Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Connaissance du corps interne : réalisation d'un logiciel d'apprentissage pour personnes ayant une déficience mentale

Fontesse, Alain; L'homme, Bernard

Award date:
1991

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 03. Jul. 2025

Année académique 1990-1991

Connaissance du corps interne: Réalisation d'un logiciel d'apprentissage pour personnes ayant une déficience mentale

Alain Fontesse **Bernard L'homme**

Promoteur: **Madame Monique Noirhomme-Fraiture**

Co-promoteur: **Monsieur Michel Mercier** , professeur au Département de Psychologie de la faculté de Médecine, F.U.N.D.P.

Mémoire présenté en vue de
l'obtention du grade de Licencié
et Maître en informatique

Abstract

Ce travail présente une application particulière de l'outil informatique : un logiciel d'aide à la connaissance du corps humain et de son fonctionnement pour personnes ayant un handicap mental. Il constitue une approche pluridisciplinaire composée, dans notre cas, par une collaboration entre informaticiens et spécialistes du handicap mental et de la pédagogie.

This study represent a particular application of the information processing: a user's guide for person with mental deficience to understand the human body and the way he work. This constitutes a multidisciplinary approach composed, in our case, of a collaboration between computer experts and specialized persons in the domain of mental handicap and education.

Nous voudrions témoigner toute notre reconnaissance à Madame Monique Noirhomme-Fraîture qui a assuré la direction et le suivi de ce mémoire, ainsi que ses recommandations éclairées. Nous remercions également Monsieur Michel Mercier, Co-promoteur du projet, pour l'attention qu'il nous a prêtée et pour les conseils qu'il n'a pas manqué de nous donner. Nous remercions également Madame Jacqueline Delville pour sa collaboration active et son aide bénéfique.

Nous désirons remercier particulièrement Monsieur Jean-Luc Collignon pour sa constante disponibilité et sa grande participation active et positive à ce mémoire.

Nous remercions de même:

- Monsieur Alexandre Waeber pour son accueil et son pilotage avisé en Suisse, ainsi que pour l'intérêt qu'il a porté à notre activité.

- le département de Psychologie, pour nous avoir accueillis dans ses infrastructures, et notamment, Martine, Odette et Jacques, pour leur aide et leur soutient.

- aux éducateurs et psychologues des institutions belges et suisses avec lesquels nous nous sommes entretenus.

Un grand merci enfin à toutes les personnes qui, d'une manière ou d'une autre, ont permis la réalisation de ce mémoire.

Table des matières

Introduction **1**

Chapitre 1 : Nécessité d'un logiciel d'apprenissage comme un des outil à l'éducation pour la santé **3**

Introduction 3
1.1.Besoins existants chez les personnes handicapées 3
1.1.1. La mortalité et l'espérance de vie chez les personnes handicapées mentales 3
1.1.2 Problème de santé des personnes handicapées mentales 4
1.1.3. Causes des problèmes de santé chez les personnes handicapées mentales 6
1.2. Apprentissage de connaissance une solution 7
1.3. Conclusions 8

Chapitre 2 : L'ordinateur média d'enseignement **9**

Introduction 9
2.1. Composantes d'un système adaptatif 9
2.2. Définitions 10
2.3. Fonctions de l'enseignement 11
2.5. Facteurs et règles favorisant l'apprentissage 14
2.5.1. La nature de la réponse 14
2.5.2 La progressivité et le recours aux indices 15
2.5.3. Renforcement et feed-back 16
2.5.4. Durée des exercices 17
2.6. L'intelligence artificielle un avenir? 17
2.7. Conclusions 19

Chapitre 3 : Le modèle de l'élève **21**

Introduction 21
3.1. Le handicap mental 21
3.1.1. Le niveau intellectuel 21
3.1.2. L'adaptation sociale 22
3.1.3. Définition de l'arriération mentale 22
3.1.4. Classification de l'arriération mentale 23
3.2. Les variables entrant en ligne de compte 23

3.2.2 Aptitudes intellectuelles demandées 24
3.2.3 Le niveau de maîtrise des objectifs assignés au cours 28

Chapitre 4 : Le modèle du domaine 32

Introduction 32
4.1. Le domaine 32
4.2. Structuration de la connaissance 32
4.3. Support de la représentation 34
4.4. Nature du domaine 35

Chapitre 5 : La genèse du projet 38

Introduction 38
5.1. Le contexte 38
5.2. Définition du projet initial 39
5.3. Les modifications apportées 40
5.4. Conclusions 42

Chapitre 6 : La démarche de développement 43

Introduction 43
6.1. Le projet
une application interactive 43
6.1.1. Modèles d'architectures pour systèmes 44
6.1.1.1. Le modèle LANGAGE 44
6.1.1.2. Le modèle ENTREE/SORTIE 45
6.1.1.2.Le modèle MULTIAGENT 45
6.1.2 Le modèle d'architecture de base 48
6.1.2.1. Description 48
6.1.2.2. Spécification d'un dialogue 51
6.2. Réalisation d'une interface adaptée 53
6.2.1 L'ergonomie 53
6.2.2 Qualités d'une bonne interface 56
6.2.3 Nos choix pour une interface de qualité 59
6.3. Implication des utilisateurs 61
6.3.1. La participation des personnes handicapées mentales 63
6.4. Notre démarche de spécification 64
6.4.1. La spécification des scénarios 64
6.4.2. La spécification de l'interface concrète 65
6.5 Conclusions 71

Chapitre 7 : Les spécification de l'outil 72

Introduction 72

7.1.1. diversité de la population cible 72
7.1.2. un outil ouvert, une préoccupation primordiale 73
7.1.3 L'outil général 75
7.2. Spécification du logiciel "Corps interne" 77
7.2.1. Présentation du logiciel 77
7.2.2. Spécification des scénarios et de l'interface concrète 78
7.2.2.1 La méthode utilisée 78
7.2.2.2 La désignation 79
A. Scénario de base 79
B. Paramètres d'une désignation 81
C. Le scénario détaillé 82
7.2.2.3 Le parcours 89
A. Scénario de base 89
B. Paramètres du parcours 90
C. Le scénario détaillé 90
1. Représentation graphique 90
2. Spécification des messages internes 91
7.2.2.4. La session d'exercices 94
A. Scénario de base 94
B. Le scénario détaillé 95
1. Représentation graphique 95
2. Spécification des messages internes 96
7.3 Spécification du logiciel "paramètrage" 98
7.3.1 Présentation du logiciel 98
7.3.2. Spécifications des fonctionnalités 98
7.3.2.1. L'initialisation d'une disquette utilisateur 98
7.3.2.2. L'impression d'une trace textuelle 99

Conclusion 100

Introduction

L'ordinateur de plus en plus s'impose dans le domaine de l'enseignement. Il est de plus en plus présent dans les écoles primaires, secondaires et les universités. Des programmes visant à introduire l'informatique dans les écoles ont vu le jour dans certains pays comme la France. Ces programmes connurent un succès assez mitigé. Une de ces causes que l'on peut retenir est la mauvaise formation et information des enseignants à l'informatique ainsi que la petite quantité de logiciels de bonne qualité. Cependant, l'ordinateur en tant qu'enseignant a un bel avenir devant lui à condition que les logiciels d'enseignement tiennent mieux compte des individualités, des aptitudes cognitives des élèves, des méthodes d'apprentissage. Les usages de l'informatique dans ce domaine peuvent être très variés. CORNER ET HEBENSTREIT [cités dans DUFOYER, 1988] le soulignent : " l'ordinateur peut être utilisé de mille façons comme un tableau d'école, comme un appareil à simuler des phénomènes, comme une machine à proposer des exercices".

Ce mémoire concerne la réalisation d'un logiciel d'apprentissage des organes et du fonctionnement du corps humain interne. Il est destiné aux personnes handicapées mentales légères et modérées. Cette application particulière de l'informatique au service de l'enseignement représente ici un moyen attractif, parmi d'autres moyens, d'accéder à une connaissance du corps humain.

Dans un premier chapitre, nous mettrons en évidence le besoin de connaissances nécessaires aux personnes handicapées mentales pour faire l'apprentissage de comportements favorables à leur santé. Par ce biais, elles acquerront une maîtrise plus grande de leur corps, leur permettant de développer leur potentialité et d'accéder à une autonomie et un bien être plus grand. Un logiciel d'apprentissage est un des moyens pertinents mis à leur disposition pour réaliser cet objectif.

Nous parlerons de l'ordinateur comme média d'enseignement dans un deuxième chapitre. Nous mettrons en évidence les mécanismes utilisés et certains concepts importants comme la motivation, la nature de la réponse, le feed-back, le renforcement,... après avoir énuméré les atouts de l'ordinateur comme enseignant.

Dans le chapitre suivant nous développerons le domaine particulier de connaissances que nous voulons transmettre à travers le logiciel, c'est-à-dire la description des organes du corps et de leur fonctionnement.

La définition d'un modèle particulier de l'élève concernant la personne handicapée dans ses caractéristiques cognitives et en tant que membre d'une population très hétérogène nous préoccupera dans un quatrième chapitre. Nous y énoncerons les prérequis nécessaires à l'utilisation du logiciel. Pour certains d'entre eux, nous tenterons, à partir d'une théorie sur le développement de l'enfant et des rapports qui peuvent exister avec les stades de développement des personnes handicapées mentales, de montrer que ces dernières possèdent bien ces prérequis.

Nous donnerons, dans le chapitre 5, l'origine du projet dans lequel s'inscrit notre logiciel avant de passer à une description de notre méthode de développement et de réalisation.

Nous exposerons, dans le chapitre 6, notre démarche de développement du logiciel. Nous identifierons les acteurs qui ont participé à la phase de spécification. Nous indiquerons les choix que nous avons adoptés pour la réalisation de l'interface.

Enfin nous donnerons les spécifications complètes du logiciel.

Chapitre1:Nécessité d'un logiciel d'apprentissage comme un des outils à l'éducation pour la santé

Introduction

Ce chapitre a pour but de mettre en évidence les problèmes de santé que peuvent rencontrer les personnes handicapées mentales, d'en examiner les causes qui sont le peu d'acquis dans le domaine de la connaissance de leur corps et de son fonctionnement, et de proposer une solution matérialisée par l'éducation à la santé. Cette dernière consiste à modifier, par la réalisation d'apprentissages, des comportements et styles de vie néfastes à la santé. L'application de l'informatique en tant que logiciel d'apprentissage est un des moyens permettant de remédier à cette situation.

1.1.Besoins existants chez les personnes handicapées mentales

1.1.1. La mortalité et l'espérance de vie chez les personnes handicapées mentales

Il semble que la mortalité soit plus élevée chez les handicapés mentaux que dans la population générale [FORSSMAN et AKESSON, 1970] [cités dans COLLIGNON et GALAND, 1990]. Elle augmente avec la profondeur du handicap [RICHARD et SYLVERSTER, 1969] [cités dans COLLIGNON, op cit] et est élevée en bas âge. L'espérance de vie des personnes handicapées mentales augmente pour celles qui survivent en bas âge. Ainsi, 50 % seulement de la population étudiée en institution, peut espérer atteindre l'âge de vingt ans (47 % chez les hommes et 52 % chez les femmes) [BALAKRISHNAN et WOLF,1976] [cités dans COLLIGNON, op cit]. L'espérance de vie à la naissance est de 29,13 ans chez les hommes et 32,27 ans chez les femmes. Cependant, pour celui qui passe l'enfance, l'espérance de vie devient plus élevée. A 20 ans, elle est de 56,76 ans chez

les hommes et de 57,59 ans chez les femmes.

Les causes de mortalité les plus fréquentes sont les maladies respiratoires. L'épilepsie constitue l'autre cause de mortalité très courante en institution. Les maladies coronairiennes et les hémorragies cérébrales sont des causes de décès moins fréquentes.

Bien que la mortalité soit plus élevée dans la population handicapées mentales que dans la population normale, l'examen des données de la littérature ne nous permet pas de sélectionner des actions possibles qui permettraient d'améliorer la santé de ces personnes.

1.1.2 Problème de santé des personnes handicapées mentales:

Puisque l'étude de la mortalité ne peut nous aider à déterminer des priorités d'actions, nous nous sommes attaché à la morbidité et aux problèmes de santé de personnes handicapées mentales. La santé est ici prise dans le sens défini par l'O.M.S.(Organisme Mondiale de la Santé) : "la santé est un état complet de bien être physique, mental et social et ne consiste pas seulement en une absence de maladies ou d'infirmités". Parmi ces problèmes de santé, plusieurs émergent de la littérature: obésité, carences alimentaires, caries dentaires, abus de tabac et d'alcool, manque d'exercice et de sport, stress.

A titre d'exemple, et de part l'importance de ce problème, nous ne développons ici que les données relatives à l'obésité. En effet, aussi bien la fréquence de la pathologie parmi cette population que les conséquences qui y sont liées sont importantes et déterminent une priorité d'action.

L'obésité

Parmi les problèmes de santé qui se posent chez les personnes handicapées mentales, l'obésité est certainement un des plus important. De nombreux auteurs observent un nombre disproportionné de poids excessifs chez les personnes handicapées mentales adultes en institution. Le problème est plus fréquent chez les femmes que chez les hommes et plus fréquent chez les handicapés de type léger et modéré que chez ceux de type sévère et profond. Ces résultats sont tirés d'études, notamment celle de FOX et ROTATORI [1982] portant sur 1152 sujets retardés de 4 institutions et résidences communautaires: ces derniers, prenant comme critère d'obésité un poids de 20% au dessus du poids désirable pour la taille observent des différences selon le handicap et le sexe des personnes. Chez les handicapés mentaux sévères et profonds, ils trouvent 6,9% d'obèses chez les hommes et 13,6% chez les femmes, et chez les handicapés mentaux légers et modérés, respectivement 27,9% et 38,2% pour les hommes et pour les femmes. L'obésité est plus fréquente chez les femmes (25,1%) que chez les hommes (15,6%), chez les handicapés légers et modérés que les sévères et profonds et la fréquence est plus élevée pour des âges plus élevés.

D'après des études faites chez les sujets normaux, l'obésité représente un facteur de risque d' insuffisance coronarienne même indépendamment de l'hypertension artérielle, du diabète et des hyperlipoprotéinémies.

Il faut également dire que l'obésité peut compliquer la chirurgie, entraîner des complications de type respiratoire, gynécologique, veineux, articulaire, digestif, cutané ou hormonal. Elle peut en outre provoquer des problèmes d'ordre psychologique du type isolement social et une pauvre image d'eux-mêmes [DELVILLE et COLLIGNON,1991].

Les comportements alimentaires sont une des nombreuses variables conduisant à l'obésité (pour certaines personnes du moins). Au niveau du style alimentaire, lors du repas principal, les études n'ont pas pu mettre en évidence des différences significatives entre obèses et non obèses. [BURKART, et coll. 1985]

Il semble y avoir un lien entre activité physique et obésité. Une relation a été montrée entre obésité et manque de condition physique [FOX, et coll. 1984]. De plus une différence de pratique d'activité physique pourrait être un facteur important de la plus grande fréquence d'obésité chez les femmes handicapées mentales selon BURDART, et coll. [1985].

La recherche de stratégies capables d'entraîner une perte de poids chez les personnes handicapées mentales devient indispensable. Elle peut s'inscrire dans une démarche de l'éducation pour la santé. Le manque de connaissance n'est cependant pas la seule cause d'obésité, causes qui comprennent aussi des pathologies.

1.1.3. Causes des problèmes de santé chez les personnes handicapées mentales

On pourrait s'intéresser à des conditions chroniques et des maladies liées à la sexualité qui sont des facteurs pouvant altérer la santé des personnes handicapées mentales. Les conditions chroniques et maladies liées au style de vie comprennent notamment l'obésité, l'abus d'alcool et d'autres substances, le tabac, le manque d'exercice et de sport, le stress excessif et l'anxiété extrême. Certains de ces problèmes dans la mesure où ils sont déterminés en partie par un style de vie, où interviennent des connaissances, attitudes et comportements peuvent être résolus en partie par une *éducation pour la santé.* Cette dernière est définie par L. GREEN [cité dans HUBENS et LECRON, 1990] comme étant une combinaison d'expériences, d'apprentissages planifiés, destinés à faciliter l'adaptation volontaire de comportements conduisant à la santé.

Une éducation pour la santé est d'autant plus importante qu'on constate des erreurs et des lacunes importantes chez les handicapés mentaux dans la connaissance et la représentation du corps et de son fonctionnement [DELVILLE et COLLIGNON, 1991]. Par exemple, au niveau de la sexualité, certains auteurs n'hésitent pas à affirmer que l'ignorance serait la première cause de dysfonctionnement sexuel chez les

couples handicapés mentaux [ANDRON, 1983] [cité dans COLLIGNON GALAND, 1990]. La nécessité d'un apprentissage cognitif apparaît dès lors. Une connaissance minimale des mécanismes physiologiques est nécessaire pour arriver à une gestion par l'individu de sa vie affective et sexuelle. Cet apprentissage doit repartir de la base et être extrêmement simple pour être compréhensible. Une connaissance très lacunaire de l'alimentation et de ses mécanismes peut expliquer la problématique de l'obésité chez les personnes handicapées mentales. Ces résultats montrent l'importance d'asseoir une base minimum de connaissance par rapport au corps et à son fonctionnement.

1.2. Apprentissage de connaissance : une solution

Des apprentissages basés sur des modifications de comportements se sont déjà montrés efficaces chez les personnes handicapées mentales dans le traitement de l'obésité, de l'hygiène dentaire et dans le domaine de la sexualité[BURCARD, et coll. 1985].

L'approche multimédias sera privilégiée : usage d'un outil audiovisuel de formation et sensibilisation, d'un outil graphique et d'un outil informatique. Un logiciel d'apprentissage apparaît donc comme un moyen à utiliser parmi d'autres pour remédier à ce manque de connaissance. Il constitue un des maillons précurseurs d'une stratégie pour l'éducation à la santé, elle-même une préoccupation récente.

1.3. Conclusions

L' éducation à la santé est d'autant plus importante que les connaissances dans ce domaine sont faibles. Une connaissance plus grande du corps et de son fonctionnement augure des améliorations du comportement permettant d'acquérir un style de vie favorable à la santé des personnes handicapées mentales. En améliorant les connaissances et représentation du corps, à l'aide notamment d'un logiciel d'apprentissage, nous espérons faciliter l'adaptation volontaire de comportements bénéfiques à la santé, et par là même, améliorer le bien être et la qualité de vie des personnes handicapées mentales. L' apparition de notre logiciel d'apprentissage permet de combler une partie du vide existant et répond pour le moins à un besoin.

Chapitre 2 : L'ordinateur Média d'enseignement

Introduction

Dans ce chapitre nous débuterons par une brève description d'un système d'enseignement adaptatif qui est le système d'enseignement pour lequel l'ordinateur est le mieux adapté. Nous donnerons dans un deuxième temps les atouts de l'ordinateur en tant qu'outil d'enseignement par rapport à d'autres moyens. En dernier lieu nous parlerons de quelques facteurs et règles influençant un apprentissage dont la prise en compte peut permettre à un ordinateur utilisé à des fins d'enseignement d'atteindre le but qu'on lui assigne à savoir apprendre.

2.1. Composantes d'un système adaptatif

Un système d'enseignement adaptatif se caractérise par la présence d'une relation bidirectionnelle entre A et E (figure 2.1). La relation de E vers A représente le transfert d'information du système d'enseignement vers l'apprenant. La relation de A vers E matérialise l'existence d'un retour d'information de l'apprenant vers l'organe d'enseignement. Ce dernier exploitera cette information afin de réguler son action de manière à atteindre les buts qui lui sont assignés (relation de E vers B).

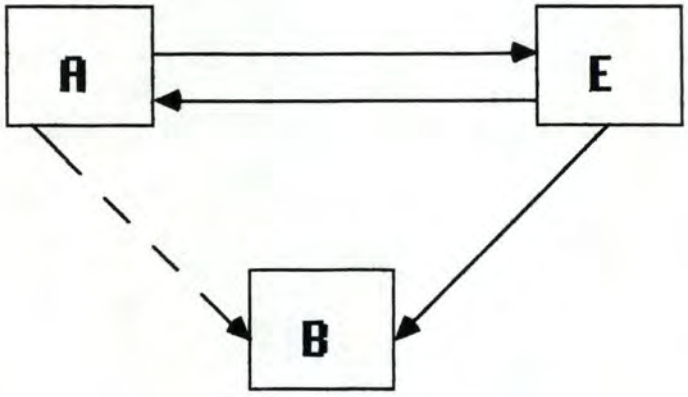


figure 2.1 : système adaptatif

La relation de A vers B (en pointillés) témoigne de la possibilité offerte à l'élève, par certains systèmes d'exercer un contrôle sur les buts qui sont poursuivis [DEPOVER, 1987].

2.2. Définitions

Nous parlerons de *système adaptatif* uniquement lorsque les possibilités de régulation existent durant l'ensemble du processus d'enseignement, autorisant ainsi un ajustement immédiat (régulation immédiate) des actions d'enseignement à l'action d'apprentissage manifestée par l'apprenant.

Par contre, lorsque la régulation se limite à certains moments précis, l'expression *système d'enseignement flexible* est utilisée. Notre logiciel se classe plutôt dans cette catégorie.

En général, tout programme informatique destiné à enseigner une matière est désigné par l'appellation: *didactitiel.*

Le domaine (fig. 2.1) jouera un rôle d'interface entre les organes A et E en fournissant l'objet d'apprentissage qui sera proposé à l'élève ainsi que le produit qui reflétera l'action de l'apprenant sur l'objet [D'HAINAUT, 1980] cité dans [DUFOYER, 1988]. Nous en parlerons dans le chapitre suivant.

2.3. Fonctions de l'enseignement

Le contrôle de l'enseignement et de rétroaction constituent deux concepts fondamentaux d'un enseignement centré sur l'efficacité. Ces principes permettent de contrôler le processus d'apprentissage grâce à l'exploitation des indices révélés par l'apprenant.

Le principe consistant à contrôler, par rapport à un effet attendu clairement défini, un organisme vivant ou une machine à travers un retour d'informations, a été énoncé par WIENER [1948] dans ses travaux concernant la cybernétique. La cybernétique, qui est la science du contrôle et de la communication, se donne pour objet d'étude les processus qui ont une visée, un but à atteindre par l'action, indépendamment du domaine de connaissance sur lequel ces processus opèrent.(loi de l'effet)

Les fonctions essentielles de l'enseignant consistent donc à provoquer chez l'élève certaines modifications comportementales d'ordre intellectuel, affectif ou psychomoteur à travers un processus d'incitation à l'action et de transmission d'informations, à contrôler en cours d'apprentissage l'évolution de ses savoir-faire et des attitudes de l'élève vers l'objectif et à réaliser ensuite les ajustements nécessaires [DUFOYER, 1988]. Cette démarche doit partir des connaissances préexistantes de l'élève comme base d' acquisition à d'autres comportements ou pour les redresser si elles sont erronées.

Pour se faire, l'enseignant entreprend une démarche en 4 phases:

- -la fixation des objectifs qu'il tente d'atteindre,
- -un contrôle, en cours de processus, des effets des stratégies appliquées par le biais de questions à l'élève,
- -l'ajustement de son enseignement à la progression de l'élève, et en fin de formation,
- -la vérification du niveau de maîtrise des objectifs et la décision des actions correctives à entreprendre qui s'avèrent utiles.

2.4. Atouts de l'ordinateur enseignant

La dimension essentielle de l'enseignement assisté par ordinateur est l'exploitation, par les enseignants, de leur compréhension des mécanismes de l'apprentissage et de leur connaissance des possibilités de l'l'ordinateur pour construire un enseignement qui conduise plus d'élèves à un niveau de capacité plus élevé [DEPOVER, 87].

L'ordinateur présente une autonomie de fonctionnement permettant de libérer l'action éducative des contraintes spatiales et temporelles qui lui sont habituellement associées.

L'ordinateur répond également bien à certaines caractéristiques nécessaires pour enseigner. Parmi celles-ci, DUFOYER [1988] souligne la possibilité de stocker une grande capacité d'informations qu'on peut extraire en des délais très courts. L'ordinateur est un moyen puissant de traiter de l'information. Or l'apprentissage cognitif est un traitement de l'information et il en va de même de l'enseignement.

On peut être frappé par le nombre important de facteurs d'apprentissage que l'ordinateur permet de contrôler et de mettre en oeuvre pour contribuer à l'enrichissement de l'apprenant selon les voies qui lui sont propres. Il est possible, par programme, de proposer des apprentissages de niveaux variables sur des thèmes différents, d'évaluer les réponses de l'élève et d'offrir divers chemins ou divers degrés de difficulté en fonction de la qualité de réponses reçues. L'ordinateur devient alors une machine à enseigner personnalisée qui s'adapte différemment à chaque usager en fonction des réponses fournies. Il est impossible d'atteindre ce niveau d'individualisation dans l'enseignement conventionnel. L'apprentissage réalisé sur base d'un professeur pour un élève n'est pas imaginable. Tout au plus, pour les petites classes, l'enseignant peut accorder plus de temps aux élèves et suivre la méthode qui consiste à lancer une activité individuelle chez chacun des élèves pour ensuite les aider individuellement tour à tour. C'est cette technique qui est utilisée dans l'enseignement spécial.

Si les logiciels sont bien construits et si les systèmes de questions réponses sont convenablement étudiés, cet enseignement devrait pouvoir pallier à la passivité coutumière des situations d'apprentissage par une attitude active et interactive plus favorable.

DE LEEUW [1984] [cité dans DUFOYER, 1988] indique qu'un certain nombre de variables de personnalité peuvent agir sur l'efficacité des tâches de résolution de problèmes, c'est le cas de la peur de l'échec qui contribue à l'inhibition intellectuelle. Un certain nombre de situations qui sont parfois socialement pénibles lors d'une interaction entre enseignant humain et élève peuvent être ressenties différemment quand l'élève se trouve seul face à l'ordinateur. Pas de jugement social public de l'enseignant ou des autres élèves, pas d'évaluation forcément connue du nombre d'essais et d'erreurs avant de parvenir au résultat demandé. Cela restera une affaire individuelle et personnelle.

DEPOVER [1987] souligne le grand intérêt de l'ordinateur comme moyen d'enseignement par rapport aux techniques conventionnelles lorsqu'on l'utilise à des fins de rattrapage. Et donc, l'ordinateur présente un interêt particulier lorsqu'on s'adresse à des élèves moins performants.

Loin de nous l'idée de tomber dans un optimisme béat concernant les capacités de l'ordinateur enseignant. S'il a des qualités, des prédispositions à l'enseignement, il a aussi des limites. Il ne faut pas tomber dans le piège de l'ordinateur monopole de l'enseignement. On s'accorde à penser que la meilleure forme d'enseignement est toujours la technique conventionnelle de l'enseignant humain. L'ordinateur peut enrichir l'enseignement par une diversification des moyens conventionnels. Il ne constitue donc pas un remplaçant mais un auxiliaire.

2.5. Facteurs et règles favorisant l'apprentissage

Des recherches centrées sur l'étude des facteurs susceptibles de favoriser l'apprentissage ont été entreprises. Elle n'ont certes pas apporté les résultats escomptés. Ces recherches n'ont fait que souligner le caractère relatif et la complexité de la plupart des variables investiguées. Nous tenterons, néanmoins, de souligner les aspects fondamentaux qui semblent influer sur l'apprentissage.

2.5.1. La nature de la réponse

Un premier facteur dégagé est la *nature de la réponse*. Plusieurs auteurs ont réalisé des expériences et divergent d'opinion. Pour Skinner [1958] [cité dans DEPOVER, 1987], psychologue béhavioriste américain, les réponses doivent être des réponses construites (opposées aux réponses à choix multiples). Par contre pour CROWDER [1960] [cité dans DEPOVER, op cit], les techniques d'analyse de réponses construites sont à ce point limitées que leur utilisation appauvrirait considérablement le processus de communication. Ces deux courants de pensée divisent les autres chercheurs .

De ces différentes expériences entreprises, il paraît difficile de soutenir l'idée d'une efficacité supérieure des réponses construites par rapport aux réponses choisies. Cependant, la tendance générale montre que les réponses construites sont plus adaptées à l'acquisition de compétences qui impliquent des productions verbales ou écrites. De même, elle est supérieure à la réponse choisie lorsque la matière enseignée atteint une complexité élevée ou lorsqu'on s'intéresse à des élèves particulièrement faibles (comme les personnes handicapées mentales).

Toutefois un apprentissage ne peut être efficace que si l'élève accomplit des activités qui soient réellement significatives par rapport aux compétences qu'on veut lui enseigner. On n'apprend pas à piloter un hélicoptère en ne lisant que la brochure technique.

2.5.2 La progressivité et le recours aux indices

Ce sont deux facteurs qui ont pour but de rapprocher l'apprenant du comportement souhaité à travers des situations d'apprentissage soigneusement graduées en termes de quantité d'informations transmises et de difficulté d'activités exigées. Pour assurer cette progressivité SKINNER fut le premier à introduire des aides afin de conduire l'élève au comportement souhaité.

Proposer des contenus ou des exercices faciles en début d'apprentissage assure une motivation dès le départ. Cette motivation est un élément important pour la suite. En effet une réussite au début tend à accroître l'intérêt pour la matière enseignée. La motivation est nécessaire pour qu'un enseignement soit efficace [DEPOVER, 1987].

Il faut cependant que le degré de difficulté ne soit pas trop bas. En effet, des contenus ou des exercices trop simples n'ont aucun intérêt pour l'apprentissage, la progression cesse. De plus le taux d'erreur quasi nul qui en résulterait signifierait que la progression de l'élève n'est plus possible. Cependant un taux d'erreur trop élevé aurait des conséquences tout aussi déplorables pour l'apprenant, une démotivation générale, un sentiment de frustration,... .

La redondance et l'exercice sont également nécessaires à l'apprentissage. Il s'agit d'assurer un dosage adéquat entre présentation d'informations nouvelles et redondances. La répétition d'un même comportement est surtout efficace lorsqu'il faut maîtriser des activités cognitives élémentaires. Cette dernière facilite l'acquisition de démarches mentales de niveau de complexité plus élevées.

2.5.3. Renforcement et feed-back

SKINNER a montré à l'occasion de recherches sur l'animal que tout comportement peut être appris au moyen d'une application adéquate des principes du conditionnement opérant à l'aide de renforcements *immédiats* et *adaptés* . Le renforcement est un stimulus qui conduit à l'augmentation de la fréquence d'apparition du comportement de la réponse correcte. Ces principes de pédagogie consistent à enseigner quelque chose à l'élève pour lui faire entreprendre une activité. Pour ce faire, des questions lui sont posées. Ses réponses sont renforcées en lui communiquant des informations qui lui permettront de décider si sa réponse est correcte.

SKINNER nous apprend de plus que l'efficacité d'un processus d'apprentissage dépend du fait que les renforcements positifs soient immédiats et rigoureusement adaptés. Dès lors une bonne élaboration des renforcements s'avère importante.

Le feed-back est défini par SKINNER comme le mécanisme de renforcement. Quant à CROWDER [1960] [cité par DUFOYER, op cit], il le définit comme ce qui sert à transmettre des informations permettant le redressement des erreurs.

De ces deux définitions [DEPOVER, 1987], le rôle communément admis du feed-back aujourd'hui consiste en un double effet :

- fournir une information essentielle à l'apprenant sur ce qu'il apprend et sur la manière qu' il apprend. On peut retrouver ici l'indice qui aide à la progression.
- remplir une fonction d'agent de renforcement se traduisant par une augmentation de la fréquence des réponses correctes de l'apprenant.

DUFOYER [1988] et DEPOVER [1987] donnent quelques règles faisant intervenir le feed-back et le renforcement pour l'élaboration de didactitiels (tout programme informatique qui a un but d'apprentissage). Le feed-back dans sa nature et son adéquation doit être au niveau et à l'âge de l'apprenant. Il doit être compréhensible par l'élève. Il faut s'assurer que l'information véhiculée par celui-ci est assimilable par l'élève. Le caractère descriminatif du feed-back est également important. La procédure de feed-

back doit être capable de souligner la divergence existant entre la réponse proposée et la réponse attendue.

Les renforcements doivent être variés nuancés et gradués en fonction de la difficulté (adaptés en se référant à SKINNER). Ils doivent apparaître directement après la réponse de l'apprenant (immédiatement). Lors d'une réponse erronée il est préférable de donner la nature de l'erreur plutôt que de simplement indiquer que la réponse est incorrecte.

Le recours aux graphiques ou à l'animation constitue non seulement une valeur informative comparable au feed-back verbaux, mais, de plus, ils contribuent à améliorer l'attention et la motivation des élèves selon une étude fort intéressante réalisée par LUTZ [1973] [cité dans DEPOVER, op cit]. Le soutient de la motivation est un attribut également important et exigé du feed-back pour réaliser son rôle de renforcement.

2.5.4. Durée des exercices

Il apparaît intéressant de contrôler la durée des exercices ou des leçons. Cette durée peut varier suivant les caractéristiques de l'élève (âge, niveau scolaire). TRUBERT [1984] indique que 5 à 10 minutes est une bonne durée pour les sessions réalisées avec les enfants de l'enseignement primaire.

2.6. L'intelligence artificielle : un avenir?

Certaines techniques mises au point dans le cadre des recherches en intelligence artificielle (I.A.) ont pu permettre la création d'environnements d'apprentissage laissant une large place à la redécouverte, à l'initiative de l'élève. Il s'agit de gérer des situations d'enseignement peu structurées au cours desquelles l'élève est placé devant des problèmes à résoudre, de le guider en infléchissant le moins possible sa démarche naturelle et lui fournir l'aide dont il a besoin au moment opportun.

Nous pouvons attendre, grâce aux méthodes de l'IA à quelques progrès. DUFOYER [1988] parle du progrès qui consisterait à munir ces logiciels d'une capacité d'analyse du langage naturel. Certes, aucun programme n'est capable encore d'entretenir une conversation avec un être

humain sur n'importe quel sujet. Cependant les recherches en IA ont permis de faire certains progrès dans cette direction. La littérature indique l'existence de dispositifs capables, dans une certaine mesure, de comprendre des histoires racontées avec un certain vocabulaire de tous les jours tel le programme SAM de l'Université de YALE [SCHANK et ABELSON, 1977] [cité dans DUFOYER, 1988]. Il nous paraît cependant important de souligner qu'il est assez difficile de se rendre compte des performances de ces dispositifs lorsqu'on ne peut que les découvrir à travers la littérature.L'enthousiasme que connaissent les chercheurs n'est pas toujours dicté par la prudence. DUFOYER, op cit, fait remarquer que ces programmes ne peuvent pas *"comprendre"* tout le langage naturel, mais ils acceptent un sous-ensemble assez large pour être compatibles avec un domaine de connaissances donné. On pourrait s'attendre à voir des logiciels autrement plus conviviaux et dès lors plus motivants.

Un autre progrès consisterait à partir du principe, comme le propose BONNET [cité dans DUFOYER, op cit], que les applications à vocation pédagogiques devraient être elle-mêmeS des experts du domaine considéré. Ce serait alors des programmes d'enseignement "intelligemment" assistéS par ordinateur qui seraient capables de donner diverses solutions au problème posé, de suivre et de contrôler la démarche de l'élève ou de l'étudiant, voire de la critiquer. BONNET a proposé un modèle assez général d'un tel système. Ce modèle peut largement être discuté mais nous nous contenterons d'en donner un aperçu succinct. Le modèle inclut trois éléments: un système expert, un module de tutoriel, une interface.

-Le premier élément est composé, comme tout système expert, d'une base de faits, d'une base de règles et d'un mécanisme d'interprétation propre au domaine concerné.

-Le deuxième élément est un dispositif qui permet de générer des problèmes en fonction des principes d'une stratégie tutorielle qui lui aura été fournie (et qui pourrait changer). Il devra également définir des stratégies de guidage donc être capable de se donner des réponses à des questions comme : jusqu'où faut-il laisser l'étudiant s'enfoncer dans son erreur ou continuer sur une fausse piste?. Le moment de faire intervenir l'aide est-il arrivé?

-Le dernier élément est une interface qui doit être très conviviale. Elle reçoit les informations de l'élève autre que par la frappe qui est un moyen peu conviviale. Ces autres moyens peuvent être: la souris, l'écran tactile, la reconnaissance vocale, le crayon optique,... .

Pour qu'un tel système puisse exister, il est nécessaire qu'une analyse précise des procédés et stratégies pédagogiques soit effectuée. Et comme il est tout à fait probable que définir le tuteur universel soit du domaine de l'utopie, il convient de travailler discipline par discipline et de procéder à l'audition de ceux qui enseignent avec une certaine réussite.

Les formes plus classiques d'enseignement par ordinateur ne sont pas pour autant condamnées à disparaître. Un certain nombre de difficultés majeures limiteront encore longtemps la diffusion à une grande échelle des dispositifs se réclamant de l'intelligence artificielle. D'une part, les méthodes de structuration des données sur lesquelles ils reposent sont fortement liées aux contenus enseignés. D'autre part, les coûts de développement et d'exploitation de ces dispositifs rendent peu probables à moyen terme une utilisation qui dépasserait le cadre d'expériences ponctuelles. Les logiciels munis de la capacité d'analyse du langage que nous avons évoqués plus haut ne peuvent être utilisés sur des ordinateurs de petite taille mémoire qui sont les ordinateurs présents dans les établissements, mais bien par une grosse machine avec un ensemble de terminaux qui pourrait supporter de tels logiciels.

L'IA utilisée dans le domaine des logiciels d'enseignement peut apporter des améliorations, mais ne pourra pas résoudre tous les problèmes que connaît l'enseignement. Il n'existe pas une méthode universelle de pédagogie, d'enseignement qu'elle soit appliquée au moyen de l'informatique ou par tout autre moyen.

2.7. Conclusions

Nous avons introduit certains facteurs ayant une influence sur l'apprentissage. Cette liste n'est pas exhaustive mais nous avons pris les facteurs qui nous semblaient être ceux dont l'influence est la plus importante sur l'apprentissage. La réalisation de tout didacticiel nécessite la

prise en compte de ces facteurs. De nombreuses études sur ces derniers montrent que l'adaptation des situations d'apprentissage aux caractéristiques individuelles devrait constituer un objectif prioritaire. Pour cette raison. il est utile de s'intéresser au modèle de l'élève et à celui du domaine.

L'apprenant sera caractérisé à partir d'un certain nombre de paramètres dont la conjonction constituera le *modèle de l'élève* . Ce modèle sera plus ou moins élaboré selon le nombre et surtout la pertinence des variables qu'il mettra en oeuvre. Il dote le système d'un ensemble d'informations sur l'élève tant à l'intérieur qu'à l'extérieur du système de formation.

Le *modèle du domaine* fournira une représentation plus ou moins sophistiquée de la matière concernée. C'est à partir de cette formalisation des contenus à enseigner que l'organe de décision choisira les objets d'apprentissage qui seront proposés à l'apprenant. Les choix offerts à l'organe de décision dépendront largement des techniques de représentation des contenus mises en oeuvre.

Ces deux modèles feront l'objet des chapitres suivants.

Chapitre 3 : Le modèle de l'élève

Introduction

Nous venons de décrire un système d'enseignement adaptatif dans le chapitre précédent. Nous allons maintenant développer une des composantes d'un tel système : le modèle de l'élève. Nous donnerons une définition de la population ayant une déficience mentale. Cette définition reposera sur deux aspects : le niveau intellectuel et l'adaptation sociale. Nous énoncerons également les différents prérequis nécessaires à l'utilisation du logiciel. Nous tenterons ensuite à la lumière de la théorie piagétienne du développement de l'enfant de montrer que les notions requises pour l'apprentissage du corps interne et de son fonctionnement, dégagés dans le chapitre précédent, sont à la portée cognitive des personnes handicapées mentales légères et modérées. Pour ces dernières nous mettrons en lumière quelques problèmes de compréhension qui pourraient intervenir dans l'utilisation du logiciel.

Notre didactitiel s'adresse à une population particulière, les personnes handicapées mentales légères et modérées.

3.1. Le handicap mental

3.1.1. Le niveau intellectuel

CLARKE et CLARKE [1975] [cités dans LAMBERT, 1986] ont analysé de nombreuses définitions de l'arriération mentale et ont conclu: "devant l'hétérogénéité des arriérés mentaux concernant l'étiologie, le fonctionnement et le pronostic, la faiblesse intellectuelle est peut-être la seule chose que ces individus ont en commun.

Certains auteurs assimilent un enfant retardé à un enfant dont l'âge chronologique est supérieur à l'âge mental défini par des épreuves. C'est ainsi que le Quotient Intellectuel a été introduit. C'est le quotient de l'âge mental sur l'âge chronologique . La limite supérieure de la déficience mentale est fixée à 70 dans les épreuves de BINET et SIMON. Ce chiffre, loin d'être fixé par décret, est la traduction de certaines exigences scolaires et sociales. Ce stade correspond à l'acquisition de la lecture et de l'écriture chez un enfant normal (8 ans) et pas à la pensée abstraite (15 ans). C'est donc un critère pédagogique qui définit le QI et non l'inverse.

3.1.2. L'adaptation sociale

Outre les lacunes intellectuelles, les arriérés mentaux présentent des troubles plus ou moins importants de l'adaptation sociale. C'est une autre de leurs grandes caractéristiques. On adopte, ici, comme critère de classification des arriérés, leur incompétence à vivre de manière autonome à l' âge adulte.

3.1.3. Définition de l'arriération mentale

Une définition couramment utilisée est celle offerte par l'American Association on Mental Deficiency (AAMD) en 1973 "l'arriération mentale se réfère à un fonctionnement intellectuel général significativement inférieur à la moyenne, existant parallèlement à des troubles adaptatifs et se manifestant à la période développementale". On remarque dans cette définition , deux critères dans l'évaluation du handicap, niveau intellectuel et adaptation sociale.

3.1.4. Classification de l'arriération mentale

Nous parlerons ici de personnes handicapées mentales légères et modérées classifiées selon le QI comme suit:

léger (+/- 50-55 < QI < +/- 70-75)
modéré (+/- 35 < QI < +/- 50-55)
sévère (+/- 20-25 < QI < +/- 35)
profond (0 < QI < +/- 20-25).

Cette classification est largement utilisée bien qu'elle soit dangereuse car elle ne porte que sur un critère, le QI. Il nous faut encore insister sur la grande hétérogénéité de la population des handicapés mentaux nécessitant une approche individuelle réalisable par un didactitiel suffisamment souple.

3.2. Les variables entrant en ligne de compte

Le modèle de l'élève est constitué d' informations concernant la structure cognitive et/ou affective qui caractérise l'élève. Ces dernières sont nécessaires afin de mettre en oeuvre des possibilités adaptatives offertes par l'enseignement par ordinateur. Une des caractéristiques qui nous intéresse particulièrement est la maîtrise des prérequis spécifiques à l'interaction.

3.2.1. Niveau de maîtrise des prérequis d'interaction

Les préprequis retenus pour utiliser le logiciel sont les suivants:

-compréhension de la consigne, savoir écouter et regarder
-maîtrise de la relation de cause à effet: comprendre qu'un ordre donné engendre tel effet dans telles circonstances.
-être capable de déplacer un mobile: orientation spatiale dans un plan
-savoir lire les symboles et leur associer une signification
-savoir reconnaître son nom au clavier ou savoir l'écrire en utilisant les touches du clavier
-avoir participé aux séances d'animation préalables sur la connaissance du corps.

L'utilisation de la souris est utile mais ne constitue pas un prérequis, car nous avons pu constater que la période d'apprentissage de cet outil est relativement brève chez les personnes possédant les prérequis cités.

3.2.2 Aptitudes intellectuelles demandées

Deux notions spécifiques à la matière enseignée dans le cadre de notre logiciel méritent qu'on s'y attarde. Il s'agit de la notion d'*ensemble*, généralisation de la notion de système que nous utilisons, et la notion de *parcours*, modélisant le trajet d'une substance dans le corps. Il serait intéressant de resituer les personnes handicapées mentales par rapport à ces notions que nous avons introduites dans les chapitres précédants. Cependant, la littérature est pauvre dans ce domaine, et plus généralement en ce qui concerne les connaissances des handicapés mentaux. C'est pourquoi, nous optons pour une démarche basée sur un parallèle fait avec le développement cognitif de l'enfant.

développement de l'enfant

Dans le développement de l'enfant décrit ci-après, nous nous limiterons aux aspects cognitifs. PIAGET [1966] découpe le développement de l'enfant en quatre stades ou périodes.

1. Stade sensori-moteur: caractérisé par l'absence de logique et d'objectivité.
2. Stade pré-opératoire: (de 2 à 7 ans) pensée intuitive et représentation symbolique.
3. Stade opératoire concret: (de 7 à 10 ans) apparition de l'objectivité, maîtrise de la logique des relations et des transformations sur matériel visible.
4. Stade opératoire formel: (après 11 ans) existence d'une objectivité et d'une logique détachée du concret.

Chaque stade exprime une nouvelle adaptation de l'individu à son environnement. D'après PIAGET [1966] le comportement intelligent réside dans la capacité à s'adapter.

Au *stade opératoire*, l'enfant peut trouver des moyens nouveaux, non plus seulement par essais et erreurs dans ses actions,mais par une réflexion qui lui fait découvrir le moyen adéquat pour atteindre un but déterminé. C'est donc par pensée qu'il combine les moyens qu'il connait et le but à atteindre jusqu'à trouver la combinaison efficace. Cette logique n'est pas encore opératoire. La coordination s'ébauche ,mais n'atteind pas la réversibilité. Selon WALLON [1987], l'enfant utilise une "pensée en îlots", c'est-à-dire fragmentaire, constituée d'éléments juxtaposés. Il est dés lors incapable d'intégrer dans une même idée deux aspects d'un même objet (exemple: blé et farine, soleil et jour).

Le *stade opératoire* s'acquiert au moment où l'on peut comprendre qu'un objet peut rester le même tout en changeant d'aspect. Vers 8 ans, l'enfant dégage ses premières relations causales: parvenu au stade logique, il anticipe les différents états ou transformations d'un système et réfléchit à leurs causes. Des coordinations entre des actions telles que réunir, ordonner, sérier dans l'ordre croissant ou décroissant, manipuler les relations entre les différents états d'un système ou ses divers éléments, sont possibles.

Les caractéristiques de la *logique formelle* sont le raisonnement sur des phrases, des signes ou des représentations. Le raisonnement va se détacher des données concrètes. Vers l'age de 12 ans, il s'appliquera à des données symbolisées et formalisées: énoncés verbaux,signes logiques ou mathématiques, représentations intériorisées (" images mentales").
Idéalement, le raisonnement formel permet des manipulations dissociées de tout support, donc aisées quel que soit leur contenu. Nous l'avons vu, l'enfant de 7-8 ans parvient à sérier méthodiquement des éléments par hauteur décroissante ou croissante. Or si le même problème est présenté verbalement ("Edith est plus blonde que Suzanne, et plus foncée que Lilli"), il ne sera résolu qu'à partir de 11 ans. Il appartient au niveau formel. C'est à ce stade que la réelle abstraction est possible.

Il nous reste à situer la personne handicapée mentale parmi ces stades.

Rapport entre handicap mental et développement cognitif

INHELDER [citée par LAMBERT, 1987] fut la première à proposer un schéma de développement cognitif des arriérés mentaux. Bien que les recherches en ce domaine restent rares, plusieurs études différentes réalisées par des auteurs comme WOODWARD, LOVELL ont confirmé ce schéma de développement basé sur le schéma de développement des enfants élaborés par PIAGET [1966]. Les arriérés profonds et sévères restent fixés aux différents stades de l'intelligence sensori-motrice. Les arriérés modérés sont incapables de dépasser la période pré-opératoire. Les arriérés légers peuvent se caractériser par une fixation au stade des opérations concrètes. Deux notions interviennent dans l'ouvrage d'INHELDER, la fixation et la viscosité. La première implique à la fois la déduction de la vitesse de développement et le maintien à un stade déterminé sans possibilité de progrès ultérieurs. La viscosité se caractérise par une réapparition des schèmes antérieurs dans le fonctionnement cognitif actuel d'un sujet. Les arriérés mentaux restent plus longtemps que les personnes normales à un stade de transition entre deux périodes développementales. La notion de fixation cognitive est actuellement remise en question. Plusieurs auteurs, KIRK [1968], TRUEN [1973] et STEPHENS et Mc LAUGHLIN [1964] [cités dans LAMBERT, 1987] montrent dans leurs études encore au stade descriptif qu'il existe un décalage d'environ deux ans entre les performances des sujets arriérés mentaux et celles de normaux appariés sur base de l'âge mental dans les tâches piagétiennes des stades pré-opératoire et concret. L'apprentissage spécifique de certaines tâches peut entraîner non seulement une amélioration des performances, mais également permettre d'atteindre un stade cognitif supérieur, à condition que les prérequis soient installés chez le sujet.

Niveau cognitif requis

Le lien entre les stades cognitifs de développement et la population cible étant fait, on peut remarquer que les techniques utilisées par notre logiciel, basées essentiellement sur le dessin s'efforcent de maintenir la capacité cognitive requise au niveau du stade opératoire. Le raisonnement demandé à l'élève s'effectue sur des données concrètes telles

que l'image (elle joue un rôle important dans la logique opératoire) et les feed-backs réguliers aux actions de l'apprenant. Tout ceci permet le non recours à l'utilisation de concept.

De ce schéma de développement cognitif, nous pouvons déduire théoriquement qu'il pourrait exister certains problèmes dans l'utilisation du logiciel pour la population handicapée modérée qui ne dépasse généralement pas le stade pré-opératoire.

Un point de la matière que l'on se propose d'enseigner pouvant présenter un problème pour les personnes handicapées mentales modérées est le trajet de l' oxygène dans le corps. L'oxygène passe par la trachée avant de se diviser pour entrer dans les deux poumons. Bien que représenté graphiquement, le changement d'aspect (division) de l'oxygène peut présenter des difficultés de compréhension pour ce type de personnes n'ayant en théorie pas atteint le stade opératoire et donc pas acquis la capacité de comprendre le maintien de l'invariance au travers de transformations diverses. Elles peuvent penser que l'air qui passe dans les poumons n'est pas l'air qui est passé dans la trachée. Le passage de la boulette de viande se désagrégeant dans le système digestif pour passer dans le sang est un autre point, parmi d'autres, qui pourrait être non compris.

Il faut remarquer cependant que l'appartenance d'un individu handicapé mental à un stade de développement défini ci-dessus peut ne pas être rigoureusement déterminé. Cette difficulté s'explique par le fait que l'usage des tests piagétiens (destinés à déterminer le stade de développement) ne prennent pas en considération des variables comme l'adaptation sociale qui entre dans la définition des personnes handicapées mentales que nous avons donnée. Le passage d'un état de développement à l'autre se fait progressivement. Nous touchons ici à la différence pouvant exister entre la théorie et la pratique. C'est pourquoi il serait intéressant de considérer le point de vue expérimental de l'existence ou non de ce type de problèmes, dont nous venons de parler, chez les personnes handicapées mentales modérées.

3.2.3 Le niveau de maîtrise des objectifs assignés au cours

Il est intéressant de s'interroger sur les connaissances que peuvent avoir les personnes handicapées mentales de la matière que nous nous engageons à enseigner. Le mémoire réalisé par HUBENS et LECRON [1990] nous a pour cela été d'un très grand secours car la littérature n'est pas riche en ce qui concerne la connaissance, l'utilisation et l'image du corps chez les personnes handicapées mentales.

Ce mémoire comprend une évaluation de l'état de la connaissance et le vécu du corps chez le handicapé mental. Parmi les thèmes principaux envisagés, nous nous intéresserons plus particulièrement à la connaissance théorique qu'il a de la structure de son corps et de son fonctionnement. L'évaluation s'est déroulée dans cinq institutions d'hébergement mixtes pour travailleurs en Belgique. L'échantillon comprend 21 hommes et 19 femmes, dont 20 personnes de handicap modéré et 20 personnes de handicap léger. Un échantillon comparatif de personnes non handicapées mentales fut constitué. L'évaluation s'est déroulée via des questionnaires, des dessins, des puzzles. De cette évaluation nous pouvons retirer les enseignements suivants.

connaissance anatomique

* reconnaissance

Dans la reconnaissance des systèmes, seul le système circulatoire à été reconnu par plus de la moitié de l'échantillon des personnes handicapées (65%). Par ordre décroissant, on retrouve plus d'un tiers des personnes qui reconnaît le système digestif, le système respiratoire, et le système sexuel. Le système rénal n'a pas été reconnu du tout. Le système nerveux a été très peu reconnu (2,5% des handicapés et 53% des normaux).

* localisation

En ce qui concerne la localisation, 86,8% des handicapés n'ont pas localisé correctement le système nerveux. Les fréquences les plus élevées de mauvaise localisation se retrouvent en parallèle chez les handicapés et les normaux. Le système rénal a été notamment mal localisé par une grande partie des handicapés (92,1%) et des normaux (53,3%). Les systèmes les mieux situés sont les systèmes circulatoire, digestif et sexuel.

* fonctionnement

On retrouve les mêmes fréquences élevées d'échec pour le fonctionnement que pour la reconnaissance et la localisation des systèmes rénal et nerveux.

* conclusions

On constate qu'il existe une différence entre les normaux et les handicapés tant sur le plan de la reconnaissance que sur le plan de la localisation et du fonctionnement. Les deux distributions se ressemmblent pourtant. Aux trois niveaux, le système circulatoire est celui qui obtient le meilleur score chez les personnes handicapées mentales (plus de 60%). Les systèmes digestif et sexuel viennent ensuite. Les taux plus grands d'échec pour les trois critères sont le système nerveux et le système rénal.

Ces renseignements sont utiles pour proposer une suite d'exercices dont la difficulté va crescendo en accord avec les principes de progressivité énoncés au chapitre 2.

Connaissance physiologique.

Ces connaissances concernent la désignation sur trois silhouettes du

a) trajet d'une boulette de viande lorsqu'elle est avalée

b) trajet d'une gorgée d'eau

c) trajet de l'air.

* le trajet d'une boulette de pain

On peut signaler que le point de départ de la boulette de viande est connue par tout l'échantillon. Par contre seulement 40% des handicapés désigne la nourriture comme arrivant au niveau de l'anus. Aucune personne n'est consciente du phénomène de diffusion de la nourriture dans le sang, bien qu'il ait conscience qu'une absorption importante de nourriture entraîne des rondeurs. Par contre chez les normaux, la totalité des personnes ont parlé de la filtration et de la diffusion dans le corps par l'intermédiaire de la circulation sanguine.

* le trajet d'une goutte d'eau

On retrouve les mêmes caractéristiques que pour le trajet de la boulette concernant le point de départ et le point d'arrivée. Bien que 30% seulement soient conscients que l'eau se répand dans tout le corps, ils ne connaissent pas le phénomène de filtration dans le sang. Par contre, ce phénomène est connu par les personnes normales.

* la respiration

En ce qui concerne le point de départ de l'air, aucun problème pour les personnes normales. Par contre 15% des handicapés pensent que l'air ne démarre ni du nez, ni de la bouche, ni de la gorge. Certains donnent comme point de départ le nombril ou la peau. Pour 41%, l'air va jusqu'aux poumons avant de ressortir directement par le nez. Aucun n'émet l'idée qu'il se répand dans le corps. Il faut remarquer que 88,2% des normaux ignorent

que l'air se répand dans tout le corps par l'intermédiaire des globules rouges. Ceci est sans doute dû au fait que respirer est un acte automatique.

* conclusions

Les lacunes sur les trajets de ces substances sont plus importantes que les lacunes sur la reconnaissance des systèmes ou des organes.

Chapitre 4 : Le modèle du domaine

Introduction

Ce chapitre a pour but de décrire une autre composante d'un système interactif décrit dans le chapitre 2 à savoir le modèle du domaine. Nous l'appliquerons à la connaissance du corps interne et à son fonctionnement. Nous insisterons sur deux notions requises pour enseigner cette connaissance.

4.1. Le domaine

La matière particulière que nous désirons enseigner à l'aide de notre didactitiel se rapporte à la connaissance et au fonctionnement du corps, et plus précisément, à la connaissance des organes internes et des mécanismes d'alimentation, de circulation du sang, de respiration, d'excrétion, etc... Cette connaissance correspond à une base minimale dans la connaissance du corps et son fonctionnement afin de comprendre les grands mécanismes qui régissent le corps. L'apprentissage doit partir de la base et être extrêmement simple car il est destiné à combler des lacunes dans ce domaine de connaissance. La matière enseignée se veut relativement simple, également, afin de rester dans les préoccupations de tous les jours.

4.2. Structuration de la connaissance

La représentation de ces connaissances repose, entre autres, sur la notion d'agrégation. Les organes internes sont groupés en systèmes anatomiques. Cette approche quelque peu controversée est opposée à l'approche qui consiste à prendre l'organe seul comme base d'enseignement.

L'organe est en effet une notion plus simple que le système, apparemment plus accessible et mieux connue. Cependant, c'est la structuration des organes du corps humain en systèmes qui a été retenue pour diverses raisons. Cette approche par système est l'approche pédagogique utilisée lors de l'apprentissage des organes. Elle se justifie par le fait que tous les organes d'un système concourent à une même fonction (le système digestif est responsable de la digestion). De plus il est difficile de parler d'un organe et de sa fonction isolément (l'estomac seul ne digère pas les aliments). Ainsi l'unité la plus petite que l'on considère est le système.

La découpe en système reflète une volonté de structuration de la connaissance. Cette structuration de la matière est intéressante lorsqu'on s'adresse à des sujets de faible capacité cognitive (tel est le cas des handicapés mentaux légers et modérés). Une étude menée à l'université de LEEDS [citée dans DEPOVER, 1987] afin de comparer l'efficacité de plusieurs programmes d'enseignement par ordinateur a permis de mettre en évidence le fait que le manque de structuration d'un cours se traduit généralement par un niveau d'efficacité global moins élevé, mais aussi par le fait que la chute des résultats est surtout sensible chez les sujets les plus faibles.

Etant donné que le domaine des connaissances à enseigner doit partir des prérequis du sujet, dont nous parlerons dans le chapitre suivant, il nous paraît intéressant de réaliser un exercice préliminaire basé sur les organes principaux de chaque système, généralement mieux connus, pour ensuite déboucher sur l'approche par systèmes. La connaissance d'organes principaux constituera donc le point d'ancrage nécessaire à l'apprentissage. Cet exercice préliminaire réconcilie quelque peu les deux approches antagonistes

La dimension dynamique du fonctionnement du corps humain est assurée par la connaissance du parcours de certaines substances comme l'oxygène, une boulette de viande, l'eau, le sang dans le corps. L'importance de la notion de parcours réside dans ses composantes de: point de départ, point d'arrivée, sens,endroit de passage (organes de passage) et continuité du parcours.

4.3. Support de la représentation

Le transfert de connaissance sera effectué sur une base, des dessins de silhouettes humaines représentées avec des organes internes constituant certains systèmes ou parcours possibles. On comprend volontiers qu'une grande importance est ici attachée à la représentation graphique de la silhouette.

Le choix du graphisme s'est élaboré au fil de nos rencontres lors de notre séjour en Suisse et dans les institutions belges et en collaboration avec J. DELVILLE et J. L. COLLIGNON du département de psychologie. La première base pour les graphiques que nous avions était constituée par les graphiques en grandeur nature des systèmes du corps humain du dossier "Migration santé" "Guide de l'animateur" [BARTHE, BRIERE, LEGENDRE ET PECHOIN, 1984] utilisé à des fins pédagogiques, ainsi que plusieurs ouvrages de médecine et biologie. Les premières propositions que nous avons faites, étaient des dessins où seuls figuraient les contours de la silhouette et des organes. Il leur a été préféré des dessins d'un style plus attractif, se rapprochant du genre dessin animé, avec des organes colorés. Les futurs utilisateurs sont intervenus également dans les choix des graphiques. Nous avons proposé à six personnes handicapées mentales légères et modérées ces différents type de dessins. La totalité des personnes exprimait nettement sa préférence pour les dessins du genre dessin animé. Cette préférence se traduisait par un attrait plus grand et par une meilleure compréhension des dessins.

Un point important sur lequel les personnes qualifiées dans le domaine du handicap mental et de la pédagogie ont insisté, est de proposer des représentations de systèmes très simplifiées et des organes schématisés, tout en restant identifiables, plutôt que de faire preuve d'une grande précision et de surcharger et compliquer le dessin. La quantité d'information transmise doit correspondre à une base minimale dans la connaissance du corps. Il n'est pas question ici de noyer l'utilisateur, qui présente déjà des difficultés cognitives comme nous l'avons vu, par un flux d'informations qu'il ne pourrait pas maîtriser. Une grande rigueur a pourtant été accordée aux emplacements des organes et à la configuration des systèmes.

La couleur a été utilisée pour faciliter l'apprentissage. En effet un ton de couleur différent a été associé à chaque système, de ce fait, une association consciente ou inconsciente peut être faite par l'utilisateur entre un ton de couleur et un système. Elle constitue une aide supplémentaire à l'élève qui peut également associer un organe à un système, plus facilement, par le biais de la couleur.

4.4. Nature du domaine

Les différents systèmes qui sont envisagés et leur composition sont les suivants :

système digestif	bouche
	gorge
	oesophage
	estomac
	intestins
	anus
	vésicule
	foie
système nerveux	cerveau
	nerfs
	moelle épinière
système reproducteur masculin	testicules
	pénis
	prostate
système reproducteur féminin	ovaires
	utérus
	col
	vagin
	vulve

système excréteur féminin	reins vessie vulve
système excréteur masculin	reins vessie pénis
système circulatoire	coeur veines artères petits vaisseaux
système musculaire	muscles abdominaux muscles de la cuisse biceps tendons
système respiratoire	nez bouche gorge trachée bronches poumons
squelette	crâne côtes colonne vertébrale bassin articulation du genoux articulation de l'épaule articulation du coude articulation de la hanche articulation de la cheville articulation du poignet

systèmes digestifs et circulatoire	bouche
	gorge
	oesophage
	estomac
	intestins
	anus
	veines
	artères
	petits vaisseaux
systèmes respiratoire et circulatoire	nez
	bouche
	gorge
	trachée
	bronches
	poumons
	coeur
	veines
	artères
	petits vaisseaux
systèmes excréteur et circulatoire	reins
	vessie
	pénis (vulve)
	coeur
	veines
	artère

Un contenu plus précis de la matière à enseigner sera développé dans le chapitre "spécification de l'outil" avec la description des exercices. Les prérequis de l'apprenant par rapport à cette matière ont été discutés dans le chapitre précédent.

Chapitre 5 : La genèse du projet

Introduction

Le but de ce chapitre est de présenter le contexte dans lequel s'inscrit notre logiciel. Nous en donnerons l'énoncé initial ainsi que les modifications qui y furent apportées dans le courant de notre stage. Nous dirons un mot des acteurs en présence ainsi que de leur implication.

5.1. Le contexte

Le logiciel s'inscrit dans un projet élaboré par un groupe pluridisciplinaire de travail. Ce groupe est composé de représentants de différentes institutions belges d'accueil pour personnes handicapées, de représentants du Département de Psychologie des Facultés Universitaires Notre-Dame de la Paix (FUNDP) ainsi que du PSINHA (Psychologie-Informatique-Handicap), dont le but est de promouvoir l'utilisation de l'ordinateur comme outil d'apprentissage, et de parents d'enfants handicapés mentaux. Ce groupe composé donc de médecins, de psychologues, de logopèdes, d'éducateurs, de biologistes, de spécialistes du handicap se sont penchés sur la manière d'élaborer des techniques, des moyens pour améliorer la connaissance du corps humain. Ils se sont aperçus que dans les institutions les moyens d'apprentissage classiques (dessins, découpages dans les magazines, pâte à modeler,...) ne s'avèrent pas particulièrement adaptés aux adultes vu à leur caractère infantile. C'est pourquoi ce groupe préconise l'emploi d'autres moyens tels le film vidéo, les programmes informatiques. Notre travail naquit de cette préoccupation et enrichit la liste des thèmes traités par l'outil informatique sur lesquels ont réfléchi le groupe de travail. Le thème du corps externe (schéma corporel) a fait l'objet de l'élaboration et de la réalisation d'un logiciel d'apprentissage. Notre logiciel concerne également la connaissance du corps mais plus spécifiquement la connaissance du corps interne, des mécanismes de la digestion, de la respiration, de la circulation du sang,... .

5.2. Définition du projet initial

Le logiciel sur la connaissance du corps humain interne adapté aux personnes handicapées mentales adultes légères et modérées reçuT un premier énoncé proposé par Madame Jacqueline DELVILLE, Monsieur Michel MERCIER et Monsieur Jean-Luc COLLIGNON. Cet énoncé reprenait les types d'exerciceS qui leur apparaissaient être les plus pertinents afin de faire acquérir aux personnes handicapées mentales les connaissances nécessaires sur leur corps interne et son fonctionnement. Ces exercices se justifient par la présence de nombreuses lacunes dans ce domaine chez cette population, comme nous l'avons indiqué précédemment. Ces types d'exerciceS suggérés sont au nombre de deux :

- un exercice de type *désignation* par lequel la personne handicapée mentale serait invitée à désigner sur l'écran de la machine des organes regroupés en systèmes anatomiques et énoncés par l'ordinateur. Cet exercice de base possède des variantes : désigner l'endroit dans le corps où se localisent des organes, désigner des systèmes ou des organes dont la fonction est énoncée par l'ordinateur, les noms des organes, des silhouettes féminines, masculines, présentées de face, de dos, de profil, tous les organes coloriés d'une même couleur ou coloriés par différentes couleurs.

- un exercice de type *parcours* par lequel la personne handicapée serait invitée à désigner sur l'écran de l'ordinateur le parcours d'une boulette à travers le système digestif, le parcours de l'oxygène à travers le système respiratoire, le parcours du sang à travers le système circulatoire ainsi que le parcours d'autres substances. Ce type d'exercice consiste également à désigner le parcours de ces substances à travers plusieurs systèmes (par exemple le trajet de l'oxygène à travers le système respiratoire et le système circulatoire). Désigner le parcours d'une substance dans un ou plusieurs systèmes signifie désigner un point d'entrée, les différents organes successifs traversés par cette substance et un point de sortie. Des variantes existent également pour cet exercice de base : silhouettes féminines, masculines, présentées de face, de

dos, de profil.

Pour chacun des exercices un feed-back entraînant un renforcement positif (se référer à la définition de ces deux notions au chapitre 2) devrait être émis immédiatement après chaque réponse correcte comme nous le faisions remarquer dans le chapitre 2. En ce qui concerne l'exercice de désignation d'un système, ce feed-back pourrait consister en une animation du fonctionnement de ce système. Une animation du passage d'une substance à travers un ou plusieurs systèmes pourrait également faire l'objet du feed-back. Après chaque réponse erronée, un feed-back entraînant un renforcement négatif devrait apparaître immédiatement. La possibilité d'enregistrer les performances de l'utilisateur devrait être prévue pour l'exercice de type parcours ainsi qu'un certain degré d'imprécision devrait être toléré.

5.3. Les modifications apportées

Lors de notre période de stage en Suisse, nous avons affiné la définition du projet proposé par Madame Jacqueline DELVILLE Monsieur Michel MERCIER et Monsieur Jean-Luc COLLIGNON. Durant cette période, nous avons été amenés à travers de nombreuses rencontres, avec Monsieur Alexandre Waeber, des psychologues, des éducateurs, des moniteurs, des responsables de l'éducation à la santé des personnes handicapées mentales, d'étoffer le cahier de charge en ajoutant une variante à l'exercice de type désignation. Cette variante proposée en grande partie par Monsieur Alexandre Waeber consiste à proposer un exercice de désignation des organes principaux de chaque système (coeur, estomac, poumons, reins, cerveau). Cette variante s'inscrit dans une approche d'apprentissage des organes du corps humain un peu différente de celle du département de Psychologie des FUNDP qui est une approche par système comme expliqué dans le chapitre 3. Elle est représentée par la figure 5.1.

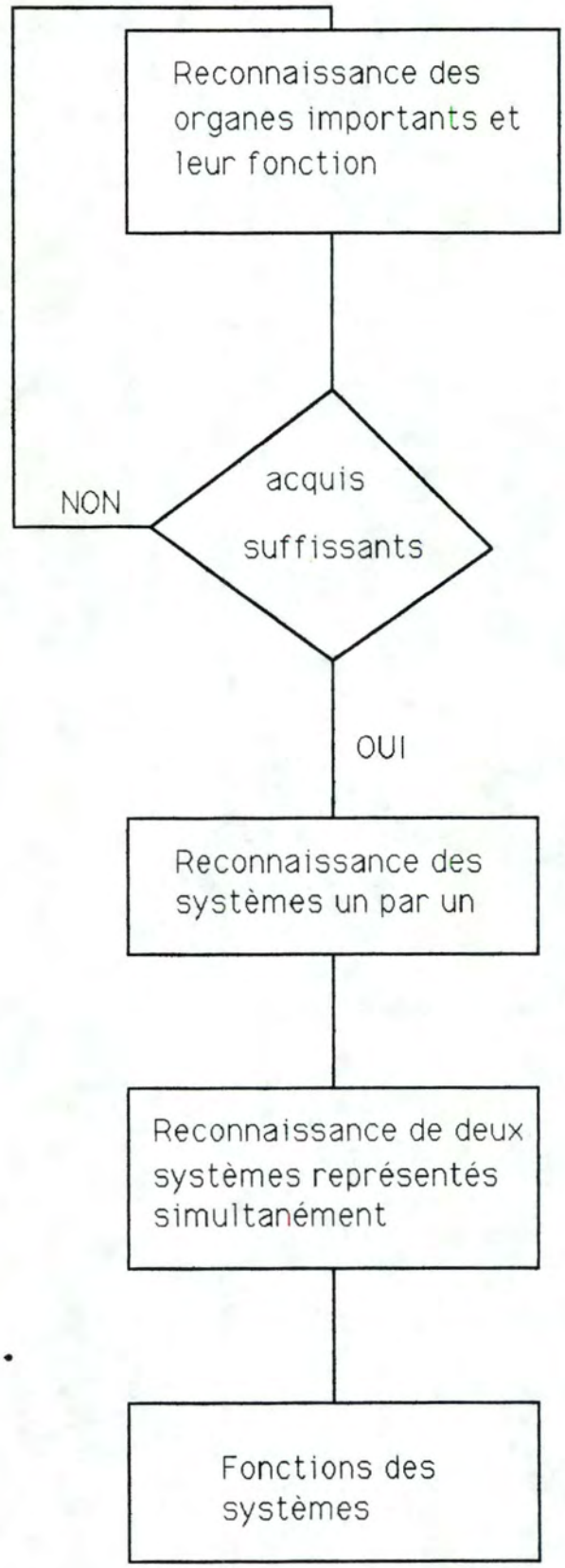


figure 5.1 : approche par organes

Cette approche se justifiant par le fait qu'il est plus facile

1. d'identifier un système par la connaissance préalable d'un organe principal de ce système

2. de commencer à apprendre les organes avant de comprendre les systèmes (connaître les chiffres avant de connaître une addition)

3. de faire découvrir ainsi la présence d'organes internes.

De plus, cet exercice peut permettre d'évaluer les connaissances des personnes sur ces organes les plus connus et, à partir de ces renseignements, de proposer à chaque personne individuellement une suite d'exercices sur les systèmes qui semblent mal connus. Cet exercice permet donc d'évaluer les prérequis de la matière. L'implémentation de cet exercice fut décidé en accord avec le Département de Psychologie des FUNDP. La variante de l'exercice de désignation d'un organe dont la fonction est énoncée par l'ordinateur fut abandonnée à cause de la difficulté de préciser la fonction d'un organe.

5.4. Conclusions

Nous venons de donner les origines du logiciel sur la connaissance du corps humain interne ainsi que des acteurs qui y ont participé. Avant de passer aux spécifications détaillées de l'application, nous allons présenter la démarche de développement que nous avons suivie.

Chapitre 6 : La démarche de développement

Introduction

Dans ce chapitre nous allons expliquer quelle fut notre démarche de développement du logiciel. Nous donnerons dans une première section des types de modèle d'architecture pour la réalisation de systèmes interactifs. Nous soulignerons les qualités d'une "bonne" interface et expliquerons nos choix pour la réalisation de l'interface du logiciel. Ensuite nous identifierons les acteurs qui ont participé à la phase de spécification.

6.1. Le projet : une application interactive

Le projet que nous développons dans ce mémoire consiste en la réalisation d'une application interactive réalisant ses fonctionnalités à travers une communication importante avec l'utilisateur. Un système interactif peut se représenter par des modèles d'architecture.

J. COUTAZ en souligne l'importance dans le cadre des systèmes interactifs en vue de faciliter leur réalisation [COUTAZ, 1989]. Elle met en évidence que le problème de la construction d'un système interactif se situe dans un espace à quatre dimensions : l'application, l'utilisateur, l'environnement de travail et les niveaux de communication. A chaque dimension correspondent des propriétés intrinsèques et des besoins à satisfaire. Le tableau de la figure 6.1 donne quelques exemples. Nous relevons, entre autres, que l'utilisateur est caractérisé par des limites sensori-motrices et cognitives et qu'il doit être guidé.

domaine de définition	APPLICATION	BESOINS Gestion des erreurs Portabilité Maintenance Lancement Terminaison,etc. CARACTERISTIQUES Centralisation/Distribution des ressources Multi/Mono utilisateur Multi/Mono tâche Multi/Mono espace de données Interaction avec d'autres applications Temps réel/traitement par lots, etc.
	UTILISATEUR	BESOINS Droit à l'initiative Protection contre les erreurs Guidage, etc. CARACTERISTIQUES Niveaux d'expertise Limites sensorimotrices et cognitives, etc.
	ENVIRONNEMENT	CARACTERISTIQUES Bruit, confort Dispositifs physiques d'interaction Procédures administratives, etc.
	NIVEAUXDE COMMUNICATION	CARACTERISTIQUES Lexicales Syntaxiques Sémantiques BESOINS Cohérence Concision, etc.

figure 6.1 : problèmes de construction d'un système intercatif

Une solution à ce problème de construction de systèmes interactifs est l'élaboration et l'utilisation de modèle d'architecture dont l'objet est de "fournir au réalisateur une structure générique à partir de laquelle il est possible de construire un système interactif particulier. Cette structure décrit le flux des données entre l'utilisateur et l'application : elle identifie les étapes de transformation des données et détermine l'agencement des composants qui assurent ces transformations. Ces composants sont des abstractions qui correspondent à des processus généraux : ils sont indépendants des techniques de réalisation" [COUTAZ, 1989]. C'est la nature des composants, leur arrangement et les transformations qu'ils opèrent qui sont en fait la solution au problème de construction soulevé.

De tels modèles ont divers objectifs. L'un d'eux est de fournir une autonomie entre les fonctions de l'application et celles du dialogue. Remarquons toutefois que cet objectif n'est pas toujours souhaitable dans les cas où la représentation présente une sémantique. Un autre objectif est d'offrir une indépendance vis-à-vis du système, de l'outil de réalisation. Une architecture en couches permet cette indépendance. Un troisième objectif est la possibilité de réutiliser les composants.

Nous allons mentionner trois types de modèles d'architecture qui offrent chacun une vue complémentaire sur la façon d'organiser les éléments constitutifs d'un système interactif.

6.1.1. Modèles d'architectures pour systèmes interactifs

6.1.1.1. Le modèle LANGAGE

Les modèles d'architecture de type LANGAGE s'inspirent de l'analogie entre l'interaction Homme-Machine et les dialogues entre individus. De la même manière que deux personnes communiquent par un même langage, l'utilisateur et le système communiquent par l'intermédiaire d'un langage commun. A l'image des modèles linguistiques, le langage utilisé par le système et l'utilisateur se définit sur trois composants. Le

premier composant est le composant *sémantique.* Ce dernier définit la signification des phrases, décrit précisément les classes d'objets auxquelles l'utilisateur et le système font référence dans leur conversation. Le deuxième composant est le composant *syntaxe* qui définit la construction des phrases du langage à partir des éléments syntaxiques. Un élément syntaxique est une unité qui ne peut être décomposée plus avant sans perdre sa signification. Le dernier composant est le composant *lexique* qui définit la production des unités syntaxiques à partir d'un vocabulaire. Dans le cas particulier du langage Homme-Ordinateur, ce vocabulaire comprend tout ce qui est possible de créer à partir des dispositifs d'entrée et de sortie.

