THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Contribution à l'Enseignement Assisté par Ordinateur dans le domaine des télécommunications. Un didacticiel pour Internet

Gobert, Xavier

Award date:
1995

Awarding institution:
Universite de Namur

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

Année académique 1994 - 1995

Contribution à l'Enseignement Assisté par Ordinateur dans le domaine des télécommunications. Un didacticiel pour Internet.

Xavier Gobert

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

Résumé

Dans ce mémoire, nous abordons l'Enseignement Assisté par Ordinateur (E.A.O.) par la présentation du développement d'un logiciel de formation destiné à fournir un apprentissage des notions de base d'Internet. Pour ce faire, nous présentons d'abord quelques notions théoriques relatives à l'E.A.O. telles qu'une méthodologie de développement d'un didacticiel, 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. Nous espérons que les résultats obtenus pourront être utilisés à l'avenir dans le cas d'une éventuelle 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 aimed at the teaching of the Internet's basic notions. First we introduce some C.A.T. theorical notions such as a methodology for the development of a C.A.T. application, centered on the concept of graphic interface. We then describe, in an educational and software engineering approach, the C.A.T. software we have developed. Finally, we carry out an evaluation of our C.A.T. software and we propose some future works. We hope that the results of our master thesis will be used for the continuation of our project.

Remerciements

Je tiens tout d'abord à exprimer mes plus vifs remerciements aux membres de l'équipe COLOS de Lyon qui m'ont aidé tout au long de mon stage. L'ambiance qui a régné durant toute la durée du stage a été réellement propice au travail. Je tiens à remercier plus particulièrement Daniel Muller, Luc Mariaux et Alain Nicolas pour leurs conseils judicieux et le temps qu'ils m'ont consacré.

Je tiens également à remercier mon promoteur, Philippe van Bastelaer, pour toute l'aide qu'il m'a apportée et le temps qu'il a consacré à la lecture de ce mémoire.

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

Table des matières

Introduction

L'enseignement assisté par ordinateur (E.A.O.) est une pratique qui, à l'heure actuelle, est encore peu utilisée. Elle permet pourtant d'établir un lien supplémentaire entre le professeur et l'élève grâce à son outil particulier : le logiciel de formation. Ce lien permet de renforcer l'apprentissage tant au point de vue compréhension d'une matière abordée que du point de vue assimilation des différentes notions relatives à cette matière. L'objet de ce mémoire est d'aborder le monde de l'E.A.O. par la présentation d'un logiciel de formation développé dans le but de fournir un apprentissage de diverses notions du monde des télécommunications.

La base de départ de ce travail est le développement d'un logiciel de formation destiné à fournir un apprentissage des notions de base d'Internet. Un tel développement nécessite cependant de la part d'un concepteur de s'intéresser d'abord aux éventuelles méthodologies ou règles qui existent et qui imposent une marche à suivre dans sa tâche. On dispose pour cela d'une littérature relativement abondante qui va du guide de conception d'un didacticiel à la grille d'analyse d'un logiciel de formation en passant par toute une série d'ouvrages relatifs à la pédagogie de l'enseignement et à des guides de conception d'interface homme-machine.

Afin de traduire aussi précisément que possible la démarche que devrait suivre le concepteur, ce mémoire a été structuré de la façon suivante : les trois premiers chapitres fournissent une approche théorique de l'E.A.O. en général et du développement d'un didacticiel en particulier; les quatre chapitres suivants présentent alors une approche plus pratique en s'intéressant au didacticiel développé dans le cadre de ce travail et aux notions abordées dans ce didacticiel; enfin, le dernier chapitre présente une évaluation du didacticiel ainsi qu'un certain nombre de perspectives pour la continuation du projet.

Le premier chapitre aborde les facteurs pédagogiques à prendre en compte lors du développement d'un didacticiel. Il propose ensuite une méthodologie de développement de didacticiels avant de présenter le projet COLOS au sein duquel nous avons travaillé tout au long de notre travail. Le second chapitre insiste sur une dimension particulière de la phase de développement d'un didacticiel qui est la conception de son interface. Quant au troisième chapitre, il aborde l'E.A.O. dans le domaine des télécommunications en présentant quelques logiciels de formation destinés à l'apprentissage de notions de ce domaine.

Le quatrième chapitre est entièrement consacré à la présentation des notions de base d'Internet qui forment le contenu de notre didacticiel. Le cinquième chapitre décrit en détail notre didacticiel d'un point de vue pédagogique mais également d'un point de vue informatique. Le sixième chapitre passe ensuite en revue quelques outils d'aide à la conception d'une interface que nous avons utilisés lors du développement du didacticiel. Le septième chapitre présente alors les choix faits et les difficultés rencontrées tout au long du développement du didacticiel.

Et le huitième et dernier chapitre offre une évaluation de notre didacticiel basée sur une grille d'analyse conçue par des spécialistes de l'E.A.O. Ce chapitre présente ensuite une critique de notre travail avant de fournir quelques perspectives pour l'éventuelle continuation du projet.

Chapitre 1 :
L'Enseignement Assisté par Ordinateur.

Tout professeur tente toujours de rendre son exposé le plus clair possible. De nos jours, ces professeurs peuvent utiliser la technologie informatique pour ce faire. Ou, pour utiliser le terme consacré, ils peuvent profiter d'un nouveau type d'enseignement appelé Enseignement Assisté par Ordinateur (E.A.O.).

Nous nous baserons pour le développement de ce chapitre sur l'ouvrage "Concevoir et utiliser un didacticiel" de Besnainou, Muller et Thouin [BESN88].

1. Qu'est-ce que l'E.A.O. ?

Il existe une multitude d'ouvrages traitant de l'E.A.O.; en tirer une définition claire et explicite n'est pas chose aisée. Toutefois nous pouvons dire que l'E.A.O. est l'enseignement de matières quelconques utilisant l'ordinateur comme moyen de communication privilégié entre l'élève et l'enseignant.

Cependant, si l'ordinateur est indispensable, il est néanmoins inutilisable sans logiciels d'enseignement ou « didacticiels ». Cette perspective met en évidence le besoin de concepteurs et, puisqu'il s'agit d'enseignement, le besoin d'une démarche pédagogique lors du développement de telles applications.

Cette démarche pédagogique est d'autant plus importante que l'E.A.O. permet une nouvelle approche de l'enseignement. En effet, « l'un des intérêts majeurs de l'E.A.O. nous semble être la possibilité de tendre au maximum vers un enseignement individualisé » [BESN88]. Nous approfondirons ultérieurement cette notion d'individualisation. L'E.A.O. se

base donc sur une pédagogie centrée sur l'élève par opposition à la pédagogie centrée sur le contenu.

Une pédagogie centrée sur l'élève est en général une pédagogie prenant en compte les motivations de l'élève : ses intérêts, ses goûts, son rythme de travail, son expérience antérieure, etc... C'est donc une « pédagogie de l'apprentissage individualisé ».

Une pédagogie centrée sur le contenu est, par opposition, fondée sur la manière d'enseigner et non sur l'apprentissage de l'étudiant. Elle est, par définition, « centrée sur l'enseignant et la transmission des connaissances ».

1.1. L'individualisation et l'interactivité

L'individualisation et l'interactivité sont deux notions fondamentales de l'E.A.O.

1.1.1. L'individualisation

Comme nous l'avons déjà souligné plus haut, un intérêt majeur de l'E.A.O. est d'offrir la possibilité de tendre vers un enseignement individualisé, c'est à dire de permettre de prendre en compte les caractéristiques propres à l'étudiant lors de l'apprentissage.

L'individualisation reprend divers processus :

- La sélection dans un contenu qui est la possibilité offerte à l'élève de choisir ce qui l'intéresse dans un contenu donné.
- L'adaptation du cheminement pédagogique en fonction des réponses apportées par les différents élèves tout au long de leur parcours d'apprentissage dans le didacticiel.
- La différenciation du temps passé selon la vitesse d'apprentissage des différents élèves.

Cette individualisation a pour conséquence un glissement vers une situation d'autoformation où les facteurs socio-psychologiques que sont les interactions élèves/enseignant ou élèves/élèves jouent un rôle beaucoup moins important que dans le cas d'un enseignement plus classique.

1.1.2. L'interactivité

L'interactivité est une des caractéristiques les plus importantes des didacticiels. Cependant, contrairement à ce qu'on lit souvent, elle n'est pas le propre de l'E.A.O. En effet, rien n'est plus interactif qu'un cours « classique » où les élèves sont libres d'interrompre le professeur, de l'interroger ou même d'émettre leur propre avis. Notons simplement que, parmi tous les médias, l'ordinateur est celui qui autorise la plus grande interactivité.

Mais nous devons, à ce niveau, considérer une nouvelle approche de cette notion d'interactivité. Nouvelle approche causée par la nécessité de prévoir et d'implémenter un maximum de manifestations interactives.

Ces manifestations interactives peuvent se représenter par ce que les auteurs de l'ouvrage « Concevoir et utiliser un didacticiel » appellent les « unités d'interaction ».

L'unité d'interaction est la structure de base de l'échange d'informations entre l'étudiant et l'ordinateur. Tout didacticiel est composé d'une suite d'unités d'interaction. Le devoir du concepteur est de maîtriser cette structure et d'assurer le chaînage des unités entre elles.

La structure d'une unité d'interaction est assez simple. On trouvera à la figure 1.1. un schéma de cette structure. Elle est composée de quatre phases :

a) La sollicitation machine qui peut être soit une question posée par la machine, soit l'initiation de simulation dans le cas d'un didacticiel conçu sous forme de simulation.

b) L'entrée des données par l'élève en fonction de la sollicitation. Cela peut être la réponse à une question posée ou l'entrée des données dans le cas d'une simulation.

c) La réaction de la machine aux données entrées par l'élève. Cette réaction se manifeste par un commentaire de la réponse donnée à une question, ou bien par l'exécution d'un ordre donné.

d) La décision de branchement vers l'unité d'interaction suivante ou vers la même unité d'interaction si la réponse de l'élève est incorrecte. Dans ce cas, soit l'élève est conduit directement vers cette unité d'interaction, soit il décide lui-même du branchement à effectuer.

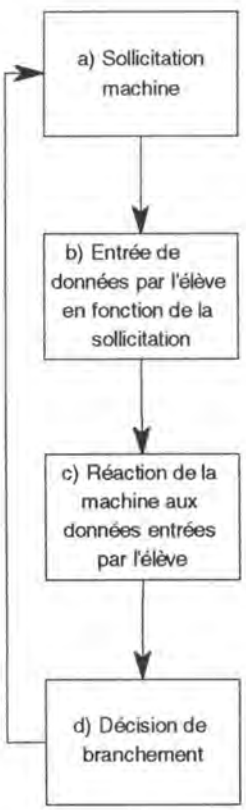


Figure 1.1. : Structure de l'unité d'interaction.

Cette structure peut être enrichie si le concepteur décide d'introduire entre les deux premières phases des fonctions telles que :

- l'appel d'aide contextuelle.
- l'appel d'une fonction de calcul.
- le refus par l'élève de la sollicitation machine.
- le retour en arrière.

Un didacticiel est d'autant plus interactif qu'il contient plus d'unités d'interaction. Une unité est dite non-interactive lorsqu'elle ne présente pas les deuxième et troisième phases. C'est le cas, par exemple, lors de la présentation d'un texte à lire ou d'un graphique à observer à l'écran.

Il est clair que, dans le cadre d'un didacticiel, cette structure seule ne suffit pas. Il faut également que l'unité d'interaction soit porteuse d'une valeur pédagogique, valeur qui est étroitement liée au niveau des efforts intellectuels provoqués chez l'élève ainsi qu'à l'adaptation du didacticiel aux réponses données par l'étudiant.

1.2. Le didacticiel et sa conception

Un didacticiel est un logiciel d'E.A.O. visant à enseigner une matière donnée. Comme pour tout logiciel, il existe des méthodologies de conception d'un didacticiel. Nous nous intéresserons ici à la méthodologie proposée dans l'ouvrage « Concevoir et utiliser un didacticiel » [BESN88]. La raison de ce choix est que la méthodologie proposée nous semble être la plus proche d'une des méthodologies présentée dans le cours « Méthodologie de Développement de Logiciel » [DUBOIS93] du professeur Eric Dubois : le modèle de la cascade.

Le modèle de la cascade (représenté à la figure 1.2.) se décompose en cinq parties. Le concepteur part du cahier des charges duquel il déduit les spécifications qui sont une description abstraite du système informatique. Ces spécifications représentent donc une analyse du cahier des charges. Le concepteur se base alors sur ces spécifications pour arriver à la conception qui est une description abstraite d'une solution au problème. Suit alors l'implémentation ou le codage dans un langage de la solution. La dernière étape est une série de tests qui peut éventuellement ramener le concepteur à une étape précédente.

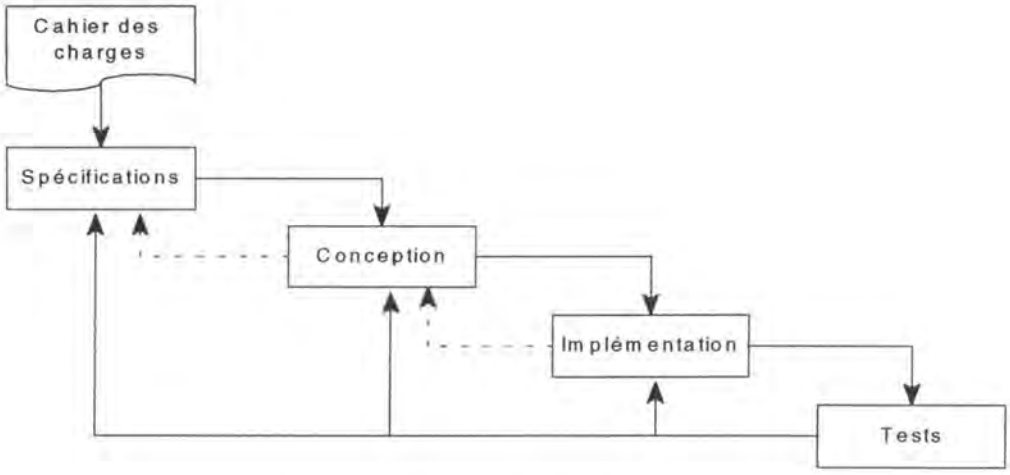


Figure 1.2. : Le modèle de la cascade.

Les deux principales caractéristiques de cette approche sont, premièrement, que chaque étape est décrite dans le langage naturel et, deuxièmement, qu'il est possible d'effectuer un retour en arrière après chaque étape suite à d'éventuelles erreurs. Il y a donc vérification après chaque étape.

La figure 1.3. présente un schéma de la méthodologie reprise dans l'ouvrage « Concevoir et utiliser un didacticiel » [BESN88] sous forme d'étapes distinctes. Chacune de ces étapes sera ensuite décrite individuellement et mise en parallèle avec une phase du modèle de la cascade présenté plus haut. Nous nous efforcerons également de faire le rapprochement avec les différentes phases de développement de notre application.

