MASTER EN SCIENCES INFORMATIQUES

Java dans un contexte d''Enseignement Assisté par Ordinateur

Wyngaert, Johan

Award date:
1996

Awarding institution:
Universite de Namur

Facultés Universitaires
Notre-Dame de la Paix
Institut d'Informatique
Namur

Java dans un contexte d'Enseignement Assisté par Ordinateur

Johan WYNGAERT

Promoteur : Jean RAMAEKERS

Mémoire présenté en vue de l'obtention du grade de Licencié en Informatique
Année Académique 1995-1996

Résumé

Dans ce mémoire, nous abordons l'Enseignement Assisté par ordinateur (E.A.O.) par la présentation du développement d'un logiciel de formation utilisant les potentialités du langage Java et destiné à fournir un apprentissage des notions de bases du protocole Ethernet. Pour ce faire, nous présentons d'abord le langage Java et son environnement ainsi que quelques notions théoriques relatives à l'E.A.O. en insistant sur la dimension visuelle de l'interface d'un tel produit. Nous décrivons ensuite en détail le didacticiel que nous avons développé, d'un point de vue pédagogique et d'un point de vue informatique. Enfin, nous terminons en procédant à une évaluation du didacticiel et en proposant des perspectives de continuation du projet.

Abstract

In this master thesis, we approach Computer Aided Teaching (C.A.T.) by the presentation of the development of an education software using potentiality of Java language and aimed at the teaching of the Ethernet protocol's basic notions. First we introduce the Java language, his environment and some C.A.T. theorical notions centered on the concept of graphic interface. We then describe in an educational and software engineering approach, the C.A.T. software we have developped. Finally, we carry out an evaluation of our C.A.T. software and we propose some future works.

Remerciements

Je tiens tout d'abord à remercier mon promoteur, Jean Ramaekers, pour toute l'aide qu'il m'a apportée.

Je tiens ensuite à exprimer mes plus vifs remerciements à madame Charlier du centre Education et Technologie pour ses conseils judicieux et sa disponibilité.

Je tiens enfin à remercier ma famille et mes proches tant pour leur soutient moral que matériel sans lesquels rien n'aurait été pareil.

Table des matières.

Table des abréviations.

API	Application Programming Interfaces
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
E.A.O.	Enseignement Assisté par Ordinateur
HTML	Hypertext Markup Language
IDL	Interface Definition Language
ISO	International Organization for Standardization
JDBC	Java Database Connectivity
MAC	Medium Access Control
RMI	Remote Method Invocation
SET	Secure Electronic Transaction
TCP/IP	Transmission Control Protocol / Internet Protocol
WWW	World Wide Web

Introduction

Le monde d'Internet est un vaste océan de données représentées sous de nombreux formats et stockées en de nombreux endroits. Une grande partie d'Internet est organisée sous forme de World Wide Web (WWW) qui utilise l'hypertexte pour faciliter sa navigation. Les browsers WWW permettent aux utilisateurs de naviguer à travers l'information rencontrée sur le net. Les fichiers les plus fréquemment rencontrés par les browsers WWW sont du type *hypertext markup language* (HTML).

Ce monde est également composé d'un grand nombre de sites où des services de nature diverse sont proposés aux utilisateurs. Cela peut aller de l'abonnement à des revues à la réservation de places de concert, en passant par des applications de type Enseignement Assisté par Ordinateur (E.A.O.). Ces dernières sont précisément celles qui ont retenu notre attention.

L'apparition d'un nouveau langage de programmation permet d'envisager de nouvelles possibilités quant à l'utilisation d'Internet : il s'agit du langage Java. Il permet, par exemple, d'insérer de petites animations sur des pages HTML qui rendent la navigation à travers les sites beaucoup plus conviviale et attrayante. De plus, ces potentialités peuvent avoir des répercutions plus primordiales dans le domaine de l'Enseignement Assisté par Ordinateur. De nombreux sites permettent aux navigateurs d'apprendre, par exemple, comment créer des pages HTML[1]. Cependant, ces sites n'utilisent généralement pas encore les avantages que peut procurer le langage Java. Nous nous proposons donc de concevoir une application de type E.A.O. utilisant ces potentialités.

En développant un cours sur le protocole CSMA/CD utilisé dans les réseaux locaux, ce mémoire va faire la synthèse des avantages et des inconvénients de certains aspects du langage Java qui ont été utilisés lors du développement de l'application. Le choix du sujet de ce cours se justifie par la possibilité de développer une animation concernant la collision des « messages » (trames) envoyés par l'ensemble des stations situées sur le réseau.

---

[1] cfr. "Un cours de HTML online" (Université de Nice http://nephi.unice.fr/html/ French/CoursHTML/cours.html

Ce mémoire a été structuré de la façon suivante : les deux premiers chapitres fournissent une description du langage Java et de l'E.A.O. restreint à l'hypermédia; les deux chapitres suivants présentent l'application développée dans le domaine des réseaux locaux ainsi qu'une synthèse des avantages et inconvénients du langage Java qui ont pu être mis en évidence lors du développement de l'application; le dernier chapitre présente une évaluation du didacticiel ainsi qu'un certain nombre de perspectives pour la continuation du projet.

Le premier chapitre est consacré à la description du langage Java : son histoire, sa plate-forme et ses caractéristiques. Le second chapitre aborde le sujet de l'apprentissage dans un contexte d'E.A.O. et d'hypermédia en particulier. Il propose une description des usages de l'hypermédia ainsi qu'une présentation de la démarche à suivre lors de la conception d'un didacticiel à partir des règles de conception d'une interface homme-machine.

Le troisième chapitre décrit en détail notre didacticiel d'un point de vue informatique en présentant, pour commencer, un rappel sur les réseaux Ethernet. Le quatrième chapitre passe en revue les avantages et inconvénients du langage Java mis en évidence lors du développement du didacticiel.

Le cinquième chapitre présente une critique du travail réalisé avant de fournir quelques perspectives pour l'éventuelle continuation du projet.

Chapitre 1 : Le langage Java.

Le langage de programmation Java et son environnement sont conçus pour résoudre un certain nombre de problèmes liés à la pratique de la programmation moderne.

Dans ce chapitre, nous allons expliquer quelles sont les origines de Java ainsi que les raisons pour lesquelles il est spécialement approprié pour l'utilisation d'Internet (plus particulièrement du World-Wide Web). Il sera divisé en cinq parties : la première introduit l'histoire du langage de programmation Java. La deuxième partie décrit la plate-forme Java et en particulier sa machine virtuelle. La partie suivante décrit le langage de programmation Java lui-même en le comparant au C++ dont il est assez proche. La quatrième partie fera une description de la sécurité dans le langage Java. Nous terminerons ce chapitre en expliquant pourquoi le langage Java convient assez bien au moyen de communication qu'est Internet.

1 L'histoire de Java.

L'histoire d'un langage de programmation peut être fort révélatrice quant au langage lui-même.

Afin d'obtenir un langage assez simple d'utilisation et fort répandu, il est important de vérifier les qualités de ce langage au cours de son développement. C'est la raison pour laquelle le langage doit être utilisé pour le développement de projets concrets : ce fut le cas pour le langage Java.

1.1 Les origines de Java.

Le langage Java a été conçu et implémenté par une petite équipe dirigée par James Gosling[1] à Sun Microsystems en Californie. Cette équipe travaillait sur le développement d'une application. Ils se sont rapidement aperçus que les langages de programmation existants tels que le C et le C++ n'étaient pas adéquats. Les programmes écrits en C++ doivent être compilés pour un environnement particulier. A chaque changement d'environnement, une nouvelle compilation doit être réalisée. De plus, les programmes en C++ doivent être recompilés lorsque changent les librairies qu'ils utilisent.

En 1990, James Gosling a commencé la création d'un nouveau langage de programmation qui ne devrait pas posséder les inconvénients des langages traditionnels tels que le C et le C++. Java en est le résultat.

1.2 Hotjava, Webrunner et les autres.

En 1993, le WWW passe d'une interface basée sur le texte à une interface plus graphique. Cela a suscité énormément d'intérêt de par le monde et en particulier celui de l'équipe de développement de Java. En effet, un langage tel que Java paraissait idéal pour la programmation d'applications à destination du Web puisqu'il pourrait tourner sur des types de machines allant des PCs à Unix en passant par les Macs. Le résultat fut un browser Web appelé Webrunner entièrement écrit en Java. Par la suite, ce browser fut renommé Hotjava pour des raisons commerciales. Hotjava est donc le premier browser Web à supporter les applets[2] Java.

---

[1] il est également responsable du développement de Unix emacs et de NeWS window system.

[2] les applets Java sont de petits programmes écrits en Java et ajoutés aux pages Web de façon à produire des effets spéciaux.

Hotjava a donc été une étape importante dans le développement du langage Java. Il a non seulement permis de produire un langage plus mature mais également de le diffuser dans le monde.

La sortie officielle de la technologie Java a été faite en mai 1995 à la conférence SunWorld à San Francisco. Lors de cette conférence Marc Andreessen, vice président de la technologie à Netscape Communications, a annoncé que Netscape Navigator supporterait les applets Java.

1.3 Pourquoi ce nom : Java ?

Au départ, James Gosling appelait ce langage Oak (du nom de l'arbre situé en face de la fenêtre de son bureau).

Par la suite, l'équipe de développement a découvert que Oak était le nom d'un langage de programmation antécédent au langage de Sun. C'est pourquoi un autre nom a dû être choisi. Après une discussion au sein de l'équipe, Java a été choisi. Contrairement à la croyance populaire, Java n'est pas un acronyme.

2 La plate-forme Java

De nombreuses plates-formes sont disponibles dans le monde des ordinateurs; parmi celles-ci on trouve Microsoft Windows, Macintosh, OS/2, Unix® et NetWare®. Les softwares doivent être compilés pour pouvoir fonctionner sur un type particulier de plate-forme. En effet, puisque les fichiers binaires sont propres à un type de machine, un même fichier binaire ne peut être utilisé sur deux plates-formes différentes.

La plate-forme Java est une nouvelle plate-forme qui permet de faire "tourner" des applets et des applications hautement interactives, dynamiques et sécurisées sur des réseaux d'ordinateurs. Ce qui différencie la plate-forme Java des autres, c'est qu'elle se situe au-dessus des autres plates-formes et que les programmes Java sont compilés en byte-code (qui ne sont pas propres à une machine particulière) qui sont des instructions machine pour une machine virtuelle. Un programme écrit en langage Java est compilé en byte-code qui peut être exécuté

chaque fois que la plate-forme Java est présente, quel que soit le système d'exploitation sous-jacent. Cette portabilité est possible parce que la machine virtuelle Java se trouve au coeur de la plate-forme Java.

Par conséquent, la plate-forme Java est idéale pour Internet, puisque tout programme peut être exécuté sur n'importe quel ordinateur du monde.

Les utilisateurs utilisent le langage Java pour écrire des applications qu'ils compilent une seule fois pour la plate-forme Java. Le code source est compilé vers un byte-code intermédiaire et portable qui sera exécuté chaque fois que la plate-forme Java est présente. Les programmeurs peuvent écrire des applications grâce au langage Java qui sont orientées objets, multitâches et liées dynamiquement.

Le langage Java est une rampe d'accès à la plate-forme Java. Des programmes écrits en langage Java et compilés s'exécutent sur la plate-forme Java. Cette dernière a deux composantes essentielles :

- la machine virtuelle Java
- Les API Java (Application Programming Interface)

Dans la figure 1.1, la plate-forme de base de Java est en noir. Elle inclut les blocs appelés *Adapter*. Les API Java incluent à la fois les *Java Base API* et *Java Standard Extension API*. La machine virtuelle Java est l'épine dorsale de la plate-forme. La *Porting Interface* a une partie indépendante de la plate-forme (en noir) et une partie dépendante de la plate-forme (les *Adapter*).

La plate-forme de base de Java est la plate-forme Java minimale permettant d'exécuter des applications améliorées par des applets Java. Cette plate-forme a la même machine virtuelle Java qu'une plate-forme complète.

Au sommet de la plate-forme Java se trouvent les *Applets* et *Applications*. Une applet est un programme nécessitant un browser pour s'exécuter. Elles sont notamment utilisés pour rendre le pages HTML plus conviviales. Un indicateur spécial (<applet>) est ajouté aux pages Web, ce qui permet, lors de l'accès à ces pages, d'importer et d'exécuter l'applet en question.

Actuellement, tous les browsers ne reconnaissent pas cet indicateur spécial; HotJava et Netscape 2.0[3] en sont capable.

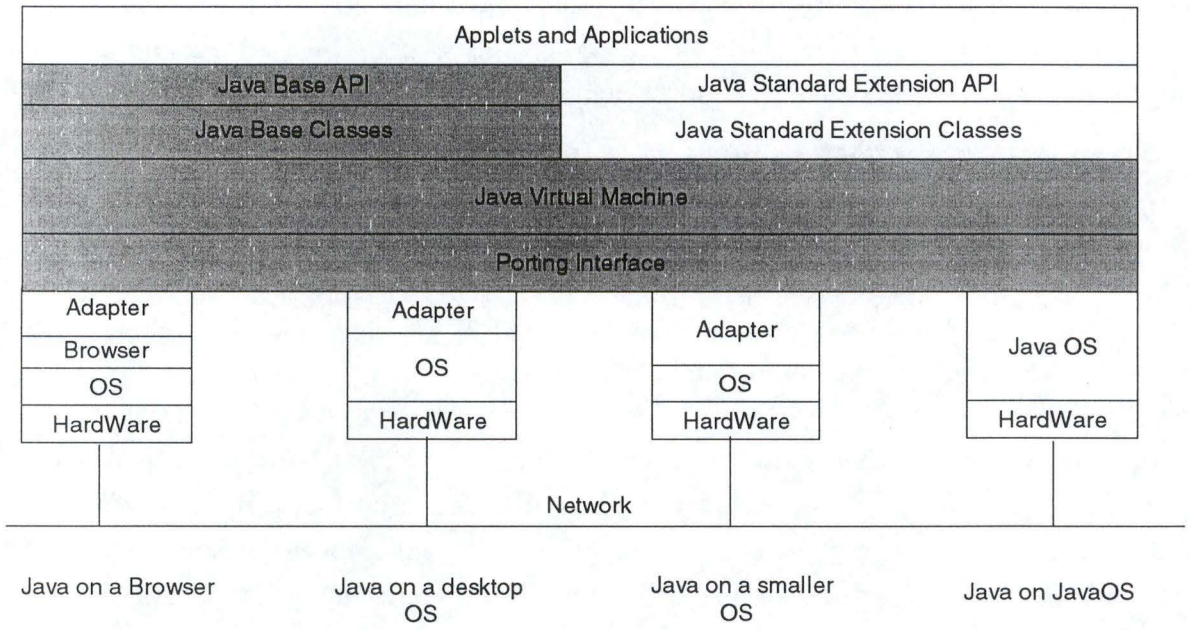

Figure 1.1 : La plate-forme[4] de base de Java.

Les applications, quant à elles, sont des programmes qui ne requièrent pas de browsers pour être exécutés. Ces applications correspondent aux programmes dans les autres langages. Elles peuvent réaliser des tâches traditionnelles telles que celles réalisées avec une application graphique ou un traitement de texte.

Bien que les *Applets* et les *Applications* soient appelées de façon différente, la plus grande partie des méthodes[5] disponibles dans les librairies du langage Java peuvent être utilisées par ces dernières. Par exemple, elles ont toutes deux accès à une base de données extérieure, sélectionnent l'information souhaitée, traitent l'information localement et stockent les résultats.

---

[3] il est cependant nécessaire qu'il soit installé sur un ordinateur ayant un OS 32 bits.

[4] cette figure est extraite de l'article : *A look Inside the Java Platform* [HTML5].

[5] le terme utilisé pour désigner des procédures ou des fonctions dans les langages orientés objets est le terme *méthode*.

Cependant, une applet requiert un réseau pour être exécutée alors qu'une application n'en nécessite pas. Les applications ont plus de libertés dans le sens où elles ont accès à tous les services du système. Par exemple, une application peut réaliser des accès en lecture ou en écriture sur des fichiers situés sur un disque. Par contre, étant donné que les applets peuvent être chargées à partir de pages HTML, ces accès lui sont refusés. Ces contraintes seront supprimées lorsque des signatures digitales pourront être assignées aux applets.

2.1 La machine virtuelle Java.

La machine virtuelle Java permet l'indépendance des applets Java et des applications vis-à-vis du système d'exploitation sous-jacent.

De plus, la machine virtuelle définit un format de fichiers binaires indépendant de la machine; ces fichiers sont appelés des classes (c'est-à-dire des fichiers .class). La représentation sous forme de byte-code d'un programme écrit en langage Java est symbolique dans le sens où les variables et les méthodes sont définies symboliquement comme des chaînes de caractères. Lors du premier appel de la méthode, une recherche sur le nom est réalisée dans le fichier .class et une valeur numérique correspondante est déterminée pour permettre des accès plus rapides lors des prochains appels.

2.2 Les Java Base API.

En incluant les API dans la plate-forme Java, les concepteurs sont certains que leurs applications s'exécuteront partout.

Actuellement, les *Java Base API* sont définies comme étant les *Java Applet API*. Cependant, au fur et à mesure du développement de la plate-forme Java, cette base se développera en faisant migrer certaines API dans cette base. Le plan actuel de migration [HTML5] est le suivant :

Transfert à Java Base API	Reste dans Java Standard Extension API
Java 2D	Java 3D
Audio	Video, MIDI

Transfert à Java Base API	Reste dans Java Standard Extension API
Java Media Framework	Java Share
Java Animation	Java Telephony
Java Entreprise	Java Server
Java Commerce	Java Management
Java Security	

L'API Java 2D fournit des possibilités graphiques et de création d'images plus grandes que celles disponibles avec l'API Applet. Elle permet la création de graphiques de haute qualité, indépendants de la plate-forme. Elle fournit également un mécanisme d'extension qui permet l'affichage des graphiques sur différents supports (imprimantes, écrans).

L'API Java 3D fournit un support à la création de graphiques en trois dimensions. Cette API simplifie la programmation d'applications en trois dimensions et fournit des accès à des interfaces de bas niveau. Cette API est en relation étroite avec les API Audio, Video et MIDI.

L'API Java Media Framework manipule des médias dont l'utilisation dépend du temps : le son, l'image et le MIDI. Elle fournit un modèle de temps commun pour la synchronisation et la composition qui permet de fusionner différents types de médias. L'API Video définit les formats des données vidéo ainsi que l'interface de contrôle. L'API Audio permet la diffusion de sons synthétisés. L'API MIDI permet des flux d'événements en temps réel. Elle fait appel au *Media Framework* pour sa synchronisation avec d'autres activités.

L'API Java Share fournit les instructions de bases nécessaires à la communication entre objets situés sur le réseau.

L'API Java Animation permet l'animation traditionnelle en deux dimensions. Elle utilise les interfaces 2D et le Media Framework pour la synchronisation et la composition.

L'API Java Telephony permet l'intégration de communications téléphoniques. Elle fournit les fonctionnalités de base pour le contrôle des communications téléphoniques (elle inclut aussi la fonctionnalité de téléconférence).

L'API Java Entreprise permet l'utilisation de ressources propres à l'entreprise : les bases de données (JDBC : Java Database Connectivity), un langage de définition d'interfaces (IDL : Interface Definition Language) et une méthode d'appel à distance (RMI : Remote Method Invocation). JDBC est une interface d'accès aux bases de données standard de type SQL. Elle définit des classes permettant de représenter des constructions telles que des connections aux bases de données, des déclarations SQL, etc. IDL est une façon neutre de spécifier une interface entre un objet et son client lorsqu'ils sont situés sur des plates-formes différentes. RMI permet aux programmeurs de créer des objets dont les méthodes peuvent être appelées à partir d'une autre machine virtuelle.

L'API Java Server permet et facilite le développement de serveur Internet permettant le chargement d'applets Java.

L'API Java Commerce apporte la sécurité au niveau des transactions financières réalisées sur le Web. En effet, depuis quelques temps déjà, de nombreuses firmes permettent aux utilisateurs d'Internet de réaliser des commandes via des catalogues présentés sous forme de pages HTML. L'API Java Commerce permet de stocker des informations personnelles (nom et adresse de la firme ou du consommateur, moyen de paiement, etc.) ainsi que de permettre l'utilisation de nouveaux protocoles (de paiement tel que le SET : Secure Electronic Transaction).

L'API Java Management est un ensemble de classes Java qui facilitent le développement de solutions de management. L'API Java Management est composée de différents composants, chacun ayant trait à un aspect particulier du management (Admin View Module, Base Object Interface, Managed Notification Interfaces, Managed Container Interface, Managed Data Interfaces, Managed Protocol Interfaces, SNMP Interfaces)[6].

L'API Java Security facilite l'ajoute de fonctionnalités de sécurité dans les applets ou les applications développées par les concepteurs. Cette fonctionnalité inclut la cryptographie avec signatures digitales, le chiffrement et l'authentification. Cette architecture fournit un niveau de sécurité sans cesse meilleur. Chaque fois qu'un algorithme plus efficace ou

[6] des détails complémentaires concernant ces interfaces peuvent être obtenus à http://www.javasoft.com/doc

qu'une implémentation plus rapide est disponible, les modules peuvent être remplacés dans la plate-forme de façon totalement transparente pour les applications.

2.3 Les bénéfices tirés de la plate-forme Java.

Ces bénéfices se situent à deux niveaux différents. Pour l'utilisateur final, la plate-forme Java ajoute un contenu interactif sur le WWW. Les applications sont immédiatement disponibles pour chaque type de système d'exploitation, libérant ainsi l'utilisateur du choix d'un système d'exploitation particulier. Pour les concepteurs d'applications, la portabilité des fichiers binaires impliquent un moindre coût de développement puisqu'il est inutile de développer les applications pouvant être supportées par des systèmes d'exploitations différents. De plus, en développant des applications qui partagent des objets, les concepteurs peuvent se concentrer sur la création de nouveaux objets.

3 Le langage de programmation Java.

Java est un langage de programmation de haut niveau similaire au C, C++ et Pascal. C'est un langage simple, orienté objet, compilé, d'architecture neutre, multi-threaded, robuste et extensible.

3.1 Java est simple.

La plupart des programmeurs utilisent le C, et ceux utilisant la programmation orientée objet utilisent le C++. Java est fort semblable au C++ dont il omet certaines fonctions (rarement utilisées ou non absolument nécessaires) dans un souci de simplification. Les principales différences entre le langage Java et le C++ (ou le C) sont :

- Le langage ne possède pas de pointeurs tels que ceux utilisés en C puisqu'ils sont responsable d'un grand nombre de « bugs » dans les programmes. Il est par conséquent impossible d'accéder à une zone mémoire en utilisant les casts sur les pointeurs. Ils sont donc remplacés, dans le langage Java par des objets et des tableaux d'objets. La manipulation complexe des pointeurs est donc remplacée par l'accès aux tableaux via leurs indices.