6.1.1.2. Le modèle ENTREE/SORTIE

Les modèles d'architecture d'entrée/sortie mettent l'accent sur l'échange et la transformation d'informations. La forme des échanges et la nature des transformations définissent le découpage fonctionnel du système. Ce découpage peut s'envisager comme un empilement de machines abstraites effectuant chacunes des opérations d'entrée/sortie et les traitements à un niveau d'abstraction donné. Les actions physiques de l'utilisateur remontent la hiérarchie en étant transformées à chaque étape jusqu'à devenir des notions abstraites interprétables par l'application. Dans le sens inverse, les concepts de l'application descendent la hiérarchie en étant traduits en des concepts reconnaissables pour l'utilisateur jusqu'à la machine physique.

6.1.1.2.Le modèle MULTIAGENT

Les modèles multiagents représentent les systèmes interactifs comme un ensemble d'agents coopérants. Un agent est un système de traitement de l'information et un interlocuteur vis-à-vis d'autres agents, y compris l'utilisateur. Ce type de modèle se base sur des systèmes stimuli-réponses (un événement provoque une réponse du système). Ce type de modèle permet la conception itérative des systèmes interactifs. Un agent est vu comme un agent de modularisation. Le parallélisme est également réalisable car plusieurs agents peuvent être à l'écoute d'événements et produire d'autres événements en réponse.

Le modèle PAC (Présentation Abstraction Contrôle) élaboré par J. Coutaz [1989] en est un exemple. Ce modèle structure l'architecture d'un système interactif en trois constituants : la Présentation, l'Abstraction et le Contrôle (figure 6.2).

-La Présentation définit l'Image du système, i.e. son comportement en entrée et en sortie.

-L'Abstraction désigne les concepts et les fonctions du système.

-Le Contrôle maintient la cohérence entre les facteurs Présentation et Abstraction.

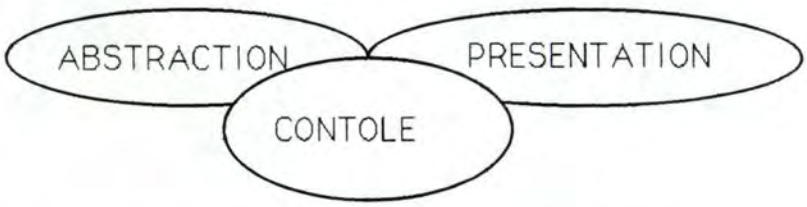


Figure 6.2 : Les composants du modèle PAC

La présentation de PAC est réalisée par un ensemble d'agents spécialisés dans l'interaction avec l'utilisateur. Coutaz les appelle "outils interactifs". Un outil interactif se caractérise par :

-une Image qui définit un comportement perceptible (c'est la Présentation visible de l'utilisateur).

-des fonctions et des attributs fonctionnels (c'est le côté Abstrait visible des autres constituants logiciel).

-la gestion des liens entre côté Abstrait et Présentation (c'est le Contrôle). Le contrôleur assure la cohérence entre l'élément abstrait (température = 10°) et la Présentation (le dessin du thermomètre).

Un objet interactif peut être composé en plusieurs objets interactifs élémentaires.

Ce modèle est un modèle multiagent qui permet de créer à partir d'agents élémentaires (les objets interactifs élémentaires) des agents multiprocesseurs (les objets interactifs composés). Il est toujours possible de structurer un système interactif en une hiérarchie d'objets PAC. Un système interactif est un objet PAC composé (figure 6.3.)

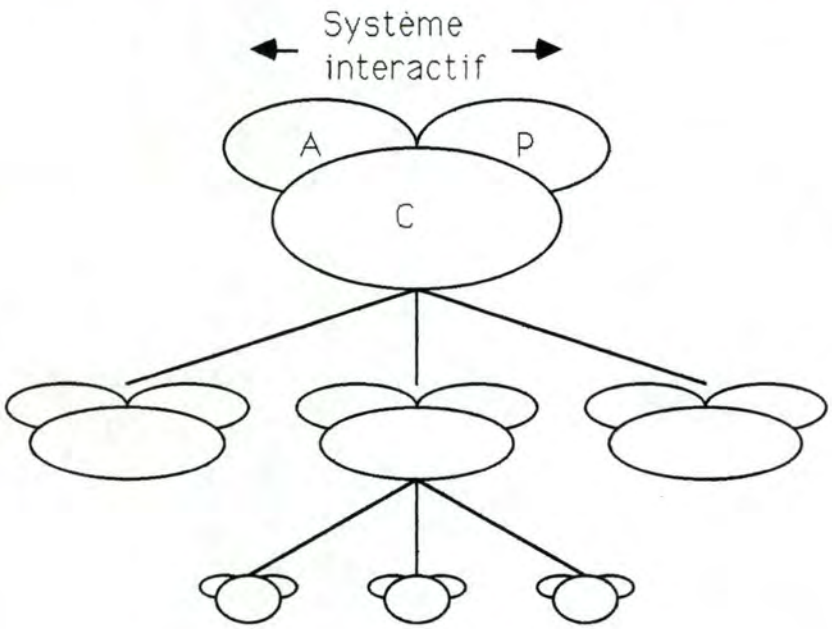


Figure 6.3 : système interactif PAC

Un objet interactif peut se concevoir comme trois coprocesseurs, l'Abstraction A, le Contrôle C et la Présentation P, dotés chacun d'une mémoire.

Nous venons de voir quelques types d'architectures logicielles de systèmes interactifs. Les modèles de type langage fournissent une explication simple du principe de fonctionnement d'un système interactif. Les notions de lexique, de syntaxe et de sémantique sont bien comprises des informaticiens. Ceux-ci peuvent s'y référer utilement et facilement pour jeter les bases structurelles du système à construire. La modularité sous-jacente au modèle, qui distingue les aspects lexicaux, syntaxiques et

sémantiques permettent la modification de la Présentation sans mettre en cause les fonctions et réciproquement. Cette possibilité permet d'apporter des modifications au système avec des coûts moins importants. Cette modularité permet également de bien dissocier les deux grandes parties : les fonctionnalités et la Présentation.

6.1.2 Le modèle d'architecture de base

Nous avons vu qu'un système interactif peut se représenter par plusieurs types d'architecture. Nous allons exposer un modèle appartenant au premier type de modèle que nous avons décrit dans la sous-section précédente : le modèle SEEHEIM.

6.1.2.1. Description

Ce modèle est le modèle de base des architectures de systèmes interactifs. Il est constitué de trois composants logiques: la PRESENTATION, le CONTROLE DU DIALOGUE et L'INTERFACE AVEC L'APPLICATION comme l'indique la figure 6.4.

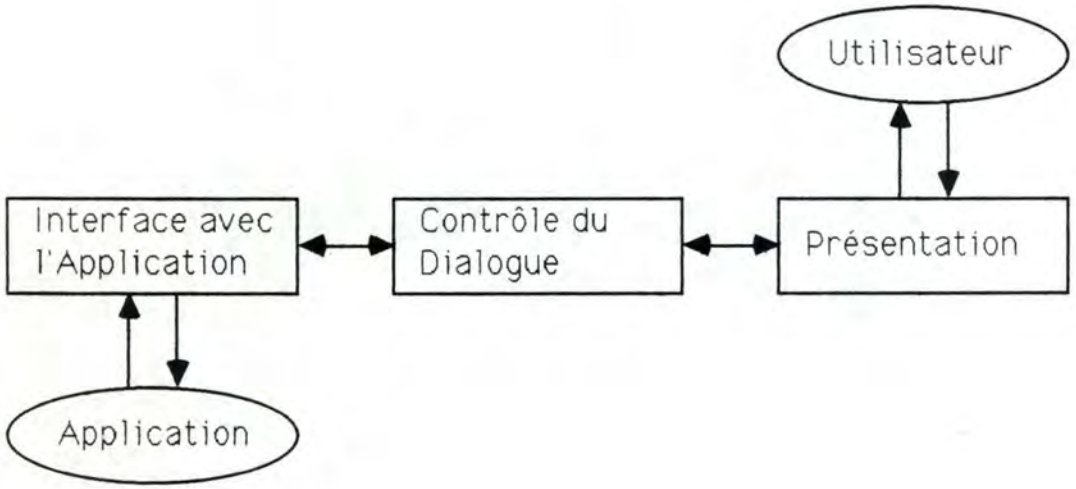


figure 6.4 : le modèle SEEHEIM

Le composant PRESENTATION est responsable de la représentation physique du système. Du point de vue linguistique, il effectue l'analyse lexicale du langage d'interaction. Il convertit les informations entre le format du monde physique externe et celui du monde informatique interne. Il traduit les objets sémantiques et syntaxiques internes (les objets informatiques) en objets interactifs (les objets visibles pour l'utilisateur). Plus précisément, la Présentation lit les données en provenance des dispositifs physiques d'entrée (ces données constituent les unités lexicales d'entrée du langage d'interaction). Elle traduit les unités lexicales d'entrée en forme abstraite compatible avec le monde informatique interne. Ces représentations définissent les unités syntaxiques du langage.

La Présentation reçoit des expressions abstraites en provenance du Contrôleur du Dialogue (ces abstractions définissent les unités syntaxiques de sortie). Elle interprète les unités syntaxiques de sortie dans les termes du lexique de sortie (primitives graphiques, sons,.....). Elle est le seul composant à manipuler directement les dispositifs physiques d'entrée/sortie. Les autres composants passent par la PRESENTATION afin d'effectuer des échanges avec l'utilisateur.

Les objets interactifs doivent ignorer l'existence des objets sémantiques qui leurs correspondent et inversement. Ainsi si l'objet sémantique "température" est un entier, l'objet interactif qui est un dessin d'un thermomètre doit ignorer que pour l'application, la notion température est modélisée par un entier.

Le composant CONTROLE DU DIALOGUE joue le rôle de médiateur entre les deux autres composants. Il est responsable de l'analyse syntaxique du langage d'interaction : les unités syntaxiques reçues de la Présentation sont assemblées en phrases. Les phrases syntaxiquement correctes correspondent à des requêtes et à des données que l'utilisateur souhaite transmettre à l'application.

Dans le sens inverse, le CONTROLEUR reçoit des phrases de sortie abstraites qu'il ventile vers les éléments spécialisés de la Présentation. Il a également un rôle de gestionnaire de l'état de l'interaction, il gère la dynamique de l'interaction. L'ensemble des états possibles, leurs relations et la composition des phrases définissent la structure du dialogue entre

l'utilisateur et l'application. Le Contrôle du dialogue connaît toutes les préconditions, les contraintes de déclenchement des fonctions. Il peut ainsi vérifier si une demande de déclenchement d'une fonction peut être acceptée ou non.

Le composant INTERFACE AVEC L'APPLICATION est le représentant auprès du CONTROLEUR. Il définit la vue que le contrôleur a de l'application. L'application réunit les concepts et les opérations propres à l'accomplissement de tâches dans un domaine précis. Ces concepts et leurs opérateurs définissent la sémantique du langage d'interaction. Il réalise les fonctions et objets du domaines en termes d'objets sémantiques.

J. COUTAZ souligne les qualités requises pour le Contrôleur. Ce dernier doit fournir à l'utilisateur un dialogue souple et un retour d'information immédiat et informatique (nous verrons dans le chapitre suivant que ceci est un des critères d'une interface de qualité). Cette souplesse doit se faire tant au niveau sémantique que syntaxique. Le contrôleur doit pouvoir, si nécessaire, offrir des vues multiples d'un objet sémantique (exemple : niveau de mercure dans un thermomètre et courbe de température sont deux vues de l'objet sémantique qu'est le concept température).

La figure 6.5. montre les liens qui existent entre les composants du modèle.

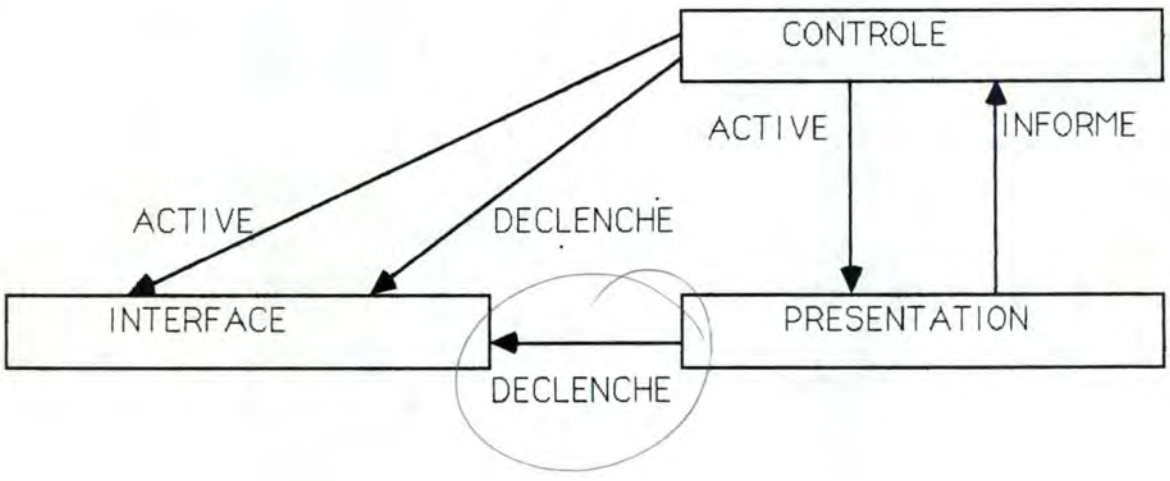


Figure 6.5 : Les relations entre les composants

La relation DECLENCHE entre le Contrôle et l'interface avec l'application indique que les services proposés par l'application sont requis par le Contrôle. La relation ACTIVE entre le Contrôleur et la Présentation indique que la Présentation renseigne le Contrôle sur l'action de l'utilisateur et du résultat. La relation INFORME entre le Contrôle et la Présentation indique que cette dernière informe, avertit le Contrôle d'une action de l'utilisateur.

Nous venons de donner une description du modèle SEEHEIM en précisant les composants, leurs interactions, leurs actions. Le dialogue relève du Contrôleur et de la Présentation. Nous allons maintenant donner une spécification des dialogues qui entrent en jeu.

6.1.2.2. Spécification d'un dialogue

G. WARNANT indique que la structure d'un dialogue peut être scindée en deux parties : la tâche et l'interface concrète [WARNANT, 1988]. La tâche est le niveau abstrait du dialogue (qui relève du Contrôleur dans le modèle de SEEHEIM) alors que l'interface en est sa représentation concrète (qui relève de la Présentation dans le modèle de SEEHEIM).

La spécification de la tâche

La tâche est l'expression fonctionnelle des opérations que l'utilisateur doit réaliser pour mener à bien l'exécution de l'application interactive. Une tâche est composée de messages interactifs, d'opérations définies sur chaque message et de règles d'enchaînement des opérations.

Un message interactif est une unité abstraite de saisie ou de communication d'information destinée à l'utilisateur. A ce niveau, aucune hypothèse concernant la représentation du message interactif n'est posée, il y a donc une indépendance avec l'outil de réalisation.

G. WARNANT distingue quatre catégories de messages interactifs. Les messages interactifs fonctionnels gérant les informations d'entrée ou de

sortie des fonctions de l'application rentrant dans la première catégorie. Ils correspondent aux messages fonctionnels c'est-à-dire reçus ou générés par les fonctions. La deuxième catégorie correspond aux messages d'erreurs du dialogue dont le but est d'informer l'utilisateur de ses erreurs syntaxiques. L'avant-dernière catégorie distinguée regroupe les messages d'aide à l'utilisateur. Les messages de contrôle du dialogue permettent à l'utilisateur d'orienter l'exécution de l'application en posant un choix face à des alternatives qui lui sont proposées.

Les messages interactifs non fonctionnels sont internes à la partie dialogue de l'application.

Les opérations que l'utilisateur peut effectuer sur un message interactif sont, par exemple la saisie de caractères ou encore la clôture du message qui va déclencher les traitements associés à sa terminaison.

Les messages interactifs ne peuvent être effectués à n'importe quel moment. Des contraintes de déclenchement des opérations sur les messages interactifs doivent être définies. Ces règles d'enchaînement des messages interactifs constituent la dynamique de l'exécution de la tâche. On peut appeler cet enchaînement le dialogue global en opposition au dialogue local. Dans le modèle SEEHEIM le dialogue global est géré par le composant Contrôle du dialogue. Le dialogue global représente donc la dynamique de l'exécution de la tâche. Il détermine l'ordre dans lequel les messages interactifs seront présentés à l'utilisateur. Nous verrons dans le chapitre 7 que les éducateurs peuvent influer sur certains "paramètres" qui orienterons l'exécution de la tâche. Ils modifient le dialogue global.

La spécification de l'interface concrète

La seconde partie consiste en la spécification de la présentation concrète de la tâche. Dans le modèle SEEHEIM, l'interface concrète n'est autre que la composante Présentation. Elle est indépendante de l'outil de réalisation. Nous considérons l'interface concrète comme la partie visible à l'utilisateur. Elle est composée d'objets interactifs concrets (fenêtre, icône) qui sont la matérialisation des messages interactifs et sur lesquels l'utilisateur peut effectuer des actions physiques (cliquer au moyen de la souris sur une icône, validité une information en appuyant sur la touche

<Return>, ...). Nous avons parlé des objets interactifs concrets car dans la littérature le terme objet interactif concret est opposé au terme objet interactif abstrait qui définit un objet du dialogue du point de vue de son comportement et des opérations primitives permettant la manipulation de la souris. Ces objets interactifs abstraits sont indépendants de l'outil de réalisation.

6.2. Réalisation d'une interface adaptée

Comme nous l'avons indiqué au début de ce chapitre, un système interactif implique une communication importante entre l'ordinateur et l'utilisateur. Cette communication se fait via l'interface Homme-machine. D.L. SCAPIN indique que "l'interface Homme-Ordinateur englobe tous les aspects des systèmes informatiques qui influencent la participation de l'utilisateur à des tâches informatisées" [SCAPIN 84].

Le but de l'interface est de permettre à la personne de réaliser le mieux possible la tâche qu'elle a à accomplir, c'est-à-dire avec précision, rapidité et sans efforts inutiles (à savoir étrangers à la nature de la tâche).

Les enjeux économiques des interfaces sont importants. Nous pouvons souligner qu'une interface adaptée peut permettre à des personnes de plus en plus étrangères au monde des ordinateurs de les utiliser dans leur domaine comme les médecins, les enfants, les personnes handicapées et bien d'autres. Elle permet également d'accroître la productivité des utilisateurs spécialisés. Un logiciel sera d'autant plus accepté et utilisé par les utilisateurs qu'il présente une interface qui leur est adaptée. Dés lors la conception d'interface adaptée aux futurs utilisateurs est un but majeur auquel devrait s'atteler les concepteurs de systèmes informatiques.

6.2.1 L'ergonomie

Pour arriver à ce but, l'apport d'autres disciplines comme l'ergonomie s'avèrent nécessaire. G. KARNAS définit l'ergonomie comme "une discipline dont l'objet est l'étude du travail de l'homme. Son objectif est l'adaptation du travail à l'homme". Liées à l'évolution du travail, G.

KARNAS distingue différentes formes d'ergonomie [KARNAS 87] :

L'ergonomie physique

Elle porte sur les aspects physiques du travail. L'homme a un corps et une force dont il faut tenir compte pour développer les postes de travail. L'ergonomie physique prend en compte la diversité des aptitudes physiques telles que par exemple la position du corps (hauteur du siège, inclinaison du buste,...); la vision (vision périphérique, vision des couleurs, sensibilité des contrastes, capacité d'identifier un objet dans un contexte,...)

L'ergonomie de l'information

Lorsque la technologie se substitue à la force physique, l'homme doit commander et contrôler le dispositif mis en oeuvre. Par conséquent, des problèmes de communication d'information apparaissent entre l'homme et le dispositif qu'il commande.

L'ergonomie des systèmes

Il s'agit de la prise en charge réelle des interactions entre une cellule de production constituée du couple (homme-machine) et les autres cellules de production par des dispositifs de communication homme-machine (communication verbale, signaux, d'affichage,...)

L'ergonomie cognitive

Elle tient compte des caractéristiques de l'homme au niveau cognitif pour améliorer les performances du couple homme-machine. Elle est la plus récente.

Par opposition à d'autres machines qui représentent des extensions du corps humain, par exemple qui fournissent des capacités et une puissance supplémentaire, l'ordinateur, et en particulier le logiciel, représente une extension du cerveau humain, en faisant appel à des processus plus cognitifs (tel la perception, la mémoire,...). Dès lors pour améliorer l'interface Homme-Ordinateur, il convient de s'intéresse plus particulièrement aux processus cognitifs humains. En conséquence

l'ergonomie cognitive est, de tous les aspects de l'ergonomie, celui dont l'impact sur le développement du projet informatique est le plus grand.

L'ergonomie cognitive joue un rôle majeur dans les différents aspects de la conception d'une interface Homme-Machine : la présentation des objets du dialogue (textes, graphiques, messages d'erreurs, messages d'aide), la conversation entre l'homme et la machine, la guidance, ou l'assistance.

L'ergonomie cognitive, désignée de plus en plus par l'expression **ergonomie de logiciel** (ou ergonomie intellectuelle) n'est devenue qu'une discipline essentielle que depuis quelques années. Cependant elle a eu historiquement, et a toujours, une part importante dans la conception et l'évaluation de systèmes critiques tels que le contrôle aérien, les applications militaires et les centrales nucléaires. Ce qui est spécifique à l'ergonomie des logiciels par rapport à d'autres aspects plus classiques de l'ergonomie est que la conception du logiciel établit le contenu des informations disponibles à l'utilisateur aussi que les relations visuelles entre ces informations.

Elle exploite des éléments de la psychologie cognitive afin d'adapter les dispositifs matériels et logiciels du système Homme-Machine aux caractéristiques cognitives de l'individus. M.F. BARTHET la définit en ces termes "La Psychologie cognitive désigne la partie de la psychologie qui s'intéresse à la manière dont l'homme traite l'information" [BARTHET 1988]. Parmi les éléments de psychologie cognitives d'intérêt pour la conception des interfaces Homme-Machine, M.F. BARTHET retient :

- **L'analyse de la tâche** : tâche effective, tâche prévue
- le modèle de **planification hiérarchique** pour la résolution d'une tâche. La description de la tâche orientée vers un but : du résultat aux actions élémentaires qui permettent de l'atteindre
- **Images mentales** et **images opératives**
- **logique du fonctionnement** et **logique d'utilisation**
- théorie de **l'apprentissage**
- système de **mémorisation** ou système cognitif.

6.2.2 Qualités d'une bonne interface

Une interface adaptée aux utilisateurs est donc une nécessité. La conception d'interface souligne COUTAZ [1989], est difficile, longue et coûteuse. Le problème principal est de pouvoir réduire le fossé qui sépare l'univers psychologique dans lequel une personne pense sa tâche à accomplir (objectif, intention) et l'univers physique de l'interface dans lequel elle accomplit les actions nécessaires à la réalisation de la tâche. Une personne exprime ses objectifs en termes significatifs pour elle, en termes liés à la représentation mentale qu'elle se fait de son problème (termes psychologiques). Elle doit traduire ses objectifs dans des actions à exécuter à l'aide des mécanismes du système interface exprimés en des termes physiques. D'une manière analogue, elle doit évaluer la réalisation de ses objectifs à partir des variables d'état du système interface qui sont des variables physiques.

La qualité d'une interface réside dans la capacité de réduire le fossé entre ces deux types de variables. Différents critères ont été proposés pour évaluer la qualité d'une interface, qualité que l'on désigne sous le nom de *convivialité* (user friendliness). SCHNEIDERMAN [SCHNEIDERMAN, 1987] propose cinq critères pour évaluer la qualité d'une interface.

- Le temps d'apprentissage des commandes nécessaires à l'exécution d'une tâche.

- La rapidité d'exécution d'une tâche, c'est-à-dire le temps nécessaire à la réalisation de la tâche.

- Le taux d'erreurs effectuées par l'utilisateur. L'évaluation doit porter sur la fréquence des erreurs en distinguant deux types d'erreurs à savoir tout d'abord les erreurs d'exécution (erreurs syntaxiques) généralement liées à des défauts de manipulation du système d'interaction (clavier, souris, ...). Le deuxième type d'erreurs à distinguer sont les erreurs d'intention lorsque l'utilisateur sélectionne une commande inappropriée. L'évaluation doit également porter sur le temps de correction.

- La période de rémanence durant laquelle un utilisateur conserve la

connaissance acquise, est une fonction directe de l'effort cognitif requis pour planifier une séquence d'actions qui réalisent une intention donnée.

- La satisfaction subjective à utiliser le système

L'appréciation de ces critères va dépendre du contexte, de la nature de la tâche. Il existe des tâches qui demandent une très grande sécurité comme le pilotage d'une centrale nucléaire, ce qui implique que les critères à rencontrer absolument sont un taux d'erreur nul et une rapidité d'exécution. Dans notre cas, rencontrer les autres critères n'est pas fondamental. Pour ce qui est d'un système d'enseignement, nous avons déjà indiqué qu'un facteur favorisant l'apprentissage était la motivation. Par conséquent, le critère à rencontrer peut être la satisfaction subjective à utiliser le système.

Pour atteindre ces objectifs, ils n'existent pas de recette miracle, tout au plus des règles qui sont mises en évidence par l'expérimentation. Nous nous contenterons d'exposer celles qui sont le plus fréquemment citées dans la littérature [SCHNEIDERMAN, op.cit.; SCAPIN, op cit. ; J. Coutaz, op.cit.] :

La lutte pour la cohérence c'est-à-dire la recherche du caractère unitaire des éléments de l'interface à tous les niveaux d'abstractions doit conduire à l'élimination des exceptions et des contradictions. La cohérence doit aussi bien être respectée au niveau des emplacements des différentes commandes et des actions que l'utilisateur doit réaliser qu'au niveau des emplacements des informations. Les informations importantes doivent se trouver toujours au même endroit (à partir du coin gauche et au centre). La cohérence doit porter également sur le choix d'une nomenclature de termes précis, didactiques, pertinents et proches de l'utilisateur. La cohérence peut également porter sur différentes applications (commandes identiques au même endroit d'une application à l'autre).

La recherche de la concision est la deuxième règle, c'est de fournir des dispositifs de communication brefs et expressifs. Elle préconise l'usage de valeur par défaut, de règles d'abréviation.

La structuration des activités concerne l'organisation de l'espace de travail de l'utilisateur. Elle concerne la structuration des commandes par couche de complexité croissante, par classe de fréquence d'utilisation. Il faut éviter que le nombre de commandes que l'utilisateur doit effectuer pour réaliser sa tâche ne soit trop important tout en ne tombant pas dans l'excès contraire, ce qui provoque un accroissement de la charge mentale.

Le retour de l'information renseigne l'utilisateur sur l'état du système dans la perception de son emploi. Le retour d'information doit avoir pour but d'indiquer ce que fait le système. Il doit avoir pour but d'indiquer à l'utilisateur les commandes qu'il ne peut effectuer dans le contexte où il se trouve. Les messages fournissant ces informations peuvent être de différents types : dessin, texte, schéma,... . Cette règle est liée au temps de réponse. Dans le cas d'utilisateurs handicapés mentaux, le temps de réponse ne devrait pas être important sans quoi le risque d' un désintéressement de la personne apparaîtra. Ce risque est d'autant plus grand que le logiciel est destiné à des utilisateurs handicapés mentaux dont il est plus difficile de maintenir l'attention.

La gestion des erreurs est une autre règle. L'interface doit permettre la détection rapide des erreurs et fournir aux utilisateurs la possibilité de les corriger. Il faut signaler immédiatement ou le plus rapidement possible ces erreurs, apporter un diagnostic aussi explicite que possible, signaler la procédure de correction si possible. La possibilité d'annuler toute action critique doit être offerte afin d'éviter l'anxiété chez l'utilisateur. Cette possibilité est d'autant plus importante pour les personnes handicapées mentales qu'elle agissent avant de réfléchir aux conséquences de leur action. En ce sens, une confirmation doit être demandée après les actions dont les conséquences peuvent être importantes. Il est également nécessaire qu'elles soient bien différenciées, physiquement et sémantiquement, afin d'éviter toute confusion dans l'esprit des personnes.

La flexibilité est définie en ces termes : "la flexibilité d'une interface désigne sa faculté d'ajustement aux variations de l'environnement, et notamment à l'utilisateur" [COUTAZ, 1989]. Cette flexibilité implique qu'il doit être possible d'adapter la terminologie, reformuler les énoncés de message par l'utilisateur, d'utiliser l'interface en fonction du profil cognitif, de pouvoir présenter un même concept de différentes manières d'après le

profil cognitif.

Le respect de ces règles devrait amener les concepteurs à réaliser des interfaces de qualité c'est-à-dire qui répondent aux critères évoqués. Il n'existe pas de lien entre ces derniers et ces règles. Ce qui signifie que l'application des règles ne garantisse pas nécessairement la qualité de l'interface. L'avis des utilisateurs s'avère par conséquent primordial. Nous verrons dans la sous-section suivante les choix que nous avons opérés pour réaliser une interface de qualité. Les représentations des écrans présentées dans ce chapitre peuvent être compulsées en parallèle avec les explications que nous donnons. Nous ne parlerons que des règles qui s'appliquent au logiciel.

6.2.3 Nos choix pour une interface de qualité

En ce qui concerne la cohérence, les icônes de commande sont toujours positionnées aux mêmes endroits; elles sont dessinées sur un fond bleu et les icônes proprement dites sont de différentes couleurs (le vert pour le "FINI" et le rouge pour le "STOP" notamment). Les icônes et leur position sont identiques à celles utilisées dans le logiciel "Corps" [LEPOUTRE, 90]. A ce niveau la cohérence inter-application est respectée. Comme l'activation des icônes se fait au moyen du bouton gauche de la souris, toutes les actions de l'utilisateur s'effectuent de la même manière. La cohérence des informations se situe dans la position des messages textuels des feed-back. Quand l'utilisateur désigne le dessin d'un organe, d'un système, la partie d'un parcours, le massage textuel du feed-back apparaît dans la partie inférieure droite de l'écran. Il en est de même pour les questions posées par la machine.

En ce qui concerne la structuration des activités, la technique, décrite par après, de la désignation du parcours permet à l'utilisateur d'exécuter sa tâche (désigner le parcours) en une seule commande répétée à savoir le clic sur la souris. Ce qui ne surcharge pas la mémoire de la personne.

Un retour d'information est présent à chaque action physique (avec la souris) réalisée part l'utilisateur. En ce qui concerne la désignation d'une partie de l'écran (désignation d'un organe par exemple), le feed-back consiste en un message textuel placé dans la partie inférieure droite de

l'écran ainsi qu'éventuellement un message sonore. Ce dernier peut quant à lui être reconsulté aisément par l'utilisateur en cliquant sur l'oreille. Cette oreille reste présente lorsque l'utilisateur doit confirmer une de ses actions et lui permet d'écouter une nouvelle fois le message vocal associé à la confirmation. Dans le cas de l'exercice de désignation par nom, l'utilisateur peut, en cliquant sur l'oreille, obtenir à nouveau la question (visualisée par un clignotement de l'organe dont le nom est demandé). Elle constitue en fait l'aide proposée à l'utilisateur.

Les différents feed-back de l'exercice parcours (rappelons qu'il s'agit soit d'un déplacement du symbole représentant une substance, soit d'une coloration, soit du symbole et de la coloration simultanément) informent l'utilisateur sur la partie du parcours qu'il a déjà indiqué depuis le début de l'exercice. Ces feed-back réduisent le fossé entre les variables psychologiques et les variables physiques de la représentation d'un parcours. La coloration correspond à une visualisation du parcours déjà désigné c'est-à-dire que l'utilisateur voit les organes par où il est passé. Ainsi l'effort cognitif requis par les utilisateurs pour la désignation du parcours est moindre que s'il n'y avait aucun retour d'information si ce n'est une indication de sa réussite. Les utilisateurs restent ainsi dans la logique opératoire décrite au chapitre 3 (qui est la logique de la majorité des personnes handicapées mentales légères et modérées).
Les temps de réponse du logiciel sont bons. Lorsque l'utilisateur reçoit un feed-back vocal, par exemple, il n'est pas bloqué dans la réalisation de la tâche. Ceci est rendu possible par la machine utilisée qui permet le multi-tâches. Lorsqu'un traitement demande un délai plus long, (notamment lors de la préparation d'un exercice), des figures animées sont présentées à l'utilisateur. Ces images devraient assurer le maintien de son attention jusqu'à la reprise de la tâche.

Le logiciel, étant un logiciel d'apprentissage, requiert une gestion tant au niveau des erreurs de connaissances qu'au niveau des erreurs d'exécution. Cette gestion d'erreur se traduit par la correction immédiate en cas d'erreur afin que l'utilisateur soit informé sur la nature de l'erreur commise. Afin d'éviter les erreurs d'exécution (mauvaise manipulation) nous avons pour l'exercice de type parcours opter pour la représentation agrandie de la partie de la silhouette contenant le parcours (par exemple un buste est représenté à l'écran avec le système digestif) afin que le parcours

soit plus grand et qu'il soit plus facile de le désigner . Cette représentation ayant donc pour but de diminuer les erreurs de manipulation peut présenter cependant un accroissement de l'effort cognitif que l'utilisateur (ne perdons pas de vue qu'il s'agit de personnes handicapées mentales) doit consentir afin de comprendre qu'il s'agit d'une silhouette humaine. Pour éviter ce désavantage nous avons décidé d'afficher à l'écran, à gauche la silhouette en entier.

Il est vrai qu'il est difficile d'être exhaustif quant aux qualités d'une interface. De plus comme nous l'avons dit il n'existe pas de lien entre les critères d'une bonne interface et les règles empiriques. Il est évident qu'une évaluation de l'interface auprès des utilisateurs est nécessaire. L'utilisation du logiciel permettra de valider les choix que nous avons effectués. Il est important de réaliser une interface adaptée. Cette réalisation est rendue plus difficile par les faibles capacités intellectuelles des personnes présentant une déficience mentale. Le fossé entre les variables psychologiques et les variables physique a une étendue difficile à percevoir d'autant plus que la diversité de ces personnes est importante.

6.3. Implication des utilisateurs

Coutaz et d'autres auteurs mettent l'accent sur l'importance dans tout développement de projet informatique de consulter les utilisateurs futurs. Cette consultation doit se faire le plus tôt possible (dés la phase de spécification des fonctionnalités, et des dialogues). Une trop faible participation des utilisateurs au niveau de la définition des fonctionnalités augmente la probabilité d'obtenir des erreurs qui ne seront détectées qu'à la fin du développement. Ce qui signifie le recommencement de l'ensemble du processus de développement avec les coûts que cela entraîne. Leur participation est également désirable pour la définition de l'interface. Un dialogue mal spécifié peut entraîner le rejet de l'application et ce même si les fonctionnalité rencontrent parfaitement les besoins des utilisateurs. Les causes de ce rejet ne sont pas toujours apparentes immédiatement et sont généralement difficilement détectables.

Dans le cadre du développement d'un projet classique en entreprise, R. LESSUISSE [1987] propose trois formes de participation des utilisateurs .

-La participation par consultation: "Un groupe de techniciens consulte les utilisateurs pour établir un diagnostic précis de la manière dont les tâches à automatiser sont actuellement exécutées et pour la relève des déficiences de développement. Ces techniciens tentent ensuite d'apporter des remèdes techniques à ces lacunes".

-La participation par représentation: "La conception et la réalisation d'un projet informatique sont entièrement pris en charge par un groupe de techniciens et de représentants d'utilisateurs à tous les niveaux de la hiérarchie de l'organisation".

-La participation par concensus: " Lors de la conception et de la mise en oeuvre d'un projet informatique, il y a implication permanente et intensive de toutes les personnes concernées par le projet".

R. LESSUISSE explique que dans la plupart des cas, la participation par représentation est la plus utilisée. Ces méthodes participatives devraient permettre une meilleure acceptation du projet par les utilisateurs. Il indique également que cette participation des utilisateurs pose un problème de communication entre les utilisateurs et les informaticiens. Il faut en effet que les utilisateurs puissent exprimer leurs besoins et comprendre les plans de solution.

La technique classique consiste en la rédaction de rapports de synthèses soumis aux utilisateurs et en la proposition de plans de solution. La difficulté pour les utilisateurs de se faire une idée précise de la solution informatique sur base du cahier de charge suggère une approche par prototypage. Le processus de développement du projet prend alors la forme suivante:

-des questionnaires, interviews, etc .. permettent aux utilisateurs d'exprimer leurs besoins.

-un prototypage de l'application reprenant les besoins des utilisateurs leur est proposé .

-les utilisateurs critique le prototype . Sur base de ces critiques, des modifications sont apportées.

L'approche par prototypage est souvent utilisée pour la création des écrans d'une application interactive. L'utilisateur comprend mieux ce qu'il voit sur un écran que des explications textuelles.

L'approche par prototypage implique une participation des utilisateurs dans le processus de développement aussi bien au niveau de l'analyse des besoins comme dans le cas du processus de développement classique, mais également tout au long du développement. Le processus devient alors itératif.

Nous verrons dans la section suivante, les difficultés liées à la participation des utilisateurs handicapés mentaux.

6.3.1. La participation des personnes handicapées mentales

Nous avons vu dans la section précédente que la participation des utilisateurs est importante au développement d'un projet informatique. Une difficulté surgit lorsque l'utilisateur est une personne ayant une déficience mentale. En effet, le problème de la communication se pose. De plus, il faut cerner les besoins des personnes qui sont peu aptes à les spécifier. Ainsi la participation ne doit pas se limiter aux seuls utilisateurs mais s'étendre aux personnes qui les connaissent le mieux (que ce soit en les contoyant, en les éduquant, ou en les étudiant). Nous avons donc collaboré étroitement avec des psychologues, éducateurs et enseignants spécialisés dans le domaine du handicap mental. L'élaboration d'un logiciel pour personnes handicapées nécessite une collaboration de spécialistes de différents domaines comme l'indique M. FRAITURE et C. MACHGEELS [1990]. Ce domaine de recherche est donc pluridisciplinaire.

Nous allons présenter dans la section suivante notre démarche de spécification et indiquer quelle fut la participation des utilisateurs.

6.4. Notre démarche de spécification

Lors de notre stage en Suisse nous avons élaboré un cahier de charge détaillé pour les deux exercices proposés dans la définition initiale du projet. En collaboration avec des représentants des utilisateurs, principalement des éducateurs et psychologues, nous avons spécifié les scénarios détaillés mais également une ébauche de la présentation de la tâche c'est-à-dire la représentation concrète des messages interactifs identifiés dans les scénarios. Nous avons utilisé à cette fin le logiciel "Corps" développé par Thierry Lepoutre et Jean-Michel Roquet [LEPOUTRE 1990]. Nous avons présenté à des psychologues, des éducateurs ce logiciel et nous l'avons également tester avec des personnes ayant un handicap mental, en vue de se donner une opinion de la manière dont les utilisateurs l'utilisent.

6.4.1. La spécification des scénarios

En ce qui concerne les scénarios, nous avons été exclusivement guidés par les pédagogues et les psychologues. Les utilisateurs n'avaient aucun intérêt dans les spécifications des scénarios.

Lors de notre stage en Suisse, nous avons présenté le logiciel "Corps" à de nombreux éducateurs ainsi qu'à des psychologues. Par ces nombreuses démonstrations, nous avons pu facilement leur soumettre nos spécifications de scénario pilotant l'enchaînement des exercices car à ce niveau le "Corps" présentent les mêmes spécifications qui furent soumises à d'autres psychologues et éducateurs lors du développement du logiciel "Corps". Nos interlocuteurs pouvaient ainsi voir un produit qui était en partie similaire à celui que nous allions développé. Aucune proposition ne fut apportée en vue d'une modification de l'enchaînement des exercices par les spécialistes en handicap mental rencontrés. C'est ainsi que nous avons décidé d'adopter ce scénario d'enchaînement des exercices.

Il se traduit par la possibilité pour l'utilisateur de recommencer l'exercice qu'il vient de terminer ainsi que la possibilité de passer à un exercice suivant ou de stopper la session d'exercices, avec la possibilité de la reprendre plus tard. Ceci donne un certain contrôle de l'utilisateur sur

l'enchaînement des exercices qu'on lui propose (nous verrons dans le chapitre suivant que c'est l'éducateur qui choisira les exercices qui seront présentés à l'utilisateur).

En ce qui concerne la technique des questions posées lors d'un exercice de désignation, la décision a été prise, à chaque occurrence d'une désignation erroné, de reposer immédiatement la même question. Lors des évaluations que nous avons faites du logiciel précédent [LEPOUTRE et ROQUET, 1990], nous remarqué la tendance des personnes testées à vouloir se corriger immédiatement après leur erreurs, sans écouter la question suivante. Cela entraînait des erreurs en chaîne. Le renforcement positif sera constitué par un feed-back musical et le passage à la question suivante.

6.4.2. La spécification de l'interface concrète

La spécification des scénarios n'a pas connu de modifications pendant le développement du logiciel. Par contre la spécification de l'interface concrète en a connu. Ces modifications ont été apportées à la suite de contacts avec les représentants des utilisateurs et avec les utilisateurs eux-mêmes.

Elles concernaient notamment les graphiques à utiliser, la représentation des organes à adopter, la désignation du parcours par l'utilisateur d'une substance dans le corps humain. Nous avons déjà parlé dans le chapitre 4 des modifications et des choix en ce qui concerne les graphiques utilisés et ainsi que de l'implication des représentants des utilisateurs et des utilisateurs eux-mêmes.

Dans l'exercice de type parcours, l'utilisateur est invité à montrer un parcours. Une question s'est posée dès le début du développement : quelle action doit-il réaliser pour désigner le parcours? De plus la difficulté de la réalisation de la représentation graphique est apparue.

En effet, pour être significative et reconnaissable, la silhouette humaine doit apparaître en entier à la vue de l'utilisateur lors de la réalisation de l'exercice. Les zones de travail effectives sur l'écran s'en

trouvent donc extrêmement réduites et en deviennent presque non significatives à cause de la limitation de la taille de l'écran. Quand il s'agit de designer un parcours dans des organes de deux pixels de large, il faut faire preuve d'une grande précision, d'une attention certaine, et il faut pouvoir y consacrer du temps même pour une personne habile. Or, nous nous adressons ici à des personnes handicapées mentales qui, de plus, peuvent présenter des problèmes psychomoteurs.

Ceci va à l'encontre du but même de l'interface, à savoir, permettre à l'utilisateur de réaliser le mieux possible la tâche qu'il a à accomplir, avec précision, rapidité et sans efforts inutiles (efforts étrangers à la nature de la tâche qui est ici la désignation d'un parcours et non un exercice de précision de manipulation de la souris).

Ce premier problème a été résolu en réalisant un agrandissement de la partie du corps dans laquelle se situe le parcours (généralement le buste), et en gardant toujours en haut à gauche de l'écran une silhouette complète en réduction comprenant le système concerné par le parcours . L'utilisateur peut ainsi comprendre que la partie agrandi est une partie du corps humain. Ceci permet, de plus, de travailler sur des parcours moins étendus , et donc, acceptant une précision moins exagérée.

En ce qui concerne la désignation même du parcours, il s'agit pour l'utilisateur de désigner un point d'entrée, un trajet continu passant par certains organes et un point de sortie. Il y a donc également un concept de sens du parcours. Le parcours est représenté à l'écran par un tuyau d'une certaine largeur.

La première technique qui nous est venue à l'esprit est le "dagging" qui consisterait à désigner le point d'entrée en y plaçant le curseur de la souris et en appuyant sur le bouton; de faire suivre le parcours par le curseur en maintenant le bouton appuyé jusqu'au point de sortie. Bien que la taille des organes se soit améliorée, la technique nécessite tout de même une grande dextérité. A aucun moment, il ne faut sortir du parcours.

Une seconde technique consiste à montrer un parcours en désignant seulement certains points dans le tuyau désignant le parcours. Il s'agit donc de "cliquer" (désigner un point de l'écran avec le curseur de la souris,

appuyer et relâcher le bouton en ayant maintenu la position. On n'exige donc pas ici une précision de tout les instants dans la position du curseur pour la désignation du parcours. Il suffit de le positionner correctement quand on appuye sur le bouton de la souris.

Nous avons remarqué l'avantage de cette dernière technique lors de quelques tests non formels que nous avons effectué en Suisse sur des personnes handicapées mentales correspondant à la population cible, à l'aide d'un logiciel de dessin. Le premier test consistait à tracer un trait continu pour désigner un parcours dans un labyrinthe, ceci correspondant au "dragging". Le second exercice consistait à colorier des cases constituant un parcours dans ce même labyrinthe, cette action correspond à "cliquer" dans une case, la coloration pouvant être interprètée comme un feed-back. La deuxième technique s'avère beaucoup plus rapide (près de 3 fois), toute chose égale par ailleurs, et donc plus aisée. Elle emportait la préférence de la majorité des personnes testées.

Nous divisons donc le parcours en cases dans lesquelles il faut "cliquer". Ces cases sont invisible à l'utilisateur,elles ne mettent donc pas en péril la notion de continuité. De plus, il est intéressant que l'utilisateur puisse désigner une case qui n'est pas directement contiguë à la dernière case désignée. La possibilité est donc donnée d'oublier une (ou plusieurs) case(s), d'autant plus qu'il n'est pas au courant de leur position. Dans ce cas, il suffirait de désigner le point de sortie pour terminer le parcours. Ce n'est pas notre souhait. C'est pourquoi, nous avons défini des ensembles de cases obligatoires (correspondant aux organes traversés par le parcours) par lesquels l'utilisateur doit obligatoirement passer. Pour désigner un parcours correct, il doit désigner au moins une case dans chacuns de ces ensembles et doit donc désigner chaque organe traversés.

La notion de sens du parcours est assurée par le fait qu'un retour en arrière dans la désignation correspond à un parcours incorrect.

Le feed-back positif associé à la désignation d'une parie correcte du parcours peut être de deux natures non exclusives:

-la coloration du parcours déjà réalisé.

-le déplacement du symbole (boulette de viande, oxygène...) de l'endroit désigné précédemment jusqu'à l'endroit actuellement désigné.

Comme on le voit, ces deux feed-backs contribuent à la notion de continuité, même si la technique de désignation est ponctuelle. On choisira volontiers le déplacement du symbole pour représenter la nourriture dans le système digestif,et plutôt la coloration pour représenter l'oxygène dans le système respiratoire.

Le feed-back négatif, apparaissant lors d'une erreur dans la désignation du parcours, correspond à la reprise du parcours au début quel que soit l'endroit où il est arrivé. L'exercice se termine quand l'entièreté du parcours est désignée(i.e. jusqu'au(x) point(s) de sortie).

Des paramètres constituants une aide pour l'utilisateur peuvent être modifiés de tel sorte que le parcours complet puisse être montré ou non en début et en fin d'exercice.Il est également possible pour la machine de désigner ou non automatiquement le point d'entrée et de sortie en début d'exercice, ces deux points de passage ne sont donc plus à désigner.

Si on prend l'exemple de l'air qui rempli les deux poumons, et qui peut entrer par la bouche ou le nez, on remarque que le parcours peut parfois se diviser en deux chemins distincts dans lesquels le feed-back (coloration et/ou déplacement du symbole) doit se faire s'opérer simultanément. Le parcours ne sera donc plus modélisé par un ensemble de cases, mais par un ensemble de paires de cases pas forcement différentes. Il suffira qu'une des cases du couple soit désignée pour que le feed-back s'opère dans les deux cases du couple. Ainsi, pour montrer que l'oxygène circule dans les poumons, il suffit à l'utilisateur de "cliquer" dans un des deux poumons pour que le feed-back se produise dans les deux simultanément

Les erreurs commises dans la désignation du parcours peuvent être de trois natures. Si on considère la case A désignée correctement précédemment, la première erreur serait de désigner une case antérieure à A. Cela correspond à un retour en arrière de la part de l'utilisateur. Il ne respecte pas la sens du parcours. Il recommence donc au début. Le deuxième type d'erreur est d'oublier de désigner une case dans un ensemble de cases (couple de cases) obligatoires . Par exemple, si A est une case du premier ensemble obligatoire (représentant l'oesophage), et qu'une case du 3ème ensemble (les intestins) est désignée, l'utilisateur a oublié le 2ème

groupe obligatoire (l'estomac). Il recommence de nouveau au début. La dernière erreur est la désignation d'un pixel hors des cases, et donc hors du parcours, ce qui entraîne le même effet que les erreurs précédentes.

L'exercice de désignation par la diversité des types de désignations possibles pose des problèmes. Il s'agit comme nous l'avons expliqué, de désigner un organe dans une silhouette représentant un système(*désignation simple*), de désigner le nom d'un organe mis en évidence dans la silhouette (*désignation par nom*), ou un système en connaissant sa fonction (*désignation par système*). Cette silhouette peut être vide, on désignera alors l'emplacement d'un organe. Il est également possible de savoir quel est le nom d'un organe désigné et l'organe correspondant à un nom désigné dans un type de désignation que nous appellerons respectivement *pseudo-passif* et *pseudo-passif par nom.*

Une pièce à sélectionner est associée à un cadre de désignation invisible qui la comprend entièrement. La désignation de la pièce est perçue par l'ordinateur si on "clique" dans ce cadre sur un pixel différent de la couleur de fond. Toute partie colorée se trouvant dans le cadre de désignation de la pièce sera considéré comme faisant partie de la pièce.

Le cas simple de cet exercice est rencontré lorsque les cadres de désignation ne contiennent entièrement que les leur pièces correspondants. Ils n'ont donc pas d'intersection. Pour notre logiciel, il est impossible d'encadrer un organe complètement, sans prendre une partie d'un autre organe. Ceci est d'autant plus vrai qu'il nous est demandé de représenté des organes qui sont, parfois, superposés les uns aux autres (comme le coeur et les poumons). Une stratégie appropriée de gestion de cadres de désignation est donc nécessaire, ainsi qu'une attention particulière lors de la réalisation graphique.

La méthode consistant à définir plusieurs cadres par pièce apparait fort compliquée et gourmande en place mémoire par rapport aux faibles avantages qu'elle propose. Définir des zones de désignation autres que des cadres est hors de question dans notre cas.

Notre stratégie arbore un visage multiple suivant les buts à atteindre et les types de désignation.

Lorsque le pixel désigné appartient à deux cadres, on traite ce cas comme si l'utilisateur voulait désigner le plus petit des cadres. Dans le cas du coeur dont le cadre de désignation C est contenu dans celui des poumons P, "cliquer" dans C signifie bien désigner le coeur, même s'il est contenu dans le cadre de désignation des poumons. Au contraire, "cliquer" sur le coeur est bien considéré comme une erreur à la question de montrer les poumons, même si le point qu'on a désigné se trouve aussi dans le cadre des poumons.

Pour d'autres types de désignation, il devrait suffire de "cliquer" à l'intérieur du cadre demandé pour obtenir le renforcement positif. C'est le cas pour la désignation par système.

La couleur du pixel désigné ne sera pas prise en compte lorsque la silhouette est vide. Il suffit de "cliquer" dans le cadre de désignation pour que l'organe désigné apparaisse un moment.

Dans certains cas, il est impossible de procéder de cette manière. C'est le cas lorsqu'il faut désigner les veines ou les artères dans le système circulatoire. La différentiation des deux pièce se fera, non plus sur base des cadres de désignation, mais sur base de la couleur. Pour utiliser cette technique, il faut une couleur unique et différente pour chaque pièce du système. Si on caractérise,par exemple, les veines par le bleu et les artères par le rouge, "cliquer" sur un pixel rouge signifie désigner une artère, et les veines pour un pixel bleu. Nous nomons cette technique: *désignation par couleur* en opposition à la *désignation par cadre.*

Les différentes icônes utilisées dans le logiciel sont similaires à celles utilisées dans le logiciel "Corps" car les utilisateurs les discernaient bien, les comprenaient bien, comme nous avons pu le constater lors de l'utilisation de "Corps" par des utilisateurs des institutions suisses et belges.

Nous avons également modifié les messages textuels proposés à l'utilisateur que nous avions spécifié lors des spécifications des messages. Nous avons constaté (lors de nos rencontres avec les utilisateurs) que certaines phrases semblaient mal appropriées. Ainsi nous avons notamment

changer la question "veux-tu montrer le parcours de l'oxygène?" par "veux-tu montrer par où passe l'oxygène". Car le terme parcours est un terme qui est mal compris sans explication extérieure. Nous nous sommes péoccupés à utiliser des phrases courtes dans lesquelles le terme le plus important apparaissaient en dernier lieu.

La nécessité des énoncés vocaux, présents dans "Corps", nous est apparue très rapidement. Beaucoup d'utilisateurs rencontrés ne présentaient pas un acquis suffisant en lecture pour se passer des énoncés vocaux.

6.5 Conclusions

Nous venons de donner notre démarche de spécification appliquée au domaine. Il nous reste, dans un dernier chapitre, à donner une interprétation formelle aux résultat de cette application.

Chapitre 7 : Les spécifications de l'outil

Introduction

L'objet de ce chapitre est de présenter les spécifications de l'outil réalisé. Dans la présentation générale de l'outil nous montrerons qu'il s'agit en fait de deux logiciels. Les sections suivantes présenteront les spécifications de chacune des applications.

7.1. Présentation générale de l'outil

7.1.1. diversité de la population cible

L'outil d'apprentissage, s'il peut être adapté, doit rencontrer les caractéristiques de la population cible comme nous l'avons indiqué précédemment. On devrait pouvoir établir à quelle population est destiné l'outil en s'aidant d'une classification. En ce qui concerne les personnes handicapées mentales, une classification est une tâche ardue, et même dangereuse. Nous l'avons indiqué au chapitre 4. Un handicap reflète toujours un manque d'harmonie entre l'individu et son entourage et est donc un phénomène extrêmement varié et individuel qu'il est impossible de cataloguer en se basant sur le seul diagnostic d'une incapacité fonctionnelle [Lepoutre, 1990]. La diversité de la population cible entraîne la nécessité de proposer des outils d'apprentissage flexibles.

7.1.2. un outil ouvert, une préoccupation primordiale

Pour atteindre cet objectif, plusieurs moyens peuvent être mis en oeuvre.

Un de ces moyens est la réalisation de logiciels hautement paramètrés. Cela signifie qu'il faut offrir aux éducateurs la possibilité de configurer le logiciel de multiples manières. L'ensemble des paramètres sur lesquels ils peuvent agir permettra d'individualiser l'ordinateur. Chaque individu devra exécuter les exercices qui lui sont le mieux adaptés compte tenu de ses caractéristiques intellectuelles, mentales et physiques... Plus le nombre de paramètres est important, plus la démarche d'individualisation sera précise. D'un autre côté, le travail et le temps de paramètrisation demandé aux éducateurs va crescendo avec le nombre de paramètres. Dés lors il est important de trouver un juste milieu. Il arrive qu'un logiciel ne soit pas utilisé à cause du nombre élevé de paramètres qu'il met en jeu et la complexité de leur travail de paramètrisation. Comme le nombre de paramètres ne peut être trop important, il est utile de bien les choisir. Ce choix doit se faire en collaboration avec les éducateurs, psychologues.

Une solution idéale dans ce contexte serait que la machine réalise elle-même les modifications aux paramètres sur base des réponses apportées par l'apprenant, c'est-à-dire d'élaborer un système adaptatif tel que nous l'avons défini. Cette solution idéale pour l'éducateur nécessite cependant des techniques d'Intelligence Artificielle avec tous les problèmes mentionnés.

Un deuxième moyen est de réaliser un logiciel suffisamment ouvert afin que les non informaticiens (les éducateurs, par exemple) puissent modifier certains éléments du logiciel. Dans le cadre de notre projet, une personne non spécialisée peut agir si elle le souhaite sur trois de ces éléments.

Le premier est constitué de *messages textuels possibles* qui sont utilisés par le logiciel et affichés à l'écran. Ces messages se trouvant dans un fichier texte, peuvent se modifier facilement au moyen d'un éditeur. Ainsi les messages peuvent être écrits dans une autre langue, le vocabulaire

peut être par exemple adapté d'après le pays.

Le deuxième élément est constitué des messages utilisés dans la trace de l'interaction. Rappelons que toutes les manipulations effectuées par l'utilisateur sont sauvées par l'application. Une manipulation de l'utilisateur se traduit par un message textuel et l'ensemble de ces messages forme la trace textuelle. L'ensemble des *messages textuels possibles*, correspondant aux manipulations possibles, est sauvé dans un fichier texte de la même manière que les messages affichés à l'écran. Il est donc tout aussi aisé d'y apporter des modifications.

Le troisième élément pouvant être modifié par un non informaticien est l'ensemble des *messages vocaux* destinés à l'utilisateur. Comme la possibilité étant offerte à des non spécialistes de pouvoir apporter des modifications aux messages affichés à l'écran, la possibilité de modifier les messages vocaux leur est également offerte. Certains messages textuels correspondent à certains messages vocaux. Dès lors un changement au niveau textuel peut facilement se réaliser également au niveau de ces messages vocaux, si toutefois une digitalisation des messages vocaux désirés est disponible.

Un troisième moyen est de proposer une boîte à outils. Cela signifie la réalisation d'un ensemble de programmes destinés à réaliser plus aisément des logiciels d'apprentissage. Ces outils de haut niveau ne nécessitant pas de grandes connaissances en informatique, peuvent être utilisés par un plus grand nombre de personnes et permettre la création de logiciels spécifiques répondant aux besoins de chacun.

Dans notre application, nous avons opté pour la paramètrisation du logiciel agrémentée de la possibilité offerte de modifier des fichiers utilisés en entrée par le programme. Nous allons montrer dans la section suivante comment s'articule la paramètrisation dans le logiciel global.

7.1.3 L'outil général

Dans le soucis de réaliser un outil "ouvert", flexible, l'outil général se composera de deux éléments, deux logiciels.

Le premier est le logiciel "Corps interne". Ce dernier propose et gère les exercices qui sont exécutés par les personnes handicapées mentales.

Le deuxième logiciel est le logiciel "Paramétrage". Ce dernier contrairement à Corps interne est destiné aux éducateurs et utilisé uniquement par ces derniers. Il leur permet de donner des valeurs à une série de paramètres afin de configurer les exercices que l'apprenant devra réaliser.

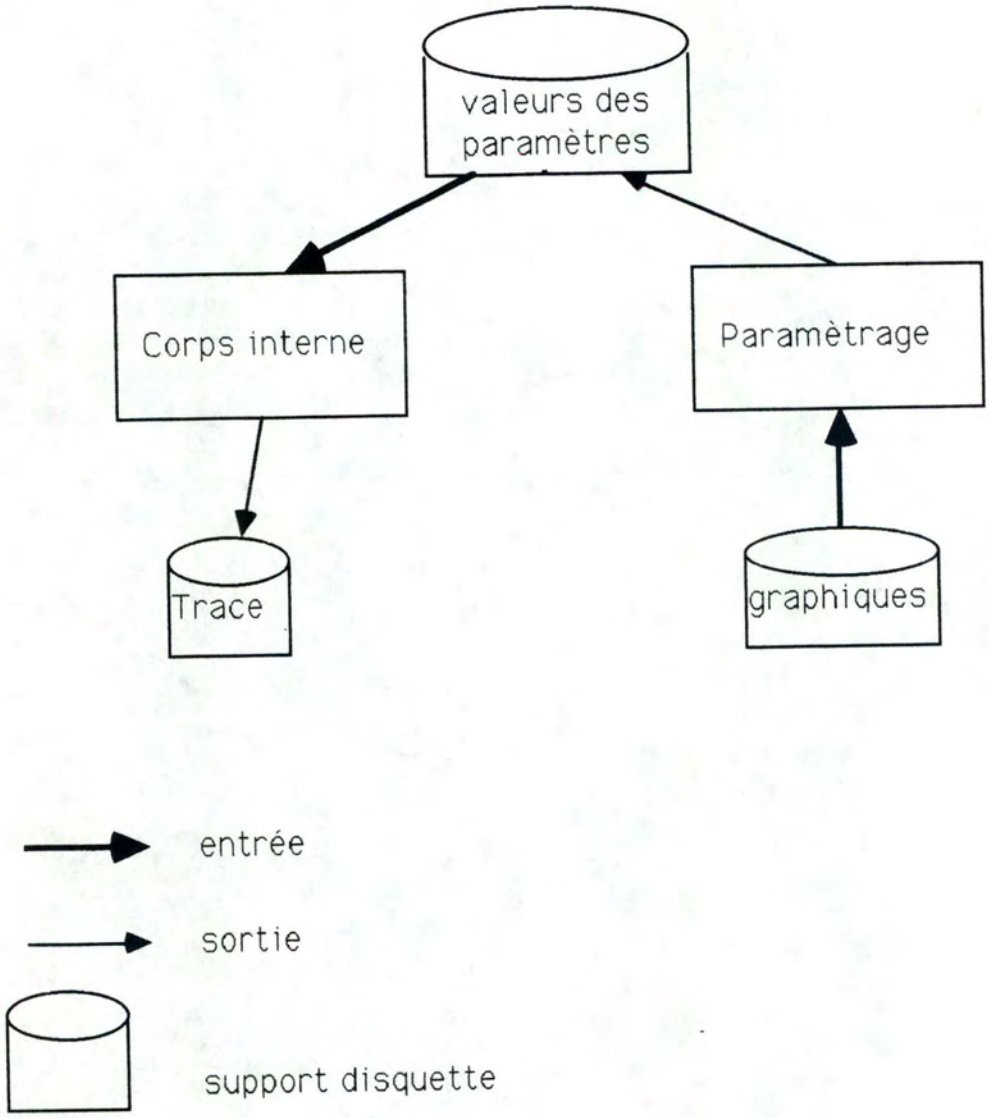


figure 7.1 : l'interaction des logiciels

Ces deux logiciels vont s'échanger des informations. La figure 7.1. montre quelles sont ces informations en entrée et en sortie qu'ils produisent.

Le logiciel Paramétrage reçoit en entrée les graphiques disponibles supportant des exercices. L'éducateur pourra opérer un choix parmi tous les graphiques proposés et donner une valeur à un ensemble de paramètres. Paramétrage recopie sur une disquette appelée "USER" les graphiques retenus ainsi que les valeurs des paramètres.

Le logiciel Corps interne reçoit en entrée les données, les graphiques se rapportant aux exercices à proposer à l'utilisateur ainsi que les paramètres se trouvant sur la disquette "USER". Il produit en sortie la "trace" de l'interaction entre le logiciel et l'apprenant. Cette "trace" est recopiée sur la disquette "USER".

Le processus associé à Paramétrage correspond à la phase d'initialisation et le processus associé à Corps interne est l'interaction proprement dite avec la personne ayant une déficience mentale.

7.2. Spécification du logiciel "Corps interne"

7.2.1. Présentation du logiciel

Le logiciel "Corps interne" propose deux types d'exercices à l'utilisateur. Un exercice de type "désignation" et un exercice de type "parcours".

Dans l'exercice de type "désignation", l'ordinateur énonce successivement les différentes parties d'un graphique à l'utilisateur et celui-ci doit situer correctement les parties énoncées, qu'il s'agisse d'organes dans un système, de noms d'organes, ou de systèmes parmi d'autres . Il est aussi possible de connaître le nom d'organe désigné par l'utilisateur dans un exercice de type pseudo-passif.

Dans l'exercice de type "parcours", il est demandé à l'utilisateur de montrer le parcours d'un symbole (oxygène, boulette de viande, eau...) dans une silhouette représentant un système (ou deux systèmes). Comme nous l'avons décrit en détail dans l'interface concrète, le renforcement négatif correspond à un feed-back sonore et textuel négatif, un flash de l'écran, et le recommencement de la désignation au début du parcours. Le feed-back positif est également sonore (musique),et visuel (déplacement d'un symbole et/ou coloration du parcours correct effectué).

Le logiciel propose ces exercices dans une session. Une session de trois exercices comprend, par exemple, un exercice de désignation , suivi d'un autre, puis d'un parcours.

Une caractéristique du logiciel est le sauvetage qu'il réalise de toutes les manipulations que l'utilisateur effectue au cours d'une séance d'exercices. Cette trace peut être consultée ultérieurement pour en analyser le contenu.

7.2.2. Spécification des scénarios et de l'interface concrète

Dans cette sous-section, les interfaces retenues seront spécifiées. Nous présenterons la méthode utilisée pour ensuite spécifier les dialogues des exercices de désignation et de parcours ainsi que la session d'exercice.

7.2.2.1 La méthode utilisée

Pour chaque type d'exercices, nous indiquerons les paramètres disponibles et comment ils influencent l'interaction. Nous présenterons alors graphiquement les scénarios détaillés et nous terminerons par la spécification des messages interactifs dégagés dans les scénarios.

Pour les messages interactifs, nous donnerons une définition, des opérations associées, sa représentation concrète ainsi que les actions physiques associées aux opérations définies. Ces deux dernières caractéristiques sont du domaine de l'interface concrète.

Les conventions utilisées pour représenter graphiquement les scénarios sont les suivantes:

Les messages sont représentés par des boîtes rectangulaires dont le coin inférieur droit est barré. L'enchaînement du dialogue est construit par les opérations que l'on peut effectuer sur les messages mais également par les paramètres de l'interaction.. Ces règles d'enchaînement liées aux paramètres sont représentées par des boîtes rectangulaires plus allongées.

Une flèche aboutissant à un message indique son activation. A ce moment, l'utilisateur peut réaliser les opérations définies sur ce message. Une opération particulière constitue la clôture du message; il devient alors inactif et la flèche qui en part indique l'enchaînement lié à sa clôture.

Un message peut également être activé lorsque le traitement associé à une opération effectuée sur un autre message l'indique (de même pour sa désactivation).

Il est entendu que plusieurs messages peuvent être actifs simultanément. Une condition "ET" en amont des messages indique une activation multiple.

On utilise également une structure itérative. Le dialogue compris entre les triangles indiquant le début et la fin de l'itération se répètera jusqu'à ce que la condition de sortie soit vérifiée.

La dernière convention concerne les points noirs au départ des flèches d'enchaînement du dialogue. Cela signifie qu'aucune opération n'est définie sur les messages et que sa clôture est immédiate et automatique.

7.2.2.2 La désignation

A. Scénario de base

Dans l'exercice de type "désignation", la machine énonce successivement les noms des différentes parties d'un graphique présenté à l'utilisateur. Ce dernier doit, à chaque énoncé, situer correctement la partie. Le graphique peut représenter une silhouette humaine contenant des organes regroupés en un ou deux systèmes anatomiques. Le graphique peut

également montrer les représentations de systèmes anatomiques. La présentation de graphiques avec des organes ou avec les dessins des systèmes dépend en fait de la liste des énoncés choisie par l'éducateur dans l'ensemble de graphiques disponibles. Le graphique de la silhouette présenté à l'utilisateur peut être dépourvu de ses organes. Dans ce cas l'utilisateur doit indiquer l'endroit de l'organe demandé. Le graphique présenté peut également être accompagné des noms des organes présent sur le graphique de la silhouette. Dans ce cas ce sont les noms des organes que l'utilisateur doit indiquer. Lors que le graphique de la silhouette est représenté à l'utilisateur sur l'écran, quatre types d'interactions sont possibles : l'apprentissage, l'évaluation, l'interaction nommée pseudo-passive et celle appelée passive. Un paramètre permet à l'éducateur de définir le type choisi.

Dans le mode passif, comme sa désignation le suggère, l'utilisateur est passif; la seule chose qu'il doit faire est de regarder l'écran. Le logiciel énonce chacun des noms d' organe ou des systèmes anatomiques du graphique contenus dans la liste. Pendant cet énoncé, le dessin de l'organe, du système ou le nom de l'organe est mis en évidence.

Dans le mode pseudo-passif, l'utilisateur est plus impliqué que dans le mode passif mais moins que dans les deux autres modes : apprentissage et évaluation. La machine demande à l'utilisateur d'indiquer le dessin d'un organe, d'un système, d'un nom d'organe. Lorsque l'utilisateur à montrer une de ces choses, l'ordinateur lui indique ce que cela représente.

Le mode apprentissage est le mode où l'utilisateur doit répondre à une question posée par la machine à savoir de montrer un organe, un nom d'organe, un système anatomique, la place d'un organe. En cas de réponse incorrecte (la réponse se matérialise par un clic de la souris), l'ordinateur met en évidence la réponse avant de faire de même pour la réponse correcte(l'ordinateur corrige). La question qui a entraîné une réponse incorrecte est reposée immédiatement avant d'être reposer une nouvelle fois plus tard.

Le mode évaluation est semblable au précédent si ce n'est que la machine ne corrige plus. L'utilisateur est libre de répondre ce qu'il entend. Ce mode correspond à des exercices dont le but est de tester les connaissances comme cela se fait lors d'examens.

B. Paramètres d'une désignation

Paramètre 1: le type d'interaction qui peut avoir trois valeurs:

-*interaction de type apprentissage.* Les réponses fournies par l'élève aux questions posées sont corrigées automatiquement par le processus des feed-backs.

-*interaction de type évaluation*: Dans ce cas, la machine ne corrige plus les réponses de l'élève. Il peut faire ce qu'il veut. On ne fait plus appel aux feed-backs.

-*interaction de type pseudo-passif*: L'accent est ici mis sur la possibilité donnée à l'utilisateur de découvrir par exploration la matière qu'on désire qu'il maîtrise. L'association est faite entre un aspect particulier de cette matière choisi par l'utilisateur, et la question qui peut lui être associée ultérieurement, dans un autre type d'interaction. Cette association constitue le feed-back.

-*interaction de type passif*: L'utilisateur n'apparaît ici, non plus comme un acteur, mais comme un observateur. C'est la machine qui effectue elle-même les exercices.

Paramètre 2: *l'ordre d'énumération* détermine l'ordre dans lequel les organes sont demandés à l'utilisateur. Il peut correspondre à un ordre logique lié à la position de l'organe dans le système (ordre défini dans le modèle du domaine), ou à un ordre aléatoire.

Paramètre 3: *énoncé vocal* des parties du graphique (oui/non).

Paramètre 4: *nombre d'itérations* sur la technique de désignation avant de reposer une partie mal désignée (1/2/3/4/5).

Paramètre 5: *silhouette vide ou non.* Il s'agira de désigner un organe visible dans un système ou de désigner l'endroit où il se situe dans une silhouette vide.

Paramètre 6: une *animation* peut être proposée en plus des feed-backs usuels. C'est le cas rencontré lors de la désignation par système.

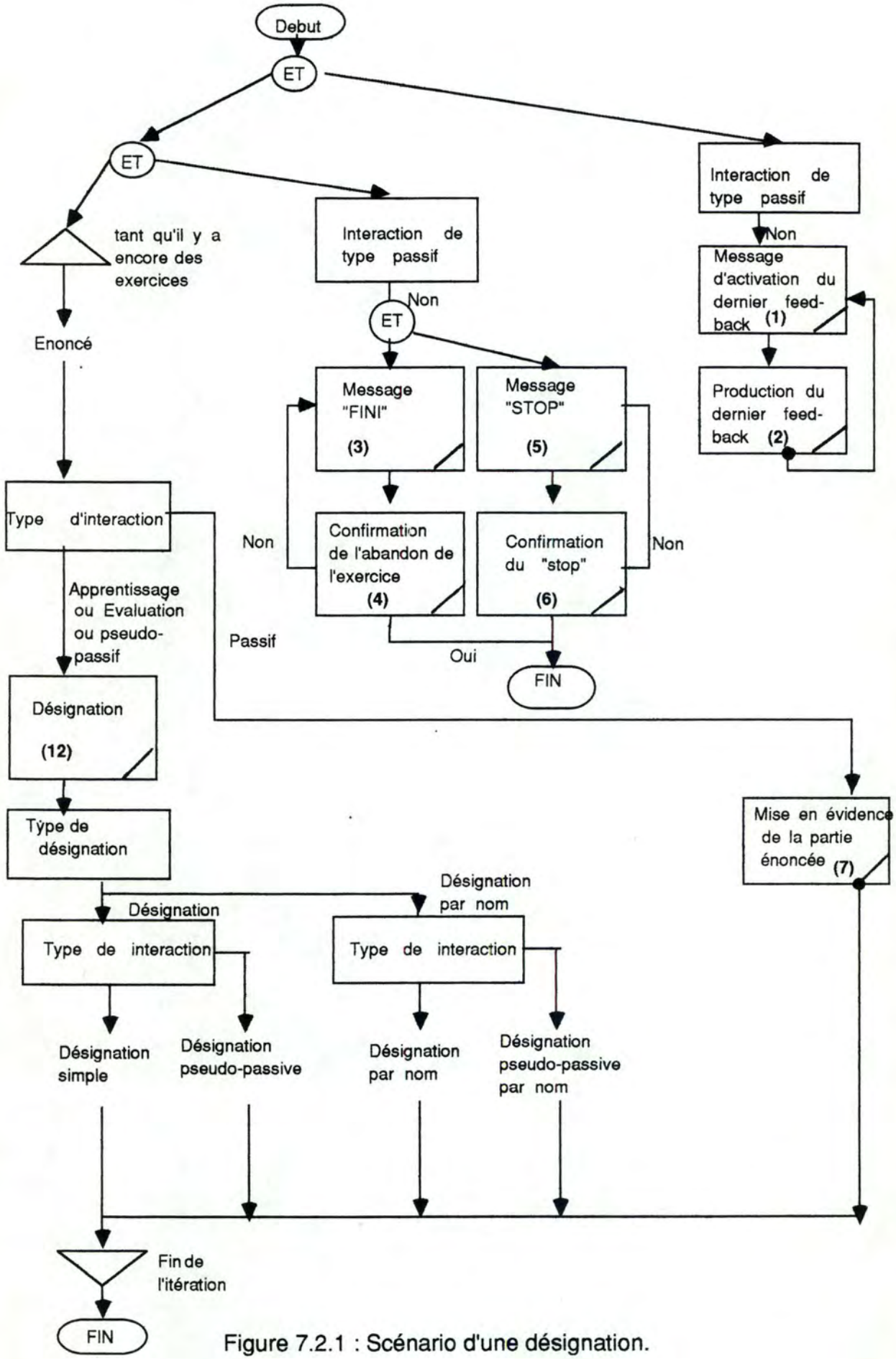


Figure 7.2.1 : Scénario d'une désignation.

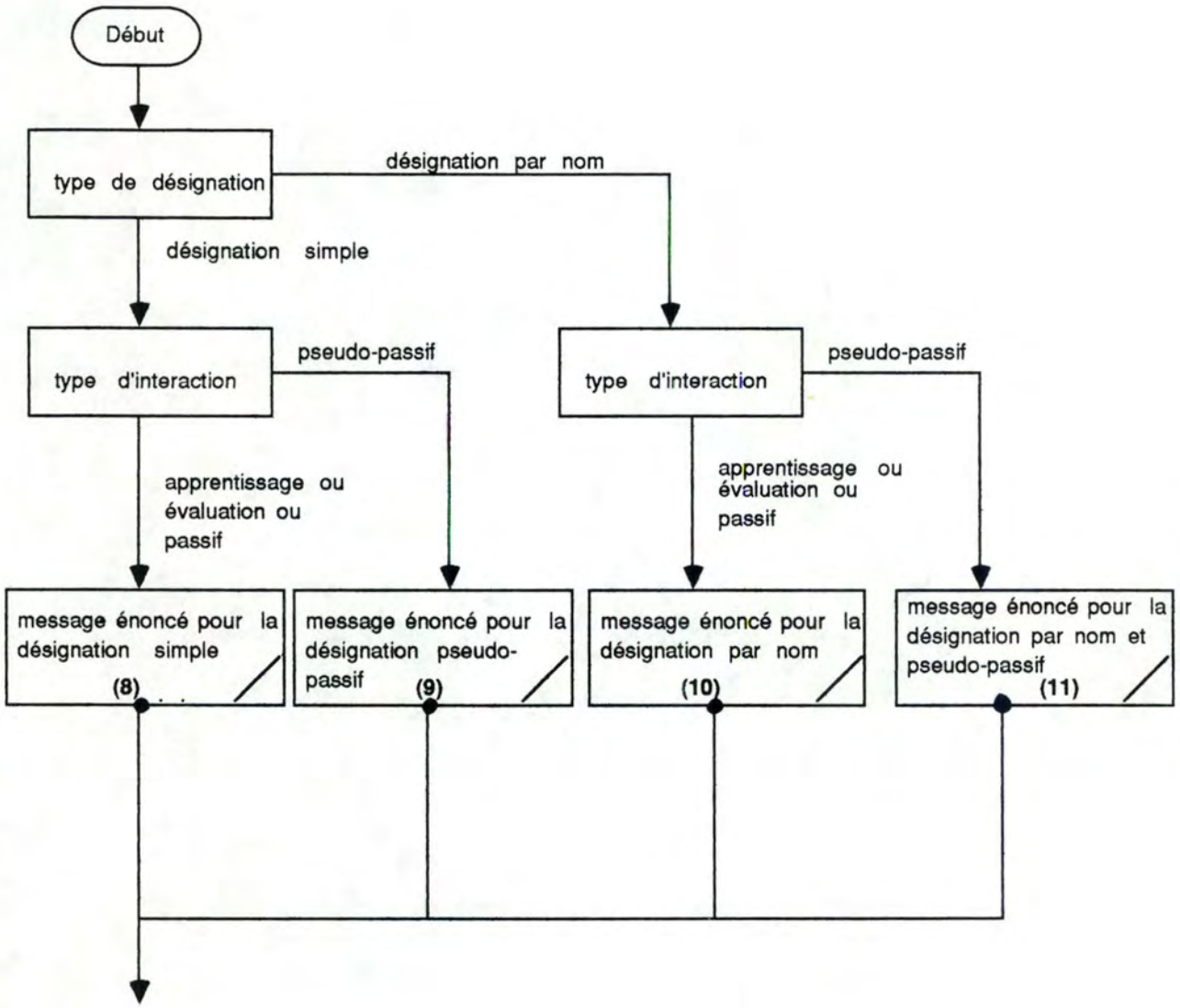


Figure 7.2.2 : Enoncé.