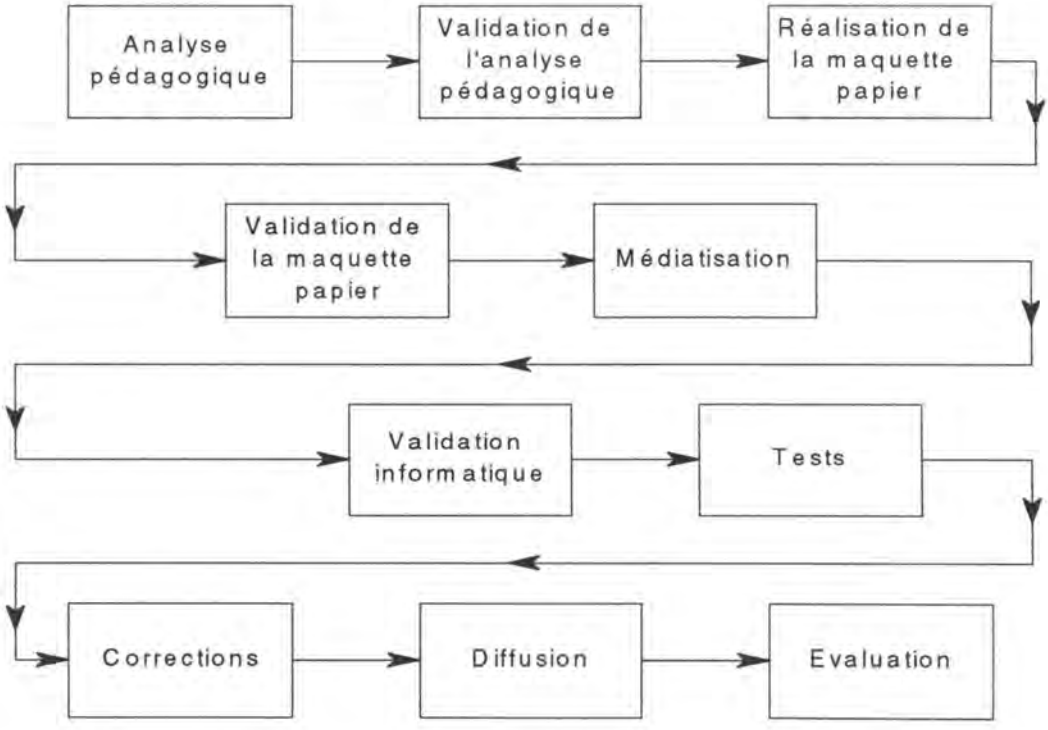


Figure 1.3. : La chaîne de production d'un didacticiel.

1.2.1. L'analyse pédagogique

L'analyse pédagogique constitue une étape importante de la démarche de conception d'un didacticiel. Notons que c'est le seul cas où on la trouve dans le monde des logiciels. C'est la conséquence naturelle de l'objet particulier du didacticiel : **l'enseignement**.

Généralement, cette étape se fait en équipe constituée d'un commanditaire, d'un responsable pédagogique et d'un concepteur médiatique. L'intérêt d'un tel travail en équipe est mis en évidence par la diversité des compétences nécessaire à la réalisation d'un produit de qualité.

L'analyse pédagogique a pour but de déterminer deux choix :

- Dire où on va.
- Dire comment on y va.

a) **Dire où on va**

La première chose à préciser à ce niveau est l'objectif pédagogique (ou objectif principal) à atteindre par les élèves. L'objectif pédagogique peut être défini comme l'opération que l'élève sera capable d'accomplir à l'aboutissement de son apprentissage. Par exemple, dans notre cas, l'élève devra être capable d'expliquer le fonctionnement général du transit des données dans Internet.

L'objectif principal servira donc :

- à évaluer l'efficacité de l'apprentissage car il représente le but à atteindre par l'élève.
- à motiver l'élève en lui montrant à quoi il doit arriver.
- à structurer le contenu à transmettre. En effet, le concepteur peut, en partant de l'objectif principal, déterminer les objectifs secondaires à maîtriser avant d'être capable d'atteindre l'objectif principal.

Une fois que l'objectif pédagogique a été déterminé, il reste à définir les principales caractéristiques des élèves. Cela se fera au cours d'une analyse approfondie de la population cible et de l'environnement dans lequel cette population utilisera le didacticiel. Cette analyse aura pour but d'obtenir un maximum de renseignements qui permettront de concevoir un didacticiel adapté de façon optimale aux caractéristiques de la population cible.

On trouvera dans le tableau 1.1. la liste des principaux facteurs caractérisant la population cible. Chacun de ces facteurs est accompagné d'une définition sommaire et des décisions qu'ils engendrent lorsqu'ils sont connus.

	Définition	En a-t-on connaissance ?	Décisions à prendre
Structure cognitive	* Organisation de l'ensemble des connaissances dont une personne dispose à un moment donné	* Non, on disposera éventuellement d'information partielle sur des points précis mais pas sur la structure générale	* Aucune
Stade du développement cognitif	* Niveau de maturation atteint par les élèves * Caractéristiques d'un stade de développement	* De façon très aléatoire en milieu scolaire. * Pas du tout en milieu professionnel adulte	* Fixer en pré-requis qu'un certain stade est atteint. Ceci ne sera pas toujours vrai.
Capacité intellectuelle	* Niveau général de l'intelligence telle qu'elle est mesurée par des tests	* En formation initiale, on pourrait déterminer ce facteur. * En formation professionnelle, non (coût)	* Fixer en pré-requis que ces capacités existent. Ici non plus, ce ne sera pas toujours vrai.
Facteurs motivationnels et attitudes	* Ce qui influence le désir de faire quelque chose et/ ou d'atteindre un objectif	* Des entretiens ou des enquêtes préalables avec la population cible permettent d'obtenir de tels renseignements	* Utiliser les éléments qui motivent la population. * Tenter de motiver la population si l'apprentissage fait l'objet de réticences
Facteurs socio-psychologiques	* Interactions élèves/ élèves, élèves/enseignants	* Ce facteur est peu intéressant dans notre cas car il s'agit ici d'une situation d'autoformation (cfr 1.1.1 L'individualisation).	
Facteurs didactiques au sens restreint	* Démarche et environnement pédagogique	* Des enquêtes auprès de la population cible permettront de ramener de précieux renseignements.	* Déterminer une ou des démarches pédagogiques
Circonstances réelles (environnement)	* Réalité de la situation dans laquelle se déroulera l'apprentissage	* Des enquêtes de terrain préalables fourniront ce genre de renseignements.	* Par exemple, éviter le son dans une salle commune. * Tenir compte des plages horaires disponibles pour l'apprentissage

Tableau 1.1. : Facteurs utiles de la population cible.

b) **Dire comment on y va**

Après avoir défini où aller, il est important de préciser comment y aller. On parlera ici de stratégie pédagogique. Stratégie dans laquelle on essaiera de combiner de façon optimale les ressources disponibles afin d'arriver au but fixé. Pour ce faire, 4 grands types de décision sont à aborder :

- Définir avec précision le matériel disponible.
- Choisir et organiser le contenu.
- Préciser la ou les démarche(s) pédagogique(s).
- Fixer des formes d'interaction.

A chacun de ces types de décision sont associées une ressource et une série de décisions à prendre. Ces ressources sont respectivement pour les 4 types de décision : la ressource matériel, la ressource contenu, la ressource élève et la ressource unité d'interaction. Détaillons chacune de ces ressources.

i. *La ressource matériel*

On entend par ressource matériel l'ordinateur, le rétroprojecteur, le projecteur de diapositives, le tableau noir, etc... C'est en fait l'environnement matériel dans lequel l'apprentissage aura lieu. Nous verrons plus loin que, dans le cadre de notre travail, l'environnement matériel se limitera à l'ordinateur.

Les principales décisions à prendre à ce niveau sont :

- La répartition des messages, des activités parmi les médias qui sont disponibles. Dans notre cas, il n'y aura pas de décision à prendre à ce niveau puisque nous ne disposerons que d'un seul média, à savoir l'ordinateur.

- La manière de gérer l'écran, d'organiser l'utilisation des couleurs, etc... Bref de prendre des décisions en ce qui concerne l'interface homme/machine. Nous ne développerons pas plus avant ce point qui fait l'objet du chapitre suivant.

Cette ressource représente la partie médiatique de la stratégie pédagogique.

ii. *La ressource contenu*

Le contenu est l'ensemble de la matière à transmettre pour atteindre l'objectif pédagogique fixé. Les principales décisions à prendre à ce sujet sont :

- La façon d'organiser et de hiérarchiser les diverses notions à transmettre afin de faciliter au maximum l'apprentissage.

- Le rejet de certaines notions, le maintien d'autres notions afin de ne pas alourdir le didacticiel.

Comme nous l'avons vu précédemment, c'est l'objectif pédagogique final qui influencera le plus le concepteur dans sa manière d'organiser le contenu, ainsi que dans ses choix de rejet ou de maintien de certaines notions. Par exemple, nous nous contenterons dans notre travail d'aborder les notions de base du routage dans Internet : à savoir l'existence et l'utilisation des tables de routage, l'utilisation des passerelles, etc. Le didacticiel développé n'est donc pas un produit fini en soi mais servira de base pour des extensions destinées à présenter des notions supplémentaires du domaine des télécommunications. Nous renvoyons le lecteur au chapitre 5 (consacré à la description de l'application développée) pour plus de détails.

Cette ressource est la partie didactique de la stratégie pédagogique.

iii. *La ressource élève*

C'est à ce niveau que l'analyse de la population cible sera utilisée. La ressource élève représente en effet les caractéristiques de la population cible que nous avions citée dans le tableau 1.1. Cette ressource amène le concepteur à définir une (des) démarche(s)

pédagogique(s) afin de permettre à l'élève d'atteindre l'objectif mais aussi afin de le motiver. Les principales décisions à prendre ici sont :

- Choisir l'activité à proposer à l'élève afin qu'il atteigne l'objectif pédagogique. C'est la dimension cognitive de la démarche pédagogique.

- Déterminer le déroulement des activités dans le didacticiel. Développer un scénario, une mise en scène afin de motiver l'élève dans son apprentissage. C'est la dimension affective de la démarche pédagogique.

Nous verrons que, dans notre cas, l'activité proposée est le pilotage d'une simulation graphique du transit des paquets au sein d'une interconnexion de réseaux simplifiée. Le déroulement des activités est totalement déterminé par l'élève : c'est lui qui décide à sa guise de ce qu'il veut faire. Les circonstances pratiques dans lesquelles se déroulera l'apprentissage seront décrites en détail au chapitre 5 (point 1.2.3.) de ce mémoire.

iv. *La ressource unité d'interaction*

Nous avons déjà longuement parlé des unités d'interaction. Nous nous bornerons donc à présenter les principales décisions que le concepteur doit prendre à ce niveau :

- Choisir les formes d'interaction pour les activités proposées. Les principales formes d'interaction sont les formes à jugement de réponse, formes dans lesquelles l'élève doit fournir des réponses à des questions (à choix multiples ou non) et les formes à simulation qui sont des formes présentant la simulation d'un modèle sous-jacent (c'est le cas du didacticiel que nous avons développé).

- Articuler les unités d'interaction entre elles afin d'assurer un suivi logique de l'apprentissage.

C'est la partie forme et design de la stratégie pédagogique. Nous en reparlerons plus en détail dans le chapitre 2 qui est consacré à cette dimension.

La stratégie pédagogique peut à présent être définie comme l'articulation entre les quatre dimensions relatives aux ressources que nous venons de développer, à savoir la dimension médiatique, la dimension didactique, la démarche pédagogique et la dimension forme et design.

Cette première étape que représente l'analyse pédagogique peut être considérée comme l'équivalent en E.A.O de la partie « spécifications » du modèle de la cascade présenté au début de ce chapitre.

1.2.2. La validation de l'analyse pédagogique

Cette étape consiste en la présentation de l'analyse pédagogique au commanditaire par le concepteur. C'est à ce niveau que le commanditaire émettra ses remarques et/ou ses conseils pour le travail fourni.

Le concepteur a tout intérêt à ne pas négliger cette phase s'il ne veut pas se voir refuser le travail des étapes suivantes. En effet s'il a divergé des idées du commanditaire dès l'analyse

pédagogique, tout ce qui suivra ne saurait être correct aux yeux de celui-ci. Un autre cas de figure qui peut également se présenter est celui où le concepteur n'est pas spécialiste des démarches pédagogiques. Dans ce cas, il pourra profiter des critiques de personnes plus expérimentées et se baser dès lors sur une analyse plus profonde et plus solide.

Dans notre cas, la validation de l'analyse pédagogique s'est faite lors d'une réunion avec le commanditaire du didacticiel. Cette réunion a permis de déceler certaines lacunes et de prendre les décisions adéquates pour corriger ces lacunes. Le lecteur trouvera au chapitre 7 les choix faits et les difficultés rencontrées lors du développement du didacticiel.

1.2.3. La réalisation de la maquette papier

La réalisation de la maquette papier ou, en d'autres termes, la conception du didacticiel sur papier est une des étapes les plus importantes et les plus longues du cycle de conception du didacticiel. C'est en fait la phase durant laquelle le concepteur conçoit le didacticiel en réalisant une maquette papier de celui-ci.

C'est ici que le concepteur décrit complètement le didacticiel. Il définit les diverses fonctionnalités que l'on rencontrera dans le logiciel ainsi que leur(s) effet(s). Il déterminera le chaînage entre les différentes unités d'interaction. Il conçoit la présentation visuelle du didacticiel c'est-à-dire qu'il créera tous les écrans qui apparaîtront tout au long du processus d'apprentissage. Nous renvoyons le lecteur au chapitre 5 de ce mémoire pour une description détaillée de la maquette papier de notre didacticiel.

Enfin, il définit une découpe en modules du didacticiel. Il crée donc une architecture logicielle. Or, comme il est précisé dans le cours de MDL du professeur Eric Dubois [DUBOIS93], deux approches sont possibles à ce niveau. Le concepteur peut procéder à une découpe en modules pour arriver soit à une architecture physique, soit à une architecture logique.

L'architecture physique peut être définie comme « *une solution efficace, correcte et exécutable sur une machine « réelle ». Mais avant tout les modules de cette architecture sont des unités de travail pour la MACHINE* ». L'architecture logique peut également être définie comme une solution efficace et correcte mais elle est exécutable sur une machine « abstraite ». C'est-à-dire qu'avant tout les modules sont ici des « *unités de travail pour l'utilisateur* ».

Dans le cadre de ce travail, nous adopterons la deuxième approche. Non pas qu'elle soit plus efficace à l'exécution mais elle facilite par définition la division du travail, la maintenance et la documentation. Nous n'entrerons pas plus en détail ici. Notons simplement que nous appliquerons cette découpe en modules à notre didacticiel au chapitre 5 : « Le didacticiel TcpIp ».

La réalisation de la maquette papier est l'équivalent de la phase de conception du modèle de la cascade présenté plus haut.

1.2.4. La validation de la maquette papier

La validation de la maquette papier est, au même titre que la validation de l'analyse pédagogique, une étape de « sécurité ». Il est prudent, en effet, de cautionner cette maquette par le commanditaire avant de commencer l'encodage proprement dit. Cette précaution évitera donc de lourdes modifications ultérieures en cas de désaccord avec le commanditaire.

Le commanditaire pourra notamment visualiser les écrans à ce niveau et, s'il le juge nécessaire, les critiquer. On trouvera au chapitre 7 de ce travail les différents choix faits et les difficultés rencontrées tout au long du développement du didacticiel.

Cette étape pourrait être la cause du retour en arrière de la conception vers les spécifications dans le modèle de la cascade.

1.2.5. La médiatisation

La médiatisation est en fait l'implémentation de la maquette papier sur machine. C'est donc le codage du didacticiel en adaptant les messages en fonction du ou des médias utilisés. Cette étape sera d'autant plus aisée que la maquette papier aura été bien réalisée et que l'on disposera de bons outils.

Nous ne nous étendrons pas plus avant sur ce sujet qui n'est finalement que la traduction de la maquette papier dans un langage donné. Il est évident que cette étape est l'équivalent de l'étape d'implémentation du modèle de la cascade.

1.2.6. La validation informatique

C'est une étape purement informatique. En effet, la validation informatique consiste en la correction d'éventuelles erreurs de codage (bugs). Mais elle ne constitue pas pour autant une étape mineure. Un didacticiel très bien conçu pédagogiquement n'arrivera qu'à énerver et « démotiver » l'élève s'il est truffé d'erreurs.