Par exemple, nous pourrions définir une classe *Voiture* qui hérite des propriétés de la classe *Véhicule*, cette dernière héritant à son tour des méthodes de la classe *Objet*.

Java ne supporte que l'héritage simple. Cela signifie que chaque classe ne peut hériter des méthodes que d'une seule autre classe. Certains langages permettent l'héritage multiple; cela peut introduire une certaine confusion et rendre le langage trop compliqué.

Lors de la déclaration d'une classe en Java, il faut indiquer les types d'accès permis aux variables ainsi qu'aux méthodes de la classe. Les classes peuvent donc être déclarées comme étant *public* (les méthodes et les instances de variables sont disponibles pour toutes les autres classes), *protected* (cela signifie que les instances de variables et les méthodes ainsi désignées ne sont accessibles qu'aux sous-classes de cette classe et nulle part ailleurs) ou *private* (les méthodes et les instances de variables sont uniquement accessible à l'intérieur de la classe dans laquelle ils sont déclarés. Elles ne sont accessibles par aucune autre classe ni même par leurs sous-classes).

3.3 Java est compilé.

Avant l'exécution d'un programme Java, il faut tout d'abord le transformer en byte-code. Le byte-code est fort semblable aux instructions machines, ce qui permet aux programmes écrits en langage Java d'être très efficaces. Cependant, le byte-code n'est pas spécifique à une machine particulière. Il peut, dès lors, être exécuté sur un grand nombre d'ordinateurs différents sans avoir à recompiler le programme de départ.

Les programmes sources en Java sont compilés en "fichiers de classes", qui contiennent les représentations en byte-code des programmes en question. Dans les fichiers de classes Java, toutes les références aux méthodes et aux instances de variables sont faites par nom et sont résolues lors de la première exécution du code. Cela rend le code plus général et moins susceptible de changer, tout en restant aussi efficace.

3.4 Java est indépendant de l'architecture.

Comme les programmes Java sont compilés en byte-code, ils peuvent être exécutés sur n'importe quelle plate-forme acceptant Java. Il n'est donc pas nécessaire de recompiler un programme Java pour le faire fonctionner sur une autre machine.

Le langage Java est identique pour chaque type d'ordinateur. De manière surprenante, ce n'est pas le cas pour les langages de programmation moderne tels que le C et le C++. De ce fait, chaque compilateur et chaque environnement de développement est légèrement différent, ce qui rend le problème de la portabilité fort difficile à résoudre.

Le système Java fournit également une libraire exhaustive de classes qui fournissent des accès au système d'exploitation sous-jacent. Voici quelques unes des principales utilisées :

- java.util

  classes des utilitaires intéressants tels que les vecteurs, les énumérations, les propriétés,... .
- java.io

  classes gérant les flux d'entrées/sorties. Elles fournissent des accès au système de fichiers.

- java.awt

  (Abstract Window Toolkit) outil de création d'interfaces graphiques.

3.5 Java est Multi-Threaded

Le Multithreading permet de programmer des applications réalisant plusieurs opérations en parallèle. La plupart des systèmes informatiques modernes tels que Unix et Windows 95 permettent le multitâche. Le langage Java offre également cette possibilité.

Un programme Java peut avoir plus d'une tâche à exécuter en parallèle. Par exemple, un programme Java peut consister en deux tâches parallèles : la première s'occupant de l'introduction de paramètres (tels que les nom, prénom et adresse de l'utilisateur) et la seconde d'une petite animation rendant l'application plus conviviale. Grâce à ce type de programmation,

l'utilisateur ne doit donc pas différer l'introduction ses données en attendant que la partie du programme Java s'occupant de l'animation ait terminé cette dernière.

Il est souvent fort difficile de programmer dans ce type d'environnement parce que de nombreux événements peuvent se produire au même moment ou dans un ordre non prévisible. Java fournit des astuces de synchronisation qui facilitent la programmation par l'intermédiaire de la classe *Thread*[8]. Cette classe contient également un ensemble de méthodes permettant de lancer ou de terminer l'exécution d'une thread ainsi que de vérifier son status.

Un autre avantage de ce type de programmation est qu'il permet les interactions en temps réel. Ce type d'interaction est cependant limitée par la plate-forme sous-jacente. Par exemple, lorsque la plate-forme sous-jacente à la plate-forme Java est du type Unix, Macintosh ou Windows, les interactions sont moins rapides.

3.6 Java est robuste.

Java est conçu pour développer des applications qui doivent être robustes. L'accent est mis sur la détection la plus rapide possible des erreurs. Le compilateur Java emploie donc une grande partie de son temps à déceler des erreurs liées à la syntaxe, avant que le programme ne soit diffusé.

Un des avantages des langages fortement typés (comme le C++) est que certaines erreurs peuvent être décelées à la compilation. Java impose donc au programmeur la déclaration explicite des variables, des méthodes et de leurs types; contrairement au C pour lequel des déclarations implicites peuvent être réalisées.

La différence principale entre Java et le C++ est que Java possède une modèle de pointeur[9] qui élimine les possibilités de corrompre les informations ou la mémoire. De plus, les conversions de type casting ne sont pas permises (par exemple, il est impossible de convertir un entier en pointeur).

---

[8] cette classe appartient à la librairie *java.lang* du langage Java.

[9] au lieu d'utiliser l'arithmétique des pointeurs, Java utilise des tables de vérité.

Cependant, même les programmes Java peuvent avoir des erreurs. Si une situation inattendue se présente, le programme ne se "plante" pas, il génère une situation d'exception. L'interpréteur Java trouvera alors les exceptions correspondantes et les traitera en conséquence.

3.7 Java est extensible.

Il est également possible d'utiliser des programmes Java dans d'autres programmes écrits dans d'autres langages. Puisque les structures de données de Java sont fort semblables à celle du C, c'est relativement facile. Le plus grand problème est que la plupart des programmes ne sont pas "multi-threated".

Un programme Java peut déclarer une méthode comme étant *native*. Ces méthodes sont donc transformées en fonctions définies en librairies qui sont liées dynamiquement dans une machine virtuelle. Bien qu'il soit possible de lier dynamiquement n'importe quelle librairie de méthodes natives, cela n'est pas toujours permis (pour des raisons de sécurité).

4 La sécurité.

Java est conçu pour être utilisé dans des environnements distribués (une applet peut utiliser des informations stockées sur un autre site que celui où est exécuté cette applet). C'est pourquoi la sécurité a pris une telle importance dans ce langage. Les techniques d'authentification sont basées sur des clés publiques de chiffrement.

Puisque les applets Java sont importées sur notre ordinateur et exécutées automatiquement lorsque cela est possible, on pourrait penser qu'il existe un risque d'infection de l'ordinateur par des virus. Ce n'est pas le cas. Aucune applet Java n'est capable de voler de l'information ou d'endommager l'ordinateur de quelque façon que ce soit.

La raison pour laquelle les applets Java sont sûres est que les programmes Java sont compilés en instructions byte-code, qui peuvent être vérifiées. Les instructions byte-code sont fort similaires aux autres ensembles d'instructions utilisés par les ordinateurs, mais elles ne sont pas spécifiques à un système particulier et elles peuvent être vérifiées pour les violations de sécurités potentielles. Ceci est rendu possible parce que le byte-code de Java contient des

informations supplémentaires sur les types qui sont utilisées pour vérifier que le programme est légal.

Dans Java, l'accès aux méthodes et aux variables se fait toujours par *nom*. Cela facilite l'identification des méthodes ou fonctions qui sont actuellement utilisées. Ce processus est appelé la *vérification*. Il est utile pour s'assurer que le byte-code n'a pas été modifié et qu'il respecte les contraintes du langage Java.

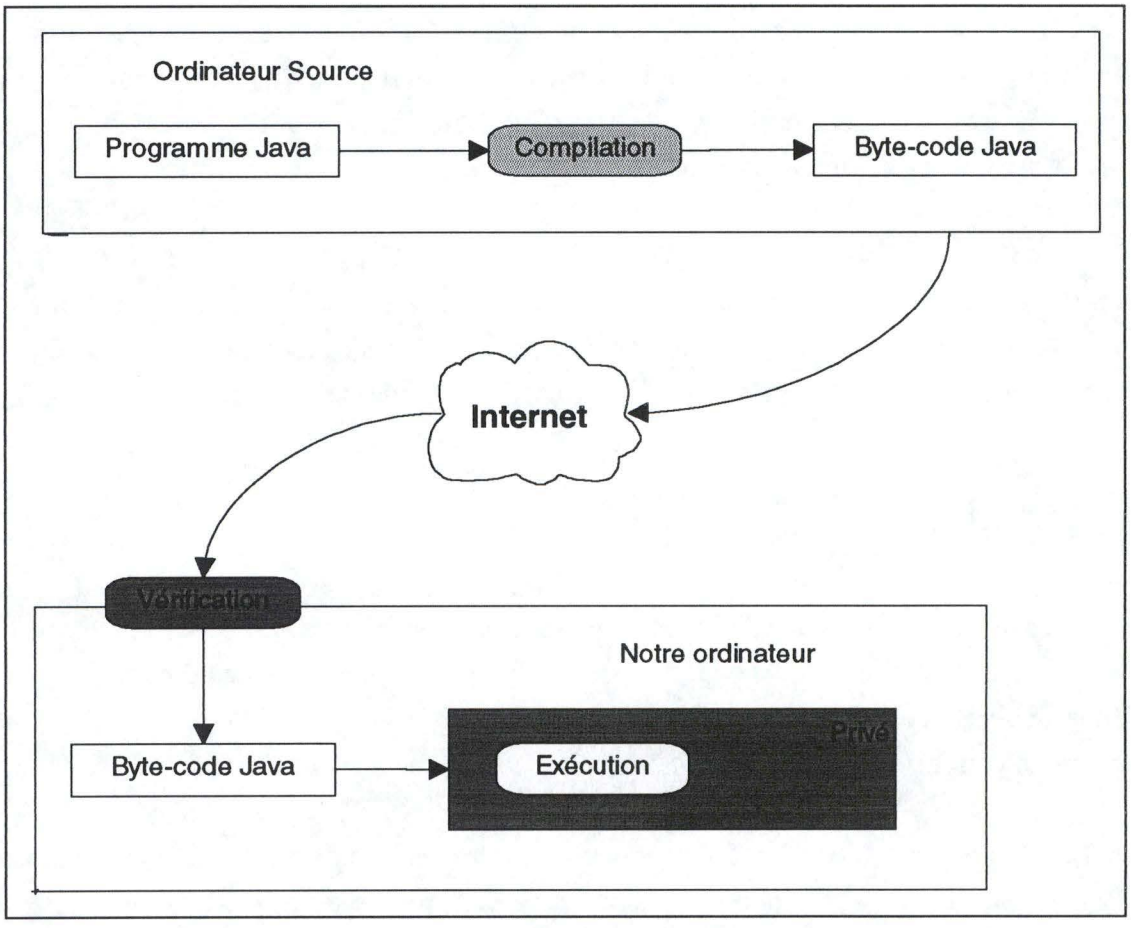

Figure 1.4 : Le processus de vérification du byte-code de Java

Une fois que l'applet a été vérifiée, elle est exécutée dans un environnement limité. Les applets ne peuvent exécuter certaines fonctions dangereuses dans cet environnement à moins qu'elles ne soient autorisées à le faire.

Dès que la vérification du byte-code a été réalisée, l'interpréteur Java transforme les noms de fonctions en adresses, comme pour un programme traditionnel. Cela signifie donc que les programmes Java peuvent être efficaces et sûrs à la fois.

5 Java et Internet.

Internet est un gigantesque réseau hétérogène qui relie des milliers d'ordinateurs. Tous ces ordinateurs utilisent le protocole TCP/IP pour communiquer. Cela signifie qu'un PC peut communiquer aussi facilement avec un Mac qu'avec un autre PC.

Puisqu'il existe un grand nombre de connections entre ordinateurs différents, nous avons besoin d'un langage qui n'est lié à aucune plate-forme particulière.

Les programmes Java sont transmis sous forme de byte-code, ce qui signifie qu'ils peuvent être exécutés sur une machine quelconque sans être compilés à nouveau. Il est possible d'importer un programme Java à partir de n'importe quel ordinateur sur Internet et de l'exécuter sans se soucier du système sur lequel le programme a été développé.

Avec les langages compilés traditionnels, ceci n'était pas possible. Ces langages imposent une compilation pour une plate-forme particulière, le programme résultant ne pouvant être exécuté sur un système différent. Pour utiliser un programme C sur de nombreuses plates-formes, un programmeur doit distribuer le code source et chaque utilisateur doit compiler ce programme avant de l'exécuter.

Afin de démontrer que Java est un bon langage pour la programmation sur Internet, l'équipe de développement de Java a décidé de l'utiliser pour implémenter un browser WWW au contenu interactif. Ce fut Hotjava.

Hotjava est capable d'importer des applets Java qui constituent des parties d'une page Web. Chaque applet est exécutée et affichée à l'intérieur du browser.

Hotjava a été le premier browser donnant la possibilité d'importer et d'exécuter des applets Java, mais il n'est plus le seul. Netscape Navigator 2.0 en est aussi capable.

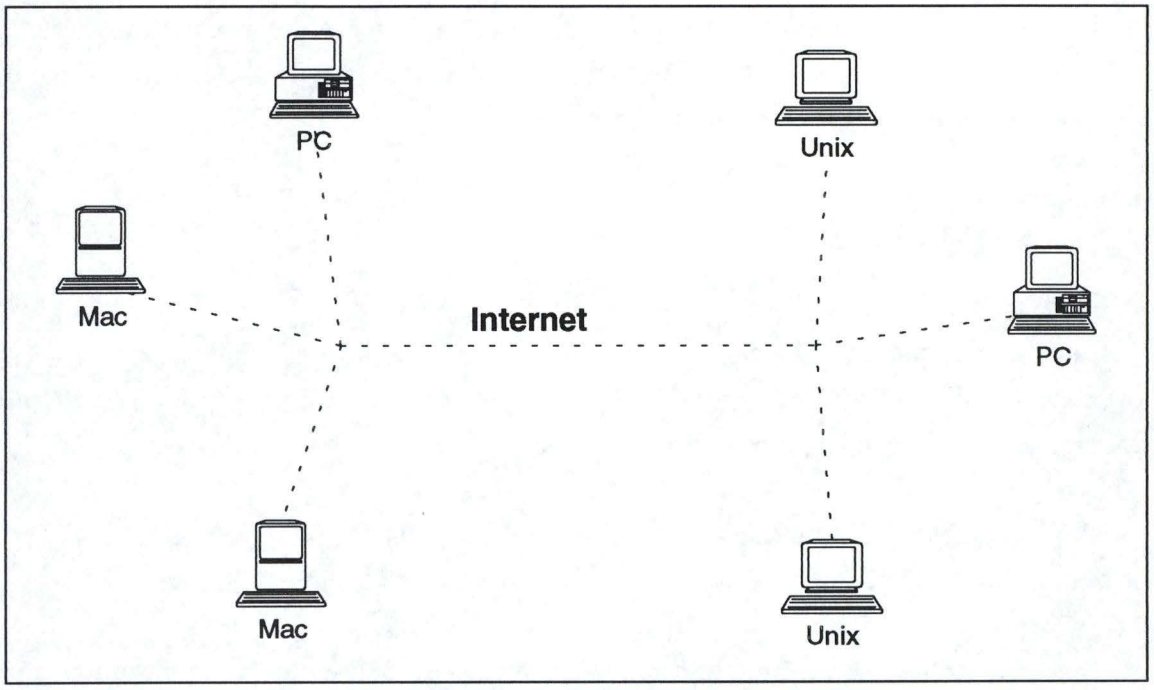

Figure 1.5 : L'hétérogénéité d'Internet

Chapitre 2 : Hypermédias et apprentissages.

Dans le domaine de l'éducation, des logiciels spécifiquement éducatifs existent et sont utilisés. Cependant, l'expression "Enseignement Assisté par Ordinateur" (E.A.O) se voit souvent remplacée par le terme "multimédia", pour désigner des applications gérant par logiciels des données de différents types (textuel, graphique, sonore).

Ces logiciels sont généralement conçus pour permettre l'exploration d'un ensemble de situations pédagogiques, la transition d'un sommet du graphe à un autre étant contrôlée par le logiciel en fonction des connaissances sur le domaine et sur l'apprenant. Cette vision pédagogique se paraphrase de la façon suivante : laissons à l'étudiant le choix de son objet d'étude, laissons-le décider s'il souhaite se soumettre à une évaluation, fournissons-lui un environnement stimulant.

Ce chapitre est divisé en quatre parties. Après une brève introduction, une description de l'hypermédia ainsi que ses usages va être réalisée. Par la suite, une description de l'ergonomie cognitive sera présentée à partir d'un ensemble de règles ergonomiques communément acceptées. Il se terminera par une synthèse des points présentés dans la troisième partie et qui figureront dans le cours sur Ethernet.

1 Introduction.

Des systèmes hypertextes et hypermédias pour micro-ordinateurs ont commencé à être exploités dans la seconde moitié des années quatre-vingt. Ils comportent maintenant de nombreux représentants[1], permettant de naviguer entre des sommets d'un graphe de situations. Ces sommets sont reliés entre eux par des "liens" que l'utilisateur peut activer pour se "transporter" ailleurs, en fonction de ses intérêts.

Une des caractéristiques communes de ces systèmes est la distinction faite entre différents modes d'utilisations : le mode auteur et le mode utilisateur final. Dans ce dernier, l'utilisateur n'a que la possibilité de naviguer entre les sommets du graphe. Ce mode correspond à la consultation d'encyclopédies, à la recherche d'informations, de documentations techniques, etc. Dans le mode auteur, on peut créer des sommets et des liens, les modifier, les supprimer, etc.

L'un des premiers intérêts de l'utilisation des hypermédias dans une situation pédagogique est certainement la rapidité et la facilité avec lesquelles un apprenant peut accéder à l'information. Enfin, un hypermédia peut, plus facilement qu'un livre, offrir au lecteur la possibilité de choisir le niveau de détail qu'il souhaite atteindre. Par exemple, à partir d'une information, il est possible d'accéder à toutes les données liées au sujet traité par un effet de "zoom", ou de se focaliser sur un aspect bien précis du domaine étudié. L'apprenant peut ainsi se familiariser avec des environnements et des démarches non linéaires.

Un hypermédia peut également aider un utilisateur à rapprocher des éléments d'informations pour les comparer, les confronter et les analyser. Cette possibilité permet à l'étudiant d'avoir plusieurs points de vue sur un même sujet (connaissances théoriques, exercices, simulations, études de cas, ...).

Les utilisations pédagogiques sont alors de deux ordres : « *à partir d'un vaste éventail de situations, l'étudiant peut abstraire et généraliser en dégageant un concept sous-jacent; ou, à l'inverse, il peut vérifier si une connaissance abstraite trouve une application dans tel ou tel cas particulier* » [BAR 91]. Quel que soit le mode choisi, il

[1] par exemple : les projets LOUPE [BOR90], LOUTI [PAQ 88], FORUM [SOU 94]

s'entraîne à sélectionner l'information selon des critères de pertinence qu'il doit définir en fonction de son objectif initial parmi les possibilités offertes par le système.

Les hypermédias peuvent être des outils efficaces pour soutenir les activités de synthèse et de production d'un élève. La possibilité d'adjoindre à une base d'information des annotations diverses (commentaires, critiques, questions, ...) est non seulement une aide à la mémorisation, mais aussi un entraînement à l'évaluation et à l'assimilation. De plus, s'il est possible de créer de nouveaux liens entre les noeuds existant, et d'ajouter de nouveaux noeuds, l'élève a ainsi l'opportunité de structurer concrètement ses acquis, voire de bénéficier de ceux d'autres étudiants d'un groupe.

Enfin, la création d'hypermédias par l'enseignant est sans doute une activité permettant à un apprenant d'acquérir des compétences plus complexes. En effet, à partir de la recherche d'informations pertinentes et de leur décomposition en unités cohérentes jusqu'à la conception des liens entre elles, l'enseignant peut élaborer un ensemble de situations d'apprentissage au cours desquelles les étudiants seront amenés à approfondir leur connaissance d'un domaine.

2 Description et usages de l'hypermédia.

L'hypermédia est d'abord décrit en termes informatiques : c'est une base de données textuelle, visuelle et sonore où chaque îlot d'information est appelé un *noeud*; l'ordinateur établit des liens entre certains noeuds et peut ainsi permettre un mouvement rapide dans cette masse d'information; une interface permet l'interaction entre l'usager et l'hypermédia.

La figure 2.1 montre un exemple de liens entre noeuds dans Netscape. Les noeuds sont des pages HTML alors que les liens sont des parties de texte souligné et ayant une couleur différente du texte normal. *CSMA/CD*, *réseaux* et *trames* sont des liens vers d'autres noeuds, c'est-à-dire vers d'autres pages HTML.