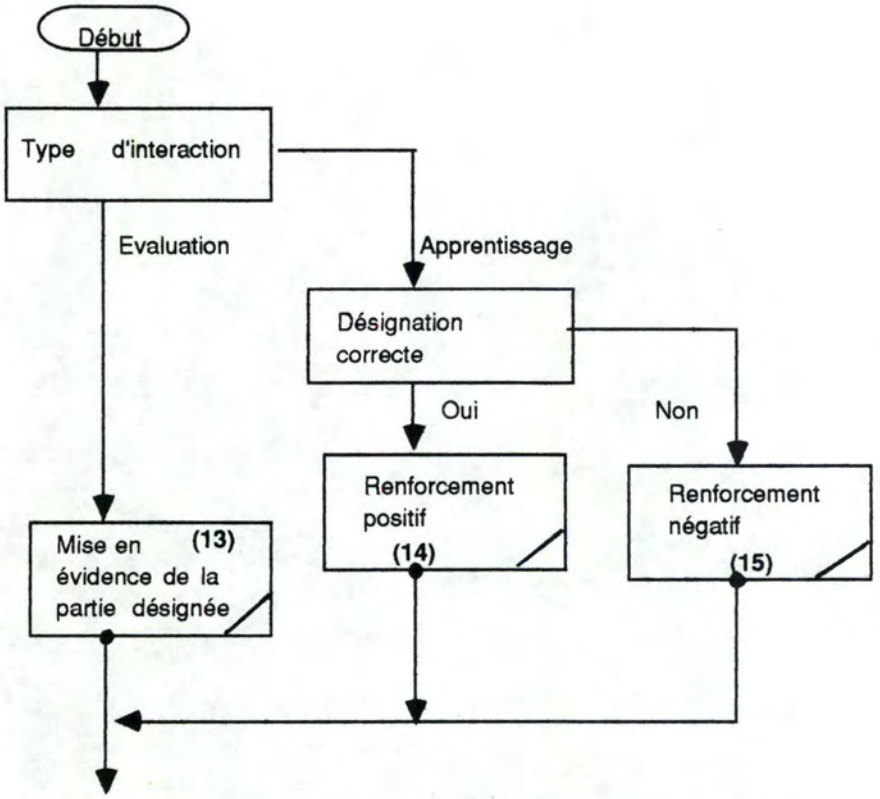


Figure 7.2.3 : Désignation simple

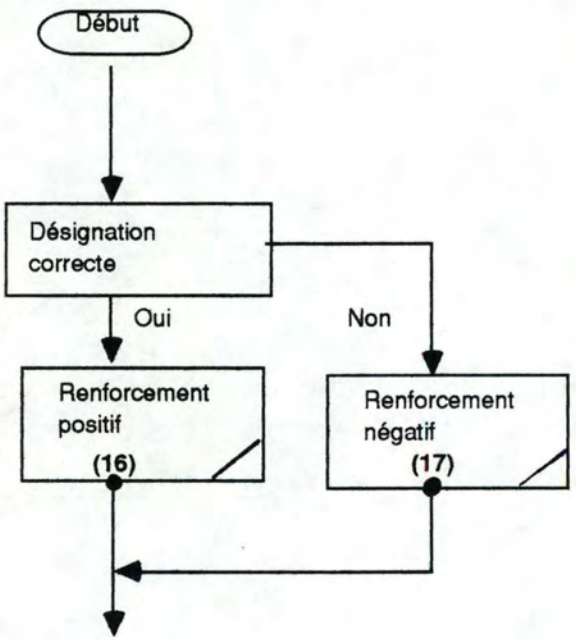


Figure 7.2.4 : Désignation pseudo-passive

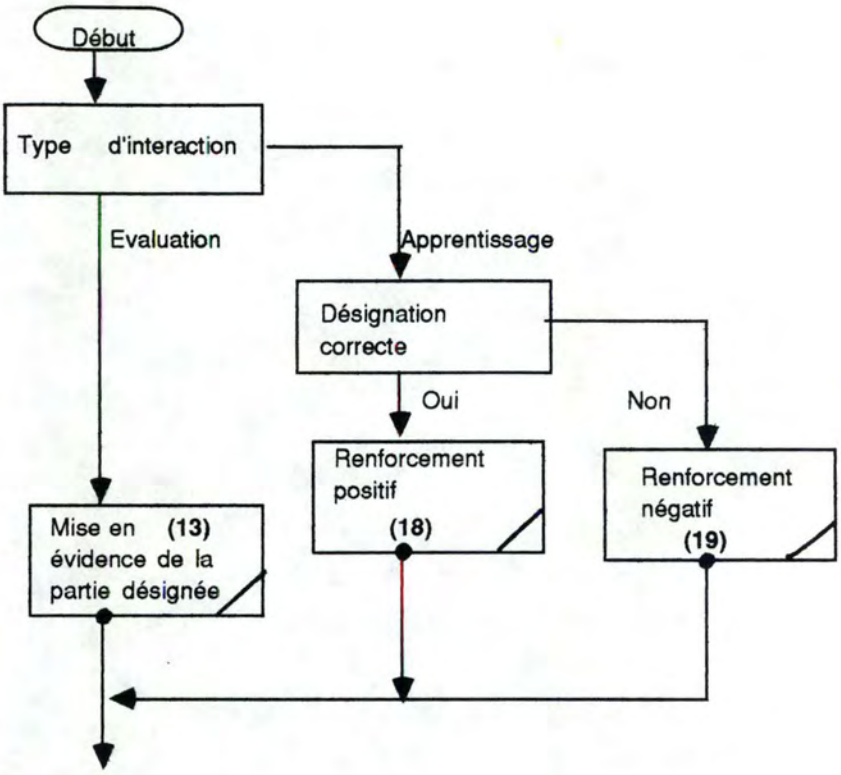


Figure 7.2.5 : Désignation par nom

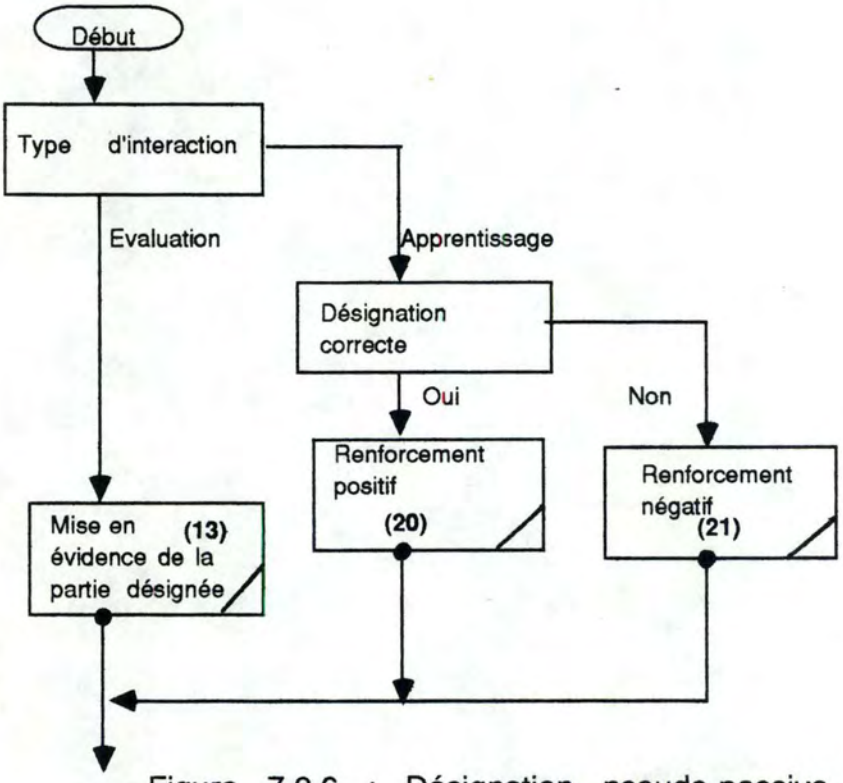


Figure 7.2.6 : Désignation pseudo-passive par nom

Paramètre 7: la détection de la pièce que l'utilisateur a voulu désigner à partir de l'endroit où il a "cliquer" peut se faire par gestion des *cadres* de désignation des organes ou par gestion des *couleurs* des organes. Ce paramètre peut être dépendant de la représentation graphique que l'on veut utiliser.

Paramètre 8: les *types de désignation simple ou par nom* peuvent être choisis. Soit on demande à l'utilisateur de montrer un organe dont le nom est énoncé, soit, c'est le nom d'un organe mis en évidence dans la silhouette qui doit être trouvé.

C. Le scénario détaillé

1. Représentation graphique

La représentation graphique de l'exercice de type "désignation" est représentée par les figures 7.2.1, 7.2.2, 7.2.3, 7.2.3. 7.2.4., 7.2.5 et 7.2.6.

2. Spécification des messages internes

-1- Message d'activation du dernier feed-back

Définition: message permettant de reproduire le dernier feed-back.
Opération: clôture du message.
Représentation: icône représentant une oreille (écran 1).
Action physique: Cliquet de la souris pour activer l'oreille.

-2- Production du dernier feed-back

Définition: Message d'aide qui consiste à reproduire le dernier feed-back produit.
Opération: Néant, clôture automatique.
Représentation: Message vocal digitalisé et mise en évidence de la pièce demandée dans la désignation par nom.
Action physique: Néant.

-3- Message "FINI"

Définition: message permettant de stipuler l'abandon de l'exercice.
Opération: clôture du message. Traitement associé : désactivation des messages courants excepté celui d'activation du dernier feed-back.
Représentation: icône représentant les caractères "FIN" (écran 1).
Action physique: cliquet de la souris pour activer l'icône.

-4- Confirmation de l'abandon de l'exercice

Définition: message permettant de confirmer l'abandon de l'exercice.
Opération: clôture du message en confirmant ou en infirmant l'abandon de l'exercice.
Traitement associé: réactivation des messages courants si infirmation de l'abandon de l'exercice.
Représentation: le message textuel et vocal "Veux-tu finir?",un bouton contenant le texte "OUI" et un autre contenant "NON" (écran 8).
Action physique: cliquet de la souris pour activer un des deux boutons.

-5- Message "STOP"

Définition: message permettant de stipuler l'arrêt de l'exercice (arrêt de la session, qui peut être reprise plus tard).
Opération: clôture du message. Traitement associé: désactivation des messages courants excepté celui d'activation du dernier feed-back.
Représentation: icône représentant les caractères "STOP" (écran 1).
Action physique: cliquet de la souris pour activer l'icône.

-6- Confirmation du stop

Définition: message permettant de confirmer l'arrêt de l'exercice.
Opération: clôture du message en confirmant ou en infirmant l'arrêt de l'exercice. Traitement associé: réactivation des messages courants si infirmation de l'abandon de l'exercice.
Représentation: le message textuel et vocal "Veux-tu stopper?",un bouton contenant le texte "OUI" et un autre contenant "NON" (même genre que l'écran 8).
Action physique: cliquet de la souris pour activer un des deux boutons.

-7- Message Mise en évidence de la partie énoncée

Définition: message visualisant sur le graphique la partie énoncée.
Opération: Néant clôture automatique.
Représentation: la parti énoncée du graphique clignote.
Action physique: Néant.

-8- Message énoncé pour la désignation simple

Définition: production du feed-back de l'organe qui doit être énoncé.
Opération: néant, clôture automatique.
Représentation: affichage textuel, dans le partie inférieure droite de l'écran: "Veux-tu montrer *énoncé*?". Si le paramètre permet l'énoncé vocal, l'énoncé est également produit vocalement. L'organe clignote.
Action physique: néant.

-9- Message énoncé pour la désignation pseudo-passive

Définition: production du feed-back de l'organe qui doit être énoncé.
Opération: néant, clôture automatique.
Représentation: affichage textuel, dans le partie inférieure droite de l'écran: "Veux-tu montrer un organe?". Si le paramètre permet l'énoncé vocal, l'énoncé est également produit vocalement. L'organe clignote.
Action physique: néant.

-10- Message énoncé pour la désignation par nom

Définition: production du feed-back de l'organe dont le nom doit être énoncé.
Opération: néant, clôture automatique.
Représentation: affichage textuel, dans le partie inférieure droite de l'écran: "Veux-tu montrer le nom de *énoncé*?". Si le paramètre permet l'énoncé vocal, l'énoncé est également produit vocalement. L'organe clignote (écran 2).
Action physique: néant.

-11- Message énoncé pour la désignation par nom pseudo-passive

Définition: production du feed-back du nom demandé.
Opération: néant, clôture automatique.
Représentation: affichage textuel, dans le partie inférieure droite de l'écran: "Veux-tu montrer le nom d'un organe?". Si le paramètre permet l'énoncé vocal, l'énoncé est également produit vocalement (écran 3).
Action physique: néant.

-12- Message de désignation

Définition: message permettant de désigner une partie
Opération: clôture du message.
Représentation: le support est la silhouette servant de base à l'exercice.
Action physique: cliquet de la souris.

-13- Mise en évidence de la partie désignée

Définition: message visualisant sur le graphique la partie désignée.
Opération: néant, clôture automatique.
Représentation: la partie désignée clignote
Action physique: cliquet gauche de la souris.

-14- Renforcement positif, désignation simple

Définition: message de renforcement signalant la désignation correcte d'un énoncé.
Opération: néant, clôture automatique.
Représentation: message sonore digitalisé
Action physique: néant.

-15- Renforcement positif, désignation pseudo-passive

Définition: message de renforcement signalant la désignation correcte d'un énoncé.
Opération: néant, clôture automatique.
Représentation: message sonore digitalisé, coloration de l'organe désigné qui était en gris (écran1).
Action physique: néant.

-16- Renforcement positif, désignation par nom

Définition: message de renforcement signalant la désignation correcte d'un énoncé.
Opération: néant, clôture automatique.
Représentation: message sonore digitalisé (écran 2).
Action physique: néant.

-17- Renforcement positif, désignation pseudo-passif par nom

Définition: message de renforcement signalant la désignation correcte d'un énoncé.
Opération: néant, clôture automatique.
Représentation: coloration du nom désigné (écran 3).

Action physique: néant.

-18- Renforcement négatif, désignation simple

Définition: message de renforcement signalant la désignation incorrecte d'un énoncé.
Opération: néant, clôture automatique.
Représentation: message sonore digitalisé, clignotement de la partie désignée puis de la partie demandée, avec les messages vocaux et textuels: "Tu as montré *énoncé1*","J'ai demandé *énoncé2*".
Action physique: néant.

-19- Renforcement négatif, désignation pseudo-passive

Définition: message de renforcement signalant la désignation incorrecte d'un énoncé.
Opération: néant, clôture automatique.
Représentation: message sonore digitalisé, messages vocal et textuel: "Tu as déjà montré cet organe".
Action physique: néant.

-20- Renforcement négatif, désignation par nom

Définition: message de renforcement signalant la désignation incorrecte d'un énoncé.
Opération: néant, clôture automatique.
Représentation: message sonore digitalisé, clignotement du nom désigné puis du nom demandé, avec les messages vocaux et textuels: "Tu as montré","J'ai demandé *le nom*".

Action physique: néant.

-21- Renforcement négatif, désignation pseudo-passive par nom

Définition: message de renforcement signalant la désignation incorrecte d'un énoncé.
Opération: néant, clôture automatique.
Représentation: message sonore digitalisé, messages vocal et textuel: "Tu as déjà montré ce nom".
Action physique: néant.

7.2.2.3 Le parcours

A. Scénario de base

Dans l'exercice de type "parcours", la machine demande à l'utilisateur de désigner sur un graphique représentant une silhouette humaine le parcours d'un substance : parcours d'une boulette à travers le système digestif, parcours de l'oxygène à travers le système respiratoire, parcours du sang à travers le système circulatoire ainsi que le parcours d'autres substances, parcours de l'oxygène à travers le système respiratoire et le système circulatoire, ...etc. La machine peut (suivant le souhait de la persoone qui spécifie les exercices) indiquer le pacours en début et/ou à la terminaison de l'exercice. Elle peut également indiquer le point le point de début et le point de sortie du parcours. Deux modes d'interaction peuvent être (via la paramétrisation des exercices) choisis : apprentissage et évaluation.

Le mode apprentissage, comme sa dénomination l'indique, a pour but d'apprendre à l'utilisateur les parcours des différentes substances dans l'organisme. L'utilisateur indique le parcours au moyen de la souris. L'ordinateur vérifie de la véracité de cette indication. En cas d'erreur l'ordinateur, après avoir indiqué le type d'erreur commise, propose à l'utilisateur de recommencer l'exercice. Les erreurs qui peuvent être commises sont de trois types. Nous avons déjà expliquer la manière dont l'utilisateur devait s'y prendre pour indiquer le parcours. Le premier type d'erreur concerne un retour en arrière dans le parcours. Le deuxième correspont à s'écarter du parcours et enfin le dernier à omettre une partie du parcours (omettre de désigner un organe). A chaque désignation correcte d'une partie du parcours, l'ordinateur donne un feed-back pouvant être de trois natures. Le premier type consiste en le déplacement du symbole, représentant la substance qui parcours le trajet demandé, de l'endroit du parcours où l'utilisateur était arrivé précédemment jusqu'à l'endroit qu'il vient d'indiquer. Le deuxième est identique au premier si ce n'est le remplacement du symbole par une coloration qui dépend du graphique de base de l'exercice. Le dernier est l'union des deux précédents.

Le mode évaluation laisse l'utilisteur indiquer ce qu'il désire sans, comme dans le mode apprentissage, le contrainte à recommencer en cas d'erreur. Les différents feed-back du mode apprentissage se retrouve dans le mode évaluation.

B. Paramètres du parcours

Paramètre 1: le *type d'interaction* pour cet exercice n' a que deux valeurs possibles: évaluation et apprentissage.

Paramètre 2: Le *parcours au début* de l'exercice est effectué par la machine ou non, comme aide à l'utilisateur.

Paramètre 3: les *points de départ et d'arrivée* sont désignés ou non comme aide à l'utilisateur. Il ne doit plus désigner le point de départ.

Paramètre 4: le *type de feed-back visuel* est ici défini. Il s'agit soit du déplacement du symbole, soit de la coloration, soit des deux en même temps.

Paramètre 5: *le parcours en fin* est présenté comme renforcement à un parcours complètement désigné ou non.

Paramètre 6: *l'énoncé vocal* est activé ou non, comme dans l'exercice de désignation.

C. Le scénario détaillé

1. Représentation graphique

La représentation graphique de l'exercice de type "parcours" est représenté par les figures 7.3.1 et 7.3.2.

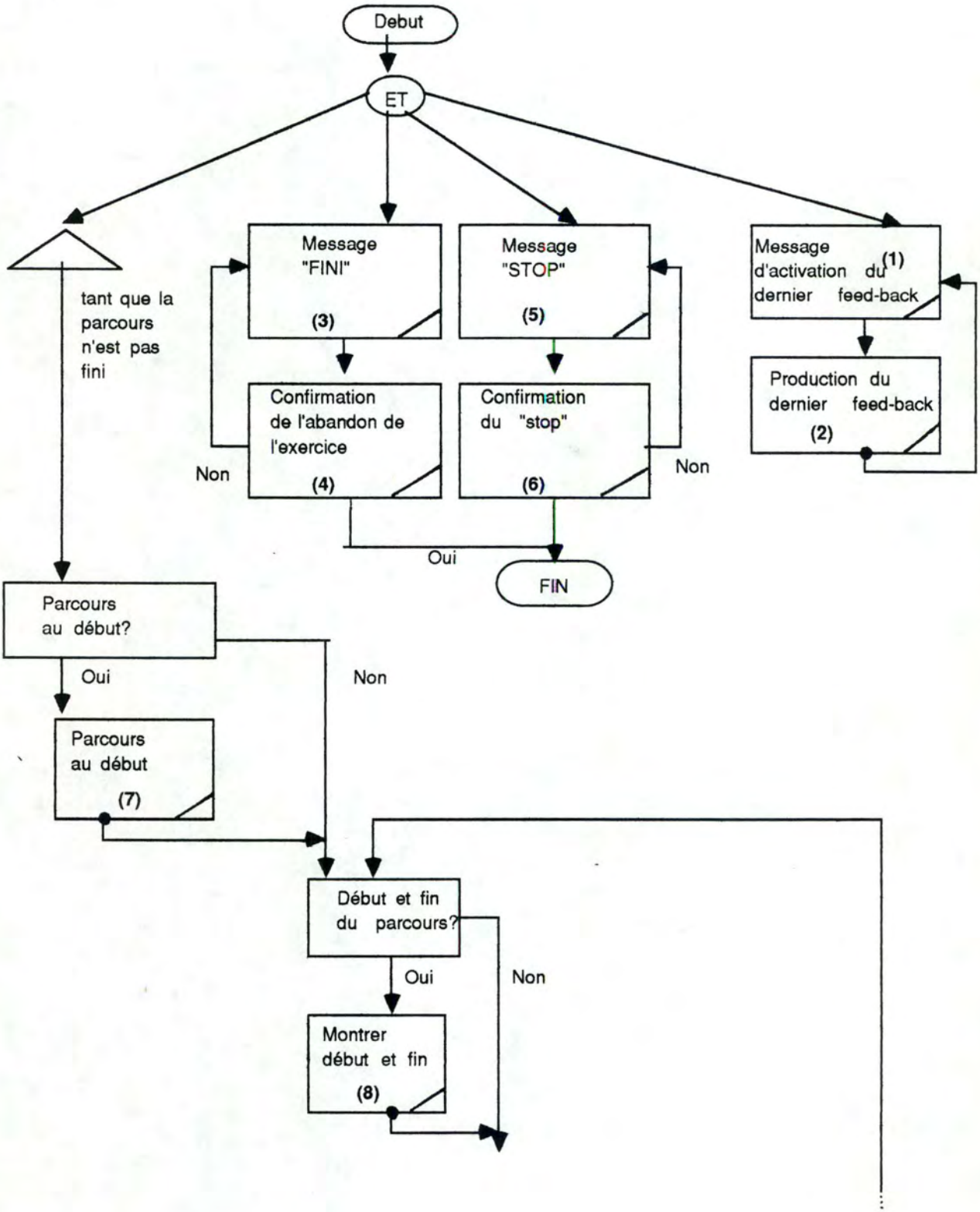


Figure 7.3.1 Scénario du parcours (début).

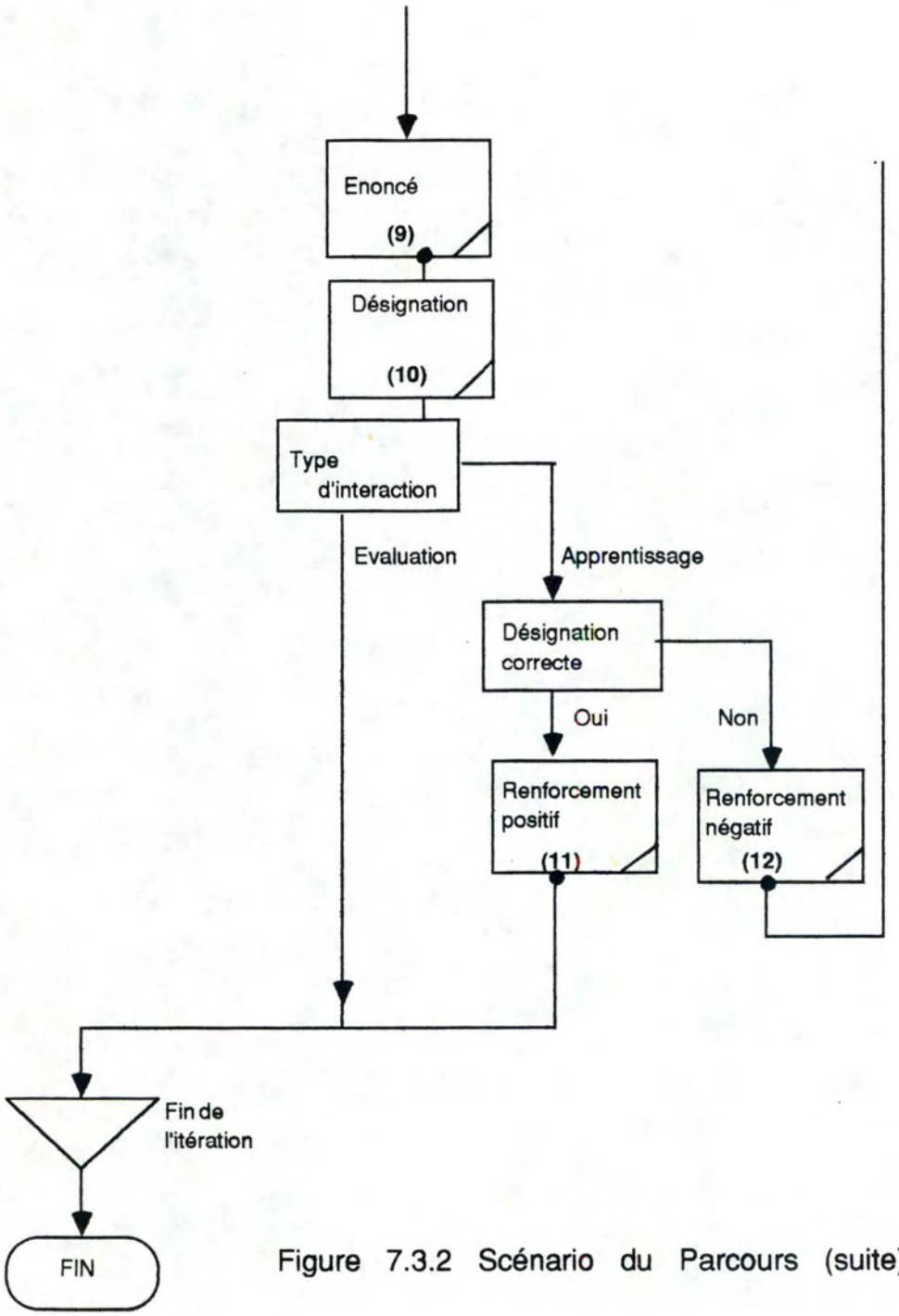


Figure 7.3.2 Scénario du Parcours (suite).

2. Spécification des messages internes

-1- Message d'activation du dernier feed-back

Définition: message permettant de reproduire le dernier feed-back.
Opération: clôture du message.
Représentation: icône représentant une oreille (écran 1).
Action physique: Cliquet de la souris pour activer l'oreille.

-2- Production du dernier feed-back

Définition: Message d'aide qui consiste à reproduire le dernier feed-back produit.
Opération: Néant, clôture automatique.
Représentation: Message vocal digitalisé et mise en évidence de la pièce demandée dans la désignation par nom.
Action physique: Néant.

-3- Message "FINI"

Définition: message permettant de stipuler l'abandon de l'exercice.
Opération: clôture du message. Traitement associé: désactivation des messages courants excepté celui d'activation du dernier feed-back.
Représentation: icône représentant les caractères "FIN" (écran 1).
Action physique: cliquet de la souris pour activer l'icône.

-4- Confirmation de l'abandon de l'exercice

Définition: message permettant de confirmer l'abandon de l'exercice.
Opération: clôture du message en confirmant ou en infirmant l'abandon de l'exercice. Traitement associé: réactivation des messages courants si infirmation de l'abandon de l'exercice.
Représentation: le message textuel et vocal "Veux-tu finir?",un bouton contenant le texte "OUI" et un autre contenant "NON" (écran 8).
Action physique: cliquet de la souris pour activer un des deux boutons.

-5- Message "STOP"

Définition: message permettant de stipuler l'arrêt de l'exercice (arrêt de la session, qui peut être reprise plus tard).
Opération: clôture du message. Traitement associé: désactivation des messages courants excepté celui d'activation du dernier feed-back.
Représentation: icône représentant les caractères "STOP" (écran 1).
Action physique: cliquet de la souris pour activer l'icône.

-6- Confirmation du stop

Définition: message permettant de confirmer l'arrêt de l'exercice.
Opération: clôture du message en confirmant ou en infirmant l'arrêt de l'exercice. Traitement associé: réactivation des messages courants si infirmation de l'abandon de l'exercice.
Représentation: le message textuel et vocal "Veux-tu stopper?",un bouton contenant le texte "OUI" et un autre contenant "NON" (même genre que l'écran 8).
Action physique: cliquet de la souris pour activer un des deux boutons.

-7- Désignation du parcours au début

Définition: message visualisant sur le graphique le le trajet du parcours (écran5).
Opération: Néant clôture automatique.
Représentation: le parcours est désigné à l'écran.
Action physique: Néant.

-8- Désignation du début et de la fin du parcours

Définition: message visualisant sur le début et la fin du parcours.
Opération: Néant clôture automatique.
Représentation: Les points d'entrée et de sortie sont désignés.
Action physique: Néant.

-9- Message énoncé pour le parcours

Définition: production du feed-back du parcours qui doit être fait.
Opération: néant, clôture automatique.
Représentation: messages textuel et sonore:"Montre le parcours de *symbole"*
Action physique: néant

-10- Message de désignation d'une partie du parcours

Définition: message permettant de désigner une partie du parcours
Opération: clôture du message.
Représentation: le support est la silhouette servant de base à l'exercice de parcours (écrans 5,6 et 7).
Action physique: cliquet de la souris.

-11- Renforcement positif, parcours

Définition: message de renforcement signalant la désignation correcte d'une partie de parcours.
Opération: néant, clôture automatique.
Représentation: message sonore digitalisé (écran 2).
déplacement du symbole et/ou coloration.
Action physique: néant.

-12- Renforcement négatif, parcours

Définition: message de renforcement signalant la désignation incorrecte d'une partie de parcours.
Opération: néant, clôture automatique.
Représentation: message sonore digitalisé, message textuel et vocal:"Recommence".
Action physique: néant.

7.2.2.4. La session d'exercices

A. Scénario de base

Nous venons de présenter les exercices de type "désignation" et "parcours". Une session d'exercices est composée d'une succession d'exercices de ces types. Le logiciel de gestion des paramètres permet à l'éducateur de spécifier les différents exercices de la session qui seront présentés à l'utilisateur.

L'interaction globale entre le logiciel gestion de l'interaction et de l'utilisateur se présente comme suit : l'utilisateur doit tout d'abord donner son nom. Pour autoriser l'accès aux exercices, le nom introduit doit être celui spécifié par l'éducateur lors de la paramétrisation de l'interaction. L'objectif est de garantir qu'un utilisateur ne puisse réaliser des exercices qui n'ont pas été spécifiés à son intention. Cette nécessité provient de l'adaptation individuelle du niveau de difficulté de ces exercices. Si le nom est correct, le premier exercice est soumis à l'utilisateur. Lorsque cet exercice est terminé, il doit indiquer s'il désire le recommencer ou non. Dans la négative, il doit signaler s'il désire en réaliser un autre. Ces requêtes seront formulées entre chaque exercice de la session et jusqu'à ce que tous les exercices soient réalisés ou bien que l'utilisateur signifie qu'il ne désire plus en réaliser. Dans ce dernier cas, lorsque le programme sera réexécuté, l'exercice proposé sera le premier de la session.

Lors du déroulement d'un exercice l'utilisateur peut le stopper de deux manières. La première est de confirmer l'arrêt (message "STOP"). Dans ce cas le programme se termine. Lorsque le programme sera réexécuté, l'exercice proposé sera celui qui a été arrêté. Il se présentera dans l'état dans lequel il se trouvait juste avant la confirmation de l'arrêt. La deuxième manière de stopper un exercice lors de son déroulement est d'en confirmer la fin (message "FINI"). Cette confirmation a le même effet que lorsque l'exercice a été terminé à savoir la confirmation de l'utilisteur de recommencer l'exercice ou non.

Le programme "Corps" peut donc se terminer de trois manières différentes :

- l'utisateur confirme le message "STOP" pendant la réalisation de l'exercice;
- entre deux exercices l'utilisateur décide d'abandonner la session;
- lorsque le dernier exercice de la session est réalisé.

B. Le scénario détaillé

1. Représentation graphique

La représentation graphique de la session d'exercice est représenté par la figure 7.4.1.

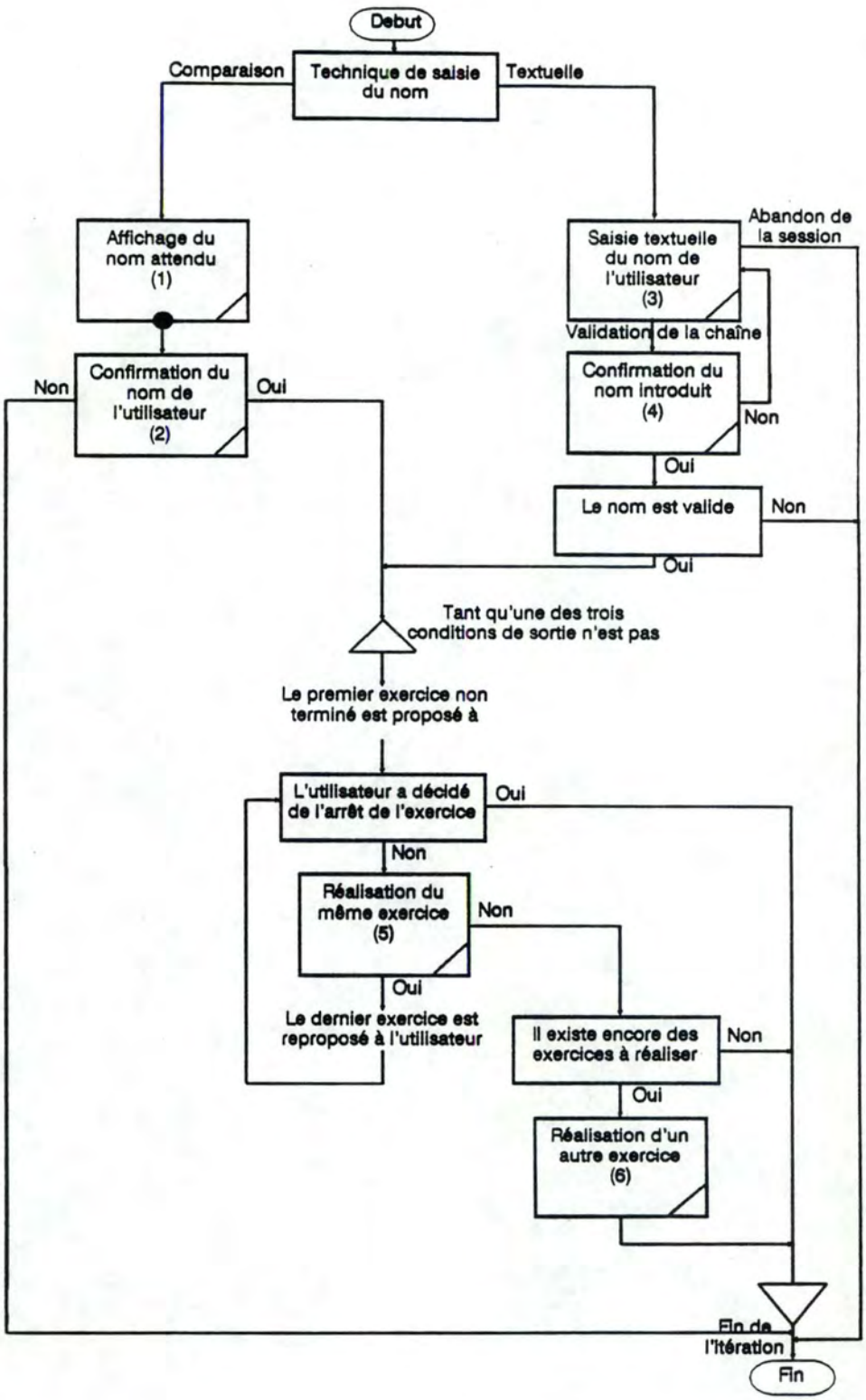


Figure 7.4 : Scénario d'une session d'exercices

2. Spécification des messages internes

-1- Affichage du nom de l'utilisateur

Définition: message indiquant le nom de l'utilisateur pour lequel la session d'exercices a été préparée
Opération: Néant, clôture automatique.
Représentation: le nom de la personne est affiché en haut de l'écran.
Action physique: Néant.

-2- Confirmation du nom par l'utilisateur

Définition: message permettant à l'utilisateur de spécifier si le nom qui lui est présenté est bien le sien.
Opération: clôture du message en confirmant ou en infirmant la conformité du nom affiché.
Représentation: le message textuel et vocal: "Est-ce bien ton nom?", un bouton contenant le texte "OUI" et un autre "NON".
Action physique: cliquet de la souris pour activer un des boutons.

-3- Saisie textuelle du nom de l'utilisateur

Définition: message permettant la saisie textuelle de ce nom ou l'abandon de la session .
Opération: 1 saisie d'une chaîne de caractères.
2 correction.
3 clôture du message.
Représentation: boîte constituant la zone de saisie du nom(max 17 caractères), le message textuel et vocal: "Quel est ton nom?", une icône "ABANDON" est affichée en dessous de la zone de saisie.

Action physique: 1 clavier et plan numérique pour la saisie de la chaîne de caractère.

2 touche <Backspace> pour corriger.

3 touche <Return> pour valider la saisie ou cliquet de la souris pour valider l'abandon.

-4- Confirmation du nom introduit

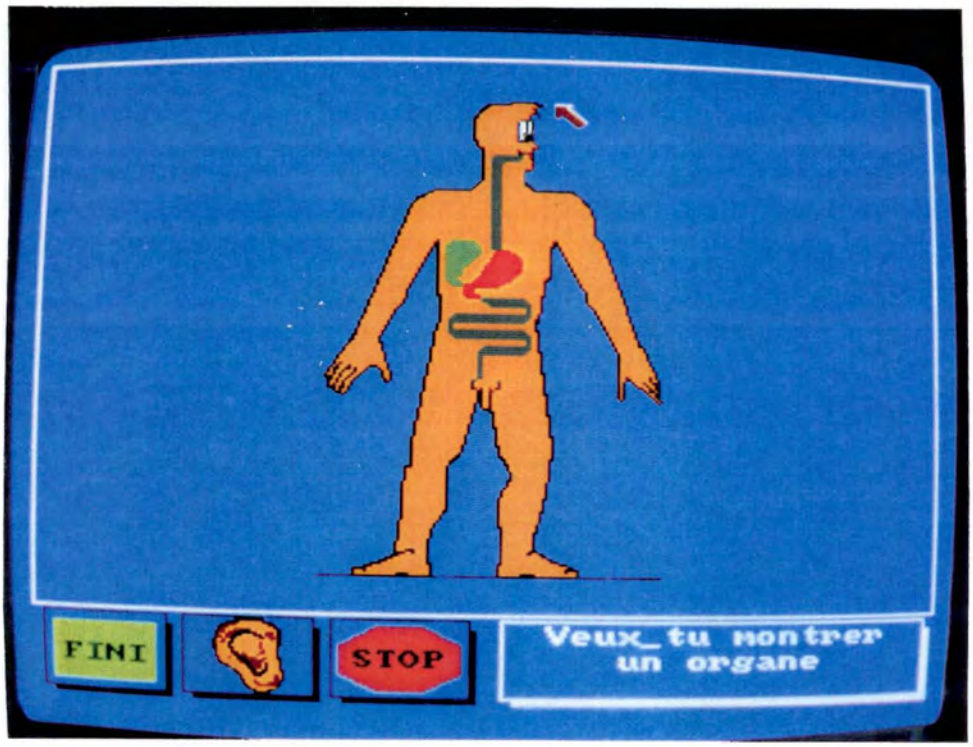
Définition: message permettant à l'utilisateur de confirmer la chaîne de caractères introduite.
Opération: clôture du message
Représentation: clôture du message en confirmant ou en infirmant la conformité du nom introduit.
Action physique: cliquet de la souris pour activer un des boutons.

-5- Réalisation du même exercice

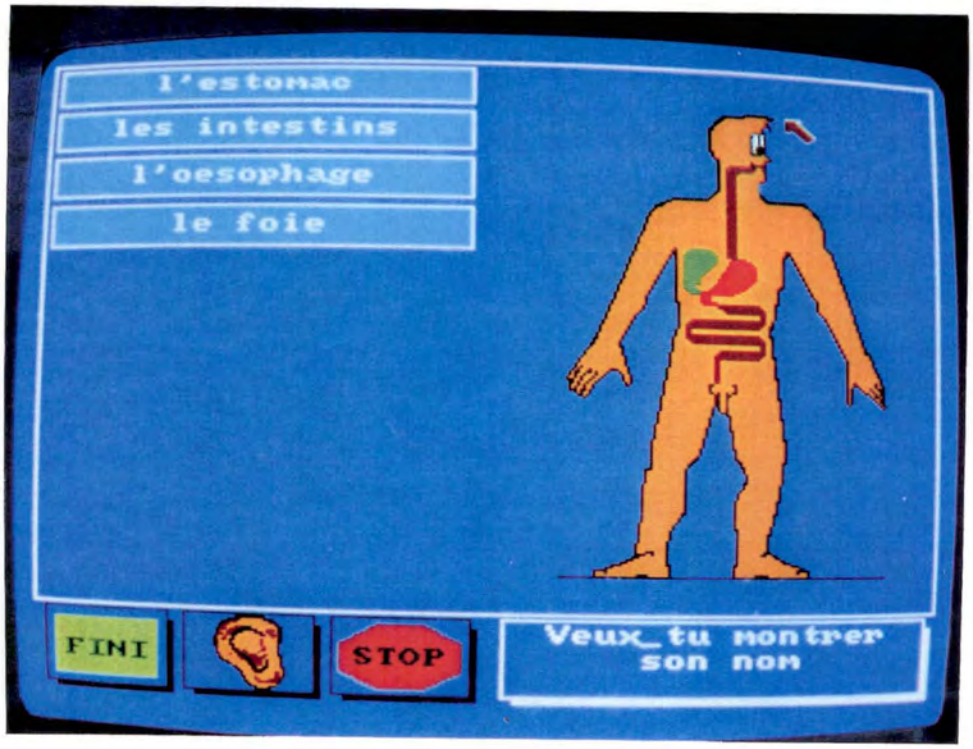
Définition: message permettant de stipuler le recommencement de l'exercice.
Opération: clôture du message en confirmant ou en infirmant le recommencement de l'exercice.
Représentation: le message textuel et vocal: "Est-ce bien ton nom?", un bouton contenant le texte "OUI" et un autre "NON".
Action physique: cliquet de la souris pour activer un des boutons.

-5- Réalisation d'un autre exercice

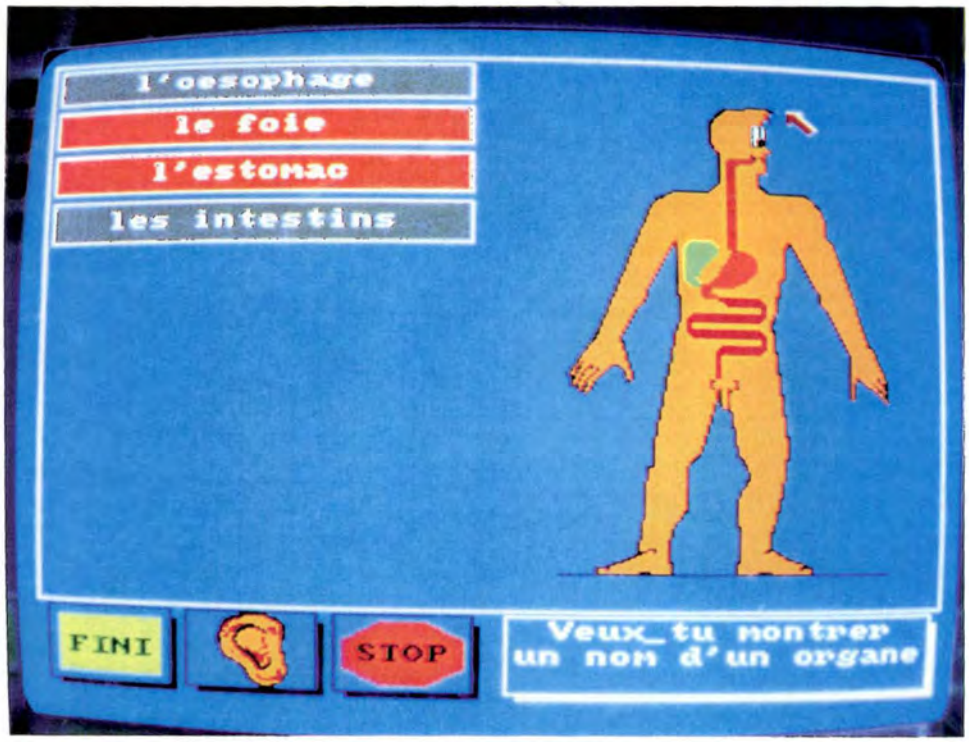
Définition: message permettant de stipuler la réalisation de l'exercice suivant de la session.
Opération: clôture du message en confirmant ou en infirmant



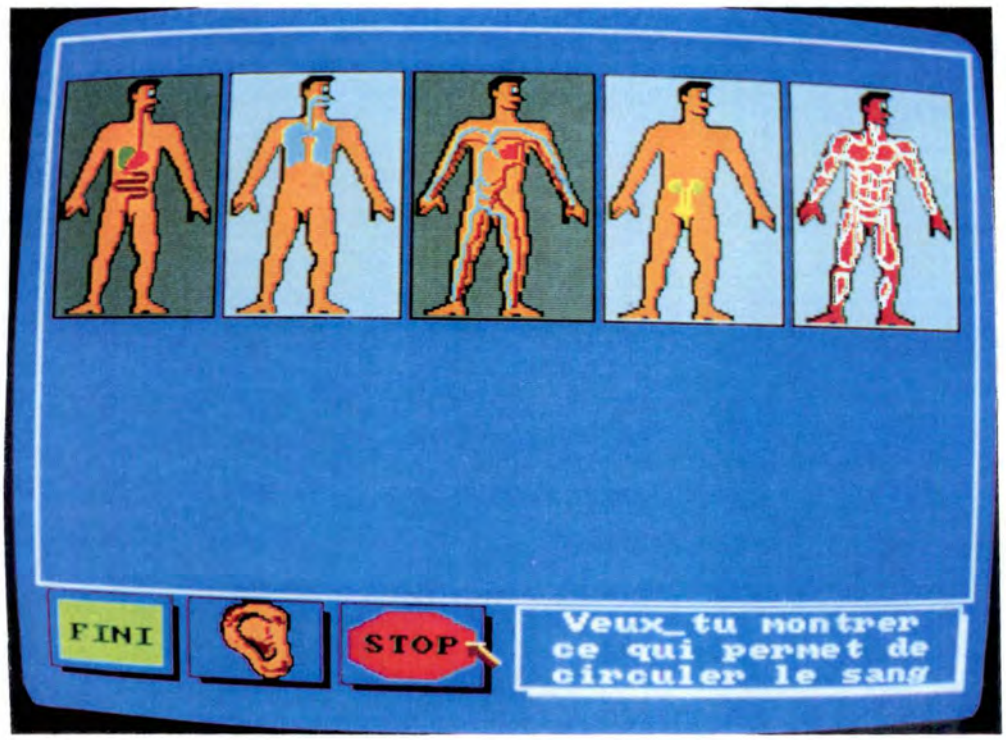
Ecran 1 : designation simple ou pseudo-passif



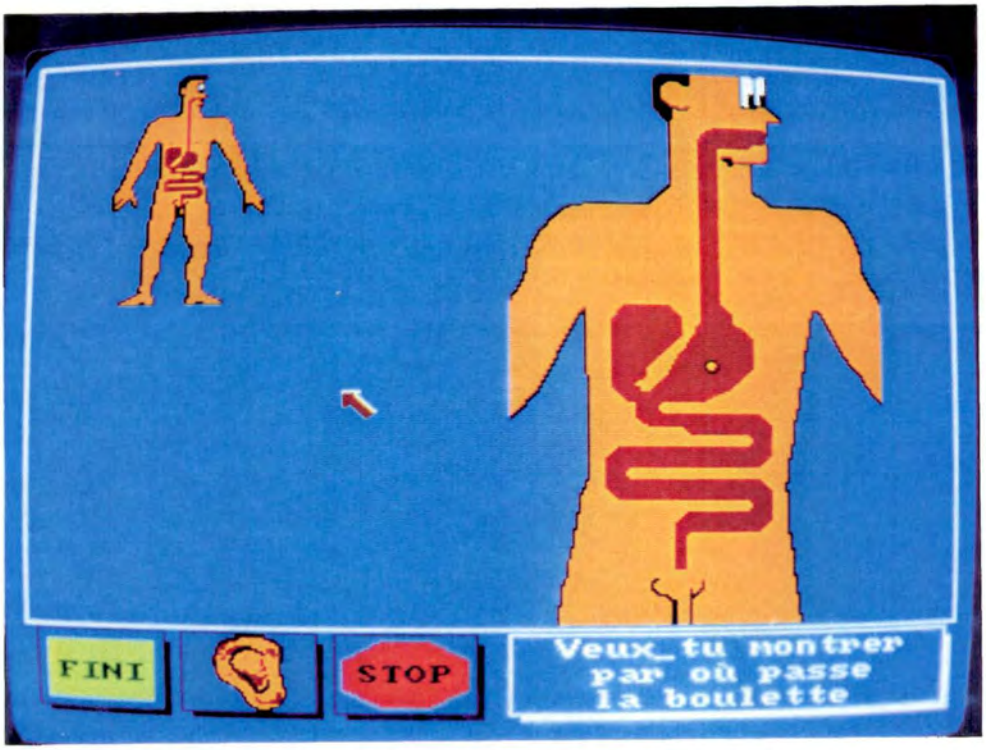
Ecran 2 : désignation par nom



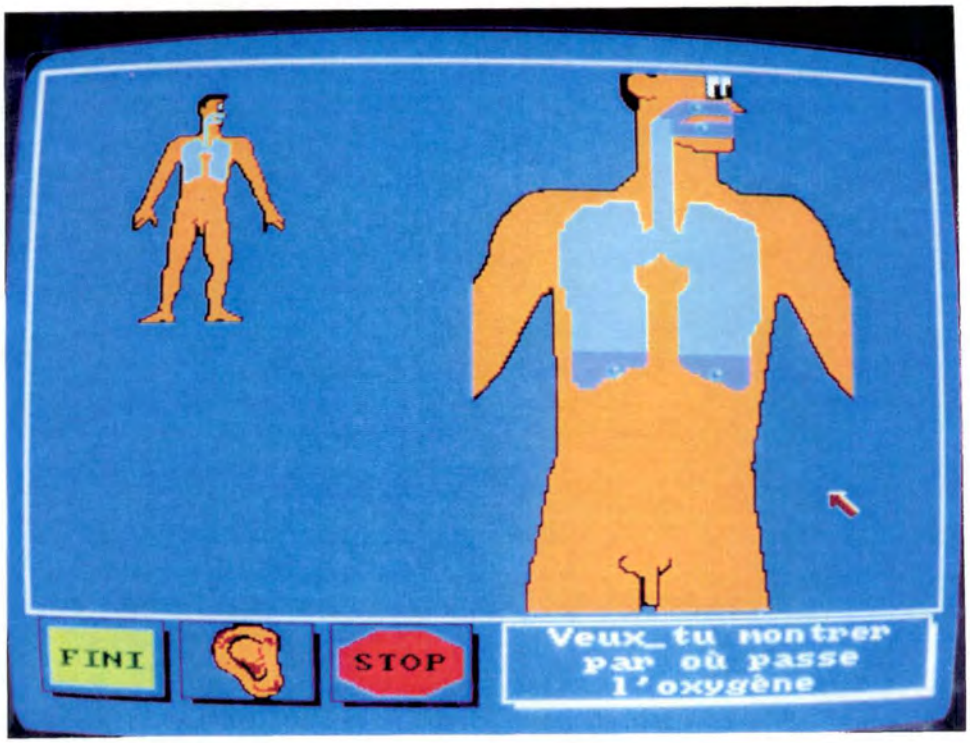
Ecran 3 : designation pseudo-passif par nom



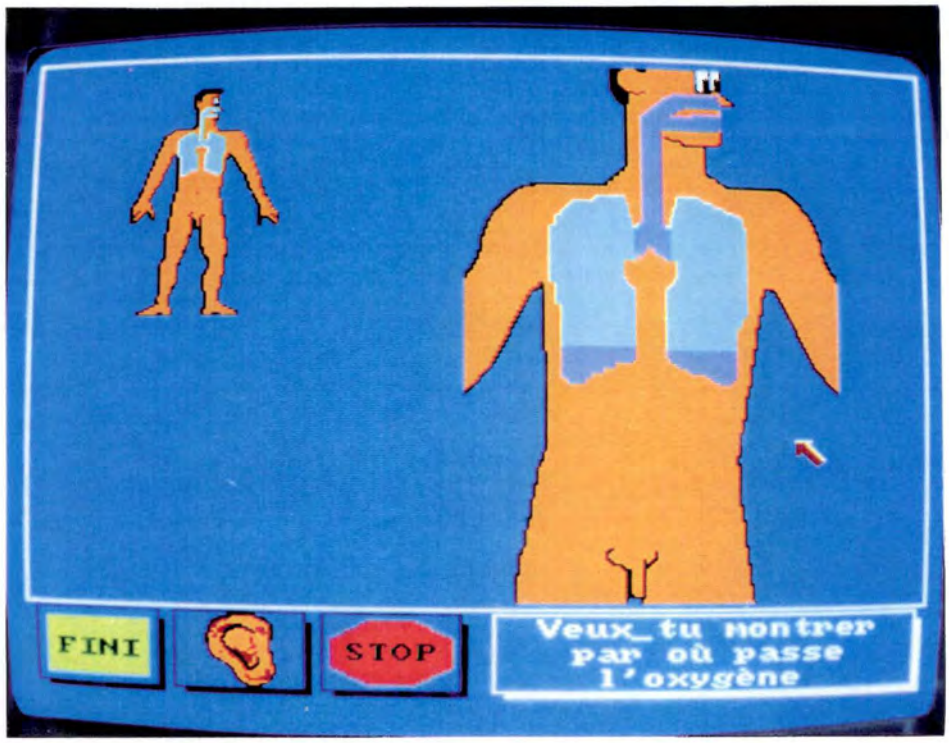
Ecran 4 : désignation des systèmes



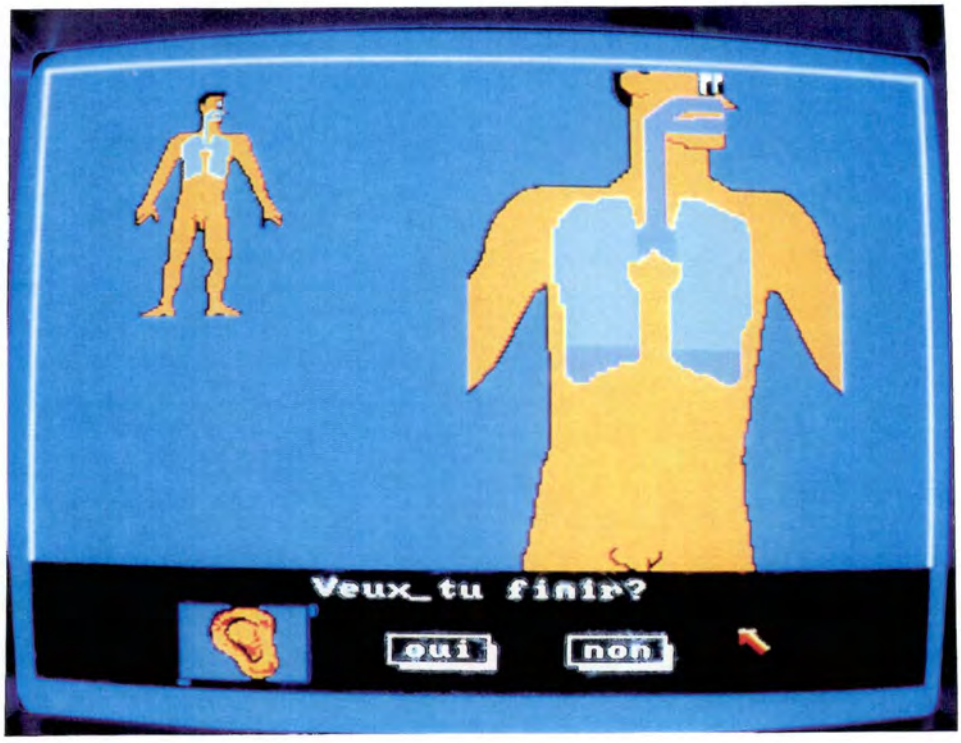
Ecran 5 : parcours avec symbole sans l'indication des extrémités



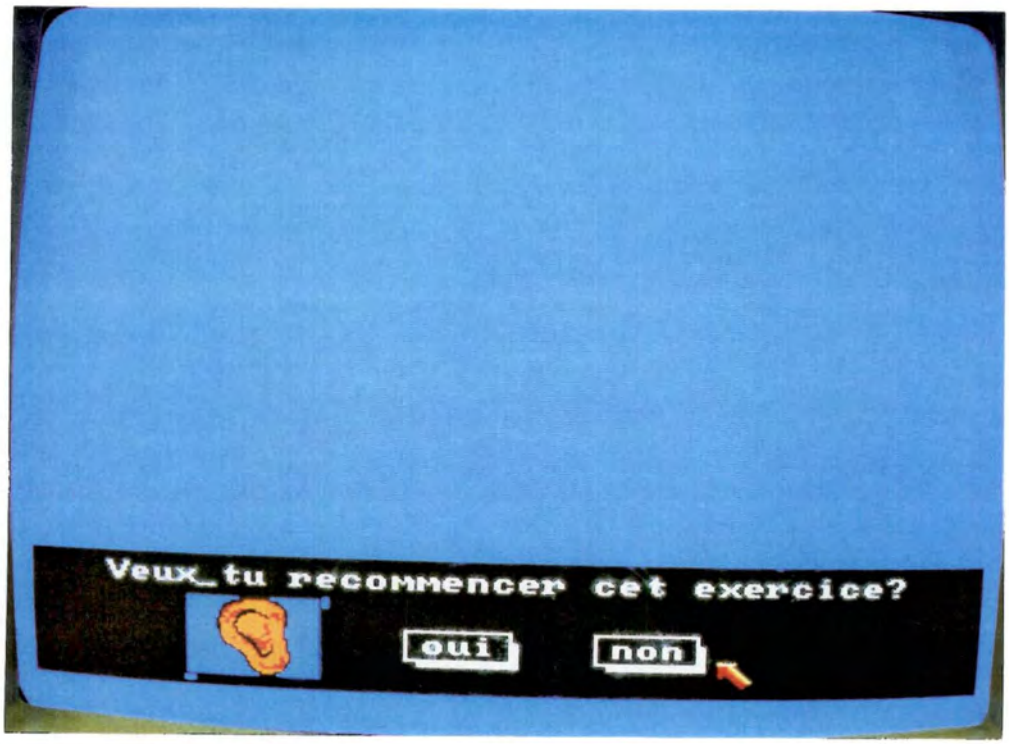
Ecran 6 : parcours avec symbole et coloration et indication des extrémités



Ecran 7 : parcours avec coloration et l'indication des extrémités



Ecran 8 : demande de confirmation de l'abandon de l'exercice parcours



Ecran 9: après chaque exercice, demande à l'utilisateur s'il désire recommencer l'exercice qu'il vient de terminer.

Ecran 10 : tant qu'il existe des exercices préparés pour l'utilisateur, confirmation de la poursuite de la session.

la réalisation de l'exercice suivant de la session.
Représentation: le message textuel et vocal: "Est-ce bien ton nom?", un bouton contenant le texte "OUI" et un autre "NON" (écran 10).
Action physique: cliquet de la souris pour activer un des boutons.

7.3 Spécification du logiciel "paramètrage"

7.3.1 Présentation du logiciel

Le but principal de ce logiciel est de permettre à l'éducateur d'initialiser des paramètres pour rendre le programme de gestion de l'interaction le plus adapté possible aux capacités de chaque utilisateur. Etant donné que le nombre de paramètres est relativement important, il est nécessaire d'aider l'éducateur dans sa tâche de préparation des exercices par un logiciel adéquat. Nous verrons que l'initilisation des paramètres destinés à configurer les exercices proposés par "Corps interne" n'est pas la seule fonctionnalité du logiciel. Il permet également à l'éducateur de visualiser les fichiers "trace" produit par le logiciel "Corps interne" contenant toutes les informations sur l'interaction entre l'utilisateur et "Corps interne".

7.3.2. Spécifications des fonctionnalités

Les deux fonctionnalités de base sont donc l'initialisation d'une disquette d'utilisateur et l'impression d'une trace textuelle.

7.3.2.1. L'initialisation d'une disquette utilisateur

Chaque utilisateur possède une disquette qui lui est personnelle. Ceci permet à chaque utilisateur de réaliser les exercices les plus adaptés à ses aptitudes comme nous l'avons déjà signalé. C'est la fonctionnalité clé du logiciel "Paramètrage" et la phase obligatoire de la future interaction entre le logiciel "Corps" et la personne handicapée. Elle permet à l'éducateur de spécifier différentes choses :

- tout d'abord le nombre et le type des exercices(désignation et/ou parcours) qui constitueront la session ainsi que l'ordre dans lequel ils seront

proposés à l'utilisateur. Le nombre d'exercice ne pourra dépasser le nombre de cinq;

- le nom de la personne à qui les exercices sont destinés. L'indiaction du nom est la manière d'individualiser les exercices. Comme nous l'avons déjà indiqué, ce nom permettra de s'assurer que la personne réalise les exercices que l'éducateur lui a préparés;

- pour chacun des exercices, les valeurs à attribuer aux différents paramètres. Ce processus ne constitue pas une lourde charge pour les éducateurs. Le nombre de paramètres auxquels ils sont amenés à donner une valeur ne dépasse pas huit pour l'exercice de type "désignation" et six pour l'exercice de type "parcours";

- pour l'ensemble de la session, le type de renforcements positif et négatif qui devra être utilisé. Ces renforcements peuvent être sonores (c'est-à-dire constitué de courtes mélodies) ou vocaux. Pour les renforcements sonores positif, l'éducateur peut choisir entre un renforcement unique ou aléatoire. Ce dernier est constitué de trois petites mélodies, et chaque fois que l'utilisateur, devra recevoir un feed-back à vocation de renforcement positif, le logiciel "Corps interne" en choisira une.

Lorsque l'interaction est préparée le logiciel copie sur la disquette de l'utilisateur, les graphiques et les paramètres retenus pour chacun des exercices spécifiés. Cette disquette fournira à "Corps interne" toutes les données dont il a besoin pour proposer à l'utilisateur de "Corps interne" les exercices spécifiés par l'éducateur.

7.3.2.2. L'impression d'une trace textuelle

Nous avons déjà à plusieurs reprises parlé de ces traces. Le logiciel "paramètrage" permet à l'éducateur de visualiser une trace textuelle à l'écran ou d'en demander l'impression sur papier. Il permet également de supprimer une trace textuelle. Cette suppression peut s'avérer intéressante dans certains cas. La disquette de l'utilisateur peut contenir un grand nombre de fichiers "trace", de sorte qu'il soit impossible d'y placer les données nécessaires à la session des exercices que l'utilisateur a spécifiés. Un fichier "trace" peut également n'être de plus d'aucune utilité.

Conclusion

Le développement d'un système d'apprentissage informatisé pour personnes ayant une déficience intellectuelle recouvre plusieurs disciplines. Une collaboration entre un ensemble d'acteurs spécialisés chacun dans leur domaine doit être la meilleure possible pour produire un outil utile et efficace. L'approche est donc pluridisciplinaire. Dans notre cas, le projet a été conduit par une équipe d'informaticiens et de spécialistes du handicap mental.

Ce logiciel répond à un besoin réel de la part de la population des personnes ayant un handicap mental. Il constitue un outil éducatif permettant de combler le manque de connaissances nécessaires pour faire l'apprentissage de comportements favorables à leur santé. La connaissance du fonctionnement de leur corps, car c'est de cela qu'il s'agit, correspond à un besoin primordial de l'être humain pour développer ses potentialités, et pour accéder à une autonomie et à un bien-être plus grand.

L'ordinateur semble présenter certaines aptitudes en tant que média de l'enseignement, notamment concernant ce domaine d'application. Il apparaît comme un moyen attractif par rapport aux techniques conventionnelles utilisées avec les personnes présentant des déficiences mentales, par ses actions de motivation et d'individualisation de l'apprentissage. Il contribue, pour le moins, à une diversification des moyens utilisés et constitue une aide efficace à la connaissance du corps.

Nous pouvons insister sur le rôle des représentations graphiques dans l'apprentissage se situant dans le domaine de la connaissance du corps. La connaissance des différents systèmes anatomiques et des mécanismes comme la digestion, la respiration requièrent une bonne représentation car ces connaissances présentent la caractéristique d'être abstraites. La seule manière pour les personnes ayant un handicap mental, et les autres, de

percevoir leur corps interne se manifeste à travers la douleur.

L'accent a été mis, également sur divers formes de spécification d'un logiciel qui se veut interactif et ouvert. Toutes ces formes sont à considérer et mettre en oeuvre d'après les objectifs que l'on se fixe.

Le projet que nous avons développé fut pour nous une expérience enrichissante tant au niveau intellectuel qu'humain. Il nous a fait entrer dans un monde chaleureux qui nous était jusque là inconnu. Nous espérons que ce projet contribuera à mettre en oeuvre encore beaucoup d'autres. L'usage de l'informatique à des fins didactiques pour personnes ayant une déficience mentale est une voie à suivre.

BIBLIOGRAPHIE

BARTHET M.F., "**Logiciel Ineractif et Ergonomie**", Dunod 1988.

BARTHE C., BRIERE S., LEGENDRE V., PECHOIN M.H., "**Comprendre son corps, guide de l'animateur**", dossier "**Migration Santé**" , Comité Médico-Social pour la santé des migrants, 1984.

BURKART, J.E., FOX, R.A. ROTATORI, A.F., Obesity of Mentally Retarded Individuals : Prevence characteristics and Intervention, American Journal of Mental Deficiency, 1985, 90, 3, pp 303 - 312.

COLLIGNON J.L., "**Programme de remédiation à l'obésité destiné aux personnes handicapées mentales de type léger et modéré en institution**", unpublished.

COLLIGNON J.L., GALAND M., "**Approche de deux problématiques chez la personne handicapée mentale: la santé et le vieillissement**", in DUTRAUX J.J., MRECIER M., "**Recherche relative aux critères devant présider à la programmation de la capacité d'accueil des établissements et services spécialisés dans le traîtement des personnes handicapées de la Communauté Française**".fascicule n° 6,1990.

COUTAZ J., "**Interface Homme-Ordinateur: Conception et Réalisation**", thèse, 1989.

DELDIME R., VERMEULEN S., "**Le développement psychologique de l'enfant**", Collection Univers des sciences humaines, De Boeck, 1981.

DELVILLE J., COLLIGNON J.L., "**Education pour la santé et handicap mental: analyse de l'image du corps et de son fonctionnement**", 2° congrès de l'Association Internationale de Recherche Scientifique en Forum des Personnes handicapées mentales, 4-5-6 Avril 1991, L'intervention en Déficience Mentale, Théories et Pratiques,Lille.

DEPOVER C., "**L'ordinateur média d'enseignement, un cadre conceptuel**", in Pédagogie en développement, Problèmatiques et Recherches, De Boeck Université, 1987.

DITTRICH, GELFAND, SCHEMMEL, "**La Bible de l'Amiga**", Edition Micro Application, 1988.

DUFOYER J.P., "**Informatique, éducation et psychologie de l'enfant**", 1988.

DONNAYJ., ROMAINVILLE M., "**Grille d'évaluation de didactitiels**", in Formation Recherche en éducation n° 11, centre O.S.E., département Education et Technologie, FUNDP Namur.

FRAITURE M., MACHGEELS C., "**Le cahier des charges d'un logiciel pour personnes handicapées mentales**", Hantepsycom Kerpape 5-7 juillet 1990.

FOX, R.A, ROTATORI, A.F., "**Prevalence of Obesity among Mentally Retarded Adults**", Americaa Journal of Mental Deficiency, 1982, 87, 2, pp 228-230.

HUBENS V., LECRON V., "**Connaissance et vécu du corps chez l'adulte handicapé mental**", Mémoire présenté en vu de l'obtention du diplôme de gradué en kinésithérapie, Ecole Provinciale Supérieure de Kinésithérapie et d'Ergothérapie, 1990.

:

HURTIG M., "**Introduction à la psychologie de l'enfant**", Pierre Mardaga Editeur, 3 volumes, 1981.

IONESCU S., "**L'intervention en déficience mentale**", in Problèmes généraux Méthodes médicales et psychpologiques", Pierre Mardaga Editeur, 1987.

LAMBERT J.L., "**Introduction à l'arriération mentale**", in Psychologie et Sciences humaines 2ème édition, Pierre Mardaga Editeur, ***.

LEPOUTRE T., ROQUET J.M., "**La personne handicapée mentale et la connaissance du corps humain: un logiciel d'apprentissage**", Mémoire présenté en vue de l'obtention du diplôme de Licencié et Maître en Informatique, FUNDP Namur, 1990.

LESUISSE R. "**La participation des utilisateurs : pourquoi, comment?**" in Journal de Réflexion sur l'Informatique, n°4, pp 19-22,1987.

PIAGET J., "**La psychologie de l'enfant**", Edition Que sais-je?, Presses Universitaires de France, 1966.

MARTIMORE E.P., "**Amiga Programmer's Handbook**", volumes I et II, seconde édition, Sybex, 1987.

MERCIER M., DELVILLE J., "**Aspects psychosociaux en éducation pour la santé**", Savoir et Santé, De Boeck Université, 1988.

NOT L., "**Perspectives nouvelles pour l'Education des Débiles Mentaux**", Science de l'homme, Privat, 1986.

ROOSEN M., "**La psychologie de l'enfant et de l'adolescent**", Sciences Humaines, Syllabus de cours, unpublished.

SCAPIN D.L., "**guide ergonomique de construction des interfaces Homme-Ordinateur,** Institut National de Recherche en Informatique et en Automatique.

SHNEIDERMAN, "**Designing the user interface, strategies for effective human computer interaction**", Edition Addison, Wusley,1987.

WALLON H., LURCAT L., "**Dessin, espace et schéma corporel chez l'enfant**, Les éditions ESF, 1987.

WARNANT G., "**Elements de modélisation du dialogue d'une application interactive**", 1988.

ZAZZO R., "**Les débilités mentales**", Armand Colin, Collection u, 1979.

Année académique 1990-1991

Connaissance du corps interne: Réalisation d'un logiciel d'apprentissage pour personnes ayant une déficience mentale

ANNEXES

Alain Fontesse

Bernard L'homme

Promoteur: **Madame Monique Noirhomme-Fraiture**

Co-promoteur: **Monsieur Michel Mercier**, professeur au Département de Psychologie de la faculté de Médecine, F.U.N.D.P.

Mémoire présenté en vue de
l'obtention du grade de Licencié
et Maître en informatique

ANNEXES

Annexe 1 : Manuel d'utilisation de l'outil

Annexe 2 : Quelques exemples de traces textuelles

Annexe 3 : Contenu des fichiers

Annexes 4 : Les "definition modules" et les "implementation modules" du logiciel "Corps interne"

Annexe 5 : Les "definition modules" et les "implementation modules" du logiciel "Parametrage"

ANNEXE 1: Manuel d'utilisation de l'outil

L'outil "corps interne":manuel d'utilisation

Introduction

L'objectif que vise cet outil didactique est la connaissance des systèmes, des organes et du fonctionnement du corps humain. Il est destiné aux personnes ayant une déficience mentale et est composé de deux logiciels distincts. Il est enfin caractérisé par un haut degré de paramétrisation.

L'environnement matériel minimal nécessaire à l'exploitation de l'outil est un AMIGA 500 possédant deux lecteurs de disquettes 3 1/2 pouces ainsi qu'une capacité mémoire de 1 MB. Si vous possédez un disque dur, un seul lecteur suffit.

1 L'outil général

Dans le soucis de réaliser un outil "ouvert", flexible, l'outil général se composera de deux éléments, deux logiciels.

Le premier est le logiciel "Corps interne". Ce dernier propose et gère les exercices qui sont exécutés par les personnes handicapées mentales.

Le deuxième logiciel est le logiciel "Parametrage". Ce dernier contrairement à Corps interne est destiné aux éducateurs et utilisé uniquement par ces derniers. Il leur permet de donner des valeurs à une série de paramètres afin de configurer les exercices que l'apprenant devra réaliser.

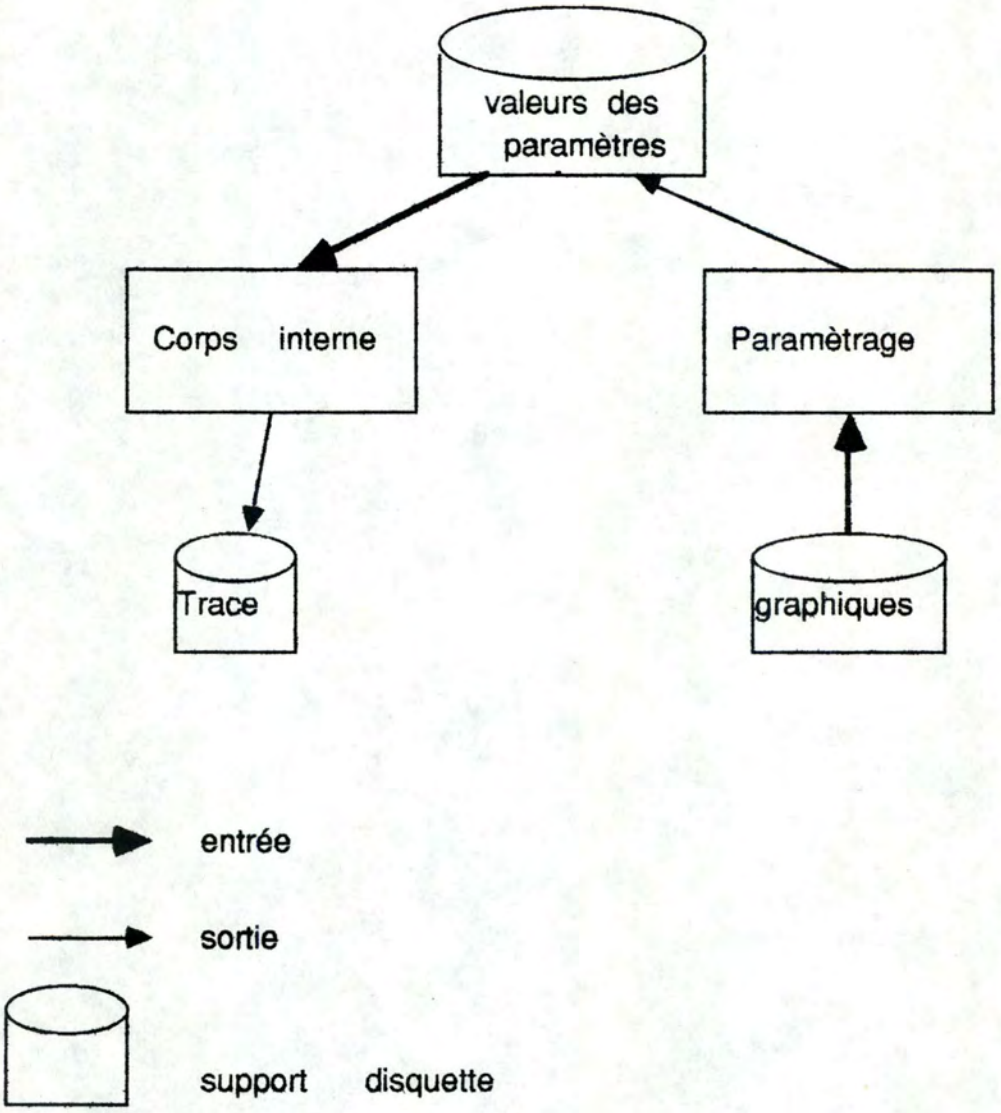


figure 1 : l'interaction des logiciels

Ces deux logiciels vont s'échanger des informations. La figure 1 montre quelles sont ces informations en entrée et en sortie qu'ils produisent.