1.2.7. Les tests de forme

Avant d'utiliser de façon systématique le didacticiel, celui-ci doit obligatoirement être testé sur un échantillon de la population cible ainsi qu'auprès de spécialistes de la matière enseignée par ce didacticiel.

Ces tests ne visent aucunement à faire découvrir de nouveaux problèmes de fond (qui ont logiquement été éliminés lors de la validation de la maquette papier) mais bien à mettre en évidence des problèmes de forme et des détails que même le concepteur le plus pointilleux ne pourrait prévoir en totalité.

Les deux étapes « validation informatique » et « tests de forme » sont équivalentes à la phase de tests du modèle de la cascade.

1.2.8. Correction

La correction est l'étape qui suit logiquement l'étape des tests. Même si cette correction ne touche que des problèmes mineurs, elle peut prendre du temps. C'est pourquoi il ne faut pas la négliger.

1.2.9. La diffusion

La diffusion est l'insertion du didacticiel au sein de la population à laquelle il est destiné. Dans le cas de logiciel destiné à la commercialisation, c'est à ce niveau que le logiciel est distribué dans les chaînes de vente.

1.2.10. L'évaluation

L'évaluation est un contrôle de qualité sur le didacticiel. L'évaluation est une étape difficile à mener de façon critique. D'une part, elle consomme du temps si on veut faire évaluer efficacement le didacticiel. D'autre part, il est difficile d'être juge et partie à la fois. Nous voulons dire par là qu'il n'est pas aisé à un concepteur d'évaluer de façon critique son propre produit. En effet, même si le concepteur est de bonne foi, comme il est le concepteur, il connaît le produit parfaitement et ne se posera pas toutes les questions qu'un usager quelconque se posera. Pour notre part, nous essaierons de procéder à une évaluation de notre produit en utilisant une grille d'analyse créée par des spécialistes de l'E.A.O. Le lecteur trouvera les résultats de cette évaluation au chapitre 8 de ce travail. Nous n'avons malheureusement pas eu le temps matériel d'effectuer des tests auprès de tiers.

Dans le cadre du projet qui nous intéresse, certaines de ces étapes ont été utilisées de manière différente. En effet, l'application développée n'étant pas destinée à la commercialisation, la partie commerciale de la phase de diffusion (1.2.9) n'a pas été abordée.

De plus, les phases de réalisation de la maquette papier (1.2.3) et de médiatisation (1.2.5) ont été intimement liées lors du développement de l'application. Celle-ci étant une simulation graphique, il nous a semblé plus opportun de la développer par "morceaux" : cette technique nous permettait alors de valider ces morceaux avant de poursuivre plus avant le développement. Cela a pour conséquence directe que les phases de la validation de la maquette papier (1.2.4) et de validation informatique (1.2.6) furent également liées. Nous reparlerons plus en détails de l'application de ces phases à notre projet dans le cadre du chapitre 5 de ce travail.

Maintenant que nous avons vu ce qu'était l'E.A.O. et que nous avons présenté diverses notions qui lui sont relatives, passons à un projet existant dont le but est d'encourager l'utilisation de l'ordinateur dans l'enseignement : le projet COLOS.

2. Le projet COLOS

COLOS (Conceptual Learning Of Sciences) est un consortium fondé en 1988 par Zvonko Fazarinc, employé aux HP Laboratories de Palo Alto (USA), et supporté principalement par la société Hewlett-Packard. COLOS est actuellement composé d'équipes de 10 universités disséminées dans 6 pays européens (Allemagne, Grande Bretagne, France, Pays-Bas, Italie et Espagne). La description que nous en donnons ici est basée sur le document HTML [MULLER95] écrit par Daniel Muller, professeur agrégé de l'Ecole Centrale de Lyon.

Le but principal de ce consortium est d'encourager et de coordonner le développement de méthodes d'enseignement afin d'améliorer la compréhension et la connaissance de concepts fondamentaux de différents domaines scientifiques et techniques. Mais en plus, une attention spéciale est accordée afin de fournir une approche intuitive et qualitative.

A l'heure actuelle, les actions de COLOS sont :

- le développement d'applications d'E.A.O.
- le développement d'outils génériques pour l'E.A.O.
- l'évaluation de didacticiels.
- le développement de cours basés sur l'E.A.O.

Afin de coordonner leur travail, les différentes équipes se rencontrent tous les six mois dans un des sites contenant une équipe. Mais la collaboration entre les équipes ne se limite pas là : elles communiquent intensivement toute l'année en utilisant les possibilités offertes par le réseau Internet. De plus un serveur COLOS est maintenu régulièrement à jour et peut fournir à toutes les équipes toute l'information qu'elles désirent (applications, outils, documentation, etc...).

2.1. La philosophie COLOS

La philosophie de COLOS est de promouvoir un apprentissage des concepts et une compréhension plus profonde dans les domaines des sciences et de la technologie. Les membres de COLOS utilisent intensivement à cet effet le potentiel didactique des technologies modernes de communication.

Cette philosophie se base sur le fait que, contrairement à la vie de tous les jours où les personnes interagissent avec des mécanismes naturellement visibles et facilement manipulables, l'enseignement des sciences telles que les mathématiques, la physique ou la chimie se base sur des concepts abstraits. Et que ces concepts sont eux-mêmes manipulés par des opérateurs encore plus abstraits tels que les théorèmes, les axiomes, etc. C'est une des raisons pour laquelle même si des étudiants sont aptes à résoudre des problèmes standards, beaucoup ne comprennent pas réellement certains principes de base.

De nos jours, les ordinateurs offrent aux professeurs un moyen de contourner cette abstraction. La puissance de ces ordinateurs et leur interface permettent la représentation sur un écran des principes de la nature. Dès lors l'étudiant, en observant, commence par comprendre le phénomène représenté sans pour autant connaître l'existence des équations décrivant ces phénomènes. Et si, par la suite, on lui présente ces équations, il les prend pour ce qu'elles sont : « *un modèle représentant la réalité* ». Et non, comme c'est souvent le cas, comme des lois que suit la nature.

Afin de résoudre ces problèmes, les membres de COLOS utilisent quelques principes didactiques :

- L'utilisation de simulations en supplément de l'enseignement classique, et cela principalement dans la présentation de modèles ou de concepts qui sont fréquemment mal compris par les étudiants.
- La réduction de la charge mathématique pour les débutants. Pour ce faire, les membres de COLOS utilisent des animations graphiques et, si possible, des représentations en 3 dimensions. Une fois la compréhension du phénomène acquise, on peut alors aller vers une abstraction mathématique.

L'utilisation de ces 2 principes doit se faire le plus souvent possible pour décrire des processus naturels ou techniques.

2.2. Quelques scénarios d'E.A.O.

COLOS a adopté plusieurs types de scénarios pour appliquer l'E.A.O. Nous entendons ici par scénario la manière d'utiliser les médias afin de présenter un cours.

2.2.1. La conférence formelle

Ce scénario regroupe une station de travail et un projecteur vidéo ou une série de moniteurs. Le professeur utilise l'ordinateur pour exposer interactivement des phénomènes complexes sous forme de simulation. Une haute interactivité et l'exécution des ordres en temps réel sont deux facteurs essentiels. Ils permettent en effet au professeur de réagir immédiatement à toute réaction de la part de ses étudiants.

Une amélioration possible serait de fournir aux étudiants des périphériques d'entrée tels que la souris afin de leur permettre d'interagir directement avec le professeur sur la représentation du phénomène qui leur est donnée.

2.2.2. L'enseignement interactif

Ce scénario nécessite une infrastructure plus importante que le précédent. L'enseignement interactif regroupe une série de stations de travail sur lesquelles travaillent les étudiants. Le professeur peut soit utiliser un tableau classique, soit un vidéoprojecteur pour présenter son propre écran. Ou encore, il peut utiliser des outils de partage (tels que SharedX) afin d'afficher le contenu de son propre écran sur les écrans des étudiants.

Ce scénario permet au professeur d'encore diriger le cours mais force les étudiants à être plus actifs que dans le cas de la conférence formelle. Ce type de scénario est déjà utilisé au laboratoire d'électrotechnique de l'Ecole Centrale de Lyon.

2.2.3. L'étude individuelle

L'étude individuelle est un scénario dans lequel l'étudiant a accès à une station de travail en « self-service ». Le didacticiel utilisé par le professeur dans l'un des deux premiers scénarios est ici directement accessible par l'étudiant. Dans ce cas de figure, l'étudiant est libre d'évoluer à son propre rythme ainsi que de s'intéresser aux concepts qui lui paraissent importants.

2.2.4. Le travail du professeur

Ce scénario, contrairement aux autres, n'est pas un scénario d'enseignement proprement dit. C'est un scénario de préparation d'un cours ou d'une simulation. Le professeur devrait pouvoir créer sa propre simulation pour illustrer son cours en utilisant des outils graphiques prédéfinis. Une telle préparation ne devrait pas prendre plus de la moitié du temps que prendra le cours en lui-même. De tels outils n'étant pas encore disponibles

aujourd'hui dans le cadre du projet COLOS, le professeur doit encore faire appel aux spécialistes en la matière que sont les programmeurs spécialisés dans l'E.A.O.

2.3. Quelques didacticiels

Nous présenterons brièvement dans ce point deux didacticiels développés par les chercheurs de l'Ecole Centrale de Lyon. Les caractéristiques principales de ces didacticiels sont la haute interactivité (réponse en temps réel aux événements, etc...) et l'interface particulièrement soignée.

Nous ne décrirons pas en détail les notions présentées dans ces didacticiels, cela sortant du cadre de ce travail. Nous nous bornerons à décrire brièvement l'usage qui est fait de ces applications ainsi que leur interface.

2.3.1. mField

Cette application a pour but de faire comprendre comment on peut générer des champs tournants à l'intérieur d'une machine électrique tournante. L'interface dont on trouvera une représentation à la figure 1.4. a été délibérément conçue simple afin de faciliter l'utilisation de l'application par des personnes qui ne sont pas familières à l'environnement informatique.

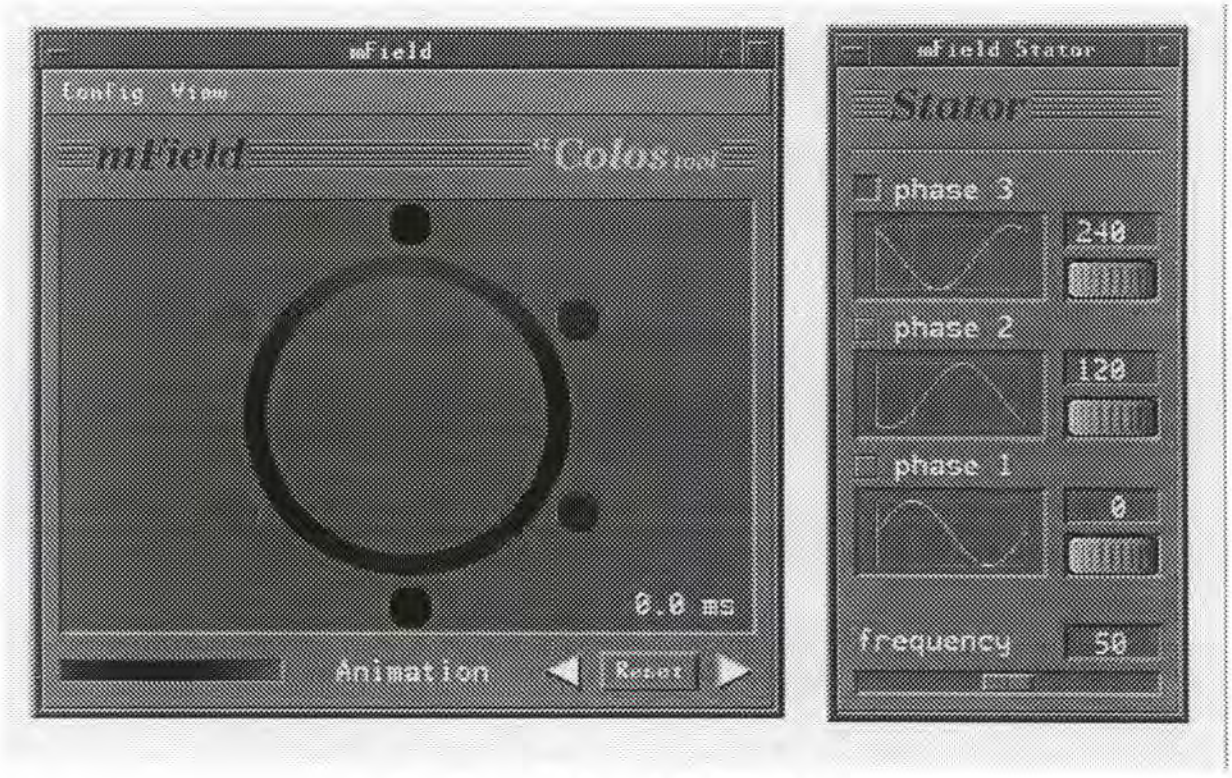


Figure 1.4. : L'interface de mField.

Une attention toute particulière a été accordée au niveau de l'ergonomie de l'interface homme/machine. Un exemple particulier de cette attention est représenté par les « boutons-molette » que l'on peut voir dans les fenêtres de contrôle de l'application. Ce type de bouton permet à l'étudiant de deviner immédiatement l'action qu'il peut effectuer sur ce type de bouton : « le faire tourner ».

Une autre qualité de cette application qui n'est pas décelable sur un schéma fixe est la haute interactivité qu'elle propose. Elle permet à l'étudiant de voir en temps réel les conséquences des actions qu'il effectue. Notons encore que, dans le but de faciliter l'utilisation de l'application par l'étudiant, toutes les actions hautement interactives ont été rendues accessibles directement à partir des fenêtres de contrôle, tandis que d'autres actions (telles que les modifications de configurations) sont accessibles par la barre de menu classique.

2.3.2. mWave

Cette application permet d'étudier les champs magnétiques ou électriques à l'intérieur d'un conducteur d'ondes de longueur infinie et de section ronde ou carrée. Nous ne présentons cette application que dans un but comparatif à l'application mField. Notons simplement que cette application est destinée à des personnes ayant déjà un certain nombre de connaissances en électricité. C'est pourquoi l'interface de cette application est plus complexe et plus fournie que l'interface de mField.

Remarquons encore qu'un soin particulier a été apporté à la conception de cette interface. La façon dont les boutons sont définis engendre bien dans l'esprit de l'étudiant l'action que celui-ci peut effectuer sur ces boutons. Et l'effet de ces actions est porté en temps réel à l'écran, ce qui est indéniablement une qualité importante de l'application.

3. L'Institut d'Informatique dans COLOS

L'Institut d'Informatique des F.U.N.D.P. a fait activement partie du consortium COLOS. L'équipe de Namur s'était donnée comme but particulier l'E.A.O. dans le domaine des télécommunications. Malheureusement, le projet dut être abandonné par manque de moyens humains.

Toutefois, l'Institut a renoué cette année des liens avec l'équipe de l'Ecole Centrale de Lyon sous forme d'un contrat de collaboration. Les termes de ce contrat sont que l'institut fournira au mois de juin 1995 un didacticiel dans le domaine des télécommunications, plus précisément sur Internet.

Nous pensons être l'écho de tout le monde en espérant que cette collaboration ne s'arrêtera pas là et que le projet débuté cette année sera continué et amélioré dans les années à venir.

Chapitre 2 :
Des règles d'interface homme-machine pour l'E.A.O.

Que ce soit lors du développement d'un didacticiel ou d'un logiciel traditionnel, la conception d'une interface homme-machine est une des dimensions les plus importantes de la phase de développement du produit. Il existe à l'heure actuelle un grand nombre de règles appelées règles ergonomiques qui ont pour but de guider le concepteur dans le développement d'une interface homme-machine. Bien que toutes ces règles soient applicables à un didacticiel, l'E.A.O. en tant que discipline utilise plus particulièrement quelques-unes de ces règles en insistant sur leur connotation pédagogique.

