvoie à la classe "Communicateur" le nom du nouveau menu à afficher.

II.6.2.1.3. La classe "Buffer".

La classe "Buffer" affiche les pavés dans la fenêtre de message. En fonction du pavé sélectionné, "Buffer" envoie le nom du nouveau menu à afficher. Les variables d'instance de la classe sont:

- type qui indique le type de représentation des pavés dans la fenêtre. Il existe trois types de fenêtre: les fenêtres "texte", "graphique" et "Bliss".

- ligne, donne le nombre de lignes de pavés qui compose le "buffer",

- pane qui contient la fenêtre associée au buffer activé,

- zone qui indique l'emplacement de la fenêtre de message (unRectangle),

- liste qui contient la liste de tous les objets mémorisés dans le buffer,

- position qui fournit les coordonnées du coin supérieur gauche du prochain pavé à afficher.

- fonction qui contient la méthode associée au "buffer". La méthode #message: mémorise et visualise l'élément informatif. La méthode #tampon visualise les primitives qui permettent la recherche de l'élément informatif.

La classe "Buffer" envoie, lorsqu'elle en reçoit l'ordre, son contenu à la classe "Transformation", afin qu'il soit transformé en un message compréhensible par l'environnement.

II.6.2.1.4. La classe "Transformation".

II.6.2.1.4.1. Introduction

Notre système doit pouvoir transformer le message abstrait primaire fourni par l'handicapé en un message abstrait secondaire compréhensible par son entourage. Le message primaire peut se présenter sous diverses formes telles que le français simplifié, le Bliss, le Par-le-si-la-b. Il faut pouvoir transformer, quelque soit sa forme, le message primaire soit:

- le plus souvent en français correct,
- en un message exploitable par l'environnement (synthétiseur vocal, commande domotique ...)

La transformation de fait en deux phases:

- une phase de traduction qui transforme le message primaire en français simplifié,
- une phase de transformation du message en français simplifié en français syntaxiquement correct.

Le buffer du système de communication contient les éléments informatifs de base sélectionnés par l'opérateur. Ces éléments font partie d'un langage quelconque. Le logiciel de traduction doit être indépendant du langage utilisé par l'opérateur. C'est pourquoi il faut transformer le message primaire en un message dans un langage unique qui lui sera traduit en français correct. Le langage intermédiaire que nous avons choisi est le français simplifié. A ce niveau, les informations connues sont constituées d'une part d'une suite de mots et d'autre part des classes syntaxiques de chacun de ces mots. Ainsi, à la phrase "je vouloir manger pain" est associée la liste: "pronom verbe_en_OIR verbe_en_ER nom".

II.6.2.1.4.2. Transformation du message primaire écrit en Bliss en français simplifié.

Dans le cas des langages pictographiques, chacun des éléments (les pavés) est un objet qui est notamment caractérisé par sa signification et par sa catégorie syntaxique.

Chacune de ces caractéristiques est définie au moment de la génération du pavé. Si la signification du pavé ne pose pas de problèmes, la définition syntaxique d'un mot peut faire appel à des connaissances scolaires oubliées. C'est pourquoi nous avons regroupé les mots en classes syntaxiques plus vastes qui ne prêtent pas à la confusion.

Ainsi le pavé  contient un élément pictographique Bliss et possède une variable d'instance appelée "traduction" qui contient le mot 'je' et une variable d'instance "catSynt" qui contient le nom de sa classe syntaxique "pronom".

II.6.2.1.4.3. Traduction d'un message en français simplifié en français syntaxiquement correct.

Après la transformation du message abstrait primaire en français simplifié, il faut le traduire en français syntaxiquement correct. Nous nous sommes limités dans notre étude à la traduction de phrases simples du type:

groupe sujet - groupe verbal - groupe complément

L'étude détaillée de la transformation est faite dans la thèse de Monsieur Jacques Tichon sous le titre: "Conception Assistée de Communicateurs pour Handicapés à l'Aide d'un Langage Orienté Objet: le problème de la Transformation".

II.6.2.1.5. La classe "Communicateur".

La classe "Communicateur" est une entité, qui permet d'initialiser les variables d'instance de tous les objets, qui constituent le communicateur. Ses données sont mémorisées dans la mémoire de masse du système (disque).

Les variables d'instance de cette classe définissent un communicateur, qui possède comme caractéristiques:

- un dictionnaire qui contient la liste des menus qui peuvent être appelés par le communicateur (variable d'instance: "menus"),

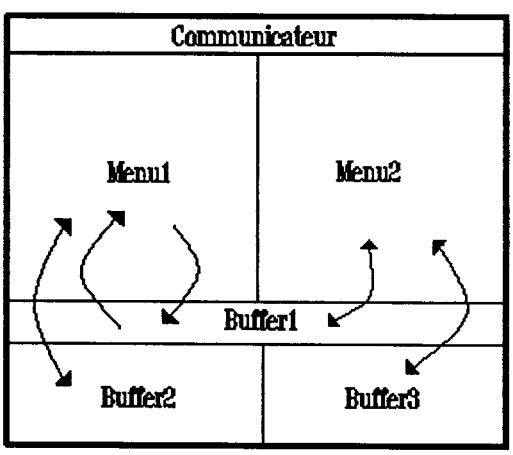
- le délai de balayage. C'est le délai par défaut. Il est pris en compte dans le cas où il n'a pas été défini dans un ou des menus (variable d'instance: "délai"),

- les caractéristiques du communicateur. Celles-ci ne peuvent être lues que par l'intermédiaire de la classe "Générateur". Elles fournissent à la personne chargée de la génération des renseignements tels que le nom de la personne qui a généré le communicateur, la date de génération, le nom de la personne utilisatrice, la liste des éventuelles modifications du communicateur... (variable d'instance: "caractéristiques"),

- la liste des "Buffers" du communicateur.

- la collection des noms des menus initiaux du communicateur (variable d'instance: initial).

- Les liens de dépendance entre les menus et les buffers sont représentés à la figure II.5.



Menu1	Agit sur le Buffer1 et sur le Buffer2
Menu2	Agit sur le buffer 2 et sur le buffer3

Dépendance des Menus aux Buffers

figure II.5

La méthode "OpenOn: unNom" permet de créer un objet de la classe "Communicateur", de créer en instanciant les objets des classes Menus et Pavés.

Cette méthode ouvre sur l'écran du communicateur la ou les fenêtres "menus", la ou les fenêtres de message ("buffers"). Elle active la classe MenuPane en lui envoyant comme message le nom du menu à visualiser ("menus: unMenu").

La structure minimale d'un communicateur en tant qu'objet ainsi que le lien entre les objets qui le constituent, sont présentés à la figure II.6.

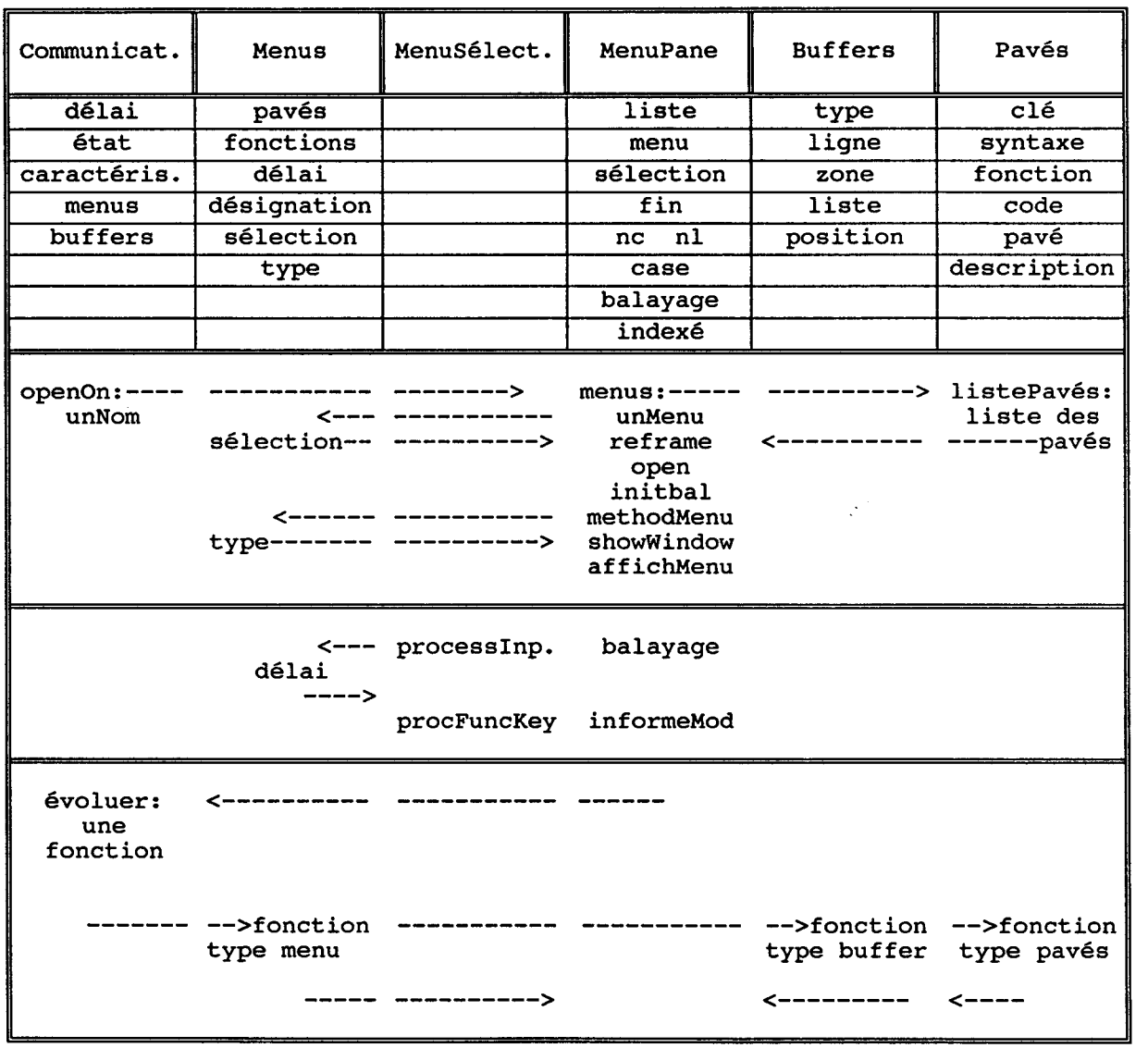


Figure II.6

II.6.2.2. Le générateur [TIC.89].

La figure II.7 présente l'organisation orientée objet du générateur de communicateurs.

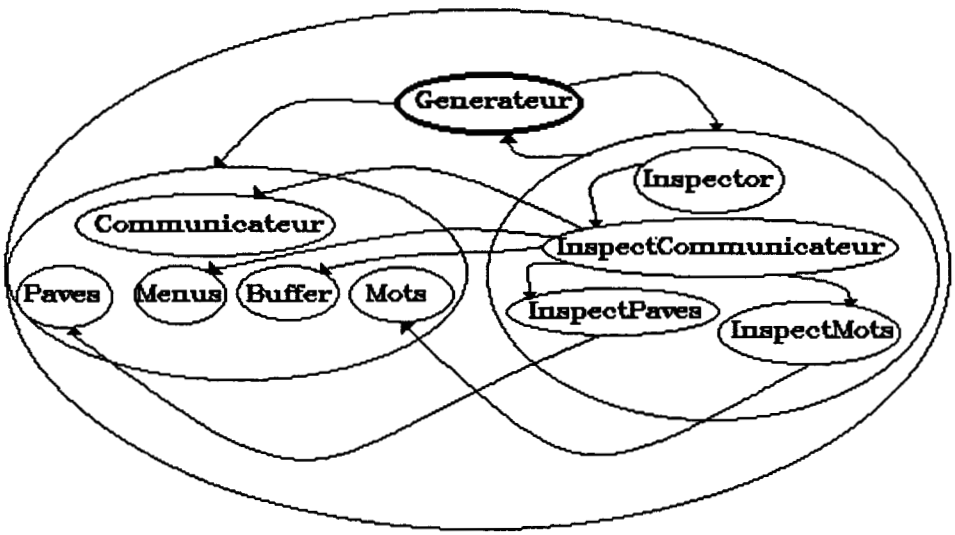


Figure II.7

II.6.2.2.1. La classe "Générateur".

Le générateur permet de créer des communicateurs.

La classe "Générateur", associée aux classes "InspectCommunicateur", "InspectMots" et "InspectPavés" permettent d'agir sur toutes les variables d'instance d'un communicateur.

Les variables d'instance du générateur permettent de le personnaliser. Ainsi la variable:

- "sigle" contient une forme représentant un logo, par exemple celui du centre hospitalier,
- "intitulé" contient une phrase telle que le nom du centre de rééducation,
- "listPane" contient la liste de tous les communicateurs déjà générés.

La méthode "open" ouvre une fenêtre sur le générateur. Celle-ci est constituée d'un ensemble de fenêtres. Grâce à divers menus, on peut soit:

- charger un communicateur existant afin de le modifier,

- créer un nouveau communicateur, le copier, le supprimer, le sauvegarder, le simuler, faire appel à l'éditeur de pavés ou à l'éditeur de mots.

Ce sont les classes "InspectCommunicateur", "InspectMots" et "InspectPavés" qui permettent d'accéder à n'importe quel objet d'un communicateur afin d'initialiser ou de modifier ses variables d'instance.

L'utilisation du générateur est décrite en annexe 1.

II.6.2.2.2. La classe "InspectCommunicateur".

Cette classe permet d'intervenir sur toutes les composantes du communicateur:

- sur lui même (sigle...)

- sur ses menus:

  création/suppression

  initialisation (type, vitesse, sélection...)

  modification (ajout/suppression d'éléments, disposition des pavés dans le menu...)

  imposer à un élément d'un menu une fonction particulière

- sur ses buffers (créer ou modifier)

- sur les pavés qui sont utilisés dans les menus afin d'en vérifier la constitution et d'imposer dans le menu la fonction de chacun des pavés (la fonction du pavé est liée au menu auquel il appartient)

- sur l'environnement du communicateur en permettant de créer des fenêtres (fenêtres, menus et buffers) et sur leurs dépendances.

Du point de vue hiérarchique, la classe "InspectCommunicateur" est une sous-classe de la classe "Inspector" qui existe dans le système Smalltalk de base. "InspectPavés" et "InspectMots" sont sous-classes de "InspectCommunicateur".

Cette structure hiérarchique permet d'utiliser les variables d'instance ainsi que les méthodes de la classe "Inspector".

La classe "Inspector" implémente une fenêtre relative à un objet. Elle permet de visualiser et de changer les variables d'instance de cet objet. La fonction "save" remplace la valeur de la variable d'instance sélectionnée par la valeur modifiée.

Les variables d'instance de la classe Inspector, utilisées par "InspectCommunicateur", "InspectPavés" et "InspectMots" sont:

- "objet": objet inspecté,
- "insList": une collection ordonnée ("Ordered Collection") des noms ou/et du nombre de fenêtres de type "liste",
- "instIndex": un entier, donne le numéro de l'objet sélectionné dans une liste affichée sur l'écran,
- "instPane": nom de la pane active.

Les variables d'instance de la classe "InspectCommunicateur" sont:

- "instance": contient la représentation ASCII de la variable d'instance sélectionnée,
- "liste": est une "Array" constituée des noms de l'ensemble des objets de la classe inspectée,
- "menu": contient le menu relatif à la "Pane" activée,
- "pane": contient le nom de la pane activée,
- "fenêtres": est une liste de rectangles qui contiennent les dimensions des menus et buffers en activité.

- "notes": est une "Array" constituée de l'ensemble des objets de la classe inspectée.

Les méthodes "Open" ouvrent des fenêtres, qui permettent la construction logique d'un communicateur. Ainsi la méthode "open: unCom" ouvre une fenêtre sur le communicateur "unCom" et permet d'accéder à tous les objets de ce communicateur.

La structure, la hiérarchie et le lien entre les divers objets sont présentés à la figure II.7

II.6.2.2.3. La classe "InspectPavés".

La classe "InspectPavés" permet de visualiser l'ensemble des pavés mémorisés dans divers dictionnaires.

"InspectPavés" possède les méthodes, qui lui permettent de créer et d'ajouter des nouveaux pavés (méthode: créeP), de retirer les pavés d'un dictionnaire (méthode: supprimeP), d'ouvrir, de supprimer, de modifier des dictionnaires (méthodes: créer, supprimer); d'appeler un éditeur de dessins afin de pouvoir construire ou modifier un pavé préalablement sélectionné.

La classe "InspectPavés", par la méthode "dessin", ouvre un éditeur de dessin. Celui-ci permet de dessiner n'importe quel pavé. Une classe "Dessin" a donc été créée.

II.6.2.2.4. La classe InspectMots

La classe InspectMots permet d'accéder aux mots du ou des dictionnaires, qui sont utilisés pour transformer le message abstrait primaire en un message secondaire en français correct (classe Transformation).

Cette classe permet d'ajouter, d'enlever des mots de ce ou ces dictionnaires.

Cette classe est décrite au chapitre III.

II.6.3. Implantation du système de communication dans Smalltalk.

La figure II.8 présente l'implantation des diverses classes du communicateur dans le système Smalltalk.

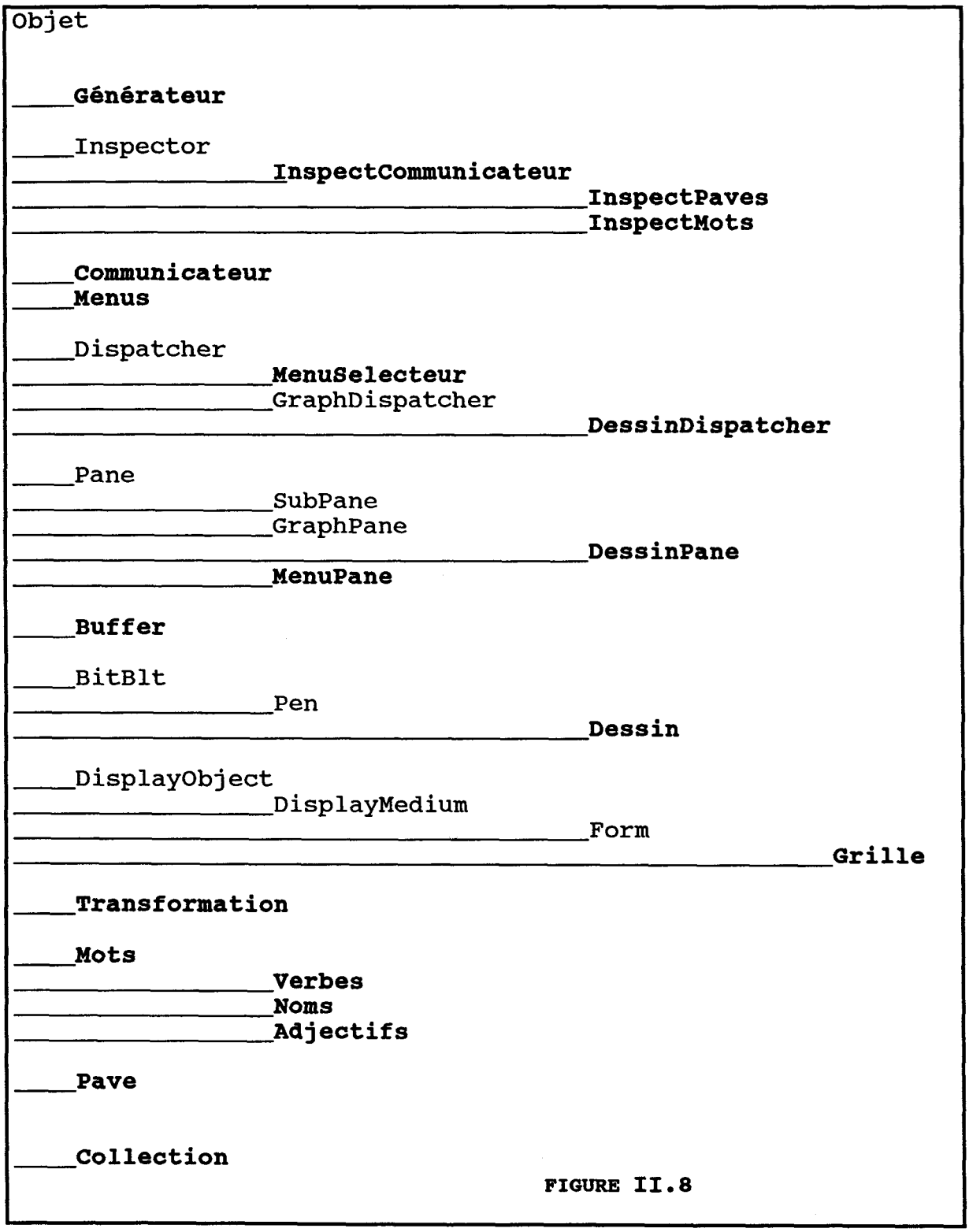


FIGURE II.8

II.7. CONCLUSION

Le projet CACHALOT (Conception Assistée de Communicateurs à l'Aide d'un Langage ObjeT) répond parfaitement aux exigences d'un générateur de communicateurs pour handicapés. La réalisation de prototypes est aisée.