Le logiciel "Parametrage" reçoit en entrée les graphiques disponibles supportant les exercices. L'éducateur pourra opérer un choix parmi tous les graphiques proposés et donner une valeur à un ensemble de paramètres. "Parametrage" recopie sur une disquette appelée "USER" les graphiques retenus ainsi que les valeurs des paramètres.

Le logiciel Corps interne reçoit en entrée les données et les graphiques se rapportant aux exercices à proposer à l'utilisateur ainsi que les paramètres se trouvant sur la disquette "USER". Il produit en sortie la "trace" de l'interaction entre le logiciel et l'apprenant. Cette "trace" est recopiée sur la disquette "USER".

Le processus associé à "Parametrage" correspond à la phase d'initialisation et le processus associé à Corps interne est l'interaction proprement dite avec la personne ayant une déficience mentale.

2 Le logiciel "Corps interne"

2.1. Présentation du logiciel

Le logiciel "Corps interne" propose deux types d'exercices à l'utilisateur. Un exercice de type "désignation" et un exercice de type "parcours".

Dans l'exercice de type "désignation", l'ordinateur énonce successivement les différentes parties d'un graphique à l'utilisateur et celui-ci doit situer correctement les parties énoncées, qu'il s'agisse d'organes dans un système, de noms d'organes, ou de systèmes parmi d'autres . Il est aussi possible de connaître le nom d'organe désigné par l'utilisateur dans un exercice de type pseudo-passif.

Dans l'exercice de type "parcours", il est demandé à l'utilisateur de montrer le parcours d'un symbole représentant une substance (oxygène, boulette de viande, eau...) dans une silhouette représentant un système anatomique du corps humain (ou deux systèmes). L'utilisateur montre le parcours en désignant des points de passage de la substance au moyen de la souris. Une désignation de parcours correspond à une suite de désignations de parties (les organes traversés), après avoir montré le point de départ. Le renforcement correspond à des feed-backs sonores et textuels qui sont produits à chaques actions de l'utilisateur.

Le logiciel propose ces exercices dans une session. Une session de trois exercices comprend, par exemple, un exercice de désignation , suivi d'une autre désignation, et finalement d'un exercice de type parcours.

2.1.1 La désignation

A. Scénario de base

Dans l'exercice de type "désignation", la machine énonce successivement les noms des différentes parties d'un graphique présenté à l'utilisateur. Ce dernier doit, à chaque énoncé, situer correctement la partie. Le graphique peut représenter une silhouette humaine contenant des organes regroupés en un ou deux systèmes anatomiques. Le graphique peut également montrer les représentations de systèmes anatomiques. La présentation de graphiques avec des organes ou avec les dessins des systèmes dépend en fait de la liste des énoncés choisie par l'éducateur dans l'ensemble de graphiques disponibles. Le graphique de la silhouette présenté à l'utilisateur peut être dépourvu de ses organes. Dans ce cas l'utilisateur doit indiquer l'emplacement de l'organe demandé. Le graphique présenté peut également être accompagné des noms des organes présents sur le graphique de la silhouette. Dans ce cas ce sont les noms des organes que l'utilisateur doit indiquer. Lorsque le graphique de la silhouette est représenté à l'utilisateur sur l'écran, quatre types d'interaction sont possibles : l'apprentissage, l'évaluation, l'interaction nommée pseudo-passive et celle appelée passive. Un paramètre permet à l'éducateur de définir le type choisi.

Dans le mode passif, comme son nom l'indique, l'utilisateur est passif; la seule chose qu'il doit faire est de regarder l'écran. Le logiciel énonce chacun des noms d' organe ou des systèmes anatomiques du graphique contenus dans la liste. Pendant cet énoncé, le dessin de l'organe, du système ou le nom de l'organe est mis en évidence.

Dans le mode pseudo-passif, l'utilisateur est plus impliqué que dans le mode passif mais moins que dans les deux autres modes : apprentissage et évaluation. La machine demande à l'utilisateur d'indiquer le dessin d'un organe, d'un système, d'un nom d'organe. Lorsque l'utilisateur a montré une de ces choses, l'ordinateur lui indique ce que cela représente.

Le mode apprentissage est le mode où l'utilisateur doit répondre à une question posée par la machine à savoir de montrer un organe, un nom d'organe, un système anatomique, la place d'un organe. En cas de réponse incorrecte (la réponse se matérialise par un clic de la souris), l'ordinateur met en évidence la réponse avant de faire de même pour la réponse correcte (l'ordinateur corrige). La question qui a entraîné une réponse incorrecte est reposée immédiatement avant d'être reposée une nouvelle fois plus tard.

Le mode évaluation est semblable au précédent si ce n'est que la machine ne corrige plus. L'utilisateur est libre de répondre ce qu'il entend. Ce mode correspond à des exercices dont le but est de tester les connaissances comme cela se fait lors d'examens.

B. Paramètres d'une désignation

Paramètre 1: le type d'interaction qui peut avoir trois valeurs:

-*interaction de type apprentissage*. Les réponses fournies par l'élève aux questions posées sont corrigées automatiquement par le processus des feed-backs.

-*interaction de type évaluation*: Dans ce cas, la machine ne corrige plus les réponses de l'élève. Il peut faire ce qu'il veut. On ne fait plus appel aux feed-backs.

-*interaction de type pseudo-passif*: L'accent est ici mis sur la possibilité donnée à l'utilisateur de découvrir par exploration la matière qu'on désire qu'il maîtrise. L'association est faite entre un aspect particulier de cette matière

choisi par l'utilisateur, et la question qui peut lui être associée ultérieurement, dans un autre type d'interaction. Cette association constitue le feed-back.

-*interaction de type passif*: L'utilisateur n'apparaît ici, non plus comme un acteur, mais comme un observateur. C'est la machine qui effectue elle-même les exercices.

Paramètre 2: *l'ordre d'énumération* détermine l'ordre dans lequel les organes sont demandés à l'utilisateur. Il peut correspondre à un ordre logique lié à la position de l'organe dans le système, ou à un ordre aléatoire.

Paramètre 3: *énoncé vocal* des parties du graphique (oui/non).

Paramètre 4: *nombre d'itérations* sur la technique de désignation avant de reposer une partie mal désignée (1/2/3/4/5). Lors d'une erreur de désignation, la question est reposée deux fois: la première fois immédiatement. La seconde fois dépendra du nombre d'itération choisi. Si le choix s'est porté sur 2, la machine posera 2 autres questions avant de reposer la question ayant reçu une réponse éronnée.

Paramètre 5: *silhouette vide ou non.* Il s'agira de désigner un organe visible dans un système (silhouette non vide) ou de désigner l'endroit où il se situe dans la silhouette vide.

Paramètre 6: une *animation* peut être proposée en plus des feed-backs usuels. C'est le cas rencontré lors de la désignation par système.

Paramètre 7: *type de désignation.* La détection de la pièce que l'utilisateur a voulu désigner à partir de l'endroit où il a "cliqué" peut se faire par gestion des *cadres* de désignation des organes ou par gestion des *couleurs* des organes. Ce paramètre peut être dépendant de la représentation graphique que l'on veut utiliser.

Paramètre 8: *objet de désignation.* Les désignations par image ou par nom peuvent être choisies. Soit on demande à l'utilisateur de montrer un organe

dont le nom est énoncé, soit, c'est le nom d'un organe mis en évidence dans la silhouette qui doit être trouvé.

2.1.2 Le parcours

A. Scénario de base

Dans l'exercice de type "parcours", la machine demande à l'utilisateur de désigner sur un graphique représentant une silhouette humaine le parcours d'une substance : parcours d'une boulette à travers le système digestif, parcours de l'oxygène à travers le système respiratoire, parcours du sang à travers le système circulatoire ainsi que le parcours d'autres substances, parcours de l'oxygène à travers le système respiratoire et le système circulatoire, ...etc. La machine peut (suivant le souhait de la personne qui spécifie les exercices) indiquer le parcours en début et/ou à la terminaison de l'exercice. Elle peut également indiquer le point de début et le point de sortie du parcours. Deux modes d'interaction peuvent être (via la paramétrisation des exercices) choisis : apprentissage et évaluation.

Le mode apprentissage, comme sa dénomination l'indique, a pour but d'apprendre à l'utilisateur les parcours des différentes substances dans l'organisme. L'utilisateur indique le parcours au moyen de la souris. L'ordinateur contrôle la véracité de cette indication. En cas d'erreur l'ordinateur, après avoir indiqué le type d'erreur commise, propose à l'utilisateur de recommencer l'exercice. Les erreurs qui peuvent être commises sont de trois types. Nous avons déjà expliqué la manière dont l'utilisateur devait s'y prendre pour indiquer le parcours. Le premier type d'erreur concerne un retour en arrière dans le parcours. Le deuxième correspond à s'écarter du parcours et enfin le dernier à omettre une partie du parcours (omettre de désigner un organe). A chaque désignation correcte d'une partie du parcours, l'ordinateur donne un feed-back pouvant être de trois natures. Le premier type consiste en le déplacement du symbole, représentant la substance qui parcourt le trajet demandé, de l'endroit du parcours où l'utilisateur était arrivé précédemment jusqu'à l'endroit qu'il vient d'indiquer. Le deuxième est identique au premier si ce n'est le remplacement du symbole par une

coloration qui dépend du graphique de base de l'exercice. Le dernier est l'union des deux précédents.

Le mode évaluation laisse l'utilisateur indiquer ce qu'il désire sans, comme dans le mode apprentissage, le contraindre à recommencer en cas d'erreur. Les différents feed-backs du mode apprentissage se retrouvent dans le mode évaluation.

B. Paramètres du parcours

Paramètre 1: le *type d'interaction* pour cet exercice n' a que deux valeurs possibles: évaluation et apprentissage.

Paramètre 2: Le *parcours au début* de l'exercice est effectué par la machine ou non, comme aide à l'utilisateur.

Paramètre 3: les *points de départ et d'arrivée* sont désignés ou non comme aide à l'utilisateur. Il ne doit plus désigner le point de départ.

Paramètre 4: le *type de feed-back visuel* est ici défini. Il s'agit soit du déplacement du symbole, soit de la coloration, soit des deux en même temps (mixte).

Paramètre 5: *le parcours en fin* est présenté comme renforcement à un parcours complètement désigné ou non.

Paramètre 6: *l'énoncé vocal* est activé ou non, comme dans l'exercice de désignation.

2.1.3. La session d'exercices

A. Scénario de base

Nous venons de présenter les exercices de type "désignation" et "parcours". Une session d'exercices est composée d'une succession d'exercices de ces types. Le logiciel de gestion des paramètres permet à l'éducateur de spécifier les différents exercices de la session qui seront présentés à l'utilisateur.

L'interaction globale entre le logiciel gestion de l'interaction et de l'utilisateur se présente comme suit : l'utilisateur doit tout d'abord donner son nom. Pour autoriser l'accès aux exercices, le nom introduit doit être celui spécifié par l'éducateur lors de la paramétrisation de l'interaction. L'objectif est de garantir qu'un utilisateur ne puisse réaliser des exercices qui n'ont pas été spécifiés à son intention. Cette nécessité provient de l'adaptation individuelle du niveau de difficulté de ces exercices. Si le nom est correct, le premier exercice est soumis à l'utilisateur. Lorsque cet exercice est terminé, il doit indiquer s'il désire le recommencer ou non. Dans la négative, il doit signaler s'il désire en réaliser un autre. Ces requêtes seront formulées entre chaque exercice de la session et jusqu'à ce que tous les exercices soient réalisés ou bien que l'utilisateur signifie qu'il ne désire plus en réaliser. Dans ce dernier cas, lorsque le programme sera réexécuté, l'exercice proposé sera celui qui suit l'exercice terminé dans la session.

Lors du déroulement d'un exercice l'utilisateur peut le stopper de deux manières. La première est de confirmer l'arrêt (clic sur "STOP"). Dans ce cas le programme se termine. Lorsque le programme sera réexécuté, l'exercice proposé sera celui qui a été arrêté. Il se présentera dans l'état dans lequel il se trouvait juste avant la confirmation de l'arrêt. La deuxième manière de stopper un exercice lors de son déroulement est d'en confirmer la fin (clic sur "FINI"). Cette confirmation a le même effet que lorsque l'exercice a été terminé à savoir la confirmation de l'utilisateur de recommencer l'exercice ou non.

Le programme "Corps" peut donc se terminer de trois manières différentes :
- l'utilisateur confirme le "STOP" pendant la réalisation de l'exercice;
- entre deux exercices l'utilisateur décide d'abandonner la session;
- lorsque le dernier exercice de la session est réalisé.

Comment exécuter le logiciel?

Pour lancer le programme "Corps interne", mettez l'ordinateur sous tension, introduisez la disquette "corps interne" dans le lecteur de disquette DF0: et la disquette "USER" dans l'autre (pour produire cette disquette "USER", référez-vous à l'utilisation du programme "Parametrage"). Cliquez deux fois sur l'icône "corps interne". Il apparaît alors une fenêtre contenant les icônes "Corps" et "Reinitialisation". Pour lancer le programme "Corps", cliquez deux fois rapidement sur l'icône "Corps".

Si vous lancez le programme "Corps" et que celui-ci se termine immédiatement, cela signifie que tous les exercices préparés sur la disquette "USER" sont terminés. Dans ce cas, deux possibilité vous sont offertes. Vous pouvez préparer une nouvelle session d'exercices en utilisant le programme "Parametrage". Par contre, si vous désirez que l'utilisateur recommence les exercices préparés sur la disquette "USER", il vous suffira de cliquer deux fois rapidement sur l'icône "Reinitialisation" (le dessin d'une disquette apparaît alors à l'écran). Lorsque la disquette "user" est réinitialisée, cliquez deux fois sur l'icône "Corps interne" pour lancer le programme.

3 Le logiciel "Parametrage"

3.1 Présentation du logiciel

Le but principal de ce logiciel est de permettre à l'éducateur d'initialiser des paramètres pour rendre le programme de gestion de l'interaction le plus adapté possible aux capacités de chaque utilisateur. Etant donné que le nombre de paramètres est relativement important, il est nécessaire d'aider l'éducateur

dans sa tâche de préparation des exercices par un logiciel adéquat. Nous verrons que l'initialisation des paramètres destinés à configurer les exercices proposés par "Corps interne" n'est pas la seule fonctionnalité du logiciel. Il permet également à l'éducateur de visualiser les fichiers "trace" produit par le logiciel "Corps interne" contenant toutes les informations sur l'interaction entre l'utilisateur et "Corps interne".

3.2. Spécifications des fonctionnalités

Les deux fonctionnalités de base sont donc l'initialisation d'une disquette d'utilisateur et l'impression d'une trace textuelle.

3.2.1. L'initialisation d'une disquette utilisateur

Chaque utilisateur possède une disquette qui lui est personnelle. Ceci permet à chaque utilisateur de réaliser les exercices les plus adaptés à ses aptitudes comme nous l'avons déjà signalé. C'est la fonctionnalité clé du logiciel "Parametrage" et la phase obligatoire de la future interaction entre le logiciel "Corps interne" et la personne handicapée. Elle permet à l'éducateur de spécifier différentes choses :

- tout d'abord le nombre et le type des exercices(désignation et/ou parcours) qui constitueront la session ainsi que l'ordre dans lequel ils seront proposés à l'utilisateur. Le nombre d'exercices ne pourra dépasser le nombre cinq;

- le nom de la personne à qui les exercices sont destinés. L'indication du nom est la manière d'individualiser les exercices. Comme nous l'avons déjà indiqué, ce nom permettra de s'assurer que la personne réalise les exercices que l'éducateur lui a préparés;

- pour chacun des exercices, les valeurs à attribuer aux différents paramètres. Ce processus ne constitue pas une lourde charge pour les éducateurs. Le nombre de paramètres auxquels ils sont amenés à donner une

valeur ne dépasse pas huit pour l'exercice de type "désignation" et six pour l'exercice de type "parcours";

- pour l'ensemble de la session, le type de renforcements positif et négatif qui devra être utilisé. Ces renforcements peuvent être sonores (c'est-à-dire constitués de courtes mélodies) ou vocaux. Pour le renforcement sonore positif, l'éducateur peut choisir entre un renforcement unique ou aléatoire. Ce dernier est constitué de trois petites mélodies, et chaque fois que l'utilisateur, devra recevoir un feed-back à vocation de renforcement positif, le logiciel "Corps interne" en choisira une.

Lorsque l'interaction est préparée, le logiciel copie sur la disquette de l'utilisateur, les graphiques et les paramètres retenus pour chacun des exercices spécifiés. Cette disquette fournira à "Corps interne" toutes les données dont il a besoin pour proposer à l'utilisateur de "Corps interne" les exercices spécifiés par l'éducateur.

3.2.2. L'impression d'une trace textuelle

Nous avons déjà à plusieurs reprises parlé de ces traces. Le logiciel "parametrage" permet à l'éducateur de visualiser une trace textuelle à l'écran ou d'en demander l'impression sur papier. Il permet également de supprimer une trace textuelle. Cette suppression peut s'avérer intéressante dans certains cas. La disquette de l'utilisateur peut contenir un grand nombre de fichiers "trace", de sorte qu'il soit impossible d'y placer les données nécessaires à la session des exercices que l'utilisateur a spécifiés. Un fichier "trace" peut également ne plus être d'aucune utilité.

3.3. Utilisation du logiciel

3.3.1. Comment exécuter le programme ?

Pour lancer le programme, il suffit d'allumer l'ordinateur, de mettre la disquette "Parametrage" dans le drive DF0:. Cliquez deux fois sur l'icône "Parametrage". Une fenêtre contenant les icônes "Parametrage" et "Préférences" apparaîtra. Cliquez deux fois rapidement sur l'icône "Parametrage". Après un certain temps, l'écran correspondant au menu (écran p1) apparaîtra.

3.3.2. Le menu principal

Quand vous lancez le programme, vous obtenez l'écran de sélection correspondant aux trois fonctionnalités de base (écran p1). Ainsi, le premier choix correspond à l'initialisation d'une disquette utilisateur.Le deuxième, à l'impression d'un fichier trace. Le troisième correspond à la suppression d'un fichier trace. La dernière option permet de quitter le programme. Pour effectuer l'un ou l'autre choix, cliquez sur la case correspondante. Nous détaillons ces fonctions ci-dessous.

3.3.3. Initialisation d'une disquette utilisateur

Rappelons qu'une disquette ne se rapporte qu'à un seul utilisateur. Le nom du volume de la disquette doit obligatoirement être "USER". Si vous ne possédez pas encore une telle disquette, vous devez en initialiser une et la renommer "USER" (les commandes "Initialise" et "Rename" du WorkBench permettent d'effectuer ces opérations).

Avant de passer en revue tous les écrans permettant l'initialisation, notez que certains d'entre eux contiennent , en bas à droite, un bouton "Abandon de la session". Cette case permet l'abandon de la préparation d'une session d'exercices et le retour au menu principal.

Le premier écran vous permet de spécifier le nom de l'utilisateur (écran p3). Le nom de l'utilisateur associé à la disquette introduite apparaît dans le cartouche. Les exercices définis seront destinés à cet utilisateur.

Si ce nom est conforme à votre souhait, cliquez dans la case qui suit le message "Nom de l'utilisateur" et tapez <Return>. Si ce n'est pas le cas, cliquez dans la case et tapez le nom de l'utilisateur désiré (en entrant indifféremment des majuscules ou minuscules) suivi de <Return>. La disquette associée à ce dernier vous sera demandée plus tard. Le nom est une chaîne de maximum 15 caractères. La touche <Backspace> vous permet d'effectuer des corrections.

Si la disquette est vierge, le cartouche est vide et la possibilité vous est offerte de fournir le nom de l'utilisateur auquel la disquette sera associée.

Dans l'écran qui suit (écran p 4), la disquette "Bibliothèque01" est demandée. Lorsque cette disquette est introduite, cliquez sur "OK" et patientez quelques secondes.

L'écran (écran p2) qui apparaît ensuite, vous demande d'introduire la disquette "USER". Lorsque cette disquette est introduite, cliquez sur "OK" et patientez quelques secondes.

Nous allons, maintenant, présenter la démarche d'initialisation d'un exercice de la session. Une session pouvant comporter jusqu' à 5 exercices, le processus doit alors être répété pour chacun des exercices.

Le premier écran d'initialisation (écran p5) vous demande le type de l'exercice. Cliquez sur le bouton correspondant à votre choix.

A. Définition d'un exercice de type "désignation"

Dans ce cas, l'écran suivant la demande du type d'exercice sera l'écran de saisie des paramètres (écran p12). Comme toujours, les valeurs par défaut sont affichées à droite de l'écran . Cliquez dans les cases correspondant à vos choix.

Certains de ces choix entraînerons des modifications d'autres paramètres. Ainsi, par exemple, si le type d'interaction pseudo-passif est choisi (voir le premier paramètre de la désignation), la silhouette présentée à l'écran ne peut pas être vide. Le paramètre "Silhouette vide ou non" sera donc bloqué à la valeur "NON". Pour débloquer ce paramètre, il suffit de changer le type d'interaction, et de le mettre, par exemple, sur "apprentissage".

Les contraintes appliquées à la définition des paramètres sont les suivantes.

Une interaction de type "passif" ou "évaluation" ou pseudo-passif (paramètre 1) n'utilise pas de nombre d'itérations (le paramètre 4 ne sera pas défini).

Pour une interaction de type "pseudo-passif" (toujours le paramètre 1), l'ordre d'énumération, n'est pas utilisé (paramètre 2 = "ordre"). La silhouette ne peut pas être vide (paramètre 5 = "non"). On ne peut utiliser les couleurs comme type de désignation, la détection doit donc se faire par cadre (paramètre 7 = "cadre").

Une silhouette vide (paramètre 5 = "non") ne peut pas utiliser un type de désignation par couleur car les organes sont invisibles dans ce cas (paramètre 7 = "cadre"). L'inverse est vrai également.

La désignation par nom (paramètre 8 = "nom") nécessite aussi une silhouette non vide(paramètre 5 = "non").

Certains paramètres ont des valeurs définies par le choix des exercices disponibles dans la rubrique "commentaires". En effet, certains graphiques, servant de base aux exercices, entraînent des contraintes sur les paramètres.

Les exercices disponibles dans "commentaires" sont de trois natures:

- "par système". L'exercice porte sur la désignation de systèmes (les cadres à désigner représentent des systèmes), et, c'est seulement dans ce cas qu'une animation peut être choisie comme feed-back (paramètre 6). La silhouette ne peut, alors, pas être vide (paramètre 5 = "non") et il n'est pas possible de détecter les pièces "par couleur" (paramètre 7 = "cadre"). La désignation des noms des systèmes n'est également pas possible (paramètre 8 = "image").

Dans tous les autres cas, donc, si l'exercice n'est pas "par système", le paramètre 6 sera bloqué à "non", il ne sera pas possible de choisir une animation.

- "par couleur". Le type de désignation "par couleur" n'est possible que pour certains exercices. Rappelons-nous les contraintes entraînées: une et une seule couleur pour chaque pièce. Dans ce type de désignation, les pièces ne peuvent être différenciées que par la couleur. Si ce type d'exercice est choisi, la silhouette ne peut être vide (paramètre 5 <-- "non").

- ni "par couleur" ni "par système". Il n'y a pas de contraintes dans ce cas.

Sélectionner "commentaires" vous permet de prendre connaissance des différents exercices proposés (écran p14). Pour choisir un exercice, revenez à l'écran p 12 et tapez le numéro correspondant à l'exercice souhaité dans le premier cadre suivant le texte "liste des énoncés".

L'accord avec les valeurs affichées pour tous les paramètres se fait en cliquant sur "FIN". Une fois que les paramètres ont leur valeur, il faut confirmer ou infirmer l'enregistrement de l'exercice (écran p15). L'écran suivant permet de signifier si vous voulez encore définir un exercice pour l'utilisateur (écran p16). La limite des 5 exercices atteinte est signalée par l'écran p20.

B. Définition d'un exercice de type "parcours"

C'est l'écran p13 qui permet de faire cette opération. Il se base sur le même principe que pour la saisie d'une désignation. Aucune contrainte ne lie les paramètres entre eux.

C. Fin de la préparation d'exercices

Il reste encore à définir deux paramètres guidant la session: le type de renforcement et le type de saisie du nom. C'est l'objet des deux écrans suivant (écrans p18 et p19).

Le logiciel vous demandera ensuite d'introduire la disquette de l'utilisateur (écran p2). Introduisez-la, cliquez sur "OK" et patientez quelques secondes.

Introduisez ensuite les disquettes "Bibliothèques" demandées par le logiciel (écran p4). Cliquez sur "OK" après chaque opération.

Il est ensuite nécessaire d'introduire la disquette utilisateur (écran p2) Cliquez sur "OK".

L'initialisation d'une disquette utilisateur se termine par le retour au menu principal.

3.3.4. Impression d'un fichier trace

Lorsque vous avez sélectionné "impression d'un fichier trace", l'écran p2 apparaît, introduisez le disquette "USER" dans l'un des drive de l'ordinateur, cliquez sur "OK" pour passer à l'écran p6.

Un bouton "Abandon" permet de revenir au menu principal. Pour sélectionner le nom du fichier des "traces"que vous désirez visualiser, cliquez dans la case se trouvant sous le message "Nom de l'utilisateur + numéro de la session" et indiquez le nom de l'utilisateur suivi du numéro de la version que

vous voulez consulter. Il peut y avoir des blancs entre le nom et le numéro. De plus, le numéro de "trace" est toujours composé de deux chiffres. Ainsi, par exemple, si vous désirez visualiser la trace n°5 de Alain, entrez "alain05". Validez votre saisie en pressant la touche <Return>. Si le fichier spécifié n'existe pas, le message "Nom de fichier inexistant" apparaît. Vous pouvez alors modifier le nom introduit.

Si le fichier existe, il vous faut spécifier le support de la visualisation (écran p7) . Cliquez sur le bouton "Impression écran" si vous désirez un sortie au moniteur de la machine ou sur le bouton "Impression imprimante" si vous désirez une sortie sur papier. . Dans ce dernier cas, l'écran suivant (écran p8) vous demandera de préparer votre imprimante. Notez que vous devez avoir initialisé, avant de débuter le programme, vos "Préférences".

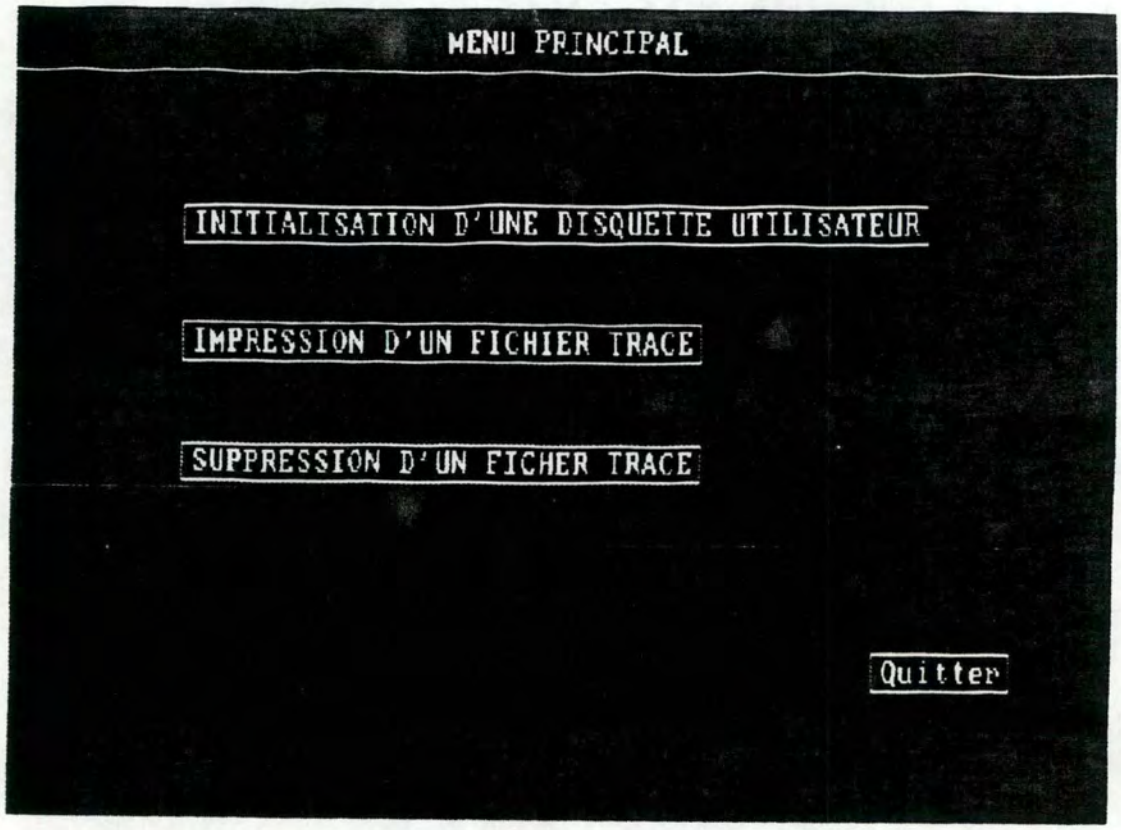
Dans le cas d'une impression écran (écran p9), le bouton "OK" vous permet de passer, chaque fois, à l'écran suivant. Après la visualisation de la trace graphique, ce bouton permet le retour au menu principal.

3.3.5. Suppression d'un fichier trace

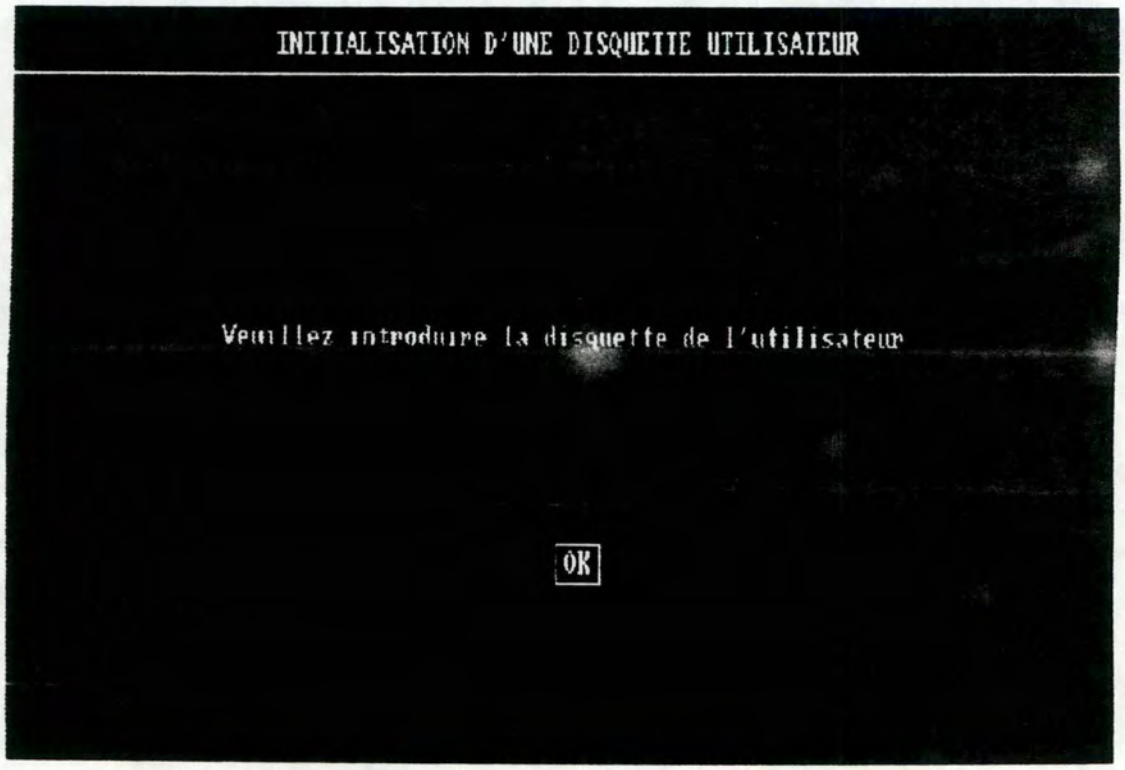
Lorsque vous avez sélectionné "Suppression d'un fichier trace", l'écran p2 apparaît, introduisez le disquette "USER" dans l'un des drive de l'ordinateur, cliquez sur "OK" pour passer à l'écran p10.

Un bouton "Abandon" permet de revenir au menu principal. Pour sélectionner le nom du fichier des "traces"que vous désirez supprimer, cliquez dans la case se trouvant sous le message "Nom de l'utilisateur + numéro de la session" et indiquez le nom de l'utilisateur suivi du numéro de la version que vous voulez supprimer, comme dans la section précédente. Si le fichier spécifié n'existe pas, le message "Nom de fichier inexistant" apparaît. Vous pouvez alors modifier le nom introduit.

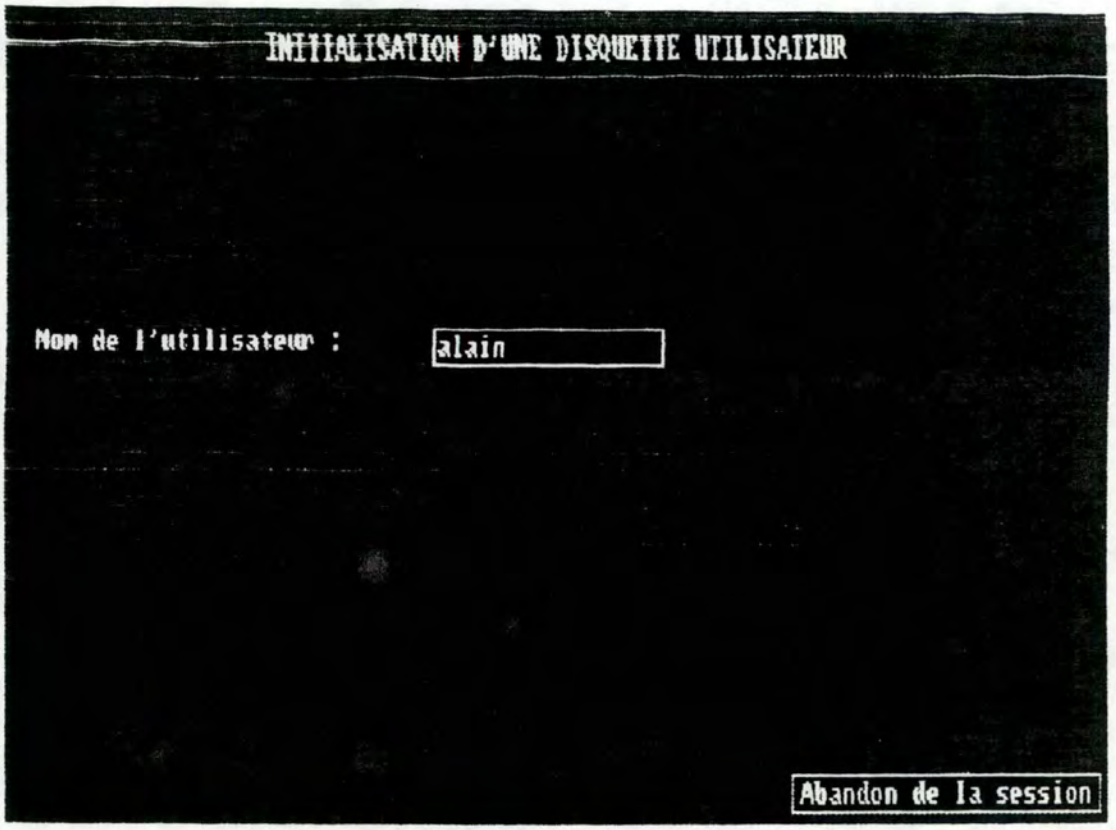
Lorsque le nom est accepté, l'écran p 11 vous permet de confirmer la suppression ou de l'infirmer. Dans les deux cas le retour à l'écran 10 s'opère.



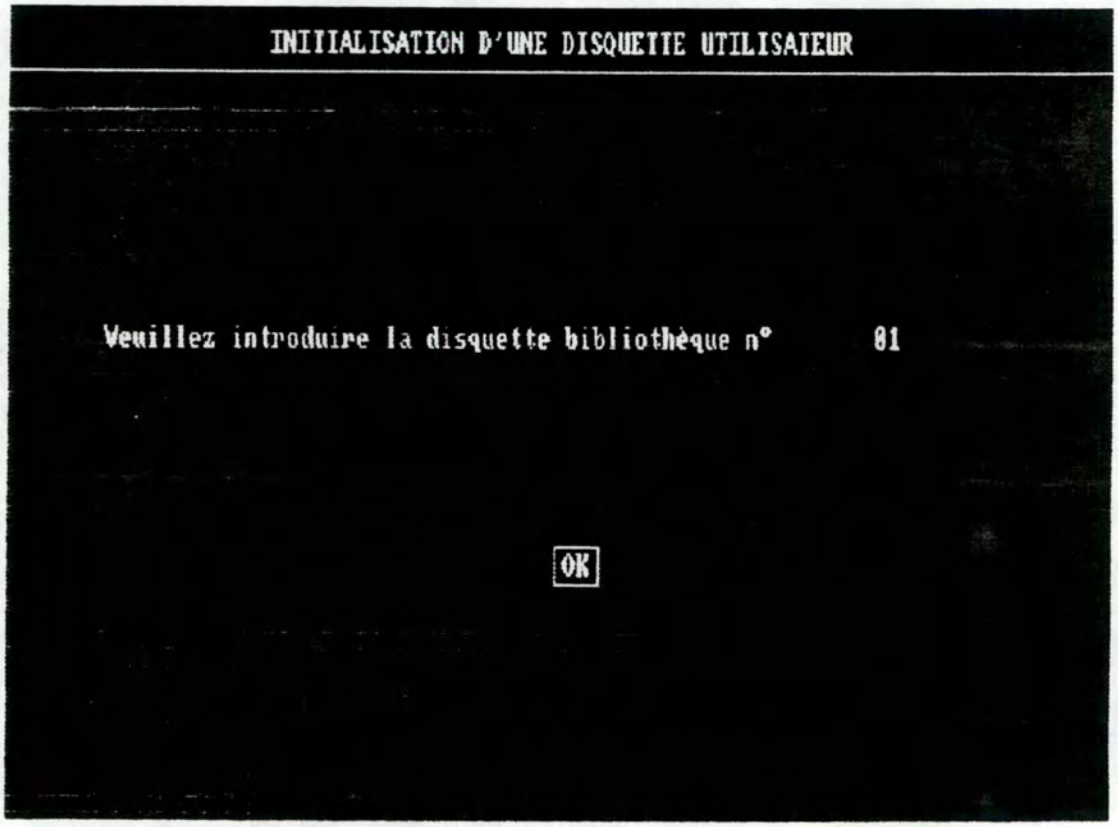
Ecran p1: Menu principal.



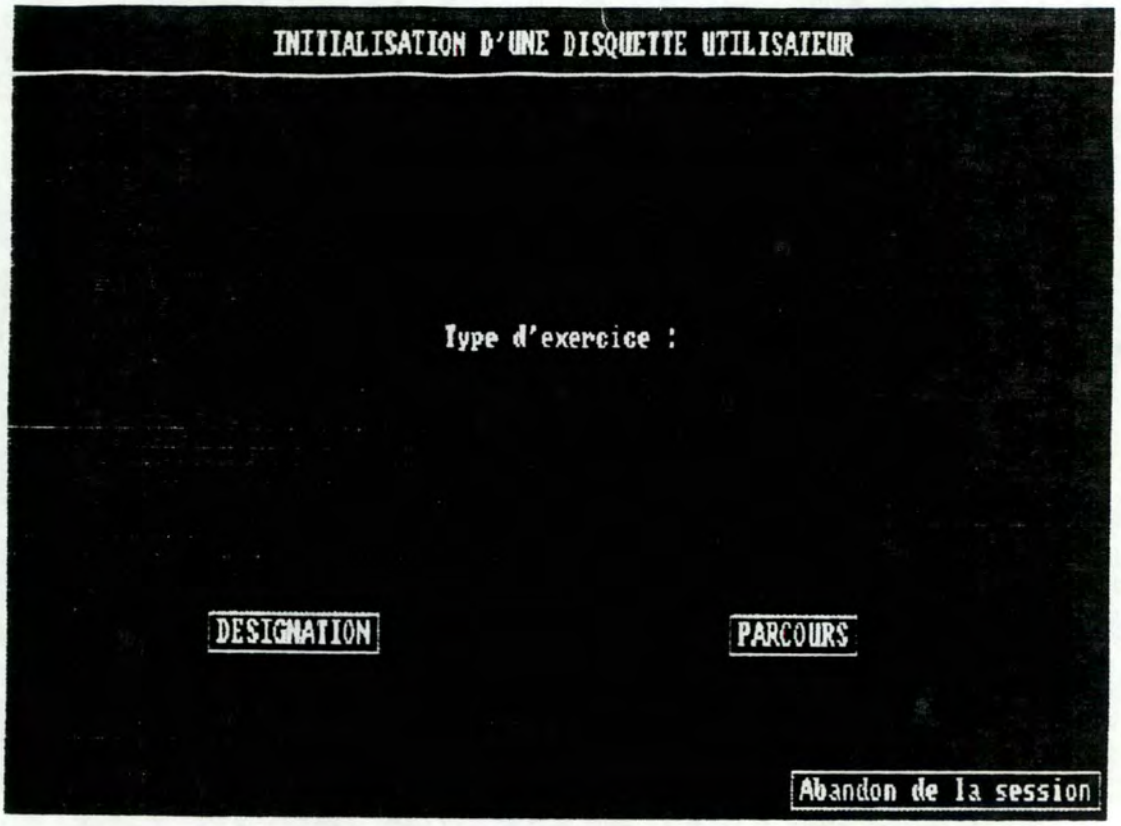
Ecran p2: Demande d'introduction de la disquette utilisateur.



Ecran p3: Saisie du nom de l'utilisateur.

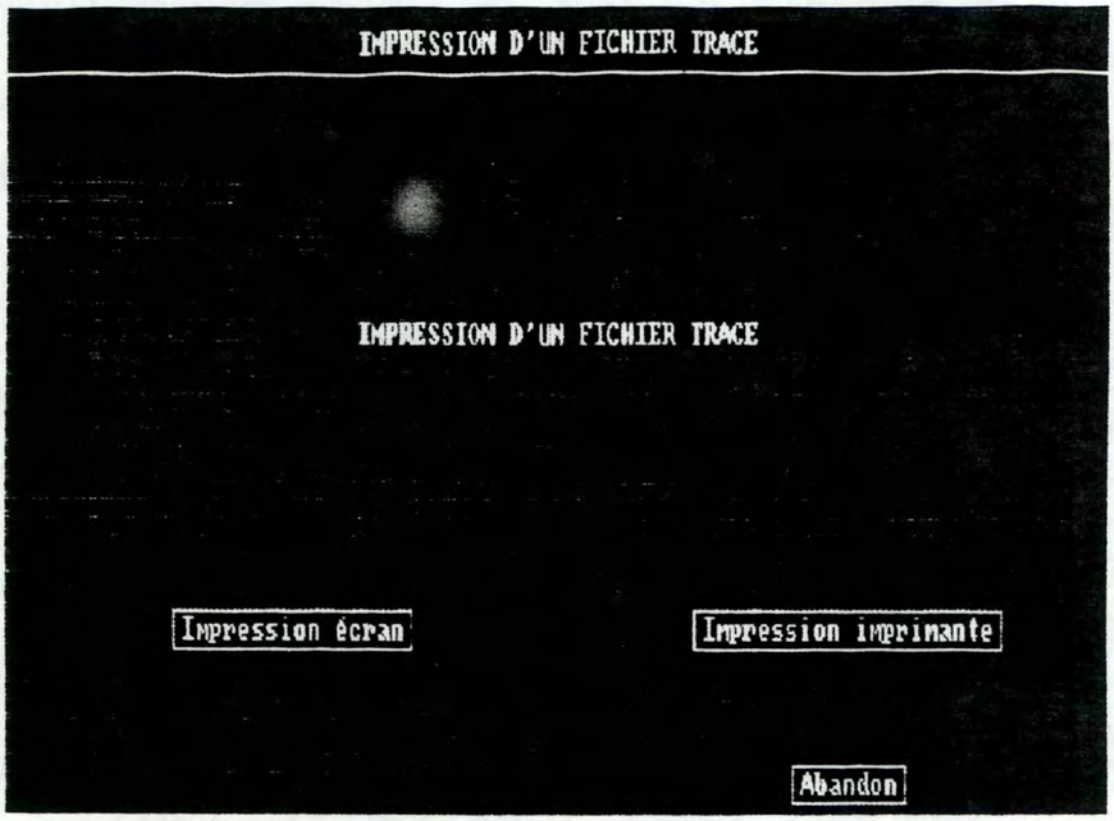


Ecran p4: Demande d'introduction de la disquette bibliothèque.



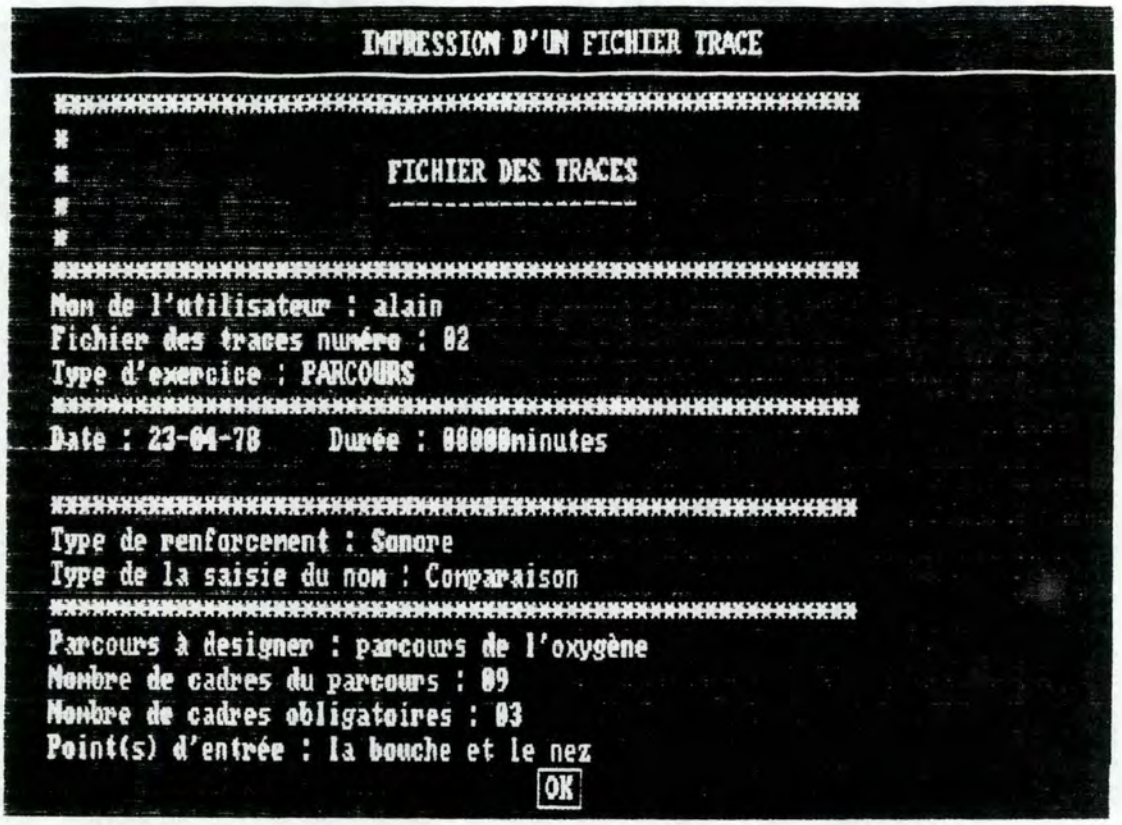
Ecran p5: Choix du type d'exercice à initialiser.

Ecran p6: Saisie du nom du fichier des traces à visualiser.



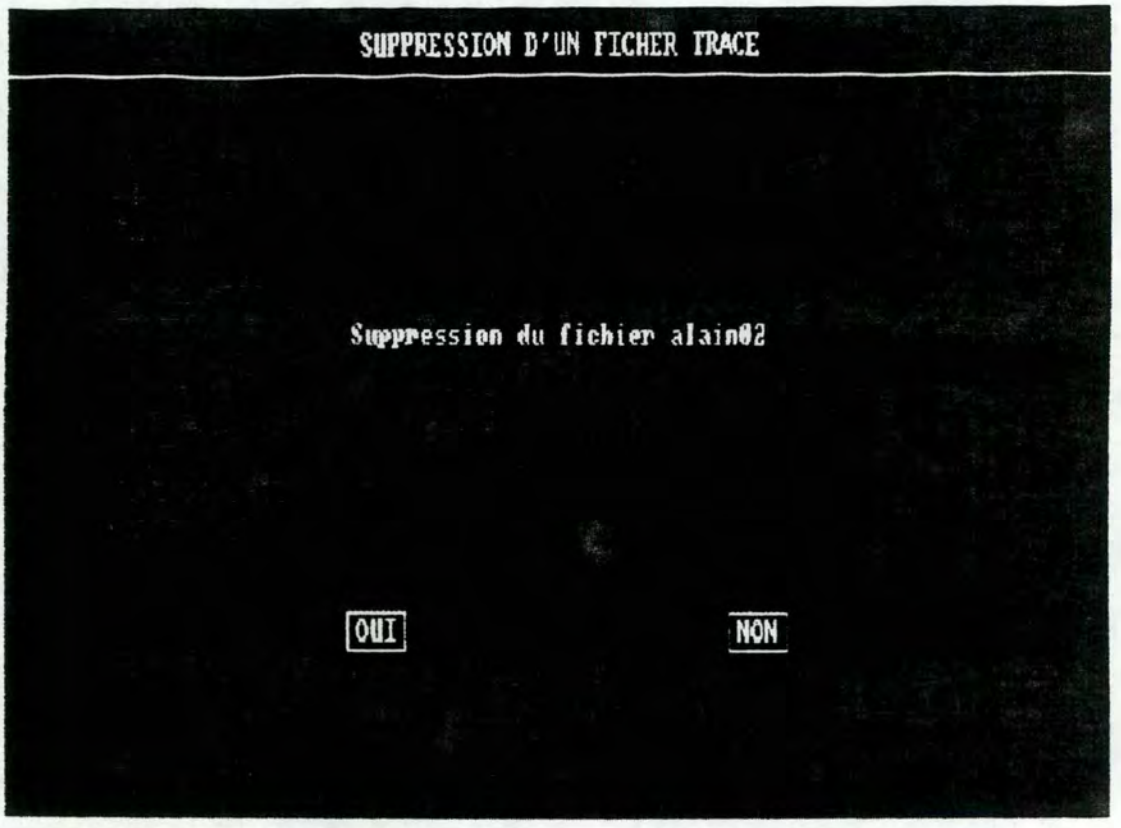
Ecran p7: Choix du type de visualisation du fichier trace.

Ecran p8: Demande de préparation de l'imprimante.

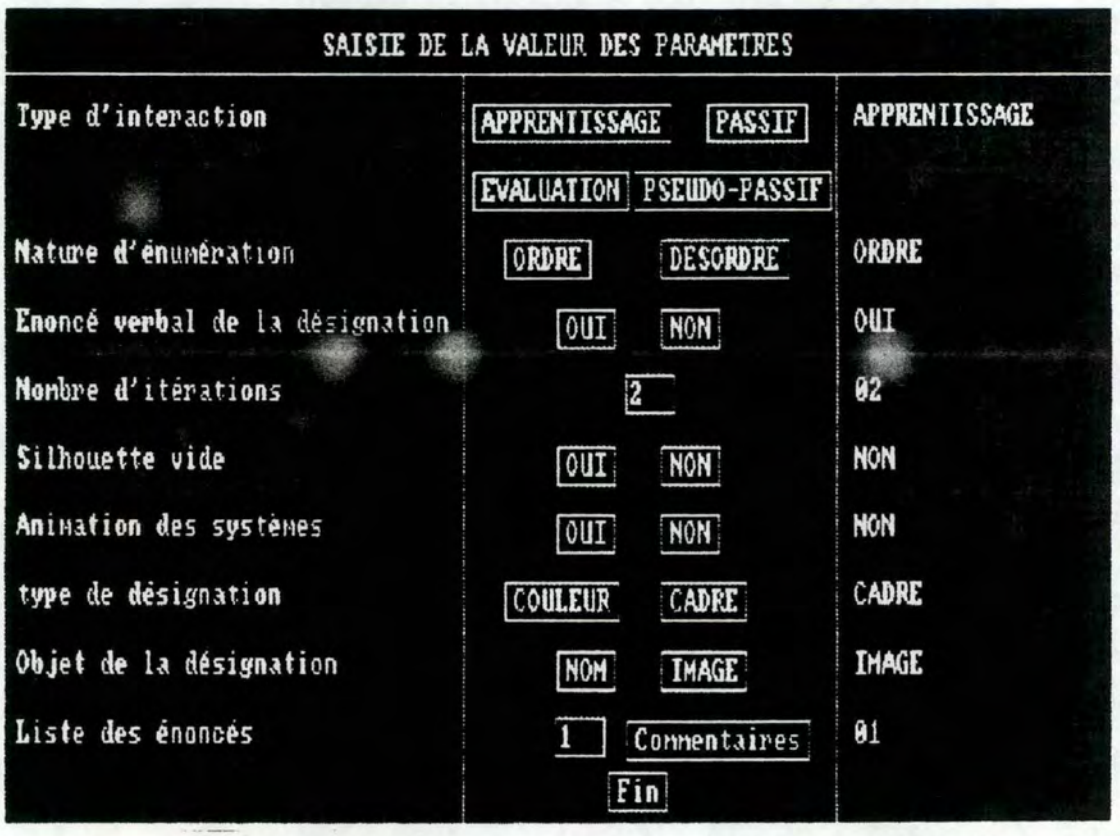


Ecran p9: Visualisation écran d'un fichier trace.

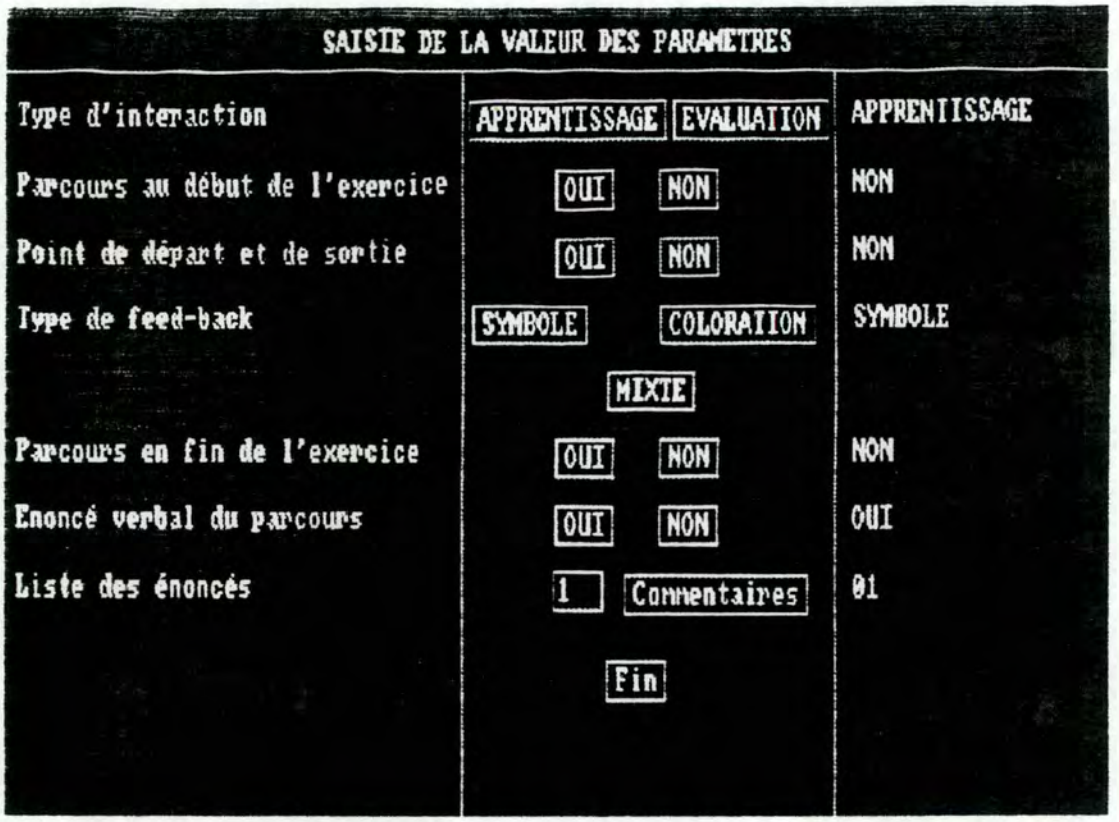
Ecran p10: Saisie du nom du fichier des traces a supprimer.



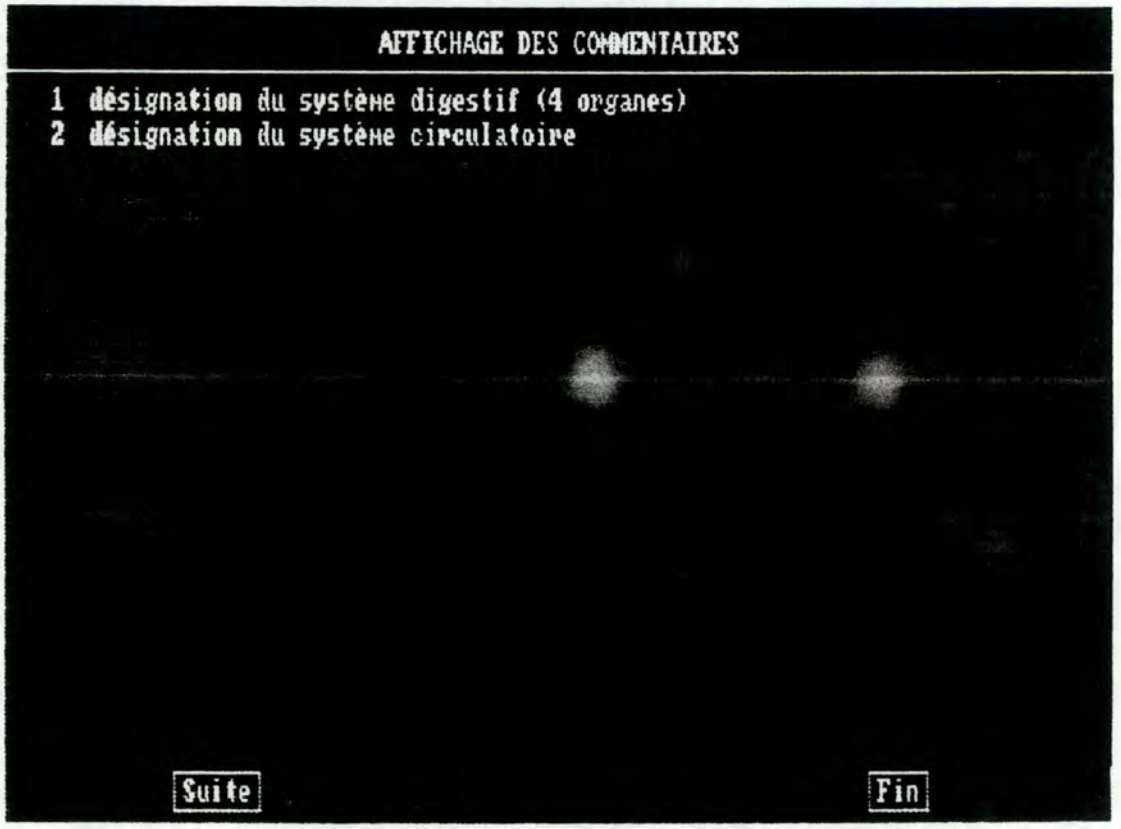
Ecran p11: Confirmation de la suppression d'un fichier.



Ecran p12: Saisie de la valeur des paramètres pour la désignation.



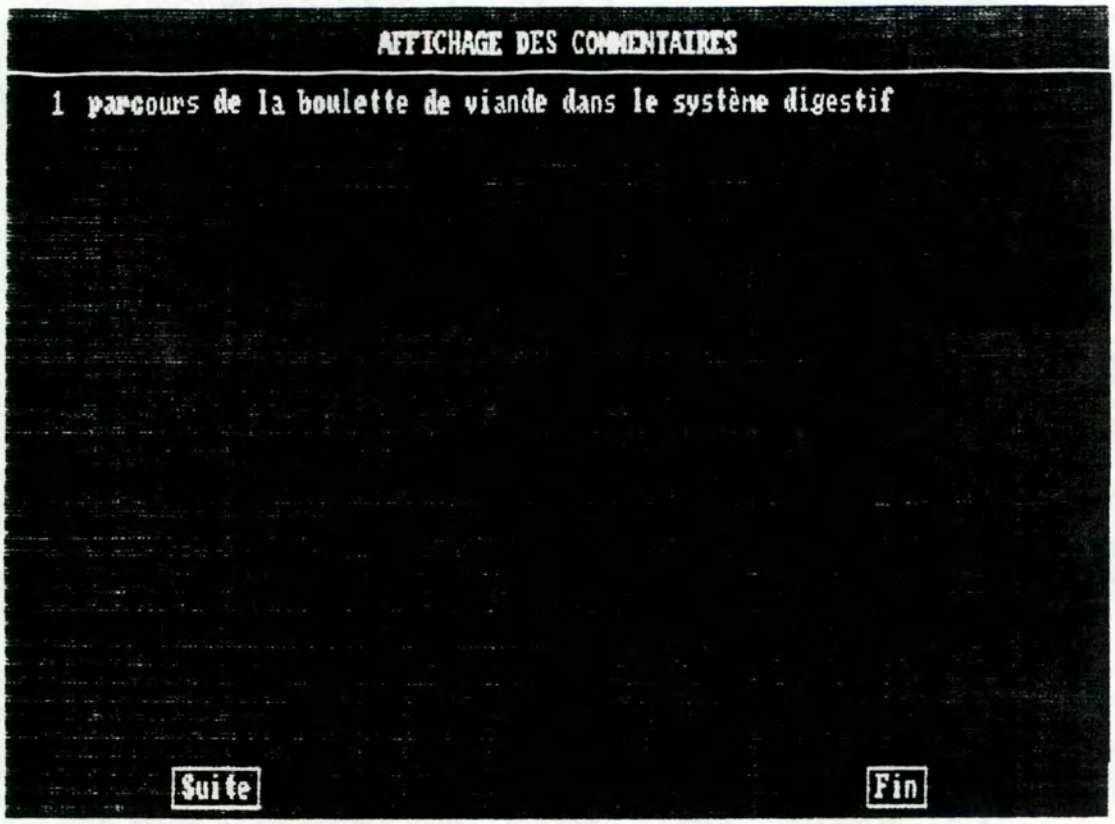
Ecran p13: Saisie de la valeur des paramètres pour le parcours.



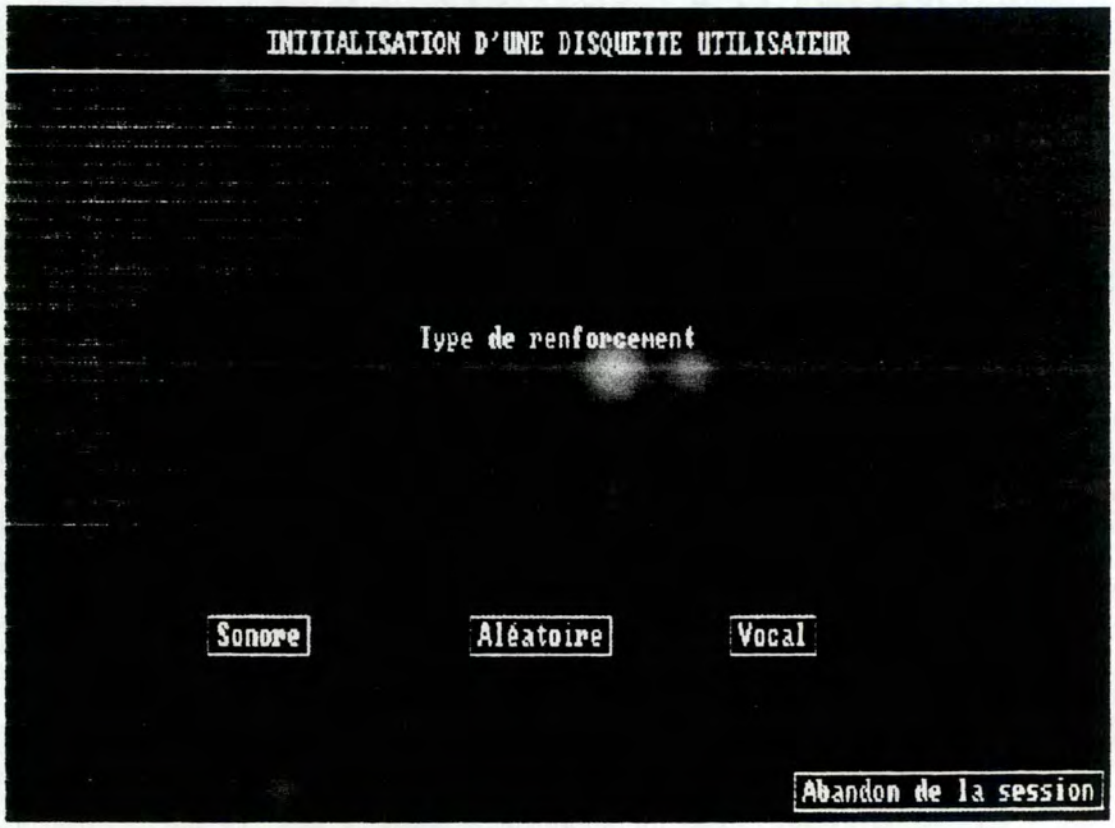
Ecran p14: Affichage des commentaires de la désignation.

Ecran p15: Demande de confirmation de l'enregistrement de l'exercice.

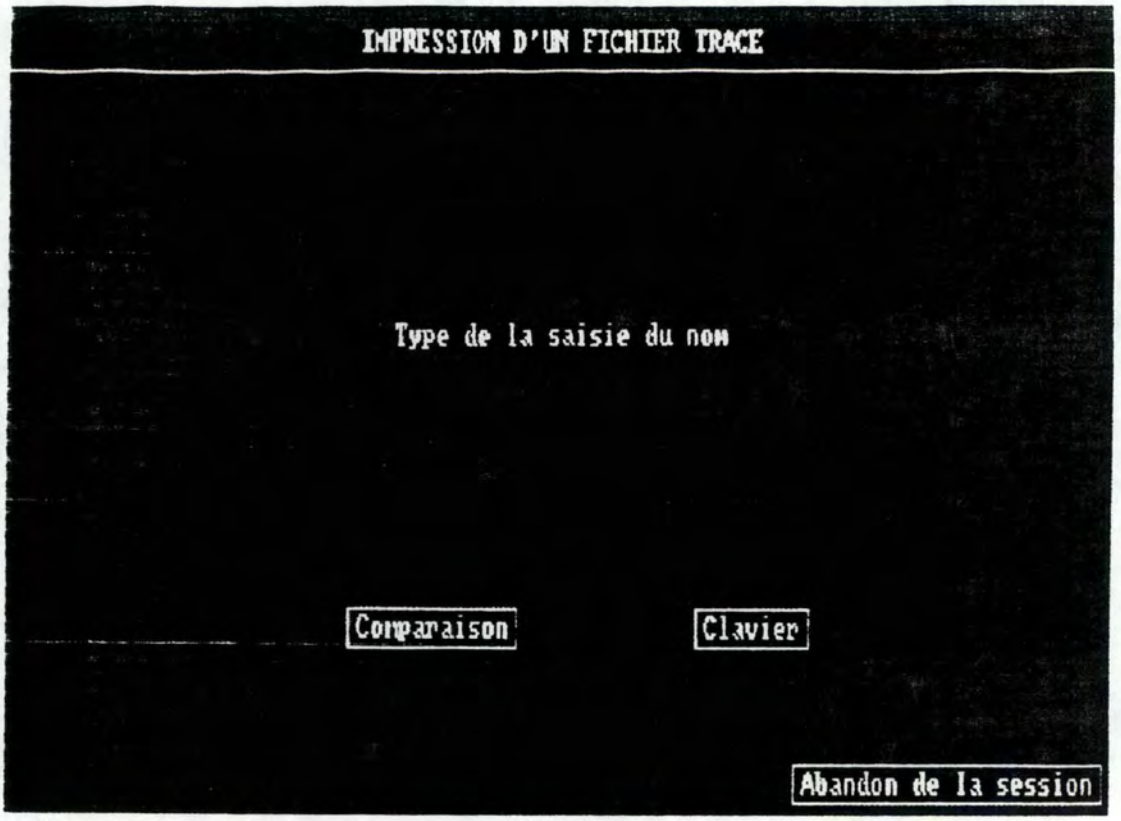
Ecran p16: Demande d'initialisation d'un exercice supplémentaire.



Ecran p17: Affichage des commentaires du parcours.



Ecran p18: Demande du type de renforcement.



Ecran p19: Choix du type de saisie du nom.

Ecran p20: Message signalant l'atteinte de la limite du nombre d'exercices initialisés.

Ecran p21: Message d'attente.

ANNEXE 2 : Quelques exemples de traces textuelles

```
**********************************************************************
*                                                                    *
*                        FICHIER DES TRACES                          *
*                        ------------------                          *
*                                                                    *
**********************************************************************
Nom de l'utilisateur : pierre
Fichier des traces numéro : Ø7
Type d'exercice : PARCOURS
**********************************************************************
Date : Ø8-Ø7-91     Durée : ØØØ4minutes

**********************************************************************
Type de renforcement : Sonore
Type de la saisie du nom : Comparaison
**********************************************************************
Parcours à désigner : parcours de la boulette
Nombre de cadres du parcours : 22
Nombre de cadres obligatoires : Ø5
Point(s) d'entrée : la bouche
l'oesophage
l'estomac
les intestins
Point(s) de sortie : l'anus
**********************************************************************
Liste des valeurs attribuées aux paramètres
1 -- Interaction de type apprentissage
2 -- Enoncé vocal : OUI
3 -- Parcours en début : NON
4 -- Montre point de départ et d'arrivée : NON
5 -- Parcours à la fin : OUI
6 -- Feedback : symbole
**********************************************************************
Est_ce bien ton nom?
Confirmation : Il clique sur 'OUI'
Il montre une partie comprise dans la bouche
Il montre une partie comprise dans l'estomac
Il se trompe
cause : Il a oublié  l'oesophage
Nombre de cadres obligatoires désignés : Ø1 / Ø5
Nombre de cadres désignés : Ø1
Il recommence le parcours
Il montre une partie comprise dans l'oesophage
Il se trompe
cause : Il a oublié  la bouche
Nombre de cadres obligatoires désignés : ØØ / Ø5
Nombre de cadres désignés : ØØ
Il recommence le parcours
Il montre une partie comprise dans la bouche
Il montre une partie comprise dans l'oesophage
Il montre une partie comprise dans la bouche
Il se trompe
cause : Il est revenu en arrière
Nombre de cadres obligatoires désignés : Ø2 / Ø5
Nombre de cadres désignés : Ø2
Il recommence le parcours
Il montre une partie comprise dans la bouche
Il montre une partie comprise dans l'oesophage
Il montre une partie comprise dans la bouche
Il se trompe
cause : Il est revenu en arrière
Nombre de cadres obligatoires désignés : Ø2 / Ø5
Nombre de cadres désignés : Ø2
Il recommence le parcours
Il montre une partie comprise dans la bouche
Il montre une partie comprise dans l'oesophage
```

Il montre une partie comprise dans l'oesophage
Il montre une partie comprise dans l'estomac
Il montre une partie comprise dans l'estomac
Il montre une partie comprise dans l'estomac
Il montre une partie comprise dans l'estomac
Il montre une partie comprise dans les intestins
Il clique sur l'oreille
Il clique sur 'FIN'
Veux_tu finir?
Confirmation : Il clique sur 'NON'
Il clique sur 'STOP'
Veux_tu stopper?
Confirmation : Il clique sur 'NON'
Il clique sur 'FIN'
Veux_tu finir?
Confirmation : Il clique sur 'OUI'

```
**********************************************************************
*                                                                    *
*                        FICHIER DES TRACES                          *
*                        ------------------                          *
*                                                                    *
**********************************************************************
Nom de l'utilisateur : françois
Fichier des traces numéro : 21
Type d'exercice : PARCOURS
**********************************************************************
Date : 24-08-91     Durée : ØØØØ1minutes

**********************************************************************
Type de renforcement : Sonore
Type de la saisie du nom : Comparaison
**********************************************************************
Parcours à designer : parcours de l'oxygène
Nombre de cadres du parcours : Ø9
Nombre de cadres obligatoires : Ø3
Point(s) d'entrée : la bouche et le nez
la trachée
Point(s) de sortie : les poumons
**********************************************************************
Liste des valeurs attribuées aux paramètres
1 -- Interaction de type évaluation
2 -- Enoncé vocal : NON
3 -- Parcours en début : NON
4 -- Montre point de départ et d'arrivée : NON
5 -- Parcours à la fin : NON
6 -- Feedback : coloration
**********************************************************************
Il montre une partie comprise dans la bouche et le nez
Il montre une partie comprise dans les poumons
Nombre de cadres désignés : Ø2
```

```
**********************************************************************
*                                                                    *
*                        FICHIER DES TRACES                          *
*                        ------------------                          *
*                                                                    *
**********************************************************************
Nom de l'utilisateur : alain
Fichier des traces numéro : 13
Type d'exercice : DESIGNATION
**********************************************************************
Date : Ø8-Ø5-91     Durée : ØØØØ1minutes

**********************************************************************
Type de renforcement : Sonore
Type de la saisie du nom : Comparaison
**********************************************************************
Liste des valeurs attribuées aux paramètres
1 -- Interaction de type pseudo_passif
2 -- Enoncé vocal : NON
3 -- Nombre d'itérations avant reproposition : ØØ
4 -- Ordre énumération logique : OUI
5 -- Silhouette vide : NON
6 -- Animation : NON
7 -- Détection par cadre
8 -- Désignation des images
**********************************************************************
Il doit montrer un organe
Il montre l'estomac
Il doit montrer un organe
Il montre les intestins
Il doit montrer un organe
Il montre l'estomac
Il doit montrer un organe
Il montre le foie
Il doit montrer un organe
Il montre l'oesophage
```

```
**********************************************************************
*                                                                    *
*                        FICHIER DES TRACES                          *
*                        ------------------                          *
*                                                                    *
**********************************************************************
Nom de l'utilisateur : alain
Fichier des traces numéro : 24
Type d'exercice : DESIGNATION
**********************************************************************
Date : Ø9-Ø8-91     Durée : ØØØØ1minutes

**********************************************************************
Type de renforcement : Sonore
Type de la saisie du nom : Comparaison
**********************************************************************
Liste des valeurs attribuées aux paramètres
1 -- Interaction de type apprentissage
2 -- Enoncé vocal : NON
3 -- Nombre d'itérations avant reproposition : Ø1
4 -- Ordre énumération logique : NON
5 -- Silhouette vide : NON
6 -- Animation : NON
7 -- Détection par cadre
8 -- Désignation des noms
**********************************************************************
Il doit montrer le nom : l'oesophage
Il montre le nom : les intestins
Il doit montrer le nom : l'oesophage
Il clique sur l'oreille
Il montre le nom : l'oesophage
Il doit montrer le nom : le foie
Il clique sur l'oreille
Il montre le nom : le foie
Il doit montrer le nom : les intestins
Il clique sur 'FIN'
Veux_tu finir?
Confirmation : Il clique sur 'OUI'
```

```
**********************************************************************
*                                                                    *
*                        FICHIER DES TRACES                          *
*                        ------------------                          *
*                                                                    *
**********************************************************************
Nom de l´utilisateur : bernard
Fichier des traces numéro : 17
Type d´exercice : DESIGNATION
**********************************************************************
Date : 12-Ø8-91      Durée : ØØØØ2minutes

**********************************************************************
Type de renforcement : Sonore
Type de la saisie du nom : Comparaison
**********************************************************************
Liste des valeurs attribuées aux paramètres
1 -- Interaction de type apprentissage
2 -- Enoncé vocal : NON
3 -- Nombre d´itérations avant reproposition : Ø1
4 -- Ordre énumération logique : NON
5 -- Silhouette vide : NON
6 -- Animation : OUI
7 -- Détection par cadre
8 -- Désignation des images
**********************************************************************
Il doit montrer ce qui permet de respirer
Il montre ce qui permet de respirer
Il doit montrer ce qui permet de boire
Il clique sur ´FIN´
Veux_tu finir?
Confirmation : Il clique sur ´OUI´
```

ANNEXE 3 : Contenu des fichiers

I. Les fichiers de "Parametrage"

A. Les fichiers en entrée.

1. Catalogue_desig (se trouve sur la disquette BIBLIO01)

Contient les commentaires des exercices de type désignation disponibles et les informations relatives aux contraintes de l'exercice, à savoir, s'il s'agit d'un exercice de désignation de systèmes, exercice de désignation "par couleur" ou pas.