Le but de ce chapitre est de présenter les notions de base relatives à la démarche de conception d'une interface homme-machine en général, pour ensuite étendre ces notions vers des règles de conception de l'interface d'un didacticiel en particulier.

1. Des règles de conception d'une interface homme-machine

Nous avons parlé plus haut de règles ergonomiques visant à aider le concepteur dans sa tâche de développement d'une interface homme-machine; elles ont pour but de faire respecter un ou plusieurs critères ergonomiques lors de la conception de l'interface homme-machine. Nous nous baserons pour cette première partie sur l'ouvrage « Guide ergonomique des interfaces homme-machine » [VANDERD94] de Jean Vanderdonckt, assistant à l'Institut d'Informatique des F.U.N.D.P.

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 » [VANDERD94]. Ce sont donc ces critères qui sont à la base de tous les choix en matière de conception d'une interface homme-machine. Les critères ergonomiques retenus à l'heure actuelle par les principales personnalités de ce domaine sont au nombre de 8 et constituent la liste suivante :

- la compatibilité;
- la cohérence;
- la charge de travail;
- l'adaptabilité;
- le contrôle du dialogue;
- la représentativité;
- le guidage;
- la gestion des erreurs.

Nous essaierons autant que possible lors de la présentation de ces critères d'établir un lien avec l'E.A.O. en général et avec notre didacticiel en particulier.

1.1.1. La compatibilité

« Une interface homme-machine est qualifiée de compatible si et seulement si le (re)codage d'informations et de tâches du monde réel en données et actions du système est réduit. » [VANDERD94]. 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 en fait la cohérence de l'interface homme-machine avec l'environnement extérieur de l'application.

Le but poursuivi par le respect de ce critère est de réduire au maximum le besoin de traduire et d'interpréter l'information réelle en données du système. Par conséquent, l'utilisation d'une application présentant une interface compatible sera facilitée par l'absence du besoin de devoir continuellement se demander ce qui est représenté à l'écran.

Par exemple, notre didacticiel sera d'autant plus compatible qu'il respectera les habitudes comportementales de l'utilisateur ou qu'il fournira une représentation la plus proche possible du monde réel.

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. »[VANDERD94]. Un utilisateur perçoit d'autant mieux les données et exécute d'autant mieux les actions que celles-ci sont présentées 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. Ce critère peut donc être scindé en deux dimensions : la <u>cohérence intra-application</u> qui correspond à la cohérence à tous les niveaux au sein d'une même

application et la cohérence inter-application qui correspond à la cohérence à tous les niveaux entre deux ou plusieurs applications.

Le but poursuivi par le respect de la cohérence est de recourir toujours aux mêmes moyens pour arriver aux mêmes résultats dans des contextes similaires; la conséquence directe de cela est une standardisation de l'interface et des procédures qui permet à l'utilisateur de prévoir les résultats qu'il va obtenir en exécutant telle ou telle action.

Par exemple, notre didacticiel peut être qualifié de cohérent « intra-application » si tous les boutons graphiques de son interface ont le même type de représentation et si leur libellé ont tous la même construction grammaticale.

L'exemple typique d'une bonne cohérence inter-application est l'exemple des applications « Microsoft » qui présentent, entre autres, toujours la même structure de menus ou la même représentation générale.

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. » [VANDERD94]. L'utilisation d'une application est en effet d'autant plus efficace et agréable que la manipulation d'un nombre restreint de 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; le premier rôle de ce critère est extrêmement important dans le cadre du développement d'un didacticiel; en effet, si un didacticiel n'est pas efficace du point de vue de la charge de travail, il peut s'ensuivre que l'utilisateur utilise une grande partie de ses facultés mentales pour assimiler et comprendre le fonctionnement du didacticiel plutôt que de porter toute son attention sur le contenu du didacticiel qui représente réellement ce qu'il doit assimiler.

Dans notre cas, notre didacticiel pourra être qualifié d'efficace en charge de travail s'il assure une découpe de la tâche de l'utilisateur suffisamment fine pour que chaque action atomique ne lui prenne pas trop de temps ni trop d'attention; ce qui résulterait en une diminution de l'efficacité de l'apprentissage prodigué par le didacticiel.

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. » [VANDERD94]. Un utilisateur est d'autant plus enclin à utiliser une application et acquiert d'autant plus d'expérience que l'interface s'adapte aux différents contextes de travail. Une interface personnalisable, par exemple, traduit une adaptabilité importante de l'application correspondante.

Le respect de ce critère lors de la conception d'une interface a donc pour but d'offrir à l'utilisateur plusieurs voies pour accomplir sa tâche; l'adaptabilité ne concerne donc pas que les paramètres visuels de l'interface mais également les paramètres de fonctionnement de l'application. Un didacticiel ayant pour objet particulier l'enseignement, ce critère est

particulièrement important dans le cadre de l'E.A.O. car il permet d'offrir à l'apprenant une interface agréable qui motivera celui-ci lors de son apprentissage.

Nous verrons lors de la présentation de notre didacticiel que celui-ci permet à l'utilisateur de personnaliser une grande partie des paramètres de l'interface et que cette interface peut dès lors être qualifiée d'adaptable. De plus, notre didacticiel permet également à l'utilisateur de déterminer son propre rythme de travail, ce qui traduit une adaptabilité non pas du point de vue aspect visuel de l'interface mais du point de vue fonctionnement de l'application.

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. » [VANDERD94].

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 qu'il désirait exécuter cette action.

Comme nous le verrons plus loin, l'interface de notre didacticiel propose à l'utilisateur de piloter une simulation graphique. Dans cette tâche, l'utilisateur détermine lui-même à tout moment ce qu'il désire faire, c'est donc lui qui « a la main » en permanence. Cette observation permet de conclure que l'interface homme-machine de notre didacticiel est à contrôle explicite. Cette caractéristique a pour conséquence directe une meilleure adaptabilité de notre didacticiel puisque l'utilisateur pourra déterminer son propre rythme de travail.

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. » [VANDERD94]. Un utilisateur a en effet 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. Ce critère rejoint par là le critère de cohérence intra-application si ce n'est qu'à ce niveau l'accent est mis sur la signification même des différents codes utilisés.

Notre didacticiel présentant une simulation graphique, il fait peu usage du texte dans son interface; nous n'avons donc apporté qu'une attention relative à cette dimension. Néanmoins, nous avons apporté une attention accrue sur les diverses représentations graphiques utilisées afin que celles-ci facilitent la rétention des notions qu'elles représentent.

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. »[VANDERD94]. 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 façon optimale du point de vue de sa lisibilité. Ce critère joue également un rôle important dans le cadre du développement d'un didacticiel, c'est en effet lui qui assure à l'apprenant une utilisation aisée du didacticiel et lui permet ainsi de porter toute son attention sur la matière qu'il doit assimiler.

Reprenons l'exemple de notre didacticiel, celui-ci propose en permanence à l'utilisateur un guidage dans son utilisation du didacticiel. Ce guidage se fait soit sous forme d'une invitation à exécuter l'action logique suivante dans la détermination des paramètres de la simulation, soit sous forme d'un commentaire illustrant une simulation qui est en cours d'exécution. Nous pouvons donc qualifier notre didacticiel d'efficace en guidage.

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. » [VANDERD94]. Il n'est pas de chose plus exaspérante en effet qu'une application qui « plante » dès que l'utilisateur fait une erreur de manipulation. 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.

Le fait de respecter ce critère revient à éviter les erreurs autant que possible. Ce critère est également important dans le domaine de l'E.A.O. car le didacticiel le mieux conçu sur le plan pédagogique ne parviendra qu'à exaspérer l'utilisateur s'il est truffé d'erreurs.

Notre didacticiel, par exemple, a été conçu pour ne pas s'arrêter à la suite des erreurs de manipulation de l'utilisateur. Il serait malhonnête de dire que tous les cas de figure ont été prévus, mais nous espérons avoir éliminé la plupart de ceux-ci. Nous ne pouvons pas préciser la convivialité des corrections de ces erreurs par le didacticiel car celui-ci ne réagit pas du tout aux erreurs de manipulation de l'utilisateur. C'est-à-dire que lorsque l'utilisateur commet une erreur de manipulation (telle que, par exemple, vouloir sélectionner une troisième machine pour une communication qui n'en implique que deux), le didacticiel reste exactement dans l'état dans lequel il se trouvait avant l'occurrence de cette erreur. Nous pensons pouvoir néanmoins qualifier notre didacticiel d'efficace en gestion des erreurs.

1.2. Les règles ergonomiques

L'ensemble des critères ergonomiques que nous venons de décrire sont respectés lors de la conception d'une interface homme-machine par l'usage 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. » [VANDERD94]. 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.

1.2.1. Les différents types de règles ergonomiques

Le nombre des règles ergonomiques étant de plusieurs milliers et le but de ce mémoire n'étant pas de décrire en détail ces règles, nous nous contenterons de présenter brièvement chacune des 12 divisions citées ci-dessus avant de décrire quelques méthodologies possibles d'utilisation des règles ergonomiques.

- **La saisie des données**

Cette division reprend l'ensemble des règles ergonomiques régissant l'interface à développer pour assurer une saisie de données. Les règles reprises 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.

- **L'affichage de données**

Cette division reprend les différentes 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 que une heure ou une date, les règles spécifiant le format des écrans à utiliser pour l'affichage de tel ou tel type de données ou encore les règles précisant les modalités d'affichage d'objets interactifs composés.

- **Le dialogue**

Cette division comprend l'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 comment implémenter une pause dans le dialogue.

- **Le graphisme**

Cette division reprend les règles ergonomiques concernant tout ce qui est relatif au graphisme dans une interface. Ces règles constituent des conseils à suivre en ce qui concerne l'utilisation des caractères, l'utilisation de graphiques animés ou non, l'emploi des couleurs, l'emploi d'icônes ou de symboles, l'emploi d'hypertexte ou encore l'utilisation du multimédia.

C'est par exemple à ce niveau que l'on retrouvera toutes les règles spécifiant les caractéristiques graphiques que doit présenter un bouton.

- **Les moyens d'interaction**

On retrouvera dans cette division l'ensemble des règles ergonomiques qui régissent l'utilisation de tous les périphériques permettant une interaction avec la machine. Ces moyens d'interaction sont abordés selon trois dimensions : les moyens d'interaction de saisie comme le clavier, les codes à barre, etc... ; les moyens d'interaction d'affichage comme l'écran tactile, la souris, le « joystick », ... ; et les moyens d'interaction multimédia comme la voix, le gant sensoriel, etc.

- **Les styles d'interaction**

Cette division reprend toutes les 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 existants reprennent entre autres le langage de commande, le langage nature, la sélection de menu, le remplissage de formes, l'interaction graphique, etc.

- **Le guidage de l'utilisateur**

Cette division comprend l'ensemble des règles ergonomiques précisant comment organiser le guidage de l'utilisateur dans l'interface. Ces règles permettent entre autres de préciser comment gérer les erreurs ou encore comment valider ergonomiquement l'information fournie par l'utilisateur.

- **Les messages**

Cette division reprend l'ensemble des règles ergonomiques régissant l'organisation spatiale ou syntaxique des différents messages que peut fournir une interface en fonction du type de message (message de guidage, message d'aide, message d'erreur, etc.).

- **L'aide en ligne**

Cette division comprend les règles ergonomiques précisant comment concevoir et implémenter une aide en ligne du point de vue contenu, présentation ou accessibilité.

- **La documentation**

On retrouve dans cette division toutes les règles ergonomiques caractérisant la conception d'une documentation appropriée du produit développé. Cet ensemble de règles reprend des règles pour la conception d'une documentation en ligne comme pour la conception d'une documentation extérieure (le manuel de l'utilisateur).

- **L'évaluation**

La division « Evaluation » reprend les 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**

On retrouve dans cette division l'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 quant aux différents facteurs à analyser ou à ne pas oublier lors de la phase d'implémentation de l'interface de l'application développée.

Nous renvoyons le lecteur au « Guide ergonomique des interfaces homme-machine » [VANDERD94] pour des exemples concrets de règles appartenant à ces différentes divisions.

1.2.2. L'utilisation des règles ergonomiques

Le nombre des règles ergonomiques est tellement important (plusieurs milliers) qu'il est matériellement impossible au concepteur d'une interface de toutes les prendre en compte; c'est pourquoi une taxonomie arborescente de ces règles dont les 12 divisions présentées plus haut constituent le premier niveau a été créée [VANDERD94]. Cette taxonomie permet d'établir une classification des règles ergonomiques en fonction des éléments de l'interface auxquels elles se réfèrent. Lorsqu'on désire dès lors s'intéresser à une dimension particulière de l'ergonomie d'une interface homme-machine, il suffit de rechercher dans la classification des règles la section correspondante et d'appliquer les règles qui s'y trouvent répertoriées.

Notons que ces règles ergonomiques peuvent être utilisées dans deux buts diamétralement opposés : elles peuvent être utilisées **comme une base** lors de la conception d'une interface ou bien être utilisées **comme une référence** lors d'une analyse et d'une évaluation d'une interface existante.

Dans le premier cas, le travail du concepteur est de concevoir une interface homme-machine pour une application dont l'usage est de faire réaliser une tâche par un utilisateur dans un mode séquentiel ou asynchrone. Cette interface est développée dans un style d'interaction donné en faisant appel à des objets interactifs (boîte de dialogue, fenêtres, ...) qui sont exploités par des dispositifs physiques (les moyens d'interaction). Le concepteur utilisera donc tout au long de la phase de développement de l'interface les règles ergonomiques qu'il estime adéquates en fonction du style d'interaction adopté, des objets interactifs utilisés et des moyens d'interactions impliqués afin de respecter les critères ergonomiques qu'il juge prépondérants.

Dans le cas de l'analyse et l'évaluation d'une interface existante par contre, la tâche est différente. En effet l'analyse d'une interface existante nécessite préalablement à toute analyse ergonomique de préciser quelle est la tâche que l'on peut exécuter à l'aide de l'interface et quels sont les profils des utilisateurs de l'interface. Une fois que ces deux paramètres sont précisés, il faut encore étudier l'application afin d'en déterminer la structure interne . Ce n'est qu'une fois que ces deux étapes ont été menées à bien que l'analyse ergonomique proprement dite peut débuter; elle est divisée en deux phases :

- la première phase est l'analyse fonctionnelle de l'interface qui permet de relever tous les types de composants de cette interface (les moyens d'interactions, les objets interactifs et les styles d'interactions);

- la deuxième phase est l'analyse ergonomique et consiste en la vérification du respect des règles ergonomiques adéquates ainsi qu'en la détermination des critères ergonomiques qui sont le plus souvent mis en jeu dans l'interface analysée.

Notons que la méthodologie d'analyse proposée ci-dessus est celle qui est présentée au cours d' « Interface Homme-Machine » [BODART92] du professeur F. Bodart.

2. Des règles ergonomiques spécifiques à l'E.A.O.

Les critères et les règles ergonomiques que nous avons vus jusqu'ici peuvent s'appliquer à l'interface de n'importe quelle application; l'E.A.O. en tant que discipline utilise néanmoins quelques-uns de ces critères et règles plus particulièrement en insistant sur leurs implications pédagogiques dans l'apprentissage de l'utilisateur. Le but de cette partie du chapitre est de préciser ces critères et règles ergonomiques et d'en retirer des conseils pour concevoir une interface efficace et conviviale pour un didacticiel. Nous nous baserons pour cela sur l'ouvrage « Concevoir et utiliser un didacticiel » [BESN88] pour la deuxième partie du chapitre.