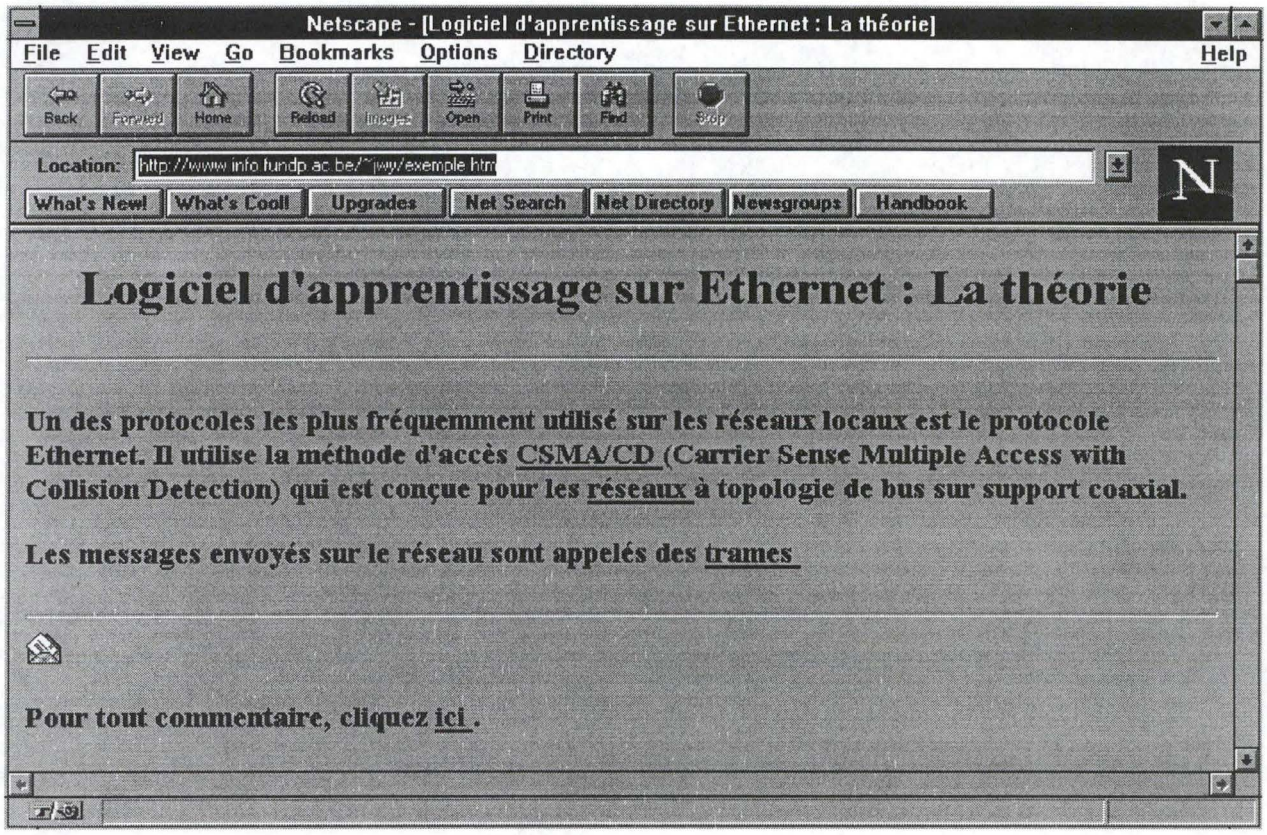

Figure 2.1 : Un exemple de liens dans Netscape.

2.1 L'exploration d'une vaste base de données.

Cet usage est facile à imaginer. Il y a beaucoup d'information disponible et l'utilisateur apprend simplement en se "promenant", en parcourant les liens accessibles à partir des informations rencontrées précédemment. Cet usage ne favorise pas l'efficacité de l'apprentissage mais en suggère plutôt les nombreuses potentialités.

Dans cette perspective, l'hypermédia devient un outil qui dynamise en quelque sorte l'encyclopédie. La question devient celle de l'opportunité d'une encyclopédie dynamique en éducation. Le principal attrait de cette approche est la liberté laissée à l'apprenant. Ceux qui parlent de transfert [PAL 90] disent que « *par cette approche le réseau sémantique de l'expert pourrait être relié à celui de l'apprenant* ». D'un autre point de vue, l'apprenant s'aperçoit que tout n'est pas relié par de simples liens de cause à effet et qu'en définitive, tout est interdépendant.

La décontextualisation par l'hypermédia est une tare qui est éveillée par la navigation dans de tels systèmes soit parce que l'information désirée n'est pas trouvée ou que la désorientation ne permet pas de replacer les items d'informations dans un tout reconnaissable. Les questions des usagers sont simples mais primordiales : d'où est ce que je viens, où suis-je, où est-ce que je vais ? Une foule de techniques permettent d'atténuer les symptômes de cette désorientation : signets, retours au point départ,... .

De façon similaire, le surcroît d'information place l'usager dans une situation défensive. Il résiste aux nouvelles parcelles d'informations, aux fenêtres qui se multiplient, aux liens non immédiatement significatifs parce que tout cela n'est pas perçu comme pertinent, n'est pas intégré dans une structure que l'usager possède ou simplement parce que l'information est trop abondante.

2.2 L'accès à une information spécifique.

L'accès à une information spécifique réduit déjà l'ampleur de la base de données au domaine proposé par l'apprenant. Comme dans toute approche pédagogique traditionnelle, seule l'information pertinente est disponible. L'accent est placé sur l'efficacité de l'apprentissage où l'hypermédia devient un nouveau genre de didacticiel. La navigation laisse un sentiment de liberté mais il n'y a pas de risque d'égarement hors matière. L'étudiant doit acquérir une information de base. Si l'information correspond à l'attente de l'étudiant, il poursuit sa démarche, sinon, il demande une série d'élaborations sous forme d'exemples ou d'explications. C'est de cette façon que l'enseignement s'individualise. L'hypermédia permet donc de répondre aux exigences personnelles des étudiants quand il s'agit de comprendre des concepts ou des relations entre les concepts dans un domaine bien particulier. L'hypermédia n'apporterait que les élaborations pertinente, sans surcharge cognitive, sans risque de désorientation et cela au moment opportun. Son atout principal serait l'efficacité d'un tel système pour l'apprentissage.

L'hypermédia qui a un rôle didactique spécifique doit posséder certaines caractéristiques de composition. « *Encore plus que dans un manuel, la matière doit être structurée de manière à être regardée sous divers angles. Chaque îlot d'information doit être suffisamment explicite et autonome pour ne pas exiger de cheminement préalable* » [RHE 91]. Cela est aussi vrai au plan des idées que de l'expression. Tant que les

informations sont composées d'éléments de même nature, le remplacement d'un élément par un autre ne crée pas de désorientation chez l'apprenant mais dès que ces éléments sont reliés les uns aux autres par une certaine causalité, la structuration prend une certaine importance.

2.3 La personnalisation d'une base de données.

La personnalisation d'une base de données fait ressortir la dimension utilitaire de l'hypermédia. L'outil permet de refaçonner une base de données existante pour répondre aux besoins spécifiques de cet usager. Par la classification, la compilation, l'analyse des données, la représentation visuelle et l'enchaînement des données, l'hypermédia devient porteur de sens. Si on admet que le travail avec l'information peut susciter l'apprentissage, on peut dire que « *l'hypermédia peut contribuer à une pédagogie de la construction, de l'innovation* »[RHE 91]. Cela s'effectue essentiellement de trois façons :

- par la possibilité de juxtaposer des îlots d'information;
- par la possibilité d'annoter la base de données en y ajoutant des commentaires personnels qui à la manière du souligné dans l'imprimé laisse la trace du nouvel auteur;
- par la possibilité de créer des liens personnalisés.

Cette approche fait en sorte que tel hypermédia n'est jamais un produit terminé mais demeure un lieu d'expression, de mémoire et de communication en constante évolution. Comme le principe de l'hypermédia est essentiellement de créer des liens entre des noeuds d'information, toute mise à jour ou relecture crée au moins des appendices à la structure de base. Ce changement peut apparaître comme une amélioration ou une détérioration aux yeux de l'auteur qui revisite son oeuvre transformée. Il reste néanmoins à savoir comment le nouvel auteur prétend avoir appris en modifiant ou comment il prétend avoir amélioré l'hypermédia pour le soumettre d'une manière itérative à un nouvel auteur. Dans cette optique, les techniques de :

- préservation des versions originales;
- d'annotations personnelles sur des champs auxiliaires prévus à cette fin;
- recherche dans un hypermédia par des termes différents de ceux exprimés par le premier auteur contribuent à atténuer l'effet des améliorations négatives et successives.

La personnalisation d'une base de données répondant aux caractéristiques de l'hypermédia fait de tout usager un auteur au sens véritable. Tout comme dans le cas de la personnalisation d'une base de données, les applications potentielles sont variées et porteuses de sens. Dans un contexte d'apprentissage individualisé, l'hypermédia s'impose puisque l'étudiant a comme tâche de bâtir ses propres connaissances, à partir de sa réflexion et de ses lectures ou en réorganisant en hypermédia une base de données. D'autre part, la collaboration peut s'effectuer entre le professeur et les étudiants : l'expert, le guide et l'apprenant confrontent des points de vues qui bâtissent une situation d'apprentissage.

En tant que média, il faut considérer les techniques de composition et tenir compte des caractéristiques du système utilisé. Composer un hypermédia, c'est essentiellement créer des noeuds et des liens. Ces noeuds doivent contenir une seule idée bien articulée et bien identifiée par un titre. La taille du noeud devrait correspondre à l'espace de la mémoire à court terme selon la technique de *l'information mapping*. Selon cette même technique, quatre principes devraient caractériser ces noeuds :

- l'information doit être partagée en petites unités ou blocs (ce qui correspond à un noeud);
- un noeud ne doit contenir que l'information relative à un aspect de la question;
- dans un sujet donné, les blocs d'information doivent présenter une certaine similitude quant aux mots, aux titres, aux formats et aux séquences;
- chaque noeud doit être étiqueté suivant des critères spécifiques.

D'autre part, les liens doivent établir des relations pertinentes entre les noeuds : unité de classe et de genre. Jamais les relations ne doivent être gratuites. Le lecteur resterait alors bouche bée avec sa quête de sens. Les liens doivent donc en quelque sorte établir le tableau d'ensemble qui montre le contour d'une question. Dans un autre ordre d'idée, la présentation doit être soignée. En effet, la lisibilité de tous les éléments graphiques ou textuels doit être bien assurée. L'écran doit être agréablement disposé, sans surcharge, « *ce qui crée une désorientation spatiale qui ne fait que précéder une désorientation cognitive* » [RHE 91]. Il ne faut pas compter sur les souvenirs du lecteur d'un écran à l'autre. Les idées ou les noeuds d'information doivent être reliés mais les textes et les écrans doivent être suffisamment autonomes et complets en eux-mêmes.

3 Hypermédias et ergonomie cognitive.

Que ce soit lors de l'élaboration d'un logiciel traditionnel ou d'une application de type Enseignement Assisté par Ordinateur (E.A.O), la conception d'une interface homme-machine est un des aspects les plus important de la phase de développement du produit. De nombreuses règles appelées règles ergonomiques ont pour but de guider le concepteur dans sa phase de développement de l'interface. Bien que l'ensemble de ces règles soient applicables à un logiciel de type E.A.O, seules certaines d'entre elles ayant une connotation pédagogique sont utilisées dans ce cadre de travail.

Le but de cette partie est de présenter les fondements relatifs à la démarche de conception d'une interface homme-machine générale et d'étendre ensuite ces notions vers des règles de conception d'un didacticiel.

3.1 Les règles de conception d'une interface homme-machine.

Les règles ergonomiques sont destinées à aider le concepteur dans sa tâche de développement d'une interface homme-machine; elles ont pour but de faire respecter un certain nombre de critères ergonomiques. Le description suivante sera basée sur l'ouvrage « Guide ergonomique des interfaces homme-machine » [VANDER 94] de Jean Vanderdonck, assistant à l'Institut d'Informatique des F.U.N.D.P.

3.1.1 Les critères ergonomiques.

> « *Un critère ergonomique constitue une dimension reconnue sur le chemin qui mène à une interface élaborée, efficace, sophistiquée, plus conviviale et moins encline à l'erreur* » [VANDER 94].

Tous les choix effectués en matière de conception d'une interface homme-machine sont fait en fonction de ces critères. A l'heure actuelle huit critères sont retenus par les personnalités de ce domaine. Il s'agit de la compatibilité, la cohérence, la charge de travail, l'adaptabilité, le contrôle du dialogue, la représentativité, le guidage et la gestion des erreurs.

3.1.1.1 La compatibilité

« *Une interface homme-machine est qualifiée de compatible si et seulement si le (re)codage d'information et de tâches du monde réel en données et actions du système est réduit* » [VANDER94]

On peut dire qu'une interface est d'autant meilleure qu'elle est compatible car la compréhension d'informations est d'autant plus rapide que le codage en données du système est réduit. La compatibilité représente donc la cohérence de l'interface homme-machine avec l'environnement extérieur de l'application.

Le but poursuivi par le respect de ce critère est la réduction du besoin de traduire et d'interpréter l'information réelle en données du système.

3.1.1.2 La cohérence

« *Une interface homme-machine est qualifiée de cohérente si et seulement si les données et les actions sont facilement identifiables, reconnaissables et utilisables* » [VANDER 94].

Un utilisateur distingue d'autant mieux les données et exécute d'autant mieux les actions que celles-ci sont représentées de manière stable et uniformisée. La cohérence traduit donc la stabilité de présentation de l'application vis-à-vis d'elle-même ou vis-à-vis d'autres applications.

Le but poursuivi par le respect de la cohérence est de toujours utiliser les mêmes moyens pour atteindre les mêmes résultats dans des contextes similaires.

3.1.1.3 La charge de travail

« *Une interface homme-machine est qualifiée d'efficace en charge de travail si et seulement si le volume de données à manipuler et d'actions à accomplir par unité de tâche est réduit* » [VANDER 94].

L'utilisation d'une application est d'autant plus efficace et agréable que la manipulation des données par l'utilisateur est courte et simple. Ce critère remplit un double rôle : garder la charge de travail dans les limites de capacité des facultés humaines et garantir une performance.

3.1.1.4 L'adaptabilité

« *Une interface homme-machine est qualifiée d'adaptable si et seulement si elle possède la faculté de mimétisme comportemental vis-à-vis de son utilisateur* » [VANDER 94] .

Un utilisateur est d'autant plus enclin à utiliser une application et acquiert d'autant plus de maîtrise que l'interface s'adapte aux différents contextes de travail. Une interface personnalisable, par exemple, traduit une adaptabilité importante de l'application en question.

Le respect de ce critère lors de la conception d'une interface a pour but d'offrir à l'utilisateur plusieurs voies pour accomplir sa tâche; l'adaptabilité ne concerne donc pas uniquement les paramètres visuels de l'interface mais également les paramètres de fonctionnement de l'application.

3.1.1.5 Le contrôle du dialogue

« *Une interface homme-machine est qualifiée d'interface à contrôle explicite si et seulement si elle peut fournir à l'utilisateur l'illusion qu'elle est placée sous son contrôle et que ses actions s'exécutent suite à des demandes explicitement formulées par l'utilisateur. Une interface est qualifiée d'interface à contrôle implicite si et*

seulement si elle est placée sous le contrôle du système. Elle est qualifiée à contrôle mixte lorsqu'elle est tour à tour à contrôle explicite et implicite » [VANDER 94].

Le but de ce critère est de fournir un contrôle maximum du déroulement du dialogue à l'utilisateur et de n'accomplir une action que lorsque l'utilisateur a spécifié de façon formelle son désir d'exécuter cette action.

3.1.1.6 La représentativité

« *Une interface homme-machine est qualifiée de représentative si et seulement si les codes utilisés, les items de menu, les libellés facilitent l'encodage et la rétention* » [VANDER 94].

Un utilisateur a d'autant plus de facilités à utiliser une application que les codes utilisés dans son interface sont clairs et explicites. Le but de ce critère est de répandre l'usage de dénominations significatives au sein du dialogue. Il rejoint en quelque sorte le critère de cohérence.

3.1.1.7 Le guidage

« *Une interface homme-machine est qualifiée d'efficace en guidage (ou en feedback) si et seulement si elle informe de manière constante l'utilisateur sur l'issue de ses actions et sur sa position dans l'accomplissement de sa tâche* » [VANDER 94].

Un utilisateur a d'autant plus de facilités et est d'autant plus efficace à accomplir sa tâche qu'il est guidé à travers toutes les étapes nécessaires pour la mener à bien.

Le but poursuivi par le respect de ce critère est de fournir à l'utilisateur une aide sur ce qu'il peut entreprendre, sur la situation dans laquelle il se trouve et sur les résultats des actions effectuées. Cette aide doit être conçue de manière optimale du point de vue de la lisibilité.

3.1.1.8 La gestion des erreurs

> « *Une interface homme-machine est qualifiée d'efficace en gestion des erreurs si et seulement si elle s'avère robuste vis-à-vis des erreurs commises par l'utilisateur et se montre conviviale dans sa manière de les corriger* » [VANDER 94].

Les performances de réalisation d'une tâche à l'aide d'une application sont d'autant plus élevées que les occasions d'erreurs sont réduites.

3.1.2 Les règles ergonomiques.

L'ensemble des critères ergonomiques décrit ci-dessus sont respectés lors de la conception d'une interface homme-machine par l'utilisation de règles appelées règles ergonomiques.

> « *Une règle ergonomique constitue un principe de conception et/ou d'évaluation à observer en vue d'obtenir et/ou de garantir une interface homme-machine ergonomique* » [VANDER 94].

Chacune de ces règles s'inscrit dans une taxonomie arborescente comprenant 12 divisions : la saisie de données, l'affichage de données, le dialogue, le graphisme, les moyens d'interaction, les styles d'interaction, le guidage de l'utilisateur, les messages, l'aide en ligne, la documentation, l'évaluation et l'implémentation.

Règles ergonomiques	Desciptions
La saisie des données	ensemble des règles ergonomiques régissant l'interface à développer pour assurer une saisie des données. Les règles comprises dans cette division vont des règles d'utilisation du clavier pour la saisie aux règles précisant les types d'objets interactifs à utiliser en fonction du type d'information à saisir en passant par des règles spécifiant la forme des curseurs à utiliser.

Règles ergonomiques	Descriptions
L'affichage des données	ensemble des règles ergonomiques qui régissent la conception d'une interface ou d'une partie d'interface assurant l'affichage d'informations à l'écran. C'est ici que l'on retrouve entre autres les règles spécifiant l'affichage de données formatées telles qu'une date, les règles spécifiant le format des écrans à utiliser pour l'affichage de tel ou tel type de données.
Le dialogue	ensemble des règles ergonomiques spécifiant le fonctionnement du dialogue. Ces règles vont de la spécification du type de dialogue à choisir en fonction du type d'utilisateur jusqu'aux règles précisant la façon d'implémenter une pause dans le dialogue.
Le graphisme	ensemble des règles ergonomiques régissant tout ce qui est relatif au graphisme dans une interface. Elles constituent des conseils à suivre en ce qui concerne l'utilisation des caractères, l'utilisation de graphiques animés ou non, l'emploi de couleurs, d'icônes ou de symboles.
Les moyens d'interaction	ensemble des règles ergonomiques qui régissent l'utilisation de tous les périphériques permettant une interaction avec la machine. Trois dimensions permettent de caractériser ces moyens d'interactions : les moyens d'interaction de saisie (le clavier, les codes à barres), les moyens d'interaction d'affichage (l'écran tactile, la souris, le « joystick ») et les moyens d'interaction multimédia (la voix, le gant sensoriel).
Les styles d'interaction	ensemble des règles qui sont d'application selon le style d'interaction qui a été choisi ainsi que les règles permettant de choisir un style d'interaction particulier. Les différents styles d'interaction reprennent entre autres le langage de commande, la sélection de menu, le remplissage de formes, l'interaction graphique, etc.

Règles ergonomiques	Descriptions
Le guidage de l'utilisateur	ensemble des règles ergonomiques indiquant comment organiser le guidage de l'utilisateur dans l'interface. Ces règles permettent, par exemple, de préciser comment gérer les erreurs ou encore comment valider ergonomiquement l'information fournie par l'utilisateur.
Les messages	ensemble des règles ergonomiques régissant l'organisation spaciale ou syntaxique des différents messages que peut fournir une interface en fonction du type de message (de guidage, d'aide, d'erreur, etc).
L'aide en ligne	ensemble des règles ergonomiques précisant comment concevoir et implémenter une aide en ligne du point de vue du contenu, présentation ou accessibilité.
La documentation	ensemble des règles ergonomiques caractérisant la conception d'une documentation appropriée au produit développé. Cet ensemble de règles reprend celles concernant la conception d'une documentation en ligne aussi bien que celles concernant la conception d'une documentation extérieure (le manuel de l'utilisateur).
L'évaluation	ensemble des règles ergonomiques à appliquer lors de l'évaluation quantitative (la distance entre les objets interactifs) ou subjective (aspect extérieur plaisant ou non) de l'aspect de l'interface.
L'implémentation	ensemble des règles ergonomiques qui constituent autant de conseils d'implémentation aux concepteurs d'une interface homme-machine. Ces règles ne constituent en aucun cas des méthodologies d'implémentation de logiciel mais des recommandations concernant les différents facteurs à ne pas négliger lors de la phase d'implémentation de l'interface de l'application développée.

Par exemple, la figure 2.2 montre une interface développée pour la saisie de paramètres. La règle ergonomique de *graphisme* indique notamment qu'étant donné la longueur des chaînes de caractères précédant les listes de sélection, les messages doivent être alignés sur les « : » précédant les listes de sélection.

Des exemples supplémentaires de règles appartenant à ces divisions peuvent être obtenu dans le « Guide ergonomique des interfaces homme-machine » [VANDER 94].

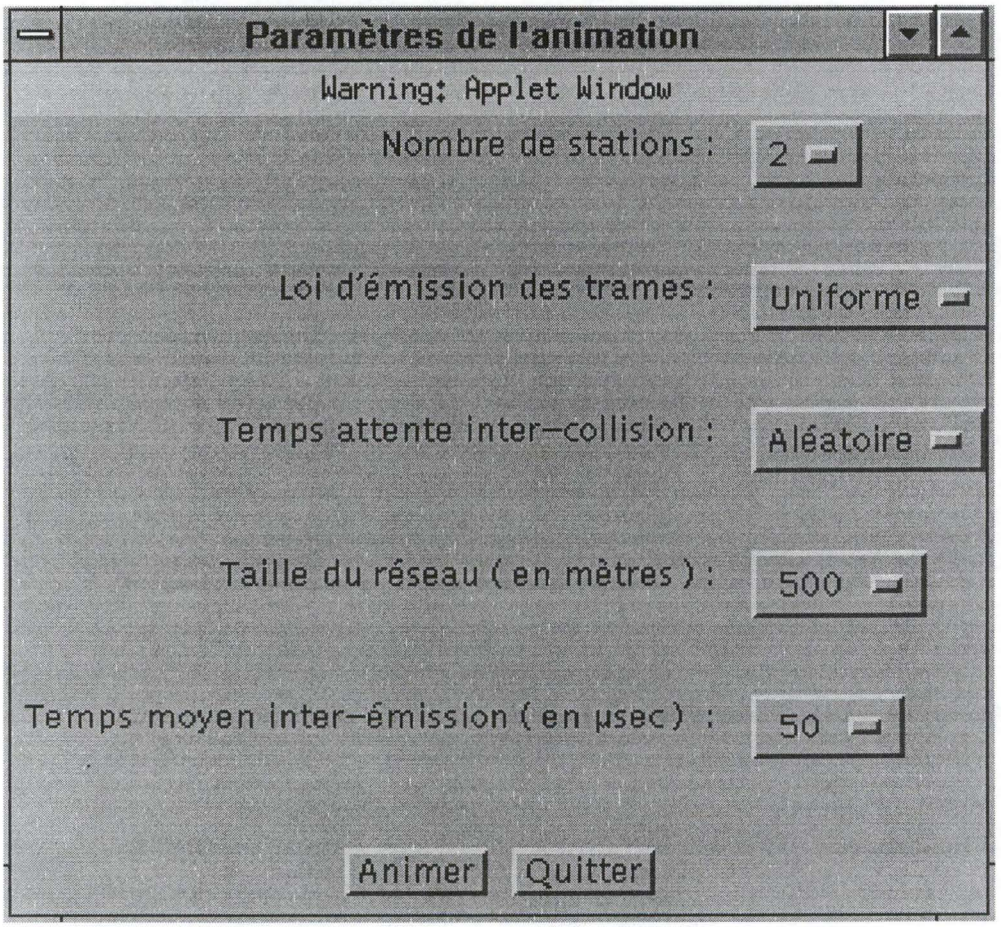

Figure 2.2 : l'interface s'occupant de la saisie de paramètres.