Structure du fichier:
<entier1> <entier2> <entier3> <entier4> <string>
autant de fois qu'on veut définir une ligne de commentaire.

<entier1> = numéro de la disquette BIBLIO où se trouvent tous les fichiers nécessaires à l'exercice.
Rem : Ce nombre sera référencé plus tard comme <numbiblio>.
<entier2> = 0 si c'est une désignation qui n'est pas "par système".
= 1 si c'est une désignation "par système".
<entier3> = 0 si c'est une désignation libre
(ie on a le choix: par couleur ou par cadre).
=1 si c'est une désignation obligatoirement par couleur.
<entier4> = nombre de mots du commentaire de l'exercice (<string>).
<string> = string contenant le commentaire et ayant <entier4> mots. Ce commentaire est affiché à l'écran lors de la saisie de la valeur des paramètres.

Remarques :
Chacun de ces éléments est séparé par au moins un blanc.

Exemple :
1 0 0 4 désignation du système digestif
1 0 1 4 désignation du système circulatoire

2. Catalogue_parcours (se trouve sur BIBLIO01)

Contient les commentaires des exercices de type parcours. Il est constitué de la même manière que le premier fichier.

Structure du fichier:

<entier1> <entier2> <string>

<entier1> = même signification que <entier1> du fichier catalogue_desig.
<entier2> = même signification que <entier4> du fichier catalogue_desig.
<string> = string contenant le commentaire et ayant <entier2> mots.

Remarques :

Chacun de ces éléments est séparé par au moins un blanc.

Exemple :

1 10 parcours de la boulette de viande dans le système digestif
9 8 parcours de l'oxygène à travers le système respiratoire

3. désignation_info (se trouve sur BIBLIO01).

Contient les renseignements sur les exercices de désignation.

Structure du fichier:

"d" <graphe> <fichd1> <paramd> <entier> <return>

"d" = caractère "d" signifiant que l'exercice est une désignation.
<graphe> = nom du graphe support de l'exercice (ce fichier se trouve sur la disquette BIBLIO<numbiblio>, est défini dans le fichier 1 et son nom se trouve également dans le fichier liste_image (défini dans le point 6) correspondant à l'exercice).

<fichd1> = nom du fichier contenant les informations sur les énoncés (cadres) constituant l'exercice.
Ce fichier est défini dans la partie C.

<paramd> = nom du fichier paramètre contenant la valeur des paramètres de l'exercice. Ce fichier est composé des paramètres qui sont des entiers séparés par au moins un blanc.
Rem : son nom est: "param"<type d'exercice><numéro dans le catalogue>.
<numéro dans le catalogue> est le numéro de la ligne dans désignation_info. Ce numéro sera appelé par la suite <num cat desig>.

<entier> = 1 si exercice de désignation des systèmes.
= 0 sinon.

Remarques :
Chacun de ces éléments est séparé par au moins un blanc.
La $i^{ème}$ ligne du fichier correspond à l'exercice dont les commentaires se trouvent à la $i^{ème}$ ligne du fichier catalogue_desig.

Exemple :
d corps fichdesig paramd3 1 <return>

4. parcours_info (se trouve sur BIBLIO01)

Contient les renseignements des exercices de type parcours

Structure du fichier :
"p" <graphe> <fichp1><ficho1><paramp1> <entier> <return>

"p" = caractère "p" signifiant que l'exercice est un exercice de type parcours.
<graphe> = nom du graphe support de l'exercice (ce fichier se trouve sur la disquette BIBLIO<numbiblio> et son nom dans le

fichier liste_image (défini dans le point 6) correspondant à l'exercice).

<fichp1> = nom du fichier contenant les coordonnées des cadres du parcours.
Ce fichier est défini dans la partie C.

<ficho1> = nom du fichier contenant les coordonnées des cadres obligatoires.
Ce fichier est défini dans la partie C.

<paramp1> = nom du fichier paramètre contenant la valeur des paramètres de l'exercice parcours.
Rem: son nom est : "param"<type d'exercice><numéro dans le catalogue>. <numéro dans le catalogue> est le numéro de la ligne dans parcours_info. Ce numéro sera appelé par la suite <num cat parc>.

Remarques :

Chacun de ces éléments est séparé par au moins un blanc.

La $i^{ème}$ ligne du fichier correspond à l'exercice dont les commentaires se trouvent à la $i^{ème}$ ligne du fichier catalogue_parcours.

Exemple :

p respiratoire fichparc fichobl paramp2 <return>

5. Les fichiers-sons (se trouvent sur BIBLIO<numbiblio>)

Contiennent chacun la liste des fichiers "son" d'un exercice de type <type_ex> correspondant à la <num cat desig>^{ième} ligne de catalogue_desig ou à la <num cat parc>^{ième} ligne de catalogue_parcours.

Le nom des fichiers sons est composé comme suit:
liste_son<type_ex><num cat parc>/<num cat desig> (selon la valeur de <type_ex> : liste_sond<num cat desig> ou liste_sonp<num cat parc>).

Exemple de fichier "son" associé au 5ème exercice de la liste l'exercices, exercice de type parcours :

iste_sonp5 contient: son1 son2 son3 son4 son5 <return>

Chacun des noms de fichiers a, maximum, 15 caractères.

6. Les fichiers-images (se trouvent dans BIBLIO01<numbiblio>)

Ils présentent la même structure que les fichiers-sons.

Le nom de la liste de fichiers est composé comme suit:
iste_image<type_ex><num cat desig>/<num cat parc> (selon la valeur de :type_ex> : liste_imaged<num cat desig> ou liste_imagep<num cat ›arc>).

Exemple de fichier "image" associé au 6ème exercice de la liste des 'xercices, exercice de type désignation :

iste_imaged6 contient: im1 im2 im3 <return>

Chacun des noms de fichiers a, maximum, 15 caractères.

3. Les fichiers en sortie

Les fichiers créés par "Parametrage" sont :

- les fichiers de nom param<type_ex> <num cat desig>/ <num cat parc>
- le fichier de nom donnee_session
- le fichier de nom <nom_user>.version. Ce fichier est créé une seule ois par "Parametrage" lors de la première initialisation d'une disquette ıser. <nom_user> est le nom de l'utilisateur à qui est destiné la disquette.

REMARQUE :

Tous les fichiers relatifs à un exercice doivent se trouver sur une même disquette (BIBLIO<num_biblio>).

C. Les fichiers des énoncés

C1. La désignation

Un exercice de type désignation est composé de plusieurs énoncés qui feront l'objet d'une question lors le l'exécution de "Corps interne". Les différentes informations nécessaires sur ces énoncés sont regroupées dans un fichier <fichd1> dont le nom se trouve dans le fichier designation_info (voir plus avant).

Structure du fichier :

<énoncé 1> <énoncé 2> ..... <énoncé n>

Le <énoncé i> correspondent aux données sur le $i^{ème}$ énoncé (partie à désigner de l'exercice).

<énoncé i> a la structure suivante :

<couleur> <nom de l'énoncé> <nom du fichier de la brush> <nom du fichier son> <coordonnées>

<couleur> = le numéro de la couleur utilisée pour la désignation "par couleur". Il correspond à la position qu'occupe la couleur dans la palette de couleur définie dans "Corps interne".

<nom de l'énoncé> = un string contenant le nom de la partie à désigner. Il est composé d'un entier indiquant le nombre de mots qui composent le nom et du nom

<nom du fichier de la brush> = string contenant le nom du fichier contenant le dessin attaché à

l'énoncé. Ce nom se trouve dans le fichier liste_image (défini dans le point 6 de la partie B) correspondant à l'exercice).

<nom du fichier son> = string contenant le nom du fichier contenant le son attaché à l'énoncé (son digitalisé). Ce nom se trouve dans le fichier liste_son (défini dans le point 5 de la partie B) correspondant à l'exercice.

<coordonnées> = 4 entiers correspondant : aux coordonnées du **point supérieur gauche** (les 2 premiers entiers), la longueur (le troisième) et la hauteur (le quatrième) du cadre. Les coordonnées sont **relatives** au coin supérieur gauche du graphique support à l'exercice.

Remarques :

Chaque <énoncé i> est séparé par au moins un blanc ainsi que chaque élément composant les <énoncé i> (pas de caractère de fin de ligne).

Il ne peut y avoir plus de 15 énoncés pour un exercice (<énoncé i>).

Si l'exercice peut faire l'objet d'un exercice de désignation "par nom" (objet d'un paramètre), les <énoncé i> ne peuvent dépasser 11.

L'ordre des <énoncé i> détermine l'ordre d'énumération des exercices (faisant l'objet d'un paramètre).

Exemple :

1 2 les poumons poum.info poum.son 25 30 10 10 4 2 les bronches bronche.info bronche.son 20 30 5 5

C2. Le parcours

Un exercice de type parcours est composé de plusieurs cadres. Les informations relatives à ces derniers se trouvent dans le fichier <fichp1> dont le nom se trouve dans parcours_info. De plus un exercice de ce type est caractérisé par des ensembles, des classes de cadres obligatoires c'est-à-

dire des cadres par où l'utilisateur doit "passer" pour indiquer le parcours (une classe correspond souvent à un organe). Les informations relatives à ces classes se trouvent dans les fichiers <ficho1> dont le nom se trouve dans parcours_info.

Structure de <fichp1>

<couleur> <nom du symbole> <nom du fichier de la brush du symbole> <nom du fichier son du symbole> <cadres 1> <cadres 2> ..... <cadres n>

<couleur> = numéro de la couleur choisie pour le feed-back (coloration).
<nom du symbole> = un string contenant le nom de la classe.
Il est composé d'un entier indiquant le nombre de mots qui composent le nom et du nom (ex: 2 la boulette).

<nom du fichier de la brush du symbole> = Ce nom se trouve dans le fichier liste_image (défini dans le point 6 de la partieB) correspondant à l'exercice

<nom du fichier son du symbole> = string. Ce nom se trouve dans le fichier liste_son (défini dans le point 5 de la partie B) correspondant à l'exercice).
<cadre i> = données relatives au(x) cadre(s) numéro i du parcours.
<cadre i> a la structure suivante :

<nb cadres de numéro i> <ensemble des coordonnées>

<nb cadres de numéro i> = nombre de cadres ayant le numéro i (1 ou 2)

<ensemble des coordonnées> : composé de <nb cadres de numéro i> <coordonnées> où <coordonnées> est défini de la même manière que dans <fichd1>. S'il y a 2 cadres de même numéro il y aura 2 <coordonnées> soit 8 entiers.

Remarques :

Tous les éléments décrits ci-dessus sont séparés par au moins un blanc.

Le nombre maximum de cadres d'un parcours est 50.

Les coordonnées des cadres doivent être placées dans le fichier dans l'ordre du parcours. C'est cet ordre qui détermine le sens du parcours.

Exemple :

5 2 la boulette sbou son5 1 x1 y1 l1 h1 2 x2 y2 l2 h2 x3 y3 l3 h3

Structure de <ficho1>

<classe 1> <classe 2>..... <classe n>

Ces éléments sont séparés par au moins un blanc

<classe i> a la structure suivante :

<nom de la classe> <nb de cadres composant la classe i> <numéros de ces cadres>

<nom de la classe> = string composé du nombre de mots du nom et du nom. Le nom d'une classe est souvent le nom d'un organe.

<nb de cadres composant la classe i> = entier, taille de la classe i.

<numéros de ces cadres> = les numéros des cadres se trouvant dans la classe i. Ces numéros sont les numéros d'ordre dans le fichier <fichp1>

Remarques

Les numéros des cadres sont triés par ordre croissants.

Le nombre maximum de classes est 50.

Exemple`:

2 la bouche 1 1 2 la trachée 3 2 3 4 1 l'estomac 2 5 6 2 les intestins 2 7 8

II. Les fichiers utilisés par "Corps interne"

Ces fichiers sont référencés dans les fichiers de paramètres.

1. liste_exercice

Contient les informations des exercices de la session. Il contient en fait des lignes de désignation.info et/ou de parcours.info

Structure du fichier :

<exercice1> <exercice2>.....<exercice5>

Structure de <exercice i> :

<type_ex> <graphique support> (<fichp1> ou <fichd1>) (<fichp2>) param<type_ex><num cat desig>/<num- cat parc> (<system>)

<type_ex> = caractère "p" ou "d". Type de l'exercice.
<graphique support> = nom du fichier de l'image du graphique support à l'exercice
(<fichp1> ou <fichd1>) = nom du fichier des informations des cadres, informations des parties, des énoncés.
(<fichp2>) = nom du fichier des cadres obligatoires si c'est un exercice de type parcours.
param<type_ex><num cat desig>/<num cat parc> = voir partie I
(<system>) dans le cas d'une désignation, =1 si "par système", =0 sinon.

Remarque :

Tous les éléments décrits ci-dessus sont séparés par au moins un blanc.

Il est créé par "Parametrage".

2. donnees_session

Contient des informations sur la session d'exercices.

Structure du fichier :

<nom de l'utilisateur> <return>
<type de saisie du nom> <return>
<type de renforcement> <return>
<num_ex> <return>
<ex_stoppe> <return>

<nom de l'utilisateur> = string contenant le nom de l'utilisateur qui réalisera les exercices de la session.
<type de saisie du nom> = caractère (k=clavier / c=comparaison)
<type de renforcement> = entier (1 si sonore, 2 si aléatoire, 3 si vocal).
<num_ex> = entier, numéro de l'exercice par lequel débute la session.
<ex_stoppe> =entier (1 si l'exercice a été stoppé, 0 sinon).

Remarques générales

Tout fichier, dont le nom se trouve dans un fichier des fichiers quelconques décrits ci-dessus, doit se trouver sur la disquette BIBLIO correspondant à l'exercice auquel il se rapporte. Ainsi les fichiers suivants doivent s'y trouver.

Pour une désignation

Les fichiers images des parties à désigner.
Le fichier du graphique support.
Les fichiers des sons des parties à désigner.

Pour un parcours

Le fichier du graphique support et du symbole.
Les fichiers sons des parties du parcours et du symbole.

Les fichiers images peuvent être créés par n'importe quel logiciel de dessin tel que DELUXPAINT.

Les fichiers sons ont été réalisés au moyen du logiciel MASTERSOUND.

Les dessins servant de graphique support à un exercice de type désignation représente des silhouettes humaines vides.

ANNEXE 4 : Les "definition modules" et les "implementation modules" du logiciel "Corps interne"

```
DEFINITION MODULE var_globale;

FROM Intuition IMPORT  ScreenPtr,WindowPtr,Image;
FROM Rasters IMPORT RastPortPtr;
FROM Views IMPORT ViewPortPtr;
FROM SoundLib IMPORT ASampPtr;

CONST

 MAX_MESS_VOC = 45;
 MAX_SONS = 51;
 MAX_MESS_TEXT = 50;
 MAX_PARAMETRES = 8;
 MAX_MESS_TRACE = 100; (* taille max du tableau contenant le fichier
                          'trace.text' les messages possibles *)
 MAX_CARTOON = 5;
 LONG_CADRE = 150;
 HAUT_CADRE = 16;

TYPE

 tableau_entier12 = ARRAY [1..12] OF INTEGER;
 tableau_entier40 = ARRAY [1..40] OF INTEGER;
 string20 = ARRAY [1..20] OF CHAR;
 string40 = ARRAY [1..40] OF CHAR;
 string80 = ARRAY [1..80] OF CHAR;
 exercice = RECORD
             type_ex : CHAR;
             graphe_support : string20;
             fichier_ptr1 : string20;
             fichier_ptr2 : string20;
             fichier_param : string20;
             par_nom : INTEGER;
             anim : INTEGER;
           END;
 tableau_exercice = ARRAY [1..20] OF exercice;
 tableau_cadre_parc = ARRAY [1..50] OF tableau_entier12;
 tableau_cadre_obl = ARRAY [1..50] OF tableau_entier12;
 tableau_anim = ARRAY [1..10] OF Image;
 tableau_icone = ARRAY [1..5] OF Image;
 tableau_cartoon = ARRAY [FALSE..TRUE] OF ARRAY [1..MAX_CARTOON] OF Image;
 tableau_mess_text = ARRAY [1..MAX_MESS_TEXT] OF string40;
 tableau_mess_trace =ARRAY [1..MAX_MESS_TRACE] OF string80;
 cadre = RECORD
            x : INTEGER;
            y : INTEGER;
            long : INTEGER;
            haut: INTEGER;
            nom_son : string20;
            brush : Image;
            nom_cadre : string40;
          END;
 tableau_cadre_desig = ARRAY [1..15] OF cadre;
 tableau_son = ARRAY [1..MAX_SONS] OF ASampPtr;
 tableau_param = ARRAY [1..MAX_PARAMETRES] OF INTEGER;
 tableau_nom_cadre_obl = ARRAY [1..50] OF string20;
 point = RECORD

            x : INTEGER;
            y : INTEGER;
          END;


VAR
```

```
scr : ScreenPtr;
rp : RastPortPtr;
vp : ViewPortPtr;
win,w : WindowPtr;

tab_icone : tableau_icone; (* tableau contenant les icones*)
tab_anim : tableau_anim;
tab_cartoon : tableau_cartoon; (* tableau contenant les images d'attente*)
tab_param : tableau_param; (* tableau contenant les paramètres des exercices*)
tab_mess_trace : tableau_mess_trace; (**)
tab_mess_text : tableau_mess_text;
tab_cadre_parc : tableau_cadre_parc;
tab_cadre_obl : tableau_cadre_obl;
tab_cadre_desig : tableau_cadre_desig;
tab_coord_nom : ARRAY[1..11] OF point; (* tableau contenant le point supérieur
                                          gauche des cadres contenant les noms
                                          des parties dans la désignation par
                                          nom *)
tab_cadre_nom : ARRAY[1..11] OF point; (* tableau tab_coord_nom dont les
                                          éléments sont mélangés.
                                          tab_cadre_nom[i] contient le pt sup
                                          gauche du cadre où se trouve
                                          tab_cadre_desig[i].nom *)
graphique_support : Image;
nb_enonce : INTEGER;
liste_enonce : tableau_entier40;
nb_liste_enonce : INTEGER;
tab_exercice : tableau_exercice;
tab_son : tableau_son;
tab_nom_cadre_obl : tableau_nom_cadre_obl;
nb_classe : INTEGER;
tab_couleur : tableau_entier40;
nom_symbole : string20;
symbole : Image;
coul_parc :INTEGER;
fini_ex : BOOLEAN;
type_ex : INTEGER;
sauvegarde_mess : ARRAY [1..4] OF INTEGER;
nb_mess_sauv : INTEGER;
nom_user : string20;
type_saisie : CHAR;
stop : BOOLEAN;
entree:BOOLEAN;
type_renfor : INTEGER;
num_ex : INTEGER;
(*num_classe : INTEGER;
num_cadre : INTEGER;*)
num_anc_cadre : INTEGER;
num_anc_classe : INTEGER;
nb_classe_desig : INTEGER;
nb_cadre_desig : INTEGER;

tick_cartoon,minute : LONGCARD;
num_cartoon : INTEGER;
cartoon : BOOLEAN;
nb_cartoon : INTEGER;
pos_tab_trace:INTEGER;

END var_globale.
(*****************************************************************************)
```

```
DEFINITION MODULE var_globale_trace;

FROM var_globale IMPORT string80;

CONST
 MAX_TRACE = 401; (* taille du tableau des traces d'un exercice *)

VAR
  tab_trace :  ARRAY [1..MAX_TRACE] OF string80;
                (*tableau contenant les traces d'un exercice *)
END var_globale_trace.




DEFINITION  MODULE coordinateur;

PROCEDURE desallocation;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Gère le déroulement d'une session d'exercice.Il initialise
*       l'environnement nécessaire à chaque exercice:
*       charge les différentes données nécessaires (images, sons...).
*       Il reprend un exercice dans l'état où il a été abandonné et
*       fournit les fichiers traces.
*
*****************************************************************************)

PROCEDURE corps_interne;

END coordinateur.
```

```
DEFINITION MODULE designation;

PROCEDURE gestion_desig (VAR stop:BOOLEAN);
(*****************************************************************************
*
* IN : /
*
* OUT: stop est vrai si l'utilisateur quitte l'exercice en cliquant sur
*      "stop".
*           est faux si l'utilisateur quitte l'exercice en cliquant sur
*      "fin".
*
* BUT : Conduite de l'exercice de type désignation.
*
*****************************************************************************)

END designation.




DEFINITION MODULE parcours;

PROCEDURE gestion_parcours(VAR stop:BOOLEAN);
(*****************************************************************************
*
* IN : /
*
* OUT: stop est vrai si l'utilisateur quitte l'exercice en cliquant sur
*      "stop".
*           est faux si l'utilisateur quitte l'exercice en cliquant sur
*      "fin".
*
* BUT : Conduite de l'exercice de type parcours.
*
*****************************************************************************)

END parcours.
```

```
DEFINITION MODULE textuel;

(*****************************************************************************
*
* Ce module regroupe un ensemble de procédures de gestion de l'affichage
* textuel.
*
*****************************************************************************)

FROM var_globale IMPORT string40;

PROCEDURE confirmation
        (mess_texte : string40;mess_voc :INTEGER;VAR confirmation : BOOLEAN);
(*****************************************************************************
*
* IN : mess-texte : message écrit dans la fenêtre.
*      mess_voc : n° du message à dire.
*
* OUT: confirmation : l'utilisateur a confirmé ou non.
*
* BUT : Demande la confirmation d'un stop ou d'un fini. Elle crée une fenêtre
*       avec les gadgets oreille, oui et non.
*
*****************************************************************************)

PROCEDURE met_nom_ds_cadre;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Place les noms des parties à désigner dans les cadres (désignation par
*       nom).
*
*****************************************************************************)

PROCEDURE ecrire_enonce(num_enonce:INTEGER);
(*****************************************************************************
*
* IN : num_enonce : n° du cadre énoncé (demandé).
*
* OUT: /
*
* BUT : Ecrit la question correspondant à l'énoncé du cadre dans le cartouche
*       d'affichage textuel (en bas à droite).
*
*****************************************************************************)

PROCEDURE saisie_nom(VAR abandonne : BOOLEAN);
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Gère la saisie du nom de l'utilisateur. Elle se déroule jusqu'à
*       l'entrée du nom correct ou de l'abandon.
*
*****************************************************************************)

END textuel.
```

```
DEFINITION MODULE base_donnees;

(*****************************************************************************
*
* Ce module regroupe un ensemble de procédures de gestion de
* la base de donnée.
*
*****************************************************************************)

FROM var_globale IMPORT string2Ø;
FROM Intuition IMPORT Image,ScreenPtr;


PROCEDURE lire_param(VAR nom_fich:string2Ø);
(*****************************************************************************
*
* IN : /
*
* OUT: nom_fich : nom du fichier dans lequel se trouvent les paramètres.
*
* BUT : Initialise le tableau tab_param en fonction des
*       paramètres définis à partir du fichier user:nom_fich.
*
*****************************************************************************)

PROCEDURE saisie_exercice (VAR nb_ex:INTEGER);
(*****************************************************************************
*
* IN : /
*
* OUT: nb_ex : nb d'exercices qui constituent la session (taille réelle
*              de tab_exercice).
*
* BUT : Initialise le tableau tab_exercice correspondant aux énoncésen lisant
*       le fichier user:liste_exercice.
*
*****************************************************************************)

PROCEDURE charger_donnees_session;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Initialise les informations concernant la session d'exercice en
*       lisant le fichier user:donnees_session et en plaçant les différents
*       éléments dans nom_user, saisie_nom, renforcement,num_exercice et stop.
*
*****************************************************************************)

PROCEDURE charger_image(VAR image:Image; VAR nom_im:string2Ø);
(*****************************************************************************
*
* IN : /
*
* OUT: image : nom du fichier IFF de l'image.
*      nom_im : nom associé à l'image dans le programme.
*
* BUT : Charge une image (fichier IFF) de nom nom_im et le placer
*       dans image.
*
*****************************************************************************)

PROCEDURE maj_donnees_session(num_exercice:INTEGER; ex_stop : CHAR);
(*****************************************************************************
```

```
*
* IN : num_exercice :n° d'exercice où on est arrivé
*      ex_stop : n° de l'exercice où on a stoppé.
*
* OUT: /
*
* BUT : Mise à jour du fichier donnees_session sur la disquette user.
*
*****************************************************************************)

PROCEDURE init_mess_text;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Initialise le tableau des messages (tab_mess) utilisé pour
*       les transactions textuelles, à partir de message_text.
*
*****************************************************************************)

PROCEDURE init_mess_trace;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Initialise le tableau des traces (tab_mess_trace) contenant
*       les messages utilisés pour réaliser les traces, en utilisant
*       user:message_trace.
*
*****************************************************************************)

PROCEDURE saisie_designation;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Lit le fichier user:tab_enonce[num_enonce].fichier_pointeur1 et place
*       les données dans tab_cadre_desig et tab_couleur.
*
*****************************************************************************)

PROCEDURE saisie_parcours;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Lit le fichier user:tab_enonce[num_enonce].fichier_pointeur1 et place
*       les données dans tab_cadre_parc, lit le fichier
*       user:tab_enonce[num_enonce].fichier_pointeur2 et place
*       les données dans tab_cadre_obl et tab_couleur. et tab_nom_obl.
*
*
*****************************************************************************)

PROCEDURE ecrire_liste_enonce;
(*****************************************************************************
*
* IN : /
```

```
*
* OUT: /
*
* BUT : Copie le fichier liste_enonce et les éléments du tableau liste_enonce.
*
*****************************************************************************)

PROCEDURE lire_liste_enonce;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Lit le fichier liste_enonce pour initialiser la tableau liste_enonce.
*
*****************************************************************************)

PROCEDURE reprise_parc( num_cadre,num_classe,
          nb_cadre_desig,nb_classe_desig:INTEGER);
(*****************************************************************************
*
* IN : num_cadre :n° du cadre désigné avant le stop.
*      num_classe : n° de la classe désignée avant le stop.
*      nb_cadre_desig : nb de cadres bien désignés dans le parcours.
*      nb_classe_desig : nb de classes obligatoires bien désignées
*      dans le parcours.
*
* OUT: /
*
* BUT : Initialise le fichier reprise_parc.
*
*****************************************************************************)

PROCEDURE lire_reprise_parc(VAR  num_cadre,num_classe,
          nb_cadre_desig,nb_classe_desig:INTEGER);
(*****************************************************************************
*
* IN : /
*
* OUT: num_cadre : n° du cadre désigné avant le stop.
*      num_classe : n° de la classe désignée avant le stop.
*      nb_cadre_desig : nb de cadres bien désignés dans le parcours.
*      nb_classe_desig : nb de classes obligatoires bien désignées
*      dans le parcours.
*
* BUT : Lit le fichier reprise_parc.
*
*****************************************************************************)

PROCEDURE cherche_date(VAR jour, mois, annee : INTEGER);
(*****************************************************************************
*
* IN : /
*
* OUT : jour,mois,annee : jour,mois et année où a lieu
*       l'exécution de la session.
*
* BUT : Cherche la date actuelle pour le calcul de la durée mise
*       dans les traces.
*
*****************************************************************************)

PROCEDURE verifiefich (ecran : ScreenPtr;nom : ARRAY OF CHAR) : BOOLEAN;
(*****************************************************************************
*
```

```
* IN : ecran :
*      nom :
*
* OUT: /
*
* BUT : vérifie que le fichier nom existe sur la disquette, sinon une fenêtre
*       apparaît sur l'écran scr.
*
*****************************************************************************)

PROCEDURE lire_cadre_nom;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Lit le fichier user:coord_cadre pour initialiser tab_cadre_nom
*       de taille nb_enonce.
*
*****************************************************************************)

PROCEDURE ecrire_cadre_nom;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Met les infos de tab_cadre_nom dans user:coord_cadre.
*
*****************************************************************************)

PROCEDURE init_icone;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Initialisation de tab_icone avec les icônes fixes.
*
*****************************************************************************)

PROCEDURE init_cartoon;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Initialisation de tab_cartoon avec les dessins d'attentes.
*
*****************************************************************************)

PROCEDURE calcul_duree;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Calcul la durée de l'exercice du début à la fin, ou du début
*       à un stop.
*
*****************************************************************************)
```

```
PROCEDURE init_anim;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Initialise l'animation.
*
*****************************************************************************)

PROCEDURE re_init_tab_trace;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Réinitialise un tableau tab_trace quand on commence un exercice
*       abandonné
*
*****************************************************************************)

PROCEDURE ecrire_fich_trace(ex_stop : BOOLEAN);
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Crée le fichier trace que l'utilisarteur vient de terminer, de nom
*       <utilisateur><n° de version>, à l'aide du fichier tab_trace.
*       L'entête contient: nom utilisateur, date, durée, les paramètres.
*       Le n° de version se trouve dans version.trace.
*
*****************************************************************************)

END base_donnees.
```

```
DEFINITION MODULE graphique;

(*****************************************************************************
*
* Ce module regroupe des "procédures outils" de gestion de l'affichage graphique
*
*
*****************************************************************************)

FROM Intuition IMPORT Image;

FROM var_globale IMPORT string40;

PROCEDURE init_couleur;
(*****************************************************************************
*
* IN : /
```

```
*
*
* OUT: /
*
* BUT : Initialise la palette des couleurs utilisées
*
*
*****************************************************************************)


PROCEDURE deplacement_symbole(num_cadre: INTEGER);
(*****************************************************************************
*
* IN : num_cadre : numéro d'un cadre
*
*
* OUT: /
*
* BUT : Déplace le graphique du symbole du centre du ou des cadres de numéro
*       num_cadre - 1 jusqu'au centre du ou des cadres de numéro num_cadre
*
* REMARQUE : les numéros des cadres correspondent à l'ordre des cadres dans le
*            parcours.
*
*****************************************************************************)

PROCEDURE renforcement_parcours(num_cadre_desig, num_dern_cadre: INTEGER);
(*****************************************************************************
*
* IN : num_cadre_desig : numéro d'un cadre désigné appartenant au parcours
*      num_dern_cadre : numéro du  cadre précédemment désigné appartenant au
*      parcours
*
*
* OUT: /
*
* BUT : Déplace le graphique du symbole du centre du ou des cadres de numéro
*       num__dern_desig jusqu'au centre du ou des cadres de numéro
*       num_cadre_desig
*
* REMARQUE : les numéros des cadres correspondent à l'ordre des cadres dans le
*            parcours.
*
*****************************************************************************)


PROCEDURE aff_image (x, y : INTEGER; image :Image );
(*****************************************************************************
*
* IN : x,y coordonnées d'un point
*      image : image à afficher
*
* OUT: /
*
* BUT : Affiche à l'écran au point de coordonnées (x,y) l'image image
*
*****************************************************************************)


PROCEDURE aff_brush_designation(num_cadre:INTEGER);
(*****************************************************************************
*
* IN : num_cadre : numéro d'un cadre d'un exercice de désignation
*
*
* OUT: /
```

```
*
* BUT : Affiche à l'écran les images se trouvant dans tab_cadre_desig à la
*       position se trouvant dans les champs x et y de ce tableau exepté le
*       cadre de numéro num_cadre si num_cadre est # 99 sinon elle affiche tous
*       les cadres
*
*
*****************************************************************************)


PROCEDURE aff_graphique(centre : BOOLEAN);
(*****************************************************************************
*
* IN : centre indique si l'affichage se fait au centre (centre = TRUE)  ou dans
*      la  partie droite de l'écran (centre = FALSE)
*
*
* OUT: /
*
* BUT : Affiche à l'écran graphe_support au centre ou à droite (selon centre)
*
*****************************************************************************)


PROCEDURE coloration_zone(x,y,long,haut,couleur1,couleur2:INTEGER);
(*****************************************************************************
*
* IN : x,y coordonnées d'un point
*      long,haut : longeur et hauteur d'une zone
*      couleur1 et couleur2 : numéro de couleur
*
* OUT: /
*
* BUT : Colorie le cadre de coordonnée (x,y), de longueur long et de largeur
*       haut. La technique de coloration est la suivante:
*       les pixels du cadre de couleur couleur1 sont coloriés en couleur
*       couleur2 si couleur # 99 sinon sont coloriés en couleur couleur orange
*       ie de numéro 17
*
*****************************************************************************)

PROCEDURE noircit_organe(tout:BOOLEAN);
(*****************************************************************************
*
* IN : tout : est vrai si tous les cadres doivent noircis (tab_cadre_desig).
*             est faux si seuls ceux de tab_cadre_desig moins ceux de
*             liste_enonce doivent l'être.
* OUT: /
*
* BUT : Colorie en noir les cadres voulus (les parties de couleur <> couleur
*       de fond).
*
*****************************************************************************)

PROCEDURE clignoter_zone(num_cadre :INTEGER);
(*****************************************************************************
*
* IN : num_cadre : n° du cadre correctement désigné (= cadre énoncé).
*
* OUT: /
*
* BUT : Faire clignoter la zone correspondant au cadre.Deux manières sont
*       utilisées suivant le cas "par couleur" ou non.
*
*****************************************************************************)
```

```
PROCEDURE effacer_cadre(x,y,haut,long:INTEGER);
(*****************************************************************************
*
* IN : x,y,haut,long : coordonnées du point supérieur gauche,hauteur,longeur.
*
* OUT: /
*
* BUT : Efface le cadre correspondant aux coordonnées.
*
*****************************************************************************)

PROCEDURE animation(VAR string : ARRAY  OF CHAR) ;
(*****************************************************************************
*
* IN : string : nom du système.
*
* OUT: /
*
* BUT : Affiche à l'écran les différents ddessins d'animation correspondant
*       au nom du système.
*
*****************************************************************************)

PROCEDURE feedback_positif(num_zone_desig:INTEGER);
(*****************************************************************************
*
* IN : num_zone_desig : n° de la zone correctement désignée (= zone énoncée).
*
* OUT: /
*
* BUT : si la désignation est correcte:
*       désignation pseudo-passive, exécuter le renforcement_positif
*               (du module son), afficher la brush correspondante.
*       ni pseudo-passif, ni silhouette vide, renforcement_positif.
*       silhouette vide, renforcement_positif, afficher brush, attendre et
*               l'effacer.
*       par système, exécuter l'animation si elle est choisie de
*               tab_cadre_desig[num_zone_desig].nom_cadre (la derière lettre).
*
*****************************************************************************)

PROCEDURE feedback_negatif(num_enonce,num_zone_desig:INTEGER);
(*****************************************************************************
*
* IN : num_enonce : n° de la zone énoncée.
*      num_zone_desig : n° de la zone mal désignée.
*
* OUT: /
*
* BUT : Gère la correction d'une zone mal désignée:
*       désignation par cadre, silhouette, non vide, pas pseudo-passif,
*               faire clignoter la zone désignée puis celle énoncée.
*       par cadre, silhouette vide, pas pseudo-passif, afficher la zne désignée,
*               la faire clignoter, l'effacer et recommencer avec la zone
*               énoncée
*       pseudo-passif par cadre, émettre "tu l'as déja montré".
*
*****************************************************************************)

PROCEDURE dessine_rect(x,y,l,h:INTEGER);
(*****************************************************************************
*
* IN : x,y,l,h : coordonnées du point supérieur gauche, longaur, hauteur
*                du rectangle.
*
* OUT: /
```

```
*
* BUT : Dessine un rectangle correspondant aux coordonnées.
*
*****************************************************************************)

PROCEDURE colorie_partie_parcours(num_cadre_desig,num_dern_cadre : INTEGER);
(*****************************************************************************
*
* IN : num_cadre_desig : n° du cadre du parcours désigné.
*      num_dern_cadre : n° du dernier cadre du parcours désigné avant.
*
* OUT: /
*
* BUT : Colorie les cadres dont les n° sont entre les deux cadres (y compris).
*       Rappelons que deux cadres au plus peuvent avoir le même n°
*
*****************************************************************************)

PROCEDURE animer_parcours;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT :Afficher l'animation du parcours.
*
*****************************************************************************)

PROCEDURE noircit_cadre(tout:BOOLEAN);
(*****************************************************************************
*
* IN : tout est vrai si tous doivent être en noir (tab_cadre_nom).
*           est faux si seulement tab_cadre_nom moins liste_enonce doivent
*           l'être (si l'exercice a été stoppé).
*
* OUT: /
*
* BUT : Colorie les cadres contenant les noms en noir.
*
*****************************************************************************)

PROCEDURE ecrire(VAR mess:ARRAY OF CHAR;num_ligne:INTEGER);
(*****************************************************************************
*
* IN : mess : n° du message dans tab_mess à écrire.
*      num_ligne : n° de la ligne où il faut l'écrire (il y a trois lignes
*      dans le cadre d'écriture).
*
* OUT: /
*
* BUT : Ecrire un message dans le cadre inférieur droit d'affichage textuel.
*
*****************************************************************************)

PROCEDURE coloration_combinee(num_cadre_desig,num_dern_cadre : INTEGER);
(*****************************************************************************
*
* IN : num_cadre_desig : n° du cadre du parcours désigné.
*      num_dern_cadre : n° du dernier cadre du parcours désigné avant.
*
* OUT: /
*
* BUT : Colorie les cadres et déplace le symbole entre les deux cadres.
*
*****************************************************************************)
```

```
PROCEDURE afficher_cartoon(afficher : BOOLEAN);
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Affiche les écrans d'attente lors du chargement de fichiers entre
*       les exercices.
*
*****************************************************************************)

END graphique.




DEFINITION MODULE son;

(* Module gérant l'emploi des sons digitalisés *)

PROCEDURE dire(num_son:INTEGER);

(*****************************************************************************
*
* IN : num_son : le message à énoncer
*
*
* OUT: /
*
* BUT : Enonce le message vocal se trouvant tab_son[num_son]
*
*
*****************************************************************************)


PROCEDURE renforcement_positif;

(*****************************************************************************
*
* IN : /
*
*
* OUT: /
*
* BUT : Produit la mélodie du renforcement positif
*
*
*****************************************************************************)

PROCEDURE renforcement_negatif;

(*****************************************************************************
*
* IN : /
*
*
* OUT: /
*
* BUT : Produit le signal du renforcement négatif
*
*
*****************************************************************************)
```

```
PROCEDURE enonce_sonore(num:INTEGER);

(*****************************************************************************
*
* IN : Numéro de la partie dont on énonce le nom
*
*
* OUT: /
*
* BUT : Enonce le nom de la partie num
*
*
*****************************************************************************)


PROCEDURE copy_son_fixe;

(*****************************************************************************
*
* IN : /
*
*
* OUT: /
*
* BUT : Copie dans tab_son les sons relatifs à la saisie du nom de
*       l'utilisateur, aux messages inter-exercices, et aux feed-back sonores
*
*
*****************************************************************************)

PROCEDURE charger_son_ex(etat: INTEGER);

(*****************************************************************************
*
* IN : etat : indique le type d'exercice proposé précédemment
*
*
* OUT: /
*
* BUT : Elle charge dans tab_sons les messages sonores relatifs à l'exercice de
*       de type état à savoir :
*       etat = 1 -> exercice de parcours
*            = 2 -> exercice de désignation par image différent qu'en pseudo-pa
*                   sif
*            = 3 -> exercice de désignation par nom en pseudo-passif
*            = 4 -> exercice de désignation par image en pseudo passif
*            = 5 -> exercice de désignation par nom pas en pseudo-passif
*            = 6 -> exercice de désignation de systèmes
*
*****************************************************************************)
END son.
```

```
DEFINITION MODULE trt_donnees;

FROM var_globale IMPORT point,string80;

PROCEDURE indice(num:INTEGER;VAR ind:INTEGER);
(*****************************************************************************
*
* IN : num : élément du tableau liste_enonce
*
* OUT: ind : indice du tableau liste_enonce
*
* BUT : Recherche dans le tableau liste_enonce l'élément num et met dans ind
*       l'indice de ce tableau tel que liste_énonce[ind] = num. Si num
*       n'appartient pas au tableau ind = 0.
*****************************************************************************)


PROCEDURE detection_cadre_desig(VAR ok : BOOLEAN; x,y : INTEGER;
                  num_enoncee : INTEGER; VAR num_zone_desig : INTEGER;
                  VAR ind:INTEGER);

(*****************************************************************************
*
* IN : x,y : coordonnées du curseur au moment de la désignation n'étant pas
*            les coordonnées d'un point de couleur de fond (de numéro 0)
*      num_enoncee : numéro de la zone énoncée.
*
* OUT: ok : résultat
*      ind : indice du tableau liste_enonce
*
* BUT : Elle détecte la zone dans laquelle l'utilisateur à cliquer num_zone_
*       desig. Si l'utilisateur a désigné un point n'appartenant aucune zone ie
*       que x,y sont les coordonnées d'un point n'appartenant à aucun cadre
*       connu ie se trouvant dans tab_cadre_desig alors num_zone_desig = 0
*       ok est vrai si la zone désignée est bien celle demandé ie appartient à
*       tab_cadre_desig[num_enonce
*       ind = l'indice du tableau liste_enonce si num_zone_desig appartient au
*       tableau et tab_param[1] # 3 (pas pseudo pasif)
*       ind = 0 si num_zone_desig n'appartient pas à liste_énonce
*       ind = 1 si tab_param[1] #= 3 (pseudo-passif)
*
*****************************************************************************)

PROCEDURE init_liste_enonce;

(*****************************************************************************
*
* IN : /
*
*
* OUT: /
*
* BUT : Initialise le tableau liste_enonce contenant les numéros des parties à
*       demander lors de l'exercice de type désignation
*
*****************************************************************************)


PROCEDURE detection_couleur(VAR ok: BOOLEAN;num_couleur,num_enonce:INTEGER;
                            VAR num_zone_desig :INTEGER);
(*****************************************************************************
*
* IN : num_couleur : numéro de la couleur désigné par l'utilisateur. Elle est
*                    différente de la couleur de fond ie couleur numéro 0
*      num_enonce : numéro de l'indice de tab_couleur contenant la zone demandée
*
```

```
 * OUT: ok : résultat
 *      num_zone_desig : numéro de la zone désignée
 *
 * BUT : ok est vrai si la zone désignée est celle demandée ie si la couleur
 *       de la zone désignée num_couleur est celle se trouvant dans
 *       tab_couleur[num_enonce].
 *       ok est faux si la zone désignée n'existe pas ou n'est pas celle
 *       demandée
 *       num_zone_desig contient le numéro de la zone désignée si elle exixte
 *       Ø sinon.
 *
 ********************************************************************************)

PROCEDURE calcul_centre_cadre(num_cadre:INTEGER;VAR centre1,centre2:point);

(********************************************************************************
 *
 * IN : num_cadre : numéro de cadre appartenant au parcours
 *
 *
 * OUT: centre1,centre2 : coordonnées (x,y) de cadre
 *
 * BUT : Calcule le centre du ou des cadres de numéro num_cadre. Elle place les
 *       coordonnées de ces centres dans centre1 et centre2. S'il n'existe qu'un
 *       cadre de numéro num_cadre alors centre1.x = centre2.x et centre1.y
 *       = centre2.y
 *
 *
 ********************************************************************************)


PROCEDURE maj_liste_enonce(ok:BOOLEAN;VAR fin_liste:BOOLEAN;indice:INTEGER);

(********************************************************************************
 *
 * IN : ok  est faux si l'utilisateur a désigné une partie déja désigné
 *          préalablement et si tab_param[1] = 3 ie pseudo_passif
 *      ok est faux sinon
 *      indice est l'indice dans le tableau liste_exercice de la zone désignée
 *
 * OUT: fin_liste indique si la fin de la liste est atteinte
 *
 * BUT : Met-à-jour le tableau liste_enonce contenant les numéros des parties à
 *       demander lors de l'exercice de type désignation
 *
 ********************************************************************************)
PROCEDURE init_tab_son_nil;
(********************************************************************************
 *
 * IN : /
 *
 *
 * OUT: /
 *
 * BUT : Initialise le tab_son avec la valeur NIL
 *
 *
 ********************************************************************************)


PROCEDURE detection_cadre_parc(VAR num_cadre,num_classe:INTEGER;
                               x,y:INTEGER);
(********************************************************************************
 *
 * IN : x,y : coordonnées du point désigné par l'utilisateur de couleur
 *      différente de la couleur Ø (de fond)
```

```
*
*
* OUT: num_cadre : numéro de cadre
*      num_classe : numéro de la classe
*
* BUT : Elle détecte le numéro de cadre et de classe auquel appartient le point
*       (x,y). Elle met dans num_cadre ce numéro de cadre s'il existe (ie
*       appartenant à tab_cadre_parc ou Ø si le cadre n'appartient pas à tab_
*       cadre_parc. Elle met dans num_classe le numéro de la classe à laquelle
*       appartient num_cadre ou Ø si num_cadre est = Ø ou que num_cadre # Ø et
*       n'appartient pas à un classe.
*
* REMARQUE : tab_cadre_obl est trié par ordre croissant si l'on ignore les Ø.
*            Le tableau tab_cadre_parc contient les cadres dans l'ordre du
*            parcours
*
*****************************************************************************)


PROCEDURE choix_cartoon;

(*****************************************************************************
*
* IN : /
*
*
* OUT: /
*
* BUT: Choisit au hasard un numéro de cartoon (num_cartoon) entre 1 et
*      nb_cartoon
*
*
*****************************************************************************)

PROCEDURE init_tab_coord_nom;

(*****************************************************************************
*
* IN : /
*
*
* OUT: /
*
* BUT : Initialise le tableau tab_coord_nom
*
*
*****************************************************************************)

PROCEDURE chercher (j,taille_liste : INTEGER;VAR trouver : BOOLEAN);

(*****************************************************************************
*
* IN : j : nombre à rechercher
*      taille_liste : taille réelle du tableau liste
*
* OUT: trouver : résultat
*
* BUT : Recherche dans le tableau liste défini dans init_tab_cadre_noms
*       l'élément j s'y trouve. Si c'est le cas trouver est mis à TRUE,
*        sinon trouver est mis à FALSE
*
*****************************************************************************)

PROCEDURE init_tab_cadre_nom;
```

```
(*****************************************************************************
*
* IN : /
*
*
* OUT: /
*
* BUT : Initialise le tableau tab_cadre_nom
*
*
*****************************************************************************)

PROCEDURE pluriel(VAR nom : ARRAY OF CHAR) : BOOLEAN;


(*****************************************************************************
*
* IN : nom : nom du nom du cadre
*
*
* OUT: /
*
* BUT : pluriel est vrai si le nom est au pluriel
*               est faux sinon
*
*****************************************************************************)

PROCEDURE ecrire_trace (mess : string80);

(*****************************************************************************
*
* IN : mees : contenu de la trace
*
*
* OUT: /
*
* BUT : Met dans le tableau tab_trace à la position pos_tab_trace
*       mess si pos_tab_trace est < MAX_TRACE-2 et augmente pos_tab_trace de 1
*
*
*
*****************************************************************************)

PROCEDURE verifie_parcours( num_cadre,num_anc_cadre,num_classe,
                            num_anc_classe:INTEGER;VAR res:INTEGER);

(*****************************************************************************
*
* IN : num_cadre contient le numéro du cadre du parcours désigné par l'utilisa-
*      teur
*      num_anc_cadre contient le numéro du cadre désigné préalablement
*      num_classe contient le numéro de la classe à laquelle appartient
*      num_cadre ou 0 si num_cadre appartient à aucune classe
*      num_anc_cadre est le numéro de la classe à laquelle appartient num_anc_
*       cadre
*
*
* OUT: res : résultat
*
* BUT : Vérifie si le cadre num_cadre désigné par l'utilisateur est correct
*       res = 1 si num_cadre est conforme au parcours
*       res = 2 si l'utilisateur à désigner un qu'il avait déjà désigné
*               auparavant
*       res = 3 si l'utilisateur a oublié un ou plusieurs cadres obligatoires
*               ie se trouvant dans tab_cadre_obl
*       res = 4 si l'utilisateur à cliqué sun un pixel différent de la couleur
```

```
*                de fond (numéro Ø) n'appartennt à aucun cadre connu.
*
*
**************************************************************************)

END trt_donnees.
```

```
IMPLEMENTATION MODULE var_globale;
BEGIN
END var_globale.
```

```
IMPLEMENTATION MODULE var_globale_trace;
BEGIN
END var_globale_trace.
```

```
MODULE corpsint;

FROM coordinateur IMPORT corps_interne;

BEGIN
     corps_interne
END.
```

```
IMPLEMENTATION MODULE coordinateur;

FROM var_globale IMPORT num_ex,tab_exercice,tab_icone,tab_mess_text,
             graphique_support,tab_param,tab_cadre_desig,tab_cadre_nom,
             tab_coord_nom,tab_son,MAX_MESS_TRACE,MAX_SONS,
             nb_enonce,string2Ø,w,rp,scr,pos_tab_trace,
             type_renfor,minute,symbole,tick_cartoon,num_anc_classe,
             num_anc_cadre,nb_classe_desig,nb_cadre_desig,stop,entree,
             sauvegarde_mess,nb_mess_sauv;
FROM var_globale_trace IMPORT tab_trace;
FROM textuel IMPORT saisie_nom,confirmation;
FROM base_donnees IMPORT charger_image,saisie_exercice,
                  lire_param,saisie_designation,ecrire_liste_enonce,
                  maj_donnees_session,charger_donnees_session,ecrire_cadre_nom,
                  init_icone,init_cartoon,init_anim,init_mess_text,
                  init_mess_trace,saisie_parcours,reprise_parc,
                  ecrire_fich_trace,re_init_tab_trace;
FROM graphique IMPORT init_couleur,effacer_cadre,afficher_cartoon;
FROM designation IMPORT gestion_desig;
FROM son IMPORT copy_son_fixe,charger_son_ex;
FROM trt_donnees IMPORT choix_cartoon;
FROM parcours IMPORT gestion_parcours;

FROM RemAlloc IMPORT RemFree;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Intuition IMPORT CloseScreen,WindowFlags,WindowFlagsSet,IDCMPFlags,
     IDCMPFlagsSet,CloseWindow,DrawImage,WindowPtr;
FROM AmigaDos IMPORT DateStampRecord,DateStamp;
FROM Memory IMPORT FreeMem;
FROM RandomNumbers IMPORT Seed;
FROM SoundLib IMPORT SBase,SoundName,ResetChan,FreeASamp;
FROM Libraries IMPORT OpenLibrary,CloseLibrary;
FROM Strings IMPORT CopyString;
FROM SYSTEM IMPORT ADR;


CONST WIDCMP=IDCMPFlagsSet{};
      WFlags = WindowFlagsSet{Activate,Borderless};

VAR fin_ex,abandon,confirm,meme_ex:BOOLEAN;
    i,etat_courant,etat_pred,nb_exercice:INTEGER;
    date:DateStampRecord;


(*****************************************************************************)

PROCEDURE desallocation;

BEGIN
      IF (tab_exercice[num_ex].type_ex = "D") OR
         (tab_exercice[num_ex].type_ex = "d")
      THEN i := 1;
           WHILE  (i <= nb_enonce) DO
             RemFree(tab_cadre_desig[i].brush.ImageData);
             i:= i+ 1;
           END;
           RemFree(graphique_support.ImageData);
      ELSE RemFree(symbole.ImageData);
           RemFree(graphique_support.ImageData);
      END;
      i:= 2Ø;
      WHILE (i <= MAX_SONS) DO
        IF tab_son[i] # NIL
        THEN FreeMem(tab_son[i]^.Samp,tab_son[i]^.Size);
```

```
            tab_son[i]:= NIL;
          END;
          i:=i+i;
        END;
END desallocation;

(*****************************************************************************)

PROCEDURE corps_interne;

BEGIN

     (* initialisation generales *)

     init_anim;
     init_cartoon;
     init_icone;
     scr:= CreateScreen(320,256,5,NIL);
     w:= CreateWindow(0,0,320,256,WIDCMP,WFlags,NIL,scr,NIL);
     rp:= w^.RPort;
     init_couleur;
     (* ecran de presentation *)
     DrawImage(rp^,tab_icone[4],0,0);
     DateStamp(date);
     minute:= date.dsMinute;
     Seed:= minute;
     tick_cartoon:= (minute * LONGCARD(3000))+ LONGCARD(date.dsTick);
     (* ouverture de la librairie son *)
     SBase:= OpenLibrary(ADR(SoundName),LONGCARD(0));
     i:= 1;
     WHILE i<= MAX_SONS DO
       tab_son[i]:= NIL;
       i:= i+1;
     END;
     choix_cartoon;
     init_mess_text;
     init_mess_trace;
     meme_ex:= FALSE;
     fin_ex := FALSE;
     entree := TRUE;
     pos_tab_trace := 1;
     saisie_exercice(nb_exercice);
     charger_donnees_session;
     IF stop THEN re_init_tab_trace;END;
     IF nb_exercice >= num_ex
     THEN copy_son_fixe;
          effacer_cadre(0,0,256,320);
          abandon := FALSE;
          saisie_nom(abandon);
          IF NOT abandon
          THEN
            choix_cartoon;
            afficher_cartoon(TRUE);
            FreeMem(tab_son[1]^.Samp,tab_son[1]^.Size);
            tab_son[1]:= NIL;
            FreeMem(tab_son[2]^.Samp,tab_son[2]^.Size);
            tab_son[2]:= NIL;
            FreeMem(tab_son[3]^.Samp,tab_son[3]^.Size);
            tab_son[3]:= NIL;
            FreeMem(tab_son[4]^.Samp,tab_son[4]^.Size);
            tab_son[4]:= NIL;


(* BOUCLE PRINCIPALE *)

            WHILE (nb_exercice >= num_ex) AND ( NOT fin_ex) DO
```

```
IF (NOT entree)
THEN
   i:= 1;
   WHILE (i <= MAX_MESS_TRACE) DO
     CopyString(tab_trace[i],"");
     i:= i+1;
   END;
   pos_tab_trace := 1;

ELSE entree := FALSE;
END;
IF (NOT meme_ex)
THEN
   lire_param(tab_exercice[num_ex].fichier_param);
   charger_image(graphique_support,
                 tab_exercice[num_ex].graphe_support);
   IF (tab_exercice[num_ex].type_ex = "D") OR
      (tab_exercice[num_ex].type_ex = "d")
   THEN

      IF tab_param[8] = 1
      THEN
         etat_courant := 5
      ELSE
         IF tab_param[1] = 3
         THEN
           IF tab_param[8] = 1
           THEN etat_courant:= 3
           ELSE etat_courant:= 4
           END;
         ELSE
           IF tab_exercice[num_ex].anim = Ø
           THEN etat_courant:= 2
           ELSE etat_courant:= 6
           END;
         END;
      END;

   END;

   IF (etat_courant # etat_pred)
   THEN charger_son_ex(etat_courant);
        etat_pred := etat_courant;
   END;

   IF (tab_exercice[num_ex].type_ex = "D") OR
      (tab_exercice[num_ex].type_ex = "d")
   THEN saisie_designation
   ELSE saisie_parcours
   END;

END;
DateStamp(date);
minute:= date.dsMinute;
CloseWindow(w^);
IF (tab_exercice[num_ex].type_ex = "D") OR
   (tab_exercice[num_ex].type_ex = "d")
THEN gestion_desig(stop);
ELSE gestion_parcours(stop);
END;

w := CreateWindow(Ø,Ø,32Ø,256,WIDCMP,WFlags,NIL,scr,NIL);
rp := w^.RPort;
ecrire_fich_trace(stop);
IF stop
THEN effacer_cadre(Ø,Ø,256,32Ø);
```

```
                    choix_cartoon;
                    afficher_cartoon(TRUE);
                    fin_ex := TRUE;
                    maj_donnees_session(num_ex,"1");
                    IF (tab_exercice[num_ex].type_ex ="D") OR
                       (tab_exercice[num_ex].type_ex = "d")
                    THEN ecrire_liste_enonce;
                    ELSE reprise_parc(num_anc_cadre,num_anc_classe,
                         nb_cadre_desig,nb_classe_desig)
                    END;

               ELSE
                    rp := w^.RPort;
                    effacer_cadre(Ø,Ø,256,32Ø);
                    sauvegarde_mess[1]:=6;
                    confirmation(tab_mess_text[6],6,confirm);
                    choix_cartoon;
                    IF (NOT confirm)
                    THEN desallocation;
                         meme_ex := FALSE;
                         rp := w^.RPort;
                         effacer_cadre(Ø,Ø,256,32Ø);
                         IF num_ex # nb_exercice
                         THEN
                            confirmation(tab_mess_text[5],5,confirm);
                            IF (NOT confirm)
                            THEN fin_ex := TRUE;
                                 maj_donnees_session(num_ex+1,"Ø");
                            ELSE choix_cartoon;
                                 num_ex := num_ex +1;
                            END;
                         ELSE num_ex := num_ex + 1;
                         END;
                    ELSE meme_ex := TRUE;
                    END;
               END;

            END; (* END du WHILE principal *)


          CloseWindow(w^);
          CloseScreen(scr^);

          ELSE (* l'utilisateur a abandonné la session *)
            CloseWindow(w^);
            CloseScreen(scr^);
          END;
     ELSE (* tous les exercices de la session sont termines *)
          CloseWindow(w^);
          CloseScreen(scr^);

     END;

     (* liberation des canaux sonores et fermeture de la librairie *)
     i:= 1;
     WHILE i <= 19 DO
       IF tab_son[i] # NIL
       THEN FreeMem(tab_son[i]^.Samp,tab_son[i]^.Size);
       END;
       i := i + 1;
     END;
     ResetChan;
     CloseLibrary(SBase^);


END corps_interne;
```

```
END coordinateur.
```

```
IMPLEMENTATION MODULE designation;

FROM var_globale IMPORT tab_param,nb_enonce,num_ex,tab_icone,win,rp,vp,scr,
     graphique_support,liste_enonce,tab_cadre_nom,LONG_CADRE,HAUT_CADRE,
     tab_exercice,nb_mess_sauv,sauvegarde_mess,tab_mess_text,tab_mess_trace,
     tab_cadre_desig,string80;
FROM graphique IMPORT dessine_rect,aff_brush_designation,aff_graphique,
     noircit_organe,noircit_cadre,feedback_positif,feedback_negatif,
     effacer_cadre,clignoter_zone,ecrire,coloration_zone;
FROM trt_donnees IMPORT init_liste_enonce,detection_couleur,
     detection_cadre_desig,maj_liste_enonce,init_tab_cadre_nom,
     init_tab_coord_nom,ecrire_trace;
FROM son IMPORT dire,renforcement_negatif,enonce_sonore;
FROM textuel IMPORT confirmation,met_nom_ds_cadre,ecrire_enonce;
FROM base_donnees IMPORT lire_liste_enonce,lire_cadre_nom;

FROM Intuition IMPORT Window,WindowPtr,WindowFlags,WindowFlagsSet,IDCMPFlags,
     IntuiMessagePtr,CloseWindow,GadgetPtr,Gadget,IDCMPFlagsSet,DisplayBeep,
     ModifyIDCMP,DrawImage;
FROM SimpleGadgets IMPORT BeginGadgetList,EndGadgetList,FreeGadgetList,
     AddGadgetImageButton;
FROM Ports IMPORT GetMsg;
FROM SimpleIDCMP IMPORT MsgData,WindowProc,ProcIMsg;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet,Wait;
FROM Drawing IMPORT Draw,Move,ReadPixel,SetAPen;
FROM SYSTEM IMPORT ADR;
FROM Strings IMPORT CopyString,ConcatString,StringLength;
FROM Conversions IMPORT ConvNumberToString;
FROM AmigaDos IMPORT Execute;

CONST WIDCMP = IDCMPFlagsSet{GadgetUp,MouseButtons};
      WFlags = WindowFlagsSet{Activate,Borderless};



VAR gliste:GadgetPtr;
    wp: WindowProc;
    sig:SignalSet;
    msg:IntuiMessagePtr;
    curx,cury,i,couleur,num_zone_desig,indice,num_mess_texte,num_son:INTEGER;
    fini,confirme,bloque_enonce,ok_clic,ok,fin_liste,arret:BOOLEAN;
    trace:string80;

(*****************************************************************************)

PROCEDURE clic_gauche(VAR w:Window;VAR message:MsgData;clic:CARDINAL);

BEGIN
  IF ok_clic
  THEN curx := w.MouseX;
       cury := w.MouseY;
       couleur := ReadPixel(rp^,curx,cury);
       IF (couleur#0) AND
          (curx>0) AND (curx<319) AND (cury>0) AND (cury<220) AND
          ((couleur#17) OR (tab_param[5]=1))
       THEN ModifyIDCMP(win^,IDCMPFlagsSet{});
            bloque_enonce := FALSE;
            IF tab_param[7]=1
            THEN detection_couleur(ok,couleur,liste_enonce[1],num_zone_desig);
            ELSE detection_cadre_desig(ok,curx,cury,liste_enonce[1],
                                       num_zone_desig,indice);
            END;
            IF (tab_param[1]=1) OR (tab_param[1]=3)
```

```
                (*apprentissage ou pp *)
            THEN IF ok
                 THEN feedback_positif(num_zone_desig);
                      maj_liste_enonce(ok,fin_liste,indice);
                 ELSE IF num_zone_desig #Ø
                      THEN feedback_negatif(liste_enonce[1],num_zone_desig);
                      ELSE renforcement_negatif;
                           bloque_enonce := TRUE;

                           (* TRACE *)
                           ecrire_trace(tab_mess_trace[13]);
                           (* FIN TRACE *)
                      END;
                 END;
            ELSE  (* evaluation *)
                 IF num_zone_desig # Ø
                 THEN
                      IF tab_param[8]=1  (* par nom *)
                      THEN coloration_zone(tab_cadre_nom[num_zone_desig].x+1,
                             tab_cadre_nom[num_zone_desig].y+1,
                             LONG_CADRE-1,HAUT_CADRE-1,1Ø,2);
                                                  (* bleu en rouge *)
                      END;
                      clignoter_zone(num_zone_desig);
                      IF tab_param[8]=1  (* par nom *)
                      THEN coloration_zone(tab_cadre_nom[num_zone_desig].x+1,
                             tab_cadre_nom[num_zone_desig].y+1,
                             LONG_CADRE-1,HAUT_CADRE-1,2,1Ø);
                                                  (* rouge en bleu *)
                      END;
                      maj_liste_enonce(ok,fin_liste,indice);
                      effacer_cadre(166,223,27,147);

                      (* TRACE *)
                      CopyString(trace,tab_mess_trace[14]);
                      ConcatString(trace,
                                   tab_cadre_desig[num_zone_desig].nom_cadre);
                      ecrire_trace(trace);
                      (* FIN TRACE *)

                 ELSE (* TRACE *)
                      ecrire_trace(tab_mess_trace[13]);
                      (* FIN TRACE *)
                 END;
            END;
            ModifyIDCMP(win^,IDCMPFlagsSet{GadgetUp,MouseButtons});
            ok_clic := FALSE;
       ELSE bloque_enonce := TRUE;
       END;
  ELSE ok_clic := TRUE;
       bloque_enonce := TRUE;
  END;
END clic_gauche;
(*****************************************************************************)

PROCEDURE gestion_gadget(VAR w: Window;VAR msg:MsgData;VAR gad:Gadget);

VAR
   j: INTEGER;

BEGIN
  CASE gad.GadgetID OF
     Ø: bloque_enonce := TRUE;
        fini := TRUE;

        (* TRACE *)
```

```
        ecrire_trace(tab_mess_trace[16]);
        (* FIN TRACE *) ¦

     1: bloque_enonce := TRUE;
        IF tab_param[3]=1  (* enonce vocal *)
        THEN dire(9);
             j := 1;
             WHILE j <= nb_mess_sauv DO
               dire(sauvegarde_mess[j]);
               j:= j+1;
             END;
        END;
        IF tab_param[8] = 1
        THEN
            clignoter_zone(liste_enonce[1]);
        END;

        (* TRACE *)
        ecrire_trace(tab_mess_trace[3]);
        (* FIN TRACE *) ¦

     2: bloque_enonce := TRUE;
        arret := TRUE;

        (* TRACE *)
        ecrire_trace(tab_mess_trace[8]);
        (* FIN TRACE *)    ¦


     END;
END gestion_gadget;

(*****************************************************************************)

PROCEDURE gestion_desig (VAR stop:BOOLEAN);

VAR temp:ARRAY[1..3] OF INTEGER;
    j : INTEGER;
    boo : BOOLEAN;

BEGIN
  WITH wp DO
    procGadgetUp := gestion_gadget;
    procMouseButtons := clic_gauche;
  END;
  IF tab_param[1] <> 4
  THEN BeginGadgetList();
       AddGadgetImageButton(4,222,tab_icone[1]);  (* fini *)
       AddGadgetImageButton(56,222,tab_icone[2]);  (* oreille *)
       AddGadgetImageButton(108,222,tab_icone[3]);  (*stop *)
       gliste := EndGadgetList();
  ELSE gliste := NIL;
  END;
  win := CreateWindow(0,0,320,256,WIDCMP,WFlags,gliste,scr,NIL);
  rp := win^.RPort;
  vp := ADR(scr^.ViewPort);
          (* fait le cadre de l'ecran *)
  dessine_rect(0,0,319,220);
          (* cadre du texte avec ombrage *)
  dessine_rect(165,222,150,31);
  dessine_rect(0,0,319,220);
  dessine_rect(165,222,150,31);
  Move(rp^,169,254);
  Draw(rp^,317,254);
  Move(rp^,169,255);
  Draw(rp^,317,255);
```

```
Move(rp^,316,226);
Draw(rp^,316,253);
Move(rp^,317,226);
Draw(rp^,317,253);


IF stop = FALSE    (* debut d'un exercice *)

THEN
  IF tab_param[8] =2 (* pas par nom *)
  THEN
      IF tab_exercice [num_ex].anim = 1
      THEN graphique_support.TopEdge:=Ø;
           graphique_support.LeftEdge:=Ø;
      ELSE graphique_support.TopEdge:=Ø;
           graphique_support.LeftEdge:=Ø;
           aff_graphique(TRUE);
      END;
  ELSE graphique_support.TopEdge:=Ø;
       graphique_support.LeftEdge:=Ø;
       aff_graphique(FALSE);
  END;
  IF tab_param[5]=2     (* silhouette non vide *)
  THEN aff_brush_designation(99);
  END;
  IF tab_param[8]=1
  THEN (* PAR NOM *)
    init_tab_coord_nom;
    init_tab_cadre_nom;
    met_nom_ds_cadre;
  END;
  IF tab_param[1]=3  (* pp *)
  THEN
     IF tab_param[8] =2  (* PAS PAR NOM *)
     THEN noircit_organe(TRUE);
     ELSE noircit_cadre(TRUE);
     END;
     i:= 1;
     WHILE i<= nb_enonce DO
       liste_enonce[i] := i;
       i:= i+1;
     END;
     liste_enonce[i] := 999;
  ELSE init_liste_enonce;
  END;


ELSE (* DEBUT D'UN EXERCICE STOPPE *)
     lire_liste_enonce;
     boo := Execute(ADR("init_ex"),NIL,NIL);
     IF tab_param[8] =2 (* PAS PAR NOM *)
     THEN
         IF tab_exercice [num_ex].anim = 1
         THEN graphique_support.TopEdge:=Ø;
              graphique_support.LeftEdge:=Ø;
         ELSE graphique_support.TopEdge:=Ø;
              graphique_support.LeftEdge:=Ø;
              aff_graphique(TRUE);
         END;
     ELSE graphique_support.TopEdge:=Ø;
          graphique_support.LeftEdge:=Ø;
          aff_graphique(FALSE);
          lire_cadre_nom;
     END;
     IF tab_param[5]=2  (* silhouette non vide *)
     THEN IF tab_param[1]# 3  (* apprent eval ou passif *)
```

```
          THEN aff_brush_designation(99);
               IF tab_param[8]=1
               THEN met_nom_ds_cadre;
               END;

          ELSE aff_brush_designation(99);
               IF tab_param[8]=2  (* PAS PAR NOM *)
               THEN noircit_organe(FALSE);
               ELSE met_nom_ds_cadre;
                    noircit_cadre(FALSE);
               END;
          END;
     END;
END;

IF tab_param[1]<>4
THEN (* APPRENTISSAGE OU EVALUATION OU PP *)

     ok_clic := TRUE;
     fini := FALSE;
     arret := FALSE;
     bloque_enonce := FALSE;
     confirme := FALSE;
     fin_liste:= FALSE;
     WHILE (NOT confirme) AND (NOT fin_liste) DO
       IF NOT bloque_enonce
       THEN ModifyIDCMP(win^,IDCMPFlagsSet{GadgetUp});
            effacer_cadre(166,223,29,148);  (* interieur cadre *)
            ecrire_enonce(liste_enonce[1]);  (* on pose la question *)
            (*ecrire_trace;*)
            IF tab_param[3] = 1
            THEN enonce_sonore(liste_enonce[1]);
            END;

            ModifyIDCMP(win^,IDCMPFlagsSet{GadgetUp,MouseButtons});
       END;
       sig :=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
       LOOP
          msg:=GetMsg(win^.UserPort^);
          IF (msg=NIL) THEN EXIT; END;
          bloque_enonce:=FALSE;
          ProcIMsg(wp,msg);
       END;
       IF (fini OR arret)
       THEN j := 1;
            WHILE j <= nb_mess_sauv DO
              temp[j]:=sauvegarde_mess[j];
              j:= j+1;
            END;
            IF fini
            THEN num_mess_texte := 7;
                 nb_mess_sauv := 1;
                 sauvegarde_mess[1]:= 7;
            ELSE num_mess_texte := 8;
                 nb_mess_sauv := 1;
                 sauvegarde_mess[1]:= 8;

            END;

            confirmation(tab_mess_text[num_mess_texte],
                         sauvegarde_mess[1],confirme);
            IF (NOT confirme)
            THEN  fini:=FALSE;
                  arret:=FALSE;
                  j := 1;
                  WHILE j <= nb_mess_sauv DO
```

```
                    sauvegarde_mess[j]:= temp[j];
                    j:= j+1;
                  END;
              END;
          END;
        END;

  ELSE   (*** interaction de type PASSIF ***)
      ModifyIDCMP(win^,IDCMPFlagsSet{});
      fin_liste:=FALSE;
      WHILE (NOT fin_liste) DO
        IF tab_param[8] = 2    (* pas par nom *)
        THEN effacer_cadre(166,223,29,148);  (* interieur cadre *)
             ecrire(tab_cadre_desig[liste_enonce[1]].nom_cadre,2);
             IF tab_param[3]=1
             THEN dire(29 + liste_enonce[1]);
             END;
             clignoter_zone(liste_enonce[1]);
             maj_liste_enonce(FALSE,fin_liste,Ø);

        ELSE (* par nom *)
             effacer_cadre(166,223,29,148);  (* interieur cadre *)
             ecrire(tab_cadre_desig[liste_enonce[1]].nom_cadre,2);
             IF tab_param[3]=1
             THEN dire(29 + liste_enonce[1]);
             END;
             coloration_zone(tab_cadre_nom[liste_enonce[1]].x+1,
                             tab_cadre_nom[liste_enonce[1]].y+1,
                             LONG_CADRE-1,HAUT_CADRE-1,1Ø,2);
                                                  (* bleu en rouge *)
             clignoter_zone(liste_enonce[1]);
             maj_liste_enonce(FALSE,fin_liste,Ø);
             effacer_cadre(166,223,29,148);  (* interieur cadre *)
             coloration_zone(tab_cadre_nom[liste_enonce[1]].x+1,
                             tab_cadre_nom[liste_enonce[1]].y+1,
                             LONG_CADRE-1,HAUT_CADRE-1,2,1Ø);
                                                  (* rouge en bleu *)
        END;
      END;
      arret := FALSE;
  END;
  CloseWindow(win^);
  IF tab_param[1] # 4
  THEN FreeGadgetList(gliste^);
  END;
  stop := arret;

END gestion_desig;


END designation.
```

```
IMPLEMENTATION MODULE parcours;

FROM var_globale IMPORT nb_enonce,tab_param,symbole,nb_classe,
     graphique_support,coul_parc,tab_cadre_parc,tab_cadre_obl,tab_mess_trace,
     rp,win,vp,scr,tab_mess_text,tab_nom_cadre_obl,nb_mess_sauv,num_anc_cadre,
     num_anc_classe,nb_cadre_desig,nb_classe_desig,sauvegarde_mess,tab_icone,
     point,tableau_entier12,nom_symbole,string80;
FROM graphique IMPORT coloration_combinee,aff_graphique,deplacement_symbole,
     effacer_cadre,ecrire,renforcement_parcours,colorie_partie_parcours,
     dessine_rect,aff_image,coloration_zone;
FROM trt_donnees IMPORT detection_cadre_parc,verifie_parcours,
     calcul_centre_cadre,ecrire_trace;
FROM son IMPORT renforcement_positif,renforcement_negatif,dire;
FROM base_donnees IMPORT lire_reprise_parc;
FROM textuel IMPORT confirmation;

FROM Intuition IMPORT Window,WindowPtr,WindowFlags,WindowFlagsSet,IDCMPFlags,
     IntuiMessagePtr,CloseWindow,GadgetPtr,Gadget,IDCMPFlagsSet,DisplayBeep,
     ModifyIDCMP,DrawImage;
FROM SimpleGadgets IMPORT BeginGadgetList,EndGadgetList,FreeGadgetList,
     AddGadgetImageButton;
FROM Ports IMPORT GetMsg;
FROM SimpleIDCMP IMPORT MsgData,WindowProc,ProcIMsg;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet,Wait;
FROM Drawing IMPORT Draw,Move,ReadPixel,SetAPen;
FROM SYSTEM IMPORT ADR;
FROM Strings IMPORT CopyString,ConcatString,StringLength,
    LocateSubString,OverwriteWithSubString,DeleteSubString;
FROM AmigaDosProcess IMPORT Delay;
FROM InOut IMPORT WriteInt,Write;
FROM AmigaDos IMPORT Execute;
FROM Conversions IMPORT ConvNumberToString;

CONST WIDCMP = IDCMPFlagsSet{GadgetUp,MouseButtons};
      WFlags = WindowFlagsSet{Activate,Borderless};

VAR
    num_cadre,num_classe,j,res,x,y,l,h,couleur,num_mess_texte: INTEGER;
    confirme,fin_parcours,arret,fini,ok_clic,reinit:BOOLEAN;
    curx,cury:INTEGER;
    wp:WindowProc;
    sig:SignalSet;
    glist:GadgetPtr;
    msg:IntuiMessagePtr;
    c1,c2:point;
    temp:tableau_entier12;
    trace:string80;

(*****************************************************************************)

PROCEDURE ecrit_entree;

VAR tampon,tampon1,tampon2:string80;
   pos:INTEGER;

BEGIN
   CopyString(tampon,tab_nom_cadre_obl[1]);
   pos := LocateSubString(tampon," et ",1,StringLength(tampon));
   IF pos # -1
   THEN
     OverwriteWithSubString(tampon," ou ",pos);
     IF StringLength(tampon) > 18
     THEN CopyString(tampon1,tampon);
          CopyString(tampon2,tampon);
          DeleteSubString(tampon1,pos,StringLength(tampon1));
```

```
            DeleteSubString(tampon2,Ø,pos);
            ecrire(tampon1,2);
            ecrire(tampon2,3);
        ELSE ecrire(tampon,2);
        END;
    ELSE ecrire(tampon,2);
    END;
END ecrit_entree;

(*****************************************************************************)

PROCEDURE trace_err_parc;

VAR str:ARRAY[Ø..2] OF CHAR;

BEGIN

    CopyString(trace,tab_mess_trace[27]);
    ConvNumberToString(str,LONGINT(num_anc_classe),FALSE,1Ø,2,"Ø");
    ConcatString(trace,str);
    ConcatString(trace,' / ');
    ConvNumberToString(str,LONGINT(nb_classe),FALSE,1Ø,2,"Ø");
    ConcatString(trace,str);
    ecrire_trace(trace);
    CopyString(trace,tab_mess_trace[28]);
    ConvNumberToString(str,LONGINT(nb_cadre_desig),FALSE,1Ø,2,"Ø");
    ConcatString(trace,str);
    ecrire_trace(trace);
    ecrire_trace(tab_mess_trace[29]);

END trace_err_parc;
(*****************************************************************************)

PROCEDURE clic_gauche(VAR w:Window;VAR message:MsgData;clic:CARDINAL);

BEGIN
  IF ok_clic
  THEN
     curx := w.MouseX;
     cury := w.MouseY;
     couleur := ReadPixel(rp^,curx,cury);
     IF  ((couleur#Ø) AND
          (curx>Ø) AND (curx<319) AND (cury>Ø) AND (cury<22Ø))
     THEN
       ModifyIDCMP(win^,IDCMPFlagsSet{});
       detection_cadre_parc(num_cadre,num_classe,curx,cury);
       IF tab_param[1] = 1 (* apprentissage *)
       THEN
           verifie_parcours(num_cadre,num_anc_cadre,num_classe,num_anc_classe,
                            res);
           CASE  res OF
            1: renforcement_positif;

               (* TRACE *)
               CopyString(trace,tab_mess_trace[14]);
               ConcatString(trace,tab_mess_trace[21]);
               ConcatString(trace,tab_nom_cadre_obl[num_classe]);
               ecrire_trace(trace);
               (* FIN TRACE *)

               IF num_classe # num_anc_classe
               THEN nb_classe_desig := nb_classe_desig +1;
               END;
               IF num_cadre # num_anc_cadre
               THEN nb_cadre_desig := nb_cadre_desig +1
               END;
```

```
IF tab_param[4]=1
THEN renforcement_parcours(num_cadre,num_anc_cadre)
     (* ecrire la trace *)
ELSE IF tab_param[4]=2
     THEN colorie_partie_parcours(num_cadre,num_anc_cadre)
          (* ecrire la trace *)
     ELSE coloration_combinee(num_cadre,num_anc_cadre)
          (* ecrire la trace *)
     END;
END;
IF (num_classe = nb_classe)
THEN IF tab_param[4]=1
     THEN renforcement_parcours(nb_enonce,num_cadre)
          (* ecrire la trace *)
     ELSE IF tab_param[4]=2
          THEN colorie_partie_parcours(nb_enonce,num_cadre)
               (*a ecrire la trace *)
          ELSE coloration_combinee(nb_enonce,num_cadre)
               (* ecrire la trace *)
          END;
     END;
END;
num_anc_cadre := num_cadre;
IF num_classe # Ø
THEN num_anc_classe := num_classe;
END;¦
```