Le grand avantage de l'utilisation de la programmation objet réside dans le fait que le générateur de communications peut facilement être complété ou modifié. Augmenter la puissance du générateur revient à ajouter des méthodes et des variables à la classe "Générateur" et cela, sans modifier les méthodes existantes. L'étude des méthodes du système initial n'est pas nécessaire. Le programmeur ne doit s'intéresser qu'au comportement externe des méthodes: quel est le message à envoyer à une méthode, quel est son effet, quel est le message réponse?

L'utilisation systématique de fenêtres, menus fugitifs, et l'utilisation de la souris procurent des logiciels très ergonomiques tout en fournissant une présentation très moderne du générateur. Cette convivialité est un atout très important permettant une grande souplesse d'utilisation du logiciel.

Qui dit prototype pense évaluation. Une aide informatique à l'évaluation devrait être développée et associée aux communicateurs générés.

Ces opérations pourraient se faire automatiquement en temps réel au moyen d'un système expert, qui pourrait évaluer à chaque instant la qualité de la communication et se charger d'adapter le communicateur de façon continue à l'état de forme de l'handicapé. Ainsi, la vitesse de balayage s'adapterait en tenant compte de la vitesse de réaction de l'handicapé dans le choix des éléments informatifs, ainsi que du taux d'erreur dans le choix de ces éléments.

Dans le même ordre d'idée, le nombre de pavés présentés dans les menus pourrait également varier automatiquement en fonction de l'état de fatigue de l'handicapé.

La position des pavés dans le menu pourrait être modifiée par le système en fonction de leur fréquence d'utilisation, afin de réduire le temps de balayage.

Les pavés, qui ne sont jamais utilisés, pourraient être supprimés. Cela n'est possible que pour des opérateurs capables d'intégrer rapidement les modifications apportées à la présentation des menus. En effet, une perte d'automatisme peut avoir un effet néfaste sur le temps de réaction de l'utilisateur et nécessite un effort intellectuel plus important.

CHAPITRE III

COMMUNICATEUR BLISS UTILISANT LA MÉTHODE BINAIRE COMME MÉTHODE D'ACCÈS À UN DICTIONNAIRE

III.1. INTRODUCTION.

Notre objectif est de réaliser des systèmes de communication pour handicapés qui améliorent le rapport "effort nécessaire à la production d'un message/vitesse de production" de ce message. Ce chapitre est décomposé en trois parties qui sont:

- la présentation du langage Bliss, langage idéographique utilisé par de nombreux handicapés.

- la description d'une méthode accélérée de recherche d'éléments informatifs d'un dictionnaire par accès désordonné.

- l'implantation de cette méthode d'accès dans le système de prototypage écrit en Smalltalk et présenté au chapitre 2.

Le Bliss est un langage utilisé par un grand nombre d'handicapés. Il utilise des symboles idéographiques comme éléments informatifs. Communiquer, c'est pour l'handicapé choisir et assembler des idéogrammes. Si l'on veut lui permettre de pouvoir utiliser toute la richesse du vocabulaire disponible, il est souvent très vite limité par le temps et l'effort physique nécessaire à la recherche de ces idéogrammes.

La solution est de pouvoir décomposer chacun des éléments idéographiques d'un dictionnaire en un nombre restreint d' idéogrammes élémentaires et d'ainsi permettre à l'utilisateur de reconstruire un idéogramme informatif à partir des composantes d'idéogrammes et cela de la manière la plus efficace possible.

III.2. PRÉSENTATION DU LANGAGE BLISS

III.2.1. Historique

Le Bliss est un langage qui a été mis au point par Charles K.Bliss (1897-1985). A Shanghai, en 1942, il remarque que les Chinois malgré des dialectes différents entraînant des difficultés à se comprendre, arrivent à se lire. Leur écriture est en effet basée sur des symboles standardisés.

Initialement, Bliss a développé un langage international dont le but était de promouvoir la compréhension entre les peuples. En 1971, le Bliss est utilisé pour la première fois dans un centre pour enfants handicapés de Toronto. L'objectif est de mettre au point un système de communication qui sert de complément ou de substitut à la parole pour des enfants au stade de la prélecture.

Bliss permet à l'utilisateur de communiquer des besoins fondamentaux aussi bien que des pensées sophistiquées ou des concepts abstraits.

Un petit nombre de composantes Bliss de base permet, en les assemblant dans un ordre logique, de générer des symboles complexes.

Le choix des symboles Bliss peut varier en fonction de l'état intellectuel de l'utilisateur.

Le Bliss s'est depuis répandu dans beaucoup d'autres pays, dont la France en 1980, et fait l'objet d'applications auprès de populations très diverses: handicapés moteurs, mentaux, personnes présentants des troubles de perception auditive...

III.2.2. Le langage Bliss

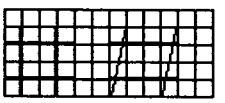
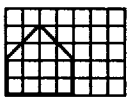
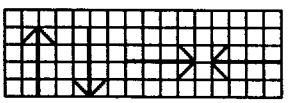
Le Bliss est un système visuel, basé sur la signification. Chaque composante apporte une orientation sémantique à l'ensemble du symbole. L'utilisation des symboles Bliss permet la représentation directe de sens dans des configurations visuelles.

Les symboles sont constitués d'éléments pictographiques, idéographiques, internationaux ou arbitraires.

Beaucoup de composantes Bliss se rapprochent du vécu des jeunes enfants. Il offre une grande variété de significations, d'émotions, d'idées, de questions, d'actions, tout en restant facile à comprendre.

L'utilisation du Bliss comme moyen de communication pour handicapés moteur est intéressante. Malgré un handicap au niveau de ses mouvements, un handicapé peut combiner un petit nombre de symboles pour accéder à un vocabulaire étendu.

Le dictionnaire, édité en 1980, comprend environ 1400 symboles et s'est depuis largement étendu. Ils sont classés soit par thèmes, soit selon les éléments de base qui les constituent. On a ainsi défini vingt six classes d'éléments. A chacune de ces classes est attribuée une lettre de l'alphabet. Ainsi:

- classe "c": lignes croisées 
- classe "d": bâtiments 
- classe "f": les flèches 
- classe "g": les roues 

Les symboles sont représentés dans le dictionnaire dans des grilles divisées en sous-cases. Cela permet aux utilisateurs de reproduire facilement les symboles en respectant le quadrillage, de positionner des symboles élémentaires l'un par rapport à l'autre pour former des symboles complexes.

III.3. STRUCTURE D'UN DICTIONNAIRE. [TOU.90]

Les dictionnaires contiennent les éléments informatifs, le lexique du langage. On peut envisager le problème de l'accès aux éléments d'un dictionnaire sous deux aspects différents:

- utiliser des méthodes performantes qui permettent d'accéder, le plus rapidement possible, à une donnée stockée, en mémoire, dans un dictionnaire,

- anticiper, à l'aide d'un minimum de données, la sélection de la donnée (méthode fréquentielle statique, fréquentielle dynamique, ..., recherche binaire).

Nous avons développé un système qui met en évidence l'intérêt de l'accès par anticipation utilisant la recherche binaire. Dans les systèmes de communication pour handicapés, il faut pouvoir accéder aux informations à partir d'un minimum d'indices (éléments informatifs de base). On décompose souvent le dictionnaire principal en ensemble de sous-dictionnaires de manière à diminuer le nombre d'éléments informatifs contenus dans les dictionnaires. Il existe, en fonction du type d'informations contenues, des dictionnaires de deux types:

- les dictionnaires à accès ordonné,

- les dictionnaires à accès désordonné.

Les dictionnaires qui contiennent des mots sont habituellement structurés de telle manière que leur accès se fasse de manière ordonnée.

Ainsi la méthode d'accélération par bigrammes tête de mot permet d'accéder à tous les mots qui commencent par ce bigramme. Les mots du dictionnaire sont classés par ordre alphabétique et l'on choisit les éléments informatifs élémentaires dans l'ordre des lettres du mot. Le bigramme **éc** permet, par exemple, d'accéder dans le sous-dictionnaire verbe, aux mots:

ÉCARTER - ÉCLAIRER - ÉCOUTER - ÉCHANGER - ÉCONOMISER....

Un dictionnaire peut contenir autre chose que des mots, par exemple des symboles idéographiques. Dans ce cas l'handicapé choisit un certain nombre de composantes graphiques élémentaires. Celles-ci permettent de retrouver le symbole informatif dans le dictionnaire [SHA.90]. Les symboles élémentaires sont présentés à l'handicapé sous forme de menus visualisés sur l'écran d'un ordinateur, ils sont choisis par l'opérateur de manière quelconque. Dans ce cas, l'accès au dictionnaire doit se faire de manière désordonnée.

Ainsi pour choisir le symbole idéographique représentant un motel  , l'opérateur pourra faire le choix des symboles élémentaires (composantes) dans n'importe quel ordre:

- 
- 
- 
- etc....

Dans ce cas, quelque soit l'ordre des symboles choisis, le premier choix est trop vague pour pouvoir trouver l'idéogramme de motel. Si l'opérateur a choisi comme premier symbole informatif  (personne) le nombre d'idéogrammes possible est supérieur à 100. L'opérateur effectue un second choix: le symbole  (maison) par exemple. Cette seconde sélection permet de réduire le nombre de idéogrammes possible à cinq. Ces cinq symboles informatifs sont visualisés sur l'écran. L'opérateur effectue alors son choix définitif.

Ces composantes élémentaires ont été choisies de manière telle qu'il soit possible, à partir de celles-ci, de reconstituer n'importe quel symbole. De plus, on anticipe le choix de l'handicapé en lui présentant les éléments informatifs les plus probables.

Un certain nombre de symboles complexes ont également été retenus car ils sont:

- utilisés fréquemment dans la conception de mots Bliss,

- ont une signification importante et sont souvent utilisés (maison, personne...).

III.4. MÉMORISATION DES SYMBOLES BLISS DANS UN DICTIONNAIRE.

Dans un système Smalltalk, un symbole Bliss peut être mémorisé en mémoire de masse en tant qu'objet de la classe "Form". Il occupe alors une place mémoire importante. En effet, la mémorisation du symbole se fait pixel par pixel. On mémorise bit par bit l'état allumé ou éteint de chacun d'eux.

Afin de diminuer la dimension du dictionnaire Bliss, nous avons codé chacun d'eux. Ce codage se fait par rapport à la grille Bliss. Un symbole est construit à l'aide de composantes Bliss. Ces composantes sont des formes installées en mémoire centrale. Ainsi le symbole représentant un motel est construit à l'aide de trois composantes (figure III.1). Le symbole est représenté en mémoire par le code suivant:

```
 0  12  )36  .  5  12  )38  .  8  14  )51
|   |    |  |
 x   y   clé  délimiteur
      du pavé
    élémentaire
                    x   y   clé  délimiteur ...
```

- x, y représentent les coordonnées du coin supérieur gauche de la forme contenant la composante,
- clé permet l'accès au dictionnaire de composantes,
- un délimiteur sépare dans le code chacune des composantes

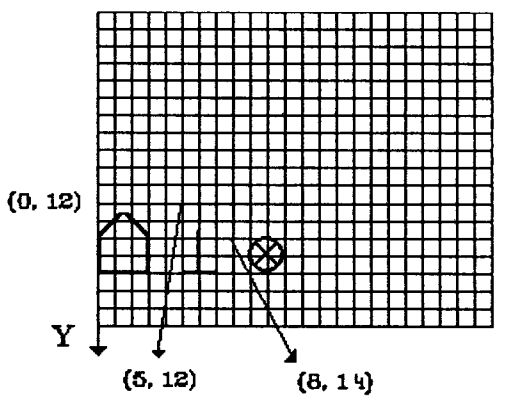


figure III.1

De même le symbole "parent"  est construit à l'aide des composantes  et .

III.5. ÉDITEUR DE DESSINS.

Les symboles Bliss et les composantes graphiques sont construits à l'aide d'un éditeur de dessins intégré que nous avons développé. Cet éditeur fait partie de la classe "Dessin".

La classe "Dessin" est sous-classe de la classe "Pen" qui est elle-même sous-classe de "BitBlt".

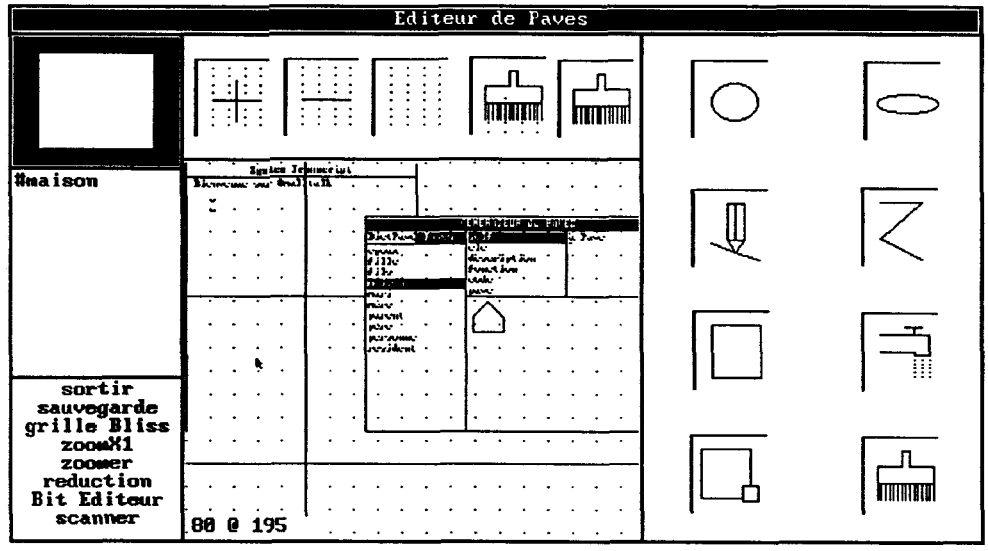


Figure III.2

- La classe "BitBlt"

La classe "BitBlt" regroupe toutes les opérations graphiques de base. Sa fonction principale est de transférer des bits d'une "area" dans une autre.

Elle inclut trois formes: la forme source,la forme destination et la forme masque.

Les bits de la forme source sont d'abord soumis à une opération logique ET avec les bits de la forme masque et ensuite fusionnés dans la forme de destination en utilisant une règle de combinaison. La règle de combinaison est une opération logique AND, OR, XOR...(figure III.3).

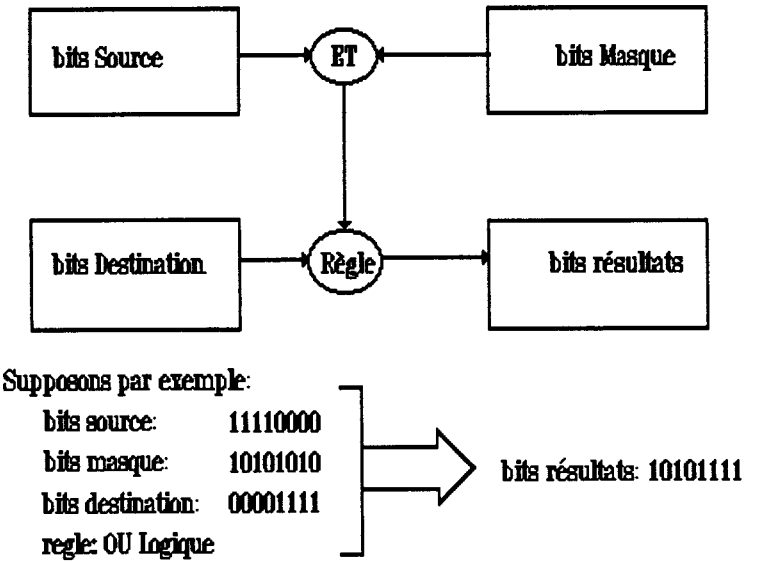


Figure III.3

- La classe "Pen"

La classe "Pen" étend les fonctions de la classe "BitBlt" en procurant une interface de type "tortue graphique".

La forme source sert à définir l'épaisseur de la plume, la forme masque détermine la couleur de la plume, la forme destination sert de canevas pour dessiner. La classe permet de dessiner en copiant de manière répétitive les bits de la forme source vers la forme destination.

- La classe "Dessin"

La classe "Dessin" implémente une série de méthodes, qui permettent de dessiner facilement:

- à main levée ("trace:"),

- des droites ("droite:"),

- des cercles ("cercle:"),

- des ellipses ("ellipse:"),

- des rectangles ("rectangle:"),

- remplir une zone ("remplir:"),

- de gommer ("gomme:").

Le curseur est soit représenté par deux droites parallèles aux axes de référence de l'écran, soit par une flèche. Ces deux droites sont mobiles. La position du curseur est définie par l'intersection de ces deux droites. Le curseur qui se déplace grâce à la méthode "déplaceCurseur" est activée par la méthode "processInput" de la classe associée "DessinDispatcher" (figure III.4). Il se déplace soit en continu (point par point), soit uniquement sur la grille du dessin.

Cette classe est le modèle d'un système dont la vue est assurée par la classe "DessinPane" et la coordination par la classe "DessinDispatcher".

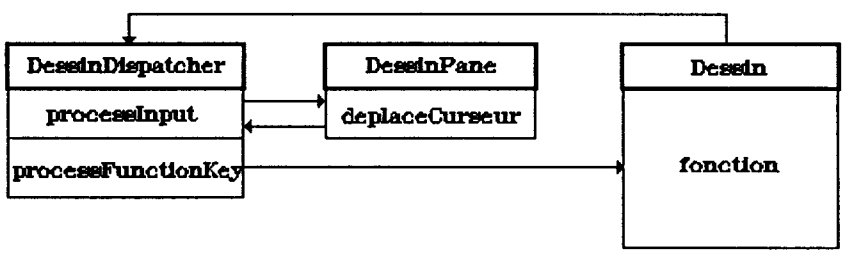


Figure III.4

Une grille, dont ont peut régler le pas et l'affichage automatique des coordonnées permet de positionner le curseur avec précision.