Remarquons que ces règles ergonomiques peuvent être utilisées dans deux contextes diamétralement opposés : elles peuvent être utilisées soit lors de la conception d'une interface, soit lors de l'évaluation d'une interface existante.

Dans un contexte de conception d'une interface homme-machine, le concepteur utilisera les règles ergonomiques qu'il estime adéquates (ou prépondérantes) en fonction du style d'interaction adopté, des objets interactifs utilisés et des moyens d'interactions impliqués.

Dans un contexte d'évaluation d'une interface homme-machine existante, la tâche consiste à préciser quelle tâche peut être exécutée à l'aide de cette interface et quels sont les profils des utilisateurs de l'interface. Une fois ces deux paramètres précisés, il faut encore étudier l'application pour en déterminer la structure interne. L'analyse ergonomique proprement dite peut commencer dès la fin des étapes précédentes, elle est divisée en deux phases :

- la première phase est l'analyse fonctionnelle de l'interface qui permet de faire l'inventaire des types de composant de cette interface (les moyens d'interactions, les objets interactifs et les styles d'interactions).
- la seconde phase est l'analyse ergonomique et consiste en la vérification du respects des règles ergonomiques adéquates ainsi qu'en la détermination des critères ergonomiques qui sont le plus souvent mis en jeu dans l'interface analysée.

3.2 L'extension des règles ergonomiques à l'ergonomie cognitive

Les systèmes hypermédias offrent généralement une structure relativement statique par rapport aux environnements d'enseignement assisté de type simulation ou dialogue. En effet, les éléments de texte et d'animation y sont relativement fixes, bien qu'accessible de façon structurée. Le contrôle de l'interaction étant d'avantage laissé à l'utilisateur, il convient de se demander de quelle façon on peut introduire des fonctions de support à l'apprentissage. Les recherches[2] sur les interactions homme-machine montrent que divers aspects de l'interface peuvent apporter du support à l'usager en situation d'apprentissage, d'abord de façon statique puis de façon dynamique. Diverses méthodes permettent de cerner l'impact de ces caractéristiques sur la navigation, les attitudes et l'apprentissage.

[2] cfr. [DUF 88]

3.2.1 Aspects statiques de l'interface et apprentissage sur les hypermédias.

L'ergonomie cognitive s'intéresse au problème d'organisation de la pensée face à une tâche, c'est-à-dire au problème de l'interface entre l'espace cognitif et l'espace de la tâche. Dans l'utilisation des systèmes informatiques, plus le fossé est large, plus il est difficile de transcrire ce que l'on pense dans le langage du système et plus il est difficile d'interpréter ce que le système fournit dans les termes de ce que l'on cherche. Quels sont les éléments de l'interface qui peuvent favoriser la transparence, en rapprochant l'espace des connaissances à apprendre de l'espace cognitif de l'usager ? Différents aspects ont fait l'objet de recherches systématiques dans un contexte d'apprentissage.

3.2.1.1 Structure de l'information.

Le premier élément de l'interface est la structure arborescente. A ce niveau diverses recherches [CHI 86] ont étudié les aspects ergonomiques de la composition des menus en relation avec la charge cognitive et l'efficacité dans la recherche de l'information. Ainsi, les menus plus larges conduisent à moins d'erreurs, mais supposent un temps de traitement plus long; un découpage orthogonal des rubriques, l'organisation verticale, le regroupement sémantique et l'utilisation de titres ou d'exemples dans le menu facilitent la sélection. Une icône ou l'emplacement d'un élément peut également servir de repère sur le contenu. De cette façon, certaines organisations des menus peuvent être assimilées plus facilement par l'usager et favoriser l'orientation, la compréhension et éventuellement la rétention du contenu.

De plus, dans un contexte d'apprentissage, la structure des arborescences peut chercher à suivre l'organisation cognitive qui sera faite du contenu. Elle peut considérer non seulement la représentation finale à développer, mais également les représentations intermédiaires par lesquelles cette élaboration se fait. Ainsi, une structure d'hypermédia peut épouser une organisation de type *graphe génétique*, qui présente successivement l'information du plus simple au plus complexe en intercalant des situations conflictuelles pouvant favoriser les restructurations cognitives exigées. La présentation des informations et des obstacles peut être organisée de façon à laisser l'utilisateur relativement libre dans son exploration.

3.2.1.2 Indices du contexte et de la progression.

Dans les systèmes informatiques, la rétroaction est importante pour que l'usager puisse constater l'impact de ses actions et ajuster celles-ci à ses intentions. La désorientation ressentie dans les systèmes hypermédias est souvent liée à cette incapacité de l'usager qui passe d'une information à une autre, de se rappeler d'où il vient et ce qu'il cherchait à faire. Certains outils comme l'historique du parcours facilitent la visualisation et l'accès à ce qui vient d'être vu. On peut aussi utiliser des *empreintes* qui sont des indices laissés pour montrer qu'un information a déjà été lue. Ces éléments du contexte n'ont pas toujours à être manifestés de façon explicite pour ne pas surcharger le processus de compréhension lui-même, ils peuvent être signifiés par des couleurs, des sons, etc. Ces éléments de rétroactions servent non seulement au plan cognitif à l'usager à organiser son interaction, ils jouent également un rôle sur le plan de la motivation.

3.2.1.3 Indexation des informations.

Il est souvent intéressant d'offrir plus d'un mode d'accès à l'information. Par exemple, selon les cas, une forme d'accès plus qu'une autre peut favoriser l'organisation des représentations chez l'usager : carte, mots clés, menus déroulants, etc.

De plus, le fait d'offrir plus d'une façon d'accéder à la même information peut favoriser l'acceptation du système tant par des novices que par des experts.

3.2.1.4 Complexité de la présentation.

La complexité de l'interface peut influencer sa compréhension. Diverses mesures de la complexité peuvent être utilisées, inspirées entre autres par les études sur la lisibilité et les textes d'instruction : proportion d'espaces vides, longueur des phrases, nombre de mots rares, etc.

3.2.2 Aspects dynamiques de l'interface et apprentissage sur les hypermédias.

Au delà des aspects statiques qui sont une extension des contraintes textuelles, les aspects dynamiques de l'interface touchent à la gestion de l'interaction par le système. Ainsi,

même à l'intérieur d'une structure statique d'information, la navigation de l'usager peut être contrôlée par le système.

3.2.2.1 Récupération dynamique du contrôle par le système.

L'accès aux informations peut être limité de façon dynamique, en fonction de ce qui a déjà été vu et réussi par l'apprenant. Ensuite, un certain nombre de messages d'aide ou de correction peuvent être fournis par le système. « *Il est important que s'harmonisent bien la zone de contrôle laissé à l'usager et celle récupérée par le système. Les différentes interventions tutorielles doivent être négociées pour ne pas nuire à la cohérence et aux processus cognitifs de l'apprenant* »[DUF 91].

Par exemple, un item *Couper* qui apparaîtrait dans un menu de la figure 2.3 serait une source de confusion pour l'utilisateur. Il est préférable de le montrer tout en signalant qu'il est momentanément inaccessible (en l'affichant par exemple en caractères grisés).

De la même façon, l'usager doit être averti lorsque le système prend le contrôle et l'amène à des informations complémentaires; il doit éventuellement être libre de les consulter.

Figure 2.3 : Un exemple d'items dans un menu.

3.2.2.2 Le modèle de la tâche.

Une tâche est définie comme « *une activité dont l'accomplissement par un opérateur (souvent appelé aussi utilisateur) produit un changement d'état significatif d'un domaine d'activité donné dans un contexte donné* » [VANDER 94].

Le modèle de la tâche doit idéalement être implicitement inscrit dans l'organisation du contenu, en respectant la hiérarchie des pré-recquis, les séquences. Les éléments de ce modèle servent à supporter le contrôle du système, mais aussi à modifier les indices de progression dans un contexte de flexibilité d'accès. Ainsi, les indices de progression doivent tenir compte de l'ensemble des chemins et des étapes possibles, c'est-à-dire signifier quels aspects ont été vus (même indirectement) peu importe par quel chemin.

3.2.2.3 Le modèle de l'usager.

Le modèle de l'usager correspond aux connaissances consultées et/ou manifestées dans les tests par l'usager. Ici encore, il est construit de façon dynamique et distribuée par le système, c'est-à-dire que chaque item d'information garde une trace des consultations et des réponses de l'usager. En effet, il est important d'avoir un modèle de l'usager qui comporte non seulement une dimension à court terme, mais aussi une dimension à plus long terme, pour reconnaître, par exemple, un type d'erreur qui se répète.

4 Synthèse des caractéristiques ergonomiques du cours.

Le développement d'une interface dépend de certains paramètres : la population cible du produit, la raison d'être du produit, les démarches pédagogiques, les activités solicitées chez l'apprenant, ...

Une série de questions peuvent nous amener à mieux cerner les caractériques propres à l'application développée :

Pour quelles raisons la conception du produit a-t-elle été décidée ?
Quel est le degré d'insertion du produit dans le contexte de formation ?

Le cours sur Ethernet s'insérait dans un contexte de formation. Ce cours est réalisé à l'initiative de l'Université de Rennes où un cours sur les protocoles utilisés dans les réseaux locaux est dispensé. Ce logiciel a pour but d'apporter un support complémentaire au cours concernant le protocole Ethernet. A longue échéance, ce cours a l'ambition de remplacer le cours donné en auditoire. De plus, ce cours sert également de prototype afin de vérifier, sur un cas pratique, l'efficacité de ce type de support pour l'enseignement.

Quelle est la population à qui le produit est destiné ?

A l'origine, cette population se composait exclusivement d'étudiants. Mais compte tenu du support de diffusion (Internet), cette population s'est étendue à l'ensemble des personnes ayant accès au Web. Cela implique donc un changement quant à la façon de penser le cours. Au départ, l'étudiant devait valider ses connaissances concernant un concept particulier avant d'avoir accès au concept suivant. Etant donné l'élargissement de la population cible, cette voie de recherche a été abandonnée au profit du développement d'une application moins stricte au niveau de la validation des connaissances. De plus, lorsque la population est limitée aux étudiants et que l'accès à une information plus précise n'est possible qu'à partir du moment où la connaissance du sujet a été prouvée, un menu n'est absolument pas nécessaire. Suite au changement de population cible, un menu est à présent indispensable.

La définition des objectifs consiste à expliciter les changements de comportement ou de performance provoquée chez les formés. Cette définition peut être obtenue en répondant à :

Qui produira le comportement souhaité ?
Quel comportement observable démontrera que les objectifs sont atteints ?

Lors de la définition originelle du produit, seuls les étudiants étaient amenés à utiliser l'application, suite à l'extension de la population visée, tout utilisateur du cours est amené à utiliser (du moins cela est souhaité) le module d'évaluation des connaissances. Ce module comprendra une série de questions concernant les concepts introduits. A chaque question sera attribué un indice de difficulté. La

somme de ces indices permettra de vérifier la compréhension globale du protocole Ethernet.

Des stratégies visant à susciter l'intérêt sont-elles prévues ?

Un des intérêt principaux de ce cours sur Ethernet est qu'il est possible d'avoir accès à une animation permettant de visualiser le mouvement des trames sur le réseau. Cette animation peut jouer ce rôle.

Quelles sont les activités solicitées chez l'apprenant ?

L'utilisateur est supposé visionner l'animation, être évalué et, dans la mesure du possible, faire quelques commentaires (soit généraux soit plus précis) au sujet du cours. L'évaluation va être basée sur la mémorisation par l'utilisateur des concepts présentés.

Le guidage est l'ensemble des moyens mis en oeuvre pour conseiller, informer et conduire l'utilisateur lors de ses interactions avec le système. Une idée plus précise peut être obtenue en répondant à :

La structure du contenu apparaît-elle à l'utilisateur ?

De par le choix de diffusion du produit sur Internet, le cours sera réalisé par l'intermédiaire de pages HTML. Ces dernières permettent de faire apparaître la structure du contenu assez facilement. Les mots correspondant à des liens vers d'autres pages HTML sont affichés dans une couleur différente des mots « normaux ». De plus, les mots « spéciaux [3]» sont des mots-clé donnant accès à des informations spécifiques sur le mot en question. Par exemple, en sélectionnant le mot PROTOCOLE, l'utilisateur a accès aux informations disponibles sur les protocoles. Cette façon de travailler tient en plus compte du critère de charge de travail; en effet, l'utilisateur n'est pas noyé sous une masse d'informations. Pour le guidage de l'utilisateur, l'écran va être divisée en deux parties. La partie supérieure contiendra les concepts développés par le cours, celle du bas aura pour but

[3] par opposition aux mots normaux.

d'indiquer à l'utilisateur où celui-ci se situe dans le graphe représentant la structure du cours[4].

Pour certaines parties du cours, l'apparition de boîtes de dialogue permet à l'utilisateur d'introduire, par exemple, les paramètres propres à l'animation. Cela signifie donc que les huit critères ergonomiques présentés au point 3.1 doivent être appliqués.

[4] ce graphe sera disponible dans le chapitre suivant lorsque la théorie sera détaillée.

Chapitre 3 : Un cours sur Ethernet.

1 Introduction.

Lors du choix de l'exploration des possibilités offertes par le langage Java, nous avons été rapidement amenés à considérer le développement d'un cours. Il restait cependant à définir quel serait le contenu de ce dernier. De façon assez surprenante, nous avons d'abord choisi le concept central du travail avant de savoir quels étaient les concepts qui allaient graviter autour de ce centre. Notre choix s'est porté sur les collisions pouvant survenir sur les réseaux locaux régis par le protocole Ethernet.

Une fois ce choix effectué, il restait à développer l'applet Java destinée à simuler le mouvement des trames et leurs collisions sur un réseau local ainsi qu'à élaborer des scénarios permettant à l'utilisateur d'accéder à cette animation.

Dans ce chapitre, une description plus précise des différentes options du cours va être proposée. Il sera divisé en cinq parties, chacune ayant pour but de développer un aspect existant du cours : le menu, la théorie et l'animation. Un bref rappel concernant le protocole Ethernet va d'abord être proposé. Il sera suivit par un rappel concernant les générateurs de nombres aléatoires.

2 Les réseaux Ethernet.

Pour ce rappel, les notions développées seront extraites du cours de seconde licence « Téléinformatique et Réseaux » [VANBAST 93] du professeur Philippe van Bastelaer.

Un des protocoles les plus fréquents sur les réseaux locaux est le protocole défini par Xérox et basé sur le protocole ALOHA[1]. Le protocole a ensuite été mis au point par Xérox, Intel et DEC sous le nom de Ethernet avant d'être adopté par le comité IEEE 802.3 après de nombreuses modifications. Strictement, il y a donc des différences entre le protocole 802.3 et le protocole Ethernet qui n'est pas prévu tel quel pour s'insérer dans l'architecture définie par IEEE. La méthode d'accès employée est néanmoins identique dans les deux protocoles.

Ces deux protocoles utilisent la méthode d'accès CSMA/CD (Carrier Sense Multiple Access with Collision Detection) et fonctionnent actuellement à 10 Mbps. Ils sont conçus initialement pour des réseaux à topologie de bus sur support coaxial. Deux versions de support coaxial sont prévues : la version sur câble coaxial fin (Thin Ethernet) de 0.25 pouce de diamètre porte le nom de 10 Base 2 car la longueur de segment de câble maximale autorisée est égale à 200 mètres alors que la seconde version sur câble coaxial épais (Thick Ethernet) de 0.5 pouce de diamètre porte le nom de 10 Base 5 puisque la longueur de segment de câble maximale autorisée est dans ce cas égale à 500 mètres. Ethernet fonctionne également sur des topologies bus logique-étoile physique; c'est le cas des versions 10 Base T (sur paire torsadée non blindée et limitée à une longueur maximale de 100 mètres) et 10 Base F (sur fibre optique). Afin d 'étendre la taille des réseaux Ethernet jusqu'à une longueur de 2.5 km, l'utilisation de répéteurs permet de relier logiquement plusieurs segments en un seul réseau.

La méthode d'accès CSMA/CD est basée sur la technique de la contention avec détection de collision. Le principe est le suivant : lorsqu'une trame est prête à être émise, le niveau MAC (Medium Access Control) attend que le niveau physique lui indique que la ligne est libre (ce que l'on appelle *absence de porteuse*); lorsque celle-ci est libre, la trame est passée au niveau physique pour sa transmission, c'est pourquoi on appelle cette technique Carrier Sense Multiple Access. Comme il est possible que plusieurs stations attendent la libération du support, plusieurs trames peuvent être émises simultanément sur le réseau par

[1] ALOHA a été développé à l'université de Hawaii pour les réseaux radiodiffusés.

plusieurs stations, cela résulte en une déformation des signaux émis et donc en une collision. Lors de la détection d'une collision par un niveau physique, celui-ci cesse immédiatement son émission et il émet alors un message spécifique de collision pour s'assurer que toutes les stations sont bien au courant de la collision. Cela s'appelle la « Collision Detection » (CD). Lors de la collision, toutes les trames déformées ne sont pas reçues.

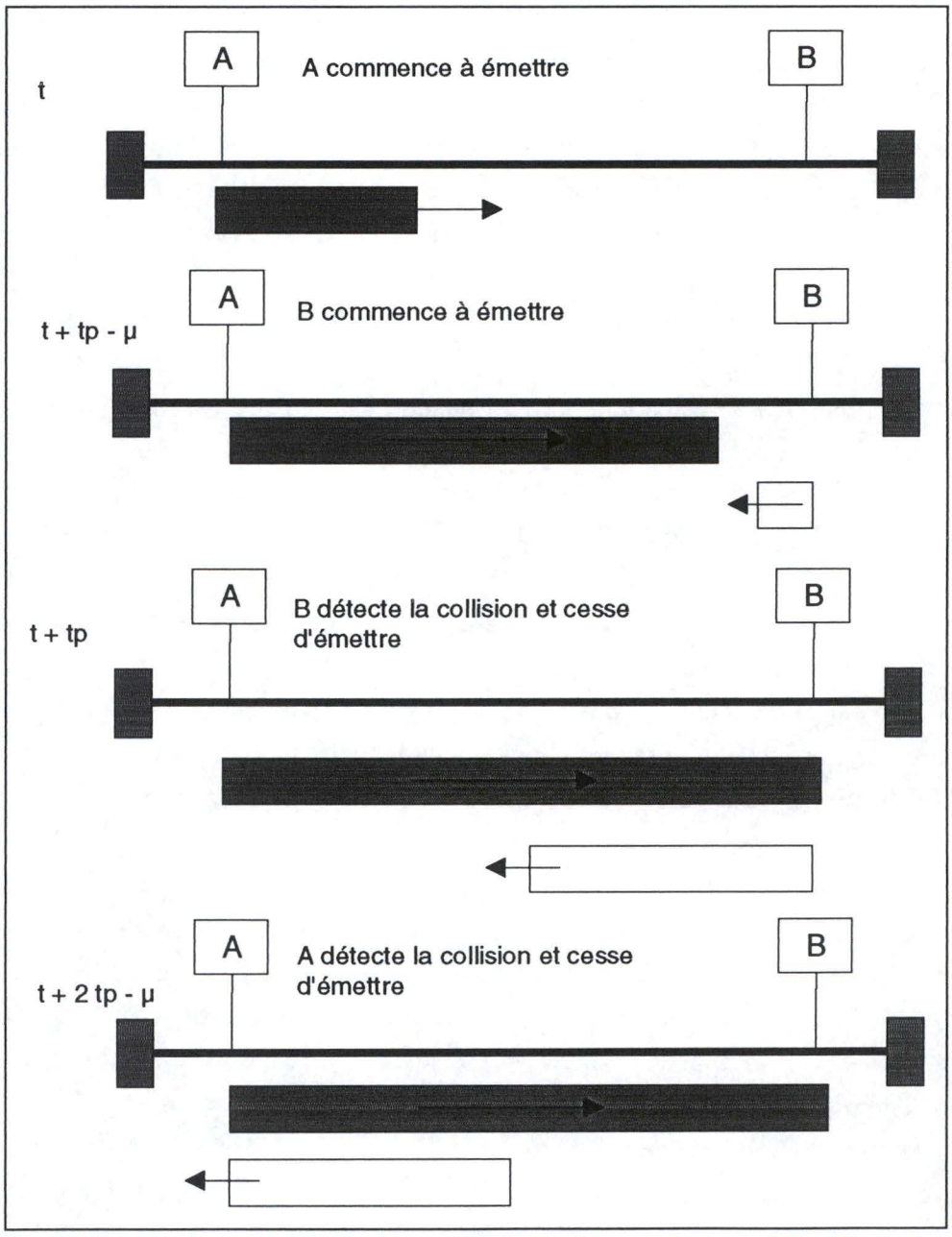

Figure 3.1 : la détection d'une collision

Après une collision, chaque station émettrice tente, après un temps d'attente aléatoire, d'émettre à nouveau sa trame. Le processus se répète selon les règles précisées plus haut. Après 16 tentatives, la station prévient les couches supérieures de son échec et abandonne sa tentative d'émission. Si une trame a été émise normalement (c'est-à-dire sans qu'une collision n'ait été détectée) et n'arrive pas à destination (généralement pour des problèmes liés à des défectuosités du support physique), on dit que la trame est perdue.

Il est évident que la technique CSMA/CD décrite plus haut implique que les émetteurs doivent absolument détecter une collision avant la fin de leur émission. Cela implique que le temps nécessaire à l'émission d'une trame soit plus long que le plus long temps nécessaire à la détection d'une collision appelé « slot-time ».

Or, comme on peut l'observer dans la figure 3.1, la pire des configurations est celle où les deux stations responsables de la collision sont situées aux deux extrémités d'un réseau de taille maximale et dans le pire des cas qui est celui où une station (B) commence à émettre juste avant de détecter que l'autre station (A) était occupée d'émettre, le temps nécessaire à la station A pour détecter la collision est égal au double du temps de propagation *tp* d'un bout à l'autre du réseau.