2: renforcement_negatif;

```
(* TRACE *)
CopyString(trace,tab_mess_trace[14]);
ConcatString(trace,tab_mess_trace[21]);
ConcatString(trace,tab_nom_cadre_obl[num_classe]);
ecrire_trace(trace);
ecrire_trace(tab_mess_trace[22]);
CopyString(trace,tab_mess_trace[23]);
ConcatString(trace,tab_mess_trace[25]);
ecrire_trace(trace);
trace_err_parc;
(* FIN TRACE *)

effacer_cadre(166,223,29,148);
ecrire(tab_mess_text[20],2);
IF tab_param[6]=1
THEN dire(17);
END;
Delay(50);
effacer_cadre(166,223,29,148);
ecrire(tab_mess_text[25],2);
IF tab_param[6]=1
THEN dire(16);
END;
Delay(50); ¦
```

3: renforcement_negatif;

```
(* TRACE *)
IF num_classe # Ø
THEN
   CopyString(trace,tab_mess_trace[14]);
   ConcatString(trace,tab_mess_trace[21]);
   ConcatString(trace,tab_nom_cadre_obl[num_classe]);
   ecrire_trace(trace);
END;
ecrire_trace(tab_mess_trace[22]);
CopyString(trace,tab_mess_trace[23]);
```

```
            ConcatString(trace,tab_mess_trace[24]);
            ConcatString(trace,tab_nom_cadre_obl[num_anc_classe+1]);
            ecrire_trace(trace);
            trace_err_parc;
            (* FIN TRACE *)

            effacer_cadre(166,223,29,148);
            ecrire(tab_mess_text[26],1);
            IF num_anc_classe = Ø
            THEN ecrit_entree;
            ELSE ecrire(tab_nom_cadre_obl[num_anc_classe+1],2);
            END;
            IF tab_param[6]=1
            THEN dire(17);
            dire(19+(num_anc_classe+1));
            END;
            Delay(5Ø);
            effacer_cadre(166,223,29,148);
            ecrire(tab_mess_text[25],2);
            IF tab_param[6]=1
            THEN dire(16);
            END;
            (* ecrire la trace *)
            Delay(5Ø); ¦

         4: renforcement_negatif;
            effacer_cadre(166,223,29,148);
            ecrire(tab_mess_text[25],2);
            IF tab_param[6]=1
            THEN dire( 16);
            END;

            (* TRACE *)
            ecrire_trace(tab_mess_trace[22]);
            CopyString(trace,tab_mess_trace[23]);
            ConcatString(trace,tab_mess_trace[26]);
            ecrire_trace(trace);
            trace_err_parc;
            (* FIN TRACE *)

            Delay(5Ø); ¦
      END;
   ELSE              (* évaluation *)
      IF num_cadre # Ø
      THEN
         IF num_anc_cadre < num_cadre
         THEN
            IF tab_param[4]=1
            THEN renforcement_parcours(num_cadre,num_anc_cadre)

            ELSE IF tab_param[4]=2
               THEN colorie_partie_parcours(num_cadre,num_anc_cadre)
               ELSE coloration_combinee(num_cadre,num_anc_cadre);
               END;
            END;

            IF (num_classe = nb_classe)
            THEN IF tab_param[4]=1
                 THEN renforcement_parcours(nb_enonce,num_cadre)
                 ELSE IF tab_param[4]=2
                      THEN colorie_partie_parcours(nb_enonce,num_cadre)
                      ELSE coloration_combinee(nb_enonce,num_cadre)
                      END;
                 END;
            fin_parcours := TRUE;
            END;
```

```
                nb_cadre_desig := nb_cadre_desig + 1;
              END;
              num_anc_cadre := num_cadre;


            (* TRACE *)
            CopyString(trace,tab_mess_trace[14]);
            IF num_classe # Ø
            THEN
              ConcatString(trace,tab_mess_trace[21]);
              ConcatString(trace,tab_nom_cadre_obl[num_classe]);
             ELSE
              ConcatString(trace,tab_mess_trace[81]);
             END;
              ecrire_trace(trace);
              (* FIN TRACE *)

          ELSE
            (* TRACE *)
            CopyString(trace,tab_mess_trace[13]);
            ecrire_trace(trace);
            (* FIN TRACE *)
          END;
       END;
       ModifyIDCMP(win^,IDCMPFlagsSet{GadgetUp,MouseButtons});
       ok_clic := FALSE;
     END;
  ELSE ok_clic := TRUE;
  END;
END clic_gauche;

(*****************************************************************************)

PROCEDURE gestion_gadget(VAR w:Window;VAR msg:MsgData;VAR gad:Gadget);

VAR
   j: INTEGER;

BEGIN
  ok_clic := TRUE;
  CASE gad.GadgetID OF
     Ø: fini := TRUE;

        (* TRACE *)
        ecrire_trace(tab_mess_trace[16]);
        (* FIN TRACE *) ¦

     1: IF tab_param[3]=1  (* enonce vocal *)
        THEN IF tab_param[6]=1
             THEN dire(9);
               j := 1;
               WHILE j <= nb_mess_sauv DO
                 dire(sauvegarde_mess[j]);
                 j:= j+1;
               END;
             END;
        END;

        (* TRACE *)
        ecrire_trace(tab_mess_trace[3]);
        (* FIN TRACE *)  ¦

     2: arret := TRUE;

        (* TRACE *)
        ecrire_trace(tab_mess_trace[8]);
```

```
        (* FIN TRACE *)  ;
    ELSE;

    END;

END gestion_gadget;

(*****************************************************************************)


PROCEDURE gestion_parcours(VAR stop:BOOLEAN);

VAR boo : BOOLEAN;
    str : ARRAY [0..2] OF CHAR;

BEGIN
    WITH wp DO
      procGadgetUp := gestion_gadget;
      procMouseButtons := clic_gauche;
    END;
    BeginGadgetList();
    AddGadgetImageButton(4,222,tab_icone[1]);  (* fini *)
    AddGadgetImageButton(56,222,tab_icone[2]);  (* oreille *)
    AddGadgetImageButton(108,222,tab_icone[3]);  (*stop *)
    glist := EndGadgetList();
    win := CreateWindow(0,0,320,256,WIDCMP,WFlags,glist,scr,NIL);
    rp := win^.RPort;
    vp := ADR(scr^.ViewPort);
          (* fait le cadre de l'ecran *)
    dessine_rect(0,0,319,220);
          (* cadre du texte avec ombrage *)
    dessine_rect(165,222,150,31);
    dessine_rect(0,0,319,220);
    dessine_rect(165,222,150,31);
    Move(rp^,169,254);
    Draw(rp^,317,254);
    Move(rp^,169,255);
    Draw(rp^,317,255);
    Move(rp^,316,226);
    Draw(rp^,316,253);
    Move(rp^,317,226);
    Draw(rp^,317,253);
    graphique_support.LeftEdge := 0;
    graphique_support.TopEdge := 0;
    aff_graphique(TRUE);
    IF stop (* debut d'un exercice *)
    THEN (* stop = vrai *)
         lire_reprise_parc(num_anc_cadre,num_anc_classe,
                           nb_cadre_desig,nb_classe_desig);
         boo := Execute(ADR("init_ex"),NIL,NIL);


    END;
    IF tab_param[2] =1
    THEN (* parcours au debut *)
              CASE  tab_param[4] OF
                  1: renforcement_parcours(nb_enonce,0);
                     (* renforcement par symbole *)
                  2: colorie_partie_parcours(nb_enonce,0);
                     (* renforcement par coloration *)
                  3: coloration_combinee(nb_enonce,0);
                     (* renforcement par symbole et coloration *)
              END;
              graphique_support.LeftEdge := 0;
              graphique_support.TopEdge := 0;
              aff_graphique(TRUE);
```

```
END;
IF stop
THEN
     CASE  tab_param[4] OF
           1: renforcement_parcours(num_anc_cadre,Ø);¦
                (* renforcement par symbole *)
           2: colorie_partie_parcours(num_anc_cadre,Ø);¦
                (* renforcement par coloration *)
           3: coloration_combinee(num_anc_cadre,Ø);¦
                (* renforcement par symbole et coloration *)
      END;
END;

ok_clic := TRUE;
arret := FALSE;
fini:=FALSE;
num_classe := Ø;
confirme := FALSE;
fin_parcours := FALSE;
reinit := TRUE;
WHILE (NOT confirme) AND (NOT fin_parcours) DO
  IF   reinit
  THEN IF (NOT stop)
       THEN graphique_support.LeftEdge := Ø;
            graphique_support.TopEdge := Ø;
            aff_graphique(TRUE);
       END;
       IF tab_param[3] =1
       THEN (* montrer l'entrée et la sortie *)
            CASE tab_param[4] OF
                 1: (* renforcement symbole *)
                    IF (NOT stop)
                    THEN
                      deplacement_symbole(1);
                    END;
                    calcul_centre_cadre(nb_enonce,c1,c2);
                    aff_image(c1.x,c1.y,symbole);
                    aff_image(c2.x,c2.y,symbole);¦
                 2: (* renforcement par coloration *)
                     IF (NOT stop)
                     THEN
                        x := tab_cadre_parc[1][1]
                             + graphique_support.LeftEdge;
                        y := tab_cadre_parc[1][2]
                             + graphique_support.TopEdge;
                        l := tab_cadre_parc[1][3];
                        h := tab_cadre_parc[1][4];
                        coloration_zone(x,y,l,h,99,coul_parc);
                        IF tab_cadre_parc[1][5]#5ØØ
                        THEN
                          x := tab_cadre_parc[1][5]
                               + graphique_support.LeftEdge;
                          y := tab_cadre_parc[1][6]
                               + graphique_support.TopEdge;
                          l := tab_cadre_parc[1][7];
                          h := tab_cadre_parc[1][8];
                          coloration_zone(x,y,l,h,99,coul_parc);
                        END;
                     END;
                     x := tab_cadre_parc[nb_enonce][1]
                          + graphique_support.LeftEdge;
                     y := tab_cadre_parc[nb_enonce][2]
                          + graphique_support.TopEdge;
                     l := tab_cadre_parc[nb_enonce][3];
                     h := tab_cadre_parc[nb_enonce][4];
                     coloration_zone(x,y,l,h,99,coul_parc);
```

```
                    IF tab_cadre_parc[nb_enonce][5] # 500
                    THEN
                      x := tab_cadre_parc[nb_enonce][5]
                           + graphique_support.LeftEdge;
                      y := tab_cadre_parc[nb_enonce][6]
                           + graphique_support.TopEdge;
                      l := tab_cadre_parc[nb_enonce][7];
                      h := tab_cadre_parc[nb_enonce][8];
                      coloration_zone(x,y,l,h,99,coul_parc);
                    END; |
                3: (* renforcement par coloration et symbole *)
                    IF (NOT stop)
                    THEN
                       x := tab_cadre_parc[1][1]
                            + graphique_support.LeftEdge;
                       y := tab_cadre_parc[1][2]
                            + graphique_support.TopEdge;
                       l := tab_cadre_parc[1][3];
                       h := tab_cadre_parc[1][4];
                       coloration_zone(x,y,l,h,99,coul_parc);
                       IF tab_cadre_parc[1][5]#500
                       THEN
                         x := tab_cadre_parc[1][5]
                              + graphique_support.LeftEdge;
                         y := tab_cadre_parc[1][6]
                              + graphique_support.TopEdge;
                         l := tab_cadre_parc[1][7];
                         h := tab_cadre_parc[1][8];
                         coloration_zone(x,y,l,h,99,coul_parc);
                       END;
                       deplacement_symbole(1);
                    END;
                    x := tab_cadre_parc[nb_enonce][1]
                         + graphique_support.LeftEdge;
                    y := tab_cadre_parc[nb_enonce][2]
                         + graphique_support.TopEdge;
                    l := tab_cadre_parc[nb_enonce][3];
                    h := tab_cadre_parc[nb_enonce][4];
                    coloration_zone(x,y,l,h,99,coul_parc);
                    IF tab_cadre_parc[nb_enonce][5] # 500
                    THEN
                      x := tab_cadre_parc[nb_enonce][5]
                           + graphique_support.LeftEdge;
                      y := tab_cadre_parc[nb_enonce][6]
                           + graphique_support.TopEdge;
                      l := tab_cadre_parc[nb_enonce][7];
                      h := tab_cadre_parc[nb_enonce][8];
                      coloration_zone(x,y,l,h,99,coul_parc);
                    END;
                    calcul_centre_cadre(nb_enonce,c1,c2);
                    aff_image(c1.x,c1.y,symbole);
                    aff_image(c2.x,c2.y,symbole);
            END;
            IF stop
            THEN
              stop:= FALSE;
            ELSE
              num_anc_cadre := 1;
              num_anc_classe := 1;
              nb_cadre_desig := 0;
              nb_classe_desig := 0;
            END;

        ELSE IF stop
            THEN
              stop := FALSE;
```

```
            ELSE
              num_anc_cadre := Ø;
              num_anc_classe := Ø;
              nb_cadre_desig := Ø;
              nb_classe_desig := Ø;
            END;
      END;
      effacer_cadre(166,223,29,148);
      ecrire(tab_mess_text[9],1);
      ecrire(tab_mess_text[24],2);
      ecrire(nom_symbole,3);
      IF tab_param[6]=1
      THEN dire(14);
           dire(15);
           dire(19);
           sauvegarde_mess[1]:=14;
           sauvegarde_mess[2]:=15;
           sauvegarde_mess[3]:=19;
           nb_mess_sauv:= 3;
      END;

ELSE reinit := TRUE;
END;
res := 1;
WHILE (res =1) AND (NOT confirme) AND (NOT fin_parcours) AND
           (NOT fini) AND (NOT arret) DO
           sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
           LOOP
                msg := GetMsg(win^.UserPort^);
                IF msg = NIL
                THEN EXIT;
                END;
                ProcIMsg(wp,msg);
           END;
           IF (NOT ok_clic)
           THEN fin_parcours := ((res=1) AND (num_classe=nb_classe));
           ELSE fin_parcours :=FALSE
           END;

END;
IF fin_parcours
THEN renforcement_positif;
     j:=Ø;
     WHILE j<=3 DO
       effacer_cadre(166,223,29,148);
       Delay(3Ø);
       IF tab_param[1] = 1
       THEN ecrire(tab_mess_text[27],2);
       ELSE ecrire(tab_mess_text[28],2);
       END;
       Delay(3Ø);
       j:=j+1;
     END;

     (* TRACE *)
     IF tab_param[1] = 1
     THEN
       ecrire_trace(tab_mess_trace[8Ø]);
     END;
       CopyString(trace,tab_mess_trace[28]);
       ConvNumberToString(str,LONGINT(nb_cadre_desig),FALSE,1Ø,2,"Ø");
       ConcatString(trace,str);
       ecrire_trace(trace);
     (* FIN TRACE *)

END;
```

```
        IF (fini OR arret)
        THEN    reinit := FALSE;
                j := 1;
                WHILE j <= nb_mess_sauv DO
                  temp[j]:=sauvegarde_mess[j];
                  j:= j+1;
                END;
                IF fini
                THEN num_mess_texte := 7;
                     nb_mess_sauv := 1;
                     sauvegarde_mess[1]:= 7;
                ELSE num_mess_texte := 8;
                     nb_mess_sauv := 1;
                     sauvegarde_mess[1]:= 8;

                END;

                confirmation(tab_mess_text[num_mess_texte],
                             sauvegarde_mess[1],confirme);
                IF (NOT confirme)
                THEN  fini:=FALSE;
                      arret:=FALSE;
                      j := 1;
                      WHILE j <= nb_mess_sauv DO
                        sauvegarde_mess[j]:= temp[j];
                        j:= j+1;
                      END;
                END;

          END; (* fini ou arrêt *)

      END; (* grand while *)
      CloseWindow(win^);
      FreeGadgetList(glist^);
      stop := arret;



END gestion_parcours;

END parcours.
```

```
IMPLEMENTATION MODULE textuel;

FROM var_globale IMPORT string80,string40,string20,scr,rp,win,tab_icone,
     tab_mess_text,nb_enonce,tab_cadre_nom,tab_cadre_desig,LONG_CADRE,
     HAUT_CADRE,sauvegarde_mess,tab_param,tab_exercice,num_ex,type_saisie,
     nom_user,tab_mess_trace;
FROM son IMPORT dire;
FROM trt_donnees IMPORT pluriel,ecrire_trace;
FROM graphique IMPORT dessine_rect,ecrire,clignoter_zone;

FROM SYSTEM IMPORT ADR,ADDRESS,BYTE;
FROM Intuition IMPORT Window,WindowPtr,GadgetPtr,Gadget,WindowFlags,
     WindowFlagsSet,IDCMPFlagsSet,IDCMPFlags,IntuiMessagePtr,CloseWindow,
     StringInfoPtr,StringInfo,CloseScreen,ActivateGadget,RefreshGadgets,
     ModifyIDCMP,GadgetFlags,GadgetFlagsSet;
FROM Ports IMPORT GetMsg;
FROM SimpleGadgets IMPORT BeginGadgetList,EndGadgetList,FreeGadgetList,
     AddGadgetImageButton,AddGadgetTextButton,AddGadgetString;
FROM SimpleIDCMP IMPORT MsgData,WindowProc,ProcIMsg;
FROM SimpleScreens IMPORT CreateScreen;
FROM SimpleWindows IMPORT CreateWindow;
FROM Tasks IMPORT SignalSet,Wait;
FROM Rasters IMPORT RastPortPtr;
FROM Text IMPORT TextLength,Text;
FROM Drawing IMPORT Move,Draw,SetAPen,RectFill,SetBPen;
FROM Strings IMPORT StringLength,ConcatString,CopyString,ExtractSubString,
                    CompareString,Relation,ConvStringToUpperCase;
FROM Memory IMPORT CopyMem;
FROM AmigaDosProcess IMPORT Delay;
FROM Intuition IMPORT DisplayBeep;


VAR confirme, fini,b,fin_saisie,abandon : BOOLEAN;
    trace : string80;
    gglob : GadgetPtr;
    wglob : WindowPtr;
    rastport : RastPortPtr;
    chaine_nom,buf : string20;
(*****************************************************************************)

PROCEDURE gestion_gadget1 (VAR w:Window;VAR msg : MsgData;VAR gad : Gadget);

VAR temps : INTEGER;
BEGIN
   CASE  gad.GadgetID OF

        0 : dire(9);
            dire(sauvegarde_mess[1]); (* gestion clic sur oreille*)
            (* TRACE *)
            ecrire_trace(tab_mess_trace[3]);
            (* FIN TRACE *) |

        1 : confirme := TRUE;
            fini := TRUE;
            (* TRACE *)
            ecrire_trace(tab_mess_trace[9]);
            (* FIN TRACE *)
            (* ecrire trace clic sur oui *) |

        2 : confirme := FALSE;
            fini := TRUE;
            (* TRACE *)
            ecrire_trace(tab_mess_trace[10]);
            (* FIN TRACE *)
            (* ecrire trace clic sur non *) |
   END;
```

```
END gestion_gadget1;

(*****************************************************************************)

PROCEDURE confirmation
        (mess_texte : string40;mess_voc :INTEGER;VAR confirmation : BOOLEAN);

CONST WIDCMP = IDCMPFlagsSet{GadgetUp};
      WFlags = WindowFlagsSet{Activate,Borderless};

VAR win : WindowPtr;
    glist : GadgetPtr;
    wp : WindowProc;
    sig : SignalSet;
    msg : IntuiMessagePtr;
    rp : RastPortPtr;

BEGIN
   WITH wp DO
      procGadgetUp:=gestion_gadget1;
   END;
   BeginGadgetList();
      AddGadgetImageButton(56,15,tab_icone[2]);          (* oreille *)
      AddGadgetTextButton(130,27,ADR(tab_mess_text[18])); (* oui *)
      AddGadgetTextButton(190,27,ADR(tab_mess_text[19])); (* non *)
   glist:=EndGadgetList();
   win:=CreateWindow(0,205,320,51,WIDCMP,WFlags,glist,scr,NIL);
   rp:=win^.RPort;
   SetAPen(rp^,11); (* noir *)
   RectFill(rp^,0,0,320,51);
   SetAPen(rp^,1);
   SetBPen(rp^,11);
   RefreshGadgets(glist^,win^,NIL);

   Move(rp^,133,38);
   Draw(rp^,164,38);
   Move(rp^,133,39);
   Draw(rp^,164,39);
   Move(rp^,163,30);
   Draw(rp^,163,38);
   Move(rp^,164,30);
   Draw(rp^,164,38);

   Move(rp^,193,38);
   Draw(rp^,224,38);
   Move(rp^,193,39);
   Draw(rp^,224,39);
   Move(rp^,223,30);
   Draw(rp^,223,38);
   Move(rp^,224,30);
   Draw(rp^,224,38);

   Move(rp^,(320-TextLength(rp^,ADR(mess_texte),StringLength(mess_texte)))
                                          DIV 2,10);
   Text(rp^,ADR(mess_texte),StringLength(mess_texte));
   dire(mess_voc);
   fini:=FALSE;

   (*** TRACE ***)
   CopyString(trace,mess_texte);
   ecrire_trace(trace);
   (*** FIN-TRACE ***)

   WHILE (NOT fini) DO
      sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
      LOOP
```

```
        msg:=GetMsg(win^.UserPort^);
        IF (msg=NIL) THEN
           EXIT;
        END;
        ProcIMsg(wp,msg);
      END;
   CloseWindow(win^);
   FreeGadgetList(glist^);
   confirmation:=confirme;
END confirmation;


(*****************************************************************************)

PROCEDURE met_nom_ds_cadre;

VAR i,x,y: INTEGER;

BEGIN
  i:= 1;
  WHILE i<= nb_enonce DO
    dessine_rect(tab_cadre_nom[i].x,tab_cadre_nom[i].y,LONG_CADRE,HAUT_CADRE);
    SetAPen(rp^,10);  (* bleu fonce *)
    RectFill(rp^,tab_cadre_nom[i].x +1,tab_cadre_nom[i].y+1,
           tab_cadre_nom[i].x-1+LONG_CADRE,tab_cadre_nom[i].y-1+HAUT_CADRE);
    SetAPen(rp^,1);
    SetBPen(rp^,10);
    x:= tab_cadre_nom[i].x+2+(LONG_CADRE DIV 2)-(TextLength(rp^,
        ADR(tab_cadre_desig[i].nom_cadre),
           StringLength(tab_cadre_desig[i].nom_cadre))DIV 2);
    y :=tab_cadre_nom[i].y+10;
    Move(rp^,x,y);
    Text(rp^,ADR(tab_cadre_desig[i].nom_cadre),
         StringLength(tab_cadre_desig[i].nom_cadre));
    i := i+1;
  END;
  SetBPen(rp^,0);
END met_nom_ds_cadre;

(*****************************************************************************)

PROCEDURE ecrire_enonce(num_enonce:INTEGER);

VAR str : string20;

BEGIN
  IF tab_param[1]=3 (* PP *)
  THEN IF tab_param[8]=1 (* par nom *)
       THEN ecrire(tab_mess_text[9],1); (*  Veux_tu montrer *)
            ecrire(tab_mess_text[22],2);  (* un nom *)

             (* TRACE*)
             CopyString(trace,tab_mess_trace[15]);
             ConcatString(trace,tab_mess_trace[78]);
             ecrire_trace (trace);
             (* FIN TRACE *)

       ELSE (* par brush *)
            ecrire(tab_mess_text[9],1); (* Veux_tu montrer *)
            ecrire(tab_mess_text[15],2); (*  un organe *)
            (* TRACE*)
            CopyString(trace,tab_mess_trace[15]);
            ConcatString(trace,tab_mess_trace[79]);
            ecrire_trace (trace);
            (* FIN TRACE *)
       END;
```

```
ELSE (* evaluation ou apprentissage *)
     IF tab_param[5]=2 (* non vide *)
     THEN IF tab_exercice[num_ex].anim=1 (* par systeme *)
          THEN ecrire(tab_mess_text[9],1); (* Veux_tu montrer *)
               ecrire(tab_mess_text[12],2); (* ce qui permet de *)
               ExtractSubString(str,tab_cadre_desig[num_enonce].nom_cadre,
                    Ø,StringLength(tab_cadre_desig[num_enonce].nom_cadre)-2);
               ecrire(str,3);

               (* TRACE*)
               CopyString(trace,tab_mess_trace[15]);
               ConcatString(trace,tab_mess_trace[2Ø]);
               ConcatString(trace,str);
               ecrire_trace (trace);
               (* FIN TRACE *)

          ELSE IF tab_param[8]=1 (* par nom *)
               THEN ecrire(tab_mess_text[9],1); (* Veux_tu montrer *)
                    ecrire(tab_mess_text[13],2); (* son nom *)
                    clignoter_zone(num_enonce);

                    (* TRACE*)
                    CopyString(trace,tab_mess_trace[15]);
                    ConcatString(trace,tab_mess_trace[19]);
                    ConcatString(trace,tab_cadre_desig[num_enonce].nom_cadre);
                    ecrire_trace (trace);
                    (* FIN TRACE *)

               ELSE ecrire(tab_mess_text[9],1); (* Veux_tu montrer *)
                    ecrire(tab_cadre_desig[num_enonce].nom_cadre,2);

                    (* TRACE*)
                    CopyString(trace,tab_mess_trace[15]);
                    ConcatString(trace,tab_cadre_desig[num_enonce].nom_cadre);
                    ecrire_trace (trace);
                    (* FIN TRACE *)

               END;
          END;
     ELSE (* vide *)
          IF pluriel(tab_cadre_desig[num_enonce].nom_cadre)
          THEN
            ecrire(tab_mess_text[9],1); (* Veux_tu montrer *)
            ecrire(tab_mess_text[21],2); (* l'endroit où sont *)
            ecrire(tab_cadre_desig[num_enonce].nom_cadre,3);

            (* TRACE*)
            CopyString(trace,tab_mess_trace[15]);
            ConcatString(trace,tab_mess_trace[18]);
            ConcatString(trace,tab_cadre_desig[num_enonce].nom_cadre);
            ecrire_trace (trace);
            (* FIN TRACE *)

          ELSE
            ecrire(tab_mess_text[9],1); (* Veux_tu montrer *)
            ecrire(tab_mess_text[17],2); (* l'endroit où est *)
            ecrire(tab_cadre_desig[num_enonce].nom_cadre,3);

            (* TRACE*)
            CopyString(trace,tab_mess_trace[15]);
            ConcatString(trace,tab_mess_trace[17]);
            ConcatString(trace,tab_cadre_desig[num_enonce].nom_cadre);
            ecrire_trace (trace);
            (* FIN TRACE *)

          END;
```

```
        END;
    END;
END ecrire_enonce;

(*****************************************************************************)

PROCEDURE gestion_gadget(VAR w:Window;VAR msg : MsgData; VAR gad : Gadget);

VAR str_info_ptr : StringInfoPtr;
    (*temp : string20;*)

BEGIN
   CASE gad.GadgetID OF
      1 : dire(9);
          dire (1);
          b:=ActivateGadget(gglob^,wglob^,NIL);
          (* TRACE *)
          ecrire_trace(tab_mess_trace[3]);
          (* FIN-TRACE *) ¦


      0 : str_info_ptr:=gad.SpecialInfo;
          fin_saisie:=TRUE;
          CopyMem(str_info_ptr^.Buffer,ADR(chaine_nom),20);
          (* TRACE *)
          CopyString(trace,tab_mess_trace[1]);
          ConcatString(trace,chaine_nom);
          ecrire_trace(trace);
          (* FIN-TRACE *) ¦

      2 : abandon:=TRUE;
          (* TRACE *)
          ecrire_trace(tab_mess_trace[2]); (* l'user a abondonné son nom*)
          (* FIN-TRACE *) ¦
 END;
END gestion_gadget;

(*****************************************************************************)

PROCEDURE saisie_nom(VAR abandonne : BOOLEAN);

CONST WIDCMP = IDCMPFlagsSet{GadgetUp};
      WFlags = WindowFlagsSet{Activate,Borderless};

VAR wp : WindowProc;
    sig : SignalSet;
    msg : IntuiMessagePtr;
    rp : RastPortPtr;
    nom_ok,confirme,OK : BOOLEAN;

BEGIN
   WITH wp DO
      procGadgetUp:=gestion_gadget;
   END;
   IF (type_saisie="K") OR (type_saisie = "k")
   THEN
      (*** saisie au clavier ***)
      ConvStringToUpperCase(nom_user);
   ELSE
      (* saisie par comparaison *)
      (* Nom_user contient le nom inscrit sur la disquette "USER" *)
      (* Cette variable est mise a jour par donnee_session dans le*)
      (* coordinateur                                              *)
   END;
   BeginGadgetList();
   IF (type_saisie="K") OR (type_saisie = "k")   THEN
```

```
   (*** saisie au clavier ***)
   AddGadgetString(100,80,15,15,ADR(buf));
   AddGadgetImageButton(56,222,tab_icone[2]);
   AddGadgetTextButton(129,115,ADR(tab_mess_text[23]));
   gglob:=EndGadgetList();
ELSE
   (*** saisie par comparaison ***)
   gglob:=NIL;
END;
wglob:=CreateWindow(0,0,320,256,WIDCMP,WFlags,gglob,scr,NIL);
rp:=wglob^.RPort;
SetAPen(rp^,1);
Move(rp^,0,220);
Draw(rp^,320,220);
Move(rp^,0,0);
Draw(rp^,0,220);
Move(rp^,0,0);
Draw(rp^,319,0);
Move(rp^,319,0);
Draw(rp^,319,220);
IF (type_saisie="K") OR (type_saisie = "k")
 THEN

   (*** relief gadget saisie nom ***)
   Move(rp^,103,90);
   Draw(rp^,224,90);
   Move(rp^,103,91);
   Draw(rp^,224,91);
   Move(rp^,223,82);
   Draw(rp^,223,90);
   Move(rp^,224,82);
   Draw(rp^,224,90);

   (*** relief gadget abandon ***)
   Move(rp^,132,126);
   Draw(rp^,195,126);
   Move(rp^,132,127);
   Draw(rp^,195,127);
   Move(rp^,194,118);
   Draw(rp^,194,126);
   Move(rp^,195,118);
   Draw(rp^,195,126);

   b:=ActivateGadget(gglob^,wglob^,NIL);
   dire(1);
   nom_ok:=FALSE;
   abandon:=FALSE;
   WHILE (NOT nom_ok) AND (NOT abandon) DO
      SetAPen(rp^,0);
      RectFill(rp^,100,35,260,60);
      SetAPen(rp^,1);
      Move(rp^,(320-TextLength(rp^,ADR(tab_mess_text[3]),
                StringLength(tab_mess_text[3]))) DIV 2,45);
      Text(rp^,ADR(tab_mess_text[1]),StringLength(tab_mess_text[1]));
      sig:=Wait(SignalSet{CARDINAL(wglob^.UserPort^.mpSigBit)});
      LOOP
         msg:=GetMsg(wglob^.UserPort^);
         IF (msg=NIL) THEN
            EXIT;
         END;
         ProcIMsg(wp,msg);
      END;
      IF fin_saisie THEN
         ModifyIDCMP(wglob^,IDCMPFlagsSet{});
         sauvegarde_mess[1] := 2;
         confirmation(tab_mess_text[2],2,confirme);
```

```
            IF confirme THEN
               ConvStringToUpperCase(chaine_nom);
               IF CompareString(nom_user,chaine_nom) = equal
               THEN nom_ok := TRUE
               ELSE fin_saisie := FALSE;
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    DisplayBeep(scr);
                    dire(3);
                    dire(1);
                    SetAPen(rp^,0);
                    RectFill(rp^,100,35,260,60);
                    Move(rp^,(320-TextLength(rp^,ADR(tab_mess_text[3]),
                    StringLength(tab_mess_text[3]))) DIV 2,45);
                    SetAPen(rp^,1);
                    Text(rp^,ADR(tab_mess_text[3]),StringLength
                                              (tab_mess_text[3]));
                    Delay(100);

                    (* TRACE *)
                    ecrire_trace(tab_mess_trace[6]);
                    (* FIN-TRACE *)

                  b:=ActivateGadget(gglob^,wglob^,NIL);
               END;
            ELSE
               fin_saisie:=FALSE;
               dire(1);
               b:=ActivateGadget(gglob^,wglob^,NIL);
            END;
            ModifyIDCMP(wglob^,WIDCMP);
         END;
      END;
   ELSE
      (*** saisie par comparaison ***)
      (*** affichage du nom ***)
      Move(rp^,(320-TextLength(rp^,ADR(nom_user),
                        StringLength(nom_user))) DIV 2,80);
      Text(rp^,ADR(nom_user),StringLength(nom_user));
      Move(rp^,90,63);
      Draw(rp^,230,63);
      Move(rp^,90,63);
      Draw(rp^,90,91);
      Move(rp^,90,91);
      Draw(rp^,230,91);
      Move(rp^,230,63);
```

```
      Draw(rp^,230,91);

      Move(rp^,94,92);
      Draw(rp^,232,92);
      Move(rp^,94,93);
      Draw(rp^,232,93);
      Move(rp^,231,67);
      Draw(rp^,231,91);
      Move(rp^,232,67);
      Draw(rp^,232,91);
      confirmation(tab_mess_text[2],2,abandon);
      abandon:=NOT abandon; (** artifice : booleen a signification multiple **)
   END;
   CloseWindow(wglob^);
   FreeGadgetList(gglob^);
   abandonne:=abandon;
END saisie_nom;


END textuel.
```

```
IMPLEMENTATION MODULE base_donnees;

FROM var_globale IMPORT tab_mess_text, string2Ø,string4Ø,scr,tab_param,string8Ø,
     tab_exercice, tab_cadre_desig,tab_couleur,symbole, nom_user,type_saisie,
     type_renfor,stop,num_ex,tab_mess_trace,tab_cadre_parc,tab_cadre_obl,
     tab_nom_cadre_obl,tab_icone,tab_cartoon,nb_classe,nb_enonce,nom_symbole,
     tab_son,liste_enonce,tab_cadre_nom,minute,coul_parc,entree,nb_cartoon,
     tab_anim,pos_tab_trace;
FROM var_globale_trace IMPORT tab_trace;
FROM graphique IMPORT afficher_cartoon;

FROM SoundLib IMPORT LoadAIFF;
FROM MathLibØ IMPORT real,entier;
FROM InOut IMPORT OpenInputFile,CloseInput,ReadInt,Done,Read,ReadString,
     OpenOutputFile,CloseOutput,WriteInt,Write,WriteString,WriteLn,EOL;
FROM AmigaDos IMPORT IoErr,ModeOldFile,FileHandle,Open,Close,DateStamp,
     DateStampRecord,DeleteFile;
FROM Intuition IMPORT ScreenToFront,CloseScreen,Image,ScreenPtr;
FROM Strings IMPORT ConcatString,CopyString;
FROM Graphics IMPORT BitMap;
FROM ReadPict IMPORT ILBMFrame,ReadPicture;
FROM SimpleScreens IMPORT CreateScreen;
FROM RemAlloc IMPORT ChipAlloc;
FROM IFF IMPORT IFFP;
FROM SYSTEM IMPORT ADR,BYTE,SHORT;
FROM Conversions IMPORT ConvNumberToString;
IMPORT FileSystem;
FROM FindImages IMPORT ImageDescTablePtr,FindImageTable;

VAR duree : LONGCARD;
    fichier_pres : FileSystem.File;
(*****************************************************************************)

PROCEDURE lire_param( VAR nom_fich:string2Ø);

VAR
    i : INTEGER;
    str : string2Ø;
    afficher : BOOLEAN;

BEGIN
    CopyString(str,'user:');
    ConcatString(str,nom_fich);
    OpenInputFile(str);
     IF IoErr() # LONGINT(Ø)
         THEN ScreenToFront(scr^);
     END;
     afficher := (NOT entree);
     i:= 1;

     WHILE Done  DO
           afficher_cartoon(afficher);
           ReadInt(tab_param[i]);
           i:=i+1;

    END;
   afficher_cartoon(afficher);
   CloseInput;

END lire_param;

(*****************************************************************************)

PROCEDURE saisie_exercice(VAR nb_ex:INTEGER);
VAR
```

```
   i:INTEGER;
   str:string20;

BEGIN
   OpenInputFile('USER:liste_exercice');
     IF IoErr() # LONGINT(0)
         THEN ScreenToFront(scr^);
     END;
     i:= 0;

     WHILE Done  DO
           i:=i+1;
           Read(tab_exercice[i].type_ex);
           ReadString(tab_exercice[i].graphe_support);
           tab_exercice[i].fichier_ptr1 := '';
           tab_exercice[i].fichier_ptr2 := '';
           tab_exercice[i].fichier_param := '';
           tab_exercice[i].anim := 0;
           ReadString(tab_exercice[i].fichier_ptr1);
           ReadString(str);
           IF (tab_exercice[i].type_ex = 'p') OR
              (tab_exercice[i].type_ex = 'P')
           THEN tab_exercice[i].fichier_ptr2 := str;
                ReadString(tab_exercice[i].fichier_param);

           ELSE tab_exercice[i].fichier_param := str;
                ReadInt(tab_exercice[i].anim);
           END;



    END;
    IF (i = 0) OR (i = 1) THEN nb_ex := -1
                          ELSE nb_ex := i-1;
    END;
    CloseInput;
 END saisie_exercice;

(*****************************************************************************)

PROCEDURE charger_donnees_session;
VAR entier : INTEGER;
BEGIN
     OpenInputFile('user:donnees_session');
     IF IoErr() # LONGINT(0)
         THEN ScreenToFront(scr^);
     END;
     ReadString(nom_user);
     Read(type_saisie);
     ReadInt(type_renfor);
     ReadInt(num_ex);
     ReadInt(entier);
     IF entier = 1
     THEN stop := TRUE;
     ELSE stop := FALSE;END;
     CloseInput;

END charger_donnees_session;

(*****************************************************************************)

PROCEDURE charger_image(VAR image:Image;VAR nom_im:string20);

VAR
   pic : FileHandle;
   test : IFFP;
```

```
    bm : BitMap;
    frame : ILBMFrame;
    str : string2Ø;
    afficher:BOOLEAN;

BEGIN
     CopyString(str,'user:');
     ConcatString(str,nom_im);
     afficher := (NOT entree);
     afficher_cartoon(afficher);
     pic := Open(ADR(str),ModeOldFile);
     test := ReadPicture(pic,bm,frame,ChipAlloc);
     WITH image DO
        LeftEdge:=Ø;
        TopEdge:=Ø;
        Width :=frame.bmHdr.w;
        Height := frame.bmHdr.h;
        Depth := INTEGER(frame.bmHdr.nPlanes);
        ImageData:=bm.Planes[Ø];
        PlanePick := BYTE(31);
        PlaneOnOff := BYTE(31);
        NextImage := NIL;
     END;
     afficher_cartoon(afficher);
     Close(pic);
END charger_image;

(*****************************************************************************)

PROCEDURE maj_donnees_session(num_exercice:INTEGER;ex_stop : CHAR);

CONST
     fin_lg=12C;

VAR fichier : FileSystem.File;
    i : INTEGER;
    car:CHAR;
    num_ex_car : ARRAY[Ø..3] OF CHAR;
BEGIN
     FileSystem.Lookup(fichier,"user:donnees_session",FALSE);
     IF IoErr() # LONGINT(Ø)
         THEN ScreenToFront(scr^);
     END;
     i :=1;
     WHILE i<= 3 DO
       afficher_cartoon(TRUE);
       FileSystem.ReadChar(fichier,car);
       WHILE car # fin_lg DO
         FileSystem.ReadChar(fichier,car);
       END;
       i:=i+1;
     END;
     ConvNumberToString(num_ex_car,LONGINT(num_exercice),FALSE,1Ø,1,"Ø");
     FileSystem.WriteChar(fichier,num_ex_car[Ø]);
     FileSystem.WriteChar(fichier,fin_lg);
     FileSystem.WriteChar(fichier,ex_stop);
     afficher_cartoon(TRUE);
     FileSystem.Close(fichier);
END maj_donnees_session;

(*****************************************************************************)

PROCEDURE init_mess_text;

CONST fin_lg = 12C;
```

```
VAR
   i    : INTEGER;
   fin  : BOOLEAN;
   car  : CHAR;

BEGIN

     OpenInputFile ('message.texte');
     IF IoErr() # LONGINT (Ø)
        THEN ScreenToFront(scr^);
     END;
     i:=1;
     fin := FALSE;
     WHILE NOT fin  DO
           CopyString(tab_mess_text[i],"");
           Read (car);
           IF NOT Done
                 THEN fin := TRUE;
           END;
           WHILE (car # fin_lg) AND (NOT fin)  DO
                ConcatString(tab_mess_text[i],car);
                Read(car);
           END;
           i:=i+1;
     END;
     CloseInput;
   END init_mess_text;

(*****************************************************************************)

PROCEDURE init_mess_trace;

CONST fin_lg = 12C;

VAR
   i    : INTEGER;
   fin  : BOOLEAN;
   car  : CHAR;

BEGIN

     OpenInputFile ('trace.texte');
     IF IoErr() # LONGINT (Ø)
        THEN ScreenToFront(scr^);
     END;
     i:=1;
     fin := FALSE;
     WHILE NOT fin  DO
           CopyString(tab_mess_trace[i],"");
           Read (car);
           IF NOT Done
                 THEN fin := TRUE;
           END;
           WHILE (car # fin_lg) AND (NOT fin)  DO
                ConcatString(tab_mess_trace[i],car);
                Read(car);
           END;
           i:=i+1;
     END;
     CloseInput;
   END init_mess_trace;

(*****************************************************************************)

PROCEDURE saisie_designation;
```

```
VAR
   ent,i :INTEGER;
   str :string20;
   afficher : BOOLEAN;

BEGIN
     nb_enonce := 1;
     CopyString(str,"user:");
     ConcatString(str,tab_exercice[num_ex].fichier_ptr1);
     OpenInputFile(str);
     IF IoErr() # LONGINT (0)
        THEN ScreenToFront(scr^);
     END;
     afficher := (NOT entree);
     str := '';
     IF tab_param[7] = 2
     THEN IF Done  = TRUE
          THEN ReadInt(ent);
          END;
          WHILE Done = TRUE DO
            afficher_cartoon(afficher);
            ReadInt(ent);
            i := 1;
            CopyString(tab_cadre_desig[nb_enonce].nom_cadre,'');
            WHILE i<= ent DO
              ReadString(str);
              ConcatString(tab_cadre_desig[nb_enonce].nom_cadre,str);
              ConcatString(tab_cadre_desig[nb_enonce].nom_cadre," ");
              i := i+1;
            END;
            ReadString(str);
            charger_image(tab_cadre_desig[nb_enonce].brush,str);
            ReadString(tab_cadre_desig[nb_enonce].nom_son);
            ReadInt(tab_cadre_desig[nb_enonce].x);
            ReadInt(tab_cadre_desig[nb_enonce].y);
            ReadInt(tab_cadre_desig[nb_enonce].haut);
            ReadInt(tab_cadre_desig[nb_enonce].long);
            ReadInt(ent);
            nb_enonce := nb_enonce+1;
          END;
     ELSE afficher_cartoon(afficher);
          WHILE Done = TRUE DO
            ReadInt(tab_couleur[nb_enonce]);
            i := 1;
            ReadInt(ent);
            WHILE i <= ent DO
              ReadString(str);
              ConcatString(tab_cadre_desig[nb_enonce].nom_cadre,str);
              ConcatString(tab_cadre_desig[nb_enonce].nom_cadre," ");
              i := i+1;
            END;
            ReadString(str);
            charger_image(tab_cadre_desig[nb_enonce].brush,str);
            ReadString(tab_cadre_desig[nb_enonce].nom_son);
            ReadInt(tab_cadre_desig[nb_enonce].x);
            ReadInt(tab_cadre_desig[nb_enonce].y);
            tab_cadre_desig[nb_enonce].haut := 999;
            tab_cadre_desig[nb_enonce].long := 999;
            ReadInt(ent);
            ReadInt(ent);
            IF Done = TRUE THEN nb_enonce := nb_enonce+1;END;
          END;
     END;
     afficher_cartoon(afficher);
     CloseInput;
     nb_enonce := nb_enonce-1;
```

```
END saisie_designation;

(*****************************************************************************
***)

PROCEDURE saisie_parcours;

VAR
   ent,i :INTEGER;
   str,str1 : string2Ø;
   afficher : BOOLEAN;

BEGIN
     nb_enonce := 1;
     CopyString(str,"user:");
     ConcatString(str,tab_exercice[num_ex].fichier_ptr1);
     OpenInputFile(str);
     IF IoErr() # LONGINT (Ø)
        THEN ScreenToFront(scr^);
     END;
     afficher := (NOT entree);
     IF Done = TRUE
     THEN
          afficher_cartoon(afficher);
          ReadInt(coul_parc);
          i:= 1;
          ReadInt(ent);
          nom_symbole:='';
          WHILE i<= ent DO
            ReadString(str);
            ConcatString(nom_symbole,str);
            ConcatString(nom_symbole," ");
            i:= i+ 1;
            str := "";
          END;
          ReadString(str);
          charger_image(symbole,str);
          CopyString(str,"user:");
          ReadString(str1);
          ConcatString(str,str1);
          tab_son[19] := LoadAIFF(ADR(str));
     END;
     WHILE Done = TRUE DO
       afficher_cartoon(afficher);
       ReadInt(ent);
       IF Done = TRUE
       THEN
         i := 1;
         WHILE i<= (ent * 4) DO
           ReadInt(tab_cadre_parc[nb_enonce][i]);
           i := i+1;
       END;
       tab_cadre_parc[nb_enonce][i] := 5ØØ;
       nb_enonce := nb_enonce + 1;
       END;
     END;
     nb_enonce :=nb_enonce-1;
     afficher_cartoon(afficher);
     CloseInput;
     CopyString(str,"user:");
     ConcatString(str,tab_exercice[num_ex].fichier_ptr2);
     OpenInputFile(str);
     IF IoErr() # LONGINT (Ø)
        THEN ScreenToFront(scr^);
     END;
     nb_classe := 1;
```

```
      WHILE Done = TRUE DO
        afficher_cartoon(afficher);
        i := 1;
        ReadInt(ent);
        tab_nom_cadre_obl[nb_classe]:='';
        WHILE i<= ent DO
          ReadString(str);
          ConcatString(tab_nom_cadre_obl[nb_classe],str);
          ConcatString(tab_nom_cadre_obl[nb_classe]," ");
          i := i+1;
        END;
        str:='';
        ReadString(str1);
        CopyString(str,"user:");
        ConcatString(str,str1);
        tab_son[19+nb_classe]:= LoadAIFF(ADR(str));
        ReadInt(ent);
        i := 1;
        WHILE i<= ent DO
          ReadInt(tab_cadre_obl[nb_classe][i]);
          i := i+1;
        END;
        tab_cadre_obl[nb_classe][i] :=Ø;
        IF Done = TRUE  THEN nb_classe := nb_classe + 1;END;
      END;
      afficher_cartoon(afficher);
      CloseInput;
      nb_classe := nb_classe-1;
END saisie_parcours;

(*****************************************************************************)

PROCEDURE ecrire_liste_enonce;

VAR
   i: INTEGER;

BEGIN
      OpenOutputFile("user:liste_enonce");
      IF IoErr() # LONGINT (Ø)
         THEN ScreenToFront(scr^);
      END;
      i := 1;
      WHILE liste_enonce[i] # 999 DO
        afficher_cartoon(TRUE);
        WriteInt(liste_enonce[i],1);
        Write(" ");
        i := i+1;
      END;
      WriteInt(liste_enonce[i],1);
      afficher_cartoon(TRUE);
      CloseOutput;
END ecrire_liste_enonce;
(*****************************************************************************)

PROCEDURE lire_liste_enonce;

VAR
   i : INTEGER;
   afficher : BOOLEAN;

BEGIN
      OpenInputFile("user:liste_enonce");
      IF IoErr() # LONGINT (Ø)
         THEN ScreenToFront(scr^);
      END;
```

```
      i := 1;
      afficher := NOT entree;
      WHILE Done = TRUE DO
        afficher_cartoon (afficher);
        ReadInt(liste_enonce[i]);
        i := i+1;
      END;
      afficher_cartoon(afficher);
      CloseInput;
END lire_liste_enonce;
(*************************************************************************)

PROCEDURE reprise_parc(num_cadre,num_classe,
          nb_cadre_desig,nb_classe_desig:INTEGER);

BEGIN
      OpenOutputFile("user:reprise_parc");
      IF IoErr() # LONGINT (Ø)
         THEN ScreenToFront(scr^);
      END;
      WriteInt(num_cadre,1);
      Write(' ');
      WriteInt(num_classe,1);
      Write(' ');
      WriteInt(nb_cadre_desig,1);
      Write(' ');
      WriteInt(nb_classe_desig,1);
      CloseOutput;
END reprise_parc;

(*************************************************************************)

PROCEDURE lire_reprise_parc(VAR num_cadre,num_classe,
          nb_cadre_desig,nb_classe_desig:INTEGER);

BEGIN
      OpenInputFile("user:reprise_parc");
      IF IoErr() # LONGINT (Ø)
         THEN ScreenToFront(scr^);
      END;
      ReadInt(num_cadre);
      ReadInt(num_classe);
      ReadInt(nb_cadre_desig);
      ReadInt(nb_classe_desig);
      CloseInput;
END lire_reprise_parc;

(*************************************************************************)

PROCEDURE cherche_date(VAR jour, mois, annee : INTEGER);

VAR
   x : CARDINAL;
   y : INTEGER;
   date : DateStampRecord;
   trouve_an, trouve_mois : BOOLEAN;
   en : INTEGER;
   re,quatre,re2,an : REAL;
   nb_jours : ARRAY [Ø..11] OF CARDINAL;

BEGIN
      DateStamp(date);
      x:=SHORT(date.dsDays);
      y:=x;
      x:=x+1;
      annee:=78;
```

```
      trouve_an:=FALSE;
      nb_jours[0]:=31;
      nb_jours[2]:=31;
      nb_jours[3]:=30;
      nb_jours[4]:=31;
      nb_jours[5]:=30;
      nb_jours[6]:=31;
      nb_jours[7]:=31;
      nb_jours[8]:=30;
      nb_jours[9]:=31;
      nb_jours[10]:=30;
      nb_jours[11]:=31;
      WHILE NOT trouve_an DO
            an:=real(annee);
            quatre:=4.0;
            re:=an/quatre;
            en:=entier(re);
            re2:=real(en);
            IF re=re2
               THEN
                   IF x>366
                      THEN
                          x:=x-366;
                          annee:=annee+1;
                      ELSE
                          nb_jours[1]:=29;
                          trouve_an:=TRUE;
                   END;
               ELSE
                   IF x>365
                      THEN
                          x:=x-365;
                          annee:=annee+1;
                      ELSE
                          nb_jours[1]:=28;
                          trouve_an:=TRUE;
                   END;
            END;
      END;
      trouve_mois:=FALSE;
      mois:=0;
      WHILE NOT trouve_mois DO
            IF x>nb_jours[mois]
               THEN
                   x:=x-nb_jours[mois];
                   mois:=mois+1;
               ELSE
                   trouve_mois:=TRUE;
            END;
      END;
      jour:=x;
      mois:=mois+1;
END cherche_date;

(*****************************************************************************)

PROCEDURE verifiefich (ecran : ScreenPtr;nom : ARRAY OF CHAR) : BOOLEAN;

BEGIN
    OpenInputFile(nom);
    IF IoErr() # LONGINT(0) THEN ScreenToFront(scr^);END;
    IF Done
       THEN CloseInput;
            RETURN TRUE;
       ELSE RETURN FALSE;
    END;
```

```
END verifiefich;

(*****************************************************************************)

PROCEDURE lire_cadre_nom;

VAR
    i : INTEGER;
    aff : BOOLEAN;

BEGIN

    OpenInputFile('user:coord_cadre');
     IF IoErr() # LONGINT(Ø)
         THEN ScreenToFront(scr^);
     END;
     i:= 1;
     aff := NOT entree;
     WHILE Done  DO
           afficher_cartoon(aff);
           ReadInt(tab_cadre_nom[i].x);
           ReadInt(tab_cadre_nom[i].y);
           i:=i+1;

    END;
   afficher_cartoon(aff);
   CloseInput;

END lire_cadre_nom;

(*****************************************************************************)

PROCEDURE ecrire_cadre_nom;

VAR
    i : INTEGER;

BEGIN
  OpenOutputFile('user:coord_cadre');
  i:=1;
  WHILE i<= nb_enonce DO
    afficher_cartoon(TRUE);
    WriteInt(tab_cadre_nom[i].x,1);
    Write(" ");
    WriteInt(tab_cadre_nom[i].y,1);
    Write(" ");
    i:= i+1;
  END;
  afficher_cartoon(TRUE);
  CloseOutput;

END ecrire_cadre_nom;

(*****************************************************************************)

PROCEDURE init_icone;

VAR
   image_pointeur : ImageDescTablePtr;
   nb_image : CARDINAL;
   i : CARDINAL;

BEGIN
     image_pointeur:=FindImageTable(87654320H,nb_image);
     FOR i:=Ø TO nb_image-1 DO
         WITH tab_icone[i+1] DO
```

```
                LeftEdge:=Ø;
                TopEdge:=Ø;
                Width:=image_pointeur^[i].Width;
                Height:=image_pointeur^[i].Height;
                Depth:=image_pointeur^[i].Depth;
                ImageData:=image_pointeur^[i].Data;
                PlanePick:=BYTE(31);
                PlaneOnOff:=BYTE(31);
            END;
        END;

END init_icone;

(*****************************************************************************)

PROCEDURE init_cartoon;

VAR
    image_pointeur : ImageDescTablePtr;
    nb_image,i : CARDINAL;
    ok : BOOLEAN;
    k,j :INTEGER;

BEGIN
     image_pointeur:=FindImageTable(21212121H,nb_image);
     nb_cartoon := nb_image DIV 2;
     j:= Ø;
     FOR i:= 1 TO (nb_image DIV 2) DO
        k:= Ø;
        ok := FALSE;
        WHILE k <= 1 DO
           WITH tab_cartoon[ok][i] DO
                LeftEdge:=Ø;
                TopEdge:=Ø;
                 Width:=image_pointeur^[j].Width;
                 Height:=image_pointeur^[j].Height;
                 Depth:=image_pointeur^[j].Depth;
                 ImageData:=image_pointeur^[j].Data;
                 PlanePick:=BYTE(31);
                 PlaneOnOff:=BYTE(31);
           END;
           ok:= (NOT ok);
           k := k+1;
           j := j+1;
        END;
     END;
END init_cartoon;

(*****************************************************************************)

PROCEDURE calcul_duree;

VAR date :DateStampRecord;
    min :LONGCARD;

BEGIN
      DateStamp(date);
      min := date.dsMinute;
      duree := min-minute;

END calcul_duree;

(*****************************************************************************)


PROCEDURE init_anim;
```

```
VAR
   image_pointeur : ImageDescTablePtr;
   nb_image : CARDINAL;
   i : CARDINAL;

BEGIN
     image_pointeur:=FindImageTable(87654321H,nb_image);
     FOR i:=Ø TO nb_image-1 DO
         WITH tab_anim[i+1] DO
              LeftEdge:=Ø;
              TopEdge:=Ø;
              Width:=image_pointeur^[i].Width;
              Height:=image_pointeur^[i].Height;
              Depth:=image_pointeur^[i].Depth;
              ImageData:=image_pointeur^[i].Data;
              PlanePick:=BYTE(31);
              PlaneOnOff:=BYTE(31);
         END;
    END;

END init_anim;

(*****************************************************************************)

PROCEDURE rech_num_der_vers (VAR num : INTEGER);
(*****************************************************************************
*
* IN : /
*
* OUT: num : n° de la dernière version.
*
* BUT : Initialise num à la dernière version des traces trouvée
*       dans <nom de l'utilisateur>.dv.
*
*****************************************************************************)

VAR nom_fich : string4Ø;

BEGIN
    CopyString(nom_fich,'user:');
    ConcatString(nom_fich,nom_user);
    ConcatString(nom_fich,'.version');
    OpenInputFile (nom_fich);
    IF IoErr() # LONGINT (Ø)
    THEN ScreenToFront (scr^);
    END;
    ReadInt(num);
    CloseInput;

END rech_num_der_vers;

(*****************************************************************************)

PROCEDURE passer_ligne_2;

CONST
     fin_ligne = 12C;
VAR
   car : CHAR;

BEGIN
     FileSystem.ReadChar(fichier_pres,car);
     WHILE car # fin_ligne DO
           FileSystem.ReadChar(fichier_pres,car);
     END;
```

```
END passer_ligne_2;

(*****************************************************************************)

PROCEDURE re_init_tab_trace;

VAR ch : CHAR;
    num : INTEGER;
    nom_fichier : string4Ø;
    num_str : ARRAY[Ø..2] OF CHAR;
    num1 : LONGINT;

BEGIN
    pos_tab_trace := 1;
    rech_num_der_vers (num);
    num1 := num;
    ConvNumberToString(num_str,num1,FALSE,1Ø,2,"Ø");
    CopyString(nom_fichier,'user:');
    ConcatString(nom_fichier,nom_user);
    ConcatString(nom_fichier,num_str);
    ConcatString(nom_fichier,'.trace');
    OpenInputFile(nom_fichier);
    IF IoErr() # LONGINT (Ø)
    THEN ScreenToFront (scr^);
    END;
    Read(ch);
    WHILE Done DO
          WHILE ch # EOL DO
                ConcatString(tab_trace[pos_tab_trace],ch);
                Read(ch);
          END;
          pos_tab_trace := pos_tab_trace + 1;
          Read(ch);
    END;
    CloseInput;
END re_init_tab_trace;

(*****************************************************************************)

PROCEDURE fichier_duree_trace;

VAR
   ligne : ARRAY [Ø..99] OF CHAR;
   duree_str : ARRAY [Ø..5] OF CHAR;
   car : CHAR;
   compteur : INTEGER;
   jour, mois, annee : INTEGER;
   jour_str : ARRAY [Ø..2] OF CHAR;
   mois_str : ARRAY [Ø..2] OF CHAR;
   annee_str : ARRAY [Ø..2] OF CHAR;
   tampon : INTEGER;
   int_str : ARRAY [Ø..5] OF CHAR;

BEGIN
     compteur:=Ø;
     WHILE compteur#99 DO
           ligne[compteur]:=ØC;
           compteur:=compteur+1;
     END;
     CopyString(ligne,tab_mess_trace[37]);
     cherche_date(jour,mois,annee);
     ConvNumberToString(jour_str,LONGINT(jour),FALSE,1Ø,2,"Ø");
     ConvNumberToString(mois_str,LONGINT(mois),FALSE,1Ø,2,"Ø");
     ConvNumberToString(annee_str,LONGINT(annee),FALSE,1Ø,2,"Ø");
     ConcatString(ligne,jour_str);
     ConcatString(ligne,"-");
```

```
        ConcatString(ligne,mois_str);
        ConcatString(ligne,"-");
        ConcatString(ligne,annee_str);
        ConcatString(ligne,"      ");
        ConvNumberToString(duree_str,LONGINT(duree),FALSE,10,5,"0");
        ConcatString(ligne,tab_mess_trace[44]);
        ConcatString(ligne,duree_str);
        ConcatString(ligne,tab_mess_trace[45]);
        (* Ouverture fichier *)
        OpenInputFile('user:duree.trace');
        IF Done
           THEN
               ReadInt(tampon);
               ConvNumberToString(int_str,LONGINT(tampon+1),FALSE,10,4,"0");
               CloseInput;
               FileSystem.Lookup(fichier_pres,"user:duree.trace",FALSE);
               compteur:=0;
               WHILE compteur<=3 DO
                     FileSystem.WriteChar(fichier_pres,int_str[compteur]);
                     compteur:=compteur+1;
               END;
               FileSystem.Close(fichier_pres);
               FileSystem.Lookup(fichier_pres,"user:duree.trace",FALSE);
               passer_ligne_2;
               compteur:=1;
               WHILE compteur<=tampon DO
                     passer_ligne_2;
                     compteur:=compteur+1;
               END;
               compteur:=0;
               WHILE (ligne[compteur]#0C) AND (compteur<100) DO
                     FileSystem.WriteChar(fichier_pres,ligne[compteur]);
                     compteur:=compteur+1;
               END;
               FileSystem.WriteChar(fichier_pres,12C);
               FileSystem.Close(fichier_pres); (* Fermeture fichier *)
           ELSE
               OpenOutputFile('user:duree.trace');
               WriteString('0001');
               WriteLn;
               WriteString(ligne);
               WriteLn;
               CloseOutput;
        END;
END fichier_duree_trace;

(*****************************************************************************)

PROCEDURE presentation_trace(num_trace : ARRAY OF CHAR);
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Initialiser la présentation du fichier trace dans le tableau
*       tab_trace, l'entête et les paramètres.
*
*****************************************************************************)

VAR fichier : FileSystem.File;
    compteur : INTEGER;
    car : CHAR;
    efface : BOOLEAN;
    ligne : string80;
    str : ARRAY [0..2] OF CHAR;
```

```
BEGIN
    WriteString(tab_mess_trace[30]);
    WriteLn;
    WriteString(tab_mess_trace[31]);
    WriteLn;
    WriteString(tab_mess_trace[32]);
    WriteLn;
    WriteString(tab_mess_trace[33]);
    WriteLn;
    WriteString(tab_mess_trace[34]);
    WriteLn;
    WriteString(tab_mess_trace[35]);
    WriteLn;
    WriteString(tab_mess_trace[36]);
    WriteString(nom_user);
    WriteLn;
    WriteString(tab_mess_trace[38]);
    WriteString(num_trace);
    WriteLn;
    IF (tab_exercice[num_ex].type_ex = "D") OR
       (tab_exercice[num_ex].type_ex = "d")
    THEN WriteString(tab_mess_trace[40]);
    ELSE WriteString(tab_mess_trace[41]);
    END;
    WriteLn;
    WriteString(tab_mess_trace[30]);
    WriteLn;
    FileSystem.Lookup(fichier,'user:duree.trace',FALSE);
    afficher_cartoon(TRUE);
    compteur := 0;
    WHILE compteur < 5 DO
          FileSystem.ReadChar(fichier,car);
          compteur := compteur +1;
    END;
    FileSystem.ReadChar(fichier,car);
    Write(car);
    WHILE (NOT fichier.eof) DO
          FileSystem.ReadChar(fichier,car);
          Write(car);
    END;
    afficher_cartoon(TRUE);
    FileSystem.Close(fichier);
    WriteLn;
    efface := DeleteFile(ADR('user:duree.trace'));
    WriteString(tab_mess_trace[30]);
    WriteLn;
    WriteString(tab_mess_trace[45 + type_renfor]);
    WriteLn;
    IF (type_saisie = "K") OR  (type_saisie = "k")
    THEN WriteString(tab_mess_trace[49]);
    ELSE WriteString(tab_mess_trace[50]);
    END;
    WriteLn;
    WriteString(tab_mess_trace[30]);
    WriteLn;
    afficher_cartoon(TRUE);
    IF (tab_exercice[num_ex].type_ex = "D") OR
       (tab_exercice[num_ex].type_ex = "d")
    THEN
        (* valeur des param *)
        WriteString(tab_mess_trace[39]);
        WriteLn;
        (* param interaction *)
         WriteString(tab_mess_trace[50 + tab_param[1]]);
         WriteLn;
```

```
      (* param enonce vocal *)
       CopyString(ligne,tab_mess_trace[55]);
       ConcatString(ligne,tab_mess_trace[70 + tab_param[3]]);
       WriteString(ligne);
       WriteLn;
       afficher_cartoon(TRUE);
       (* nb d'intération avant reproposition *)
       ConvNumberToString (str,LONGINT(tab_param[4]),FALSE,10,2,"0");