Des méthodes récupèrent des pavés qui existent, les positionnent dans la fenêtre dessin, les modifient, les fusionnent.

La méthode "scanner" visualise des images provenant d'un scanner.

La méthode "macro" agrandi la forme dessinée et permet de dessiner avec précision, point par point.

Le pavé étant dessiné, on peut choisir la manière dont le graphique sera codé afin d'être mémorisé (bitmap, bitmap compressé, code graphique Bliss, chaîne de caractères).

III.6. Accès à un dictionnaire Bliss.

Comment accéder à un symbole idéographique du dictionnaire? L'accès au dictionnaire se fait d'une manière désordonnée par l'intermédiaire de menus, l'handicapé choisit ses symboles élémentaires dans un ordre quelconque. Dans le cas du langage Bliss, les symboles informatifs peuvent être considérés comme des assemblages d'éléments graphiques élémentaires. Nous avons réalisé une étude graphique qui nous a permis de prévoir une structure d'élaboration des formes Bliss. En étudiant le dictionnaire Bliss afin de connaître le nombre et la taille des symboles informatifs. Cela nous a permis de déduire les normes à adopter pour la réalisation d'un symbole. Nous avons ensuite décomposé les différents symboles en éléments simples que nous avons appelés composantes élémentaires ou primitives.

Afin de les dimensionner, on représente les symboles Bliss dans une grille. Une première analyse nous a conduit à dénombrer les symboles existants actuellement et de définir leur taille en fonction du nombre de sous-cases de la grille qu'ils utilisent (figure III.5). On ne tient compte que de la longueur des idéogrammes. Sur un total de 1369 symboles (dictionnaire 1980):

- 51,13% d'entre eux ont une taille inférieure à 5 sous-cases.
- 31,3% ont une taille comprise entre 6 et 9 sous-cases.
- 16,87% ont une taille comprise entre 10 et 14 sous-cases.
- 0,065% ont une taille de plus de 14 sous-cases (8 symboles).

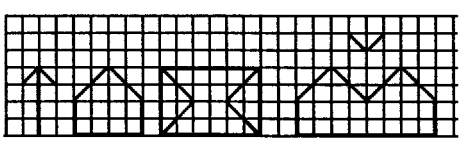


figure III.5

Nous nous sommes servis de la décomposition en éléments alphabétiques des symboles pour dégager des éléments simples. Nous avons ainsi trouvé 76 formes élémentaires différentes (sans tenir compte des orientations). Ces formes sont présentées dans la table des primitives de la figure III.3 (cette figure contient 49 primitives, nous n'avons pas représenté les primitives de même forme mais de taille différente).

Ces primitives se composent de chiffres, de signes de ponctuation, de figures géométriques simples telle que des lignes, des carrés, des triangles, des cercles, de plusieurs tailles) d'éléments en trait fin représentants les indicateurs et enfin d'éléments plus représentatifs tel que le coeur, l'oreille ou la maison (figure III.6).

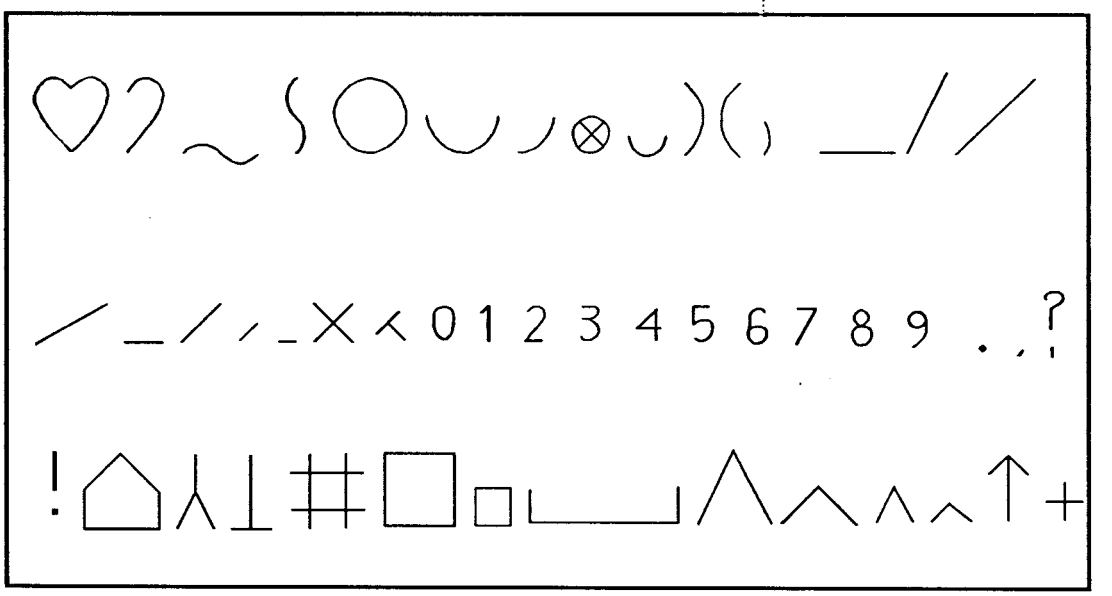


figure III.6

Après avoir fait cette décomposition, nous avons vérifié que l'ensemble des primitives permet la reconstruction de n'importe quel symbole informatif.

Cette décomposition va nous permettre:

- d'appliquer la recherche binaire à la recherche d'un mot Bliss dans un dictionnaire.

- d'utiliser ces composantes pour coder les pavés Bliss. Cela permet de diminuer efficacement la taille mémoire occupée par le dictionnaire des pavés Bliss . Le gain de temps ainsi obtenu est partiellement perdu par la nécessité de reconstituer les pavés.

III.7. ACCÈS À UN DICTIONNAIRE BLISS PAR RECHERCHE BINAIRE.

Le problème est donc de choisir une méthode qui permette à partir du plus petit nombre possible d'éléments sélectionnés dans un ordre quelconque de retrouver un symbole graphique dans un dictionnaire.

Nous avons choisi la méthode par recherche binaire. Ce type de méthode a été développé dans le cadre de recherches d'informations à partir de systèmes qui utilisent des bases de données importantes. On crée une matrice binaire (de dimension deux) ayant en ordonnée les mots et en abscisse les symboles élémentaires. Ainsi, si chacun des symboles informatifs d'un langage peut être décomposé, par exemple, en 76 idéogrammes élémentaires, on crée un dictionnaire structuré suivant le modèle d'une matrice binaire de type (m x 76) ou m représente le nombre de symboles informatifs du dictionnaire.

Dans ce dictionnaire on affecte à chaque mot Bliss 76 variables. Chacune de ces variables se trouve dans l'état "1" ou "0" suivant que la composante élémentaire liée à une variable participe à la construction du mot. Le contenu de cette matrice est facilement lisible. Ainsi dans une matrice (ixj), il suffit de lire le mot de la ligne i pour connaître tous les symboles élémentaires qui le constituent. De même, en lisant la colonne j, on obtient tous les mots ayant pour propriété d'utiliser l'idéogramme élémentaire correspondant.

Il est facile de déterminer un sous-ensemble ou un seul mot ayant les propriétés requises par multiplication de la matrice: M(m,n) par une matrice question: Q(n,p). On obtient la matrice réponse: R(m,p)

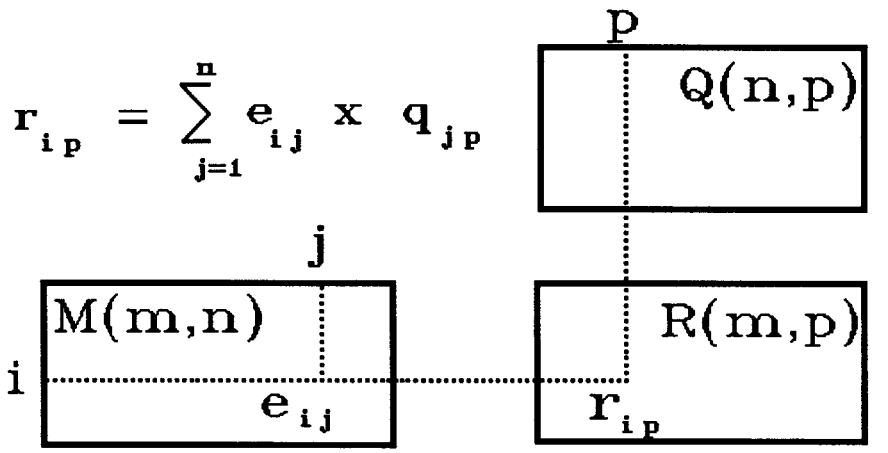


figure III.7

Le mot i possède toutes les propriétés requises si:

ri = q (q = poids du vecteur question Q)

(figure III.7).

Dans le cas d'un dictionnaire Bliss auquel on accède par recherche binaire. La matrice présente en horizontal l'alphabet de la base de données et en vertical les symboles Bliss. La matrice est représentée dans la figure III.8:

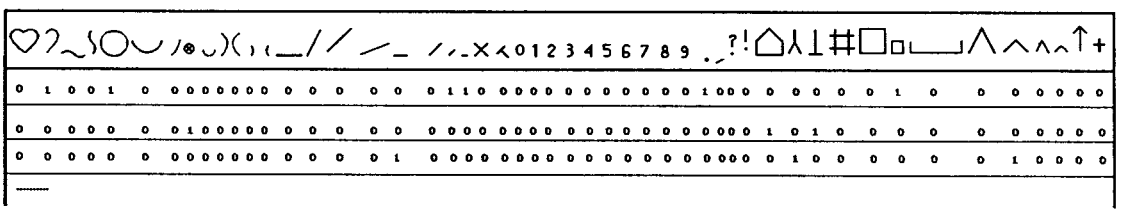
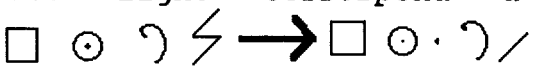


Figure III.8

La première ligne correspond à l'idéogramme de "télévision"  , la seconde de l'idéogramme de "motel", la troisième ligne de celui de "maman"

Remarquons toutefois que des ambiguïtés sont possibles. Ainsi un même mot peut être construit à partir de composantes élémentaires différentes.

Une même composante élémentaire peut être utilisée à plusieurs reprises dans la description d'un idéogramme mais n'est visible qu'une seule fois dans la matrice M(m,n) d'où une perte d'information.

Enfin, l'utilisation des matrices binaires ne permet pas de positionner les composantes élémentaires. L'ensemble des symboles élémentaires sélectionnés n'est alors pas toujours suffisant pour retrouver l'idéogramme informatif. Il convient alors de lever l'ambiguïté en affichant les idéogrammes incertains afin d'effectuer un choix définitif.

Cependant cette technique présente, surtout pour les handicapés profonds, un compromis très intéressant entre la vitesse de production d'un message et l'effort nécessaire à la production de ce message.

D'autant plus qu'elle ne nécessite pas de l'handicapé de devoir positionner les symboles dans une grille. Ceci est d'autant plus vrai que l'écriture directe nécessite un positionnement des symboles dans une grille, ce qui peut s'avérer difficile voir impossible dans le cas des handicapés lourds.

Par ailleurs un menu de symboles permet difficilement un accès à un vocabulaire important à cause des temps de balayage.

Elle présente l'avantage d'utiliser une structure moins complexe que celle que l'on retrouve dans les bases de données courantes ce qui permet d'écrire les logiciels de gestion de la matrice binaire en langage machine. Le temps d'exécution des logiciels est ainsi fortement réduit. La matrice binaire présente aussi l'avantage d'utiliser peu d'espace mémoire ce qui permet de l'implanter dans la mémoire vive de l'ordinateur. Le système ne fait pas appel à la mémoire de masse (disque dur) et accélère donc la gestion de la matrice. En effet, un mot Bliss est codé en huit octets. Un dictionnaire de 1.400 mots Bliss n'utilise donc que: 1.400 x 8 = 11.200 octets.

Ceci permet à la personne chargée de générer un communicateur de créer et de manipuler facilement une banque de données avec un minimum de connaissances informatiques.

Le contenu des menus peut varier en fonction des composantes choisies grâce à l'utilisation de pavés modulateurs.

III.8. BASE DE DONNÉES [DEL.82].

III.8.1. Introduction.

L'implantation de la méthode de recherche par matrice binaire se fait au sein d'un système de gestion d'une base de donnée. Les renseignement fournis par la recherche dans la matrice doivent nous permettre d'accéder rapidement à l'information stockée en mémoire de masse.

Après avoir rappelé la théorie générale liée à l'organisation des fichiers et des bases de données, nous décrivons la structure que nous avons choisie ainsi que son implantation dans le système de base Smalltalk.

III.8.2. Organisation des fichiers.

III.8.2.1. Introduction

D'une manière générale, l'organisation des fichiers peut être faite de manière:

- séquentielle,
- séquentielle indexée,
- adressée.

III.8.2.2. Fichiers séquentiels.

Dans ce type de fichier, les enregistrements sont placés les uns derrière les autres, sans ordre. Tout nouvel enregistrement est stocké en fin de fichier.

L'avantage de ce type d'organisation est le taux élevé d'occupation du support physique. Par contre les procédures de suppression et de modification d'un enregistrement sont coûteuses puisqu'elles impliquent en général une réorganisation complète du fichier; en particulier si l'on veut réutiliser les espaces libérés et éviter une dégradation des performances.

L'inconvénient majeur est que la recherche d'un enregistrement donné demande un balayage séquentiel de toutes les données stockées, opération d'autant plus coûteuse en temps que le fichier est important.

III.8.2.3. Fichiers séquentiels indexés.

Afin d'améliorer les performances de la structure précédente, on lui associe une table d'index permettant d'accélérer la recherche et d'assurer un accès quasi direct aux données stockées en mémoire de masse.

La table d'index associe à la clé de chaque donnée sa position dans le fichier.

L'avantage de cette organisation, par rapport à celle décrite précédemment, est de permettre un accès plus rapide à la donnée. Il peut rester inacceptable si la table d'index est importante et stockée en mémoire secondaire.

III.8.2.4. Fichiers triés

Cette organisation utilise des techniques plus complexes afin d'insérer, de modifier et de supprimer des données.

L'idée de base est de répartir les données dans des blocs composés d'une ou plusieurs pages.

III.8.2.4.1. Tri croissant ou décroissant

Le tri se fait par valeur croissante ou décroissante des clés associées aux enregistrement.

Cette technique permet des recherches élaborées telle que la recherche dichotomique.

III.8.2.4.2. Hash coding

A chaque clé d'enregistrement correspond une valeur obtenue grâce à une fonction de hachage (algorithme de conversion de clé).

Tous les enregistrements qui ont la même valeur de clé sont regroupés dans le même bloc.

Afin que ce type d'organisation soit efficace, il faut que la technique de hachage découpe l'ensemble des données en blocs équilibrés. Cela demande de déterminer la fonction de hachage adéquate (peu facile!). Il n'existe aucune méthode qui permette de trouver cette fonction.

III.8.2.4.3. Conclusion

L'organisation en fichiers triés, si l'on choisit la solution la mieux adaptée au problème posé, présente un coût de recherche peu élevé et permet une recherche plus élaborée des données (possibilité de questions).

Par contre, le taux d'occupation est beaucoup plus faible que dans les fichiers séquentiels et séquentiels indexés (perte de place mémoire).

Elle demande des connaissances en informatique de la part de l'utilisateur du générateur afin d'optimiser la base de données.

III.8.2.5. Amélioration des performances d'un fichier indexé.

D'une manière générale, toute table d'index peut être considérée comme un fichier. On peut ainsi lui appliquer, afin d'accélérer la recherche dans la table d'index, les techniques de tri de type "hash coding". On peut lui appliquer les mêmes méthodes d'accélération que celles qui sont utilisées pour les fichiers normaux.

Comme pour les fichiers triés, il existe un gaspillage de la place mémoire mais moins important que celle qui est perdue dans un fichier cible.

Le "hash coding" permet de travailler sur une partie réduite de la table d'index qui sera alors implantée en mémoire centrale.

III.8.3. Base de donnée de notre système.

III.8.3.1. Introduction

Dans notre système, la base de données est composée (figure III.9):

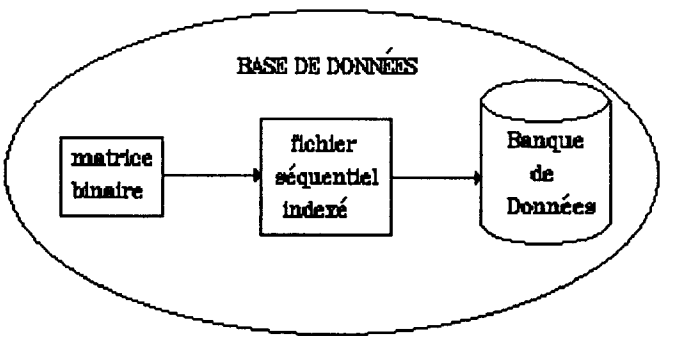


Figure III.9

- d'une banque de données qui est une mémoire de masse (disque dur). Elle contient l'ensemble des objets (pavés) auxquels nous désirons accéder,

- une table d'index qui contient la position en numéro d'enregistrement de tous les objets dans la mémoire de masse,

- d'un système de gestion de la base de données qui comprend:

  - une table binaire ou matrice binaire installée en mémoire centrale et structurée suivant le modèle d'une matrice binaire,

  - un modèle fonctionnel du système de gestion de la base de données. Il permet:

    - l'interrogation des données

    - la mise à jour (addition, modification, suppression de données).

Nous avons créé la classe "BDSeq" (Base de Données Séquentielles) qui remplit ces fonctions.

III.8.3.2. Implantation de la base de données dans Smalltalk.

Les données auxquelles nous devons accéder sont contenues dans la Banque de Données (disque dur). Le système Smalltalk de base contient une classe spécialisée qui remplit cette fonction: la classe "FileStream". Elle permet le transfert, à partir d'une position courante du contenu de fichiers DOS et cela en lecture et écriture (figure III.10).

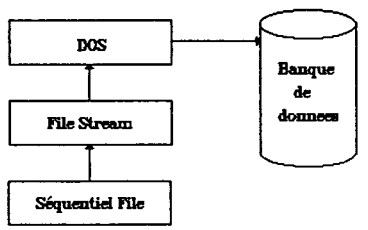


Figure III.10

Nous avons créé une sous classe de "FileStream" qui contient les méthodes spécifiques à notre système de gestion de la base de données: la classe "SéquentielFile". Elle contient les méthodes qui permettent d'accéder à un enregistrement à partir d'un fichier d'index. Cette classe possède une variable d'instance:

- indice: c'est une table d'octets (ByteArray) qui contient la première position de chaque objet de la banque de données et qui reste en mémoire centrale.

Notre système de gestion de la base de données est constitué par la classe "BDSeq" (Base de Données Séquentielles). Elle contient toutes les méthodes qui permettent le fonctionnement classique d'une base de données (figure III.11).