Donc, pour une taille maximale de 5 kilomètres, le slot-time est de 50 µsec; ce qui au rythme de 10 Mbps, correspond à l'émission d'une trame de 500 bits. C'est pourquoi la taille minimale normalisée d'une trame est de 512 bits.

3 Les générateurs de nombres pseudo-aléatoires.

Pour ce rappel, les notions développées seront extraites du cours de seconde licence « Simulation de Systèmes » [NOIR 93] du professeur M. Noirhomme-Fraiture.

Les générateurs de nombres pseudo-aléatoires ont la forme générale suivante :

$$x_{n+1} = a\, x_n + c \pmod m$$

où a, c et m sont des entiers positifs

Ces générateurs doivent vérifier deux propriétés essentielles qui garantissent leur efficacité. Il s'agit de l'uniformité de la distribution des nombres générés et de la non corrélation de ces nombres. Des tests peuvent être réalisés afin de vérifier ces deux propriétés. Il s'agit du test de χ^2 pour la distribution uniforme des valeurs au sein de classes de taille identique. Pour ce faire, la distance au carré séparant les valeurs observées des valeurs théoriques est calculée pour chaque classe :

$$D^2 = \sum_{i=1}^{N} (Th_i - T_i)^2 / Th_i$$

où Th_i est la valeur théorique de la taille de la classe i
T_i est la valeur observée de la taille de la classe i
N est le nombre de classe

Le test consiste à comparer ce D^2 avec le quantille de χ^2 à N - 1 degré de liberté. Si la valeur de D^2 obtenue est supérieure au quantille correspondant, le test d'uniformité rejette la série.

Le test de corrélation sérielle consiste à vérifier que les séries de nombres ne sont pas corrélées entre elles. Cela peut-être réalisé de la manière suivante : il suffit de calculer, pour chaque série et pour chaque déphasage entre les nombres générés, le coefficient de corrélation sérielle. L'estimateur est :

$$r_k = \frac{\frac{1}{n-k}\sum_{i=1}^{n-k}(z_i - \bar{z})(z_{i+k} - \bar{z})}{\frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2}$$

où

n est le nombre d'éléments de la série
z_i est le $i^{ème}$ élément de la série
k est le déphasage

$\bar{z}$ est la moyenne arithmétique des éléments de la série.

et la variable :

$$t_k = r_k \sqrt{\frac{n-2}{1-r_k^2}}$$

En se basant sur l'hypothèse qu'il n'y a pas de corrélation sérielle (c'est-à-dire $r_k = 0$), alors la variable t_k a une distribution de Student à N-k-2 degrés de liberté. La région critique (c'est-à-dire la zone de rejet) peut être déduite de :

$$\left\{\omega : |t_k| > Q_{t_{N-k-2}}\left(1-\frac{\alpha}{2}\right)\right\}$$

Pour connaître les valeurs du quantile, il suffit de consulter les tables de Student avec deux entrées (1-α/2 et N-k-2 où α représente la probabilité d'erreur).

Lors de simulations, on est souvent amené à utiliser des variables aléatoires ayant une loi bien déterminée (Erlang(k , λ), Normale(0 , 1), etc.) alors que les générateurs de nombres pseudo-aléatoires génèrent des nombres distribués uniformément sur l'intervalle [0,1]. En pratique, tous les types de variables aléatoires peuvent être obtenus à partir des variables aléatoires uniformes.

Par exemple, l'animation présentée dans la suite de ce chapitre montre l'envoi, la réception et la collision de trames (sur un réseau local régi par le protocole Ethernet) dont les moments d'émissions sont tirés aléatoirement suivant une distribution exponentielle. Or, le générateur de nombres aléatoires fourni dans les librairies du langage Java a une distribution uniforme. Pour obtenir une variable aléatoire de distribution exponentielle à partir d'une variable aléatoire de distribution uniforme, on utilise la méthode dite de la transformation inverse. Cette méthode est la suivante :

Soient $r \in Unif(0,1)$ et $F(a)$ la fonction de répartition de la variable aléatoire souhaitée

Alors $x = F^{-1}(r)$ a la distribution $F(.)$ recherchée[2].

Par exemple, la variable aléatoire exponentielle dont la fonction de répartition est $F(a) = 1 - e^{-\lambda a}$ vaut $x = (-1/\lambda) \ln(1 - r)$.

4 Le menu.

Au départ, notre choix s'était posé sur un cours où l'utilisateur n'avait accès à une information qu'à partir du moment où la preuve de sa maîtrise des connaissances utiles à la compréhension de cette nouvelle information était faite. Cependant, cette orientation a été abandonnée assez rapidement pour permettre une plus grande diffusion du logiciel (il peut de cette façon être accessible aussi bien à des experts du protocole Ethernet qu'à des novices). Un menu s'est donc avéré nécessaire.

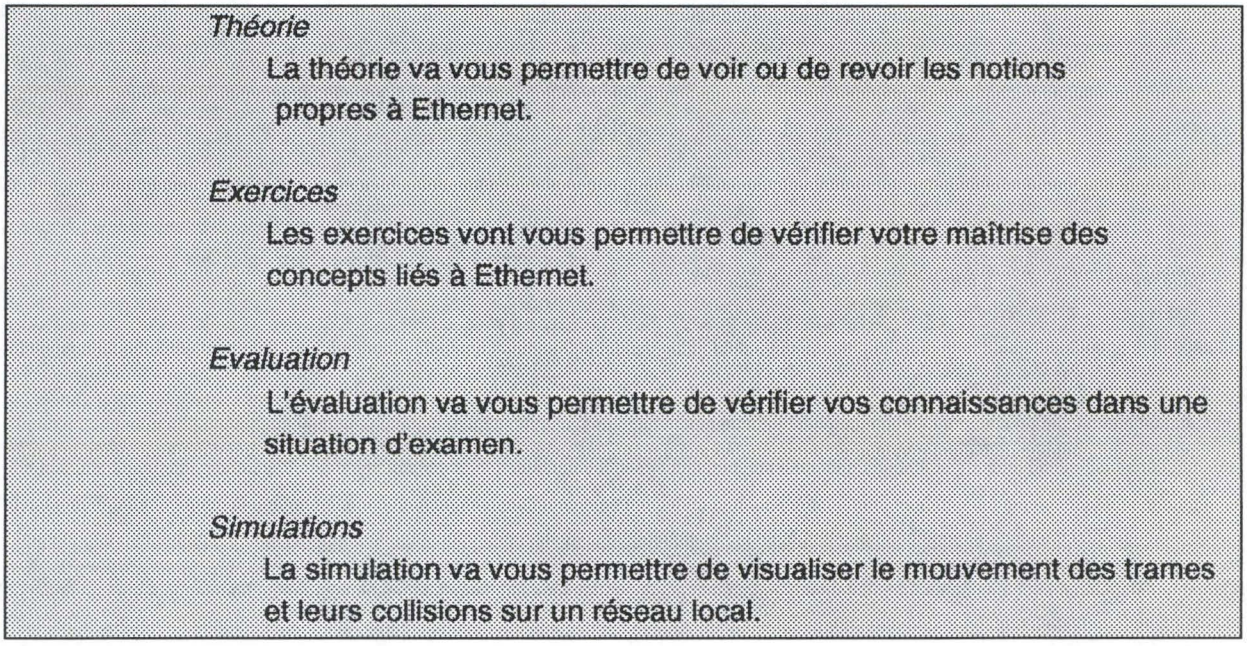

Figure 3.2: Le menu sous forme de texte.

Ce menu se présente sous deux formes. La figure 3.2 nous montre le menu sous forme de texte : chacun des items de ce menu (théorie, simulation, exercices et évaluation) sont des

[2] en effet $\Pr(x \leq a) = \Pr(F^{-1}(r) \leq a) = \Pr(r \leq F(a)) = F(a)$

liens vers les pages HTML correspondant a l'item choisi. La figure 3.3 nous montre ce menu sous une forme plus visuelle : il s'agit de l'utilisation d'une image réactive. Ces images ont l'avantage de permettre d'aiguiller l'utilisateur vers différentes destinations en fonction de la partie de l'image sélectionnée par ce dernier.

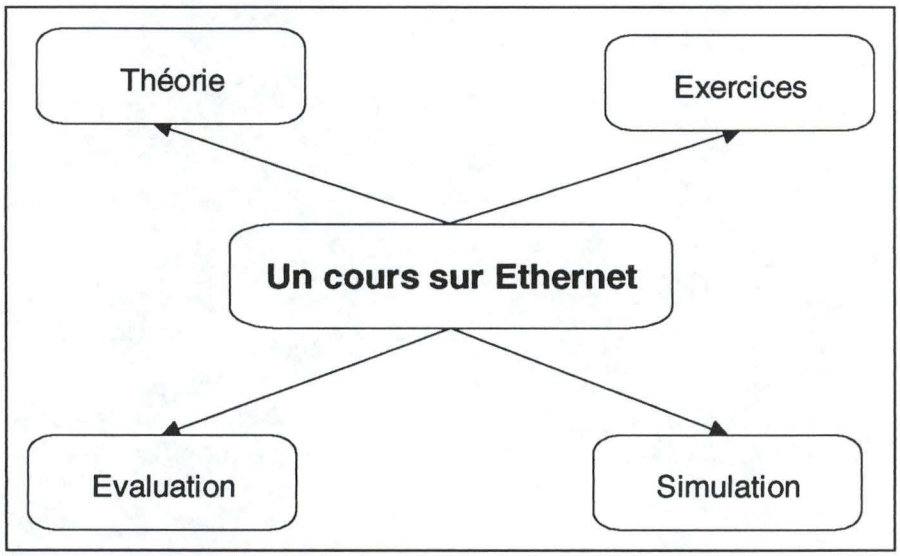

Figure 3.3 : Le menu sous forme visuelle

Ces images réactives peuvent être réalisées en utilisant soit des applets java soit un éditeur particulier. L'applet Java qui peut être utilisée est l'applet *ImageMap* disponible à l'adresse http://www.javasoft.com/people/flar ([HOF 96]). Des essais ont été effectués en vue d'utiliser cette applet, de toute évidence, elle ne fonctionne pas parfaitement. La seconde possibilité s'est donc avérée utile. Il s'agit du logiciel MapEdit. Il permet de définir les zones sur l'image de départ, à partir desquelles des liens vont être réalisés vers d'autres pages HTML.

5 La partie théorie.

Après avoir choisi l'animation à développer, la définition d'un contexte de travail s'imposait. Cela signifie qu'il fallait définir les concepts minimaux à enseigner à l'utilisateur afin qu'il puisse comprendre le fonctionnement du réseau Ethernet et le comportement des trames sur le réseau.

De par le critère de la charge de travail[3], les concepts enseignés vont des considérations plus générales aux considérations plus précises. Cela peut se traduire par un graphe d'enchaînement des concepts à traiter.

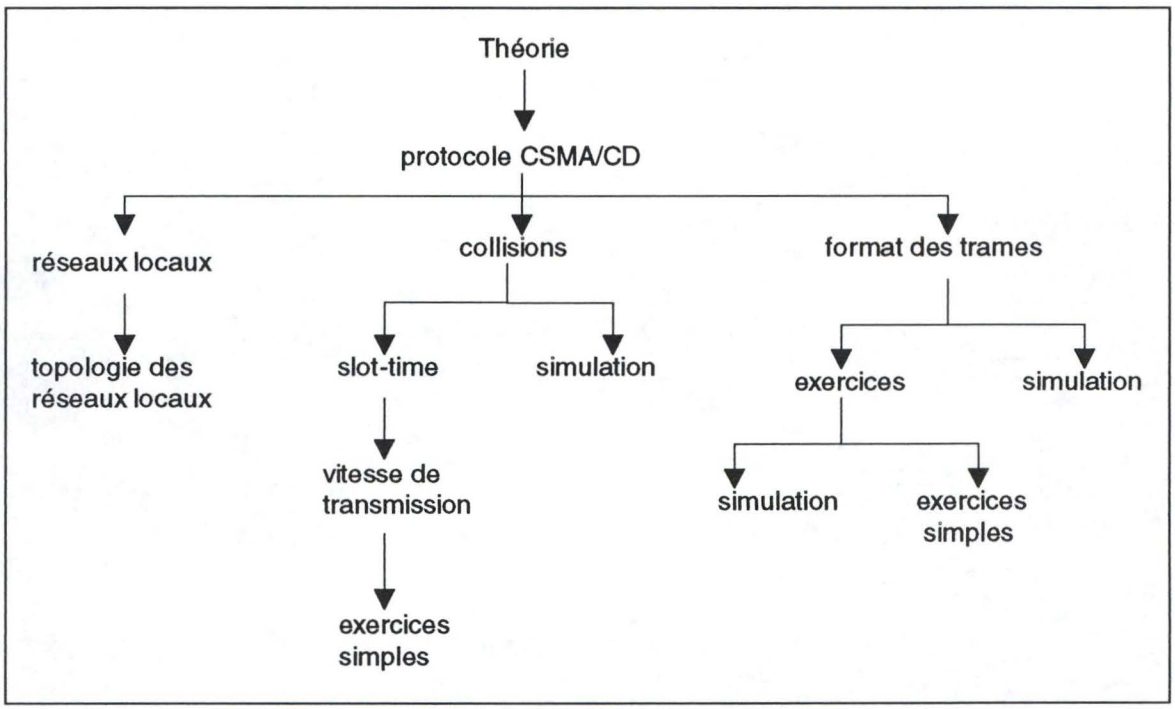

Figure 3.4 : graphes d'enchaînement des concepts à développer.

Chaque noeud du graphe représenté à la figure 3.4 décrit un aspect de la théorie à traiter. Par exemple, l'aspect *collisions* a pour but d'expliquer à l'utilisateur ce qu'est une collision ainsi que les événements à l'origine d'une collision. Chaque fils d'un noeud père représente les concepts accessibles à partir de la page HTML décrivant les concepts relatifs au noeud père. Par exemple, les concepts de *réseaux locaux*, *collisions* et *format des trames* sont accessibles à partir de la page HTML décrivant ce qu'est le *protocole CSMA/CD*.

Pratiquement, les nouveaux concepts accessibles à partir du noeud père seront reconnaissables sur les pages HTML parce qu'ils seront affichés dans un format particulier :

[3] cfr. les critères ergonomiques du chapitre 2.

chaque mot indiquant à l'utilisateur un nouveau concept pouvant être développé sera affiché dans une couleur différente de celle des autres mots.

6 La partie animation.

Le centre de notre travail consiste en la création d'une animation permettant à l'apprenant de visualiser le comportement des trames sur le réseau (lorsque celui-ci est régi par le protocole Ethernet).

L'animation décrite ci-dessous peut être utilisée dans des contextes forts différents : la théorie, les exercices, l'évaluation ou l'animation proprement dite. Dans la théorie, elle peut servir de support visuel à la compréhension du protocole Ethernet. Pour l'évaluation, certains paramètres de l'animation peuvent être modifiés afin de produire un comportement anormal[4] des trames. Cela peut permettre de vérifier le degré de compréhension du protocole par l'utilisateur.

Cette animation peut être divisée en trois parties distinctes mais non tout à fait indépendantes. La première s'occupe de l'introduction des paramètres propres à l'animation par l'utilisateur. La seconde va créer une liste d'événements qui se présenteront lors de l'animation. Il s'agit de l'envoi des trames, de leur réception ou de leur collision. Les paramètres introduits dans la première partie vont influencer l'initialisation de cette liste. La dernière partie consiste en l'animation proprement dite, c'est-à-dire le mouvement des trames sur le réseau : celle-ci est une simple "traduction visuelle" de la liste d'événements.

6.1 Les paramètres de l'animation.

Pour permettre à l'utilisateur de se faire une idée la plus proche possible de la réalité, il faut développer un modèle régissant l'envoi et l'arrivée des trames sur le réseau ainsi que les collisions pouvant s'y produire.

[4] c'est-à-dire non conforme au comportement prévu par la norme IEEE.

Etant donné que l'animation peut être utilisée dans des contextes fort différents, il nous faut la paramétriser au maximum afin de pouvoir la réutiliser différemment entre deux affichages successifs. De plus cette paramétrisation joue un rôle important sur le plan de la motivation (de l'intérêt) de l'utilisateur. En effet, plus ce dernier a la possibilité de jouer avec ces paramètres, plus son sentiment de maîtrise de l'animation est grand.

Les paramètres qui peuvent être modifiés par l'utilisateur sont les suivants :

- *NbrSta* : indique le nombre de stations reliées au réseau. Ce nombre est compris entre 2 et 6. Le choix de cette borne supérieure a été réalisé pour deux raisons : tout d'abord, il faut laisser à l'utilisateur un éventail assez large de possibilités et ensuite, il ne faut pas permettre un trop grand nombre de stations afin de ne pas surcharger l'animation.
- *LoiEm* : indique quelle est la loi (aléatoire ou de poisson) régissant l'émission des trames sur le réseau. Il faut remarquer que les lois proposées à l'utilisateur ne répondent à aucun critère réel. En effet, il n'est pas possible de prédire de façon certaine la loi qui va être suivie par cette émission.
- *TempsAtt* : indique le temps d'attente avant la ré-émission d'une trame suite à la détection d'une collision. Normalement, ce temps est aléatoire mais on peut prévoir la possibilité de laisser à l'utilisateur le choix d'un autre mode d'émission. Cela peut également s'avérer utile dans le cadre des exercices ou de l'évaluation : en assignant un temps d'attente constant (par exemple), la ré-émission de ces trames produit chaque fois une nouvelle collision. Cela permet donc de mettre en évidence les raisons d'être de ce temps d'attente dans le protocole.
- *TempsMoy* : indique le temps moyen entre l'envoi de deux trames sur le réseau. Il est utilisé par la loi d'émission pour la génération aléatoire du moment d'émission de la trame suivante.
- *TailleReseau* : indique la taille occupée par le réseau. Puisque les réseaux locaux ont une taille limitée à 5 km, cette variable pourra prendre au plus cette valeur.

Parmi les paramètres cités ci-dessus, certains vont influencer la création de la liste des événements (il s'agit de *NbrSta*, *LoiEm*, *TempsAtt*, *TailleReseau et TempsMoy*) alors que d'autres influenceront l'affichage des trames (*NbrSta*).

L'introduction des paramètres de l'animation va se faire par l'intermédiaire d'une interface obtenue à partir d'une applet Java. Cette dernière va être découpée en deux méthodes regroupées dans une même classe : la première va s'occuper de l'affichage des différentes listes

de sélection alors que la seconde va traiter les actions effectuées par l'utilisateur (le déclenchement de fonctions par un bouton de commande, etc.). La figure 3.5 nous montre cette classe.

```
public class ParamSimul extends Frame {
        Choice TempsMoyTF, TailleReseauTF;
        Choice NbrStaTF,TypeResTF,LoiEmTF,TempsAttTF;

        public ParamSimul()
                {
                }
        public boolean handleEvent(Event evt)
                {
                }
```

Figure 3.5 : La classe *ParamSimul.*

Les variables globales de cette classe sont toutes de type *Choice*. Cela permet de créer des listes de sélection qui empêchent l'utilisateur d'introduire des paramètres non significatifs pour l'animation. Par exemple, l'introduction par l'utilisateur d'un temps moyen inter-émission de cinq minutes conduirait à une animation ayant peu d'événements de type collision.

6.2 La création de la liste des événements.

Un certain nombre de classes ont dû être développées afin d'aboutir à cette liste d'événements. Nous allons nous attarder quelque peu sur la description des classes ainsi que des méthodes associées à ces dernières. Cependant nous allons commencer par examiner les types d'événements existant et les hypothèses posées pour l'élaboration de cette liste d'événements.

6.2.1 Description des événements.

L'animation des trames sur le réseau va être implémentée en utilisant une liste d'événements. Ces derniers correspondent aux différentes situations qui peuvent se produire sur le réseau.

Trois types d'événements ont été mis en évidence. Il s'agit des types :

- EMISSION : il correspond à l'envoi d'une trame par une station quelconque située sur le réseau.
- RECEPTION : il correspond à la fin de la réception d'une trame par la station à laquelle cette trame est destinée.
- COLLISION : cela correspond à la collision de deux trames sur le réseau.

La liste d'événements est construite de la manière suivante : lors de chaque envoi d'une trame, la liste des événements est mise à jour en fonction du moment d'émission de cette trame et en fonction des autres trames encore situées sur le réseau. Par exemple, à un certain moment une trame doit être envoyée par la station numéro 3 à destination de la station 6. S'il n'y a pas d'autre trame sur le réseau, un événement de type EMISSION va être généré au moment d'émission de cette trame. Par contre, si d'autres trames se trouvent sur le réseau, il faut s'assurer que le trame peut être émise[5] et il faut également vérifier si des collisions peuvent se produire suite à cette émission.

6.2.2 Les hypothèses concernant cette liste d'événements.

Afin de pouvoir développer cette liste d'événements, plusieurs hypothèses ont dû être posées. Elles permettent la simplification du modèle sous-jacent à la création de cette liste. Ces suppositions sont les suivantes :

- Chaque station située sur le réseau est identifiée par un entier.
- Les stations sont numérotées de 1 à 6 en fonction des paramètres introduits par l'utilisateur.
- Les nombres qui correspondent aux numéros des stations sont rangés par ordre croissant sur le réseau.
- Les stations sont réparties uniformément sur le réseau. Cela signifie que la distance séparant deux stations consécutives est constante.
- Deux trames ne peuvent jamais être émises exactement au même moment.

[5] en termes utilisés par le protocole Ethernet, il faut vérifier l'absence de porteuse.

- Les paramètres introduits par l'utilisateur ne peuvent être modifiés en cours d'animation. Cela signifie que si l'utilisateur a choisi une animation comportant cinq stations, les cinq stations auront toujours la possibilité d'émettre ou de recevoir des trames au cours de cette animation. Aucun cas n'est prévu pour leur permettre de tomber en panne.
- Une trame émise sur le réseau n'est jamais perdue (le support est donc supposé non défectueux).

La figure 3.6 montre, dans le cas où le réseau est composé de trois stations, comment les stations sont disposées dans la fenêtre contenant l'animation.