2.1. Les critères ergonomiques importants pour l'E.A.O.

L'E.A.O. apporte une grande importance à quatre des critères ergonomiques présentés dans la première partie de ce chapitre. Ces critères sont la représentativité, la charge de travail, le contrôle du dialogue et l'adaptabilité. Comme on va le voir, ces quatre critères sont équivalents à trois dimensions abordées par Besnainou [BESN88] : la simplicité et la convivialité pour l'apprenant, le contrôle du dialogue et la facilité de déplacement.

2.1.1. Simplicité et convivialité pour l'apprenant

Le didacticiel peut être vu comme un service fournissant de l'information à l'apprenant et fournissant différents moyens de traiter cette information.

« L'accès aux informations, la fourniture d'informations et le traitement de ces informations par l'apprenant doit se faire par des actions rapidement compréhensibles, de façon à minimiser le temps de réflexion sur la signification des messages et le nombre d'actions stériles. » [BESN88]. Par exemple, si les différentes touches de fonction n'ont pas été définies clairement à l'entrée dans le didacticiel, l'apprenant risque très vite de se retrouver bloqué ou d'appeler des fonctionnalités non disponibles à ce moment ou inexistantes. De même si à un moment ou à un autre, un apprenant en situation d'autoformation se trouve bloqué devant un message obscur, il est presque certain que cet apprenant n'utilisera plus le didacticiel avant longtemps.

Les consignes d'utilisation du didacticiel doivent être très claires. Un apprenant prendra en effet rarement l'initiative, lors d'un premier contact avec la machine, d'essayer telle ou telle touche si le didacticiel ne les lui indique pas de façon précise. « Il a tendance à se méfier des réactions de l'ordinateur. Celui-ci peut déclencher des phénomènes émotionnels importants (identification, peur, rejet de toute responsabilité sur la machine, etc.) » [BESN88].

Ces constatations nous permettent de déduire que l'interface d'un didacticiel, pour être efficace, doit respecter les deux critères ergonomiques suivants : la charge de travail et la représentativité; la charge de travail en ce sens que toutes les actions exécutables par l'apprenant doivent être simples et courtes à utiliser et la représentativité en ce sens que les différents codes ou symboles utilisés pour représenter les actions exécutables doivent être clairs et explicites.

Dans notre cas , par exemple, le didacticiel permet à l'utilisateur d'exécuter entre autres des actions à partir de menus. Nous avons donc veillé à ce que le libellé de ces menus soit clair et explicite vis-à-vis des actions correspondantes.

2.1.2. Le contrôle du dialogue

« Une façon de dédramatiser le dialogue est de laisser l'apprenant manipuler des objets graphiques ou textuels et contrôler la relation. » [BESN88].

Le moyen le plus simple et le plus efficace pour cela est de prévoir des fonctionnalités permettant de « donner la main » à l'utilisateur. Certaines de ces fonctionnalités doivent être optionnelles telles que, par exemple :

- passer une question sans y répondre, soit que l'utilisateur la trouve sans intérêt, soit qu'il désire avancer plus loin pour y revenir ultérieurement;
- revenir en arrière et modifier sa réponse;
- appeler des fonctionnalités telles que un lexique, une aide, un zoom, etc... ;
- ...

Etant donné le caractère optionnel de ces fonctionnalités, c'est à l'utilisateur de choisir s'il les utilise ou non; il faut cependant que le concepteur de l'interface prévoie d'indiquer clairement les branchements optionnels et, éventuellement, la façon de s'en servir car « l'expérience montre que (...) les utilisateurs de didacticiels, et particulièrement les adultes, ont tendance à rester sur le cheminement principal (...) » [BESN88].

D'autres fonctionnalités visant à assurer le contrôle du dialogue par l'apprenant sont par contre obligatoires. L'exemple typique d'une de ces fonctionnalités obligatoires est le passage à l'écran suivant lors de la présentation d'une suite d'écrans. Il est en effet primordial de respecter le rythme de lecture de chacun, d'autant plus que la lecture sur un écran n'est pas une chose aisée. « La succession d'écrans-textes déferlant sans l'intervention de l'utilisateur, outre son aberration pédagogique, est à bannir. » [BESN88]. La temporisation est donc à réserver à l'affichage de graphiques ou encore aux animations et à exclure dans le cas de l'utilisation du texte.

Cette dimension est également mise en évidence par les membres du projet COLOS lorsqu'ils exigent des didacticiels qu'ils développent que ceux-ci soient hautement interactifs, c'est-à-dire que ces didacticiels doivent toujours réagir directement et de façon appropriée aux différentes actions de l'utilisateur. Cela traduit particulièrement bien leur souci de laisser à l'utilisateur le « pouvoir » de contrôler le dialogue lors de l'utilisation d'un didacticiel. C'est donc tout naturellement que, ayant développé notre didacticiel au sein de l'équipe COLOS de Lyon, celui-ci respecte cette philosophie.

Comme nous le verrons plus loin, notre didacticiel propose comme activité à l'apprenant de piloter une simulation graphique. Dans le cadre d'une telle activité, nous avons « laissé la main » à l'utilisateur. C'est donc lui qui assure en permanence le contrôle du dialogue. En effet, même lors de l'exécution de la simulation graphique, l'apprenant a la possibilité de stopper celle-ci et de la redémarrer ensuite.

Nous parlerons des différents moyens techniques permettant de contrôler le dialogue lorsque nous examinerons les règles ergonomiques correspondantes.

2.1.3. La facilité de déplacement

La possibilité pour un utilisateur de se déplacer comme il l'entend dans le didacticiel est primordiale. L'apprenant doit pouvoir se déplacer de module en module. Il peut très bien décider d'avoir besoin d'un module plus que d'un autre, ou décider de consulter à nouveau le didacticiel sur un point précis. « En lui permettant de se servir du didacticiel comme d'une base de données interactive, on augmente les potentialités de l'E.A.O. » [BESN88].

Il incombe donc au système de permettre à l'utilisateur de se déplacer à sa guise au sein du didacticiel tout en lui fournissant des repères. Lorsque, par exemple, un didacticiel travaille par menus et sous-menus, l'apprenant doit pouvoir revenir à tout moment aux têtes de chapitre à partir de n'importe quel endroit du didacticiel.

« Il est utile de prévoir un symbole ou un changement de couleur qui viendra s'afficher devant une rubrique quand l'utilisateur en aura consulté le contenu. Il saura où il en est. » [BESN88]. Ce genre de signalisation, si elle est retenue en fonction de l'apprenant, est également utile quand l'apprentissage a été interrompu ou lorsque le contenu du didacticiel est trop volumineux pour être vu en une seule fois. Mais cela implique que le didacticiel s'adapte en fonction de l'apprenant qui l'utilise et, par conséquent, que son interface respecte non seulement le critère ergonomique d'adaptabilité mais également le critère ergonomique de guidage de l'utilisateur.

2.2. Quelques règles ergonomiques pour l'interface d'un didacticiel

Après avoir exposé les critères ergonomiques importants à respecter lors de la conception d'une interface d'un didacticiel, il nous semble important de préciser quelques règles quant à la présentation visuelle de cette interface.

2.2.1. La présentation globale

Dès qu'il conçoit un didacticiel, l'auteur construit la présentation visuelle de l'ensemble du produit. Il prévoit pour cela une organisation des différents écrans en fonction des potentialités du système qu'il utilise et du didacticiel qu'il crée.

Il est bon de prévoir une division d'une fenêtre en zones dans lesquelles l'utilisateur retrouvera toujours le même type d'information. Par exemple, les zones que l'on peut retrouver dans une fenêtre sont :

- une zone de type boîte à outils;
- une zone pour les commentaires;
- une zone pour les consignes;
- ...

Une fenêtre est « une image unique, ronde, qu'il convient de structurer pour que l'utilisateur reconnaisse sans effort ses articulations et ses points forts. Les informations doivent être hiérarchisées visuellement pour faciliter la mémorisation. » [BESN88]. Lorsque l'utilisateur observe un écran, il fait d'abord appel au registre d'informations sensorielles (RIS) qui va stocker directement énormément d'informations. Malheureusement, seulement une partie de ces informations transitera d'abord du RIS à la mémoire à court terme et, encore

seulement une partie des informations passera alors de la mémoire à court terme à la mémoire à long terme. Ce qui a pour conséquence que dans une image écran, « (...) l'abondance de détails ne fait que compliquer le travail. » [BESN88]. En fait, un utilisateur moyen ne peut mémoriser que cinq à plus ou moins deux éléments d'information (Loi de Miller [BESN88]) car pendant qu'il travaille avec un didacticiel, il n'utilise que le RIS et la mémoire à court terme. Ce n'est qu'après l'utilisation du didacticiel que se fait le transfert de la mémoire à court terme vers la mémoire à long terme.

Il ne faut donc pas hésiter à reprendre des éléments plusieurs fois et à utiliser des mots clé et des associations. « Il est souhaitable que la sollicitation, la zone réponse et le commentaire à la réponse se trouvent sur le même écran. » [BESN88]. Cela évite à l'apprenant de devoir revenir en arrière pour vérifier le contenu d'une question ou le contenu d'une réponse qu'il a donnée.

Enfin, les expériences en psychologie sur le traitement de l'information ont démontré qu'un contenu est mieux retenu lorsqu'il est intégré dans une histoire, relié par un fil narratif, et hiérarchisé. Le concepteur d'un didacticiel devra donc veiller à ce que le parcours de son didacticiel corresponde à un scénario préétabli. Par exemple, dans le cas de notre didacticiel, le scénario mis en jeu est la simulation elle-même.

2.2.2. Les moyens d'interaction

Les moyens d'interaction reprennent tous les moyens techniques mis à la disposition de l'utilisateur dans l'interaction de celui-ci avec le didacticiel. Ces moyens techniques sont principalement les différents périphériques de saisie ou d'affichage qui existent à l'heure actuelle.

Les périphériques jouent donc un rôle primordial dans le dialogue entre l'utilisateur et la machine. Aujourd'hui la majorité des utilisateurs de l'informatique ne sont pas informaticiens de formation et n'ont donc pas à programmer : leur rapport à l'informatique se traduit donc par leur rapport aux périphériques.

Mais quels sont les périphériques principaux et quel est leur rôle :

- *la souris* : elle constitue à l'heure actuelle un des périphériques les plus répandus. Son usage est à éviter lorsqu'il s'agit de dessiner à main levée mais elle est idéale pour se déplacer sur l'écran ou pour « pointer » un objet;

- *l'écran tactile* : il est moins répandu. Il est agréable à utiliser pour l'apprenant lorsque les zones sont bien définies mais devient très vite agaçant lorsque la précision est faible. Il faut éviter une utilisation systématique de tels écrans pour l'entrée des réponses qui oblige l'utilisateur à rester très près de l'écran et entraîne chez lui une fatigue visuelle;

- *le crayon optique* : les mêmes remarques que pour l'écran tactile sont applicables ici;

- *le clavier* : il reste le périphérique le plus utilisé. Il est bien adapté à la saisie de réponse écrite mais est peu confortable pour pointer ou déplacer des objets;

- *l'écran* : il est le périphérique le plus important dans l'interaction entre la machine et l'utilisateur. Les caractéristiques à prendre en compte lors du choix d'un écran sont sa résolution, sa taille et le nombre de couleurs qu'il peut afficher.

2.2.3. Le texte

« L'imprimerie est un métier. Elle atteint de nos jours une qualité proche de la perfection due à des siècles de recherche. » [BESN88]

Il est important de pouvoir choisir la police des caractères qui seront affichés à l'écran. En effet, le pouvoir d'expression n'est pas le même selon qu'on utilise une police de caractères ou l'autre. De même, un caractère gras ou en italique n'impressionnera pas le lecteur de la même manière. Il s'agit donc pour le concepteur de choisir soigneusement la typographie qu'il utilisera en fonction de l'impression qu'il veut créer sur l'utilisateur.

En ce qui concerne l'espace utilisé à l'écran par le texte, il faut savoir qu'un écran ne supporte aucune surcharge; on ne lit pas un écran comme on lit une feuille de papier. En règle générale, un écran « doit être occupé au tiers, ou à la moitié de sa surface au maximum. » [BESN88]. De plus, les lignes doivent avoir une moyenne de 50 à 65 caractères pour un écran de 80 colonnes. Des lignes trop courtes ne rendent pas plus lisible un texte que des lignes trop longues.

2.2.4. Le graphisme

Le monde dans lequel nous vivons actuellement est un monde où la communication est fortement visuelle. Jusqu'à maintenant, l'enseignement a privilégié une approche rationnelle et analytique des notions qu'il abordait; l'usage du graphisme permet de rétablir un équilibre vis-à-vis de l'approche intuitive de ces mêmes notions.

Dans un didacticiel, l'unité d'échange est l'image écran; elle permet de concilier les deux approches : le texte permettra de conserver l'approche analytique alors que le graphique fera plutôt appel à l'intuitif.

On distingue 4 types de communication graphique.

a) **La représentation figurative**

La représentation figurative n'implique aucun apprentissage car elle présente une image ressemblant à l'objet réel. Elle permet de visualiser des exemples mais il faut veiller à ce qu'elle ne soit pas uniquement illustrative et, dans ce cas, redondante. Elle peut servir d'agent motivateur important mais ne facilite pas du tout le passage à l'abstraction.

Il faut également remarquer que, pour être de qualité, la représentation figurative nécessite un bon environnement graphique et du temps pour sa conception et sa mise au point. Elle est donc coûteuse.

La figure 2.1. montre un exemple de la représentation figurative d'un ordinateur. L'image fournie ressemble bien à l'objet réel mais ne permet pas de se rendre compte de ce que représente réellement un ordinateur du point de vue matériel utilisé par exemple.

Figure 2.1. : La représentation figurative d'un ordinateur.

b) **La représentation schématique**

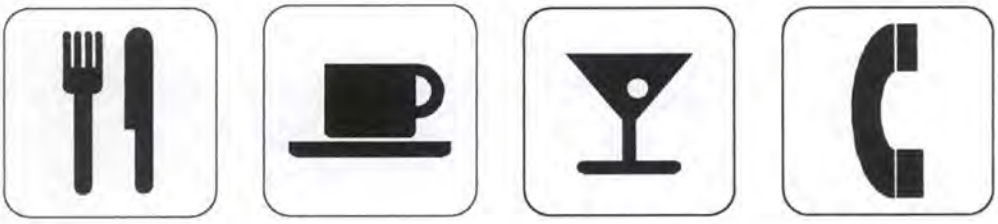
La représentation schématique n'a pas une signification immédiatement perceptible par l'observateur; elle nécessite au contraire une certaine réflexion de la part de celui-ci.

Par exemple, un schéma conceptuel va permettre à l'utilisateur d'avoir une vision globale d'un raisonnement décrit par un texte.

c) **La représentation symbolique**

La représentation symbolique se sert de pictogrammes. Un pictogramme est une image qui se rapporte à une image réelle mais qui, pour avoir la caractéristique d'être une information claire et rapide, a été stylisée. Ce type de représentation est utilisée dans le but de respecter le critère ergonomique de représentativité.

La figure 2.2. reprend la représentation symbolique de quelques objets réels courants que le lecteur n'aura aucun mal à reconnaître.



Des couverts Une tasse Un verre Un téléphone

Figure 2.2. : La représentation symbolique de quelques objets courants.

d) **La représentation abstraite**

La représentation abstraite nécessite un apprentissage car la signification donnée est de pure convention. Ce type de représentation n'est à utiliser que lorsque l'on est sûr qu'elle est bien comprise par la population que vise le didacticiel.

Nous verrons, lors de la présentation de l'interface de notre didacticiel, que la représentation adoptée pour les machines est une représentation abstraite.