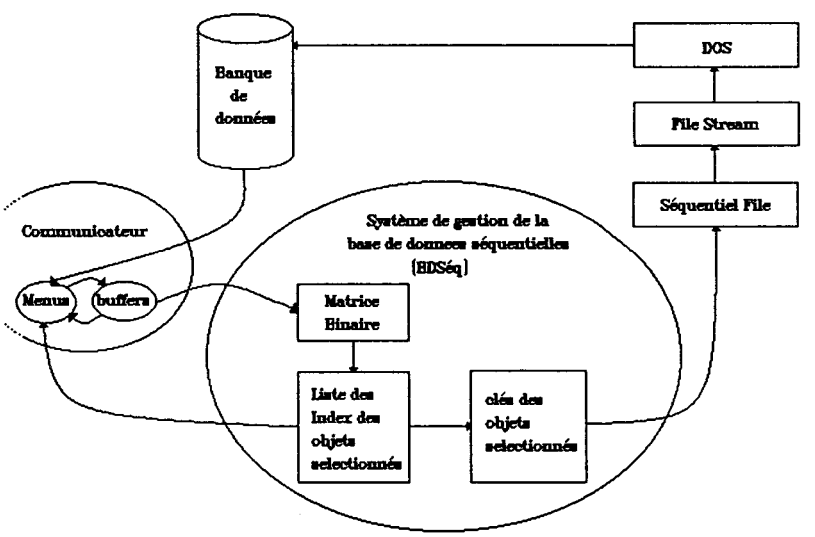


Figure III.11

Les variables d'instance principales de cette classe sont:

- "alphabet": rassemble la liste des constituants. Dans le cas d'un communicateur Bliss, c'est la liste des composantes élémentaires du langage Bliss.

- "cléAccès": est une liste qui contient toutes les variables d'instances (des objets stockés dans 'BDSeq') qui servent à la sélection d'un ou plusieurs éléments dans la base de données.

- "selectMax": est utilisé pour la recherche binaire. Elle contient un entier qui représente le nombre d'objets à partir duquel on arrête la recherche pour la sélection finale.

- "buffer": est utilisée pendant la recherche binaire. Elle contient, soit la liste d'index de tous les objets sélectionnés, soit la liste des objets possibles (quand la recherche est terminée).

- "matrice": c'est la matrice binaire. Elle est structurée comme une "bytesArray". "matrice" est définie par la classe "BytesArray". Cette classe, sous-classe de "FixedSizeCollection" contient les méthodes qui permettent de gérer la matrice (ajouter, modifier supprimer des symboles), de limiter la recherche..."BytesArray" a deux variables d'instance:

  - "nbytes": contient le nombre d'octets nécessaires au codage d'un objet dans la matrice binaire,

  - "matrice": une "ByteArray" organisée sous forme de matrice, elle contient les informations nécessaires à la recherche binaire.

La recherche d'un objet commence dans la matrice binaire. Dans cette matrice, chaque objet est défini par un certain nombre de données. Ces données font partie de l'alphabet de la base de données. Dans notre cas, les données appelées constituants de l'alphabet, sont les symboles élémentaires du langage Bliss. Pour chaque objet, la présence ou l'absence d'un constituant est matérialisée par un "un" ou un "zéro".

Dans la matrice chaque objet est répertorié par un numéro (indice) et classé dans le même ordre que les données du fichier séquentiel indexé (et donc de la table d'index du fichier).

L'utilisateur choisit dans un menu une des composantes Bliss élémentaires qui fait partie du symbole qu'il recherche. Le système questionne la matrice binaire pour obtenir toutes les données qui possèdent ce symbole élémentaire et crée une matrice réduite composée de l'ensemble de ces données. L'opérateur choisit un second symbole élémentaire, le système réduit à nouveau la matrice à l'ensemble des données qui contiennent ce symbole. Et ainsi de suite jusqu'à ce que le nombre de données possible soit inférieur à selectMax (16 par exemple).

Le système recherche alors dans la table d'index, grâce au numéro d'indice de chaque donnée sélectionnée, la position absolue de chaque objet dans le fichier séquentiel puis renvoie ces données dans une liste pour sélection finale.

III.8.4. Conclusions

L'implantation d'une matrice binaire au sein d'un système de gestion de base de données présente de nombreux avantages.

Au niveau communicateur, elle permet d'une part d'intégrer une méthode d'accélération efficace, et d'autre part d'accéder, de manière non ordonnée, aux informations contenues dans un dictionnaire.

Au niveau machine, la matrice binaire utilise peu d'espace mémoire. Elle est donc implantée en mémoire centrale. Le temps d'accès à la matrice est donc court. Remarquons que l'on pourrait augmenter ses performances en écrivant les méthodes de gestion de la matrice en langage assembleur.

Au niveau pédagogique, dans le cas d'un communicateur Bliss, l'utilisation de symboles élémentaires comme éléments informatifs de base est intéressante. L'opérateur construit des symboles complexes à partir de composantes élémentaires. Chacun des symboles a une signification idéographique. L'association d'idées simples permet la compréhension, par l'opérateur, d'idées complexes.

Nous avons mis au point un outil qui doit maintenant être évalué. Au niveau des communicateurs Bliss, une étude plus approfondie du choix des composantes élémentaires serait digne d'intérêt. Rappelons que notre but est d'accéder aux symboles Bliss du dictionnaire le plus rapidement possible en choisissant le plus petit nombre de composantes. Du choix de ses composantes découle directement les performances de la base de données.

III.9. Les tables de données.

Nous avons vu que dans les bases de données, les mots Bliss sont tous stockés en mémoire de masse. Il faut donc accéder à cette mémoire chaque fois que l'on veut manipuler une de ses données. La rapidité du système s'en trouve ainsi gravement compromise.

Il est, par conséquent, nécessaire de stocker les pavés non pas dans une structure de type Base de Données mais dans une table de données installée en permanence en mémoire secondaire.

III.10.1. Structure d'une table de données.

Les tables de données sont des instances de la classe "Array" (classe appartenant à Smalltalk). Cette classe contient toutes les méthodes qui permettent de gérer les tables de données:

- recherche d'une donnée,
- mise à jour de la table de données (addition, suppression, modification...),
- mémorisation des données en mémoire de masse.

La table de données est structurée comme une liste de données (objets de la classe "Pavés" dans CACHALOT), installée en mémoire secondaire et donc directement accessible par le système.

III.10.2. Table de données en mémoire de masse.

Les tables de données sont mémorisées dans un fichier séquentiel ("FileStream"). Les données (pavés) sont stockées séquentiellement dans le fichier ce qui permet un taux d'occupation mémoire très élevé (proche de 100%).

Les méthodes d'accès aux données sont limitées à la lecture séquentielle de tous les pavés stockés et à leur regroupement dans une liste mémorisée dans la variable de classe "Pavés".

III.10.3. Conclusion.

La table de données est intégrée dans Smalltalk au premier appel d'une de ses données. Ceci limite le nombre d'accès au disque et accélère ainsi le fonctionnement du logiciel à l'appel d'un objet (pavé).

Ces listes peuvent, cependant, occuper une place importante en mémoire et ralentir le fonctionnement de Smalltalk au niveau de la gestion de sa mémoire virtuelle.

III.11. Conclusion générale.

Le concepteur peut choisir entre une base de données et une table de données. Son choix doit être dicté par:

- le nombre d'objets associés au dictionnaire,
- le taux d'utilisation des objets dans le communicateur,
- le compromis espace mémoire utilisé/vitesse du logiciel.

On peut, dans le cas du dictionnaire Bliss, décomposer le problème de la façon suivante:

- mémorisation des pavés qui représentent les composantes Bliss dans une table de données. Ces pavés sont utilisés dans des menus pour reconstituer un pavé Bliss codé. Ils sont souvent utilisés dans les communicateurs pour retrouver les symboles Bliss. Pour accélérer la reconstitution d'un symbole Bliss, ces pavés sont codés sous la forme d'un "Bitmap ("Form")".
- mémorisation des pavés Bliss qui représentent les symboles Bliss et qui constituent le dictionnaire Bliss dans la base de données. Ce dictionnaire peut contenir jusqu'à 5000 symboles. Ces pavés sont, en moyenne, utilisés de manière peu fréquente, en général lors de la sélection finale d'un symbole Bliss.

CONCLUSION GENERALE

Le projet CACHALOT répond parfaitement aux exigences d'un générateur de communicateurs pour handicapés. La réalisation de prototypes est aisée, l'évolution du générateur possible.

Afin d'augmenter encore les performances de notre système, il serait intéressant d'y ajouter des modules liés à l'intelligence artificielle.

Ces modules pourraient intervenir dans différentes parties du générateur, ainsi:

- afin d'évaluer les performances de l'ensemble utilisateur-machine, il serait intéressant d'ajouter au système une base de données individuelle des messages produits (BDIMP), implantée en mémoire de masse, mémorisant toutes les actions sur l'organe d'accès, le temps et les éléments sélectionnés (figure C.1).

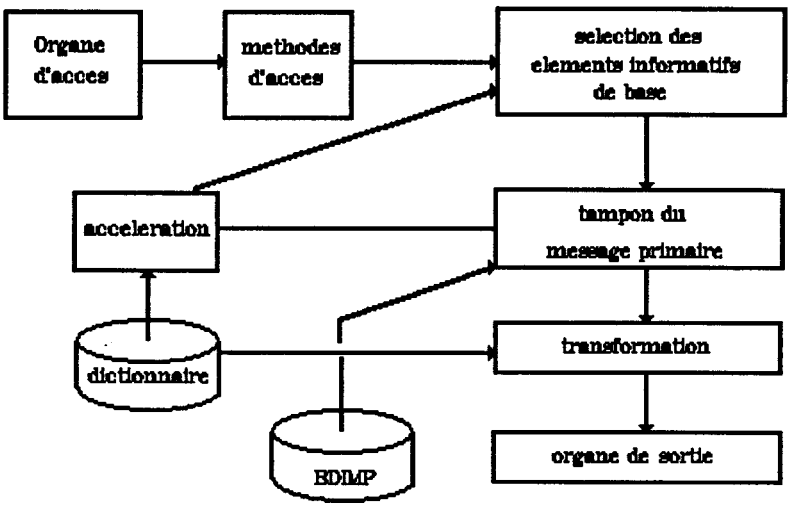


Figure C.1

La figure C.2 présente la structure d'un système de génération automatique contenant deux nouvelles bases de données.

- La première, la "base de données orientée objet des composantes des communicateurs (BDOOCC)" contiendra l'ensemble des connaissances liées à la communication:

  - les éléments informatifs de base,
  - les configurations des menus,
  - les méthodes d'accélération,
  - l'organisation et le contenu des dictionnaires,
  - les organes et méthodes d'accès,
  - les méthodes de transformation.

Cette base de donnée permettra la génération automatique du logiciel et la description de la partie matérielle du communicateur généré.

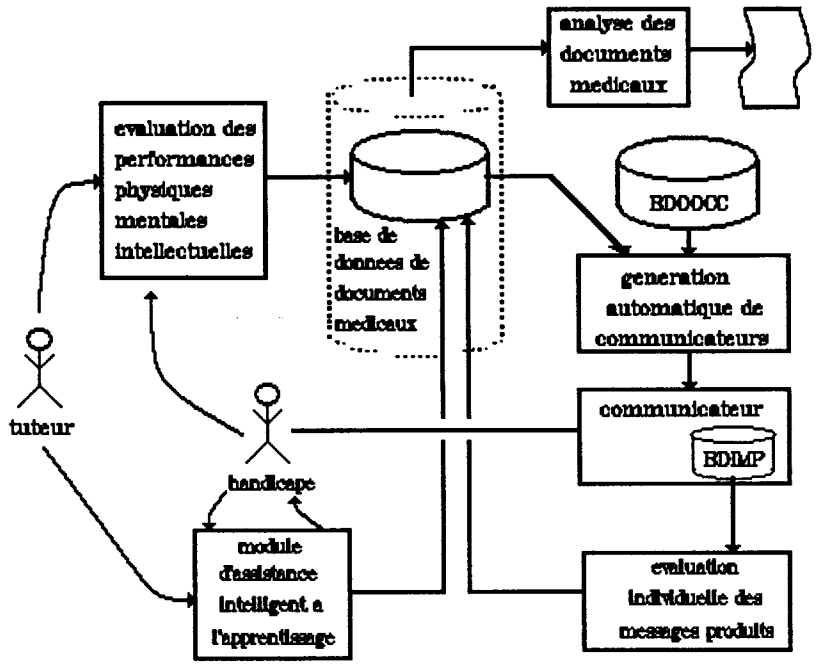


figure C.2

Afin de permettre une adaptation aux handicaps spécifiques, le système utilisera les informations contenues dans la deuxième base de données: la "base de données de documents médicaux (BDDM)". Elle contiendra tous les éléments qui décrivent les performances physiques, mentales et intellectuelles de l'handicapé au niveau de la communication. Ses informations sont obtenues grâce à un évaluateur spécifique et sont complétées par les résultats obtenus par un système d'évaluation des messages produits.

On pourra ajouter des procédures qui permettront, par exemple, l'analyse de documents médicaux ou d'aide à l'apprentissage.

L'apport de l'Intelligence Artificielle pourra se faire à différents niveaux:

- au niveau du communicateur, un système expert accélérera la production de message en tenant compte du contexte, du lieu, du jour, de l'heure...

- on intégrera la transformation sémantique ou pragmatique dans un système I.A.

- au niveau de la génération, un système expert permettra le choix de procédures adaptées aux informations médicales, optimisera l'espace mémoire. La technique de compression des données sera ainsi choisie en fonction de leur fréquence d'utilisation.

ANNEXE 1

AN.1. INTRODUCTION

Nous nous proposons dans cette annexe de présenter la conception générale d'un système de génération automatique de logiciels de communication pour handicapés.

Le système génère des communicateurs basés sur les concepts qui ont été décrits dans les chapitres précédents.

Le générateur est conçu suivant le concept de la programmation orientée objet. Générer, dans ce cas, consiste à créer des objets spécifiques au communicateur et à les placer dans une base de donnée.

Le communicateur est un logiciel qui gère la communication entre objets. Ce logiciel est tout à fait général. Seule la base de données est spécifique, elle contient l'ensemble des objets qui caractérisent un communicateur.

La mise en route est automatique. Un programme "batch" permet à l'utilisateur d'appeler soit le logiciel générateur de communicateurs soit la liste des communicateurs déjà générés (figure AN1.1).

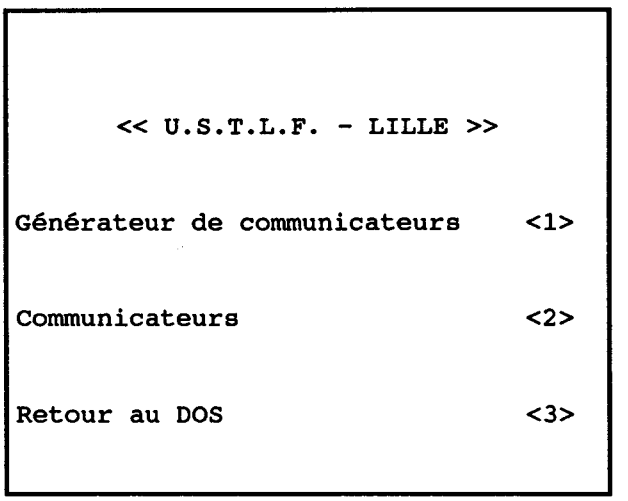


figure AN1.1

Si l'utilisateur a choisi de charger un communicateur, la liste des communicateurs lui est présentée et il peut y faire son choix (figure AN1.2).

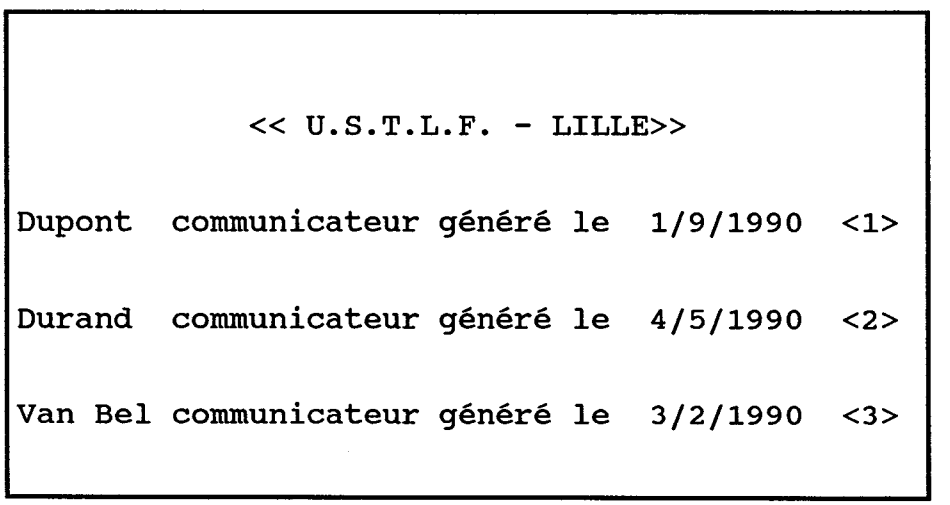


figure AN1.2

Si l'utilisateur veut générer un nouveau communicateur, le logiciel de génération est chargé. Il apparaît sur l'écran vidéo une première image (figure AN1.3).

AN.2. SCÉNARIO DE GÉNÉRATION D'UN COMMUNICATEUR

L'image initiale est composée d'un ensemble de fenêtres aux qu'elles sont associés des menus spécifiques.

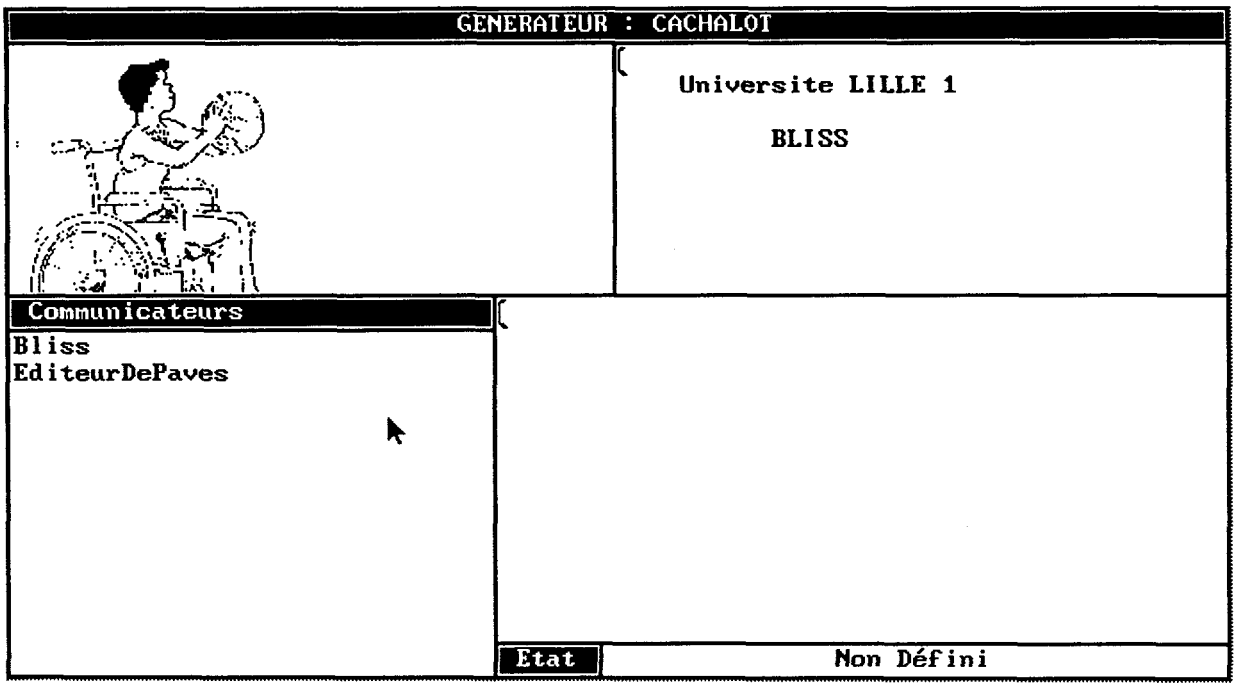


figure AN1.3

AN.2.1. Fenêtre "sigle".