       CopyString(ligne,tab_mess_trace[56]);
       ConcatString(ligne,str);
       WriteString(ligne);
       WriteLn;
       (* ordre d'énumération *)
       CopyString(ligne,tab_mess_trace[58]);
       ConcatString(ligne,tab_mess_trace[70 + tab_param[2]]);
       WriteString(ligne);
       WriteLn;
       (* silhouette vide *)
       CopyString(ligne,tab_mess_trace[60]);
       ConcatString(ligne,tab_mess_trace[70 + tab_param[5]]);
       WriteString(ligne);
       WriteLn;
       (* animation*)
       CopyString(ligne,tab_mess_trace[66]);
       ConcatString(ligne,tab_mess_trace[70 + tab_param[6]]);
       WriteString(ligne);
       WriteLn;
       (* param détection couleur/cadre*)
       WriteString(tab_mess_trace[66 + tab_param[7]]);
       WriteLn;
       (* param par nom *)
       WriteString(tab_mess_trace[68 + tab_param[8]]);
       WriteLn;
ELSE
       CopyString(ligne,tab_mess_trace[73]);
       ConcatString(ligne,nom_symbole);
       WriteString(ligne);
       WriteLn;
       CopyString(ligne,tab_mess_trace[74]);
       ConvNumberToString(str,LONGINT(nb_enonce),FALSE,10,2,"0");
       ConcatString(ligne,str);
       WriteString(ligne);
       WriteLn;
       CopyString(ligne,tab_mess_trace[75]);
       ConvNumberToString(str,LONGINT(nb_classe),FALSE,10,2,"0");
       ConcatString(ligne,str);
       WriteString(ligne);
       WriteLn;
       CopyString(ligne,tab_mess_trace[76]);
       ConcatString(ligne,tab_nom_cadre_obl[1]);
       WriteString(ligne);
       WriteLn;
       compteur := 2;
       WHILE compteur<nb_classe DO
          WriteString(tab_nom_cadre_obl[compteur]);
          WriteLn;
          compteur := compteur + 1;
       END;
       CopyString(ligne,tab_mess_trace[77]);
       ConcatString(ligne,tab_nom_cadre_obl[nb_classe]);
       WriteString(ligne);
       WriteLn;
       WriteString(tab_mess_trace[30]);
       WriteLn;
       WriteString(tab_mess_trace[39]);
       WriteLn;
```

```
        (* parm type d'interaction *)
        WriteString(tab_mess_trace[5Ø + tab_param[1]]);
        WriteLn;
        (* param enonce vocal*)
        CopyString(ligne,tab_mess_trace[55]);
        ConcatString(ligne,tab_mess_trace[7Ø+ tab_param[6]]);
        WriteString(ligne);
        WriteLn;
        (* param parcours début *)
        CopyString(ligne,tab_mess_trace[57]);
        ConcatString(ligne,tab_mess_trace[7Ø+ tab_param[2]]);
        WriteString(ligne);
        WriteLn;
        (* param monter début et fin *)
        CopyString(ligne,tab_mess_trace[59]);
        ConcatString(ligne,tab_mess_trace[7Ø+ tab_param[3]]);
        WriteString(ligne);
        WriteLn;
        (* param parcours en fin *)
        CopyString(ligne,tab_mess_trace[61]);
        ConcatString(ligne,tab_mess_trace[7Ø+ tab_param[5]]);
        WriteString(ligne);
        WriteLn;
        (* param feed_back partiel*)
        WriteString(tab_mess_trace[62 + tab_param[4]]);
        WriteLn;
    END;
    WriteString(tab_mess_trace[3Ø]);
    WriteLn;
END presentation_trace;

(*****************************************************************************)

PROCEDURE ecrire_fich_trace(ex_stop : BOOLEAN);

VAR i,num : INTEGER;
    nom_fich : string4Ø;
    num_str : ARRAY[Ø..2] OF CHAR;
    num1 : LONGINT;

BEGIN

   calcul_duree;
   afficher_cartoon(TRUE);
   fichier_duree_trace;
   rech_num_der_vers(num);
   num1 := num;
   ConvNumberToString(num_str,num1,FALSE,1Ø,2,"Ø");
   CopyString(nom_fich,'user:');
   ConcatString(nom_fich,nom_user);
   ConcatString(nom_fich,num_str);
   ConcatString(nom_fich,'.trace');
   OpenOutputFile(nom_fich);
   afficher_cartoon(TRUE);
   IF IoErr() # LONGINT (Ø)
   THEN ScreenToFront (scr^);
   END;
   IF ex_stop = FALSE
   THEN presentation_trace(num_str);
   END;
   i := 1;
   WHILE i<= pos_tab_trace - 1 DO
         WriteString(tab_trace[i]);
         WriteLn;
         i := i + 1;
         IF (i MOD 5Ø) = Ø THEN afficher_cartoon(TRUE);
```

```
            END;
        END;
        WriteLn;
        CloseOutput;
        IF (NOT ex_stop)
        THEN
            num := num + 1;
            CopyString(nom_fich,'user:');
            ConcatString(nom_fich,nom_user);
            ConcatString(nom_fich,'.version');
            afficher_cartoon(TRUE);
            OpenOutputFile(nom_fich);
            IF IoErr() # LONGINT (Ø)
            THEN ScreenToFront (scr^);
            END;
            WriteInt(num,3);
            CloseOutput;
        END;
    END ecrire_fich_trace;

    END base_donnees.
```

```
IMPLEMENTATION MODULE graphique;

FROM var_globale IMPORT scr,rp,point,symbole,vp,nb_enonce,graphique_support,
     tab_cadre_desig,tab_cadre_parc,tab_param,tab_couleur,tab_son,
     tab_mess_trace,
     tab_cadre_nom,LONG_CADRE,HAUT_CADRE,liste_enonce,tab_mess_text,string80,
     string40,tab_exercice,num_ex,coul_parc,tab_cartoon,num_cartoon,cartoon,
     tick_cartoon,w,win,tab_anim;
FROM trt_donnees IMPORT calcul_centre_cadre,indice,pluriel,ecrire_trace;
FROM son IMPORT renforcement_positif,renforcement_negatif,dire;

FROM Drawing IMPORT ReadPixel,WritePixel,SetAPen,RectFill,Move,Draw;
FROM SYSTEM IMPORT ADR,BYTE;
FROM Views IMPORT LoadRGB4;
FROM Rasters IMPORT RastPortPtr;
FROM Intuition IMPORT DrawImage,Image;
FROM AmigaDosProcess IMPORT Delay;
FROM AmigaDos IMPORT
     DateStamp,DateStampRecord;
FROM SimpleGels IMPORT CreateBob,CreateGelsInfo,DeleteBob,DeleteGelsInfo;
FROM Gels IMPORT AddBob,SortGList,DrawGList,RemBob,GelsInfoPtr,BobPtr;
FROM Strings IMPORT ExtractSubString,StringLength,CopyString,ConcatString;
FROM Text IMPORT TextLength,Text;
FROM InOut IMPORT WriteString;

VAR trace : string80;


(*****************************************************************************)
PROCEDURE init_couleur;
(*****************************************************************************)

VAR
   CMAP : ARRAY [0..31] OF CARDINAL;

BEGIN
     CMAP[0]:=007CH;
     CMAP[1]:=0FFFH;
     CMAP[2]:=0E00H;
     CMAP[3]:=0A00H;
     CMAP[4]:=0E83H;
     CMAP[5]:=0FE0H;
     CMAP[6]:=08F0H;
     CMAP[7]:=0080H;
     CMAP[8]:=00B6H;
     CMAP[9]:=00DDH;
     CMAP[10]:=00AFH;
     CMAP[11]:=0000H;
     CMAP[12]:=000FH;
     CMAP[13]:=070FH;
     CMAP[14]:=0C0EH;
     CMAP[15]:=0C08H;
     CMAP[16]:=0620H;
     CMAP[17]:=0E84H;
     CMAP[18]:=0A52H;
     CMAP[19]:=0FCAH;
     CMAP[20]:=0333H;
     CMAP[21]:=0444H;
     CMAP[22]:=0555H;
     CMAP[23]:=0666H;
     CMAP[24]:=0777H;
     CMAP[25]:=0888H;
     CMAP[26]:=0999H;
     CMAP[27]:=0AAAH;
     CMAP[28]:=0BBBH;
     CMAP[29]:=0CCCH;
```

```
    CMAP[30]:=0DDDH;
    CMAP[31]:=0EEEH;
    LoadRGB4(scr^.ViewPort,ADR(CMAP),32);
END init_couleur;


(*****************************************************************************)

PROCEDURE deplacement_symbole(num_cadre: INTEGER);

VAR ptg,ptd,a1,a2,b1,b2 : point;
    couleur : INTEGER;

BEGIN
  IF num_cadre = 1
  THEN
     calcul_centre_cadre(num_cadre,b1,b2);
     aff_image(b1.x,b1.y,symbole);
     aff_image(b2.x,b2.y,symbole);
 ELSE
     calcul_centre_cadre(num_cadre,b1,b2);
     calcul_centre_cadre((num_cadre - 1) ,a1,a2);
     ptg := a1;
     ptd := a2;
     couleur := ReadPixel(rp^,a1.x,a1.y);
     SetAPen(rp^,couleur);

     WHILE ( ptg.x <> b1.x) OR (ptg.y <> b1.y) OR
           (ptd.x <> b2.x) OR (ptd.y <> b2.y) DO
          IF (ptg.x <> b1.x) OR ( ptg.y <> b1.y) THEN
                RectFill(rp^,ptg.x,ptg.y,
                         ptg.x + symbole.Width,ptg.y + symbole.Height);

                IF ptg.x <> b1.x
                THEN     IF ptg.x < b1.x
                         THEN ptg.x := ptg.x + 1
                         ELSE ptg.x := ptg.x - 1;
                         END;
                END;

                IF ptg.y <> b1.y
                THEN     IF ptg.y < b1.y
                         THEN ptg.y := ptg.y + 1
                         ELSE ptg.y := ptg.y - 1;
                         END;
                END;
           END;

           (*DrawImage(rp^,symbole,ptg.x,ptg.y);*)
           aff_image(ptg.x,ptg.y,symbole);
           Delay (5);

           RectFill(rp^,ptd.x,ptd.y,ptd.x + symbole.Width,
                    ptd.y + symbole.Height);

           IF (ptd.x <> b2.x) OR (ptd.y <> b2.y) THEN
                IF ptd.x <> b2.x
                THEN     IF ptd.x < b2.x
                         THEN ptd.x := ptd.x + 1
                         ELSE ptd.x := ptd.x - 1;
                         END;
                END;
                IF ptd.y <> b2.y
                THEN     IF ptd.y < b2.y
                         THEN ptd.y := ptd.y + 1
```

```
                        ELSE ptd.y := ptd.y -1;
                        END;
                END;
            END;
            (*DrawImage(rp^,symbole,ptd.x,ptd.y);*)
            aff_image(ptd.x,ptd.y,symbole);
            Delay (5);

        END;
  END;
END deplacement_symbole;

(*****************************************************************************)

PROCEDURE renforcement_parcours(num_cadre_desig, num_dern_cadre: INTEGER);

VAR
   num_cadre_courant : INTEGER;

BEGIN
    num_cadre_courant := num_dern_cadre + 1;
    WHILE num_cadre_courant <= num_cadre_desig DO
        deplacement_symbole(num_cadre_courant);
        num_cadre_courant := num_cadre_courant + 1;
    END;
END renforcement_parcours;

(*****************************************************************************)

PROCEDURE aff_image(x,y :INTEGER; image : Image);

VAR
 g: GelsInfoPtr;
 dessin : BobPtr;

BEGIN
  g:= CreateGelsInfo(rp^);
  dessin:= CreateBob(x,y,image.Width,image.Height,image.Depth,image.ImageData,
                     BYTE(31),BYTE(31));
  AddBob(dessin^,rp^);
  SortGList(rp^);
  DrawGList(rp^,vp^);
  RemBob(dessin^);
  DeleteBob(dessin^);
  DeleteGelsInfo(g^);
END aff_image;

(*****************************************************************************)

PROCEDURE aff_brush_designation(num_cadre : INTEGER);

 VAR x_sup,y_sup,i : INTEGER;

 BEGIN
      i := 1;
      WHILE i <= nb_enonce DO
          IF i # num_cadre
            THEN
              x_sup := tab_cadre_desig[i].x + graphique_support.LeftEdge;
              y_sup := tab_cadre_desig[i].y + graphique_support.TopEdge;
              aff_image(x_sup,y_sup,tab_cadre_desig[i].brush);
            END;
          i := i+1;
      END;
 END aff_brush_designation;
```

```
(*****************************************************************************)

PROCEDURE aff_graphique(centre : BOOLEAN);

VAR x,y : INTEGER;

  BEGIN
       IF centre
       THEN
          x := 160-(graphique_support.Width DIV 2);
          y := 128-(graphique_support.Height DIV 2)-17;
       ELSE
          x := 240-(graphique_support.Width DIV 2);
          y := 128-(graphique_support.Height DIV 2)-17;
       END;
          DrawImage(rp^,graphique_support,x,y);
          graphique_support.LeftEdge := x;
          graphique_support.TopEdge := y;

  END aff_graphique;

(*****************************************************************************)

PROCEDURE coloration_zone(x,y,long,haut,couleur1,couleur2:INTEGER);

VAR x1,y1,color,i : INTEGER;

BEGIN
    i := 1;
    y1 := y;
    WHILE y1 <= y + haut      DO

       x1:=x;
       WHILE x1<=x +long  DO

             color := ReadPixel(rp^,x1,y1);
             IF couleur1 = 99
               THEN IF color <> 17
                       THEN SetAPen(rp^,couleur2);
                            i := WritePixel(rp^,x1,y1);
                    END;
               ELSE  IF color = couleur1
                       THEN
                           SetAPen(rp^,couleur2);

                           i := WritePixel(rp^,x1,y1);
                       END;
             END;
             x1 := x1 + 1;
       END;
      y1:=y1 +1;
    END;
END coloration_zone;

(*****************************************************************************)

PROCEDURE noircit_organe(tout:BOOLEAN);

VAR
   i,j: INTEGER;

BEGIN

 IF NOT tout
```

```
 THEN i:= 1;
      WHILE liste_enonce[i] <>999 DO
        aff_image(tab_cadre_desig[liste_enonce[i]].x +
                  graphique_support.LeftEdge,
                  tab_cadre_desig[liste_enonce[i]].y +
                  graphique_support.TopEdge,
                  tab_cadre_desig[liste_enonce[i]].brush);
        coloration_zone(tab_cadre_desig[liste_enonce[i]].x +
                        graphique_support.LeftEdge,
                        tab_cadre_desig[liste_enonce[i]].y +
                        graphique_support.TopEdge,
                        tab_cadre_desig[liste_enonce[i]].brush.Width,
                        tab_cadre_desig[liste_enonce[i]].brush.Height,
                        99,24(*gris*));
        i:= i+1;
      END;
      i:=1;
      WHILE i<= nb_enonce DO
        indice(i,j);
        IF j = Ø
        THEN aff_image(tab_cadre_desig[i].x +
                       graphique_support.LeftEdge,
                       tab_cadre_desig[i].y +
                       graphique_support.TopEdge,
                       tab_cadre_desig[i].brush);
        END;
        i:= i+1;
      END;
 ELSE
      i:= 1;
      aff_brush_designation(99);
      WHILE  i<= nb_enonce DO
        coloration_zone(tab_cadre_desig[i].x +
                        graphique_support.LeftEdge,
                        tab_cadre_desig[i].y +
                        graphique_support.TopEdge,
                        tab_cadre_desig[i].brush.Width,
                        tab_cadre_desig[i].brush.Height,
                        99,24(*gris*));
             i := i + 1;
       END;
 END;
END noircit_organe;

(*****************************************************************************)

PROCEDURE clignoter_zone(num_cadre :INTEGER);

VAR i,xsup,ysup:INTEGER;

  BEGIN

       i := 1;
       WHILE i < 4 DO
          IF tab_exercice[num_ex].anim =1
          THEN SetAPen(rp^,Ø);
          ELSE SetAPen(rp^,17);
          END;
          IF tab_param[7] = 2  (* pas par couleur *)
          THEN RectFill(rp^,
                  tab_cadre_desig[num_cadre].x + graphique_support.LeftEdge,
                  tab_cadre_desig[num_cadre].y + graphique_support.TopEdge,
                  tab_cadre_desig[num_cadre].x + graphique_support.LeftEdge +
                  tab_cadre_desig[num_cadre].brush.Width,
                  tab_cadre_desig[num_cadre].y + graphique_support.TopEdge +
                  tab_cadre_desig[num_cadre].brush.Height);
```

```
            ELSE   coloration_zone(
                    tab_cadre_desig[num_cadre].x + graphique_support.LeftEdge,
                    tab_cadre_desig[num_cadre].y + graphique_support.TopEdge,
                    tab_cadre_desig[num_cadre].brush.Width,
                    tab_cadre_desig[num_cadre].brush.Height,
                    tab_couleur[num_cadre],17);
            END;
            IF tab_exercice[num_ex].anim =1
            THEN
            ELSE aff_brush_designation(num_cadre);
            END;
            xsup:= tab_cadre_desig[num_cadre].x + graphique_support.LeftEdge;
            ysup:= tab_cadre_desig[num_cadre].y + graphique_support.TopEdge;
            Delay(40);
            aff_image(xsup,ysup,tab_cadre_desig[num_cadre].brush);
            Delay(40);
            i := i+1;

  END;
END clignoter_zone;

(*****************************************************************************)

PROCEDURE effacer_cadre(x,y,haut,long:INTEGER);

BEGIN
        SetAPen(rp^,0);
        RectFill(rp^,x,y,x + long,y + haut);

END effacer_cadre;

(*****************************************************************************)

PROCEDURE animation(VAR string : ARRAY  OF CHAR) ;

VAR x,y,i:INTEGER;

BEGIN
  effacer_cadre(0,0,220,319);
  effacer_cadre(166,223,29,148);
  CASE string[0] OF
    'd','D':i:= 1;
            WHILE i<=3 DO
               x := 160-(tab_anim[9].Width DIV 2);
               y := 128-(tab_anim[9].Height DIV 2)-17;
               aff_image(x,y,tab_anim[9]);
               Delay(30);
               effacer_cadre(x,y,tab_anim[9].Height,tab_anim[9].Width);
               x := 160-(tab_anim[10].Width DIV 2);
               y := 128-(tab_anim[10].Height DIV 2)-17;
               aff_image(x,y,tab_anim[10]);
               Delay(30);
               effacer_cadre(x,y,tab_anim[10].Height,tab_anim[10].Width);
               i:= i+1;
            END;|
    'r','R':i:= 1;
            WHILE i<=3 DO
               x := 160-(tab_anim[3].Width DIV 2);
               y := 128-(tab_anim[3].Height DIV 2)-17;
               aff_image(x,y,tab_anim[3]);
               Delay(30);
               effacer_cadre(x,y,tab_anim[3].Height,tab_anim[3].Width);
               x := 160-(tab_anim[4].Width DIV 2);
               y := 128-(tab_anim[4].Height DIV 2)-17;
               aff_image(x,y,tab_anim[4]);
```

```
                Delay(30);
                effacer_cadre(x,y,tab_anim[4].Height,tab_anim[4].Width);
                i:= i+1;
             END;|
    'e','E':i:= 1;
             WHILE i<=3 DO
                x := 160-(tab_anim[7].Width DIV 2);
                y := 128-(tab_anim[7].Height DIV 2)-17;
                aff_image(x,y,tab_anim[7]);
                Delay(30);
                effacer_cadre(x,y,tab_anim[7].Height,tab_anim[7].Width);
                x := 160-(tab_anim[8].Width DIV 2);
                y := 128-(tab_anim[8].Height DIV 2)-17;
                aff_image(x,y,tab_anim[8]);
                Delay(30);
                effacer_cadre(x,y,tab_anim[8].Height,tab_anim[8].Width);
                i:= i+1;
             END;|
    'm','M':i:= 1;
             WHILE i<=3 DO
                x := 160-(tab_anim[5].Width DIV 2);
                y := 128-(tab_anim[5].Height DIV 2)-17;
                aff_image(x,y,tab_anim[5]);
                Delay(30);
                effacer_cadre(x,y,tab_anim[5].Height,tab_anim[5].Width);
                x := 160-(tab_anim[6].Width DIV 2);
                y := 128-(tab_anim[6].Height DIV 2)-17;
                aff_image(x,y,tab_anim[6]);
                Delay(30);
                effacer_cadre(x,y,tab_anim[6].Height,tab_anim[6].Width);
                i:= i+1;
             END;|
    'c','C':i:= 1;
             WHILE i<=3 DO
                x := 160-(tab_anim[1].Width DIV 2);
                y := 128-(tab_anim[1].Height DIV 2)-17;
                aff_image(x,y,tab_anim[1]);
                Delay(30);
                effacer_cadre(x,y,tab_anim[1].Height,tab_anim[1].Width);
                x := 160-(tab_anim[2].Width DIV 2);
                y := 128-(tab_anim[2].Height DIV 2)-17;
                aff_image(x,y,tab_anim[2]);
                Delay(30);
                effacer_cadre(x,y,tab_anim[2].Height,tab_anim[2].Width);
                i:= i+1;
             END;|
  END;  (* réaffichage des systèmes et du cadre *)
  effacer_cadre(0,0,220,319);
  dessine_rect(0,0,319,220);
  aff_brush_designation(99);


END animation;

(*****************************************************************************)

PROCEDURE feedback_positif(num_zone_desig:INTEGER);

VAR
  str : string40;
  longstr : CARDINAL;

BEGIN
   renforcement_positif;
   IF (tab_param[1] = 3) OR (tab_param[5] =1)    (* pp ou vide *)
   THEN
```

```
IF tab_param[6]=1       (* par nom *)
THEN
   dire(15);    (* tu as montré *)
   dire(29+num_zone_desig);
   effacer_cadre(166,223,29,148);  (* interieur cadre *)
   ecrire(tab_mess_text[10],1);    (* tu as montré *)
   ecrire(tab_cadre_desig[num_zone_desig].nom_cadre,2);
   coloration_zone(tab_cadre_nom[num_zone_desig].x+1,
                   tab_cadre_nom[num_zone_desig].y+1,
                   LONG_CADRE-1,HAUT_CADRE-1,24,2);
                                   (* gris en rouge *)
   clignoter_zone(num_zone_desig);

   (* TRACE *)
   CopyString(trace,tab_mess_trace[14]);
   ConcatString(trace,tab_mess_trace[19]);
   ConcatString(trace,tab_cadre_desig[num_zone_desig].nom_cadre);
   ecrire_trace(trace);
   (* FIN TRACE *)
ELSE
   (* ecrire(tab_cadre_nom[num_zone desig].nom_cadre); *)
   dire(15);
   dire(29+num_zone_desig);
   effacer_cadre(166,223,29,148);  (* interieur cadre *)
   ecrire(tab_mess_text[10],1);
   ecrire(tab_cadre_desig[num_zone_desig].nom_cadre,2);
   aff_image(tab_cadre_desig[num_zone_desig].x +graphique_support.LeftEdge,
             tab_cadre_desig[num_zone_desig].y +graphique_support.TopEdge,
             tab_cadre_desig[num_zone_desig].brush);
   Delay(100);
   IF tab_param[5] = 1       (* vide *)
   THEN Delay(100);
        SetAPen(rp^,17);
        RectFill(rp^,tab_cadre_desig[num_zone_desig].x +
                     graphique_support. LeftEdge,
                     tab_cadre_desig[num_zone_desig].y +
                     graphique_support. TopEdge,
                     tab_cadre_desig[num_zone_desig].x +
                     graphique_support. LeftEdge +
                     tab_cadre_desig[num_zone_desig].brush.Width,
                     tab_cadre_desig[num_zone_desig].y +
                     graphique_support. TopEdge +
                     tab_cadre_desig[num_zone_desig].brush.Height);
        (* TRACE *)
        CopyString(trace,tab_mess_trace[14]);
        IF pluriel(tab_cadre_desig[num_zone_desig].nom_cadre)
        THEN ConcatString(trace,tab_mess_trace[18])
        ELSE ConcatString(trace,tab_mess_trace[17])
        END;
        ConcatString(trace,tab_cadre_desig[num_zone_desig].nom_cadre);
        ecrire_trace(trace);
        (* FIN TRACE *)

    ELSE   (* TRACE *)
           CopyString(trace,tab_mess_trace[14]);
           ConcatString(trace,tab_cadre_desig[num_zone_desig].nom_cadre);
           ecrire_trace(trace);
           (* FIN TRACE *)
    END;
END;
ELSE  longstr:=StringLength(tab_cadre_desig[num_zone_desig].nom_cadre);
      ExtractSubString(str,tab_cadre_desig[num_zone_desig].nom_cadre,
                           longstr-2,1);
      IF tab_param[6] = 1   (* animation *)
      THEN
           animation(str);
```

```
          END;

          (* TRACE *)
          CopyString(trace,tab_mess_trace[14]);
          IF tab_exercice[num_ex].anim = 1
          THEN ConcatString(trace,tab_mess_trace[2Ø]);
               ExtractSubString(str,tab_cadre_desig[num_zone_desig].nom_cadre,
                                Ø,longstr-1);
               ConcatString(trace,str);
          ELSE IF tab_param[8]=1
               THEN ConcatString(trace,tab_mess_trace[19]);
               END;
               ConcatString(trace,tab_cadre_desig[num_zone_desig].nom_cadre);


          END;
          ecrire_trace(trace);
          (* FIN TRACE *)
     END;
END feedback_positif;

(*****************************************************************************)

PROCEDURE feedback_negatif(num_enonce,num_zone_desig:INTEGER);

VAR
  str : string4Ø;
  longstr : CARDINAL;
  i : INTEGER;

BEGIN
   renforcement_negatif;
   IF tab_param[1] <> 3              (*  pas pseudo_passif *)
   THEN IF tab_param[5] = 1              (* silhouette vide *)
        THEN
             effacer_cadre(166,223,29,148);  (* interieur cadre *)

             IF pluriel(tab_cadre_desig[num_zone_desig].nom_cadre)
             THEN
               ecrire(tab_mess_text[1Ø],1);  (* tu as montré où sont *)
               ecrire(tab_mess_text[21],2);

               (* TRACE*)
               CopyString(trace,tab_mess_trace[14]);
               ConcatString(trace,tab_mess_trace[18]);
               ConcatString(trace,tab_cadre_desig[num_zone_desig].nom_cadre);
               ecrire_trace (trace);
               (* FIN TRACE *)

             ELSE
               ecrire(tab_mess_text[1Ø],1);  (* tu as montré où est *)
               ecrire(tab_mess_text[17],2);

               (* TRACE*)
               CopyString(trace,tab_mess_trace[14]);
               ConcatString(trace,tab_mess_trace[17]);
               ConcatString(trace,tab_cadre_desig[num_zone_desig].nom_cadre);
               ecrire_trace (trace);
               (* FIN TRACE *)
             END;

             ecrire(tab_cadre_desig[num_zone_desig].nom_cadre,3);
             IF tab_param[3] = 1
             THEN IF pluriel(tab_cadre_desig[num_enonce].nom_cadre)
                  THEN  dire(15);
                        dire(18);
```

```
            ELSE  dire(15);
                  dire(17);
            END;
            dire(num_zone_desig+19)
        END;
        i := 1;
        WHILE i < 4 DO
          aff_image(
           tab_cadre_desig[num_zone_desig].x + graphique_support.LeftEdge,
           tab_cadre_desig[num_zone_desig].y + graphique_support.TopEdge,
           tab_cadre_desig[num_zone_desig].brush);
          Delay(40);
          SetAPen(rp^,17);
          RectFill(rp^,
           tab_cadre_desig[num_zone_desig].x + graphique_support.LeftEdge,
           tab_cadre_desig[num_zone_desig].y + graphique_support.TopEdge,
           tab_cadre_desig[num_zone_desig].x + graphique_support.LeftEdge+
           tab_cadre_desig[num_zone_desig].brush.Width,
           tab_cadre_desig[num_zone_desig].y + graphique_support.TopEdge+
           tab_cadre_desig[num_zone_desig].brush.Height);
          Delay(40);
          i := i+1;
        END;

        (* trt de la zone_enoncee *)
        effacer_cadre(166,223,29,148);  (* interieur cadre *)
        IF pluriel(tab_cadre_desig[num_enonce].nom_cadre)
        THEN
          ecrire(tab_mess_text[11],1);  (* j'ai demandé où sont *)
          ecrire(tab_mess_text[21],2);
        ELSE
          ecrire(tab_mess_text[11],1);  (* j'ai demandé où est *)
          ecrire(tab_mess_text[17],2);
        END;
        ecrire(tab_cadre_desig[num_enonce].nom_cadre,3);
        IF tab_param[3] = 1
        THEN IF pluriel(tab_cadre_desig[num_enonce].nom_cadre)
             THEN  dire(15);
                   dire(18);
             ELSE  dire(15);
                   dire(17);
             END;
             dire(num_zone_desig+19)
        END;
        i := 1;
        WHILE i < 4 DO
          aff_image(
           tab_cadre_desig[num_enonce].x + graphique_support.LeftEdge,
           tab_cadre_desig[num_enonce].y + graphique_support.TopEdge,
           tab_cadre_desig[num_enonce].brush);
          Delay(40);
          SetAPen(rp^,17);
          RectFill(rp^,
           tab_cadre_desig[num_enonce].x + graphique_support.LeftEdge,
           tab_cadre_desig[num_enonce].y + graphique_support.TopEdge,
           tab_cadre_desig[num_enonce].x + graphique_support.LeftEdge+
           tab_cadre_desig[num_enonce].brush.Width,
           tab_cadre_desig[num_enonce].y + graphique_support.TopEdge+
           tab_cadre_desig[num_enonce].brush.Height);
          Delay(40);
          i := i+1;
        END;

    ELSE       (* non vide *)
        IF tab_exercice[num_ex].anim = 1 (* par systeme *)
        THEN longstr :=
```

```
          StringLength(tab_cadre_desig[num_zone_desig].nom_cadre);
        ExtractSubString
         (str,tab_cadre_desig[num_zone_desig].nom_cadre,Ø,longstr-2);
        effacer_cadre(166,223,29,148);  (* interieur cadre *)
        ecrire(tab_mess_text[1Ø],1);
        ecrire(tab_mess_text[12],2);
        ecrire(str,3);              (* Tu as montré ce qui permet de *)
        IF tab_param[3] = 1
        THEN dire(15);
             dire(17);
             dire(num_zone_desig+19)
        END;
        clignoter_zone(num_zone_desig);
        longstr :=
         StringLength(tab_cadre_desig[num_enonce].nom_cadre);
        ExtractSubString
         (str,tab_cadre_desig[num_enonce].nom_cadre,Ø,longstr-2);
        effacer_cadre(166,223,29,148);  (* interieur cadre *)
        ecrire(tab_mess_text[11],1);
        ecrire(tab_mess_text[12],2);
        ecrire(str,3);   (* J'ai demandé ce qui permet de *)
        IF tab_param[3] = 1
        THEN dire(16);
             dire(17);
             dire(num_enonce+19)
        END;
        clignoter_zone(num_enonce);

        (* TRACE*)
        CopyString(trace,tab_mess_trace[14]);
        ConcatString(trace,tab_mess_trace[2Ø]);
        ConcatString(trace,str);
        ecrire_trace (trace);
        (* FIN TRACE *)
ELSE                    (* non vide et pas par systeme *)
        effacer_cadre(166,223,29,148);  (* interieur cadre *)
        ecrire(tab_mess_text[1Ø],1);      (* Tu as montré *)
        ecrire(tab_cadre_desig[num_zone_desig].nom_cadre,2);
        IF tab_param[3] = 1
        THEN dire(15);
             dire(num_zone_desig+19)
        END;
        IF tab_param[8]=1
        THEN coloration_zone(tab_cadre_nom[num_zone_desig].x+1,
                  tab_cadre_nom[num_zone_desig].y+1,
                  LONG_CADRE-1,HAUT_CADRE-1,1Ø,2);
                                        (* bleu par rouge *)
        END;
        clignoter_zone(num_zone_desig);
        IF tab_param[8]=1
        THEN coloration_zone(tab_cadre_nom[num_zone_desig].x+1,
                  tab_cadre_nom[num_zone_desig].y+1,
                  LONG_CADRE-1,HAUT_CADRE-1,2,1Ø);
                                        (* rouge par bleu *)
        END;

        effacer_cadre(166,223,29,148);  (* interieur cadre *)
        ecrire(tab_mess_text[11],1);  (* J'ai demandé *)
        ecrire(tab_cadre_desig[num_enonce].nom_cadre,2);
        IF tab_param[3] = 1
        THEN dire(16);
             dire(num_enonce+19)
        END;
        IF tab_param[8]=1
        THEN coloration_zone(tab_cadre_nom[num_enonce].x+1,
                  tab_cadre_nom[num_enonce].y+1,
```

```
                                LONG_CADRE-1,HAUT_CADRE-1,10,2);
                                                    (* bleu par rouge *)
                        (* TRACE*)
                        CopyString(trace,tab_mess_trace[14]);
                        ConcatString(trace,tab_mess_trace[19]);
                        ConcatString(trace,
                                    tab_cadre_desig[num_zone_desig].nom_cadre);
                        ecrire_trace (trace);
                        (* FIN TRACE *)

                    ELSE  (* TRACE*)
                        CopyString(trace,tab_mess_trace[14]);
                        ConcatString(trace,
                                    tab_cadre_desig[num_zone_desig].nom_cadre);
                        ecrire_trace (trace);
                        (* FIN TRACE *)

                    END;
                    clignoter_zone(num_enonce);
                    IF tab_param[8]=1
                    THEN coloration_zone(tab_cadre_nom[num_enonce].x+1,
                              tab_cadre_nom[num_enonce].y+1,
                              LONG_CADRE-1,HAUT_CADRE-1,2,10);
                                                    (* rouge par bleu *)
                    END;

                END;
            END;
        ELSE (* cas  pseudo_passif *)
            effacer_cadre(166,223,29,148);  (* interieur cadre *)
            ecrire(tab_mess_text[20],2); (* Tu as déjà montré *)
            IF tab_param[3] = 1
            THEN dire(16);
                 dire(num_zone_desig+19)
            END;
            Delay(100);

            (* TRACE*)
            CopyString(trace,tab_mess_trace[14]);
            ConcatString(trace,tab_cadre_desig[num_zone_desig].nom_cadre);
            ecrire_trace (trace);
            (* FIN TRACE *)

        END;
END feedback_negatif;

(*****************************************************************************)

PROCEDURE dessine_rect(x,y,l,h:INTEGER);

BEGIN
  SetAPen(rp^,1);
  Move(rp^,x,y);
  Draw(rp^,x+l,y);
  Move(rp^,x,y);
  Draw(rp^,x,y+h);
  Move(rp^,x,y+h);
  Draw(rp^,x+l,y+h);
  Move(rp^,x+l,y+h);
  Draw(rp^,x+l,y);

END dessine_rect;
(*****************************************************************************)

PROCEDURE colorie_partie_parcours(num_cadre_desig,num_dern_cadre : INTEGER);
```

```
VAR num_cadre_courant,x,y,l,h : INTEGER;

BEGIN

     num_cadre_courant := num_dern_cadre + 1;
     WHILE num_cadre_courant <= num_cadre_desig DO
          x := tab_cadre_parc[num_cadre_courant][1]
               + graphique_support.LeftEdge;
          y := tab_cadre_parc[num_cadre_courant][2]
               + graphique_support.TopEdge;
          l := tab_cadre_parc[num_cadre_courant][3];
          h := tab_cadre_parc[num_cadre_courant][4];
          coloration_zone(x,y,l,h,99,coul_parc);
          IF tab_cadre_parc[num_cadre_courant][5] # 500
          THEN
               x := tab_cadre_parc[num_cadre_courant][5]
                    + graphique_support.LeftEdge;
               y := tab_cadre_parc[num_cadre_courant][6]
                    + graphique_support.TopEdge;
               l := tab_cadre_parc[num_cadre_courant][7];
               h := tab_cadre_parc[num_cadre_courant][8];
               coloration_zone(x,y,l,h,99,coul_parc);
          END;
          num_cadre_courant := num_cadre_courant+1;
     END;
END colorie_partie_parcours;

(*****************************************************************************)

PROCEDURE animer_parcours;
BEGIN
END animer_parcours;

(*****************************************************************************)

PROCEDURE noircit_cadre(tout:BOOLEAN);
(* appelée si par nom et pp *)
VAR
   i,j: INTEGER;

BEGIN

 IF NOT tout
 THEN i:= 1;
      WHILE liste_enonce[i] <>999 DO
        coloration_zone(tab_cadre_nom[liste_enonce[i]].x+1,
                        tab_cadre_nom[liste_enonce[i]].y+1,
                        LONG_CADRE-1,HAUT_CADRE-1,
                        10,24);                    (* bleu par gris*)
        i:= i+1;
      END;
      i:=1;
      WHILE i<= nb_enonce DO
        indice(i,j);
        IF j = 0
        THEN coloration_zone(tab_cadre_nom[i].x+1,
                        tab_cadre_nom[i].y+1,
                        LONG_CADRE-1,HAUT_CADRE-1,
                        10,2);
                                        (*bleu par rouge *)
        END;
        i:= i+1;
      END;
 ELSE
      i:= 1;
      WHILE  i<= nb_enonce DO
```

```
            aff_brush_designation(99);
            coloration_zone(tab_cadre_nom[i].x+1,
                        tab_cadre_nom[i].y+1,
                        LONG_CADRE-1,HAUT_CADRE-1,
                        10,24);                    (*bleu par gris*)
            i:=i+1;
      END;
 END;
END noircit_cadre;

(*****************************************************************************)

PROCEDURE ecrire(VAR mess: ARRAY OF CHAR;num_ligne:INTEGER);

VAR x,y:INTEGER;

BEGIN
  SetAPen(rp^,1);
  x:= 240 -(TextLength(rp^,ADR(mess),StringLength(mess))DIV 2);
  CASE num_ligne OF

    1: y := 230; |
    2: y := 240; |
    3: y := 250; |
  END;
  Move (rp^,x,y);
  Text(rp^,ADR(mess),StringLength(mess));

END ecrire;

(*****************************************************************************)

PROCEDURE coloration_combinee(num_cadre_desig,num_dern_cadre:INTEGER);

VAR
    x,y,l,h,num_cadre_courant:INTEGER;

BEGIN
     num_cadre_courant:=num_dern_cadre+1;
     WHILE num_cadre_courant <= num_cadre_desig DO
       x := tab_cadre_parc[num_cadre_courant][1]
            + graphique_support.LeftEdge;
       y := tab_cadre_parc[num_cadre_courant][2]
            + graphique_support.TopEdge;
       l := tab_cadre_parc[num_cadre_courant][3];
       h := tab_cadre_parc[num_cadre_courant][4];
       coloration_zone(x,y,l,h,99,coul_parc);
       IF tab_cadre_parc[num_cadre_courant][5] # 500
       THEN
             x := tab_cadre_parc[num_cadre_courant][5]
                  + graphique_support.LeftEdge;
             y := tab_cadre_parc[num_cadre_courant][6]
                  + graphique_support.TopEdge;
             l := tab_cadre_parc[num_cadre_courant][7];
             h := tab_cadre_parc[num_cadre_courant][8];
             coloration_zone(x,y,l,h,99,coul_parc);
       END;
       deplacement_symbole(num_cadre_courant);
       num_cadre_courant := num_cadre_courant+1;
     END;
END coloration_combinee;

(*****************************************************************************)

PROCEDURE afficher_cartoon(afficher : BOOLEAN);
```

```
CONST
   tempo_tick = 30; (*** 50 = 1 seconde ***)

VAR
   r : RastPortPtr;
   x,y : INTEGER;
   current_tick : LONGCARD;
   date : DateStampRecord;

BEGIN
     r:=w^.RPort;
     IF afficher
        THEN
            DateStamp(date);
            current_tick:=LONGCARD((date.dsMinute*LONGINT(3000))+date.dsTick);
            IF current_tick >= (tick_cartoon + LONGCARD(tempo_tick))
            THEN
               (*** flip du cartoon ***)
               SetAPen(r^,0);
               x:=(320-tab_cartoon[cartoon][num_cartoon].Width) DIV 2;
               y:=(256-tab_cartoon[cartoon][num_cartoon].Height) DIV 2;
               DrawImage(r^,tab_cartoon[cartoon][num_cartoon],x,y);
               cartoon:=NOT cartoon;
               tick_cartoon:=current_tick;
            END;
     END;
END afficher_cartoon;


END graphique.
```

```
IMPLEMENTATION MODULE son;

FROM var_globale IMPORT type_renfor,scr,tab_son,nb_mess_sauv,sauvegarde_mess,
     tab_exercice,num_ex,tab_param,tab_cadre_desig;
FROM trt_donnees IMPORT pluriel;

FROM Intuition IMPORT DisplayBeep;
FROM RandomNumbers IMPORT Random;
FROM SoundLib IMPORT PlaySamp,Chan3,LoadAIFF;
FROM SYSTEM IMPORT ADR;

(*****************************************************************************)

PROCEDURE dire(num_son:INTEGER);

BEGIN
   PlaySamp(tab_son[num_son]^.Samp,3,364,64,tab_son[num_son]^.Size,
                                                        1,1);
END dire;

(*****************************************************************************)

PROCEDURE renforcement_positif;

VAR ran_resultat : INTEGER;

BEGIN


      CASE type_renfor OF
      Ø : (*** renforcement musical non aleatoire ***)
          dire(1Ø);¦
      1 : (*** renforcement vocal ***)
          dire(1Ø);¦           (* "c'est juste" *)
      2 : (*** renforcement musical aleatoire ***)
          ran_resultat:=Random(3);
          CASE ran_resultat OF
          Ø : dire(1Ø); ¦
          1 : dire(12); ¦
          2 : dire(13); ¦
          END;
      END;


END renforcement_positif;

(*****************************************************************************)
PROCEDURE renforcement_negatif;

BEGIN

      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
```

```
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      DisplayBeep(scr);
      IF (type_renfor=0) OR (type_renfor=2) THEN
         (*** Renforcement de type musical ***)
         dire(11);
      ELSE
         (*** Renforcement de type vocal ***)
         dire(11);  (* "ce n'est pas juste" *)
      END;

END renforcement_negatif;

(*****************************************************************************)

PROCEDURE enonce_sonore(num:INTEGER);

BEGIN
  IF (tab_param[6]=2) AND (tab_exercice[num_ex].anim=0) AND (tab_param[8]=2)
  THEN dire(14);dire(29+num);
       sauvegarde_mess[1]:=14;
       sauvegarde_mess[2]:=29+num;
       nb_mess_sauv:=2;
  ELSE IF pluriel(tab_cadre_desig[num].nom_cadre)
       THEN  dire(14);
             dire(18);
             sauvegarde_mess[2]:=18;
       ELSE  dire(14);
             dire(17);
             sauvegarde_mess[2]:=17;
       END;dire(29+num);
       sauvegarde_mess[1]:=14;
       sauvegarde_mess[3]:=29+num;
       nb_mess_sauv := 3;
  END;


END enonce_sonore;

(*****************************************************************************)

PROCEDURE copy_son_fixe;

BEGIN
       tab_son[1]:= LoadAIFF(ADR("son1")); (* quel est ton nom *)
       tab_son[2]:= LoadAIFF(ADR("son2")); (* est_ce bien ton nom *)
       tab_son[3]:= LoadAIFF(ADR("son3")); (* ce nom m'est inconnu *)
       tab_son[4]:= LoadAIFF(ADR("son4")); (* pour continuer cliqu..oui *)
       tab_son[5]:= LoadAIFF(ADR("son5")); (* veux_tu realiser un autre ex *)
       tab_son[6]:= LoadAIFF(ADR("son6")); (* veux_tu recommencer cet ex *)
       tab_son[7]:= LoadAIFF(ADR("son7")); (* veux_tu finir *)
       tab_son[8]:= LoadAIFF(ADR("son8")); (* veux_tu stopper *)
       tab_son[9]:= LoadAIFF(ADR("son9")); (* je repete *)
  IF type_renfor = 1 (* sonore *)
  THEN
       tab_son[10]:= LoadAIFF(ADR("son10")); (* premiere musique *)
       tab_son[11]:= LoadAIFF(ADR("son11")); (* son negatif *)
  ELSE IF type_renfor = 2 (* aleatoire *)
       THEN tab_son[10]:= LoadAIFF(ADR("son10")); (* idem *)
            tab_son[11]:= LoadAIFF(ADR("son11")); (* idem *)
            tab_son[12]:= LoadAIFF(ADR("son14")); (* deuxieme musique *)
```

```
            tab_son[13]:= LoadAIFF(ADR("son15")); (* troisieme musique *)

        ELSE (* vocal *)
          tab_son[10]:= LoadAIFF(ADR("son12")); (* c'est juste *)
          tab_son[11]:= LoadAIFF(ADR("son13")); (* ce n'est pas juste *)
        END;
  END;

END copy_son_fixe;

(*****************************************************************************)

PROCEDURE charger_son_ex(etat: INTEGER);

BEGIN

  CASE etat OF
    1: (* parcours *)
       tab_son[14]:= LoadAIFF(ADR("son16")); (* veux_tu montrer *)
       tab_son[15]:= LoadAIFF(ADR("son30")); (* par où passe *)
       tab_son[16]:= LoadAIFF(ADR("son31")); (* recommence *)
       tab_son[17]:= LoadAIFF(ADR("son24")); (* tu l'as déjà montré *)
       tab_son[18]:= LoadAIFF(ADR("son32")); (* tu as oublié *)|

    2: (* designation normale, non pp *)
       tab_son[14]:= LoadAIFF(ADR("son16")); (* veux_tu montrer *)
       tab_son[15]:= LoadAIFF(ADR("son17")); (* tu as montre *)
       tab_son[16]:= LoadAIFF(ADR("son18")); (* j'ai demande *)
       tab_son[17]:= LoadAIFF(ADR("son28")); (* l'endroit ou est *)
       tab_son[18]:= LoadAIFF(ADR("son29")); (* l'endroit ou sont *)|

    3: (* pp nom *)
       tab_son[14]:= LoadAIFF(ADR("son16")); (* veux_tu montrer *)
       tab_son[15]:= LoadAIFF(ADR("son23")); (* tu montres *)
       tab_son[16]:= LoadAIFF(ADR("son24")); (* tu l'as deja montre *)
       tab_son[17]:= LoadAIFF(ADR("son26")); (* le nom d'un organe  *)
       tab_son[18]:= LoadAIFF(ADR("son27")); (* ce qui s'appelle *)|

    4: (* pp cadre *)
       tab_son[14]:= LoadAIFF(ADR("son16")); (* veux_tu montrer*)
       tab_son[15]:= LoadAIFF(ADR("son23")); (* tu montres *)
       tab_son[16]:= LoadAIFF(ADR("son24")); (* tu l'as deja montre *)
       tab_son[17]:= LoadAIFF(ADR("son25")); (* un organe *)|
    5: (* par nom et pas pp *)
       tab_son[14]:= LoadAIFF(ADR("son20")); (* veux_tu montrer son nom *)
       tab_son[15]:= LoadAIFF(ADR("son17")); (* tu as montre *)
       tab_son[16]:= LoadAIFF(ADR("son18")); (* j'ai demande *)
       tab_son[17]:= LoadAIFF(ADR("son21")); (* ce qui s'appelle *)|

    6: (* par syst *)
       tab_son[14]:= LoadAIFF(ADR("son16")); (* veux_tu montrer *)
       tab_son[15]:= LoadAIFF(ADR("son17")); (* tu as motre *)
       tab_son[16]:= LoadAIFF(ADR("son18")); (* j'ai demande *)
       tab_son[17]:= LoadAIFF(ADR("son19")); (* ce qui permet de *)|
  END;

END charger_son_ex;

(*****************************************************************************)

END son.
```

```
IMPLEMENTATION MODULE trt_donnees;

FROM var_globale IMPORT nb_enonce,tab_cadre_desig,graphique_support,
                        liste_enonce,tab_param,tab_couleur,point,string80,
                        tab_cadre_parc,tab_son,tab_cadre_obl,MAX_SONS,
                        nb_classe,nb_cartoon,num_cartoon,tab_cadre_nom,
                        tab_coord_nom,LONG_CADRE,HAUT_CADRE,pos_tab_trace
                        ,tab_mess_trace,symbole;
FROM  var_globale_trace IMPORT tab_trace,MAX_TRACE;
FROM  RandomNumbers IMPORT Random;
FROM  Strings IMPORT ExtractSubString;

VAR liste : ARRAY[1..11] OF INTEGER;

(*****************************************************************************)

PROCEDURE indice(num:INTEGER;VAR ind:INTEGER);

VAR
   i: INTEGER;
   trouve:BOOLEAN;
BEGIN
     i:= 1;ind := 0;
     trouve := FALSE;
     WHILE (liste_enonce[i] <> 999) AND (NOT trouve) DO
       IF liste_enonce[i] = num
       THEN ind := i;trouve:= TRUE;
       ELSE i:=i+1;
       END;
     END;
END indice;

(*****************************************************************************)

PROCEDURE detection_cadre_desig(VAR ok : BOOLEAN; x,y : INTEGER;
                  num_enonce : INTEGER; VAR num_zone_desig : INTEGER;
                  VAR ind:INTEGER);

VAR i,rect_x,rect_y,rect_l,rect_h,min_zone,surface : INTEGER;
    trouve : BOOLEAN;

BEGIN

      min_zone:=32000;
      i:=1;
      ok:=FALSE;
      num_zone_desig:=0;
      WHILE (i<=nb_enonce) AND (ok = FALSE)  DO
         IF tab_param[8] =1 (* PAR NOM *)
         THEN
            rect_x:=tab_cadre_nom[i].x;
            rect_y:=tab_cadre_nom[i].y;
            rect_l:=LONG_CADRE;
            rect_h:=HAUT_CADRE;
         ELSE
            rect_x:=tab_cadre_desig[i].x + graphique_support.LeftEdge;
            rect_y:=tab_cadre_desig[i].y + graphique_support.TopEdge;
            rect_l:=tab_cadre_desig[i].brush.Width;
            rect_h:=tab_cadre_desig[i].brush.Height;
         END;
         IF (x>=rect_x) AND (x<=rect_x+rect_l) AND (y>=rect_y) AND
                                                  (y<=rect_y+rect_h) THEN
            (*** J'ai trouve une zone dans laquelle l'utilisateur a clique ***)
```

```
            (*** calcul de la surface de la zone désignée                    ***)

             IF tab_param[6]=1  (* par systeme *)
             THEN
               IF num_enonce = i
               THEN ok := TRUE;
                    num_zone_desig := i;

               ELSE surface := rect_h * rect_l;
                    IF surface<min_zone
                    THEN min_zone:=surface;
                         num_zone_desig:=i;
                    END;
               END;

             ELSE
               surface:=rect_h * rect_l;
               IF surface<min_zone
               THEN min_zone:=surface;
                    num_zone_desig:=i;
               END;
             END;
          END;
          i:= i+1;
       END;

       IF tab_param[1]<>3 (* pas pp *)
       THEN IF num_zone_desig = num_enonce
            THEN    (* Designation correcte de la part de l'utilisateur      ***)
                ok:=TRUE;

            END;
            ind := 1;
       ELSE (* pp *)

            indice(num_zone_desig,ind);
            IF ind <> Ø THEN ok:= TRUE END;
       END;


END detection_cadre_desig;

(*****************************************************************************)

PROCEDURE init_liste_enonce;

VAR
   i,j,k,pos,resultat :INTEGER;
   tab : ARRAY [1..4Ø] OF INTEGER;
   max : LONGCARD;

BEGIN
     i:= 1;
     IF tab_param[1] =3      (*  PSEUDO_DESIGNATION*)
     THEN
          WHILE i<=nb_enonce DO
            liste_enonce[i] :=i;
            i := i+1;
          END;
          liste_enonce[i] := 999;
     ELSE
       IF tab_param[2] = 1          (* dans l'ordre logique d'enumeration *)
       THEN

          i:=1;
          WHILE i<=nb_enonce DO
```

```
            liste_enonce[i] :=i;
            liste_enonce[i+nb_enonce+1] := i;
            i := i+1;
          END;
          liste_enonce[i] := 99;
          liste_enonce[i+nb_enonce+1] := 999;
        ELSE
          pos := 1;
          WHILE i<= 2 DO
            k := 1;
            WHILE k <= nb_enonce DO
              tab[k] := k;
              k := k+1
            END;
            k := nb_enonce;
            WHILE k >=1 DO
              max := k;
              resultat := Random(max);
              resultat := resultat +1;
              liste_enonce[pos] := tab[resultat];
              j := resultat;
              WHILE j < k DO
               tab[j] := tab[j+1];
               j := j+1;
              END;
              k := k-1;
              pos := pos+1;
           END;
           IF i=1 THEN liste_enonce[pos] := 99;
           ELSE liste_enonce[pos] := 999;
           END;
          pos := pos+1;
          i := i+1;
        END;
      END;
    END;
END init_liste_enonce;

(*************************************************************************)


PROCEDURE detection_couleur(VAR ok: BOOLEAN;num_couleur,num_enonce:INTEGER;
                            VAR num_zone_desig :INTEGER);
VAR
    trouve : BOOLEAN;
    i : INTEGER;
BEGIN
     ok := FALSE;
     IF tab_couleur[num_enonce] =num_couleur
     THEN ok := TRUE;
          num_zone_desig := num_enonce;
     ELSE i :=1;
          trouve := FALSE;
          num_zone_desig := Ø;
          WHILE (i <= nb_enonce) AND (NOT trouve) DO
            IF tab_couleur[i] = num_couleur
            THEN num_zone_desig :=i;
                 trouve := TRUE;
            END;
            i :=i+1;
          END;
     END;

END detection_couleur;

(*************************************************************************)
```

```
PROCEDURE calcul_centre_cadre(num_cadre:INTEGER;VAR centre1,centre2:point);

BEGIN
     centre1.x := (tab_cadre_parc[num_cadre][1] +
                   graphique_support.LeftEdge +
                  (tab_cadre_parc[num_cadre][3] DIV 2) -
                  (symbole.Width DIV 2));
     centre1.y := (tab_cadre_parc[num_cadre][2] +
                   graphique_support.TopEdge +
                  (tab_cadre_parc[num_cadre][4] DIV 2) -
                  (symbole.Height DIV 2));
     IF tab_cadre_parc[num_cadre][5] <> 500
     THEN centre2.x := (tab_cadre_parc[num_cadre][5] +
                        graphique_support.LeftEdge +
                       (tab_cadre_parc[num_cadre][7] DIV 2) -
                       (symbole.Width DIV 2));
          centre2.y := (tab_cadre_parc[num_cadre][6] +
                        graphique_support.TopEdge +
                       (tab_cadre_parc[num_cadre][8] DIV 2) -
                       (symbole.Height DIV 2));
     ELSE centre2.x := centre1.x;
          centre2.y := centre1.y;
     END;

END calcul_centre_cadre;

(*****************************************************************************)

PROCEDURE maj_liste_enonce(ok:BOOLEAN;VAR fin_liste:BOOLEAN;indice:INTEGER);

VAR i,saut,num_reinser : INTEGER;
    stop:BOOLEAN;

BEGIN
     fin_liste := FALSE;
     IF NOT ok
     THEN num_reinser := liste_enonce[1];
          i:=1;
          stop := FALSE;
          WHILE (i<= tab_param[3]) AND (NOT stop) DO
            liste_enonce[i] :=liste_enonce[i+1];
            i:= i+1;
            IF (liste_enonce[i] = 99) OR (liste_enonce[i] = 999)
            THEN stop := TRUE;
            END;
          END;
          IF NOT stop
          THEN liste_enonce[i] := num_reinser;
          ELSE liste_enonce[i-1] := num_reinser;
          END;
     ELSE i:= indice +1;
          saut := 0;
          liste_enonce[indice] := liste_enonce[indice+1];
          IF liste_enonce[indice+1] = 99
          THEN saut := -1;
          END;
          WHILE liste_enonce[i] <> 999 DO
            liste_enonce[i+saut] := liste_enonce[i+1];
            i := i+1;
          END;
          IF liste_enonce[1] = 999
          THEN fin_liste := TRUE;
          ELSE fin_liste := FALSE;
          END;
```

```
    END;

END maj_liste_enonce;

(*****************************************************************************)

PROCEDURE init_tab_son_nil;

VAR i: INTEGER;

BEGIN
     i:= 1;
     WHILE i<= MAX_SONS DO
       tab_son[i] := NIL;
       i:= i+1;
     END;

END init_tab_son_nil;

(*****************************************************************************)

PROCEDURE detection_cadre_parc(VAR num_cadre,num_classe:INTEGER;
                               x,y:INTEGER);

VAR
   cadrex,cadrey,cadrel,cadreh,i,j,k:INTEGER;
   trouve:BOOLEAN;
BEGIN
     num_cadre:= Ø;
     trouve:= FALSE;
     i := 1;
     WHILE (i<= nb_enonce) AND (NOT trouve) DO
       cadrex := tab_cadre_parc[i][1] + graphique_support.LeftEdge;
       cadrey := tab_cadre_parc[i][2] + graphique_support.TopEdge;
       cadrel := tab_cadre_parc[i][3];
       cadreh := tab_cadre_parc[i][4];
       IF (x >= cadrex) AND (x<= cadrex + cadrel) AND
          (y >= cadrey) AND (y<= cadrey + cadreh)
       THEN num_cadre := i;
            trouve := TRUE;
       ELSE IF tab_cadre_parc[i][5] <> 5ØØ
            THEN cadrex := tab_cadre_parc[i][5] + graphique_support.LeftEdge;
                 cadrey := tab_cadre_parc[i][6] + graphique_support.TopEdge;
                 cadrel := tab_cadre_parc[i][7];
                 cadreh := tab_cadre_parc[i][8];
                 IF (x >= cadrex) AND (x<= cadrex + cadrel) AND
                    (y >= cadrey) AND (y<= cadrey + cadreh)
                 THEN num_cadre:= i;
                      trouve := TRUE;
                 END;
            END;
       END;
       i:= i+1;
     END;
     num_classe := Ø;
     IF num_cadre <> Ø
     THEN j :=1;
        trouve:= FALSE;
        WHILE (j <= nb_classe) AND (NOT trouve) DO
          k := 1;
          WHILE (tab_cadre_obl[j][k] >=1) AND (NOT trouve) DO
             IF (tab_cadre_obl[j][k]<= num_cadre)
             THEN IF tab_cadre_obl[j][k] = num_cadre
                  THEN num_classe := j;
                       trouve :=TRUE;
                  END;
```

```
            ELSE trouve := TRUE;
            END;
            k:= k+1;
          END;
          j :=j+1;
        END;
      END;

END detection_cadre_parc;

(*****************************************************************************)

PROCEDURE choix_cartoon;

BEGIN
   num_cartoon := Random(LONGCARD(nb_cartoon));
   num_cartoon := num_cartoon + 1;
END choix_cartoon;

(*****************************************************************************)

PROCEDURE init_tab_coord_nom;

BEGIN
     tab_coord_nom [1].x := 3;
     tab_coord_nom [1].y := 3;
     tab_coord_nom [2].x := 3;
     tab_coord_nom [2].y := 23;
     tab_coord_nom [3].x := 3;
     tab_coord_nom [3].y := 43;
     tab_coord_nom [4].x := 3;
     tab_coord_nom [4].y := 63;
     tab_coord_nom [5].x := 3;
     tab_coord_nom [5].y := 83;
     tab_coord_nom [6].x := 3;
     tab_coord_nom [6].y := 103;
     tab_coord_nom [7].x := 3;
     tab_coord_nom [7].y := 123;
     tab_coord_nom [8].x := 3;
     tab_coord_nom [8].y := 143;
     tab_coord_nom [9].x := 3;
     tab_coord_nom [9].y := 163;
     tab_coord_nom [10].x := 3;
     tab_coord_nom [10].y := 183;
     tab_coord_nom [11].x := 3;
     tab_coord_nom [11].y := 203;
END init_tab_coord_nom;

(*****************************************************************************)

PROCEDURE chercher (j,taille_liste : INTEGER;VAR trouver : BOOLEAN);

VAR i : INTEGER;
BEGIN
    trouver := FALSE;
    i := 1;
    WHILE (i<=taille_liste) AND (trouver = FALSE) DO
          IF liste [i] = j THEN trouver := TRUE END;
          i := i+1;
    END;
END chercher;

(*****************************************************************************)

PROCEDURE init_tab_cadre_nom;
```

```
VAR i,j,taille_liste : INTEGER;
    trouver : BOOLEAN;
BEGIN
    i := 1;
    taille_liste := Ø;
    WHILE i <= nb_enonce DO
        j := Random(nb_enonce);
        j := j +1;
        chercher(j,taille_liste,trouver);
        IF trouver = FALSE (* tab_cadre_nom[j] n'a pas encore reçu de valeur*)
        THEN
            tab_cadre_nom [j].x := tab_coord_nom[i].x;
            tab_cadre_nom [j].y := tab_coord_nom[i].y;
            taille_liste := taille_liste + 1;
            liste[taille_liste]:= j;
            i := i + 1;
        END;
    END;
END init_tab_cadre_nom;
(*****************************************************************************)

PROCEDURE pluriel (VAR nom : ARRAY OF CHAR) : BOOLEAN;

VAR str : ARRAY [1..2] OF CHAR;

BEGIN
     ExtractSubString (str,nom,2,1);
     IF (str[1] = "s") OR (str[1] = "S")
     THEN RETURN TRUE;
     ELSE RETURN FALSE;
     END;
END pluriel;
(*****************************************************************************)

PROCEDURE ecrire_trace (mess : string8Ø);

BEGIN
     IF pos_tab_trace <= MAX_TRACE
     THEN IF pos_tab_trace<MAX_TRACE-2
          THEN tab_trace[pos_tab_trace]:=mess;
               pos_tab_trace := pos_tab_trace+1;
          ELSE tab_trace[pos_tab_trace]:= tab_mess_trace[42];
               tab_trace[pos_tab_trace+1]:= tab_mess_trace[43];
               pos_tab_trace := pos_tab_trace+2;
          END;
     END;
END ecrire_trace;

(*****************************************************************************)

PROCEDURE verifie_parcours( num_cadre,num_anc_cadre,num_classe,
                            num_anc_classe:INTEGER;VAR res:INTEGER);

BEGIN
     IF num_cadre #Ø
     THEN IF num_cadre >= num_anc_cadre
          THEN IF num_classe = Ø
               THEN IF num_anc_classe = nb_classe
                    THEN res := 1;
                    ELSE IF num_cadre<= tab_cadre_obl[num_anc_classe+1][1]
                         THEN res := 1;
                         ELSE res := 3;
                         END;
                    END;
               ELSE IF num_classe = num_anc_classe
                    THEN res := 1;
```

```
                    ELSE IF num_classe = num_anc_classe +1
                         THEN res := 1;
                         ELSE res := 3;
                         END;
                    END;
               END;
          ELSE res :=2;
          END;
     ELSE res := 4;
     END;

END verifie_parcours;



END trt_donnees.
```