2.2.5. Les couleurs

La couleur, comme tous les autres aspects d'une interface homme-machine, a « ses propres contraintes et ses limites basées sur la physiologie et la psychologie, et que mal utilisée, elle peut pervertir la présentation de l'information. » [BESN88]

Diverses études ont démontré que l'usage de la couleur dans le cadre d'un didacticiel est efficace à condition de respecter les règles suivantes :

- Les couleurs doivent être peu nombreuses.

  Il est souhaitable de ne pas dépasser 4 couleurs sur le même écran. Si on emploie tout de même plus de 5 couleurs simultanément, il est nécessaire d'augmenter la taille des objets codés par la couleur.

- Le contenu nécessite une hiérarchisation.

- Les couleurs utilisées doivent bien se différencier entre elles.

- Le codage social et culturel commun de la couleur doit être respecté.

  Les couleurs ont un sens social et culturel. Les couleurs de base sont généralement perçues de la façon suivante [BESN88] :

  Rouge : force vitale et active.
  Bleu : effet calmant.
  Jaune : effet stimulant.
  Vert : effet de stabilité.
  Violet : effet de passage et de métamorphose.
  Gris : négation.
  Noir : négation absolue.
  Blanc : effet positif

- La couleur doit être associée à une forme.

  Ce genre de précaution permet de prendre en compte les utilisateurs daltoniens qui constituent un pourcentage non négligeable de la population.

De plus, des recherches dans ce domaine montrent également que :

- les couleurs attirent l'attention;
- les performances d'apprentissage des jeunes enfants ne sont pas améliorées par l'utilisation des couleurs;
- la couleur aide à mémoriser certains détails mais a peu d'influence sur le contenu principal;
- la couleur a surtout un impact émotionnel sur l'apprenant;
- les écrans très contrastés sont mieux perçus.

Toutes ces constatations nous montrent que l'usage des couleurs est un domaine très sensible qu'il faut aborder avec précaution. N'étant pas des experts, nous avons prévu que toutes les couleurs utilisées dans notre didacticiel puissent être personnalisées par l'utilisateur.

Chapitre 3 : L'E.A.O. dans le domaine des télécommunications

Au vu des premiers chapitres, on voit que l'enseignement assisté par ordinateur constitue une discipline à part entière; discipline qui instaure ses propres règles notamment quant à la création des didacticiels. Mais l'E.A.O. n'en reste pas moins ouvert à tous les domaines d'enseignement. L'ordinateur, qui constitue le moyen de communication privilégié de l'E.A.O., offre en effet une souplesse d'utilisation et de modélisation tellement importante qu'à peu près n'importe quelle matière peut faire l'objet d'une modélisation qui résultera en la création d'un didacticiel dédié à l'enseignement de cette matière.

Le didacticiel que nous avons développé dans le cadre de ce travail est, comme nous l'avons déjà précisé, dédié à l'enseignement des notions principales du monde d'Internet. Cependant, il nous semble important de présenter quelques applications d'E.A.O. dans le domaine des télécommunications afin de rendre compte de l'état de l'art dans ce domaine avant d'entamer la description détaillée de notre didacticiel.

La première constatation que nous nous sommes faits lors de la recherche de tels didacticiels est que ce domaine ne semble pas avoir déjà été exploité de façon soutenue. Il nous a été difficile en effet de retrouver les quelques didacticiels présentés dans le cadre de ce chapitre. Notons cependant qu'il existe bien entendu d'autres didacticiels dans le domaine des télécommunications que ceux présentés ici; mais nous n'avons pas pu en acquérir de documentation faute de temps et de moyens financiers.

1. Un didacticiel pour Ethernet

Le premier didacticiel que nous avons reçu est un didacticiel couvrant quelques notions propres aux réseaux locaux de type Ethernet. Ce didacticiel nous a été gracieusement envoyé par Philippe Garcia-Suarez, étudiant à Paris et concepteur du produit [GARCIA94].

L'objet du didacticiel est la simulation d'un bus Ethernet dont les paramètres sont définis par l'utilisateur, le but est de calculer le nombre de collisions survenues et le nombre de trames perdues sur le bus lors de la simulation. Avant d'entrer dans la description détaillée du didacticiel, il nous semble important de rappeler quelques notions de base du protocole Ethernet.

1.1. Les réseaux Ethernet

Nous nous baserons pour le rappel de ces notions sur le cours de seconde licence « Téléinformatique et Réseaux » [VANBAST93] 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 repris et 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.

Les deux protocoles utilisent la méthode d'accès CSMA/CD (Carrier Sense Multiple Access with Collision Detection) et fonctionnent actuellement à 10 Mbps[2]. Ils sont initialement conçus pour des réseaux à topologie de bus sur support coaxial; deux versions de support coaxial peuvent être 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 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 ici égale à 500 mètres. Mais 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). On peut utiliser des répéteurs pour relier logiquement plusieurs segments en un seul réseau, répéteurs qui régénèrent en fait les signaux binaires. Une telle technique permet d'étendre des réseaux Ethernet jusqu'à une longueur de 2,5 km.

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; lorsque celle-ci est libre, la trame est passée au niveau physique, c'est pourquoi on appelle cette technique Carrier Sense Multiple Access. Comme il est possible que plusieurs stations attendent que le support soit libre, il se peut que plusieurs trames soient émises simultanément par plusieurs stations, cela résulte en une déformation des signaux émis et donc en une collision. Lorsqu'un niveau physique détecte une collision, celui-ci arrête

[1] ALOHA a été développé à l'université de Hawaii pour des réseaux radiodiffusés.
[2] Mbps signifie Mega bits par seconde.

immédiatement d'émettre et un message spécifique de collision est émis pour s'assurer que toutes les stations soient bien au courant de la collision. On appelle cela la « Collision Detection (CD) ». Toutes les trames qui ont été déformées par une collision ne sont pas reçues.

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

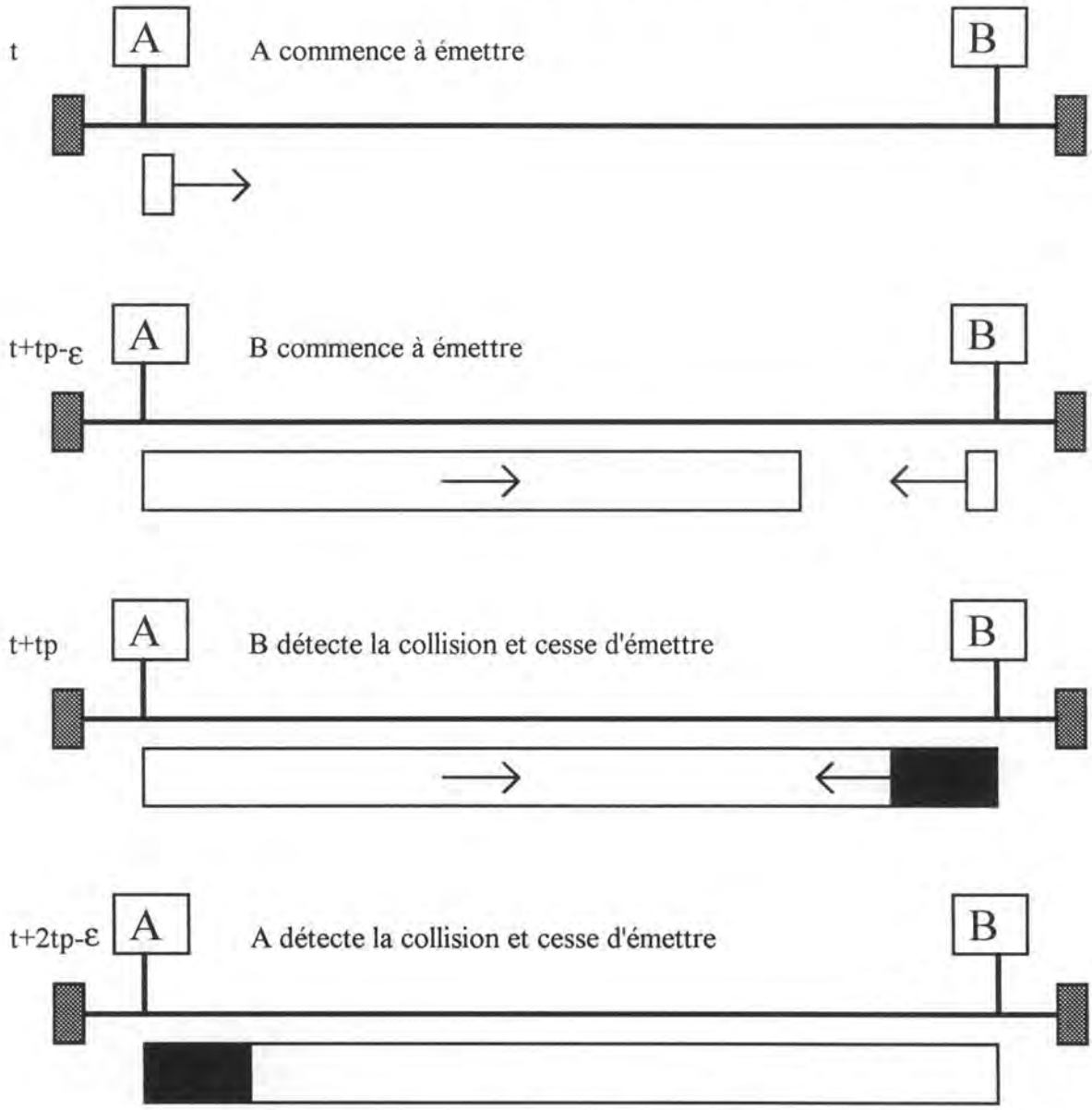


Figure 3.1. : La détection d'une collision.

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 a pour conséquence 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 le voit à la

figure 3.1., dans la pire des configurations qui 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 (la station B de la figure 3.1.) commence à émettre juste avant qu'elle ne se rende compte que l'autre station (la station A de la figure 3.1.) était en train d'émettre, le temps nécessaire à la station A pour détecter la collision est égal au double du temps de propagation d'un bout à l'autre du réseau (le temps noté tp dans la figure 3.1.).

Donc, pour une taille maximale de 5 kilomètres, le slot-time est de l'ordre de 50 microsecondes; 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 a été fixée à 512 bits.

1.2. Le didacticiel

Le didacticiel impliqué propose la simulation d'un bus Ethernet dont les paramètres peuvent être définis par l'utilisateur final. Le didacticiel fournit alors comme résultats le nombre de collisions survenues et de trames perdues lors de la simulation.

1.2.1. L'utilisation du didacticiel

L'utilisation du didacticiel est assez simple. La première chose que doit faire l'utilisateur est de spécifier les paramètres qu'il désire pour la simulation. Les paramètres à spécifier sont au nombre de trois : la longueur du bus, la longueur des trames émises et la probabilité d'émission des stations.

- **La longueur du bus**

  Le didacticiel propose un choix de trois longueurs de bus différentes : 100 mètres, 1000 mètres et 2500 mètres; le didacticiel impose alors le nombre de stations placées sur le bus en fonction de sa longueur. Le bus de 100 mètres relie 10 stations réparties tous les 10 mètres, le bus de 1000 mètres relie 50 stations réparties tous les 20 mètres et, enfin, le bus de 2500 mètres reprend 200 stations réparties tous les 12,5 mètres.

- **La longueur des trames émises**

  Comme pour la longueur du bus, le didacticiel offre trois possibilités pour la longueur des trames émises : 500 bits, 2000 bits et 10000 bits. Il n'est possible de spécifier qu'une longueur de trame par simulation; toutes les trames émises lors de la simulation auront donc toutes la même longueur.

- **La probabilité d'émission des stations**

  Ce paramètre permet en fait de régler la densité du trafic sur le bus Ethernet lors de la simulation. La probabilité d'émission correspond à un pourcentage du temps de la simulation durant lequel une station émet, sans contraintes sur le moment où l'émission a lieu. Les probabilités d'émission des stations proposées par le didacticiel sont 3% pour un trafic faible, 10% pour un trafic moyen et 100% pour un trafic à saturation.

Lorsque les paramètres sont spécifiés, il reste à l'utilisateur à lancer la simulation et à attendre les résultats. Une fois la simulation terminée, le didacticiel affiche les résultats à l'écran; ces résultats reprennent entre autres le nombre de collisions survenues et le nombre de trames perdues lors de la simulation.

Le didacticiel offre encore d'autres fonctionnalités telles que l'affichage des résultats de simulations passées, l'enregistrement sur disque des résultats des simulations exécutées, la terminaison du didacticiel à tout moment ou encore une fonction d'aide décrivant les fonctionnalités et l'interface du didacticiel.

1.2.2. Les principales caractéristiques pédagogiques du didacticiel

Après avoir décrit l'utilisation du didacticiel, il nous semble important d'en rechercher les principales caractéristiques pédagogiques afin de pouvoir nous rendre compte des buts poursuivis par les concepteurs de celui-ci. Nous nous intéresserons principalement aux paramètres pédagogiques suivants : l'objectif pédagogique, la stratégie pédagogique, les pré-requis nécessaires et l'interface développée.

- **L'objectif pédagogique**

L'objectif pédagogique poursuivi par ce didacticiel est la compréhension par l'apprenant de l'impact des différents paramètres modifiables sur le fonctionnement réel d'un bus Ethernet. Cette objectif étant particulièrement restreint, nous n'avons décelé aucun objectif secondaire, si ce n'est bien sûr la compréhension de l'impact de la modification d'un paramètre unique, toute autre chose restant égale par ailleurs.

- **La stratégie pédagogique**

La stratégie pédagogique a été définie dans le premier chapitre comme la combinaison des ressources disponibles afin d'arriver au but fixé. Elle représente donc les moyens mis en oeuvre pour atteindre l'objectif.

La stratégie pédagogique mise en oeuvre par ce didacticiel est le pilotage d'une simulation par les étudiants d'un cours de télécommunications dans le but de comprendre l'impact de certains paramètres sur le fonctionnement d'Ethernet.

- **Les pré-requis nécessaires**

Le didacticiel exige de la part des utilisateurs d'avoir déjà acquis un certain nombre de notions dans le domaine des télécommunications en général et du monde d'Ethernet en particulier. En effet, pour que l'utilisateur puisse comprendre les différents résultats fournis par le didacticiel, il faut qu'il sache ce qu'est un bus Ethernet, comment fonctionne un réseau Ethernet et ce qu'est une collision sur un bus Ethernet. Ce didacticiel est donc destiné à des utilisateurs ayant déjà atteint un niveau relativement haut dans le domaine des télécommunications.

- **L'interface développée**

L'interface développée, représentée à la figure 3.2., est relativement simple et assez agréable. On peut diviser l'écran en cinq zones : la zone de titre située en haut de l'écran, la zone d'affichage des résultats qui est située dans la partie centrale de l'écran, la zone des

paramètres qui reprend les différents paramètres modifiables par l'utilisateur, la zone de témoin de simulation qui reprend une ligne de point de couleur qui indique l'état d'avancement de la simulation et, enfin, la zone des boutons qui reprend les boutons correspondant aux différentes fonctionnalités offertes par le didacticiel.