Cette fenêtre contient un sigle propre au générateur. Ce sigle peut être changé. Le menu associé à cette fenêtre est composé d'une série de commandes qui sont:

- "fermer"
  Fait disparaître l'image initiale.

-" recadrer"
  Permet de déplacer l'image initiale dans l'écran.

- "cycle"
  Visualise les autres fenêtres.

- "dictPaves"
  Permet d'accéder à l'éditeur de pavés

- "sauvegarde"
  Sauvegarde sur disque le communicateur généré.

- "sigle"
  Permet de choisir un sigle dans l'ensemble des pavés existants

AN.2.2. Fenêtre "présentation".

Cette fenêtre contient les informations telles que le nom des possesseurs du logiciel de génération.

AN.2.3. Fenêtre "communicateurs" (figure AN.4).

La fenêtre communicateurs présente la liste des communicateurs déjà générés. A cette fenêtre est associé un menu qui permet:

- la sauvegarde sur disque d'un communicateur: "sauvegarde",
- d'accéder à l'éditeur de pavés: " dictpaves",
- de modifier un communicateur existant: " modification"
- d'accéder à l'éditeur de mots: " dictmots",
- de créer un nouveau communicateur: " création",
- de copier un communicateur existant: " copier",
- de supprimer un communicateur: " supprimer",
- d'exécuter un des communicateurs déjà généré: "exécuter".

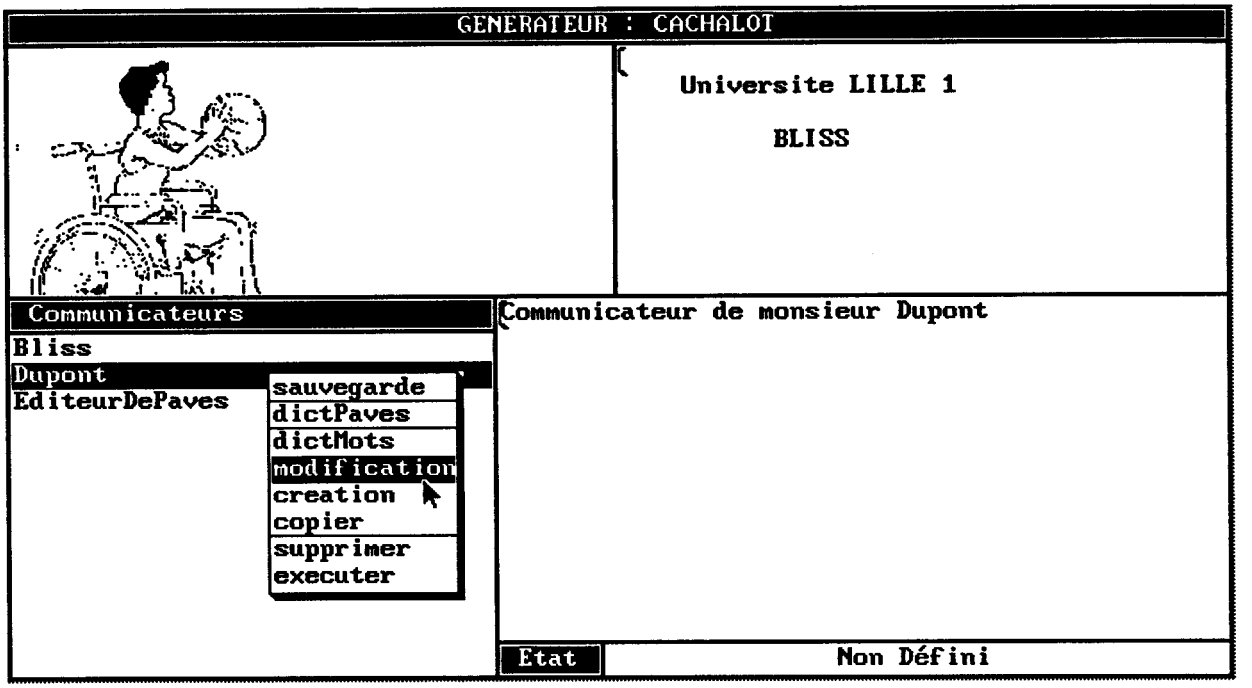


figure AN1.4

AN.2.4. Fenêtre "libellé"

On retrouve dans cette fenêtre le libellé des informations qui concernent le générateur sélectionné dans la fenêtre "Communicateurs" telles que le nom du communicateur et sa date de génération ou de modification. A cette fenêtre est associé un menu qui contient des commandes d'édition de texte.

AN.2.5. Etat du communicateur

Les communicateurs déjà générés sont renseignés comme étant dans un état défini, le communicateur en génération est dans un état indéfini.

AN.3. Génération d'un communicateur

Supposons que nous devions générer un communicateur. Nous choisissons dans le menu de la fenêtre Communicateurs l'option création. Le système nous demande de donner un nom au communicateur à générer. Nous choisissons par exemple de

générer le communicateur de monsieur Dupont. Le communicateur "Dupont" est ajouté à la liste des communicateurs qui existent déjà.

Le communicateur "Dupont" est initialement vide. Nous allons le générer en le modifiant. Nous choisissons dans le menu associé à la fenêtre Communicateur l'option "modification" (figure AN1.4).

Une nouvelle fenêtre est ouverte. Elle se subdivise en deux grandes parties (figure AN1.5).

La fenêtre "Communicateur: DUPONT" et la fenêtre environnement.

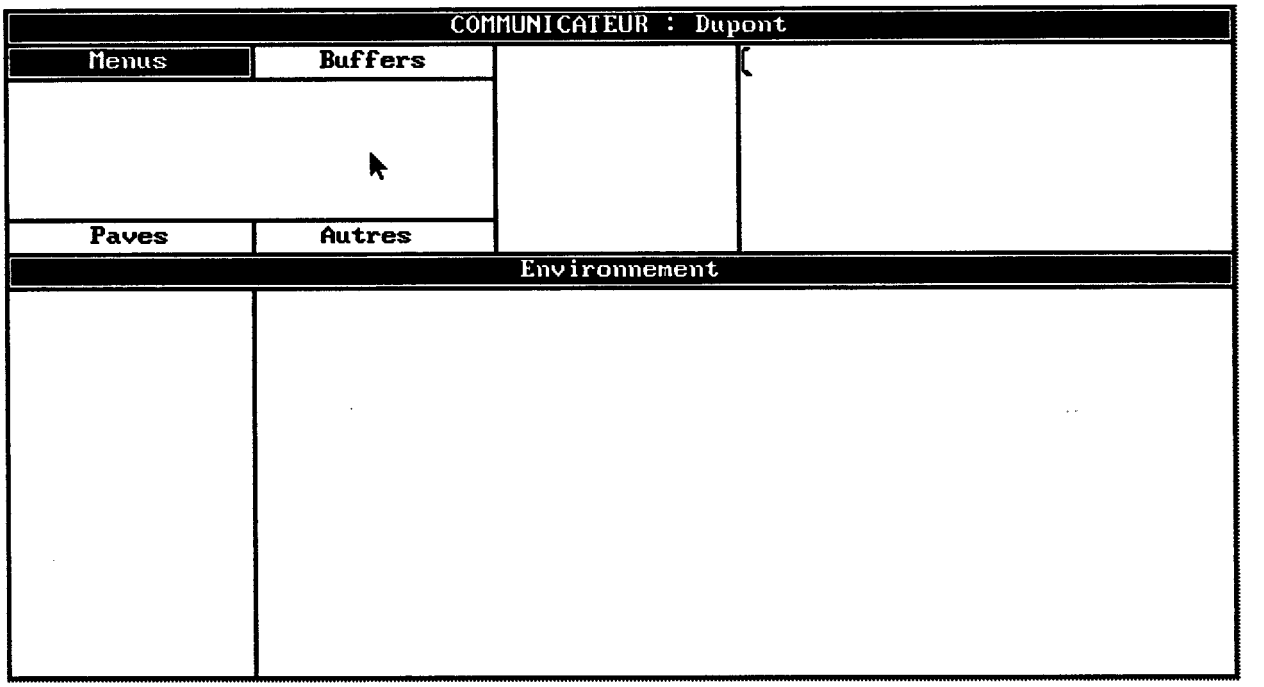


figure AN1.5

AN.3.1. Fenêtre "Communicateur"

La fenêtre "Communicateur" est elle-même subdivisée en trois fenêtres. La première sous-fenêtre permet la génération du communicateur. Elle permet en fonction de quatre options: "menus", "buffers", "pavés" et "autres" de créer des nouveaux menus, de leur associer des pavés et de générer les zones de visualisation des éléments informatifs sélectionnés (ce sont les mémoires tampons de message ou "buffers").

Les deux autres fenêtres sont des fenêtres de contrôle. Elles permettent au concepteur d'avoir continuellement sous les yeux les différents paramètres de l'élément du communicateur qui est en élaboration.

Le menu général associé à la première fenêtre contient les commandes "sauvegarde", "exécuter", "dictPaves" et "environnement" (figure AN1.6).

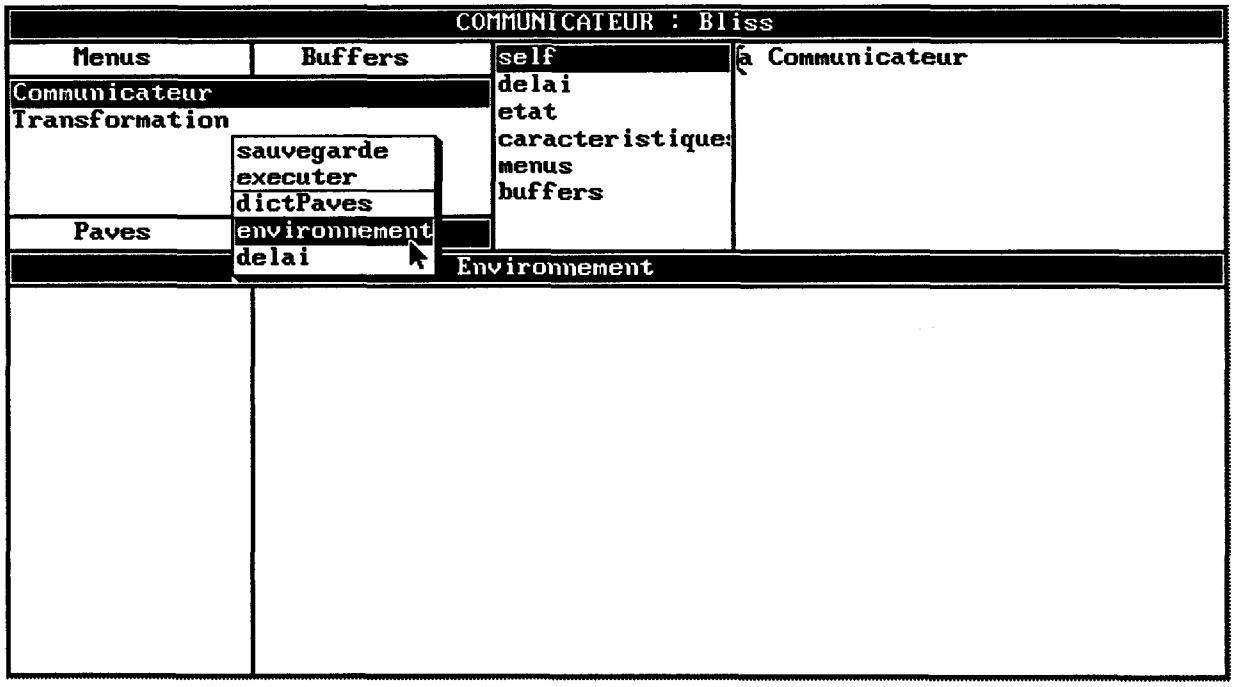


figure AN1.6

a) "sauvegarde" permet de sauver sur disque le nouveau communicateur.

b) "exécuter" lance à partir du générateur le communicateur généré et permet ainsi de le tester immédiatement. La commande est expliquée en AN.3.5.

c) "dictPaves" accède au dictionnaire de Pavés. La commande est expliquée en AN.4.

d) "environnement" permet de choisir le nombre, la disposition et les dimensions des différents menus et tampons de message ("buffers") qui constituent l'image vidéo du communicateur présenté à l'handicapé utilisateur. Cette commande est utilisée après avoir défini les menus et mémoires tampons du communicateur. La commande est expliquée en AN.3.3.

AN.3.1.1. Création d'un menu

Les éléments informatifs de base (pavés) sont logiquement organisés en menus et sous-menus. Un menu (ou sous-menu) représente une page visualisable de pavés dans une fenêtre d'affichage. Tous les menus ont la même structure. Ils sont définis dans la classe "Menus" Cette classe définit d'une part la structure interne de menu, d'autre part les opérations qui peuvent être effectuées sur les menus. Chaque menu est une instance de type Menu.

Les variables de la classe Menus sont: pavés, délai, désignation, sélection, type.

L'option "Menus" donne accès à un menu qui facilite la génération de menus spécifiques. Ce sont:

a) ajout (figure AN1.7) et (figure AN1.8).

Cette fonction permet d'ajouter un menu au communicateur. Une fenêtre question appelée "prompter" s'ouvre. L'opérateur frappe le nom du menu à générer, le menu Bliss par exemple. Ce menu, pour l'instant vide, est ajouté à la liste des menus existants.

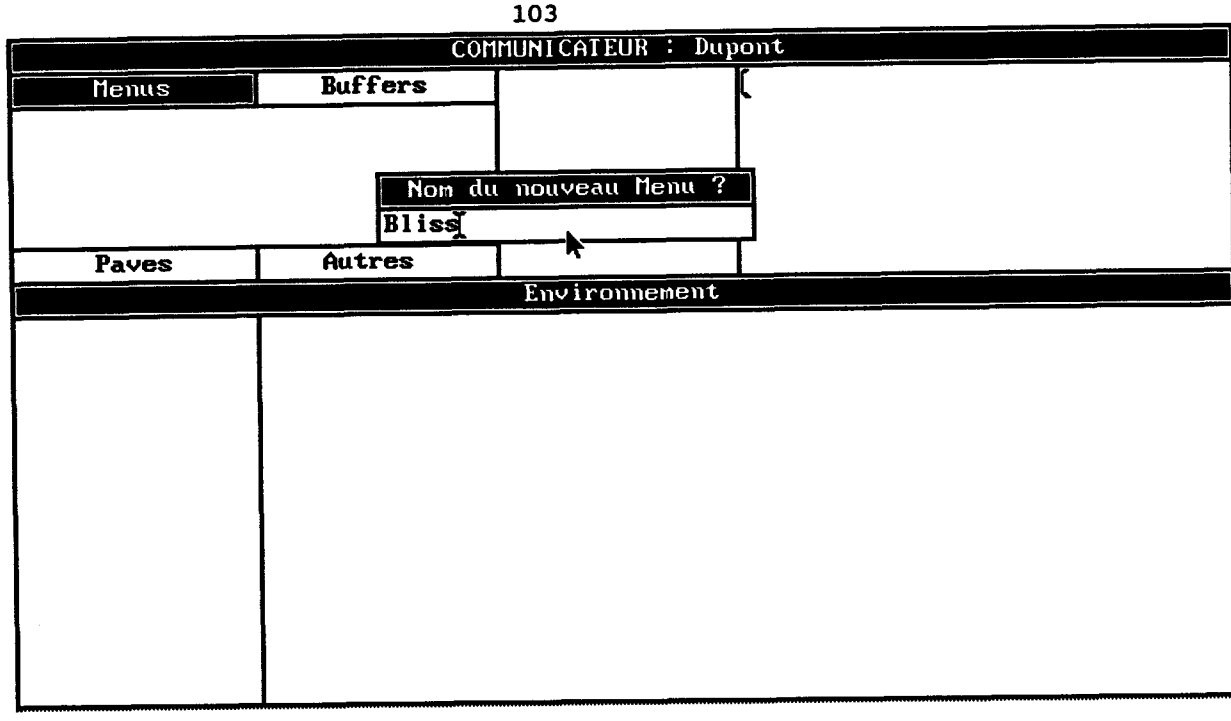


figure AN1.7

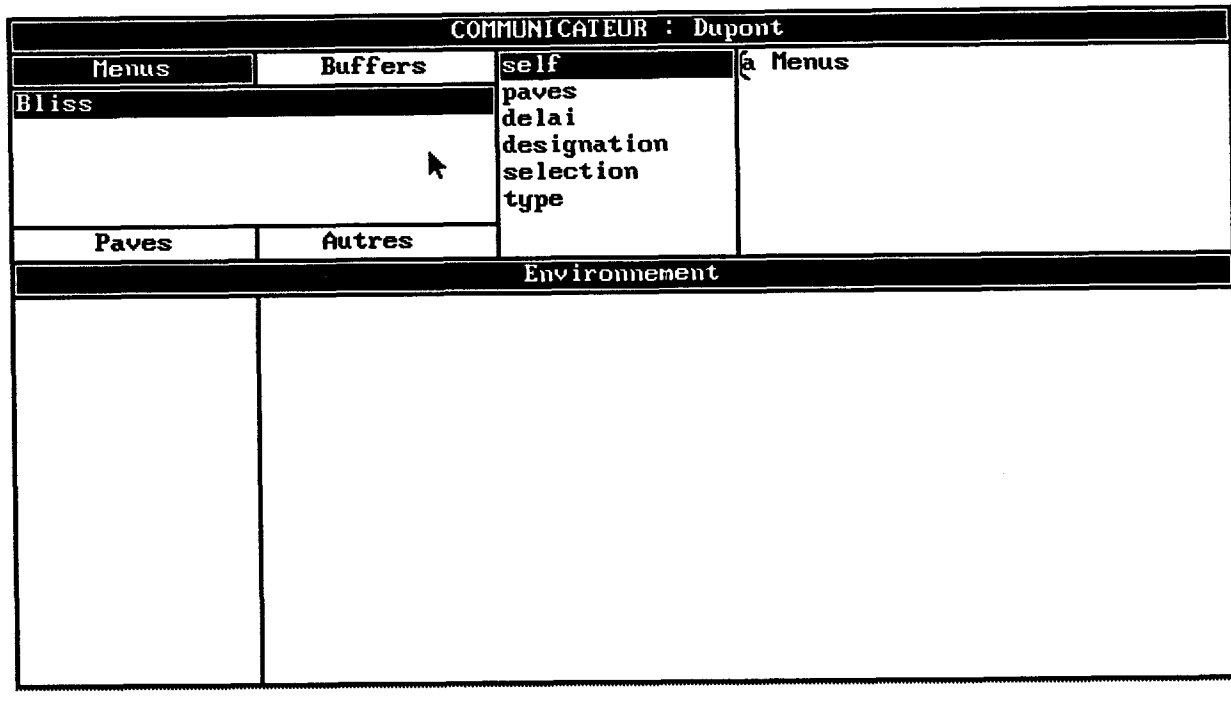


figure AN1.8

b) suppression

Cette fonction permet de retirer un menu de la liste des menus du communicateur.