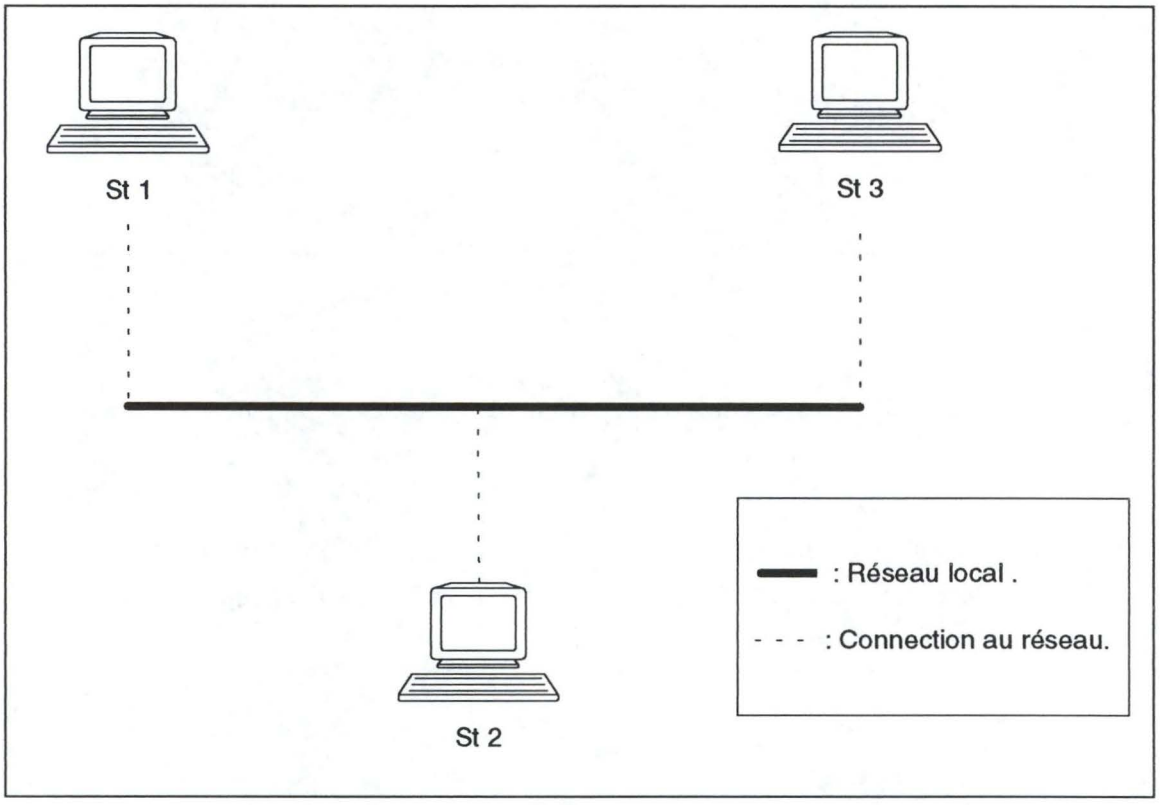

Figure 3.6 : un exemple de disposition des stations

6.2.3 La classe « Trame ».

Puisque le concept de base de l'animation est la trame, une classe correspondante a été utilisée. Voici la définition de cette classe ainsi que les méthodes en faisant partie :

```
class Trame {
        protected int NumStatEm;
        protected int NumStatDest;
        protected long MomEmiss;

        Trame next;

        public Trame(int NbrS, String Loi, long Temps, Random Gen)
                {
                }
        public  int GenerNumStat (int NbrS, Random Gen)
                {
                }
        public void GenerTempsInterEmiss(String LoiEmiss, long TempsMoyen,
                                Random Gen)
                {
                }
```

Figure 3.7 : La classe *Trame.*

Les trames sont caractérisées par certains paramètres qui permettent de compléter les informations à propos des événements en relation avec ces trames :

- *NumStatEm* : ce paramètre correspond au numéro de la station émettrice de la trame;
- *NumStatDest* : ce paramètre indique le numéro de la station à laquelle est destinée la trame émise.

Ces deux paramètres sont tirés aléatoirement suivant une distribution uniforme et ont une valeur limite supérieure correspondant au nombre total de stations reliées au réseau[6]. Il faut cependant s'assurer, lors du tirage de ces nombres, que ces deux paramètres sont différents. En effet, il n'est pas concevable qu'une station désire s'envoyer un message à elle-même.

- *MomEmiss* : ce paramètre indique à quel moment l'émission de la trame va être effectuée. Il peut être généré en suivant soit une distribution uniforme, soit une distribution de Poisson.

[6] il correspond au paramètre *NbrS* du constructeur de la classe.

De plus, les nombres générés sont strictement positifs, ce qui implique que deux trames ne peuvent jamais être émises exactement au même moment.

Les méthodes utilisées dans cette classe sont :

Méthodes	Descriptions
public Trame(int NbrS, String Loi, long Temps, Random Gen)	il s'agit du constructeur de la méthode. Il a pour fonction d'initialiser les variables présentées ci-dessus en fonction de paramètres propres à l'animation. *NbrS* indique le nombre total de stations situées sur le réseau. *Loi* indique la loi devant être utilisée pour l'initialisation de la variable *MomEmiss*. Quant à la variable *Temps*, elle indique quel est le temps moyen séparant l'émission de deux trames successives.
public int GenerNumStat(int NbrS, Random Gen)	il s'agit de la méthode permettant de tirer aléatoirement le numéro d'une station, en fonction du nombre total de station sur le réseau *NbrS*. Elle s'assure également que les deux numéros de station tirés ne sont pas identiques.
public void GenerTempsInterEmiss(String LoiEmiss, long TempsMoyen, Random Gen)	cette méthode initialise la variable *MomEmiss* en fonction de la loi d'émission des trames *LoiEmiss* et du temps moyen séparant l'émission de deux trames *TempsMoyen*.

Nous pouvons remarquer que le paramètre *Gen*, figurant comme argument dans les trois méthodes ci-dessus n'a pas été décrit. Il indique le générateur de nombre pseudo-aléatoires utilisé.

La raison pour laquelle la variable *Gen* est passée comme paramètre lors de l'appel de fonction est la suivante : la variable *Gen* est initialisée par le constructeur de la classe Math. Lors de chaque appel à ce constructeur les premiers nombres générés sont souvent identiques. Pour éviter d'avoir des animations se déroulant dans la plupart des cas de la même manière, il

suffit de passer *Gen* comme paramètre puisque les nombres générés par une même initialisation de cette variable sont de distribution uniforme et non corrélés entre eux.

6.2.4 La classe « ListOfTrame ».

La classe suivante est celle permettant d'implémenter les listes de trames. Ces listes vont être utilisées afin de garder une trace des trames se trouvant encore sur le réseau lors de l'émission d'une nouvelle trame par une station particulière. Les variables et les méthodes de la classe *ListOfTrame* sont décrites à la figure 3.8.

Les variables *premier*, *dernier* indiquent respectivement la première et dernière trame située dans la liste. La variable *courant* est utilisée pour parcourir la liste.

```
class ListOfTrame {

        protected Trame premier, dernier, courant;

        public ListOfTrame ()
                {
                }
        public void Ajout (Trame UneTrame)
                {
                }
        public void Retrait (long MomentEmission)
                {
                }
```

Figure 3.8 : La classe *ListOfTrame*.

Les listes de trames utilisent les méthodes suivantes :

Méthodes	Descriptions
public ListOfTrame()	il s'agit du constructeur de cette classe. Elle a pour but d'initialiser les variables *premier* (le premier élément de la liste), *dernier* (le dernier élément de la liste) et *courant* (une variable qui permet de parcourir cette liste).
public void Ajout(Trame UneTrame)	cette méthode permet d'ajouter la trame *UneTrame* à la fin de la liste de trames utilisées. En ajoutant la trame à la fin de la liste, on est certain que les éléments de cette liste sont rangés par ordre croissant (puisque les temps générés sont strictement positifs).
public void Retrait(long MomentEmission)	cette méthode supprime de la liste de trames la trame dont le moment d'émission est *MomentEmission*. De plus, une seule trame de cette liste sera supprimée. En effet, les hypothèses indiquent que deux trames ne sont jamais émises au même moment.

6.2.5 La classe « Evenement ».

Etant donné que l'animation va utiliser une liste d'événements pour simuler l'envoi, la réception et la collision de trames sur un réseau local, une classe *Evenement* a dû être implémentée. La figure 3.9 nous décrit cette classe.

Les variables propres à cette classe sont les suivantes :

- *TypeEvt* : cette variable indique quel est le type d'événement correspondant à l'événement courant. Ce type peut être soit une émission (EMISSION), une fin d'émission (RECEPTION) ou une collision (COLLISION).
- *MomSurvEvt* : elle indique à quel moment a lieu l'événement.
- *StatEmission* : cette variable peut avoir différentes significations suivant le type d'événement. S'il s'agit de l'émission ou de la fin d'émission (RECEPTION) d'une trame, cette variable indique la station émettrice de la trame, s'il s'agit d'une collision (

COLLISION), elle indique une des deux stations émettrices des trames à l'origine de la collision.

- *StatDestination* : Tout comme la variable précédente, elle peut indiquer soit la station de destination d'une trame (s'il s'agit d'un événement de type EMISSION ou RECEPTION) ou de la seconde station à l'origine d'une collision (s'il s'agit d'un événement de type COLLISION).

```
class Evenement {

        protected String TypeEvt;
        protected long MomSurvEvt;
        protected int StatEmission;
        protected int StatDestination;
        protected Evenement suivant;

        public Evenement()
                {
                }

        public Evenement(Evenement UnEv)
                {
                }
```

Figure 3.9 : La classe *Evenement.*

La classe ci-dessus ne contient que deux constructeurs qui ont pour but d'initialiser un événement en fonction des arguments passés au constructeur.

6.2.6 La classe « ListOfEvenement ».

La classe *ListOfEvenement* va permettre de créer une liste d'événements correspondant à l'émission, à la réception ou à la collision de trames sur le réseau. Cette classe est décrite à la figure 3.10.

Les variables propres à la classe ListOfEvenement sont les suivantes :

- *first*, *last* et *current* : elles indiquent respectivement les premier, dernier et événements courant situés dans la liste d'événements.
- *MomDernSurvEvt* : elle indique le moment auquel s'est produit le dernier événement situé dans la liste. Cela permet de garder une liste triée par ordre croissant sur la valeur du moment de survenance d'un événement. Ces événements peuvent être des émissions, des réceptions ou des collisions de trames.
- *MomDernEmiss* : elle indique le moment auquel s'est produit l'émission de la dernière trame émise par une station. C'est en partie grâce à cette variable que *MomDernSurvEvt* est tenue à jour[7].

```
class ListOfEvenement {

protected Evenement first, last, current;
protected long MomDernSurvEvt;
protected long MomDernEmiss;

public ListOfEvenement ()
        {}

public void AjoutEvenement (Evenement UnEvenement)
        {}

public String Appart(Trame Ancienne, Trame Nouvelle)
        {}

public void MAJEvenement()
        {}

public void Creation()
        {}
```

Figure 3.10 : La classe *ListOfEvenement*.

Les différentes méthodes ayant pour sujet les listes d'événements sont :

[7]En effet lors d'une collision, il faut avoir recours à d'autres méthodes pour cette mise à jour.

Méthodes	Descriptions
public ListOfEvenement()	il s'agit du constructeur de la classe. Elle va initialiser les variables décrites ci-dessus.
public void AjoutEvenement (Evenement UnEvenement)	cette méthode a pour but d'ajouter à la fin de la liste d'événements, l'événement *UnEvenement*.
public String Appart(Trame Ancienne, Trame Nouvelle)	cette méthode a pour but de renvoyer une chaîne de caractères indiquant si au moins une des stations émettrice ou destinatrice de la trame *Nouvelle* est comprise dans l'intervalle déterminé par les deux stations de la trame *Ancienne*.
public void Creation()	cette méthode a pour but de créer la liste d'événements. C'est au niveau de cette méthode que les appels aux méthodes des autres classes vont être réalisés. Elle s'occupe de l'initialisation et de la mise à jour des liste d'événements et de trames, de la création des trames.
public void MAJEvenement()	cette méthode a pour but de mettre à jour la liste des événements lorsqu'une trame peut être émise sur le réseau. Cette mise à jour s'effectue de la manière suivant : pour chaque trame *Tr* se trouvant encore sur le réseau (c'est-à-dire pour chaque trame située dans la liste de trames) on vérifie que le moment d'émission de la nouvelle trame à émettre est supérieur au moment d'arrivée de la fin de la trame *Tr* à sa station de destination. Dans l'affirmative, un nouvel événement RECEPTION est généré. Dans le cas contraire, on examine les trames suivantes.

Supposons par exemple que les trames *Nouvelle Trame* et *Ancienne Trame* soient définies par :

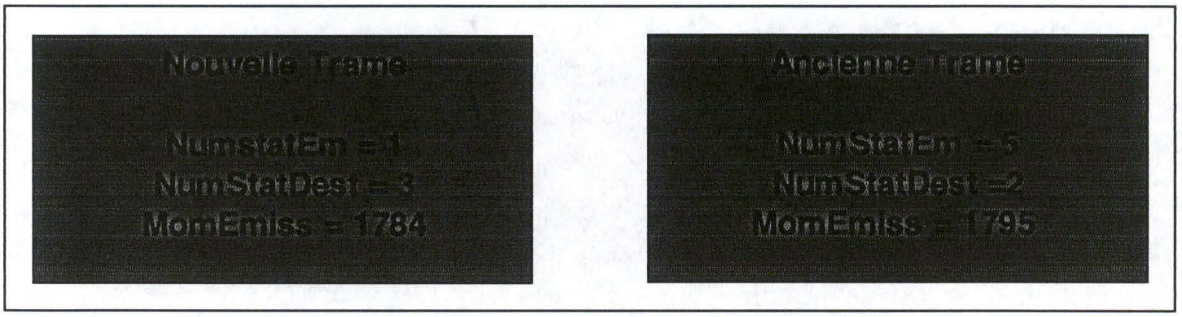

Figure 3.11 : Un exemple d'utilisation de la méthode *Appart*.

Le résultat renvoyé par Appart(*AncienneTrame*, *NouvelleTrame*) sera la chaîne de caractère « Appartient » puisque la station numéro 3 appartient à l'intervalle [2,5].

6.3 L'animation.

A partir du moment où la liste des événements est initialisée, il est possible de créer l'animation correspondant à cette liste.

Deux façons de procéder peuvent être envisagées pour développer cette animation. Il est possible d'utiliser des applets, développées par les concepteurs de Java, qui permettent d'afficher une suite d'images les unes à la suite des autres pour donner une impression d'animation. Cela peut être réalisé avec l'applet ImageLoop[8].

Cette première voie de travail s'avère assez difficile à utiliser dans le cadre du cours sur Ethernet. En effet, étant donné que l'utilisateur a la possibilité de modifier un grand nombre de paramètres (dont le plus important au niveau de l'animation est le nombre de stations), il faudrait prévoir un nombre assez grand d'images pouvant couvrir toutes les situations réalisables. Par exemple, supposons que le réseau comporte quatre stations, il faut prévoir toutes les possibilités d'émissions et de collisions des trames sur ce réseau particulier. Lors d'une autre animation traitant cinq stations, tout est à refaire. De plus, la gestion de toutes ces images va s'avérer fort difficile à réaliser. Par conséquent, cette façon de procéder a été abandonnée.

[8] développée par James Gosling. Elle est disponible à l'adresse http://www.javasoft.com/people/jag/

La seconde consiste à développer une applet s'adaptant automatiquement aux paramètres de l'animation. Elle sera capable, par exemple, de dessiner le nombre de stations correspondant au nombre introduit par l'utilisateur.

Cependant, l'utilisation d'une liste d'événements tirés aléatoirement peut être fort inefficace. Cela peut se produire pour différentes raisons :

- il est possible que, suite aux différents tirages aléatoires, aucune collision ne se produise sur le réseau. Puisque le but de l'animation est de montrer à l'apprenant ces collisions, cela perd de son intérêt. Ces listes doivent donc être rejetées et remplacées par d'autres contenant au moins une collision.
- il n'est pas intéressant de voir de nombreuses trames circuler sur le réseau avant de se trouver face à une collision : la liste devra donc être allégée (c'est-à-dire ne contenir qu'au plus deux ou trois émissions de trames avant la première collision).
- pour éviter d'attendre un long moment entre l'émission de deux trames, il est souhaitable d'utiliser une animation en temps réel uniquement lorsqu'une trame est transmise sur le réseau. Puisque le temps de transfert d'une trame est de l'ordre de quelques millièmes de seconde, il faudra cependant « dilater » ce temps réel pour que l'utilisateur puisse profiter de l'animation. Lorsqu'aucune trame n'est transmise sur le réseau, des bonds dans le temps seront réalisés jusqu'à la prochaine émission.

6.3.1 Conventions de représentation du réseau.

Certaines conventions ont été adoptées pour la représentation du réseau. Il s'agit de :

- la représentation du réseau à l'écran sera identique quelque soit la valeur de la taille du réseau introduite par l'utilisateur. Cela procure une représentation du réseau moins chargée puisqu 'elle occupe un maximum de place à l'écran. Les paramètres *TailleReseau* = 500 mètres et *NbrSta* = 6 surchargeraient trop la représentation du réseau si la taille de celui-ci variait en fonction de la valeur du paramètre *TailleReseau*.
- chaque station aura sa propre couleur et chaque trame véhiculée sur le réseau aura la couleur de sa station émettrice. L'utilisateur prenant l'animation en cours pourra donc identifier la station émettrice d'une trame grâce à la couleur de cette dernière.

- lors d'une collision, les parties de trames qui sont déformées par une collision seront dessinées en noir.

De plus, le protocole Ethernet prévoit l'envoi d'un message spécifique par la station venant de détecter une collision. Ce message particulier n'est pas affiché par l'animation.

6.3.2 La classe « AnimTrame ».

La classe *AnimTrame* s'occupe de l'animation proprement dite. Elle dessine le réseau en fonction des paramètres introduits par l'utilisateur et s'occupe de la mise à jour du réseau lors de l'émission, de la réception ou de la collision de trames.

La comparaison de la classe *AnimTrame* avec celles définies précédemment permet de mettre en évidence une différence essentielle au niveau de la façon d'aborder la création de l'animation. Cette différence consiste en *implements Runnable* située à la fin de la définition de la classe. Cela signifie pratiquement que l'applet AnimTrame peut être exécutée dans son propre contexte. Le browser sur lequel est exécutée l'applet n'est donc pas pris en otage par cette dernière. L'utilisateur est donc libre de choisir une autre option sur le browser (par exemple), une autre page Web ou même de quitter l'application.

L'utilisation de contextes d'exécution différents est possible grâce à l'utilisation de variables de type *Thread*. La machine virtuelle Java[9] permet aux applications d'avoir plusieurs thread s'exécutant concurremment. Chaque thread possède une priorité; celles de priorité plus grande étant exécutées de façon préférentielle par la machine virtuelle Java, contrairement à celles de priorité plus basse.

Les variables utilisées par la classe *AnimTrame* dont on peut voir le contenu à la figure 3.12 sont :

- *TextFont* : cette variable permet d'initialiser la taille et le style des caractères utilisés pour l'affichage de certains paramètres de l'animation (les numéros de stations, le moment où se produisent les événements, etc.).
- *AnimTr* : cette variable permet de définir un contexte d'exécution pour l'animation.

[9] cfr. le chapitre 1

- *TrameColor[]* : ce tableau permet d'initialiser les couleurs que prendront les stations ainsi que les trames émises par ces dernières.

```
public class AnimTrame extends Applet implements Runnable {
        Font TextFont;
        Color TrameColor[];
        Thread AnimTr = null;
        public void init()
        { }
        public void start()
        { }
        public void stop()
        { }
        public void run()
        { }
        public void DrawReseau ( Graphics g )
        { }
        public void DrawStation( Graphics g, int x, int y, int NumerStat )
        { }
        public static void initValeur ( ListOfEvenement ListEv )
        { }
        public void paint()
        { }
        public void update()
        { }
}
```

Figure 3.12 : la classe *AnimTrame*.

Les méthodes utilisées par cette classe sont :

Méthodes	Descriptions
public void init()	cette méthode permet l'initialisation des variables de la classe AnimTrame ainsi que le passage de certains paramètres de la page HTML à destination de l'applet.

Méthodes	Descriptions
public void start()	cette méthode lance l'exécution de l'applet.
public void stop()	cette méthode arrête l'exécution de l'applet.
public void run()	cette méthode implémente le caractère *Runnable* de la classe *AnimTrame*.
public void DrawReseau(Graphics g)	cette méthode a pour objectif de dessiner le réseau correspondant au paramètre *NbrSta* introduit par l'utilisateur dans la classe ParamSimul.
public void DrawStation(Graphics g, int x, int y, int NumerStat)	cette méthode va dessiner la station de numéro *NumerStat* ainsi que sa connection au réseau. Cette connection se faisant à la coordonnée (x,y).
public void paint(Graphics g)	cette méthode a pour but de créer l'environnement de l'animation, c'est-à-dire le définition de la taille de la fenêtre dans laquelle cette animation aura lieu ainsi que la couleur de fond utilisée.
public void update(Graphics g)	elle met à jour le dessin du réseau en effaçant les anciennes positions des trames et en dessinant les trames à leurs nouvelles positions.

La figure 3.13 montre un instantané de l'animation où l'on voit la station numéro 3 émettre une trame à destination de la station numéro 1.