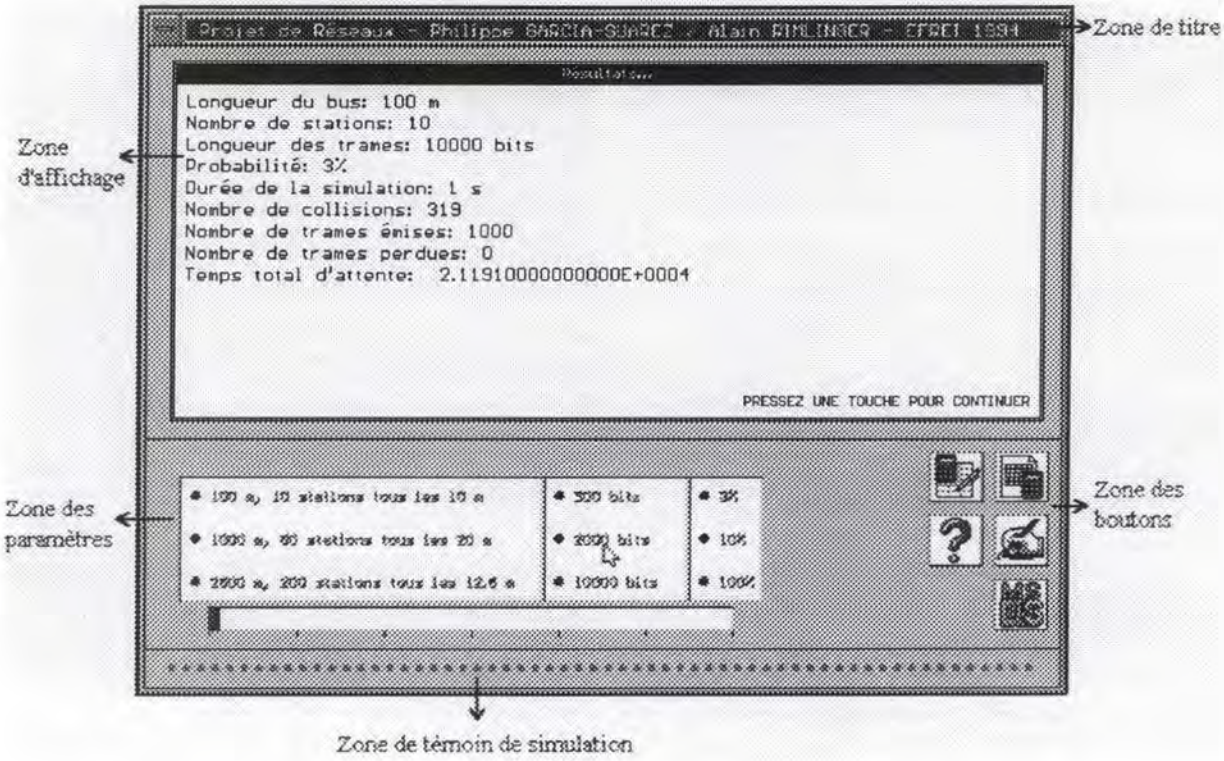


Figure 3.2. : L'interface du didacticiel pour Ethernet.

On peut cependant déplorer les deux choses suivantes : d'une part, la représentation des boutons n'est pas toujours explicite, il n'est pas évident en effet de deviner que le bouton situé dans le coin supérieur gauche de la zone des boutons est celui qui déclenche la simulation; et, d'autre part, la simulation se fait en « background » c'est-à-dire que mis à part le témoin d'avancement de la simulation, rien n'illustre la simulation qui est en cours.

En conclusion, nous pouvons dire que ce didacticiel reprend des notions relativement poussées du monde d'Ethernet et, par là, est destiné à des utilisateurs ayant déjà atteint un certain niveau d'expérience dans ce domaine. Du point de vue de l'interface, le didacticiel est assez simple à utiliser mais les deux défauts que nous avons soulevés rendent moins agréable sa manipulation par l'utilisateur; ce qui pourrait résulter en une baisse de la motivation de l'apprenant. Remarquons encore qu'il n'existe pas de réelle interactivité entre le didacticiel et l'apprenant.

2. Le guide Intel pour les réseaux

Le guide Intel pour les réseaux [INTEL94] est un logiciel de formation développé par la société Intel et destiné à expliquer à un utilisateur novice ce qu'est un réseau informatique. Ce didacticiel présente pour cela un certain nombre d'informations illustrées par de petites animations; c'est l'utilisateur qui, d'une part, détermine à l'aide d'un menu les informations qu'il désire recevoir et, d'autre part, décide du rythme de travail qu'il veut suivre.

2.1. L'utilisation du didacticiel

L'utilisation du didacticiel est très simple; en voici la description.

La première étape lorsqu'un utilisateur démarre le didacticiel est de choisir la langue dans laquelle cet utilisateur désire que l'information lui soit fournie. Le didacticiel propose l'anglais, le français, l'espagnol et l'allemand. Une fois que l'utilisateur a choisi la langue qu'il désirait, le didacticiel entre dans la phase de formation proprement dite : il propose à l'utilisateur de choisir un chapitre particulier de l'ensemble des chapitres proposés. Ces chapitres forment en fait une découpe de toute l'information que peut fournir le didacticiel; plus l'utilisateur avancera dans sa formation, plus il rencontrera des chapitres compliqués.

Lorsque l'utilisateur a choisi un chapitre, le didacticiel fournit alors l'information relative à ce chapitre. Cette information est organisée en pages écran, c'est-à-dire que le didacticiel fournit l'information page par page et l'utilisateur détermine lui-même lorsqu'il veut passer à la page suivante en activant le bouton correspondant. Le didacticiel, en plus de fournir l'information sous forme de texte, fournit également quelques animations qui illustrent l'information textuelle qui est fournie simultanément. Une telle procédure permet à l'apprenant d'avoir une meilleure compréhension de l'information fournie.

Lors de sa manipulation, l'apprenant décide donc lui-même du rythme de travail puisque c'est lui qui décide de passer à la page d'information suivante. De plus, s'il n'a pas eu le temps de bien observer l'animation qui s'est déroulée en illustration du texte présenté, le didacticiel lui offre la possibilité de recommencer cette animation directement; il lui suffit pour cela de déclencher la fonctionnalité requise en activant le bouton correspondant.

L'utilisateur peut passer d'un chapitre à l'autre de deux manières différentes : soit il suit l'ordre logique des pages d'information et lorsqu'il arrive à la fin du chapitre courant, le fait de changer de page le fait automatiquement passer dans le chapitre suivant; soit il choisit dans l'ensemble des chapitres présenté en permanence à l'écran le chapitre qu'il désire parcourir directement. Comme nous le verrons dans la description de l'interface, le didacticiel présente donc en permanence à l'écran l'ensemble des chapitres qu'il propose. En plus, le didacticiel met en évidence par un changement de couleur le chapitre dans lequel l'utilisateur se trouve; cette indication permettra donc à l'utilisateur de reprendre une manipulation interrompue directement au chapitre auquel il s'était arrêté.

Enfin, l'utilisateur peut quitter le didacticiel à n'importe quel moment; il suffit pour cela qu'il active le bouton correspondant à la fonctionnalité de terminaison d'exécution. S'il veut reprendre la formation à l'endroit où il s'était arrêté, il ne doit cependant pas oublier de noter le chapitre dans lequel il se trouve à l'arrêt du didacticiel afin de pouvoir sélectionner ce chapitre directement au début de l'utilisation suivante.

2.2. Les principales caractéristiques pédagogiques du didacticiel

Tout comme pour le premier didacticiel, il nous semble important de préciser les principales caractéristiques pédagogiques du didacticiel qui nous permettront de préciser les buts poursuivis par le concepteur de celui-ci. Les caractéristiques qui nous semble les plus importantes à retenir sont l'objectif pédagogique, la stratégie pédagogique, les pré-requis nécessaires à l'utilisation du didacticiel et l'interface développée.

- **L'objectif pédagogique**

L'objectif pédagogique à atteindre par les utilisateurs de ce didacticiel est l'assimilation et la compréhension des notions de base du monde des réseaux informatiques en général. Cette objectif principal a été bien entendu divisé en une série d'objectifs secondaires par les concepteurs du didacticiel : chaque objectif secondaire est représenté par un chapitre d'information.

Au fur et à mesure que l'apprenant avancera dans sa formation, il atteindra de nouveaux objectifs secondaires en assimilant les chapitres correspondants. Lorsque tous les chapitres auront été assimilés, tous les objectifs secondaires auront été atteints et l'objectif pédagogique principal sera alors également atteint.

- **La stratégie pédagogique**

La stratégie pédagogique principale de ce didacticiel est le parcours de pages d'information par l'apprenant à la manière du parcours des pages d'un livre. Cette stratégie est cependant consolidée de façon non négligeable par l'ajout d'animations illustrant le texte présenté et qui permettent à l'apprenant de comprendre de façon plus profonde les diverses notions présentées.

- **Les pré-requis nécessaires**

Les pré-requis nécessaires à l'utilisation de ce didacticiel sont quasi inexistants. Les seules choses qui nous semblent importantes à connaître sont la manipulation d'un programme informatique à l'aide d'une souris d'une part et la notion de ce qu'est un ordinateur en général d'autre part. Ces deux notions étant connues à l'heure actuelle de la majorité des personnes, nous pouvons dire que ce didacticiel ne requiert aucun pré-requis particulier.

- **L'interface développée**

L'interface développée qui est représentée à la figure 3.3. peut être qualifiée de conviviale. L'écran peut être divisée en 4 zones distinctes : la zone des chapitres qui reprend l'ensemble des chapitres présentés par le didacticiel, la zone de présentation de l'information dans laquelle toute l'information sous forme de texte est présentée, la zone des animations dans laquelle toutes les animations sont exécutées et, enfin, la zone des boutons qui reprend les boutons correspondant aux différentes fonctionnalités offertes par le didacticiel, à savoir le bouton qui permet de passer à la page suivante, le bouton qui permet de redémarrer l'animation, le bouton permettant de remonter vers le menu principal et le bouton qui permet de quitter le didacticiel.

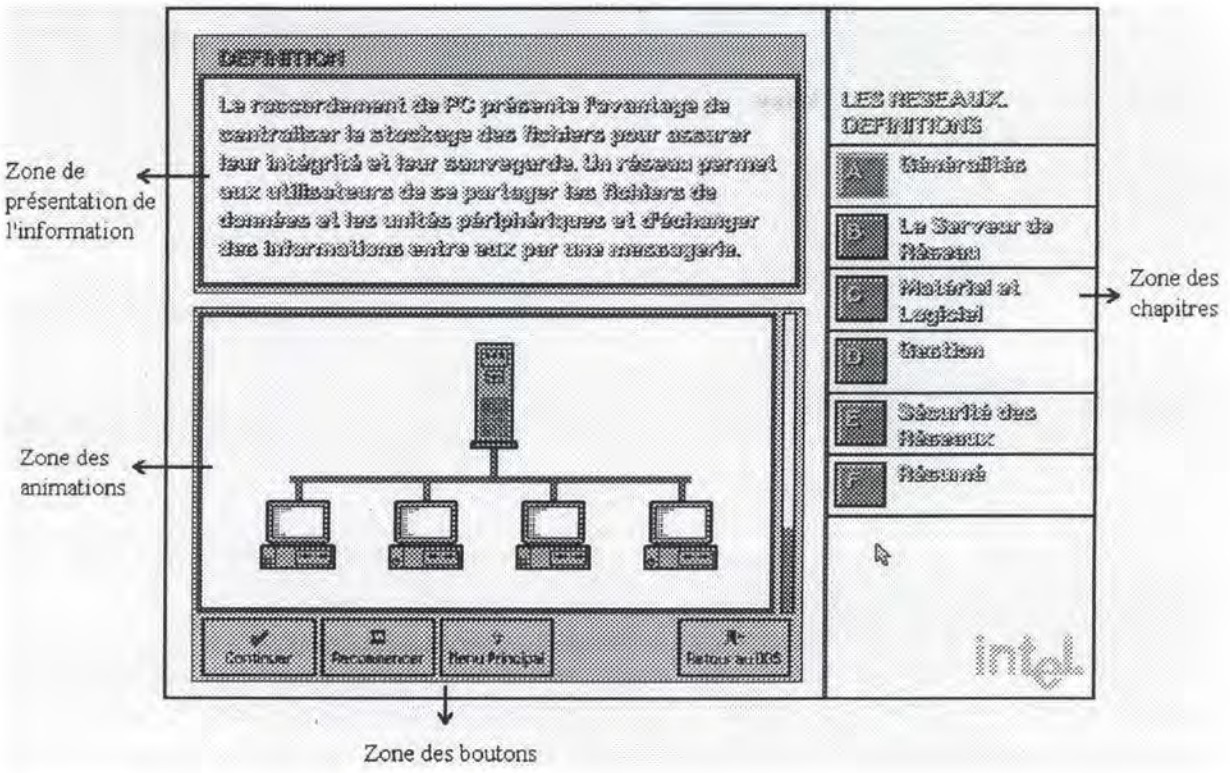


Figure 3.3. : L'interface du guide Intel pour les réseaux.

On peut regretter que les concepteurs n'aient pas prévu de fonctionnalité permettant de revenir une page en arrière, ce qui peut s'avérer utile dans le cas où on n'est plus sûr d'une information qui vient juste de passer. Dans l'état actuel des choses, lorsque l'utilisateur veut reculer d'une page, il est obligé de reprendre le chapitre (dans lequel cette page se trouve) au tout début et de passer de page en page jusqu'à ce qu'il arrive à la page souhaitée.

On peut également déplorer le fait qu'il n'y ait pratiquement aucune interactivité entre le didacticiel et l'utilisateur mis à part le passage de page en page qui est déclenché par l'utilisateur. C'est une conséquence directe de la stratégie pédagogique adoptée qui n'est finalement que de présenter de l'information à l'utilisateur, à l'image du livre qui fournit de l'information au lecteur.

Nous pouvons toutefois dire en conclusion que ce didacticiel extrêmement simple d'emploi constitue un outil idéal pour la formation de personnes novices dans le domaine de l'informatique. Le didacticiel ne nécessite en effet aucun pré-requis particulier de la part des utilisateurs et son interface est suffisamment simple et conviviale pour que les utilisateurs ne soient pas « démotivés » par une manipulation trop complexe de l'outil.

3. Notre didacticiel

La deuxième partie de ce mémoire étant entièrement consacrée à la présentation de notre didacticiel, nous n'en donnerons ici qu'une brève description à titre comparatif vis-à-vis des deux didacticiels décrits plus haut.

Notre didacticiel tente de faire assimiler et comprendre les principes de fonctionnement d'une interconnexion basée sur le modèle d'Internet à l'aide d'une simulation graphique de ce

fonctionnement. Au contraire du didacticiel sur Ethernet, notre didacticiel n'a donc pas pour objet de fournir des chiffres exacts traduisant les performances d'une interconnexion mais bien de représenter graphiquement comment un datagramme transite au sein d'une interconnexion simple afin que l'apprenant puisse se rendre compte par lui-même des différents principes du monde d'Internet.

La stratégie pédagogique adoptée dans notre didacticiel consiste à laisser l'utilisateur totalement libre de piloter la simulation comme il l'entend. Cela consiste pour lui à sélectionner les machines qu'il désire voir communiquer ensemble et ensuite à démarrer la simulation de la communication entre ces deux machines. La simulation est représentée à l'aide de l'animation graphique du transit d'un datagramme entre les deux machines sélectionnées. L'utilisateur peut bien sûr arrêter et redémarrer la simulation autant de fois qu'il le désire; il est donc libre d'avancer à son propre rythme lors de la formation.

Le didacticiel permet également à l'utilisateur d'observer des principes tels que l'exécution de l'algorithme de routage par les passerelles ou encore la modification du format du datagramme lors de son transit à travers des sous-réseaux de l'interconnexion. Il nécessite cependant de la part de l'utilisateur que celui-ci ait certaines notions de base du monde d'Internet comme savoir ce qu'est une interconnexion, connaître la différence entre une passerelle et un hôte classique ou encore avoir une idée de ce qu'est la découpe en couches des protocoles de communication. Mais l'ensemble de ces pré-requis semble naturel lorsque l'on sait que notre didacticiel est destiné à être utilisé par des étudiants en informatique ayant déjà suivi certains cours de télécommunications.

On peut donc dire que du point de vue des pré-requis nécessaires, ce didacticiel se situe entre les deux premiers didacticiels présentés. Il nécessite en effet plus de pré-requis que le guide Intel pour les réseaux mais sûrement moins que le didacticiel pour Ethernet.