c) balayage (figure AN.9)

Cette commande permet de choisir un des neuf types de balayage suivant:

-"balCurseur" ou balayage par curseur

-"balSequentiel" ou balayage séquentiel

-"balSeqTriangle" ou balayage séquentiel triangulaire

-"balLigCol" ou balayage ligne/colonne et "balColLig" ou balayage colonne/ligne

-"balDichoLC" ou balayage dichotomique ligne/colonne et "balDichoCL" ou balayage dichotomique colonne/ligne

-"balMixteLC" ou balayage mixte ligne/colonne et "balMixteCL" ou balayage mixte colonne/ligne

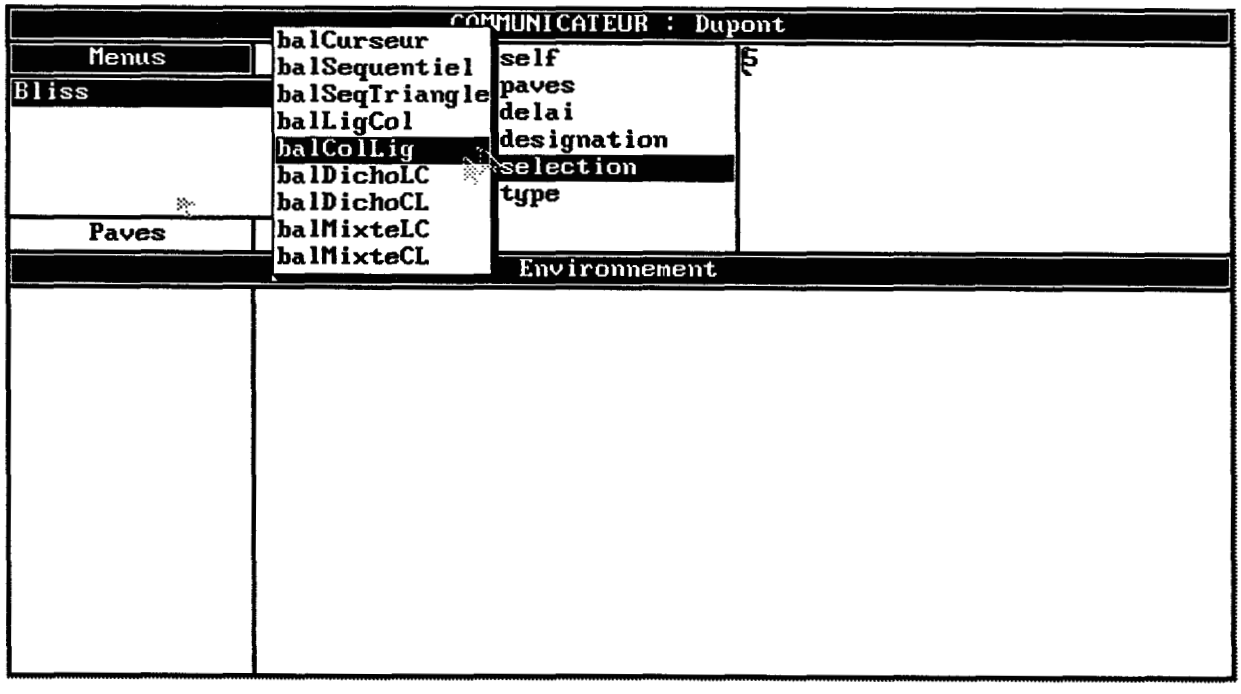


figure AN1.9

d) délai

Le choix de délai provoque l'ouverture d'un "prompter". On indique en milliseconde la valeur du délai de balayage (figure AN1.10).

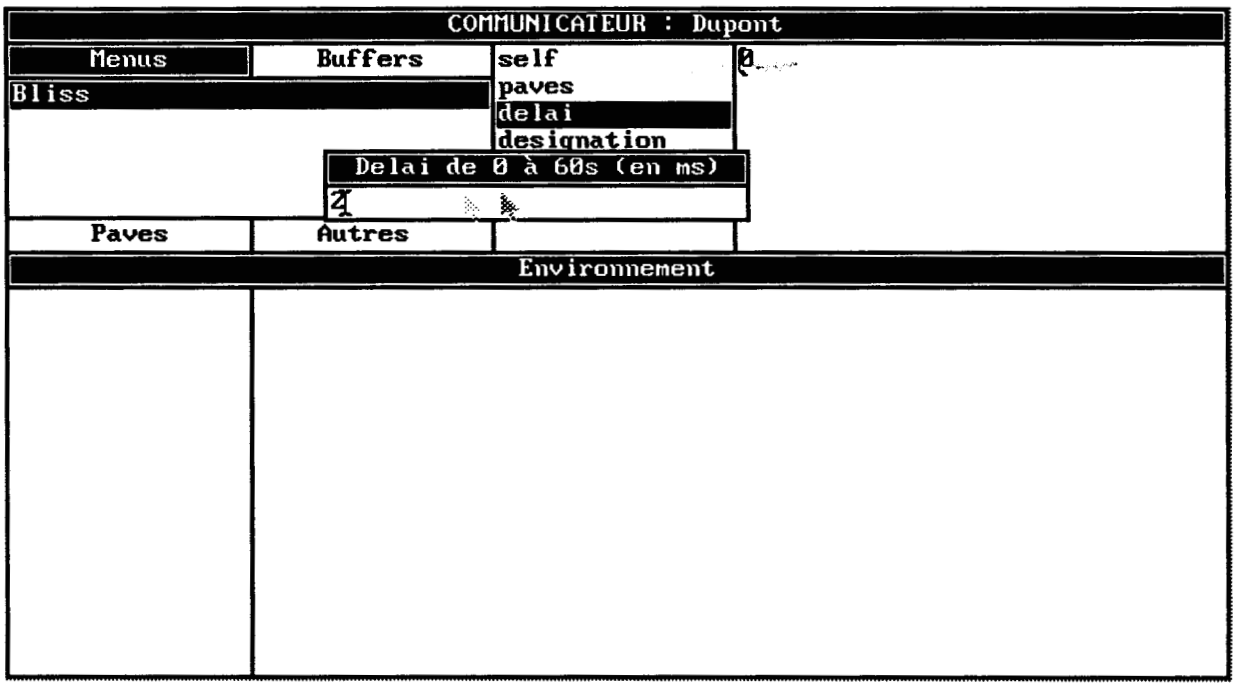


figure AN1.10

e) type

Trois présentations des pavés dans le menu sont prévues, ce sont les dispositions "triangulaire", "rectangulaire" et "menu" (figure AN1.11).

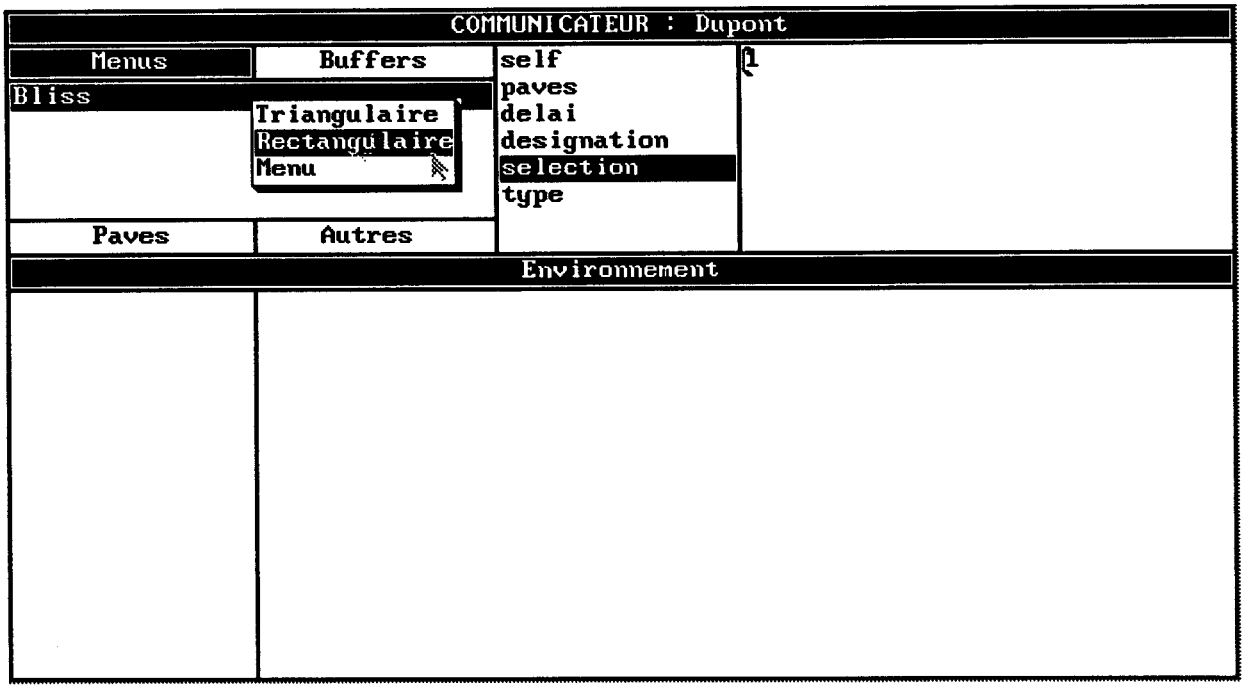


figure AN1.11

AN.3.1.2. Ajout de pavés dans un menu

Le menu étant défini, il faut maintenant lui associer ses pavés. Activons la commande "Paves", elle nous permet d'accéder à une série de commande. Ce sont: "dictPaves", "fonction", "codage", "ajout" et "suppression".

a) "DictPavés": cette commande est expliquée en AN.3.4.

b) "Fonction":

c) "Codage": indique sous quelle forme le pavé est mémorisé. Il est mémorisé soit sous la forme d'une chaîne de caractères ("String"), sous la forme d'un "bitmap" ou soit sous la forme d'un pavé codé.

d) "Ajout": ajoute un pavé au menu. Une fenêtre question est ouverte et demande le nom du pavé à ajouter.

e) "Suppression": supprime le pavé sélectionné de la liste des pavés du menu.

AN.3.1.3. Création de "buffers"

L'option "buffers" (mémoire tampon de message) permet à l'aide d'un menu d'initialiser la fenêtre du communicateur qui est associée au menu qui vient d'être défini. Ce "buffer" contiendra les éléments informatifs sélectionnés (pavés).

a) type

La commande "type" permet de définir les "buffers" qui sont, dans l'environnement Smalltalk, des "panes". Ces "panes" peuvent être soient:

- des "EditeurPanes" si les éléments informatifs ont été construits à l'aide de l'éditeur de pavés.
- des "GraphPanes"
- des "TextPanes" si les éléments informatifs sont composés de caractères alpha-numériques.

c) ajout

La commande "ajout" permet de lier au menu qui vient d'être défini un certain nombre de "buffers"

d) suppression

Cette commande supprime un "buffer" préalablement défini

AN.3.1.4. L'option "autres"

L'action sur cette commande initialise les commandes du générateur.

AN.3.2. Fenêtre "Environnement"

La fenêtre "environnement" se décompose en deux parties. D'une part une fenêtre liste qui contient la clé de chacun des pavés du menu et d'autre part une fenêtre qui présente l'aspect du menu en élaboration (figure AN1.12).

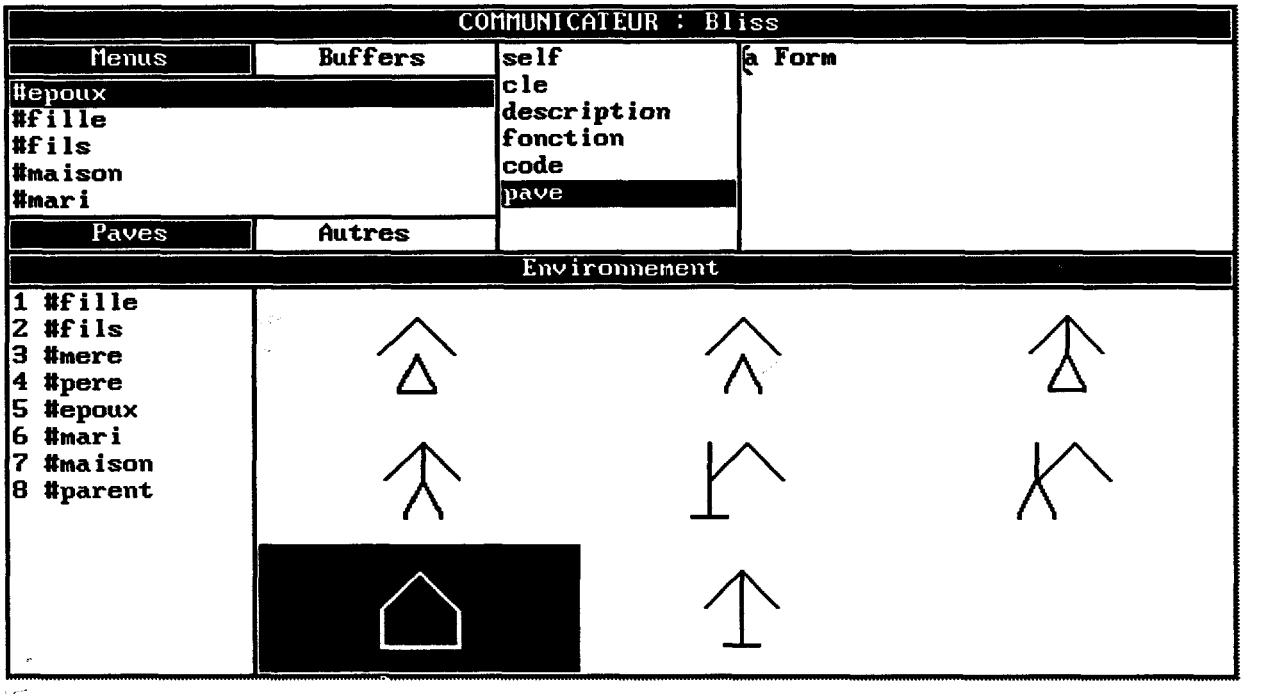


figure AN1.12

AN.3.2.1. La fenêtre "liste"

Cette fenêtre présente la liste des pavés, ils sont numérotés et sont définis par leur clé. L'ordre des pavés dans la liste correspond à l'ordre dans lequel ils ont été introduits dans le menu.

Il est possible de modifier cet ordre. Il suffit de sélectionner à l'aide de la souris le pavé à déplacer et d'ensuite indiquer dans la fenêtre menu l'emplacement désiré. La modification est instantanée et se fait sous les yeux du concepteur.

AN.3.2.2. La fenêtre "menu"

La fenêtre "menu" visualise le menu en cours d'élaboration. Les pavés sont disposés automatiquement dans le menu. Cette disposition tient compte du nombre de pavés contenus dans le menu ainsi que de leurs dimensions.

Cette fenêtre permet au concepteur de tester le menu en élaboration. Elle visualise l'aspect du menu: son type, son mode et sa vitesse de balayage. Le concepteur peut à tout moment modifier les paramètres du menu et visualiser immédiatement l'effet des modifications.

AN.3.3. La commande "environnement"

La commande "environnement" permet de choisir le nombre, la disposition et les dimensions des différents menus et tampons de message ("buffers") qui constituent l'image vidéo du communicateur présenté à l'handicapé utilisateur (figure AN1.13).

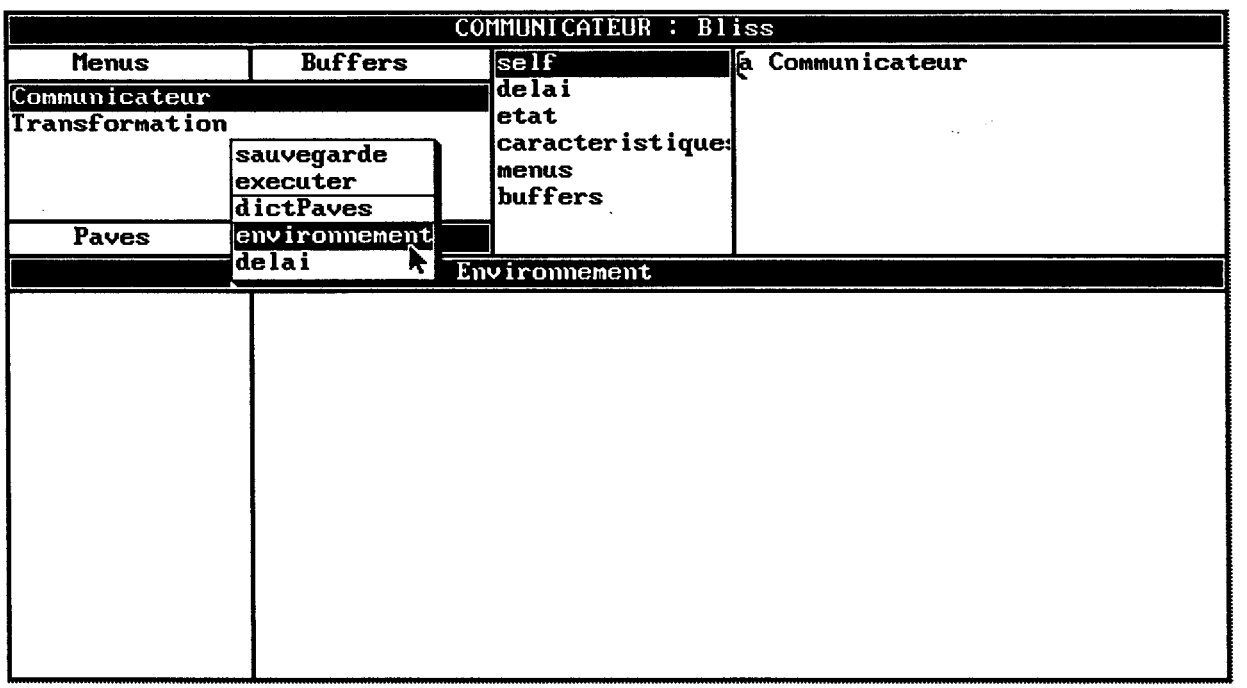


figure AN1.13

Cette commande agit sur la fenêtre "menu" de la fenêtre "environnement". Une fenêtre question ("prompter") demande le nombre de menus qui sont affichés simultanément sur l'écran vidéo (figure AN1.14).