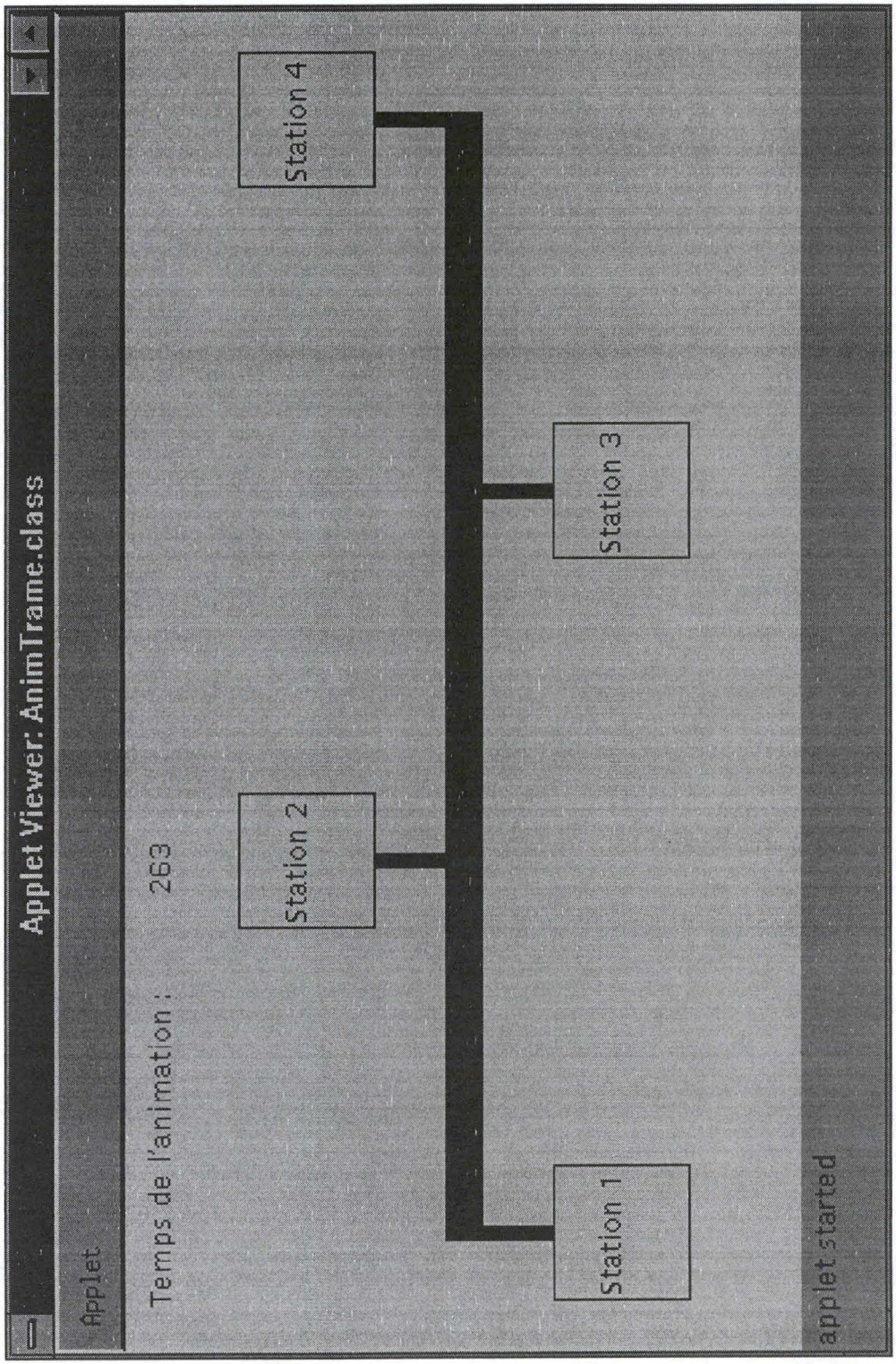

Figure 3.13 : un instantané de l'animation.

Chapitre 4 : Evaluation des potentialités de Java.

1 Introduction.

Ce chapitre a pour objectif de mettre en évidence les avantages et les inconvénients du langage Java qui ont été décelés lors du développement du cours sur Ethernet. Il va être divisé en deux parties : la première concernant le développement de l'animation et la seconde concernant certaines fonctionnalités ajoutées au cours (par exemple la possibilité d'ajouter des commentaires sur les pages HTML contenant les applets).

Les fenêtres permettant de visualiser les applets développées ont été obtenues par l'intermédiaire de l'application AppletViewer.

2 La partie animation.

La partie centrale du cours sur Ethernet concernait l'animation des trames sur le réseau. Cette animation est découpée en trois parties : la première concerne l'introduction des paramètres de l'animation par l'utilisateur, la deuxième s'occupe de la création de la liste des événements et la dernière, l'animation proprement dite.

2.1 Les paramètres de l'animation.

L'introduction des paramètres de cette animation est réalisée par l'intermédiaire d'une applet Java.

Un des avantages du langage Java est qu'il contient de nombreuses classes contenant des objets prédéfinis permettant de créer une interface assez rapidement. Cela peut aller de la création d'un menu déroulant à l'insertion de boutons de commande en passant par la saisie de caractères. Cinq classes contenues dans les librairies du langage Java permettent de créer des interfaces différentes. Il s'agit des classes :

- *FlowLayout* : elle affiche les composants de la fenêtre de la gauche vers la droite. Cette classe est souvent utilisée pour afficher des boutons de commandes, mais elle peut être utilisée dans d'autres situations. Il y a cependant moyen de choisir la façon dont les composants vont être positionnés dans la fenêtre : ils peuvent être soit alignés à gauche, à droite ou centrés.

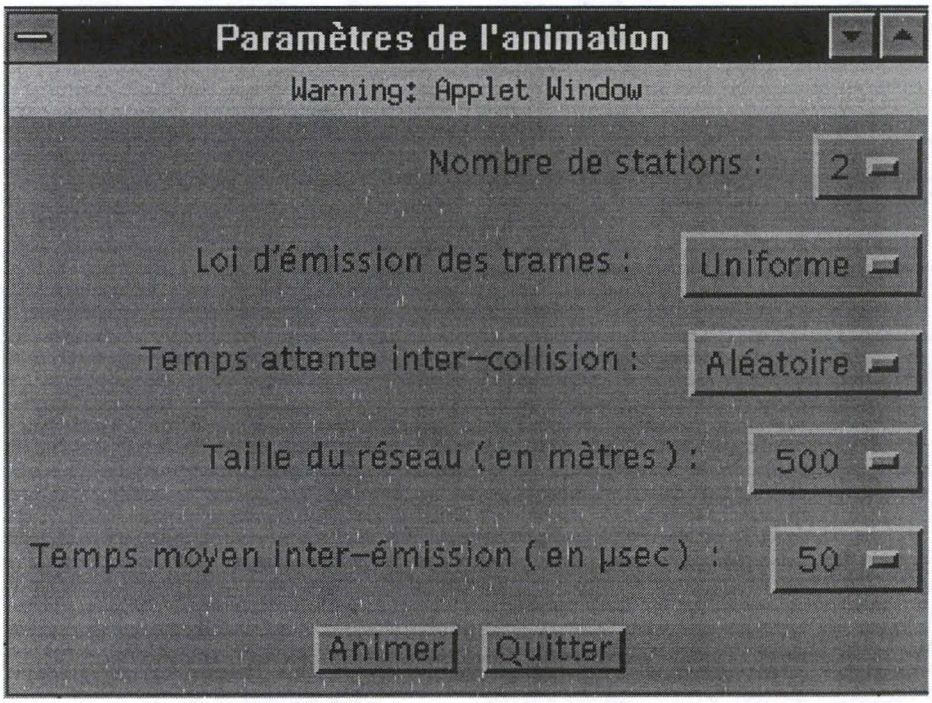

Figure 4.1 : un exemple d'interface créée à partir de la classe *FlowLayout* et dont les composants sont alignés à droite.

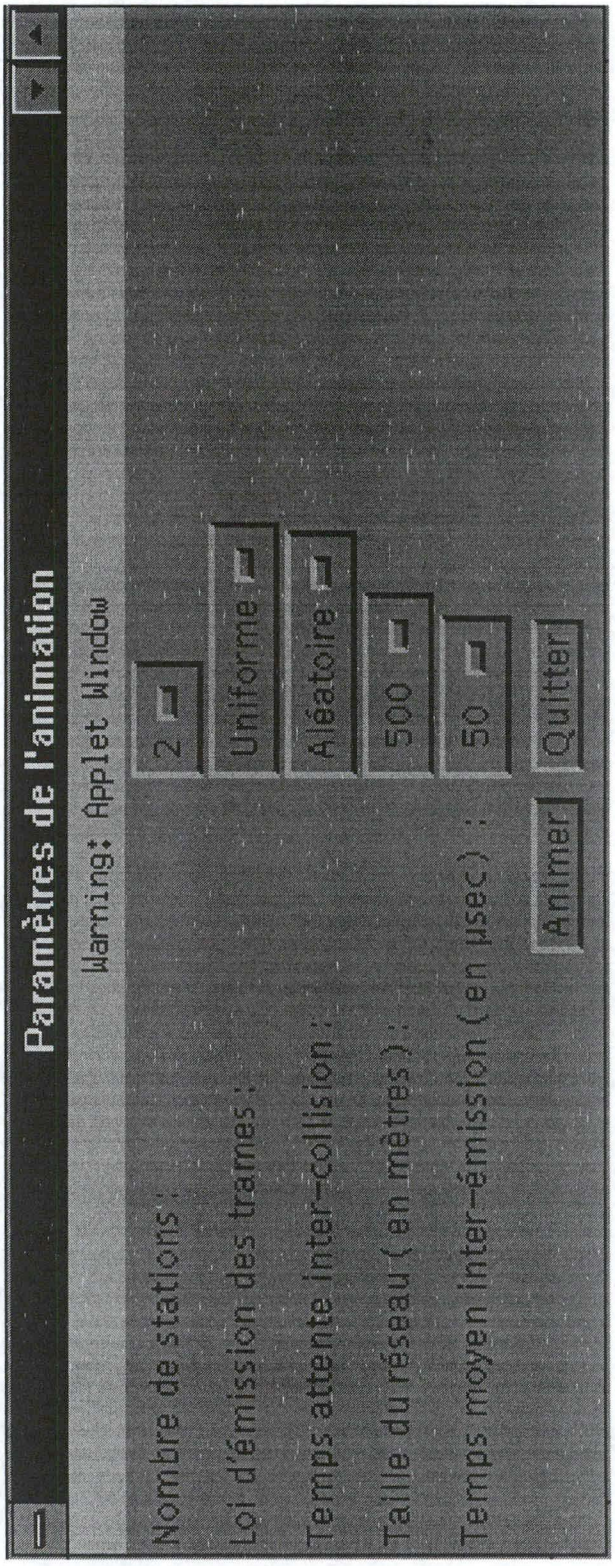

Figure 4.2 : Un exemple de fenêtre créée à partir de la classe *GridLayout*.

- *BorderLayout* : elle affiche une fenêtre dont les composants sont appelés *North*, *South*, *East*, *West* et *Center*. La taille d'un composant est calculée en fonction de son contenu et en fonction de la taille de la fenêtre.
- *CardLayout* : elle crée une fenêtre pouvant contenir plusieurs "cartes"; une seule carte pouvant être visible à un instant donné. L'application peut bien entendu voyager d'une carte à l'autre.
- *GridBagLayout* : elle permet l'affichage flexible des composants de la fenêtre. Cet affichage permet d'aligner horizontalement et verticalement les composants sans exiger qu'ils possèdent tous la même taille.
- *GridLayout* : elle crée une fenêtre divisée en un certain nombre de lignes et de colonnes; chaque composant ajouté à cette fenêtre est rangé dans une des cases du tableau ainsi créé. La différence essentielle avec le *GridBagLayout* est qu'ici toutes les cases du tableau ont la même taille. La figure 4.2 nous montre une interface obtenue à partir de ce layout.

Parmi l'ensemble des interfaces développées en utilisant les classes prédéfinies citées ci-dessus, aucune ne s'est avérée être totalement satisfaisante d'un point de vue ergonomique. Examinons par conséquent l'une des interfaces développées pour mettre en évidence les lacunes liées à ce type particulier de fenêtre.

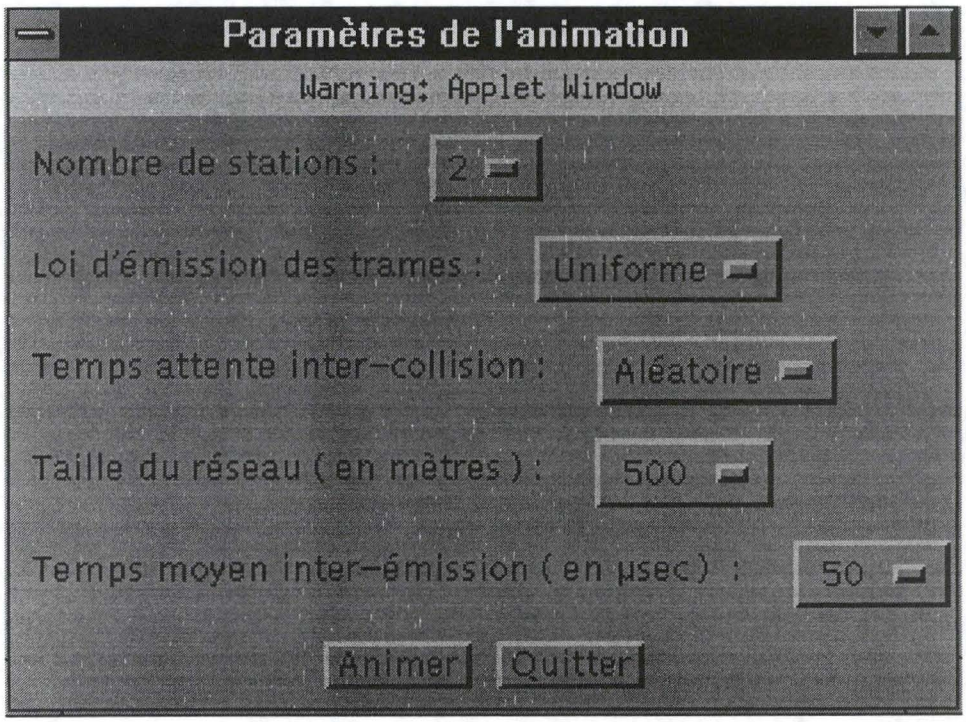

Figure 4.3 : la fenêtre de saisie des paramètres de l'animation

La figure 4.3 montre une des nombreuses interfaces développées pour l'introduction des paramètres de l'animation : elle utilise la classe *FlowLayout* dont les composants sont alignés à gauche.

L'examen de la fenêtre de la figure 4.3 nous montre que certaines règles ergonomiques n'ont pas été respectées[1]. Il s'agit du positionnement et de la taille des listes de sélection et des boutons ainsi que le positionnement des labels des listes de sélection. En effet, les listes de sélection doivent être alignées à gauche sur les « : » alors que les labels des listes correspondantes doivent être alignés à droite sur ce même symbole.

Chaque fenêtre (telle que celle représentée ci-dessus) est divisée en autant de zones que le souhaite le concepteur. Chaque zone de la fenêtre peut utiliser un affichage qui lui est propre parmi la liste des cinq classes permettant de créer une interface. Cependant, il n'est pas possible de positionner de façon précise un objet à l'intérieur d'une zone déterminée; les seules possibilités sont l'alignement à gauche, à droite ou centré dans le cas de la classe *FlowLayout*. La fenêtre de la figure 4.3 est divisée en six zones définies chacune par l'intermédiaire de la classe *FlowLayout* : les cinq premières zones sont utilisées pour introduire les paramètres et la dernière pour les boutons de commande *Animer* et *Quitter*. Les paramètres à introduire sont alignés à gauche alors que les boutons de commande sont centrés.

Etant donné le type d'alignement choisi pour les paramètres de l'animation (c'est-à-dire les cinq premières zones), la longueur de la chaîne de caractère influence donc le positionnement des listes de sélection. Il est donc fort peu probable qu'elles soient toutes alignées les unes en dessous des autres. La première esquisse de solution consiste à introduire des caractère « blancs » au début des messages[2] précédant les listes de sélection pour essayer d'aligner l'ensemble sur le symbole « : ». Cela ne s'avère pas fort satisfaisant. La seconde solution consiste à redéfinir la classe des listes de sélection afin de pouvoir positionner ces dernières à l'endroit désiré. Cette solution sera décrite par la suite.

Le problème suivant est lié à la taille des listes de sélection. En effet, on peut remarquer que deux listes de sélection quelconques n'ont jamais une taille identique. Cela s'explique de la façon suivante : la taille de la cellule contenant les items de la liste de sélection correspondante est calculée à partir de la longueur maximale d'une chaîne de caractère contenue dans la liste des items à sélectionner. Par exemple, la loi d'émission des trames est soit *uniforme* soit de

[1] cfr la description des règles ergonomiques du chapitre 2.

[2] par exemple *Taille du réseau (en mètres)*.

poisson. La taille de la cellule est donc calculée sur la longueur de la chaîne de caractères *uniforme* puisque cette dernière est la plus longue. La première solution consiste de nouveau à insérer des caractères « blancs » dans ces chaînes de caractères afin d'obtenir des listes de sélection de taille constante. Cependant, cette solution n'est pas réalisable avec les listes de sélection contenant des entiers. Par conséquent, une re-définition de la classe doit à nouveau être envisagée.

Le code permettant de créer la liste de sélection correspondant au nombre de stations de la figure 4.3 est présenté à la figure 4.4.

```
1     Panel northNorthCenterPanel = new Panel();
2     northNorthCenterPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
3     northNorthCenterPanel.add(new Label("Nombre de stations :"));
4     NbrStaTF = new Choice();
5     NbrStaTF.addItem("2");
6     NbrStaTF.addItem("3");
7     NbrStaTF.addItem("4");
8     NbrStaTF.addItem("5");
9     NbrStaTF.addItem("6");
10    northNorthCenterPanel.add(NbrStaTF);
11    northCenterP.add("North",northNorthCenterPanel);
```

Figure 4.4 : Le code[3] de la liste de sélection *Nombre de stations*.

Examinons à présent ligne par ligne ce que signifie le code présenté ci-dessus :

- la ligne 1 indique la création d'une nouvelle zone appelée *northNorthCenterPanel* dans la fenêtre contenant les paramètres de l'animation.
- la ligne 2 indique comment seront affichés les composant de la zone *northNorthCenterPanel*, dans notre cas, ils seront alignés à gauche en utilisant le layout *FlowLayout*.
- la ligne 3 ajoute à la zone *northNorthCenterPanel* le label correspondant à la liste de sélection *NbrStaTF*.
- les lignes 4 à 9 définissent la liste de sélection et son contenu. La ligne 4 déclare la variable *NbrStaTF* comme étant de type *Choice*. Les ligne 5 à 9 définissent le contenu de la liste de

[3] Un entier a été ajoué dans la figure 4.3 au début de chaque ligne de code afin de pouvoir commenter plus facilement ce dernier.

sélection. La valeur par défaut de *NbrStaTF* est le premier élément ajouté à la liste de sélection à savoir la valeur 2 (ligne 5 de la figure 4.4).

- la ligne 10 affiche cette liste de sélection dans la zone *northNorthCenterPanel* à la suite du message « *Nombre de stations :* » ajouté à cette même zone à la ligne 3.
- la ligne 11 clôture la définition de cette zone.

Les problèmes mis en évidence précédemment sont de toute évidence issus de la ligne 2 de la figure 4.2. En effet, c'est à cette ligne qu'est défini l'affichage utilisé dans la zone *northNorthCenterPanel*. Pour résoudre de problème, il suffit de créer une nouvelle classe permettant de positionner et de choisir la dimension des listes de sélection ainsi que les commentaires les précédant. Cette nouvelle classe ne tiendra compte que des problèmes cités auparavant.

Pour obtenir une interface ergonomique, il suffit :

- d'utiliser l'idée contenue dans la classe *FlowLayout*. C'est-à-dire qu'il suffit d'afficher les composants de la fenêtre en les justifiant tous à droite.
- de changer la taille de toutes les listes de sélection de façon à ce que ces dernières possèdent toutes la même dimension.

Les deux conditions précédentes garantissent donc une interface ergonomique lorsqu'une interface est du type de celle que l'on doit développer.

Décrivons à présent cette nouvelle classe et examinons en détails quels sont les changements apportés par cette nouvelle classe après avoir décrit ses variables et méthodes :

```
public class NewLayout implements LayoutManager {

        private int preferredWidth = 0, preferredHeight = 0;

        public NewLayout() { }
        public NewLayout(int v) { }
        public void removeLayoutComponent (Component comp) { }
        public void addLayoutComponent (String name, Component comp) { }
        private void setSizes (Container parent) { }
        public Dimension preferredLayoutSize(Container parent) { }
        public Dimension minimumLayoutSize(Container parent) { }
        public void layoutContainer(Container parent) { }
        public String toString() { }
```

Figure 4.5 : la classe *NewLayout*.

Les variables *preferredWidth* et *preferredHeight* indiquent la taille idéale que prendra la fenêtre contenant les listes de sélection ainsi que leurs labels. Les méthodes sont décrites dans le tableau suivant :

Méthodes	Descriptions
public NewLayout()	il s'agit du constructeur de la classe. Il crée une nouvelle fenêtre dont les éléments seront espacés horizontalement et verticalement de 5 pixels[4].
public NewLayout(int v)	ce constructeur permet de créer une fenêtre dont les éléments seront espacés horizontalement et verticalement d'une valeur de *v* pixels.
public void addLayoutComponent(String name, Component comp)	cette méthode se charge d'ajouter à la zone utilisant le layout *NewLayout* un nouveau composant *comp* de nom *name*.
public void removeLayoutComponent (Component comp)	cette méthode enlève le composant *comp* d'une zone de la fenêtre.

[4] cette valeur de 5 pixels est la valeur utilisée par défaut pour toutes les classes citées précédemment.

Méthodes	Descriptions
public void setSize(Container parent)	cette méthode recalcule la taille de la fenêtre *parent* à partir de la taille de chacun de ses composants.
public Dimension preferredLayoutSize (Container parent)	cette méthode détermine la taille idéale[5] de la fenêtre *parent* contenant les paramètres de l'animation. La taille idéale de la fenêtre *parent* est la valeur de la taille idéale des zones qui composent la fenêtre augmentée de: (nombre de composants de la zone) * espacement horizontal[6]. Il ne faut pas oublier d'ajouter les marges de gauche et de droite de la fenêtre pour avoir cette taille idéale.
public Dimension minimumLayoutSize (Container parent)	cette méthode détermine la taille minimale de la fenêtre *parent* contenant les paramètres de l'animation.
public void layoutContainer(Container parent)	cette méthode affiche la zone *parent* dans la fenêtre.
public String toString()	cette méthode renvoie une représentation sous forme de string du layout *NewLayout*.

La figure 4.6. montre l'interface obtenue à partir de la classe *NewLayout*.

Nous pouvons remarquer que l'interface obtenue ne vérifie pas encore tout à fait les critères exposés précédemment. En effet, les listes de sélection ont encore des tailles différentes.

Puisque la taille de la fenêtre est recalculée lors de chaque ajout d'un composant à celle-ci, il semblait réalisable d'utiliser un procédé semblable pour changer les dimensions des listes de sélection. Pour ce faire, il suffisait de déclarer une variable *MaxSizeButton* globale à la classe *NewLayout* et de la mettre à jour lors de l'affichage d'une nouvelle liste de sélection. Cette mise à jour étant réalisée uniquement si la taille de la nouvelle liste de sélection est plus grande que la valeur contenue dans la variable *MaxSizeButton*. D'une manière assez

[5] la taille idéale de la fenêtre est la taille minimisant la superficie de la fenêtre.

[6] ou vertical puisqu'ils sont égaux.

incompréhensible, la valeur de la variable *MaxSizeButton* vaut la taille de la liste de sélection courante. Par la suite, des essais ont été réalisés en vue de modifier la taille des listes de sélection. De nouveau, cela n'a produit aucun résultat probant. La dernière solution envisageable pour obtenir une interface plus ou moins ergonomique a été d'essayer de connaître la taille de la plus grande liste de sélection afin d'aligner les listes de sélection et les labels correspondants sur les deux points les séparant. Cela implique donc que la classe *NewLayout* développée est donc particulière à notre problème.