ANNEXE 5 : Les "definition modules" et les "implementation modules" du logiciel "Parametrage"

```
DEFINITION MODULE varglobale;

FROM Intuition IMPORT WindowPtr,GadgetPtr,ScreenPtr;
FROM Rasters IMPORT RastPortPtr;

CONST MAX_EX = 5;
TYPE
     string30=ARRAY [1..30] OF CHAR;
     string50=ARRAY [1..50] OF CHAR;
     string70=ARRAY [1..70] OF CHAR;

     exercice = RECORD
                     typeex:CHAR;
                     numbiblio:INTEGER;
                     numcat:INTEGER;
                END;
      tableauex = ARRAY [1..MAX_EX] OF exercice;
      commentaire = RECORD
                        numbiblio:INTEGER;
                        syst : INTEGER;
                        coul: INTEGER;
                        com: string70;
                    END;
      tableauentier = ARRAY[1..8] OF INTEGER;


VAR tabex: tableauex;
    scr: ScreenPtr;
    rp: RastPortPtr;
    gl: GadgetPtr;
    abandontotal,abandonpartiel: BOOLEAN;
    win: WindowPtr;
    nbex: INTEGER;
    tabparam: ARRAY [1..MAX_EX] OF tableauentier;
    tabcommentaired: ARRAY [1..50] OF commentaire;
    tabcommentairep: ARRAY [1..50] OF commentaire;
    nbcommentaired: INTEGER;
    nbcommentairep: INTEGER;
    tabmess: ARRAY [1..120] OF string50;
    typesaisienom: CHAR;
    typerenfor: INTEGER;
    nomuser: string30;
    num_cour_biblio : INTEGER;

PROCEDURE init_tabmess;
(*****************************************************************************
*
* IN : /
*
* OUT : /
*
* BUT : Initialisation du tableau des messages écran.
*
*****************************************************************************)

END varglobale.
```

```
DEFINITION MODULE menu;

PROCEDURE menu_principal;
(*****************************************************************************
 *
 * IN : /
 *
 * OUT : /
 *
 * BUT : Gère le choix dans le menu principal
 *
 *****************************************************************************)

END menu.
```

```
DEFINITION MODULE saisiedonnees;

PROCEDURE saisie_donnees_init;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Gére la saisie de toutes les données nécessaires à l'initialisation
*       d'une disquette user. Elle copie tous les  fichiers  contenant  ces
*       données sur la disquette USER.
*****************************************************************************)

END saisiedonnees.
```

```
DEFINITION MODULE ecran;

FROM varglobale IMPORT string30;

PROCEDURE saisie_binaire(VAR res : INTEGER);
(*****************************************************************************
 *
 * IN : /
 *
 * OUT : res : résultat.
 *
 * BUT : Gestion de 3 gadgets: res = 1 si le 1er gadget est choisi,
 *                                 = 2 pour le 2ème
 *                                 = 3 pour le 3ème.
 *
 *****************************************************************************)

PROCEDURE titre_ecran(num : INTEGER);
(*****************************************************************************
 *
 * IN : num : numéro du commentaire.
 *
 * OUT : /
 *
 * BUT : Affiche et centre en haut de l'écran tabmess[num].
```

```
 ********************************************************************************)

PROCEDURE ecran_1;
(********************************************************************************
 *
 * IN : /
 *
 * OUT : /
 *
 * BUT : Crée l'écran de saisie du nom de l'utilisateur et place les caractères
 *       saisis au clavier dans nomuser.
 *
 ********************************************************************************)

PROCEDURE ecran_2(VAR type_ex : CHAR);
(********************************************************************************
 *
 * IN : /
 *
 * OUT : type_ex : type d'exercice (d ou p)

 *
 * BUT : Crée l'écran de saisie du choix du type d'exercice. type_ex = "d" si
 *       le choix est désignation. type_ex = "p" si c'est un parcours.
 *
 ********************************************************************************)

PROCEDURE ecran_3;
(********************************************************************************
 *
 * IN : /
 *
 * OUT : /
 *
 * BUT : Crée l'écran de confirmation de l'enregistrement de l'exercice.
 *
 ********************************************************************************)

PROCEDURE ecran_4;
(********************************************************************************
 *
 * IN : /
 *
 * OUT : /
 *
 * BUT : Crée l'écran indiquant que le nb maximum d'exercice est atteint.
 *
 ********************************************************************************)

PROCEDURE ecran_5(VAR encore : BOOLEAN);
(********************************************************************************
 *
 * IN : /.
 *
 * OUT : encore = TRUE si on veut encore un exercice; =FALSE sinon.
 *
 * BUT : Rééalisation e l'éécran de demande d'un exercice supleémentaire.
 *
 ********************************************************************************)

PROCEDURE ecran_6;
(********************************************************************************
 *
 * IN : /.
 *
```

```
 * OUT : /.
 *
 * BUT : Réalisation de l'ecran demandant le type de renforcement à
 * utiliser pour la session.
 *
 ***************************************************************************)

PROCEDURE ecran_7;
(***************************************************************************
 *
 * IN : /.
 *
 * OUT : /.
 *
 * BUT : Réalisation de l'ecran demandant le type de saisie du nom à
 * utiliser pour la session. Place la valeur c (comparaison) ou k (clavier)
 * dans typesaisienom.
 *
 ***************************************************************************)

PROCEDURE ecran_8(mauvaise : BOOLEAN);
(***************************************************************************
 *
 * IN : mauvaise = TRUE si la disquette introduite n'est pas bonne.
 *
 * OUT : /.
 *
 * BUT : Réalisation de l'écran de demande d'introduction d'une disquette
 * utilisateur.
 *
 ***************************************************************************)

PROCEDURE ecran_9(VAR abandon,type_impression_ecran : BOOLEAN);
(***************************************************************************
 *
 * IN : /.
 *
 * OUT : abandon : =TRUE si l'utilisaeur abandonne la fonction
 *       type_impression_ecran : =TRUE si l'utilisateur desire une
 *                   impression a l'ecran.
 *
 * BUT : Demande à l'utilisateur où il désire la sortie du fichier trace
 *
 ***************************************************************************)

PROCEDURE ecran_1Ø(num_disk : INTEGER);
(***************************************************************************
 *
 * IN : num_disk : numero de la disquette bibliothèque a introduire.
 *
 * OUT : /.
 *
 * BUT : Realisation de l'écran de demande d'introduction de la disquette
 * Biliotheque numero num_disk.
 *
 ***************************************************************************)

PROCEDURE prep_prt;
(***************************************************************************
 *
 * IN : /
 *
 * OUT : /
 *
 * BUT : Réalisation de l'écran demandant de préparer l'imprimante.
 *
```

```
(*****************************************************************************)

PROCEDURE ecran_11(VAR fich:string30);
(*****************************************************************************
 *
 * IN : /.
 *
 * OUT : fich : nom du fichier à supprimer.
 *
 * BUT :  Gestion de la suppression du fichier des traces fich.
 *
 *****************************************************************************)

PROCEDURE demande_nom_fichier(VAR fichier:string30; VAR abandon:BOOLEAN;
                              type : CHAR);
(*****************************************************************************
 *
 * IN : /
 *
 * OUT : abandon : =TRUE si l'utilisateur abandonne la fonction
 *       fichier : correspond au nom du fichier trace que l'utilisateur
 *                 désire imprimer.
 *       type : valeur de la m-à-j, suppression ou visualisation.
 *
 * BUT : Saisie d'un nom de fichier existant sur la disquette introduite
 *       dans un des drives.
 *
 *****************************************************************************)

PROCEDURE impression_trace_ecran(nomfichier : string30);
(*****************************************************************************
 *
 * IN : nomfichier : nom du fichier des traces à imprimer a l'écran.
 *
 * OUT : /.
 *
 * BUT : Gestion de l'impression du fichier des traces a l'écran.
 *
 *****************************************************************************)

PROCEDURE impression_com(typeex : CHAR);
(*****************************************************************************
 *
 * IN : nom_fichier : nom du fichier des commentaires à imprimer a l'ecran.
 *
 * OUT : /.
 *
 * BUT : Gestion de l'impression du fichier des commentaires a l'écran.
 *
 *****************************************************************************)


END ecran.




DEFINITION MODULE basedonnees;

FROM varglobale IMPORT string30;


PROCEDURE  lit_catalogue_desig;
```

```
(*****************************************************************************
*
* IN : /
*
* OUT : /
*
* BUT : Lit le fichier catalogue_desig se trouvant dans biblioØ1 et place
*       les différents enregistrements de ce fichier dans tabcommentaired
*       et dans nbcommentaired (nb d'enregistrements du fichier). S'il n'y a
*       pas d'enregistrement, nbcommentaired = Ø.
*
*****************************************************************************)

PROCEDURE  lit_catalogue_parc;
(*****************************************************************************
*
* IN : /
*
* OUT : /
*
* BUT : Lit le fichier catalogue_parcours se trouvant dans biblioØ1 et place
*       les différents enregistrements de ce fichier dans tabcommentairep
*       et dans nbcommentairep (nb d'enregistrements du fichier). S'il n'y a
*       pas d'enregistrement, nbcommentairep = Ø.
*
*****************************************************************************)

PROCEDURE cherchernomuser(VAR nom:string3Ø);
(*****************************************************************************
*
* IN :  /
*
* OUT : Nom de l'utilisateur.
*
* BUT : Elle place dans NOM le nom de l'utilisateur associé à la disquette
*       USER c'est-à-dire le nom se trouvant dans donnees_session de USER
*
*
*****************************************************************************)

PROCEDURE verifiediskuser(VAR cor : BOOLEAN);
(*****************************************************************************
*
* IN : /
*
* OUT : cor est vrai si la disquette USER  est la bonne
*           est faux sinon.
*
* BUT : Elle vérifie si la disquettte USER est bien la bonne, i.e. si
*       le nom introduit par l'éducateur (nomuser) est celui se trouvant
*       dans le fichier donnee_session, ou, si la disquette USER est vierge.
*
*
*****************************************************************************)

PROCEDURE copiefichier(entree,sortie : ARRAY OF CHAR);
(*****************************************************************************
*
* IN : entree : nom du fichier à copier (fichier source).
*      sortie : nom de la copie (fichier destination).
*
* OUT : /
*
* BUT : Copie le fichier entree sur le fichier sortie.
*
*****************************************************************************)
```

```
PROCEDURE ajout_exercice(numex: INTEGER);
(*****************************************************************************
*
* IN : numex : numéro de l'exercice.
*
* OUT : /
*
* BUT : Copie dans le fichier ram:liste_exercice, la tabex[numex].numcat ième
*       ligne du fichier désignation_info (si tabex[numex].typeex = "d")
*       ou parcours_info (si tabex[numex].typeex = "p") se trouvant dans
*       biblioØ1.
*
*****************************************************************************)

END basedonnees.
```

```
DEFINITION MODULE utilitaire;

FROM Rasters IMPORT RastPortPtr;
FROM Intuition IMPORT  ScreenPtr;

PROCEDURE centrage(rpt : RastPortPtr; x,y,largeur : INTEGER;
                   message : ARRAY OF CHAR);
(*****************************************************************************
 *
 * IN : rp : pointeur vers un rasport
 *      y  : valeur du Y ou doit s'afficher le message
 *      x,largeur : largeur dans laquelle le message doit s'afficher
 *      message : message a centrer dans une largeur Largeur a partie de
 *                la colonne x.
 *
 * OUT : /
 *
 * BUT : Afficher le message centré entre les colonnes x et
 * x+largeur. La position y du message est y.
 *
 *****************************************************************************)

PROCEDURE verifiefich(ecran:ScreenPtr;nom,drive : ARRAY OF CHAR) : BOOLEAN;
(*****************************************************************************
 *
 * IN : nom : nom de fichier, de directory ou de volume;
 *      drive : nom du drive sur lequel il faut effectuer la vérification;
 *      ecran : pointeur vers la structure ecran definie;
 *
 * OUT : TRUE si le nom en entree existe bien sur le drive en entree;
 *
 * BUT : vérifier l'existence sur le drive en entree du fichier, du
 * directory ou du volume donné en entree.
 *
 *****************************************************************************)

PROCEDURE init_couleur;
(*****************************************************************************
 *
 * IN : /
 *
 * OUT : /
 *
```

```
* BUT : Initialisation de la palette de couleur utilisée par le logiciel
*
**********************************************************************)

END utilitaire.
```

```
IMPLEMENTATION MODULE varglobale;

PROCEDURE init_tabmess;

VAR
   compteur : INTEGER;
   i :INTEGER;

BEGIN
     compteur:=1;
     WHILE compteur<=12Ø DO
           i:=1;
           WHILE i<=5Ø DO
                 tabmess[compteur][i]:=ØC;
                 i:=i+1;
           END;
           compteur:=compteur+1;
     END;
     tabmess[1]:="MENU PRINCIPAL";
     tabmess[2]:="ORDRE";
     tabmess[3]:="DESORDRE";
     tabmess[4]:="Initialisation d'une disquette utilisateur";
     tabmess[5]:="Quitter";
     tabmess[6]:="MENU DE GESTION DES PARAMETRES PARCOURS";
     tabmess[7]:="Liste des énoncés";
     tabmess[8]:="COULEUR";
     tabmess[9]:="CADRE";
     tabmess[1Ø]:="Fin";
     tabmess[11]:="MENU DE GESTION DES PARAMETRES DESIGNATION";
     tabmess[12]:="SAISIE DE LA VALEUR DES PARAMETRES";
     tabmess[13]:="";
     tabmess[14]:="Enoncé verbal de la désignation";
     tabmess[15]:="Nombre d'itérations";
     tabmess[16]:="Type d'interaction";
     tabmess[17]:="Nature d'énumération";
     tabmess[18]:="Silhouette vide";
     tabmess[19]:="Animation des systèmes";
     tabmess[2Ø]:="type de désignation";
     tabmess[21]:="NOM";
     tabmess[22]:="IMAGE";
     tabmess[23]:="Parcours au début de l'exercice";
     tabmess[24]:="Point de départ et de sortie";
     tabmess[25]:="Type de feed-back";
     tabmess[26]:="Parcours en fin de l'exercice";
     tabmess[27]:="SYMBOLE";
     tabmess[28]:="APPRENTISSAGE";
     tabmess[29]:="EVALUATION";
     tabmess[3Ø]:="OUI";
     tabmess[31]:="NON";
     tabmess[32]:="COLORATION";
     tabmess[33]:="MIXTE";
     tabmess[34]:="INITIALISATION D'UNE DISQUETTE UTILISATEUR";
     tabmess[35]:="Nom de l'utilisateur :";
     tabmess[36]:="Type d'exercice :";
     tabmess[37]:="Exercice supplémentaire ? ";
     tabmess[38]:="PARCOURS";
     tabmess[39]:="DESIGNATION";
     tabmess[4Ø]:="Objet de la désignation";
     tabmess[41]:="PASSIF";
     tabmess[42]:="PSEUDO-PASSIF";
     tabmess[43]:="N'existe pas";
     tabmess[44]:="OK";
     tabmess[45]:="Veuillez introduire la disquette de l'utilisateur";
     tabmess[46]:="Cette disquette n'est pas la bonne...";
     tabmess[47]:="Veuillez introduire la disquette bibliothèque n°";
     tabmess[48]:="Le nombre maximum d'exercices est atteint ...";
```

```
      tabmess[48]:="Copies en cours ...";
      tabmess[50]:="Suite";
      tabmess[51]:="Abandon";
      tabmess[52]:="Page précédente";
      tabmess[53]:="Page suivante";
      tabmess[54]:="SUPPRESSION D'UN FICHER TRACE";
      tabmess[55]:="IMPRESSION D'UN FICHIER TRACE";
      tabmess[56]:="Type de renforcement";
      tabmess[57]:="Sonore";
      tabmess[58]:="Vocal";
      tabmess[59]:="Comparaison";
      tabmess[60]:="Clavier";
      tabmess[61]:="Type de la saisie du nom";
      tabmess[62]:="Nom de l'utilisateur + numéro de la session";
      tabmess[63]:="Veuillez préparer l'imprimante";
      tabmess[64]:="IMPRESSION D'UN FICHIER TRACE";
      tabmess[65]:="Aléatoire";
      tabmess[66]:="Abandon de la session";
      tabmess[67]:="Confirmez-vous l'enregistrement de cet exercice ?";
      tabmess[68]:="Nom de fichier inexistant";
      tabmess[69]:="Impression écran";
      tabmess[70]:="Impression imprimante";
      tabmess[71]:="AFFICHAGE DES COMMENTAIRES";
      tabmess[72]:="Commentaires";
      tabmess[73]:="Veuillez introduire la disquette InfoBiblio";
      tabmess[74]:="Pas de pacours présent dans la bibliothèque";
      tabmess[75]:="Pas de désignation présente dans la bibliothèque";
      tabmess[76]:="Veuillez patienter";
      tabmess[77]:="Veuillez introduire un nom s'il-vous-plaît?";
      tabmess[78]:="Enoncé verbal du parcours";
      tabmess[79]:="Suppression du fichier ";

END init_tabmess;

END varglobale.
```

```
MODULE parametrage;

FROM varglobale IMPORT init_tabmess,scr,(* *)tabex,tabparam,nbex;
FROM menu IMPORT menu_principal;

FROM SimpleScreens IMPORT CreateScreen;
FROM Intuition IMPORT CloseScreen;


BEGIN
  init_tabmess;
  scr:= CreateScreen(640,240,3,NIL);
  menu_principal;
  CloseScreen(scr^);

END parametrage.
```

```
IMPLEMENTATION MODULE menu;

FROM varglobale IMPORT string50,string30,tabmess,gl,scr,win,rp;
FROM saisiedonnees IMPORT saisie_donnees_init;
FROM ecran IMPORT demande_nom_fichier,ecran_9,prep_prt,ecran_8,
     impression_trace_ecran,ecran_11;
FROM utilitaire IMPORT centrage,init_couleur;

FROM Strings IMPORT ConcatString, CopyString;
FROM SYSTEM IMPORT ADR;
FROM AmigaDOS IMPORT Execute;
FROM Intuition IMPORT CloseWindow,IntuiMessagePtr,Gadget,Window,WindowFlags,
     WindowFlagsSet,IDCMPFlags,IDCMPFlagsSet;
FROM SimpleGadgets IMPORT FreeGadgetList,BeginGadgetList,EndGadgetList,
     AddGadgetTextButton;
FROM Drawing IMPORT Move,Draw;
FROM Ports IMPORT GetMsg;
FROM SimpleIDCMP IMPORT MsgData,WindowProc,ProcIMsg;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleWindows IMPORT CreateWindow;

CONST WIDCMP= IDCMPFlagsSet{GadgetUp};
      WFlags= WindowFlagsSet{};

VAR num: INTEGER;
    fin,ok: BOOLEAN;


(*****************************************************************************)

PROCEDURE choix_menu(VAR w : Window; VAR msg: MsgData; VAR gad:Gadget);

BEGIN
  CASE gad.GadgetID OF
       0: num:= 1;|
       1: num:= 2;|
       2: num:= 3;|
       3: num:= 4;|
  END;
  fin := TRUE;

END choix_menu;

(*****************************************************************************)

PROCEDURE saisie_menu(VAR choix : INTEGER);
(*****************************************************************************
*
* IN : /
*
* OUT: choix : choix du gadget;
*
* BUT : Crée l'écran du menu principal.
*
*****************************************************************************)

VAR wp : WindowProc;
    msg : IntuiMessagePtr;
    sig : SignalSet;

BEGIN
  BeginGadgetList();
  AddGadgetTextButton(100,60,ADR(tabmess[34]));
  AddGadgetTextButton(100,95,ADR(tabmess[55]));
  AddGadgetTextButton(100,130,ADR(tabmess[54]));
```

```
  AddGadgetTextButton(500,190,ADR(tabmess[5]));
  gl := EndGadgetList();
  win:=CreateWindow(0,0,640,240,WIDCMP,WFlags,gl,scr,NIL);
  rp:=win^.RPort;
  init_couleur;
  Move(rp^,0,20);
  Draw(rp^,640,20);
  centrage(rp,0,14,640,tabmess[1]);
  WITH wp DO
       procGadgetUp := choix_menu;
  END;
  fin := FALSE;
  WHILE (NOT fin) DO
        sig := Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
        LOOP
          msg := GetMsg(win^.UserPort^);
          IF msg = NIL
          THEN EXIT;
          END;
          ProcIMsg(wp,msg);
        END;
  END;
  choix := num;
  CloseWindow(win^);
  FreeGadgetList(gl^);

END saisie_menu;

(*************************************************************************************)

PROCEDURE menu_principal;

VAR
   ok : BOOLEAN;
   choix : INTEGER;
   nom_fichier : string30;
   abandon : BOOLEAN;
   name : string50;
   impression_ecran : BOOLEAN;

BEGIN
     ok:=TRUE;
     WHILE ok DO
           saisie_menu(choix);
           CASE choix OF
                1: saisie_donnees_init;|
                2: ecran_8(FALSE);
                   demande_nom_fichier(nom_fichier,abandon,"i");
                   IF NOT abandon
                      THEN
                          ecran_9(abandon,impression_ecran);
                          IF NOT abandon
                             THEN
                                 IF NOT impression_ecran
                                    THEN
                                        prep_prt;
                                        CopyString(name,'Copy ');
                                        ConcatString(name,nom_fichier);
                                        ConcatString(name,'.trace');
                                        ConcatString(name,' prt:');
                                        abandon:=Execute(ADR(name),NIL,NIL);
                                    ELSE
                                        impression_trace_ecran(nom_fichier);
                                    END;
                          END;
                   END; |
```

```
                3: ecran_8(FALSE);
                   demande_nom_fichier(nom_fichier,abandon,"s");
                   IF (NOT abandon)
                   THEN
                        ecran_11(nom_fichier);
                   END;
                4: ok := FALSE
            END;
        END;

END menu_principal;


END menu.
```

```
IMPLEMENTATION MODULE saisiedonnees;

FROM varglobale IMPORT abandontotal,abandonpartiel,nbex,tabex,tabparam,
     typerenfor,typesaisienom,num_cour_biblio,MAX_EX,nomuser,tabmess,
     gl,win,rp,scr,string5Ø,nbcommentairep,nbcommentaired,string3Ø,
     tabcommentairep,tabcommentaired;
FROM ecran IMPORT ecran_1,ecran_2,ecran_3,ecran_4,ecran_5,ecran_6,ecran_7,
     ecran_8,ecran_1Ø,impression_com,titre_ecran;
FROM basedonnees IMPORT lit_catalogue_desig,lit_catalogue_parc,
     verifiediskuser,copiefichier,ajout_exercice;
FROM utilitaire IMPORT centrage,init_couleur;

FROM Drawing IMPORT Draw, Move;
FROM Text IMPORT Text, TextLength;
FROM SimpleGadgets IMPORT AddGadgetTextButton, FreeGadgetList,
                    BeginGadgetList, EndGadgetList, AddGadgetInteger;
FROM Intuition IMPORT IDCMPFlags, IDCMPFlagsSet, WindowFlags,
     WindowFlagsSet, StringInfoPtr,  IntuiMessagePtr, Window,
     Gadget, GadgetPtr, CloseWindow, GadgetFlags, GadgetFlagsSet,
     ScreenToFront;
FROM SimpleIDCMP IMPORT WindowProc, MsgData, ProcIMsg;
FROM Strings IMPORT StringLength,ConcatString,CopyString,CompareStringCAP,
     equal;
FROM Tasks IMPORT SignalSet, Wait;
FROM Ports IMPORT GetMsg;
FROM SimpleWindows IMPORT CreateWindow;
FROM SYSTEM IMPORT ADR;
FROM Conversions IMPORT ConvNumberToString;
FROM AmigaDos IMPORT Execute,DeleteFile,IoErr;
FROM InOut IMPORT OpenOutputFile,OpenInputFile,CloseOutput,CloseInput,Done,
     ReadString,Read,ReadInt,WriteInt,WriteString,WriteLn,Write;
CONST
  WIDCMP = IDCMPFlagsSet{GadgetUp};
  WFlags = WindowFlagsSet{};

TYPE
  fichnouv = RECORD
       nom : string3Ø;
       acopier : BOOLEAN;
            END;

  fichanc = RECORD
       nom : string3Ø;
       efface : BOOLEAN;
            END;

VAR
  encore : BOOLEAN;
  fin : BOOLEAN;
  g1, g2, g3, g4, g5, g6, g7, g8, g9, g1Ø, g11, g12, g13, g14, g15,
  g16, g17 : GadgetPtr;
  valeurint : INTEGER;
  tabfichiers : ARRAY [1..15Ø] OF fichnouv;
  nbfichiercopies : INTEGER;
  tabfichiersanciens : ARRAY [1..15Ø] OF fichanc;
  nbfichiersanciens : INTEGER;



(*************************************************************************)

PROCEDURE selection_param_d(VAR w:Window;VAR msg:MsgData;VAR gad:Gadget);

VAR
   chaine : string5Ø;
```

```
    y : INTEGER;
    valeur : LONGINT;
    temp : StringInfoPtr;
    num_str : string50;

BEGIN

  CASE gad.GadgetID OF
       0 : tabparam[nbex][1] := 1;
           tabparam[nbex][4] := 2;
           chaine := tabmess[28];
           y := 35;
           g4^.Flags := GadgetFlagsSet{};
           g5^.Flags := GadgetFlagsSet{};
           g8^.Flags := GadgetFlagsSet{};
           IF (tabcommentaired[(valeurint)].syst <> 1) AND
              (tabcommentaired[(valeurint)].coul <> 1)
           THEN
                g13^.Flags := GadgetFlagsSet{};
                g14^.Flags := GadgetFlagsSet{};
                g9^.Flags  := GadgetFlagsSet{};
                g10^.Flags := GadgetFlagsSet{};

           END;|
       1 : tabparam[nbex][1] := 4;
           tabparam[nbex][4] := 2;
           chaine := tabmess[29];
           y := 35;
           g8^.Flags := GadgetFlagsSet{GadgDisabled};

           g4^.Flags := GadgetFlagsSet{};
           g5^.Flags := GadgetFlagsSet{};
           IF (tabcommentaired[(valeurint)].syst <> 1) AND
              (tabcommentaired[(valeurint)].coul <> 1)
           THEN
                g9^.Flags := GadgetFlagsSet{};
                g10^.Flags := GadgetFlagsSet{};
                g13^.Flags := GadgetFlagsSet{};
                g14^.Flags := GadgetFlagsSet{};

           END;
           Move(rp^,490,115);
           Text(rp^,ADR("                "),15);
           Move(rp^,490,115);
           Text(rp^,ADR('/'),1);|
       2 : tabparam[nbex][1] := 2;
           tabparam[nbex][4] := 2;
           chaine := tabmess[41];
           y := 35;
           g8^.Flags := GadgetFlagsSet{GadgDisabled};

           g5^.Flags := GadgetFlagsSet{};
           g6^.Flags := GadgetFlagsSet{};
           IF (tabcommentaired[(valeurint)].syst <> 1) AND
              (tabcommentaired[(valeurint)].coul <> 1)
           THEN
                g9^.Flags  := GadgetFlagsSet{};
                g10^.Flags := GadgetFlagsSet{};
                g13^.Flags := GadgetFlagsSet{};
                g14^.Flags := GadgetFlagsSet{};

           END;
           Move(rp^,490,115);
           Text(rp^,ADR("                "),15);
           Move(rp^,490,115);
           Text(rp^,ADR('/'),1);|
```

```
3 : tabparam[nbex][1] := 3;
    tabparam[nbex][2] := 1;
    tabparam[nbex][4] := Ø;
    tabparam[nbex][5] := 2;
    tabparam[nbex][7] := 2;
    chaine := tabmess[42];
    y := 35;
    g4^.Flags  := GadgetFlagsSet{GadgDisabled};
    g5^.Flags  := GadgetFlagsSet{GadgDisabled};
    g8^.Flags  := GadgetFlagsSet{GadgDisabled};
    g9^.Flags  := GadgetFlagsSet{GadgDisabled};
    gIØ^.Flags := GadgetFlagsSet{GadgDisabled};
    g13^.Flags := GadgetFlagsSet{GadgDisabled};
    g14^.Flags := GadgetFlagsSet{GadgDisabled};

    Move(rp^,49Ø,75);
    Text(rp^,ADR("                "),15);
    Move(rp^,49Ø,75);
    Text(rp^,ADR(tabmess[2]),StringLength(tabmess[2]));
    Move(rp^,49Ø,115);
    Text(rp^,ADR("                "),15);
    Move(rp^,49Ø,115);
    Text(rp^,ADR('/'),1);
    Move(rp^,49Ø,135);
    Text(rp^,ADR("                "),15);
    Move(rp^,49Ø,135);
    Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));
    Move(rp^,49Ø,175);
    Text(rp^,ADR("                "),15);
    Move(rp^,49Ø,175);
    Text(rp^,ADR(tabmess[9]),StringLength(tabmess[9]));¦
4 : tabparam[nbex][2] := 1;
    chaine := tabmess[2];
    y := 75;¦
5 : tabparam[nbex][2] := 2;
    chaine := tabmess[3];
    y := 75;¦
6 : tabparam[nbex][3] := 1;
    chaine := tabmess[3Ø];
    y :=95;¦
7 : tabparam[nbex][3] := 2;
    chaine := tabmess[31];
    y := 95;¦
8 : temp := gad.SpecialInfo;
    valeur := temp^.LongInt;
    IF (valeur<=LONGINT(Ø)) OR (valeur>LONGINT(5))
         THEN
             chaine:=tabmess[43];
             valeur:=2;
         ELSE
             ConvNumberToString(num_str,valeur,FALSE,1Ø,2,"Ø");
             chaine:=num_str;
      END;
    tabparam[nbex][3] := valeur;
    y := 115;¦
9 : tabparam[nbex][5] := 1;
    tabparam[nbex][7] := 2;
    chaine := tabmess[3Ø];
    y := 135;
    g13^.Flags := GadgetFlagsSet{GadgDisabled};
    g14^.Flags := GadgetFlagsSet{GadgDisabled};

    Move(rp^,49Ø,175);
    Text(rp^,ADR("                "),15);
    Move(rp^,49Ø,175);
    Text(rp^,ADR(tabmess[9]),StringLength(tabmess[9]));¦
```

```
10: tabparam[nbex][5] := 2;
    chaine := tabmess[31];
    IF tabcommentaired[(valeurint)].coul <> 1
       THEN
         g13^.Flags := GadgetFlagsSet{};
         g14^.Flags := GadgetFlagsSet{};
       END;
    y := 135;¦
11: tabparam[nbex][6] := 1;
    chaine := tabmess[30];
    y := 155;¦
12: tabparam[nbex][6] := 2;
    chaine := tabmess[31];
    y := 155;¦
13: tabparam[nbex][7] := 1;
    tabparam[nbex][1] := 1;
    tabparam[nbex][5] := 2;
    chaine := tabmess[8];
    y := 175;
    g9^.Flags := GadgetFlagsSet{GadgDisabled};
    g10^.Flags := GadgetFlagsSet{GadgDisabled};

    Move(rp^,490,35);
    Text(rp^,ADR("                    "),15);
    Move(rp^,490,35);
    Text(rp^,ADR(tabmess[28]),StringLength(tabmess[28]));
    Move(rp^,490,135);
    Text(rp^,ADR("                    "),15);
    Move(rp^,490,135);
    Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));¦
14: tabparam[nbex][7] := 2;
    chaine := tabmess[9];
    IF (tabcommentaired[(valeurint)].syst <> 1) AND
       (tabcommentaired[(valeurint)].coul <> 1)
    THEN
      g9^.Flags := GadgetFlagsSet{};
      g10^.Flags := GadgetFlagsSet{};
    END;
    y := 175;¦
15: tabparam[nbex][8] := 1;
    tabparam[nbex][5] := 2;
    chaine := tabmess[21];
    y := 195;

    g9^.Flags := GadgetFlagsSet{GadgDisabled};
    g10^.Flags := GadgetFlagsSet{GadgDisabled};

    Move(rp^,490,135);
    Text(rp^,ADR("                    "),15);
    Move(rp^,490,135);
    Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));¦

16: tabparam[nbex][8] := 2;
    chaine := tabmess[22];
    IF (tabcommentaired[(valeurint)].coul <> 1) AND
       (tabcommentaired[(valeurint)].syst <> 1)
    THEN
      g9^.Flags := GadgetFlagsSet{};
      g10^.Flags := GadgetFlagsSet{};
    END;
    y := 195;¦
17: temp:=gad.SpecialInfo;
    valeur:=temp^.LongInt;
    valeurint := valeur;
    IF (valeur <= LONGINT(nbcommentaired)) AND (valeur > LONGINT(0))
    THEN
```

```
IF tabcommentaired[valeurint].syst = 1
THEN
   tabparam[nbex][5] := 2;
   tabparam[nbex][7] := 2;
   tabparam[nbex][8] := 2;
   g9^.Flags  := GadgetFlagsSet{GadgDisabled};
   g10^.Flags := GadgetFlagsSet{GadgDisabled};
   g13^.Flags := GadgetFlagsSet{GadgDisabled};
   g14^.Flags := GadgetFlagsSet{GadgDisabled};
   g15^.Flags := GadgetFlagsSet{GadgDisabled};
   g16^.Flags := GadgetFlagsSet{GadgDisabled};

   g11^.Flags := GadgetFlagsSet{};
   g12^.Flags := GadgetFlagsSet{};

   Move(rp^,490,135);
   Text(rp^,ADR("                "),15);
   Move(rp^,490,135);
   Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));
   Move(rp^,490,175);
   Text(rp^,ADR("                "),15);
   Move(rp^,490,175);
   Text(rp^,ADR(tabmess[9]),StringLength(tabmess[9]));
   Move(rp^,490,195);
   Text(rp^,ADR("                "),15);
   Move(rp^,490,195);
   Text(rp^,ADR(tabmess[22]),StringLength(tabmess[22]));
ELSE
     tabparam[nbex][6] := 2;
     g11^.Flags := GadgetFlagsSet{GadgDisabled};
     g12^.Flags := GadgetFlagsSet{GadgDisabled};

     IF tabparam[nbex][1] # 3
     THEN
       g13^.Flags := GadgetFlagsSet{};
       g14^.Flags := GadgetFlagsSet{};
     END;
     g15^.Flags := GadgetFlagsSet{};
     g16^.Flags := GadgetFlagsSet{};

     Move(rp^,490,155);
     Text(rp^,ADR("                 "),15);
     Move(rp^,490,155);
     Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));
     IF tabcommentaired[valeurint].coul = 1
     THEN
        tabparam[nbex][1] := 1;
        tabparam[nbex][7] := 1;
        tabparam[nbex][5] := 2;

        g3^.Flags := GadgetFlagsSet{GadgDisabled};
        g9^.Flags := GadgetFlagsSet{GadgDisabled};
        g10^.Flags := GadgetFlagsSet{GadgDisabled};
        g13^.Flags := GadgetFlagsSet{GadgDisabled};
        g14^.Flags := GadgetFlagsSet{GadgDisabled};

        g4^.Flags := GadgetFlagsSet{};
        g5^.Flags := GadgetFlagsSet{};
        g8^.Flags := GadgetFlagsSet{};

        Move(rp^,490,35);
        Text(rp^,ADR("                 "),15);
        Move(rp^,490,35);
        Text(rp^,ADR(tabmess[28]),StringLength(tabmess[28]));
        Move(rp^,490,135);
        Text(rp^,ADR("                 "),15);
```

```
                        Move(rp^,490,135);
                        Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));
                        Move(rp^,490,175);
                        Text(rp^,ADR("                "),15);
                        Move(rp^,490,175);
                        Text(rp^,ADR(tabmess[8]),StringLength(tabmess[8]));
                     ELSE
                        g3^.Flags := GadgetFlagsSet{};
                        IF tabparam[nbex][1] # 3
                        THEN
                           IF tabparam[nbex][8] # 1
                           THEN
                             g9^.Flags  := GadgetFlagsSet{};
                             g10^.Flags := GadgetFlagsSet{};
                           END;
                           g13^.Flags := GadgetFlagsSet{};
                           g14^.Flags := GadgetFlagsSet{};
                        END;
                     END;
                  END;
                  ConvNumberToString(num_str,valeur,FALSE,10,2,"0");
                  chaine:=num_str;
               ELSE
                  chaine := tabmess[43];
                  valeurint := 1;
               END;
               tabex[nbex].typeex := 'd';
               tabex[nbex].numbiblio := tabcommentaired[valeurint].numbiblio;
               tabex[nbex].numcat := valeurint;
               y := 215;  |
         18:  impression_com ('d');|
         19: fin := TRUE;|
   END;
   IF ((gad.GadgetID#18) AND (gad.GadgetID#19))
   THEN
              Move(rp^,490,y);
              Text(rp^,ADR("                "),15);
              Move(rp^,490,y);
              Text(rp^,ADR(chaine),StringLength(chaine));

   END;


END selection_param_d;

(*****************************************************************************)

PROCEDURE saisie_desig;

(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Elle gère la saisie des paramètres d'un exercice de type désignation.
*       Elle met à jour le tableau tab_param.
*
*****************************************************************************)

CONST
     WIDCMP=IDCMPFlagsSet{GadgetUp};
     WFlags=WindowFlagsSet{};

VAR
   wp : WindowProc;
```

```
    sig : SignalSet;
    msg : IntuiMessagePtr;
    compteur : INTEGER;

BEGIN
    (* Initialisation des valeurs prises par defaut *)
    compteur:=1;
    WHILE compteur<=8 DO
          tabparam[nbex][compteur]:=1;
          compteur:=compteur+1;
    END;
    tabparam[nbex][4]:=2;
    tabparam[nbex][5]:=2;
    tabparam[nbex][6]:=2;
    tabparam[nbex][7]:=2;
    tabparam[nbex][8]:=2;
    valeurint := 1;
    tabex[nbex].typeex := 'd';
    tabex[nbex].numbiblio := 1;
    tabex[nbex].numcat := 1;
    (* Initialisation de la procédure qui gère les gadgets *)
    WITH wp DO
         procGadgetUp:=selection_param_d;
    END;
    (* Creation des gadgets de saisie des paramètres de la désignation *)
    BeginGadgetList();

      (* Type d'interaction *)
      AddGadgetTextButton(272,30,ADR(tabmess[28]));
      AddGadgetTextButton(272,50,ADR(tabmess[29]));
      AddGadgetTextButton(407,30,ADR(tabmess[41]));
      AddGadgetTextButton(365,50,ADR(tabmess[42]));

      (* Ordre d'énumération logique *)
      AddGadgetTextButton(290,70,ADR(tabmess[2]));
      AddGadgetTextButton(380,70,ADR(tabmess[3]));

      (* Enoncé vocal *)
      AddGadgetTextButton(320,90,ADR(tabmess[30]));
      AddGadgetTextButton(380,90,ADR(tabmess[31]));

      (* Nombre d'iterations *)
      AddGadgetInteger(360,110,3,3,2);

      (* Silhouette vide ou non *)
      AddGadgetTextButton(320,130,ADR(tabmess[30]));
      AddGadgetTextButton(380,130,ADR(tabmess[31]));

      (* Animation des systèmes ou non *)
      AddGadgetTextButton(320,150,ADR(tabmess[30]));
      AddGadgetTextButton(380,150,ADR(tabmess[31]));

      (* Type de désignation: couleur/cadre *)
      AddGadgetTextButton(290,170,ADR(tabmess[8]));
      AddGadgetTextButton(380,170,ADR(tabmess[9]));

      (* Objet de la désignation : nom/image *)
      AddGadgetTextButton(320,190,ADR(tabmess[21]));
      AddGadgetTextButton(380,190,ADR(tabmess[22]));

      (* Choix de la liste des énonces *)
      AddGadgetInteger(320,210,3,3,1);

      (* Gadget servant pour l'affichage des commentaires **)
      AddGadgetTextButton(360,210,ADR(tabmess[72]));
```

```
  * Fin *)
  AddGadgetTextButton(350,225,ADR(tabmess[10]));

gl:=EndGadgetList();
g1:=gl^.NextGadget;g2:=g1^.NextGadget;g3:=g2^.NextGadget;
g4:=g3^.NextGadget;g5:=g4^.NextGadget;g6:=g5^.NextGadget;
g7:=g6^.NextGadget;g8:=g7^.NextGadget;g9:=g8^.NextGadget;
g10:=g9^.NextGadget;g11:=g10^.NextGadget;g12:=g11^.NextGadget;
g13:=g12^.NextGadget;g14:=g13^.NextGadget;g15:=g14^.NextGadget;
g16:=g15^.NextGadget;g17:=g16^.NextGadget;

(* Création de la fenêtre => Affichage des gadgets *)
win:=CreateWindow(0,0,640,240,WIDCMP,WFlags,gl,scr,NIL);
rp:=win^.RPort;
(* désallocation des gadgets d'animation des systèmes *)
g11^.Flags := GadgetFlagsSet{GadgDisabled};
g12^.Flags := GadgetFlagsSet{GadgDisabled};
(* Dessin de l'éécran*)
Move(rp^,0,20);
Draw(rp^,640,20);
Move(rp^,265,20);
Draw(rp^,265,240);
Move(rp^,480,20);
Draw(rp^,480,240);
(* Centrage et affichage du titre *)
centrage(rp,0,14,640,tabmess[12]);

(* Affichage des commentaires dans la colonne de gauche *)
Move(rp^,9,35);
Text(rp^,ADR(tabmess[16]),StringLength(tabmess[16]));
Move(rp^,9,75);
Text(rp^,ADR(tabmess[17]),StringLength(tabmess[17]));
Move(rp^,9,95);
Text(rp^,ADR(tabmess[14]),StringLength(tabmess[14]));
Move(rp^,9,115);
Text(rp^,ADR(tabmess[15]),StringLength(tabmess[15]));
Move(rp^,9,135);
Text(rp^,ADR(tabmess[18]),StringLength(tabmess[18]));
Move(rp^,9,155);
Text(rp^,ADR(tabmess[19]),StringLength(tabmess[19]));
Move(rp^,9,175);
Text(rp^,ADR(tabmess[20]),StringLength(tabmess[20]));
Move(rp^,9,195);
Text(rp^,ADR(tabmess[40]),StringLength(tabmess[40]));
Move(rp^,9,215);
Text(rp^,ADR(tabmess[7]),StringLength(tabmess[7]));

(* Affichage des commentaires dans la colonne de droite *)
Move(rp^,490,35);
Text(rp^,ADR(tabmess[28]),StringLength(tabmess[28]));
Move(rp^,490,75);
Text(rp^,ADR(tabmess[2]),StringLength(tabmess[2]));
Move(rp^,490,95);
Text(rp^,ADR(tabmess[30]),StringLength(tabmess[30]));
Move(rp^,490,115);
Text(rp^,ADR('02'),LONGINT(2));
Move(rp^,490,135);
Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));
Move(rp^,490,155);
Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));
Move(rp^,490,175);
Text(rp^,ADR(tabmess[9]),StringLength(tabmess[9]));
Move(rp^,490,195);
Text(rp^,ADR(tabmess[22]),StringLength(tabmess[22]));
Move(rp^,490,215);
Text(rp^,ADR('01'),LONGINT(2));
```

```
    (* Gestion des gadgets proprements dits *)
    fin:=FALSE;
    WHILE NOT fin DO
          sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
          LOOP
              msg:=GetMsg(win^.UserPort^);
              IF msg=NIL
                 THEN EXIT;
              END;
              ProcIMsg(wp,msg);
          END;
    END;
    CloseWindow(win^);
    FreeGadgetList(gl^);

END saisie_desig;

(*****************************************************************************)


PROCEDURE selection_param_p(VAR w:Window;VAR msg:MsgData;VAR gad:Gadget);

VAR
   chaine : string5Ø;
   y : INTEGER;
   valeur : LONGINT;
   temp : StringInfoPtr;
   num_str : string5Ø;

BEGIN

  CASE gad.GadgetID OF
       Ø : tabparam[nbex][1] := 1;

           chaine := tabmess[28];
           y := 35;¦
       1 : tabparam[nbex][1] := 2;
           chaine := tabmess[29];
           y := 35;¦
       2 : tabparam[nbex][2] := 1;
           chaine := tabmess[3Ø];
           y := 55;¦
       3 : tabparam[nbex][2] := 2;
           chaine := tabmess[31];
           y := 55;¦
       4 : tabparam[nbex][3] := 1;
           chaine := tabmess[3Ø];
           y := 75;¦
       5 : tabparam[nbex][3] := 2;
           chaine := tabmess[31];
           y := 75;¦
       6 : tabparam[nbex][4] := 1;
           chaine := tabmess[27];
           y :=95;¦
       7 : tabparam[nbex][4] := 2;
           chaine := tabmess[32];
           y := 95;¦
       8 : tabparam[nbex][4] := 3;
           chaine := tabmess[33];
           y := 95;¦
       9 : tabparam[nbex][5] := 1;
           chaine := tabmess[3Ø];
           y := 135;¦
       1Ø: tabparam[nbex][5] := 2;
```

```
            chaine := tabmess[31];
            y := 135;¦
        11: tabparam[nbex][6] := 1;
            chaine := tabmess[30];
            y := 155;¦
        12: tabparam[nbex][6] := 2;
            chaine := tabmess[31];
            y := 155;¦

        13: temp:=gad.SpecialInfo;
            valeur:=temp^.LongInt;
            valeurint := valeur;
            y:= 175;
            IF (valeur <= LONGINT(nbcommentairep)) AND (valeur > LONGINT(0))
            THEN
               ConvNumberToString(num_str,valeur,FALSE,10,2,"0");
               chaine:=num_str;

            ELSE
               chaine := tabmess[43];
               valeurint := 1;
            END;
            tabex[nbex].typeex := 'p';
            tabex[nbex].numbiblio := tabcommentairep[valeurint].numbiblio;
            tabex[nbex].numcat := valeurint;¦
        14:  impression_com ('p');¦
        15: fin := TRUE;¦
    END;
    IF ((gad.GadgetID#14) AND (gad.GadgetID#15))
    THEN
             Move(rp^,490,y);
             Text(rp^,ADR("                "),15);
             Move(rp^,490,y);
             Text(rp^,ADR(chaine),StringLength(chaine));

    END;


END selection_param_p;

(*****************************************************************************)

PROCEDURE saisie_parc;

(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Gère la saisie des valeurs des paramètres d'un exercice de type
*       parcours. Elle met à jour le tableau tab_param.
*
*****************************************************************************)
CONST
     WIDCMP=IDCMPFlagsSet{GadgetUp};
     WFlags=WindowFlagsSet{};

VAR
   wp : WindowProc;
   sig : SignalSet;
   msg : IntuiMessagePtr;
   compteur : INTEGER;

BEGIN
     (* Initialisation des valeurs prises par defaut *)
```

```
compteur:=1;
WHILE compteur<=6 DO
      tabparam[nbex][compteur]:=1;
      compteur:=compteur+1;
END;
tabparam[nbex][2]:=2;
tabparam[nbex][3]:=2;
tabparam[nbex][5]:=2;
valeurint := 1;
tabex[nbex].typeex := 'p';
tabex[nbex].numbiblio := 1;
tabex[nbex].numcat := 1;
(* Initialisation de la procédure qui gère les gadgets *)
WITH wp DO
     procGadgetUp:=selection_param_p;
END;

(* Creation des gadgets de saisie des parametres parcours *)
BeginGadgetList();
  (* Type d'interaction *)
  AddGadgetTextButton(270,30,ADR(tabmess[28]));
  AddGadgetTextButton(388,30,ADR(tabmess[29]));
  (* Parcours au début *)
  AddGadgetTextButton(320,50,ADR(tabmess[30]));
  AddGadgetTextButton(380,50,ADR(tabmess[31]));
  (* Point de départ et d' arrivée *)
  AddGadgetTextButton(320,70,ADR(tabmess[30]));
  AddGadgetTextButton(380,70,ADR(tabmess[31]));
  (* Feed_back *)
  AddGadgetTextButton(272,90,ADR(tabmess[27]));
  AddGadgetTextButton(380,90,ADR(tabmess[32]));
  AddGadgetTextButton(350,110,ADR(tabmess[33]));
  (* Parcours en fin *)
  AddGadgetTextButton(320,130,ADR(tabmess[30]));
  AddGadgetTextButton(380,130,ADR(tabmess[31]));
  (* Enoncé vocal *)
  AddGadgetTextButton(320,150,ADR(tabmess[30]));
  AddGadgetTextButton(380,150,ADR(tabmess[31]));
  (* Choix de la liste des énonces *)
  AddGadgetInteger(320,170,3,3,1);
  (* Gadget servant pour l'affichage des commentaires **)
  AddGadgetTextButton(360,170,ADR(tabmess[72]));
  (* OK *)
  AddGadgetTextButton(350,195,ADR(tabmess[10]));

gl:=EndGadgetList();


(* Creation de la fenetre => Affichage des gadgets *)
win:=CreateWindow(0,0,640,240,WIDCMP,WFlags,gl,scr,NIL);
rp:=win^.RPort;

(* Dessin de l'eécran*)
Move(rp^,0,20);
Draw(rp^,640,20);
Move(rp^,265,20);
Draw(rp^,265,240);
Move(rp^,480,20);
Draw(rp^,480,240);
(* Centrage et affichage du titre *)
centrage(rp,0,14,640,tabmess[12]);

(* Affichage des commentaires dans la colonne de gauche *)
Move(rp^,9,35);
Text(rp^,ADR(tabmess[16]),StringLength(tabmess[16]));
Move(rp^,9,55);
```

```
        Text(rp^,ADR(tabmess[23]),StringLength(tabmess[23]));
        Move(rp^,9,75);
        Text(rp^,ADR(tabmess[24]),StringLength(tabmess[24]));
        Move(rp^,9,95);
        Text(rp^,ADR(tabmess[25]),StringLength(tabmess[25]));
        Move(rp^,9,135);
        Text(rp^,ADR(tabmess[26]),StringLength(tabmess[26]));
        Move(rp^,9,155);
        Text(rp^,ADR(tabmess[78]),StringLength(tabmess[78]));
        Move(rp^,9,175);
        Text(rp^,ADR(tabmess[7]),StringLength(tabmess[7]));


        (* Affichage des commentaires dans la colonne de droite *)
        Move(rp^,490,35);
        Text(rp^,ADR(tabmess[28]),StringLength(tabmess[28]));
        Move(rp^,490,55);
        Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));
        Move(rp^,490,75);
        Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));
        Move(rp^,490,95);
        Text(rp^,ADR(tabmess[27]),StringLength(tabmess[27]));
        Move(rp^,490,135);
        Text(rp^,ADR(tabmess[31]),StringLength(tabmess[31]));
        Move(rp^,490,155);
        Text(rp^,ADR(tabmess[30]),StringLength(tabmess[30]));
        Move(rp^,490,175);
        Text(rp^,ADR('01'),LONGINT(2));


        (* Gestion des gadgets propres dits *)
        fin:=FALSE;
        WHILE NOT fin DO
              sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
              LOOP
                  msg:=GetMsg(win^.UserPort^);
                  IF msg=NIL
                     THEN EXIT;
                  END;
                  ProcIMsg(wp,msg);
              END;
        END;
        CloseWindow(win^);
        FreeGadgetList(gl^);

END saisie_parc;



(*****************************************************************************)

PROCEDURE lire_liste_fichiers;
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Elle saisit toutes les données de la session et place les  fichiers
*       dans ram avant de les recopier.
*
*****************************************************************************)

VAR
   correct : BOOLEAN;
   compteur : INTEGER;
```

```
BEGIN
     correct:=FALSE;
     ecran_8(FALSE);
     verifiediskuser(correct);
     WHILE NOT correct DO
           ecran_8(TRUE);
           verifiediskuser(correct);
     END;
     nbfichiersanciens:=Ø;
     OpenInputFile('User:fichiers.liste');
     IF IoErr()#LONGINT(Ø)
        THEN ScreenToFront(scr^);
     END;
     IF NOT Done
        THEN
            CloseInput;
            RETURN;
     END;
     ReadInt(nbfichiersanciens);
     compteur:=1;
     WHILE compteur<=nbfichiersanciens DO
           ReadString(tabfichiersanciens[compteur].nom);
           tabfichiersanciens[compteur].efface:=TRUE;
           compteur:=compteur+1;
     END;
     CloseInput;
END lire_liste_fichiers;

(*****************************************************************************)

PROCEDURE trt_copie_fichier(nomin,nomout : string3Ø);
(*****************************************************************************
*
* IN : /
*
* OUT: /
*
* BUT : Elle copie fichier nomin dans nomout si le fichier nomout
*       n'existe pas dans la   contenant  ces sur la disquette USER.
*
*****************************************************************************)

VAR
   compteur : INTEGER;
   nomfichiernouveau : string3Ø;

BEGIN
     CopyString(nomfichiernouveau,nomout);
     nomfichiernouveau[1]:='u';
     nomfichiernouveau[2]:='s';
     nomfichiernouveau[3]:='e';
     nomfichiernouveau[4]:='r';
     nomfichiernouveau[5]:=':';
     compteur:=1;
     WHILE compteur<=nbfichiersanciens DO
           IF CompareStringCAP(tabfichiersanciens[compteur].nom,
                               nomfichiernouveau)=equal
              THEN
                   tabfichiersanciens[compteur].efface:=FALSE;
                   nbfichiercopies:=nbfichiercopies+1;
                   CopyString(tabfichiers[nbfichiercopies].nom,nomout);
                   tabfichiers[nbfichiercopies].acopier:=FALSE;
                   RETURN;
              ELSE compteur:=compteur+1;
           END;
```

```
      END;
      compteur:=1;
      WHILE compteur<=nbfichiercopies DO
            IF CompareStringCAP(tabfichiers[compteur].nom,nomout)=equal
               THEN RETURN;
               ELSE compteur:=compteur+1;
            END;
      END;
      copiefichier(nomin,nomout);
      nbfichiercopies:=nbfichiercopies+1;
      CopyString(tabfichiers[nbfichiercopies].nom,nomout);
      tabfichiers[nbfichiercopies].acopier:=TRUE;
END trt_copie_fichier;

(*****************************************************************************)


PROCEDURE copie_ram;

VAR i,j,k: INTEGER;
    nomfich,tampon,nomfich1,nomfich2:string30;
    enreg: ARRAY [1..15] OF CHAR;
    car: CHAR;
    numstr: ARRAY [0..2] OF CHAR;

BEGIN
  lire_liste_fichiers;
  nbfichiercopies := 0;
  IF num_cour_biblio # 1
  THEN ecran_10(1);
       num_cour_biblio := 1;
  END;
  win := CreateWindow(0,0,640,240,WIDCMP,WFlags,NIL,scr,NIL);
  rp := win^.RPort;
  init_couleur;
  (* dessin de l'écran *)
  Move( rp^,0,20);
  Draw(rp^,640,20);
  titre_ecran(34);
  centrage(rp,0,128,640,tabmess[76]);
  centrage(rp,0,148,640,tabmess[49]);
  OpenOutputFile('RAM: liste_exercice');
  IF IoErr() # LONGINT(0)
  THEN ScreenToFront(scr^);
  END;
  CloseInput;
  i:=1;
  WHILE i<=nbex DO
    ajout_exercice(i);
    i:= i+1;
  END;
  CloseWindow(win^);
  (* copie des fichiers des sons et des images *)
  i:=1;
  WHILE i<= nbex DO
    IF tabex[i].numbiblio # num_cour_biblio
    THEN ecran_10(tabex[i].numbiblio);
         num_cour_biblio := tabex[i].numbiblio;
    END;
    win := CreateWindow(0,0,640,240,WIDCMP,WFlags,NIL,scr,NIL);
    rp := win^.RPort;
    init_couleur;
    (* dessin de l'écran *)
    Move( rp^,0,20);
    Draw(rp^,640,20);
    titre_ecran(34);
```

```
centrage(rp,Ø,128,64Ø,tabmess[76]);
centrage(rp,Ø,148,64Ø,tabmess[49]);

CopyString(nomfich1,"bibio");
ConvNumberToString (numstr,LONGINT(tabex[i].numbiblio),FALSE,1Ø,2,"Ø");
ConcatString(nomfich1,numstr);
ConcatString(nomfich1,":");
CopyString(tampon,nomfich1);
CopyString(nomfich2,"RAM: ");
CopyString(nomfich,tampon);
ConcatString(nomfich,"liste_son");
ConcatString(nomfich,tabex[i].typeex);
ConcatString(nomfich,numstr);
OpenInputFile(nomfich);
IF IoErr() # LONGINT(Ø)
THEN ScreenToFront(scr^);
END;
ReadString(enreg);
WHILE Done DO
  ConcatString(nomfich1,enreg);
  ConcatString(nomfich2,enreg);
  trt_copie_fichier(nomfich1,nomfich2);
  ReadString(enreg);
  CopyString(nomfich1,tampon);
  CopyString(nomfich2,"RAM: ");
END;

CopyString(nomfich,tampon);
ConcatString(nomfich,"liste_image");
ConcatString(nomfich,tabex[i].typeex);
ConcatString(nomfich,numstr);
OpenInputFile(nomfich);
IF IoErr() # LONGINT(Ø)
THEN ScreenToFront(scr^);
END;
ReadString(enreg);
WHILE Done DO
  ConcatString(nomfich1,enreg);
  ConcatString(nomfich2,enreg);
  trt_copie_fichier(nomfich1,nomfich2);
  ReadString(enreg);
  CopyString(nomfich1,tampon);
  CopyString(nomfich2,"RAM: ");
END;
CloseInput;
(* copie des fichiers du ièmme exercice de liste exercice *)
CopyString(nomfich1,tampon);
CopyString(nomfich2,"RAM: ");
OpenInputFile("RAM:liste_exercice");
j:= 1;
WHILE j< nbex DO
  Read(car);
  ReadString(enreg);
  ReadString(enreg);
  ReadString(enreg);
  ReadString(enreg);
  j:= j+1;
END;
j:= 1;
Read(car);
WHILE j<= 2 DO
  ReadString(enreg);
  ConcatString(nomfich1,enreg);
  ConcatString(nomfich2,enreg);
  trt_copie_fichier(nomfich1,nomfich2);
  CopyString(nomfich1,tampon);
```

```
      CopyString(nomfich2,"RAM:");
      j:=j+1;
    END;
    IF (car = "p") OR (car = "P")
    THEN ReadString(enreg);
         ConcatString(nomfich1,enreg);
         ConcatString(nomfich2,enreg);
         trt_copie_fichier(nomfich1,nomfich2);
         CopyString(nomfich1,tampon);
         CopyString(nomfich2,"RAM: ");
    END;
    i:= i+1;
  END;
  CloseWindow(win^);

END copie_ram;

(*****************************************************************************)

PROCEDURE copie_user;

VAR i,j,compteur:INTEGER;
    efface : BOOLEAN;
    nomfichier : string30;
    numstr : ARRAY [0..2] OF CHAR;
BEGIN
    win := CreateWindow(0,0,640,240,WIDCMP,WFlags,NIL,scr,NIL);
    rp := win^.RPort;
    init_couleur;
    (* dessin de l'écran *)
    Move( rp^,0,20);
    Draw(rp^,640,20);
    titre_ecran(34);
    centrage(rp,0,128,640,tabmess[76]);
    centrage(rp,0,148,640,tabmess[49]);
    (* effacement des fichiers devenus inutiles sur "user" *)
    compteur := 1;
    WHILE compteur<= nbfichiersanciens DO
      IF tabfichiersanciens[compteur].efface
      THEN efface := DeleteFile(ADR(tabfichiersanciens[compteur].nom));
      END;
      compteur := compteur +1;
    END;
    (* effacement des fichiers parametres *)
    efface := Execute(ADR("Delete user:param#?"),NIL,NIL);
    (* création et copie sur fichier user des paramètres *)
    compteur := 1;
    WHILE compteur <= nbex DO
      CopyString(nomfichier,"user:param");
      ConcatString(nomfichier,tabex[compteur].typeex);
      ConvNumberToString(numstr,ADR(tabex[compteur].numcat),FALSE,10,2,"0");
      ConcatString(nomfichier,numstr);
      OpenOutputFile(nomfichier);
      IF IoErr()#LONGINT(0)
        THEN ScreenToFront(scr^);
      END;
      j:= 1;
      WHILE j<= 6 DO
        WriteInt(tabparam[compteur][j],1);
        Write(" ");
        j:= j+1;
      END;
      IF (tabex[compteur].typeex = "d") OR
         (tabex[compteur].typeex = "D")
      THEN WriteInt(tabparam[compteur][7],1);
           Write(" ");
```

```
        WriteInt(tabparam[compteur][d],1 );
    END;
    CloseInput;
    compteur:= compteur+1;
END;
(* copie de liste exercice *)
efface := Execute(ADR("Copy RAM:liste_exercice to user:"),NIL,NIL);
(* création et copie des données session *)
OpenOutputFile("user:donnees_session");
IF IoErr()#LONGINT(Ø)
THEN ScreenToFront(scr^);
END;
WriteString(nomuser);
WriteLn;
Write(typesaisienom);
WriteLn;
WriteInt(typerenfor,1);
WriteLn;
WriteInt(1,1);
WriteLn;
WriteInt(Ø,1);
CloseInput;
(* copie des fichiers nécessaires sur user *)
i:= 1;
WHILE i<= nbfichiercopies DO
  CopyString(nomfichier,tabfichiers[i].nom);
  nomfichier[1] := 'u';
  nomfichier[2] := 's';
  nomfichier[3] := 'e';
  nomfichier[4] := 'r';
  nomfichier[5] := ':';
  IF tabfichiers[i].acopier
  THEN copiefichier(tabfichiers[i].nom,nomfichier);
  END;
  tabfichiers[i].nom:= nomfichier;
  i:= i+1;
END;
(* écriture du fichier contenant tous les fichiers de user *)
OpenOutputFile("user: fichiers.liste");
IF IoErr()#LONGINT(Ø)
THEN ScreenToFront(scr^);
END;
WriteInt(nbfichiercopies,4);
WriteLn;
i:= 1;
WHILE i<= nbfichiercopies DO
  WriteString(tabfichiers[i].nom);
  WriteLn;
  i:= i+1;
END;
CloseOutput;
(* créer <nomuser>.version s'il n'existe pas *)
CopyString(nomfichier,"user:");
ConcatString(nomfichier,nomuser);
ConcatString(nomfichier,".version");
OpenInputFile(nomfichier);
IF (NOT Done)
THEN CloseInput;
     OpenOutputFile(nomfichier);
     WriteInt(1,1);
     CloseOutput;
ELSE CloseInput;
END;
(* effacement de la mémoire RAM *)
efface := Execute(ADR("Delete RAM:#?"),NIL,NIL);
```

```
END copie_user;

(*****************************************************************************)

PROCEDURE saisie_donnees_init;

VAR
  fin,correct : BOOLEAN;
  type_ex : CHAR;

BEGIN
  abandontotal := FALSE;
  abandonpartiel := FALSE;
  ecran_8(FALSE);
  nomuser := "";
  ecran_1;
  IF abandontotal
  THEN RETURN;
  END;
  fin:= FALSE;
  ecran_10(1);
  num_cour_biblio:= 1;
  nbex := 0;
  lit_catalogue_desig;
  lit_catalogue_parc;
  WHILE (NOT fin) DO
        nbex := nbex+ 1;
        ecran_2(type_ex);
        IF abandontotal
        THEN RETURN;
        END;
        tabex[nbex].typeex := type_ex;
        IF type_ex ='d'
        THEN saisie_desig;
        ELSE saisie_parc;
        END;
        ecran_3;
        IF abandontotal
        THEN RETURN;
        END;
        IF abandonpartiel
        THEN nbex := nbex- 1;
        END;
        IF nbex = MAX_EX
        THEN fin := TRUE;
             ecran_4;
        ELSE ecran_5(encore);
             IF abandontotal
             THEN RETURN;
             ELSE fin := (NOT encore);
             END;
        END;
  END;
  IF nbex = 0
  THEN RETURN;
  END;
  ecran_6;
  IF abandontotal THEN RETURN; END;
  ecran_7;
  IF abandontotal THEN RETURN; END;
  (*copie_ram;
  ecran_8(FALSE);
  num_cour_biblio := 0;
  correct:= FALSE;
  verifiediskuser(correct);
  WHILE NOT correct DO
```

```
        ecran_8(TRUE);
        verifiediskuser(correct);
    END;
    copie_user;*)


END saisie_donnees_init;


END saisiedonnees.
```

```
IMPLEMENTATION MODULE ecran;

FROM varglobale IMPORT abandontotal,abandonpartiel,string30,gl,win,rp,nomuser,
     typerenfor,typesaisienom,tabmess,scr,nbcommentaired,nbcommentairep,
     tabcommentaired,tabcommentairep,string70,string50;
FROM basedonnees IMPORT cherchernomuser;
FROM utilitaire IMPORT centrage, verifiefich,init_couleur;

FROM SYSTEM IMPORT ADR,BYTE;
FROM Conversions IMPORT ConvNumberToString;
FROM Drawing IMPORT Move, Draw, SetAPen, RectFill;
FROM Text IMPORT Text, TextLength;
FROM Strings IMPORT StringLength, ConcatString, CopyString,CompareString,
     DeleteSubString,equal;
FROM Intuition IMPORT IntuiMessagePtr, CloseWindow, Window, Gadget, IDCMPFlags,
     IDCMPFlagsSet, WindowFlags, WindowFlagsSet, StringInfoPtr,ActivateGadget,
     ScreenPtr, WindowPtr, GadgetPtr,DrawImage,RefreshGadgets,
     CloseScreen, Image,ScreenToFront;
FROM SimpleIDCMP IMPORT WindowProc, ProcIMsg, MsgData;
FROM Ports IMPORT GetMsg;
FROM Tasks IMPORT SignalSet, Wait;
FROM SimpleGadgets IMPORT BeginGadgetList, EndGadgetList, AddGadgetString,
                  AddGadgetInteger, AddGadgetTextButton, FreeGadgetList;
FROM SimpleWindows IMPORT CreateWindow;
FROM SimpleScreens IMPORT CreateScreen;
FROM AmigaDOSProcess IMPORT Delay;
FROM Memory IMPORT CopyMem;
FROM Rasters IMPORT RastPortPtr;
FROM InOut IMPORT Read,CloseInput,OpenInputFile,Done;
FROM AmigaDos IMPORT IoErr,DeleteFile;



CONST
     flags_idcmp = IDCMPFlagsSet{GadgetUp};
     win_flags = WindowFlagsSet{};

VAR
  fin,impression_sur_ecran,stop: BOOLEAN;
  res,num_ligne: INTEGER;
  g: GadgetPtr;
  w: WindowPtr;
  r: RastPortPtr;
  file_name: string30;
  fichier_nom : string30;

(*****************************************************************************)

PROCEDURE choix(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);


BEGIN

     CASE gad.GadgetID OF
          0: res:=1;|
          1: res:=2;|
          2: res:=3;|
     END;
     fin:=TRUE;
END choix;

(*****************************************************************************)

PROCEDURE saisie_binaire(VAR res : INTEGER);
```

```
VAR
   wp : WindowProc;
   sig : SignalSet;
   msg : IntuiMessagePtr;

BEGIN
     WITH wp DO
          procGadgetUp:=choix;
     END;
     fin:=FALSE;
     WHILE NOT fin DO
           sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
           LOOP
               msg:=GetMsg(win^.UserPort^);
               IF msg=NIL
                  THEN EXIT;
               END;
               ProcIMsg(wp,msg);
           END;
     END;
END saisie_binaire;

(*****************************************************************************)

PROCEDURE titre_ecran(num : INTEGER);


VAR
   lg : INTEGER;

BEGIN
     Move(rp^,0,20);
     Draw(rp^,640,20);
     centrage(rp,0,14,640,tabmess[num]);
END titre_ecran;

(*****************************************************************************)

PROCEDURE choix_ecran_1(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);


VAR
   temp : StringInfoPtr;
   longueur : LONGCARD;

BEGIN
     CASE gad.GadgetID OF
          0: temp:=gad.SpecialInfo;
             longueur:=SIZE(nomuser);
             CopyMem(temp^.Buffer,ADR(nomuser),longueur);|
          1: abandontotal:=TRUE;|
     END;
     IF CompareString(nomuser,'') # equal
     THEN
       fin:=TRUE;
     ELSE
       Move(rp^,30,150);
       Text(rp^,ADR(tabmess[77]),StringLength(tabmess[77]));
     END;

END choix_ecran_1;

(*****************************************************************************)

PROCEDURE ecran_1;
```

```
VAR
   wp : WindowProc;
   sig : SignalSet;
   msg : IntuiMessagePtr;
   variable : BOOLEAN;
   name : string30;


BEGIN
     cherchernomuser(name);
     WITH wp DO
          procGadgetUp:=choix_ecran_1;
     END;
     BeginGadgetList();
       AddGadgetString(250,96,16,16,ADR(name));
       AddGadgetTextButton(457,225,ADR(tabmess[66]));
     gl:=EndGadgetList();
     win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
     rp:=win^.RPort;
     init_couleur;
     Move(rp^,20,100);
     Text(rp^,ADR(tabmess[35]),StringLength(tabmess[35]));
     titre_ecran(34);
     fin:=FALSE;
     WHILE NOT fin DO
           variable:=ActivateGadget(gl^,win^,NIL);
           sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
           LOOP
               msg:=GetMsg(win^.UserPort^);
               IF msg=NIL
                  THEN EXIT;
               END;
               ProcIMsg(wp,msg);
           END;
     END;
     CloseWindow(win^);
     FreeGadgetList(gl^);
END ecran_1;

(*****************************************************************************)

PROCEDURE ecran_2(VAR type_ex : CHAR);


BEGIN
     BeginGadgetList();
       AddGadgetTextButton(120,180,ADR(tabmess[39]));
       AddGadgetTextButton(420,180,ADR(tabmess[38]));
       AddGadgetTextButton(457,225,ADR(tabmess[66]));
     gl:=EndGadgetList();
     win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
     rp:=win^.RPort;
     init_couleur;
     centrage(rp,0,100,640,tabmess[36]);
     titre_ecran(34);
     saisie_binaire(res);
     IF res=3
        THEN abandontotal:=TRUE;
     END;
     IF res=1
        THEN type_ex:='d';
        ELSE type_ex:='p';
     END;
     CloseWindow(win^);
     FreeGadgetList(gl^);
```

```
END ecran_2;

(*****************************************************************************)

PROCEDURE ecran_3;

BEGIN
     BeginGadgetList();
       AddGadgetTextButton(200,180,ADR(tabmess[30]));
       AddGadgetTextButton(420,180,ADR(tabmess[31]));
       AddGadgetTextButton(457,225,ADR(tabmess[66]));
     gl:=EndGadgetList();
     win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
     rp:=win^.RPort;
     init_couleur;
     centrage(rp,0,100,640,tabmess[67]);
     titre_ecran(34);
     saisie_binaire(res);
     IF res=3
        THEN abandontotal:=TRUE;
             CloseWindow(win^);
             FreeGadgetList(gl^);
             RETURN;
     END;
     IF res=1
        THEN abandonpartiel:=FALSE;
        ELSE abandonpartiel:=TRUE;
     END;
     CloseWindow(win^);
     FreeGadgetList(gl^);
END ecran_3;

(*****************************************************************************)

PROCEDURE choix_4(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);


BEGIN
     CASE gad.GadgetID OF
          0: fin:=TRUE;|
     END;
END choix_4;

(*****************************************************************************)

PROCEDURE ecran_4;


VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;

BEGIN
     WITH wp DO
          procGadgetUp:=choix_4;
     END;
     BeginGadgetList();
       AddGadgetTextButton(318,160,ADR(tabmess[44]));
     gl:=EndGadgetList();
     win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
     rp:=win^.RPort;
     init_couleur;
     centrage(rp,0,100,640,tabmess[48]);
     titre_ecran(34);
     fin:=FALSE;
```

```
    WHILE NOT fin DO
          sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
          LOOP
              msg:=GetMsg(win^.UserPort^);
              IF msg=NIL
                 THEN EXIT;
              END;
              ProcIMsg(wp,msg);
          END;
    END;
    CloseWindow(win^);
    FreeGadgetList(gl^);
END ecran_4;

(*****************************************************************************)

PROCEDURE ecran_5(VAR encore : BOOLEAN);

BEGIN
    BeginGadgetList();
      AddGadgetTextButton(200,180,ADR(tabmess[30]));
      AddGadgetTextButton(420,180,ADR(tabmess[31]));
      AddGadgetTextButton(457,225,ADR(tabmess[66]));
    gl:=EndGadgetList();
    win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
    rp:=win^.RPort;
    init_couleur;
    centrage(rp,0,100,640,tabmess[37]);
    titre_ecran(34);
    saisie_binaire(res);
    IF res=3
       THEN
           abandontotal:=TRUE;
            CloseWindow(win^);
           FreeGadgetList(gl^);
           RETURN;
    END;
    IF res=1
       THEN encore:=TRUE;
       ELSE encore:=FALSE;
    END;
    CloseWindow(win^);
    FreeGadgetList(gl^);
END ecran_5;

(*****************************************************************************)

PROCEDURE choix_6(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);

VAR
   temp : StringInfoPtr;
   variable : BOOLEAN;


BEGIN
    CASE gad.GadgetID OF
         0: typerenfor:=1;|
         1: typerenfor:=2;|
         2: typerenfor:=3;|
         3: abandontotal:=TRUE;|
    END;
    fin:=TRUE;
END choix_6;


(*****************************************************************************)
```

```
PROCEDURE ecran_6;


VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;

BEGIN
     WITH wp DO
          procGadgetUp:=choix_6;
     END;
     BeginGadgetList();
       AddGadgetTextButton(120,180,ADR(tabmess[57]));
       AddGadgetTextButton(270,180,ADR(tabmess[65]));
       AddGadgetTextButton(420,180,ADR(tabmess[58]));
       AddGadgetTextButton(457,225,ADR(tabmess[66]));
     gl:=EndGadgetList();
     win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
     rp:=win^.RPort;
     init_couleur;
     centrage(rp,0,100,640,tabmess[56]);
     titre_ecran(34);
     fin:=FALSE;
     WHILE NOT fin DO
           sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
           LOOP
               msg:=GetMsg(win^.UserPort^);
               IF msg=NIL
                  THEN EXIT;
               END;
               ProcIMsg(wp,msg);
           END;
     END;
     CloseWindow(win^);
     FreeGadgetList(gl^);
END ecran_6;

(*****************************************************************************)

PROCEDURE choix_7(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);


BEGIN
     CASE gad.GadgetID OF
          0: typesaisienom:='c';|
          1: typesaisienom:='k';|
          2: abandontotal:=TRUE;
     END;
     fin:=TRUE;
END choix_7;

(*****************************************************************************)

PROCEDURE ecran_7;


VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;

BEGIN
     WITH wp DO
          procGadgetUp:=choix_7;
```

```
      END;
      BeginGadgetList();
        AddGadgetTextButton(200,180,ADR(tabmess[59]));
        AddGadgetTextButton(400,180,ADR(tabmess[60]));
        AddGadgetTextButton(457,225,ADR(tabmess[66]));
      gl:=EndGadgetList();
      win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
      rp:=win^.RPort;
      init_couleur;
      centrage(rp,0,100,640,tabmess[61]);
      titre_ecran(64);
      fin:=FALSE;
      WHILE NOT fin DO
            sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
            LOOP
                msg:=GetMsg(win^.UserPort^);
                IF msg=NIL
                   THEN EXIT;
                END;
                ProcIMsg(wp,msg);
            END;
      END;
      CloseWindow(win^);
      FreeGadgetList(gl^);
END ecran_7;



(*****************************************************************************)

PROCEDURE choix_8(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);


BEGIN
      CASE gad.GadgetID OF
           0: fin:=TRUE;|
      END;
END choix_8;

(*****************************************************************************)

PROCEDURE ecran_8(mauvaise : BOOLEAN);


VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;

BEGIN
      WITH wp DO
           procGadgetUp:=choix_8;
      END;
      BeginGadgetList();
        AddGadgetTextButton(318,160,ADR(tabmess[44]));
      gl:=EndGadgetList();
      win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
      rp:=win^.RPort;
      init_couleur;
      centrage(rp,0,100,640,tabmess[45]);
      titre_ecran(34);
      IF mauvaise
         THEN
              centrage(rp,0,75,640,tabmess[46]);
      END;
      fin:=FALSE;
```

```
     WHILE NOT fin DO
           sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
           LOOP
               msg:=GetMsg(win^.UserPort^);
               IF msg=NIL
                  THEN EXIT;
               END;
               ProcIMsg(wp,msg);
           END;
     END;
     Delay(150);
     CloseWindow(win^);
     FreeGadgetList(gl^);
END ecran_8;


(*****************************************************************************)

PROCEDURE choix_9(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);


BEGIN
     CASE gad.GadgetID OF
          0: impression_sur_ecran:=TRUE;
             stop:=FALSE;|
          1: impression_sur_ecran:=FALSE;
             stop:=FALSE;|
          2: stop:=TRUE;
     END;
     fin:=TRUE;
END choix_9;

(*****************************************************************************)

PROCEDURE ecran_9(VAR abandon,type_impression_ecran : BOOLEAN);


VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;

BEGIN
     WITH wp DO
          procGadgetUp:=choix_9;
     END;
     BeginGadgetList();
       AddGadgetTextButton(100,180,ADR(tabmess[69]));
       AddGadgetTextButton(400,180,ADR(tabmess[70]));
       AddGadgetTextButton(457,225,ADR(tabmess[51]));
     gl:=EndGadgetList();
     win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
     rp:=win^.RPort;
     init_couleur;
     centrage(rp,0,100,640,tabmess[64]);
     titre_ecran(64);
     fin:=FALSE;
     WHILE NOT fin DO
           sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
           LOOP
               msg:=GetMsg(win^.UserPort^);
               IF msg=NIL
                  THEN EXIT;
               END;
               ProcIMsg(wp,msg);
           END;
```

```
     END;
     type_impression_ecran:=impression_sur_ecran;
     abandon:=stop;
     CloseWindow(win^);
     FreeGadgetList(gl^);
END ecran_9;

(*****************************************************************************)

PROCEDURE choix_10(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);


BEGIN
     CASE gad.GadgetID OF
          0: fin:=TRUE;
     END;
END choix_10;

(*****************************************************************************)

PROCEDURE ecran_10(num_disk : INTEGER);


VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;
   numero_disk : LONGINT;
   disk : ARRAY [0..2] OF CHAR;

BEGIN
     WITH wp DO
          procGadgetUp:=choix_10;
     END;
     BeginGadgetList();
       AddGadgetTextButton(318,160,ADR(tabmess[44]));
     gl:=EndGadgetList();
     win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
     rp:=win^.RPort;
     init_couleur;
     centrage(rp,0,100,500,tabmess[47]);
     numero_disk:=LONGINT(num_disk);
     ConvNumberToString(disk,numero_disk,FALSE,10,2,"0");
     Move(rp^,500,100);
     Text(rp^,ADR(disk),StringLength(disk));
     titre_ecran(34);
     fin:=FALSE;
     WHILE NOT fin DO
           sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
           LOOP
               msg:=GetMsg(win^.UserPort^);
               IF msg=NIL
                  THEN EXIT;
               END;
               ProcIMsg(wp,msg);
           END;
     END;
     Delay(150);
     CloseWindow(win^);
     FreeGadgetList(gl^);
END ecran_10;

(*****************************************************************************)

PROCEDURE prep_prt;
```

```
VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;

BEGIN
     WITH wp DO
          procGadgetUp:=choix_4;
     END;
     BeginGadgetList();
       AddGadgetTextButton(318,160,ADR(tabmess[44]));
     gl:=EndGadgetList();
     win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
     rp:=win^.RPort;
     init_couleur;
     centrage(rp,0,100,640,tabmess[63]);
     titre_ecran(64);
     fin:=FALSE;
     WHILE NOT fin DO
           sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
           LOOP
               msg:=GetMsg(win^.UserPort^);
               IF msg=NIL
                  THEN EXIT;
               END;
               ProcIMsg(wp,msg);
           END;
     END;
     CloseWindow(win^);
     FreeGadgetList(gl^);
END prep_prt;

(*****************************************************************************)

PROCEDURE choix_11(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);

VAR tampon :string30;
    bool : BOOLEAN;


BEGIN
     CASE gad.GadgetID OF
          0: CopyString(tampon,fichier_nom);
             ConcatString(tampon,".trace");
             bool:= DeleteFile(ADR(tampon));|
          1: |
     END;
     fin:=TRUE;

END choix_11;

(*****************************************************************************)

PROCEDURE ecran_11( VAR nom:string30);


VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;
   nomfich : ARRAY [1..80] OF CHAR;

BEGIN
     WITH wp DO
          procGadgetUp:=choix_11;
```

```
    END;
    BeginGadgetList();
      AddGadgetTextButton(200,180,ADR(tabmess[30]));
      AddGadgetTextButton(420,180,ADR(tabmess[31]));
    gl:=EndGadgetList();
    win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
    rp:=win^.RPort;
    init_couleur;
    CopyString(fichier_nom,nom);
    DeleteSubString(nom,0,5);
    CopyString(nomfich,tabmess[79]);
    ConcatString(nomfich,nom);
    centrage(rp,0,100,640,nomfich);
    titre_ecran(54);
    fin:=FALSE;
    WHILE NOT fin DO
          sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
          LOOP
              msg:=GetMsg(win^.UserPort^);
              IF msg=NIL
                 THEN EXIT;
              END;
              ProcIMsg(wp,msg);
          END;
    END;


    CloseWindow(win^);
    FreeGadgetList(gl^);
END ecran_11;

(*****************************************************************************)

PROCEDURE saisie_fichier(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);


VAR
   temp : StringInfoPtr;
   longueur : LONGCARD;
   ecran : ScreenPtr;
   fichier : string30;
   fich : string30;

BEGIN
    CASE gad.GadgetID OF
         0: temp:=gad.SpecialInfo;
            longueur:=SIZE(fichier);
            CopyMem(temp^.Buffer,ADR(fichier),longueur);
            ecran:=scr;
            CopyString(file_name,'user:');
            ConcatString(file_name,fichier);
            CopyString(fich,'user:');
            ConcatString(fich,fichier);
            ConcatString(fich,'.trace');
            fin:=verifiefich(ecran,fich,'');
            IF NOT fin
               THEN centrage(rp,0,165,640,tabmess[68]);
            END;
            stop:=FALSE;|
         1: fin:=TRUE;
            stop:=TRUE;|
    END;
END saisie_fichier;
```

```
(*****************************************************************************)

PROCEDURE demande_nom_fichier(VAR fichier:string30; VAR abandon:BOOLEAN;
                              type : CHAR);


VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;
   variable : BOOLEAN;

BEGIN
     WITH wp DO
          procGadgetUp:=saisie_fichier;
     END;
     BeginGadgetList();
       AddGadgetString(188,140,31,31,ADR(""));
       AddGadgetTextButton(500,225,ADR(tabmess[51]));
     gl:=EndGadgetList();
     win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
     rp:=win^.RPort;
     init_couleur;
     centrage(rp,0,100,640,tabmess[62]);
     IF type = "i" THEN titre_ecran(64);
                   ELSE titre_ecran(54);
                   END;
     fin:=FALSE;
     WHILE NOT fin DO
           variable:=ActivateGadget(gl^,win^,NIL);
           sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
           LOOP
               msg:=GetMsg(win^.UserPort^);
               IF msg=NIL
                  THEN EXIT;
               END;
               ProcIMsg(wp,msg);
           END;
     END;
     IF NOT stop
        THEN CopyString(fichier,file_name);
     END;
     abandon:=stop;
     CloseWindow(win^);
     FreeGadgetList(gl^);
END demande_nom_fichier;


(*****************************************************************************)

PROCEDURE efface(rastport : RastPortPtr;x,y,h,l : INTEGER);

BEGIN
   SetAPen(rastport^,0);
   RectFill(rastport^,x,y,x+l,y+h);
   SetAPen(rastport^,1);
END efface;

(*****************************************************************************)


PROCEDURE lire_1_ligne(VAR termine : BOOLEAN; VAR line : ARRAY OF CHAR);


VAR
   car : CHAR;
```

```
   Fini : BOOLEAN;

BEGIN
     CopyString(line,' ');
     Read(car);
     IF NOT Done
        THEN
            termine:=TRUE;
            RETURN;
     END;
     WHILE (car#12C) DO
           ConcatString(line,car);
           Read(car);
     END;
END lire_1_ligne;

(*****************************************************************************)

PROCEDURE affiche_1_ecran(VAR termine : BOOLEAN; fich_trace : string30);


VAR
   line : ARRAY [1..60] OF CHAR;
   compteur : INTEGER;
   y : INTEGER;

BEGIN
     compteur:=1;
     y:=31;
     WHILE compteur<=20 DO
           lire_1_ligne(termine,line);
           IF NOT termine
              THEN
                  Move(rp^,20,y);
                  Text(rp^,ADR(line),StringLength(line));
                  compteur:=compteur+1;
                  y:=y+10;
              ELSE
                  compteur:=21;
           END;
     END;
END affiche_1_ecran;

(*****************************************************************************)

PROCEDURE choix_imp_trace(VAR w : Window; VAR msg : MsgData; VAR gad : Gadget);

BEGIN
     fin:=TRUE;
END choix_imp_trace;

(*****************************************************************************)

PROCEDURE impression_trace_ecran(nomfichier : string30);


VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;
   termine : BOOLEAN;
   nomfich : string50;

BEGIN
     WITH wp DO
          procGadgetUp:=choix_imp_trace;
```

```
      END;
      BeginGadgetList();
        AddGadgetTextButton(310,225,ADR(tabmess[44]));
      gl:=EndGadgetList();
      win:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,gl,scr,NIL);
      rp:=win^.RPort;
      titre_ecran(64);
      num_ligne:=1;
      termine:=FALSE;
      CopyString(nomfich,nomfichier);
      ConcatString(nomfich,".trace");
      OpenInputFile(nomfich);
      IF IoErr()#LONGINT(0)
         THEN ScreenToFront(scr^);
      END;
      WHILE NOT termine DO
            efface(rp,2,21,201,636);
            affiche_1_ecran(termine,nomfichier);
            fin:=FALSE;
            WHILE NOT fin DO
                  sig:=Wait(SignalSet{CARDINAL(win^.UserPort^.mpSigBit)});
                  LOOP
                      msg:=GetMsg(win^.UserPort^);
                      IF msg=NIL
                         THEN EXIT;
                      END;
                      ProcIMsg(wp,msg);
                  END;
            END;
      END;
      CloseInput;
      CloseWindow(win^);
      FreeGadgetList(gl^);

END impression_trace_ecran;


(*****************************************************************************)

PROCEDURE affiche_1_ecran_com(typeex : CHAR);


VAR
   ligne : string70;
   compteur : INTEGER;
   y : INTEGER;
   num_str : ARRAY [0..2] OF CHAR;

BEGIN
     compteur:=1;
     y:=31;
     WHILE compteur<=19 DO
         IF typeex = "d"
         THEN
              IF num_ligne <= nbcommentaired
              THEN
                 ligne:= tabcommentaired[num_ligne].com;
                 Move(r^,20,y);
                 ConvNumberToString(num_str,LONGCARD(num_ligne),FALSE,10,2," ");
                 Text(r^,ADR(num_str),StringLength(num_str));
                 num_ligne:=num_ligne+1;
                 Move(r^,50,y);
                 Text(r^,ADR(ligne),StringLength(ligne));
                 compteur:=compteur+1;
                 y:=y+10;
              ELSE
```

```
                compteur:=21;
            END;
        ELSE
            IF num_ligne <= nbcommentairep
            THEN
                ligne:= tabcommentairep[num_ligne].com;
                Move(r^,20,y);
                ConvNumberToString(num_str,LONGCARD(num_ligne),FALSE,10,2," ");
                Text(r^,ADR(num_str),StringLength(num_str));
                num_ligne:=num_ligne+1;
                Move(r^,50,y);
                Text(r^,ADR(ligne),StringLength(ligne));
                compteur:=compteur+1;
                y:=y+10;
            ELSE
                compteur:=21;
            END;
        END;
    END;
END affiche_1_ecran_com;


(*****************************************************************************)

PROCEDURE choix_impc(VAR w: Window;VAR msg:MsgData;VAR gad:Gadget);


BEGIN
    CASE gad.GadgetID OF
        0:;|
        1: fin := TRUE;|
    END;
END choix_impc;

(*****************************************************************************)

PROCEDURE impression_com(typeex : CHAR);


VAR
   sig : SignalSet;
   msg : IntuiMessagePtr;
   wp : WindowProc;

BEGIN
    WITH wp DO
        procGadgetUp:=choix_impc;
    END;
    BeginGadgetList();
      AddGadgetTextButton(100,225,ADR(tabmess[50]));
      AddGadgetTextButton(500,225,ADR(tabmess[10]));
    g:=EndGadgetList();
    w:=CreateWindow(0,0,640,240,flags_idcmp,win_flags,g,scr,NIL);
    r:=w^.RPort;
    Move(r^,0,20);
    Draw(r^,640,20);
    centrage(r,0,14,640,tabmess[71]);
    num_ligne:=1;
    fin:=FALSE;
    WHILE NOT fin DO
          efface(r,2,21,201,636);
          affiche_1_ecran_com(typeex);
          IF num_ligne <= nbcommentaired
          THEN
                num_ligne:=1;
          END;
```

```
            sig:=Wait(SignalSet{CARDINAL(w^.UserPort^.mpSigBit)});
            LOOP
                msg:=GetMsg(w^.UserPort^);
                IF msg=NIL
                   THEN EXIT;
                END;
                ProcIMsg(wp,msg);
            END;
        END;
        CloseWindow(w^);
        FreeGadgetList(g^);
    END impression_com;


END ecran.
```

```
IMPLEMENTATION MODULE basedonnees;

FROM varglobale IMPORT tabcommentaired,nbcommentaired,scr,rp,gl,string7Ø,
     tabcommentairep,nbcommentairep,string3Ø,nomuser,tabex;

FROM InOut IMPORT OpenInputFile,CloseInput,ReadInt,Done,Read,ReadString,
     OpenOutputFile,CloseOutput,WriteInt,Write,WriteString,WriteLn,EOL;
FROM AmigaDos IMPORT IoErr,ModeOldFile,FileHandle,Open,Close,DateStamp,
     DeleteFile;
FROM Strings IMPORT CopyString,ConcatString,CompareStringCAP,equal;
FROM Intuition IMPORT ScreenToFront,ScreenPtr;
IMPORT FileSystem;

VAR file1,file2: FileSystem.File;

(*****************************************************************************)

PROCEDURE lit_catalogue_desig;

VAR

    ligne: string7Ø;
    ent,i : INTEGER;

BEGIN
     OpenInputFile("biblioØ1:catalogue_desig");
     IF IoErr() # LONGINT(Ø)
         THEN ScreenToFront(scr^);
     END;
     nbcommentaired := Ø;
     WHILE Done DO
           nbcommentaired := nbcommentaired + 1;
           ReadInt (tabcommentaired[nbcommentaired].numbiblio);
           ReadInt (tabcommentaired[nbcommentaired].syst);
           ReadInt (tabcommentaired[nbcommentaired].coul);
           CopyString(tabcommentaired[nbcommentaired].com,"");
           ReadInt(ent);
           i := 1;
           WHILE (i <= ent ) DO
                 ReadString(ligne);
                 ConcatString(tabcommentaired[nbcommentaired].com,ligne);
                 ConcatString(tabcommentaired[nbcommentaired].com," ");
                 i := i+1;
           END;

      END;
      nbcommentaired:=nbcommentaired-1;
      CloseInput;

END lit_catalogue_desig;

(*****************************************************************************)

PROCEDURE lit_catalogue_parc;

VAR

    ligne: string7Ø;
    ent,i : INTEGER;

BEGIN
     OpenInputFile("biblioØ1:catalogue_parc");
     IF IoErr() # LONGINT(Ø)
         THEN ScreenToFront(scr^);
     END;
     nbcommentairep := Ø;
```

```
    WHILE Done DO
          nbcommentairep := nbcommentairep + 1;
          ReadInt (tabcommentairep[nbcommentairep].numbiblio);
          CopyString(tabcommentairep[nbcommentairep].com,"");
          ReadInt(ent);
          i := 1;
          WHILE (i <= ent ) DO
                ReadString(ligne);
                ConcatString(tabcommentairep[nbcommentairep].com,ligne);
                ConcatString(tabcommentairep[nbcommentairep].com," ");
                i := i+1;
          END;

     END;
     nbcommentairep:=nbcommentairep-1;
     CloseInput;

END lit_catalogue_parc;

(*****************************************************************************)

PROCEDURE cherchernomuser(VAR nom:string30);

BEGIN

  OpenInputFile("user:donnees_session");
  IF IoErr() # LONGINT(0)
  THEN ScreenToFront(scr^);
  END;
  IF Done THEN ReadString(nom) ELSE nom :='' END;
  CloseInput;

END cherchernomuser;

(*****************************************************************************)

PROCEDURE verifiediskuser(VAR cor : BOOLEAN);

VAR nom : string30;

BEGIN
  cherchernomuser(nom);
  IF (CompareStringCAP(nom,nomuser)=equal) OR
     (CompareStringCAP(nom, "")=equal)
  THEN cor:= TRUE;
  ELSE cor := FALSE;
  END;

END verifiediskuser;


(*****************************************************************************)

PROCEDURE copiefichier(entree,sortie : ARRAY OF CHAR);


VAR
   fichier1 : FileSystem.File;
   fichier2 : FileSystem.File;
   car : CHAR;

BEGIN
     FileSystem.Lookup(fichier1,entree,FALSE);
     IF IoErr()#LONGINT(0)
        THEN ScreenToFront(scr^);
     END;
```

```
    FileSystem.Lookup(fichier2,sortie,TRUE);
    IF IoErr()#LONGINT(Ø)
       THEN ScreenToFront(scr^);
    END;
    FileSystem.ReadChar(fichier1,car);
    WHILE NOT fichier1.eof DO
          FileSystem.WriteChar(fichier2,car);
          FileSystem.ReadChar(fichier1,car);
    END;
    FileSystem.Close(fichier1);
    FileSystem.Close(fichier2);

END copiefichier;

(*****************************************************************************)

PROCEDURE passer_ligne;

CONST
  finlg = 12C;

VAR car: CHAR;

BEGIN
  FileSystem.ReadChar(file1,car);
  WHILE car # finlg DO
    FileSystem.ReadChar(file1,car);
  END;

END passer_ligne;

(*****************************************************************************)

PROCEDURE ajout_exercice(numex: INTEGER);

CONST
  finlg = 12C;

VAR i:INTEGER;
    car: CHAR;

BEGIN
 IF (tabex[numex].typeex = "d") OR
    (tabex[numex].typeex = "D")
 THEN FileSystem.Lookup(file1,"biblioØ1:designation_info",FALSE);
      IF IoErr()#LONGINT(Ø)
      THEN ScreenToFront(scr^);
      END;
 ELSE FileSystem.Lookup(file1,"biblioØ1:parcours_info",FALSE);
      IF IoErr()#LONGINT(Ø)
      THEN ScreenToFront(scr^);
      END;
 END;
 FileSystem.Lookup(file2,"RAM:liste_exercice",FALSE);
 IF IoErr()#LONGINT(Ø)
 THEN ScreenToFront(scr^);
 END;
 i:= 1;
 WHILE i< tabex[numex].numcat DO
   passer_ligne;
   i:= i+1;
 END;
 FileSystem.ReadChar(file2,car);
 WHILE (NOT file2.eof) DO
   FileSystem.ReadChar(file2,car);
 END;
```

```
IMPLEMENTATION MODULE utilitaire;

FROM AmigaDOS IMPORT  IoErr;
FROM Rasters IMPORT RastPortPtr;
FROM SYSTEM IMPORT ADR;
FROM Text IMPORT Text, TextLength;
FROM Strings IMPORT StringLength;
FROM Drawing IMPORT Move;
FROM Intuition IMPORT ScreenToFront, ScreenPtr;
FROM InOut IMPORT OpenInputFile, Done, CloseInput;
FROM Views IMPORT LoadRGB4;

FROM varglobale IMPORT scr;


(*****************************************************************************)

PROCEDURE centrage(rpt : RastPortPtr; x,y,largeur : INTEGER;
                   message : ARRAY OF CHAR);


VAR
   lg : INTEGER;

BEGIN
     lg:=TextLength(rpt^,ADR(message),StringLength(message));
     Move(rpt^,((largeur-lg) DIV 2)+x,y);
     Text(rpt^,ADR(message),StringLength(message));
END centrage;


(*****************************************************************************)

PROCEDURE verifiefich(ecran:ScreenPtr;nom,drive : ARRAY OF CHAR) : BOOLEAN;


BEGIN
     OpenInputFile(nom);
     IF IoErr()#LONGINT(Ø)
        THEN ScreenToFront(scr^);
     END;
     IF Done
        THEN CloseInput;
             RETURN TRUE;
        ELSE RETURN FALSE;
     END;
END verifiefich;

(*****************************************************************************)

PROCEDURE init_couleur;


VAR
   CMAP : ARRAY [Ø..31] OF CARDINAL;

BEGIN
     CMAP[Ø]:=ØØØØH;
     CMAP[1]:=ØFFFH;
     CMAP[2]:=ØEØØH;
     CMAP[3]:=ØAØØH;
     CMAP[4]:=ØE83H;
     CMAP[5]:=ØFEØH;
     CMAP[6]:=ØFØØH;
     CMAP[7]:=ØEØØH;
     CMAP[8]:=ØØB6H;
```

```
  FileSystem.WriteChar(file2," ");
  FileSystem.ReadChar(file1,car);
  WHILE (car # finlg) AND (NOT file1.eof) DO
    FileSystem.WriteChar(file2,car);
    FileSystem.ReadChar(file1,car);
  END;
  FileSystem.Close(file1);
  FileSystem.Close(file2);


END ajout_exercice;

END basedonnees.
```

```
      CMAP[9]:=ØØDDH;
      CMAP[10]:=ØØAFH;
      CMAP[11]:=Ø07CH;
      CMAP[12]:=ØØØFH;
      CMAP[13]:=Ø70FH;
      CMAP[14]:=ØCØEH;
      CMAP[15]:=ØCØ8H;
      CMAP[16]:=Ø62ØH;
      CMAP[17]:=ØE84H;
      CMAP[18]:=ØA52H;
      CMAP[19]:=ØFCAH;
      CMAP[20]:=Ø333H;
      CMAP[21]:=Ø444H;
      CMAP[22]:=Ø555H;
      CMAP[23]:=Ø666H;
      CMAP[24]:=Ø777H;
      CMAP[25]:=Ø888H;
      CMAP[26]:=Ø999H;
      CMAP[27]:=ØAAAH;
      CMAP[28]:=ØBBBH;
      CMAP[29]:=ØCCCH;
      CMAP[30]:=ØDDDH;
      CMAP[31]:=ØEEEH;
      LoadRGB4(scr^.ViewPort,ADR(CMAP),32);
END init_couleur;

END utilitaire.
```

Réalisation, impression et reliure : SPRL Centre de Reprographie - Tél. : 081/22.87.72