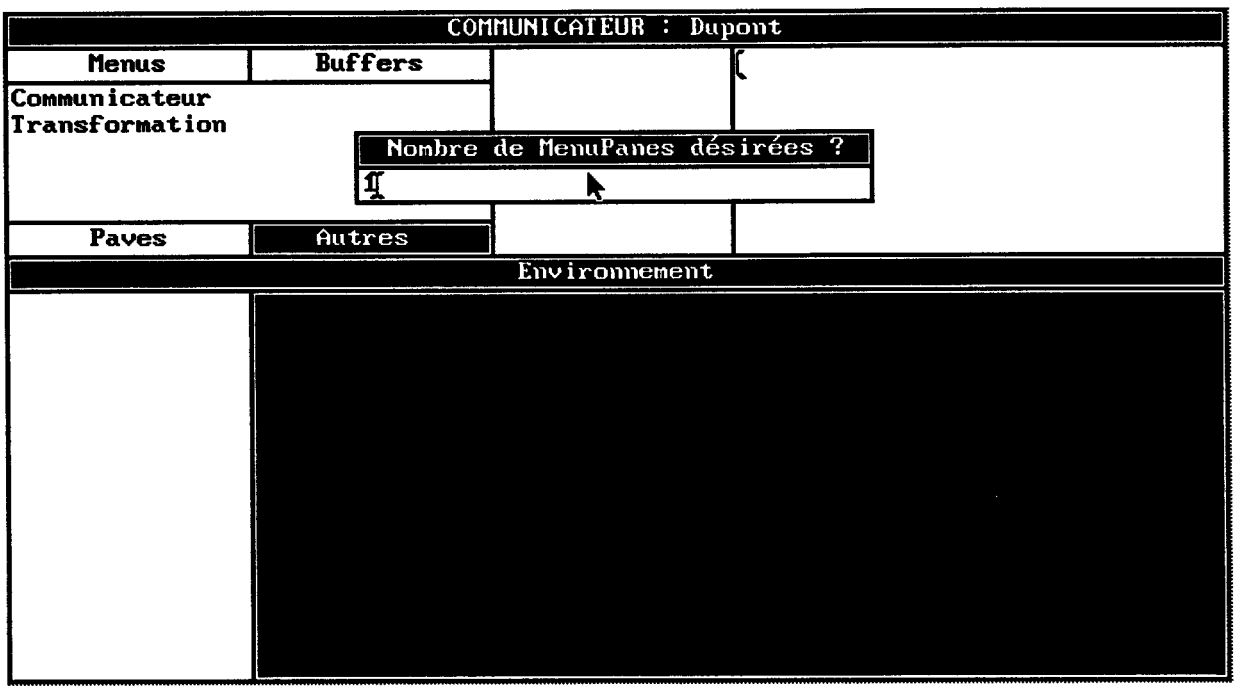


figure AN1.14

La fenêtre "liste" contient alors le nom des menus définis dans le communicateur ainsi que le nom des tampons de message qui lui sont associés. En déplaçant la souris dans la fenêtre "menu" on choisi la position et les dimensions des menus et tampons du communicateur (figure AN1.15).

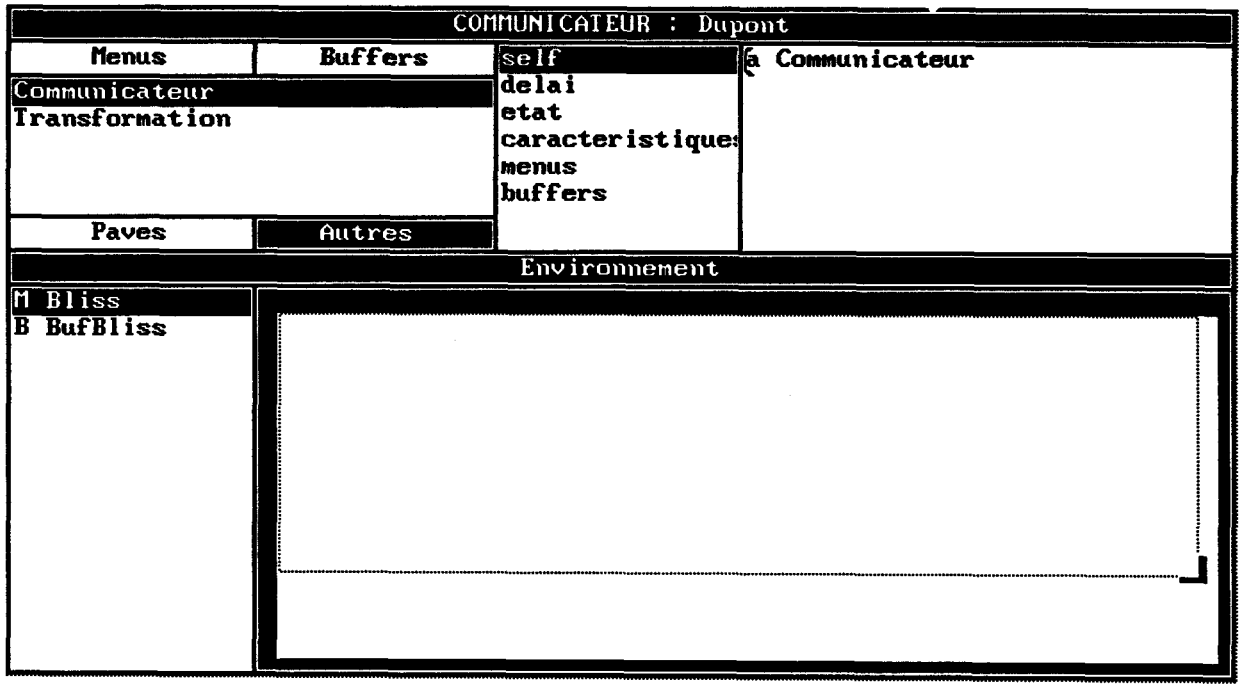


figure AN1.15

AN.3.4. La commande "dictPavés"

"DictPavés" donne accès à un dictionnaire de pavés.

Cette commande ouvre une fenêtre appelée "générateur de pavés" (figure AN1.16). Cette fenêtre se décompose en un ensemble de sous-fenêtres.

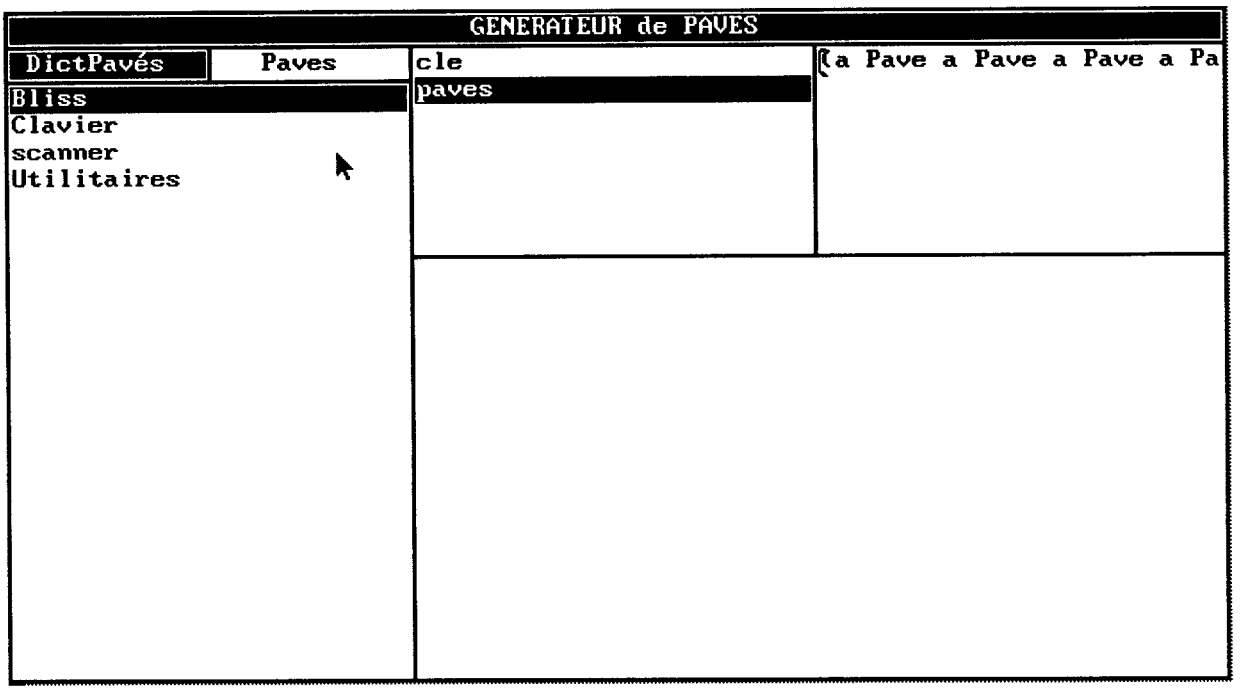


figure AN1.16

a) Sous-fenêtre liste

La fenêtre de gauche est une fenêtre liste. Elle contient la liste des dictionnaires (figure AN1.18). Les commandes qui lui sont associées sont "sauvegarde", "suppression", "créer" et "fusion".

b) Sous fenêtre pavé

Cette fenêtre se divise en trois. Deux des fenêtres sont utilisées pour "inspecter" les caractéristiques du pavé où elles peuvent être modifiées. La dernière fenêtre permet de visualiser le pavé sélectionné.

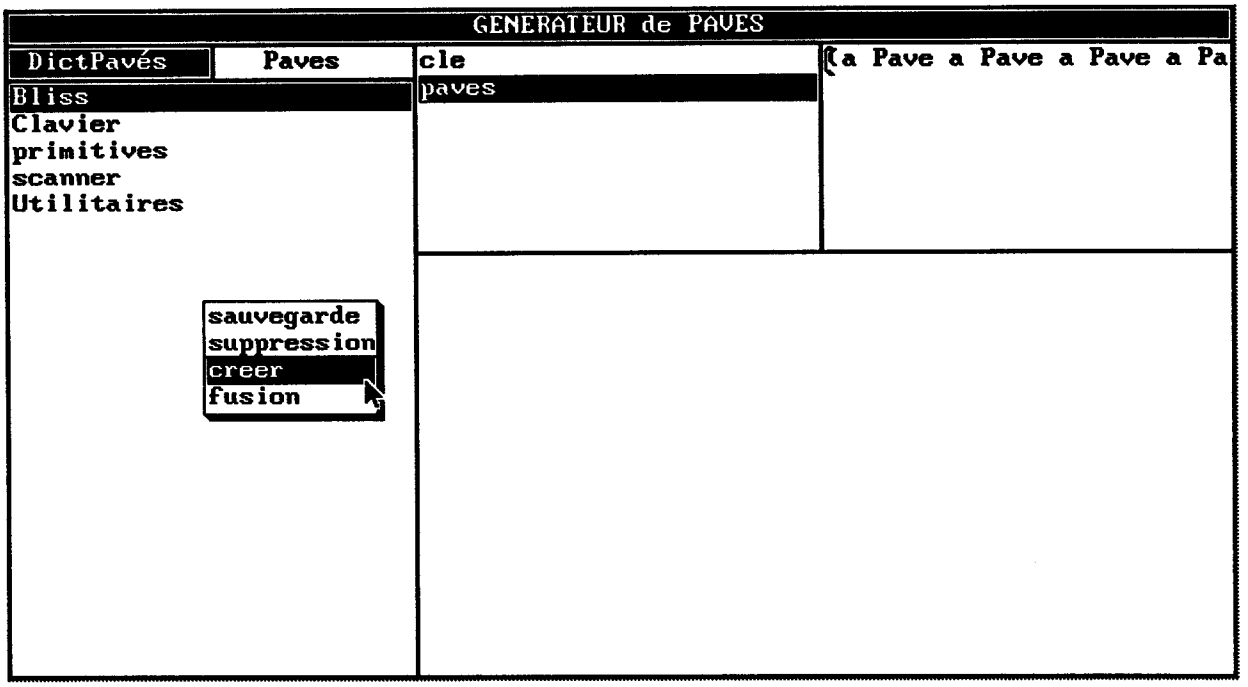


figure AN1.18

"sauvegarde" mémorise sur disque dans le répertoire "Paves" le dictionnaire sélectionné.

"suppression" supprime le dictionnaire sélectionné de la liste.

"créer" ajoute un dictionnaire à la liste. Une fenêtre question ("prompter") permet de donner un nom au nouveau dictionnaire. Ce dictionnaire est initialement vide.

"fusion" fusionne un nombre quelconque de dictionnaires. Une fenêtre question demande le nom du nouveau dictionnaire puis les noms des dictionnaires à fusionner.

Après avoir choisi un dictionnaire, en sélectionnant avec la souris l'option "Paves", on visualise la liste de tous les pavés de celui-ci (figure AN1.19). Les commandes associées ont soit une action sur les pavés, soit une action sur le contenu d'une mémoire de transfert utilisée pour créer les menus.

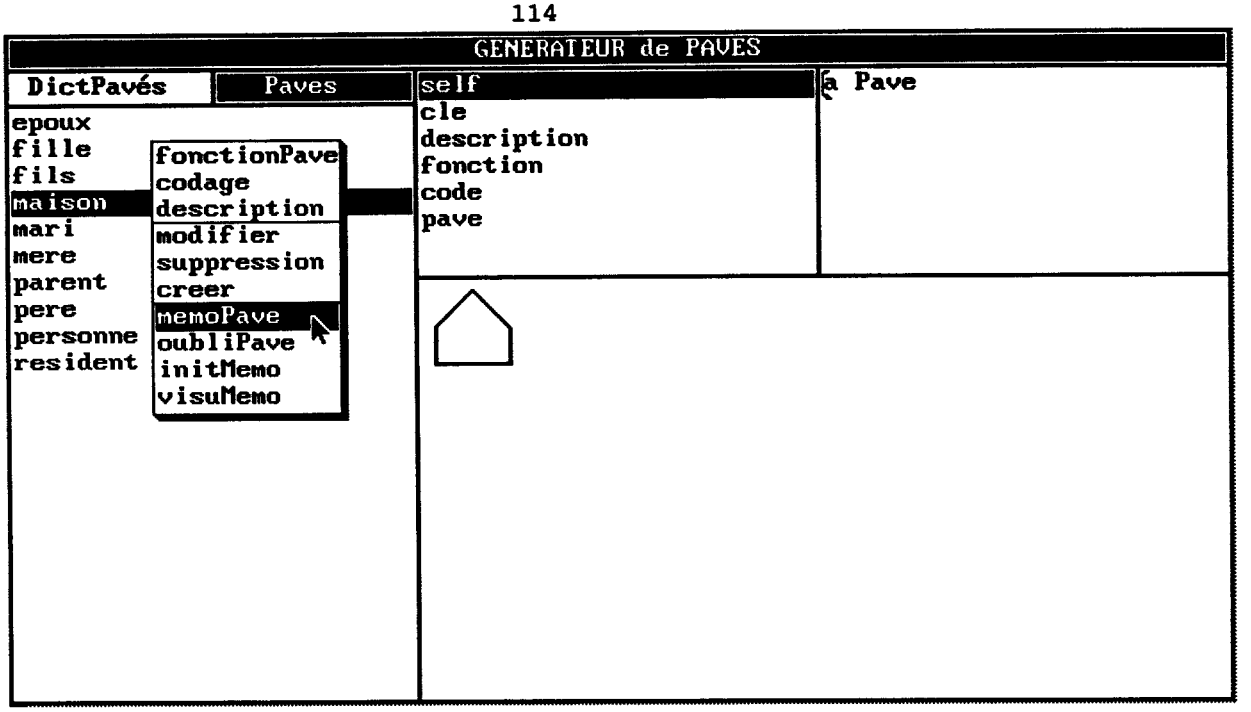


figure AN1.19

- Commandes qui agissent sur les pavés:

  Les commandes "fonctionPave", "codage" et "description" ont déjà été expliquées.

  "modifier" ouvre une fenêtre "Editeur de Pavés". Cet éditeur permet de représenter graphiquement le pavé. Il est expliqué en AN.4.

  "suppression" enlève le pavé du dictionnaire sélectionné. Une fenêtre question demande de confirmer la suppression du pavé.

  "créer" ouvre une fenêtre question qui permet de donner un nom au nouveau pavé et de l' ajouter à la liste des pavés du dictionnaire. Le pavé n'est caractérisé que par son nom. Il faut donc, par la suite le définir en détail.

- Commandes qui agissent sur la mémoire de transfert.

  "memoPave" mémorise le nom du pavé sélectionné dans la mémoire de transfert.

  "oubliPave" enlève le pavé sélectionné de la mémoire de transfert.

"initMemo" initialise le contenu de la mémoire de transfert.

"visuMemo" visualise le contenu de la mémoire de transfert.

AN.3.5. La commande "exécuter".

La commande exécuter permet de faire travailler le logiciel de communication en élaboration exactement de la même manière que le communicateur (figure AN1.20). Ce communicateur peut être tester à n'importe quel moment, son fonctionnement peut être interrompu, l'opérateur peut le modifier, le tester à nouveau jusqu'à ce que il corresponde exactement aux besoins.

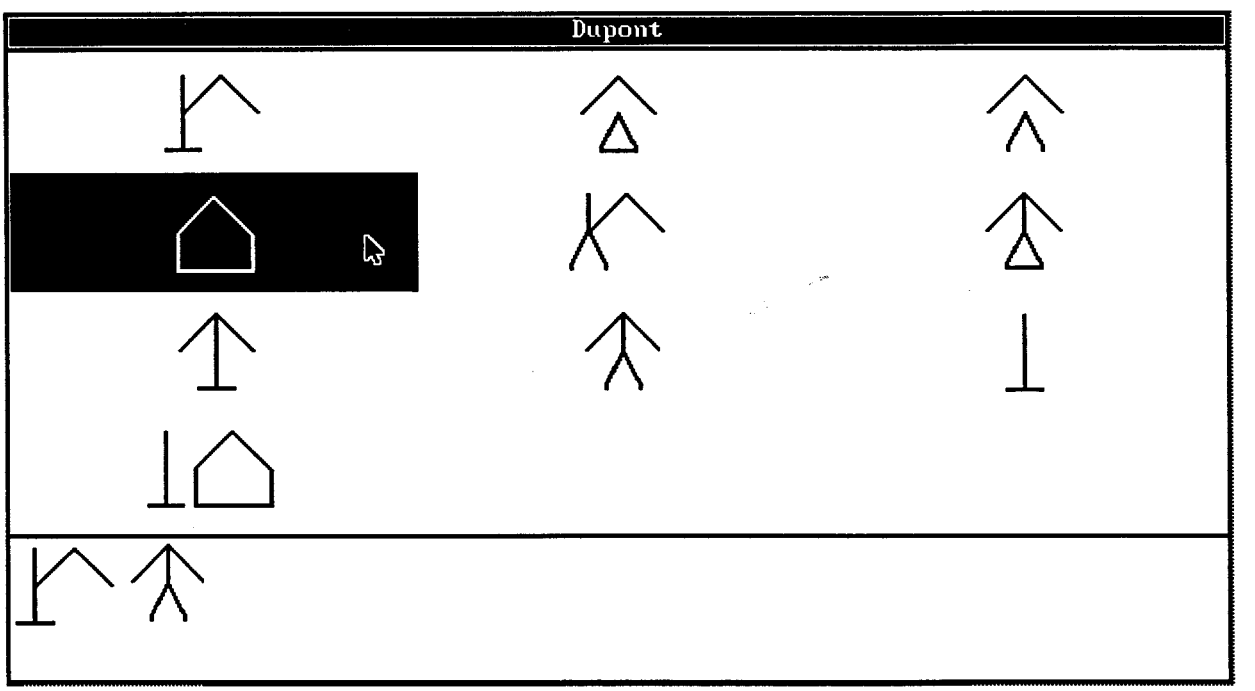


figure AN1.20

AN.4. Editeur de "pavés"

L'éditeur de pavés permet de créer n'importe quel pavé graphique. Cet éditeur permet de:

- visualiser les pavés d'un dictionnaire,
- de les modifier,
- de créer des nouveaux pavés en:
  - faisant appel à un fichier d'images créées à l'aide d'un scanner
  - utilisant les commandes de l'éditeur (figure AN1.21)

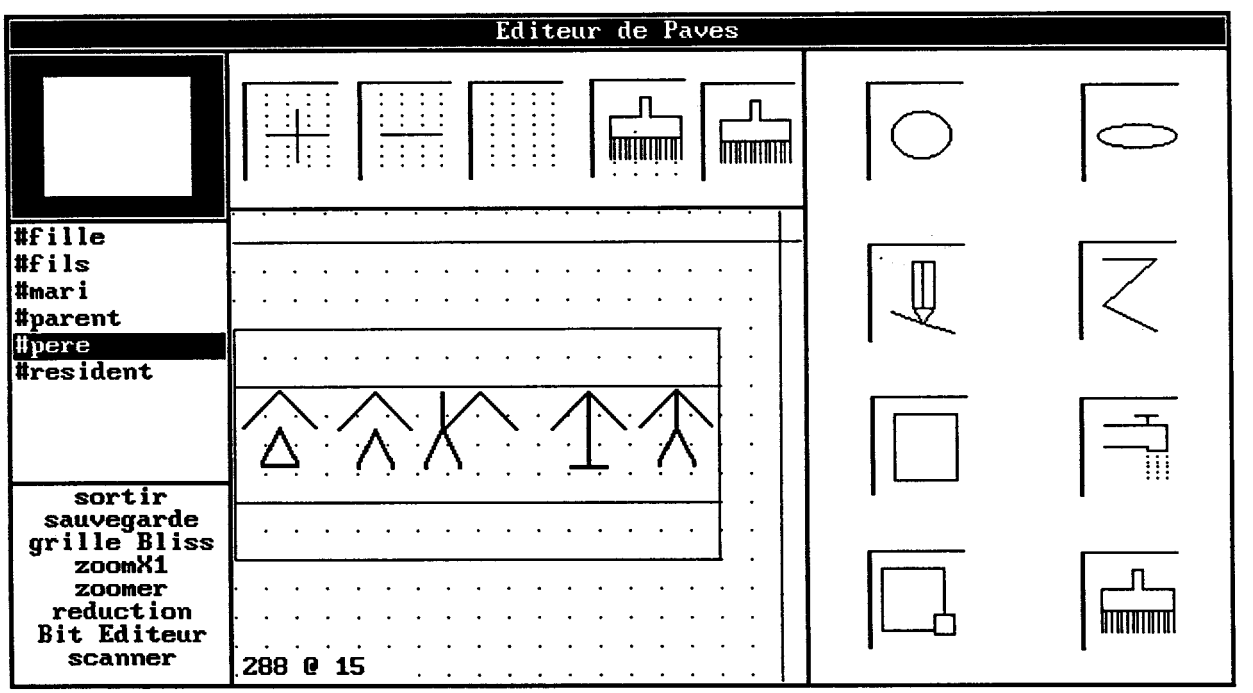


figure AN1.21

La fenêtre présentant l'éditeur de pavés est divisée en six sous-fenêtres: deux fenêtres de visualisation des pavés, une fenêtre liste, deux fenêtres constituées de commande présentées sous forme graphique, une fenêtre de commande présentée sous forme alpha-numérique.