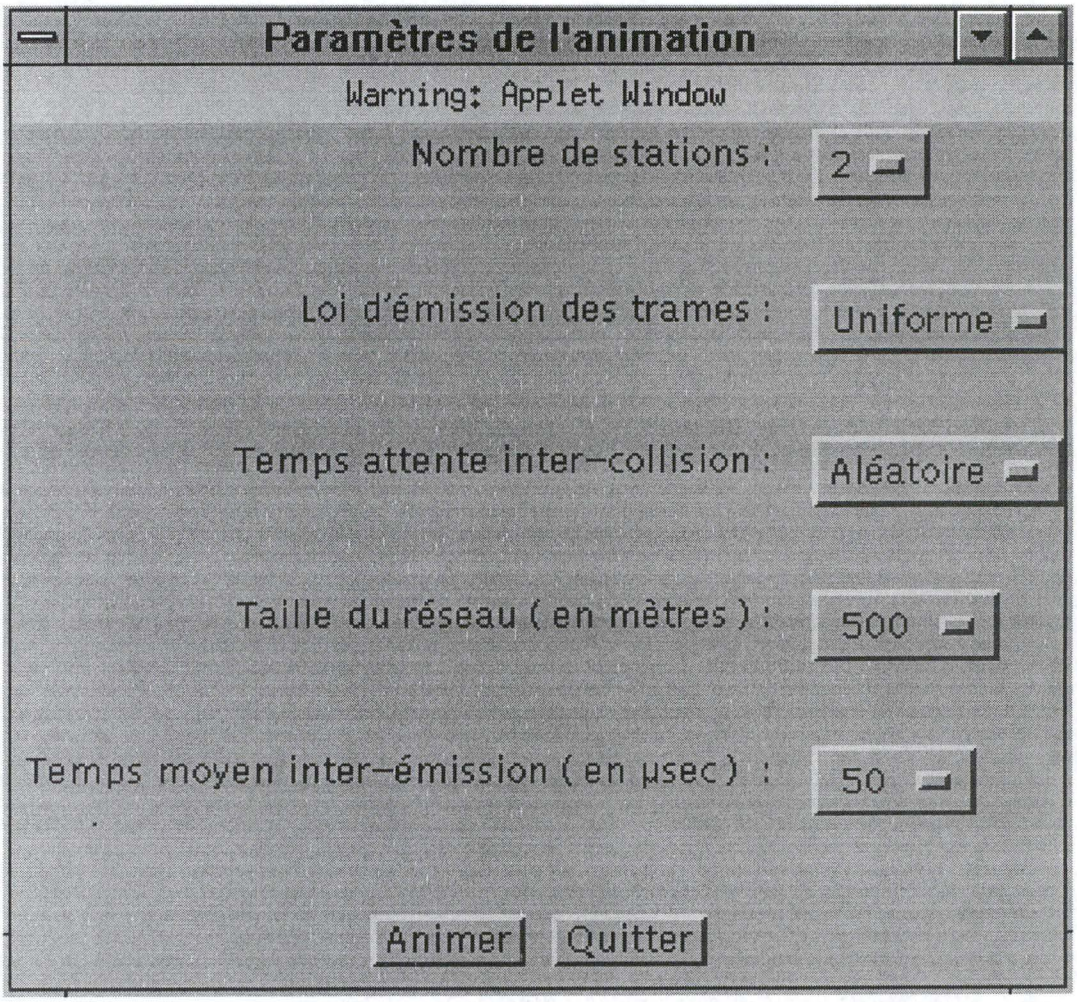

Figure 4.6 : L'interface réalisée avec la classe *NewLayout*.

2.2 La création de la liste des événements.

La liste des événements est initialisée par les méthodes de la classe *ListOfEvenement* à partir d'un appel de la classe *AnimTrame*.

L'unique problème rencontré lors de la création de cette liste a été un problème de communication d'informations entre les classes *AnimTrame* et *ListOfEvenement*. En effet, il a fallu créer une méthode particulière *initValeur* dans la classe *AnimTrame* afin de pouvoir initialiser la liste des événements de la classe *AnimTrame*.

En examinant la déclaration de cette méthode, on peut remarquer qu'elle diffère des autres en ce sens qu'elle est déclarée statique :

```
public static void initValeur(ListOfEvenement ListEv)
```

Java, tout comme les autres langages orientés objets, fournit des méthodes de classes et des variables de classes. Normalement, les variables déclarées à l'intérieur d'une classe sont des *instances* de variables; c'est-à-dire que ces variables existent dans chaque objet instancié (créé) à partir de cette classe. Par contre, une variable de classe est locale à la classe elle-même; c'est-à-dire qu'il existe une seule copie de la variable et que cette dernière est partagée par tous les objets instanciés à partir de cette même classe.

Les méthodes de classes sont donc également communes à l'entièreté de la classe. Ces méthodes sont généralement utilisées lorsqu'on rencontre un comportement commun pour tous les objets de la classe.

L'unique restriction est que les méthodes de classes ne peuvent utiliser que des variables de classes. Les méthodes de classes ne peuvent avoir accès aux instances de variables ni invoquer des instances de méthodes.

2.3 L'animation proprement dite.

Les problèmes qui se sont posés au cours de la création de l'animation sont essentiellement dû à l'utilisation des *Threads* pour éviter que le browser ne soit pris en otage par l'applet.

Par exemple, lorsqu'une collision se produit sur le réseau, une fenêtre apparaît et indique quelle station a détecté la collision. Puisqu'il existe un contexte d'exécution pour l'animation, il faut suspendre ce dernier afin que l'utilisateur puisse lire le message sans que l'animation ne se poursuive. Un fois le message lu, l'utilisateur ferme la boîte de dialogue en pressant un bouton; ce qui rétablit le contexte d'exécution de l'animation : l'animation peut ainsi poursuivre son cours à partir du moment où elle a été suspendue.

3 Les commentaires concernant le cours.

Lors de la création du cours, il a été prévu de permettre à l'utilisateur d'ajouter ou de consulter une base de données contenant les commentaires au sujet soit du contenu du cours soit de sa présentation.

A l'heure actuelle, seule l'interface a été réalisée. Le code permettant de l'obtenir est fort semblable à celui utilisé pour réaliser l'interface s'occupant de l'introduction des paramètres de l'animation.

La figure 4.7 nous montre l'interface réalisée pour la saisie des commentaires.

Puisqu'il n'est pas possible de prévoir quelles seront les informations introduites par l'utilisateur, les listes de sélection ne peuvent être utilisées pour l'introduction des commentaires. Elles sont remplacées par des champs d'édition.

Contrairement à l'interface réalisée pour l'introduction des paramètres de l'animation, il est ici possible d'imposer la taille des champs d'édition. Cela implique donc que l'ergonomie de l'interface réalisée est meilleure que pour l'introduction des paramètres de l'animation.

Pour définir un champs d'édition, il suffit de déclarer une variable de type *TextField* ayant comme paramètre la taille du champs d'édition souhaité. Par exemple, l'utilisation du code :

```
Panel centerCenterPanel = new Panel();
northNorthCenterPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));
northNorthCenterPanel.add(new Label("Nom et prénom :"));
NomTF = new TextField(30);
CenterPanel.add(NomTF);
CenterP.add("Center",centerCenterPanel);
```

permet d'obtenir le champs d'édition suivant dont la taille est de 30 caractères :

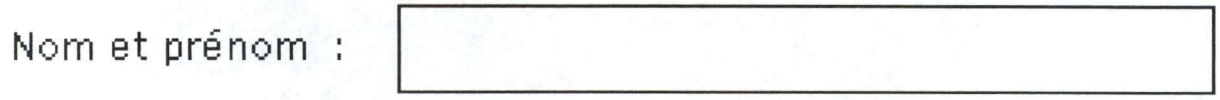

Figure 4.7 : un exemple de champs d'édition.

L'utilisation des champs d'édition est efficace pour les champs ne possédant pas un format spécial (*Nom et prénom*, *Commentaires*) mais est inefficace pour les données formatées. Par exemple, l'introduction d'une adresse électronique dans un champs d'édition n'est pas parfaite. En effet, il est fort difficile de vérifier que l'adresse introduite possède un format correct.

4 Conclusion.

Les avantages principaux du langage Java mis en évidence par notre didacticiel sont que ce langage permet une utilisation assez aisée de la programmation parallèle (grâce à l'utilisation des threads) ainsi que le développement assez rapide d'une interface (grâce aux classes de la librairie *java.awt*).

Cependant, l'inconvénient majeur de ce langage se situe au niveau des interfaces développées; ces dernières ne répondent à aucun critère ergonomique. Il serait fort utile de penser au développement de classes permettant de créer des interfaces ergonomiques.

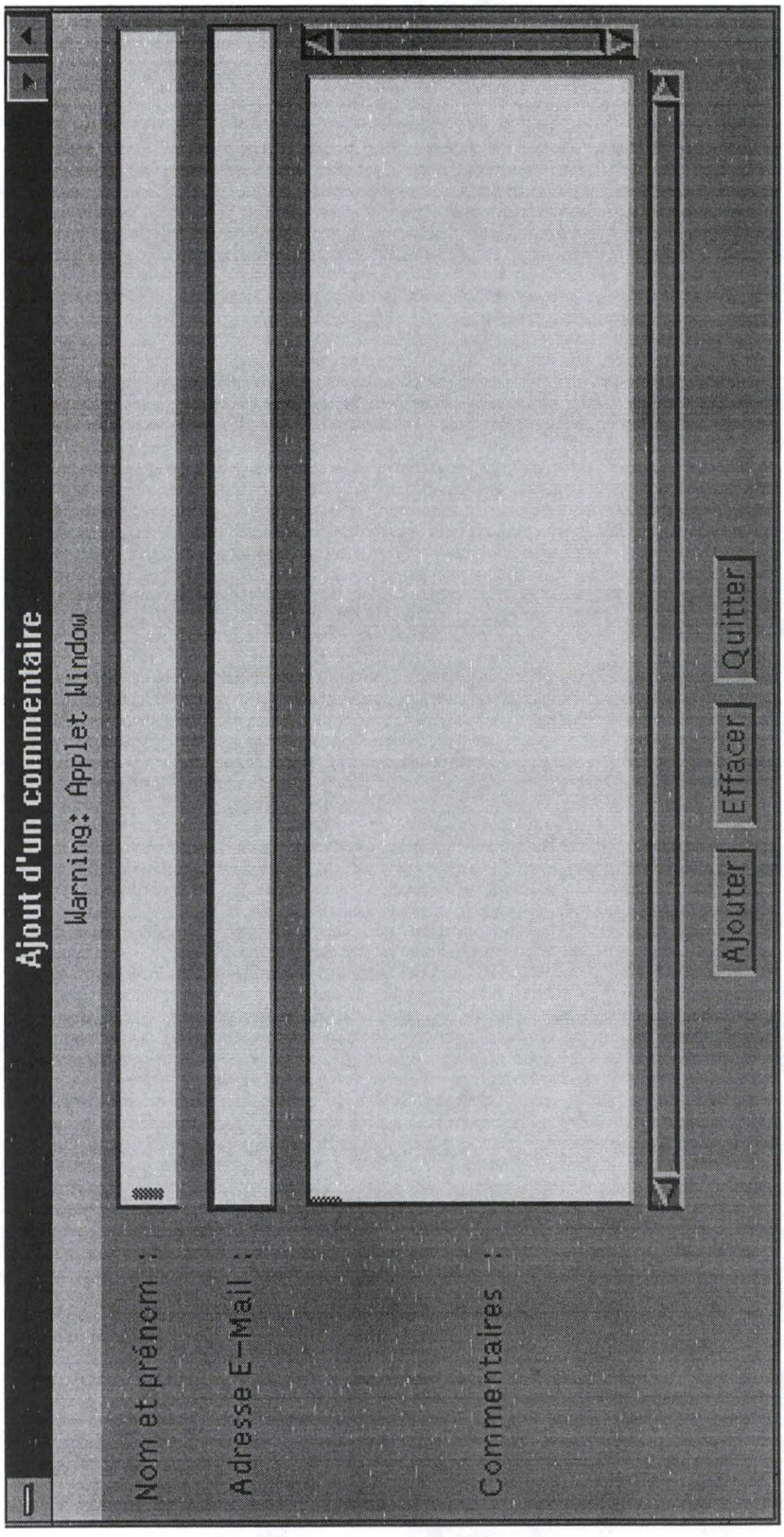

Figure 4.8 : la fenêtre de saisie des commentaires.

Chapitre 5 : Critiques et perspectives.

Ce chapitre a donc pour objectif de critiquer les résultats obtenus ainsi que de proposer quelques pistes pour une continuation du développement du didacticiel : c'est ce que nous appellerons perspectives.

1 Critique.

Pour réaliser ce travail, nous avons dû acquérir de nombreuses connaissances dans de nombreux domaines. Ainsi nous n'avons pu qu'effleurer les différents sujets présentés. C'est pourquoi nous ne pouvons en aucune manière prétendre que ce travail constitue une référence en la matière.

Il semble important de rappeler ici que les notions présentées dans la partie théorique de l'application ne couvrent pas l'entièreté des concepts liés au protocole Ethernet mais qu'elles permettent à l'utilisateur d'apprendre les notions utiles à la compréhension de l'animation.

De plus, nous pouvons également souligner le fait que nous avons développé notre didacticiel par « morceau » ou , autrement dit, par prototypage. De cette façon, chaque étape

du développement du didacticiel a pu être discutée avant de continuer son développement. Ce genre de pratique permet d'assurer une certaine qualité du produit fini mais ne facilite certainement pas la rédaction de l'architecture globale de l'application.

Nous pouvons regretter de n'avoir pas eu suffisamment de temps matériel pour pouvoir compléter notre didacticiel dont il manque malheureusement les parties *exercices* et *évaluation*. En plus de ces concepts centraux liés à l'apprentissage, il nous paraît dommage de décrire un produit ne présentant pas de fonctions d'aide.

Il est également regrettable de ne pas avoir eu le temps de procéder à une évaluation d'utilisation de ce didacticiel grâce à un test sur une population cible. Cette évaluation aurait certainement contribué à enrichir notre expérience de le domaine de l'Enseignement Assisté par Ordinateur en plus de mettre en évidence certaines lacunes de notre didacticiel.

2 Perspectives.

Il nous semble également important de fournir à nos successeurs quelques pistes de continuation du projet réalisé. Celles nous venant à l'esprit peuvent se diviser en deux catégories : celles ayant pour sujet le cours lui-même et celles issues du cours.

2.1 Les perspectives ayant pour sujet le cours lui-même.

Ces perspectives peuvent se découper en deux classes distinctes. Tout d'abord celle située au *niveau du cours*, ensuite celle située *au dessus du cours* sur Ethernet.

Ce que nous appelons les *perspectives au niveau du cours* sur Ethernet concernent les éventualités de continuation du développement du didacticiel soit pour compléter les dimensions encore inexistantes telles que les exercices ou l'évaluation de l'utilisateur soit pour englober la partie développée dans le cadre d'un autre projet du même type.

En ce qui concerne la continuation du développement du cours sur Ethernet, les points suivants nous paraissent primordiaux :

- l'implémentation de l'aspect d'*évaluation* qui poursuit deux objectifs. Le premier consiste à vérifier que l'apprenant maîtrise les concepts enseignés dans la partie théorique. Le deuxième aspect consiste à mettre en évidence, de par l'étude des résultats obtenus par les utilisateurs, les points faibles de la théorie. En effet, si l'on remarque que certaines erreurs se produisent assez souvent lors de l'étape d'évaluation, peut-être est-ce dû à un mauvais enchaînement des concepts enseignés. Quoi qu'il en soit, une remise en question de la partie théorie sera plus que probablement nécessaire.
- l'implémentation de la partie « exercices » dont l'objectif principal de permettre à l'utilisateur de vérifier s'il maîtrise la théorie. Il serait même concevable de fournir non seulement la réponse correcte à la question à laquelle l'apprenant a répondu mais également les étapes menant à cette solution. De cette façon, l'utilisateur pourrait localiser l'origine de ses erreurs et focaliser son attention sur la partie théorique y correspondant.
- l'implémentation des fonctions permettant l'ajout ou la consultation de commentaires relatifs au cours ou à la présentation du cours à une base de données. Cela poursuit également un double objectif. Le premier objectif consiste à permettre à l'utilisateur de poser des questions relatives au cours ou à sa présentation à d'autres personnes l'ayant également suivi. Le second permet au créateur du didacticiel d'avoir un feed-back des utilisateurs afin d'améliorer ou d'éclaircir certains aspects du cours.

Ce que nous appelons *perspectives au dessus du cours* sont les éventualités de continuation de développement du didacticiel afin qu'il englobe des notions relatives à d'autres sujets; par exemple, l'intégration d'un autre aspect du cours visant à expliquer le fonctionnement d'Internet. L'utilisateur aurait alors accès à un menu principal lui permettant d'accéder soit au cours sur le protocole Ethernet soit sur Internet.

2.2 Les perspectives issues du cours.

Ces perspectives concernent essentiellement le développement d'un cours ayant pour sujet une matière quelconque mais dont notre didacticiel pourrait servir de modèle.

Le travail présenté précédemment constituerait alors une source de documentation concernant les choses à faire ou à ne pas faire lors du développement d'une application de type Enseignement Assisté par Ordinateur utilisant les potentialités du langage Java.

De plus, grâce à la synthèse des avantages et inconvénients du langage Java, mis en évidence par notre didacticiel, il serait utile de suggérer la création de nouvelles classes pour la librairie *java.awt* permettant de créer des interfaces prenant en compte certains critères ergonomiques.

Comme nous pouvons le remarquer, les perspectives de continuation de ce projet ne manquent donc certainement pas.

Conclusion.

Le développement d'un didacticiel constitue une tâche particulièrement complexe puisqu'elle nécessite l'intégration de connaissances diverses allant de notions pédagogiques aux notions ergonomiques en passant par les notions relatives à la matière abordée par le didacticiel. Nous espérons que ce mémoire aborde avec suffisamment de précision ces diverses dimensions afin de fournir une base solide à d'éventuels successeurs.

Nous avons essayé, dans la mesure de nos possibilités, d'appliquer au maximum les notions abordées dans la partie théorique pour le développement de notre didacticiel. Pour ce faire, nous avons essayé d'être méthodique et de ne rien laisser au hasard. Etant donné la difficulté de cette tâche, nous espérons que la méthode utilisée est acceptable.

Nous n'avons cependant pas eu le temps matériel de réaliser toutes les étapes de la phase de développement de notre didacticiel. Les prochaines étapes à accomplir sont de réaliser les aspects d'exercices et d'évaluation de l'utilisateur ainsi qu'un test auprès de la population cible afin d'évaluer le didacticiel d'un point de vue utilisation.

Bibliographie.

[BAR 91] Georges-Louis Baron, Brigite de la Passardière : *Médias, Multi et Hypermédias pour l'apprentissage : Points de repères sur l'émergence d'une communauté scientifique*, Actes des premières journées scientifiques 24-25 septembre 1991 : pp 7-15

[BOR 90] J. Bordier, G. Paquette et S. Carrier : *Building Learning Environments using Generic Software*, Proceedings of the 4th World Conference on Computer and Education, Sydney, Australie, juillet 1990.

[CHI 86] Chi : *Proceeding of the conference on Computer Human Interface*, ACM Press, Addisson Wesley, New-York, N.J., 1986.

[DUF 88] Aude Dufresne : *Aspects intelligents des interfaces de communication et utilisation en éducation.* In Sogides (Ed.), Education, Apprentissage et Ordinateur. Montréal, 1988.

[DUF 91] Aude Dufresne : *Ergonomie cognitive, Hypermédias et apprentissages*, Actes des premières journées scientifiques 24-25 septembre 1991 : pp 121-131

[HIL 96] David R C Hill : *Object-Oriented Analysis and Simulation.* Addison Wesley Developers Press, 1996.

[HOF 96] Arthur van Hoff, Sami Shaio, Orca Starbuck : *Hooked on JAVA.* Addison Wesley Developers Press , janvier 1996.

[LEC 91] Dieudonné Leclercq : *Hypermédia et Tuteurs Intelligent : vers un compromis*, Actes des premières journées scientifiques 24-25 septembre 1991 : pp 19-35

[LIN 95] M. Linard : *New debate on learning support*, Journal Of Computer Assisted Learning (1995) 11, 239-253

[MEN 95] Patrick Mendelsohn, *EIAO et psychologie cognitive*, Sciences et techniques éducatives Vol.2 - n° 1/1995, pages 9-29

[NOIR 93] M. Noirhomme-Fraiture : *Simulation de systèmes*, Institut d'Informatique, Facultés Universitaires Notre Dame de la Paix, Namur.

[PAL 90] J. Palmer, T. Duffy, B. Mehlenbacher : *A System for Aiding Designers of Online Help*. Lotus : acm SIGCHI, 1990.

[PAQ 88] G Paquette : *Environnements d'apprentissages à bases de connaissances*. Conférence invitée au colloque international de pédagogie informatique, Actes publiés par l'Université de Bologne, Italie, juin 1988.

[RHE 91] Jacques Rhéaume : *Hypermédias et Stratégies Pédagogiques* , Actes des premières journées scientifiques 24-25 septembre 1991 : pp 45-58

[SOU 94] Gérard Soula, Jean-Michel Bartoli, Mario Fieschi, *Hypermédia et apprentissage en médecine : le projet Forum*, Sciences et techniques éducatives Vol.1 - n°4/1994, pages 521-538

[VANDER 94] VANDERDONCKT J. (1994), *Guide ergonomique des interfaces homme-machines*, Les Presses Universitaires de Namur, Namur.

[VANBAST 93] VAN BASTELAER P., NACHTERGAELE V., COTET M. (1993), *Notes provisoires pour les cours de Téléinformatique et réseaux*, Institut d'Informatique, Facultés Universitaires Notre Dame de la Paix, Namur.

[HTML1] *Hotjava(tm) : The Security Story.*
http://java.dimension.c..c/security/security.html

[HTML2] *The HOTJAVA™ Browser : A White Paper.*
http://java.dimension.c..rview/hotjava/index.html.

[HTML3] James Gosling, Henry Mc Gilton :*The Java™ Language : A White Paper.*
http://www.labri.u-bordeaux.fr/chaumett/java/sundoc/java-blancpapier/java-whitepaper-1.html

[HTML4] *What is the Java Platform ?*
http://www.javasoft.com/doc...JavaPlatform.doc1.html#5760

[HTML5] *A Look Inside the Java Platform*
http://www.javasoft.com/doc...JavaPlatform.doc2.html#7205

[HTML6] Arthur van Hoff :*Java and Internet Programming.*
http://www.ddj.com/ddj/1995/1995.08/hoff.html