AN.4.1. Fenêtres de visualisation des pavés

AN.4.1.1. Fenêtre de travail.

Les pavés sont construits dans la fenêtre de travail. Le curseur est constitué de deux droites, une verticale et une horizontale. La position du curseur se trouve à l'intersection de ces deux droites. Il se déplace sur une grille dont le pas est variable. Cette grille peut être supprimée. La position du curseur est indiquée, en pixel, dans la partie inférieure gauche de la fenêtre. Une grille Bliss peut remplacer la grille générale. Une commande permet d'effacer le contenu de la fenêtre. Il est possible de choisir une zone de la fenêtre et de lui faire subir un agrandissement. La zone agrandie est visualisée dans la fenêtre.

AN.4.1.2. Fenêtre à dimensions réelles.

Cette fenêtre visualise un dessin en grandeur réelle lorsque le fenêtre de travail présente un dessin agrandi.

AN.4.2. Fenêtres de commande.

AN.4.2.1. Commandes "aspect" de la fenêtre de travail.

Une fenêtre contient les commandes qui permettent de modifier l'aspect de la grille de la fenêtre de travail. Ces commandes permettent de supprimer ou de faire apparaître la grille, de diminuer ou augmenter le pas de la grille, d'effacer le contenu de la fenêtre.

AN.4.2.2. Commandes "dessin".

Une fenêtre contient les commandes d'aide au dessin. Elles permettent de dessiner: à main levée, des cercles, des ellipses, des lignes, des rectangles, de remplir le contenu d'une forme, d'effacer le contenu de la fenêtre de travail ou d'agrandir une zone de cette fenêtre.

AN.4.2.3. Commandes alpha-numériques

Ces commandes permettent:

- de sortir de l'éditeur ("sortir"),
- de sauvegarder un pavé.

Le pavé est mémorisé sous un nom choisi par l'opérateur grâce à l'ouverture d'une fenêtre "question". La fonction sauvegarde permet de délimiter dans la fenêtre la zone qui représente le pavé.

Le pavé est soit mémorisé sous la forme d'un bitmap compressé, soit sous la forme d'une liste qui reprend le nom et la position de chacun des pavés élémentaires qui entrent dans la réalisation du pavé final.

- de sélectionner une grille Bliss

- d'agrandir les dimensions d'un pavé ("zoom")(figure AN.22)

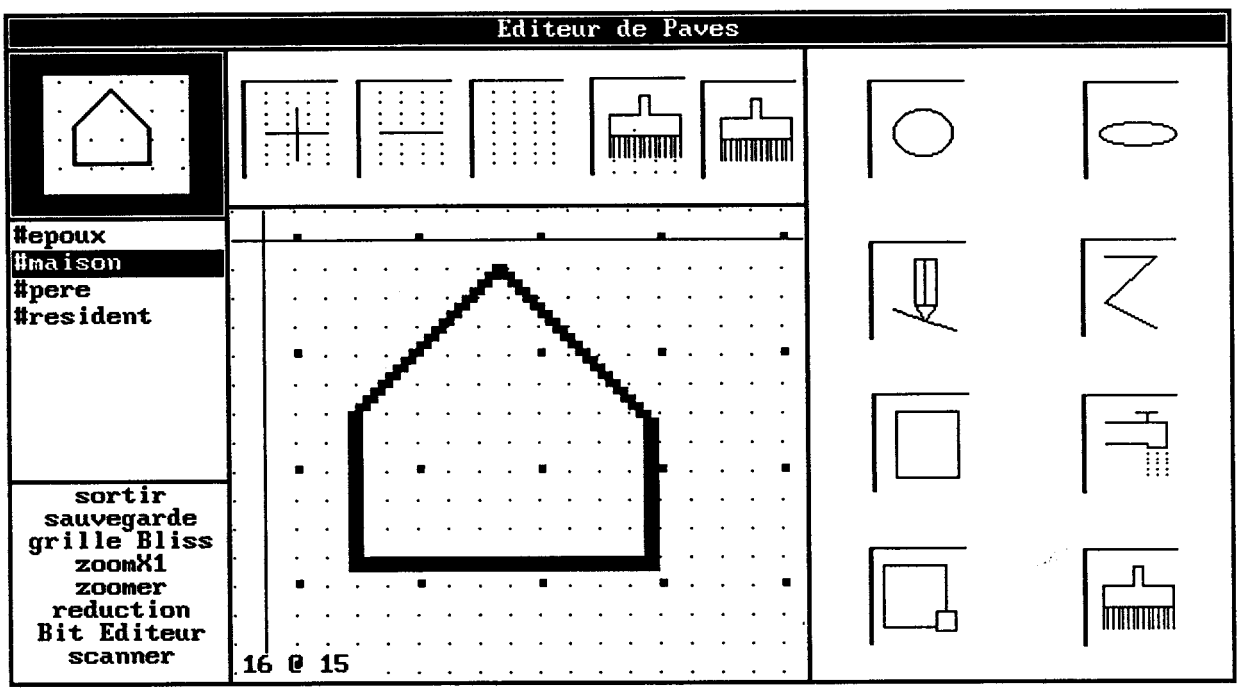


figure AN.22

Afin d'augmenter la précision, il est possible d'agrandir une partie de l'écran choisie par l'opérateur. La visualisation du pavé en grandeur réelle se fait dans la fenêtre de visualisation située sur la partie supérieure gauche de l'écran.

- de quitter la fonction d'agrandissement

Le pavé est représenté sous ses dimensions exactes dans la fenêtre de travail.

- de faire appel au bit-éditeur

Le bit-éditeur permet le dessin point par point. Le dessin est agrandi. Chacune des cases de la grille représente l'état d'un pixel de l'écran vidéo. En positionnant un curseur et en agissant sur le bouton gauche de la souris on peut modifier l'état de chacun des pixels. Une fenêtre de visualisation en grandeur nature permet de voir quel sera le résultat final.

- d'appeler la fonction scanner

Cette fonction permet de charger, à partir d'un nom, le fichier d'un pavé obtenu grâce à un scanner. Les dimensions du pavé, et la position du pavé dans la fenêtre de visualisation peuvent être choisies par l'opérateur.

AN.4.3. Fenêtre "liste".

Cette fenêtre contient la liste de pavés qui peuvent être visualisés dans la fenêtre de travail. Ces pavés peuvent être disposés dans la fenêtre d'une façon quelconque, ils peuvent être modifiés ou associés suivant les désirs de l'opérateur. Initialement cette liste contient les pavés contenus dans la mémoire "memoPave".

Le pavé mémorisé est immédiatement ajouté à la liste. Il peut donc être visualisé, modifié, dupliquer.. avant d'être à nouveau mémorisé.

BIBLIOGRAPHIE

[BEN.82] **BENTEUX**
Structuration d'un dictionnaire Bliss sur Micro-ordinateur.
Rapport de DEA d'Automatique et Informatique Industrielle,
U.S.T.L.F.A. 1984.

[DEL.82] **C. DELOBEL, M.ADIBA.**
Bases de Données et systèmes relationnels.
Dunod. 1982

[DIG.86] **DIGITALK INC.**
Smaltalk/V.
Tutorial and Programming Handbook.

[FLO] **A. FLORY**
Bases de Données. Conception et Réalisation.
Les Editions d'Organisation.

[GOL.83] **A. GOLDBERG, D. RUBSON**
Smalltalk-80.
The Language and its Implementation.
Addison-Wesley Publishing Company.

[JAU] **C.JAULT**
Les Bases de Données relationnelles ou le libre accès aux informations.
Les Editions d'Organisation.

[LAU.82] **J.L. LAURIERE**
Représentation et utilisation des connaissances.
TSI, Vol. 1, N°1 et 2, 1982.

[MCN.85] **S. MCNAUGHTON**
Communicating with BLissymbolics.
The Blissymbolics Communication Institute,
Toronto, Canada.

[MEV.87] **A. MEVEL, T. G. GUEGUEN**
Smalltalk-80.
Ed. EYROLLES.

[PAI.79] **C. PAIR, M.C. GAUDEL**
Les structures de données et leur représentation en mémoire.

[PEC.84] **H.PECKMAN**
Le Basic sur le Bout des Doigts.
McGraw-Hill Editeurs, 1984.

[PEE.82] **E. PEETERS**
Conception et Gestion des Banques de Données.
Les Editions d'Organisation. 1982

[SLI.86] **Y. SLIMANI**
Structures de données et langages non procéduraux en informatique graphique.
These U.S.T.L.F.A. Février 1986

*[CHA.74] **CHARNIAK**
He will make you take it back.
Pragmatics of the language. Memo N°5.
Institute for Semantic and Cognitive Studies.

*[CHO] **N. CHOMSKY**
Aspects of the Theory of Syntax.
Cambridge-Mass.: MIT.

*[FER.84] **J. FERBER**
La compréhension automatique du texte.
Micro Systèmes, Septembre-Octobre-Novembre 1984.

*[FIL.77] **C.J. FILLMORE**
The Case for Case in Universals Linguistic Theory
Rinehart and Winston Inc.

[GRO.75] **M.GROSS**
Méthodes en Syntaxe.
Herman

[GRE] **GREVISSE**
Précis de Grammaire française.

*[GRO.77] **M.GROSS**
Grammaire Transformationnelle du Français: Syntaxe du Nom.
Larousse.

*[HAR.71] **HARRIS**
Methods in Structural Linguistics.
University of Chicago.

*[IRI.79] **IRIA**
Reconnaissance et Synthèse de la Parole.
Etat de la recherche et du développement.

*[JAY.82] **J-H. JAYEZ**
Compréhension Automatique du Langage Naturel.
Masson.

*[SAG] **N. SAGER**
Natural Language Analysis and Processing.
Encyclopedia of computer Science and Technology.
Dekker Inc.

*[SAL.76] **SALKOFF**
L'analyse de la Langue Ecrite et Parlée et ses Applications.
Ecole Informatique de l'IRIA.

*[SCH.75] **R.C. SCHANK**
Conceptual Information Processing.
Amsterdam. North-Holland.

---

* Bibliograhie spécifique au problème de la transformation.
Thèse de TICHON Jacques. Conception Assistée de Communicateurs pour Handicapés. Le problème de la Transformation. (U.S.T.L. 1991)

[SHA.90] **A. SHALIT, D.A. BOONZAIER**
MACINTOSHtm Based 'Semantographic' Technique with Adaptive-Predictive Algorithm for Blissymbolic Communication.
Augmentative and Alternative Communication.
AAC, Vol. 6, N°2, June 1990.

[VER.87] **E. VERMOTE**
Etude de la traduction de français condensé en français clair.
Mémoire de E.E.A LILLE Juin 1987

[WAL.90] **A. WALLER**
Using a Predictive Word Processor - A Case Study.
Augmentative and Alternative Communication.
AAC, Vol. 6/N°2, June 1990.

*[WIL.75] **Y. WILKS**
Natural Language Understanding System within the A.I. Paradigm: a Survey and Some Comparisons in Linguistic Structures Processing.
Zampolli Ed., Amsterdam, North Holland.

*[WOO.75] **WOODS**
Transtion Networks for Natural Language Analysis.
Fondation for Semantic Networks.
Academic Press.

[ASH.82] **G. ASCH**
Les Capteurs en Instrumentation Industrielle.
Dunod

[BAU.88] **B. BAUDEL, J.M. TOULOTTE, J. TICHON**
Dispositif d'aide à la communication.
Congrès Internationnal "Communication et Handicaps"
PARIS février 1988

[BIE.76] **A.W. BIERMANN, R.KRISHNASWAMY**
Speeding Up the Synthesis of Programs
IEEE Transaction on Computers,
Vol. C-24(2), 1975.

[CIC.83] **J.M. CICHOWLAS, J.M. TOULOTTE**
Preeditor and modular structure.
International symposium MIMI84
Bari Italie June 5-9, 1984

[DEL.87] **M. DELFORGE**
Développement d'une interface programmable permettant aux personnes handicapées d'exprimer leurs desiderata en milieu hospitalier.
Mémoire d'ingénieur ISIB Bruxelles, Juin 1987

[HAN.85] **D.F HANSON et P.C. POWER**
Electronic Scanners with Speech Output - A Communication System for the Physically Handicapped and Mentally Retarded.
IEEE Micro, Vol.5, N°2, Apr. 1985, pp.25-52.

[HE.87a] **B. HE**
Génération Automatique de Logiciels d'Aide à la Communication pour Handicapés.
Thèse, U.S.T.L.F.A., Octobre 1987. p3

[HE.87b] **B.HE, J. TICHON, J.M. TOULOTTE**
Génération De Communicateurs Spécifiques Pour Handicapés.
Colloque International sur la technologie au service des personnes handicapées - Technology for disabled persons.
HANDITEC. Decembre 7-8, 1987.
Paris, France.

[HEC.83] **C.W. HECKATHORNE et D.S. CHILDESS**
Applying Anticipatory Text Selection in a Wrinting Aid for People with Severe Motor Impairement.
IEEE Micro, Vol.3, N°3, June 1983, pp.17-23.

[HEW.75] **C.E. HEWIT, B. SMITH**
Towards a Programming Apprentice.
IEEE Trans. Soft Eng. SE-1 (1), Mar. 1975.

[KRA.82] **S. KRAKORIAK**
Systèmes Intégrés de Production de Logiciel: Concepts et Réalisation.
Actes des Journées BIGRE, Grenoble, Jan.1982.

[MAN.84] **A. MANDAR et J.P. DUBUS**
Nouvelles Techniques de Communication à l'Usage des Infirmes Moteurs Cérébraux par Edition de Matrice Dynamique.
ITBM, Vol.5, N°6, 1984, 657-664.

[MYE.82] **W.MYERS**
Personal Computers Aid the Handicapped.
IEEE Micro, Vol.2, N°1, Feb. 1982, pp. 7-17.

[OUA. 86] **B. OUASSIR**
Techniques d'Accélération pour un Système d'Aide à la Communication.
Rapport de DEA d'Automatique et Informatique Industrielle, U.S.T.L.F.A., 1986.

[ROS.82] **M.J. ROSEN**
Communication Systems for the Nonvocal Motor Handicapped: Practice & Prospects
IEEE Engineering in Medicine and Biology Magazine Vol.1, N°4, Dec 1982, pp.31-35.

[THO.81] **A. THOMAS**
Communication Devices for the Nonvocal Disabled.
IEEE Computer, Vol.14, N°1, Jan.1981, pp.25-30.

[TIC.85] **J. TICHON**
Conception et Réalisation d'un Système de Communication pour Handicapés, utilisant des Techniques d'Accès à un Dictionnaire.
Thèse d'Université, U.S.T.L.F.A., Juillet 1985.

[TIC.86] **J. TICHON, J.M. TOULOTTE**
Conceiving and development of a system of communication for handicapped people.
Fourth International Symposium IASTED Applied Information Insbruck-Austria February 18-20, 1986

[TIC.87] **J. TICHON, B. HE, J.M. TOULOTTE**
Generation of Specific Communicators for the Handicapped Persons.
34th ISMM International Conference on Mini and Microcomputers, Lugano, Switzerland, July 1987

[TIC.87] **J. TICHON,B. BAUDEL,J.M. TOULOTTE**
Object Oriented Language Aided Design For Disabled Persons - Project Cachalot.
9th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, BOSTON (USA) November 1987

[TIC.89] **J. TICHON, G. TREHOU, J.M.TOULOTTE**
Use of Object Oriented Language in Fast Prototyping of Communicators for Disabled.
11th Annual International Conference of the IEEE Engineering in Medicine and Biology Society
Seattle (USA), 9-12 novembre 1989.

[TOU.83] **J.M. TOULOTTE, J.M. CICHOWLAS**
La Préédition dans l'Aide à la Communication.
Handitec 1983.

[TOU.86] **J.M. TOULOTTE, J. TICHON**
Development of a communication system for handicapped persons - Generation of system.
Proceedings of the eight annual conference of the IEEE Engineering in Medicine and Biology Society
DALLAS Forth Worth Texas November 7-10, 1986

[TOU.87] **J.M. TOULOTTE, J. TICHON**
Méthodes d'accélération dans un système de communication pour handicapés.
Innovation et Technologie en Biologie et Médecine.
VOL. 8, N°5, 1987.

[TRE.90] **J.M. TOULOTTE, G. TREHOU, B. BAUDEL-CANTEGRIT**
Acceleration Method Using a Dictionary Access in a Blissymbolics Communicator.
The Fourth Biennial International ISAAC Conference on Augmentative and Alternative Communication.
Stockholm, August 13-16,1990.

[VAN.82] **A. VAN LAMSWEERDE**
Automatisation de la Production de Logiciels d'Application: Quelques Approches.
TSI, Vol.1, N°6,1982; Vol.2, N°1,2, 1983.