En ce qui concerne l'interactivité par contre, notre didacticiel peut être qualifié de hautement interactif puisqu'il laisse l'utilisateur libre de le manipuler comme celui-ci l'entend et qu'il réagit en conséquence des actions exécutées par cet utilisateur.

Nous renvoyons aux chapitres suivants pour une description plus approfondie de notre didacticiel.

Chapitre 4 :
Un didacticiel pour Internet.

De nos jours, les télécommunications prennent une place de plus en plus prépondérante dans le monde. Que ce soit par obligation professionnelle ou par détente, les gens utilisent de plus en plus les moyens de communication moderne pour entrer en contact avec l'extérieur.

Une des technologies de télécommunication qui a connu un développement très important ces dernières décennies est sans aucun doute la technologie des réseaux informatiques. De tels réseaux peuvent être comparés au réseau téléphonique : c'est un ensemble de postes (les ordinateurs dans ce cas précis) qui peuvent communiquer entre eux en respectant un protocole.

Cette notion de protocole est une des notions, si pas la notion la plus importante, du domaine des réseaux informatiques. Les protocoles qui ont connu le succès le plus retentissant sont la suite de protocoles TCP/IP. Des réseaux s'appuyant sur cette suite de protocoles forment alors une interconnexion dont un exemple est le réseau mondial Internet. Internet est donc un réseau de sous-réseaux.. Dans la suite de l'exposé, nous parlerons de réseau pour une interconnexion et de sous-réseaux pour des réseaux interconnectés. Aujourd'hui, ce réseau est tellement étendu (plus de trois millions de machines et plus de 10 millions d'utilisateurs) qu'il est presque devenu un standard. Voilà pourquoi nous avons choisi Internet comme contenu pour notre didacticiel. Il nous semble en effet important dans le cadre de l'enseignement des télécommunications d'acquérir une compréhension profonde des divers mécanismes sous-jacents au fonctionnement des réseaux basés sur Internet.

Toutefois avant de décrire notre didacticiel, ce qui fera l'objet du chapitre 5, il nous paraît important de préciser les principales caractéristiques d'Internet. Cette présentation s'articulera en quatre points. Tout d'abord nous donnerons un aperçu de **l'historique** d'Internet. Nous parlerons ensuite d'un principe de base d'Internet : **l'interconnexion**; ce qui

nous amènera au principe d'**adressage** pour finir avec le **routage**. Nous nous baserons pour cela sur l'ouvrage : "Internetworking with TCP/IP" [COMER90].

1. Un bref historique d'Internet

Le réseau Internet a vu le jour dans les années 1970 sous les auspices de la DARPA (Defence Advanced Research Project Agency), une organisation américaine favorisant la recherche à des fins militaires. Ce sont les chercheurs de cette agence qui sont à la base de la suite de protocoles TCP/IP. Assez rapidement, d'autres chercheurs se sont intéressés à ce type de réseau et se sont connectés à celui établi par la DARPA. Voyant l'intérêt des chercheurs, DARPA décide alors d'organiser des " réunions informelles " (news group) entre ces chercheurs afin de partager leurs idées et de discuter les résultats des expérimentations effectuées. Devant le succès de ces " réunions ", les autorités américaines ont alors décidé d'imposer TCP/IP comme protocole au long cours et de scinder le réseau mis en place par la DARPA en un réseau civil, à des fins de recherche (ARPANET) et un réseau militaire (MILNET).

Pour favoriser la connexion des universités américaines, la DARPA a financé l'implémentation des protocoles TCP/IP dans le système BSD Unix (de l'université de Berkeley), alors fort répandu dans les universités américaines. Les centres de recherche ont été incités à choisir cette implémentation par son faible coût et les facilités de programmation offertes.

La National Science Foundation, devant le succès rencontré par la technologie TCP/IP, prend un rôle actif et, en 1986, les centres de super-ordinateurs sont connectés et forment le réseau dorsal NSFNet, lui-même relié au réseau ARPANET.

La croissance d'Internet est extraordinaire. Fin 1987, le taux de croissance mensuel est de 15 % ! En 1990, 3000 sous-réseaux sont connectés, ce qui représente 200.000 ordinateurs. Au-delà des universités, centres de recherche et organisations gouvernementales américains, des grandes compagnies sont connectées à ce réseau. De plus, le réseau Internet ne se limite plus aux seuls USA, mais devient véritablement mondial, avec des pays d'Europe, d'Asie, d'Afrique et d'Amérique. De nos jours, plus de trois millions de machines sont connectées à Internet et plus de 10 millions d'utilisateurs en font un usage courant.

Le réseau Internet est mis actuellement en avant en tant que possibilité pour une " *information highway* ", chère au Président Clinton et, surtout, à son Vice-Président Al Gore.

2. Le principe d'interconnexion

Nous avons dit plus haut que Internet s'étendait aujourd'hui sur une grande partie de la surface de la terre et que plus de 10 millions de personnes l'utilisaient régulièrement. Ces constatations impliquent que, si l'on veut garder des performances acceptables, il n'est pas possible qu'un réseau mondial unique prenne en charge tous les utilisateurs. La solution à ce problème est de relier des sous-réseaux plus petits (LAN et MAN par exemple) mais qui fournissent des vitesses de communication plus élevées et, une fois que ces sous-réseaux sont reliés, essayer de faire abstraction des détails relatifs à chacun d'entre-eux pour obtenir un

mode de communication uniforme entre toutes les machines connectées sur ces sous-réseaux. C'est ce qu'on appelle l'**interconnexion**.

Deux méthodes particulières sont possibles pour faire abstraction des détails des sous-réseaux interconnectés : une interconnexion au niveau des applications ou une interconnexion au niveau des réseaux [COMER90].

2.1. Une interconnexion au niveau des applications

Le principe ici est d'utiliser un programme tournant au niveau de la couche application afin de cacher les détails caractérisant le sous-réseau. Dans de tels systèmes, un programme, s'exécutant au niveau de la couche application de chaque machine connectée, comprend les détails de fonctionnement et de connexions des sous-réseaux et assure le bon fonctionnement des applications à travers les interconnexions.

Une telle approche peut sembler naturelle au premier abord mais est assez limitée et s'alourdit très vite. Ajouter de nouvelles fonctionnalités au système par exemple implique l'implémentation de nouveaux programmes d'interface pour chaque type de machine. Installer un nouveau sous-réseau implique la modification ou la création de nouveaux programmes d'interface pour chaque application possible.

Ainsi, on se rend compte que si l'interconnexion grandit jusqu'à recueillir quelques centaines, voire milliers de sous-réseaux, une telle approche n'est pas réalisable.

2.2. Une interconnexion au niveau des sous-réseaux

Une autre possibilité pour cacher les détails des sous-réseaux aux programmes de la couche application est un système basé sur l'interconnexion au niveau des sous-réseaux. L'interconnexion au niveau des sous-réseaux fournit un mécanisme, situé au niveau de la couche réseau, qui délivre des paquets de taille réduite à partir de l'émetteur jusqu'au destinataire en temps réel c'est-à-dire sans devoir préalablement ouvrir une connexion.

Transporter des paquets de données de taille réduite plutôt que des messages ou des fichiers de taille importante présente plusieurs avantages non négligeables.

a) Cela autorise l'utilisation directe de la technologie matérielle des sous-réseaux traversés, c'est-à-dire que le système peut directement utiliser le format des unités de communications défini par le protocole propre au sous-réseau traversé. Cela permet bien évidemment d'optimiser l'efficacité de la communication.

b) Cela permet de scinder les activités de communications des activités de la couche application, ce qui permet aux machines de contrôler le trafic du sous-réseau sans pour autant comprendre le fonctionnement des programmes de la couche application, le programme chargé de la communication recevant de petites unités de données et les transmettant via le sous-réseau sous-jacent.

c) Cela garde une certaine flexibilité au système. Il est possible de développer de nouveaux protocoles de réseau et de les introduire facilement dans le système

existant, sans pour autant devoir modifier de quelque façon que ce soit les programmes de la couche application.

d) Cela permet au gestionnaire du réseau d'installer de nouvelles technologies de réseau en ne devant que modifier ou ajouter un module au niveau du programme de la couche réseau, sans devoir rien changer au niveau de la couche application.

C'est ce principe d'interconnexion que met en oeuvre Internet. Il détache la notion de communication des détails technologiques des réseaux et cache ces détails technologiques à l'utilisateur.

2.3. L'interconnexion dans Internet

Le but d'Internet est de construire une interconnexion de réseaux qui supporte un service universel de communication.

Au sein des sous-réseaux interconnectés, les machines utilisent les primitives de communication propres au sous-réseau auquel elles sont connectées. Au dessus de ces primitives, on trouve de nouveaux programmes (au niveau de la couche réseau) qui cachent les détails technologiques de bas niveau et font apparaître l'ensemble des sous-réseaux interconnectés comme un grand réseau unique. C'est ce schéma d'interconnexion particulier que l'on appelle " Internetwork " ou " Internet ".

2.3.1. Les passerelles ou routeurs

Nous avons parlé jusqu'ici de sous-réseaux interconnectés mais nous n'avons pas encore précisé comment interconnecter deux sous-réseaux. Il est clair que la seule façon envisageable d'interconnecter deux sous-réseaux est d'utiliser un ordinateur particulier qui sera connecté à ces deux réseaux. Un simple lien physique tel qu'un cable par exemple ne pourrait en effet fournir le mécanisme d'interconnexion comme nous l'avons envisagé jusqu'ici. Ces ordinateurs, appelés **passerelle** ou **routeur**, doivent fournir un service minimum de livraison de paquets d'un sous-réseau à l'autre.

Pour mieux comprendre ce service de livraison, considérons la figure 4.1. présentant 3 sous-réseaux interconnectés par les passerelle G1 et G2.

Figure 4.1. : 3 sous-réseaux interconnectés par 2 passerelles.

Dans cette exemple, la passerelle G1 doit communiquer au sous-réseau N2 toutes les données issues du sous-réseau N1 et destinées aux sous-réseaux N2 et N3 et inversément. De

même la passerelle G2 doit communiquer au sous-réseau N2 toutes les données issues du sous-réseau N3 et destinées aux sous-réseaux N1 et N2. L'idée de passerelle peut paraître simple mais elle est néanmoins très importante. C'est elle en effet qui assure l'interconnexion dans Internet. De plus, elle implique deux autres notions importantes : l'adressage et le routage. Une passerelle se base en effet sur l'adresse du destinataire pour router le paquet correctement. Nous détaillerons ces deux notions plus loin.

2.3.2. Les sous-réseaux dans Internet

Nous avons décrit Internet comme un ensemble de sous-réseaux interconnectés et coopérants. Il est cependant important de comprendre que, du point de vue d'Internet, tous les sous-réseaux sont considérés de la même manière. Que ce soit un sous-réseau local Ethernet, un sous-réseau mondial X25 ou encore une simple ligne louée, ces systèmes sont tous considérés comme de simples réseaux. Ce point de vue est clairement mis en évidence dans la figure 4.2.

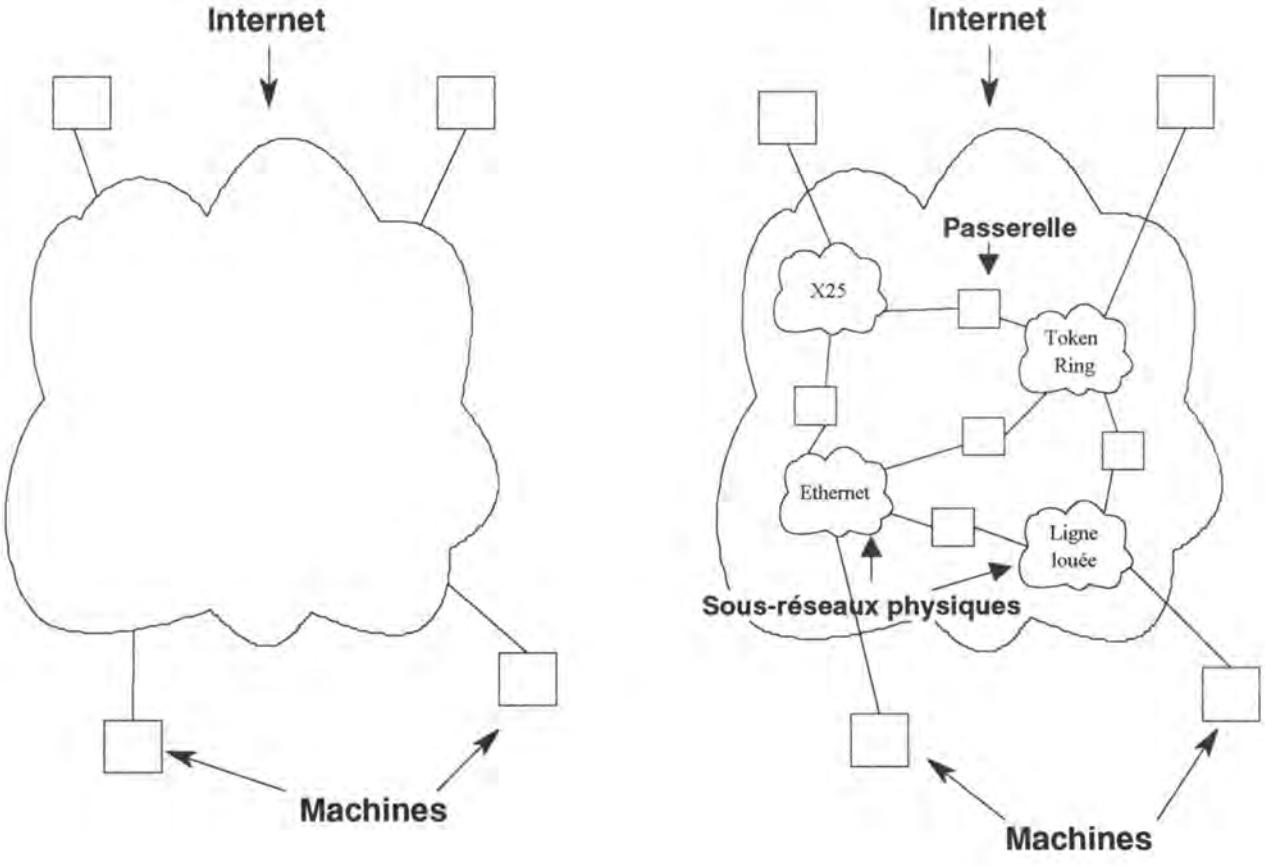


Figure 4.2. : Internet comme un réseau global et comme un réseau de sous-réseaux.

Dans la partie gauche de la figure nous pouvons voir Internet comme un réseau global sur lequel sont connectées des machines. Dans la partie droite, nous pouvons voir Internet comme un réseau de sous-réseaux. Notons que chacun des sous-réseaux est représenté par un nuage de même taille, ce qui traduit symboliquement l'égalité des réseaux au sein d'Internet.

3. L'adressage dans Internet

Dans le protocole IP original tel que défini dans la RFC[1] 791 [ISIUSC81], on définit le format des adresses et on les classifie suivant l'importance du réseau. L'adresse est sous la forme de 32 bits et est souvent représentée sous la forme de 4 nombres séparés par des points. Chaque adresse est unique sur le réseau c'est-à-dire qu'une adresse ne désigne qu'un seul hôte. On peut distinguer plusieurs classes d'adresses suivant le nombre de sous-réseaux permis par la classe et le nombre d'hôtes par sous-réseaux.

Les trois classes principales sont les classes A, B et C. On trouvera une représentation schématique des ces 3 classes dans la figure 4.3. Le format des adresses est sous la forme **Classe-Identité du réseau-Identité de l'hôte**.

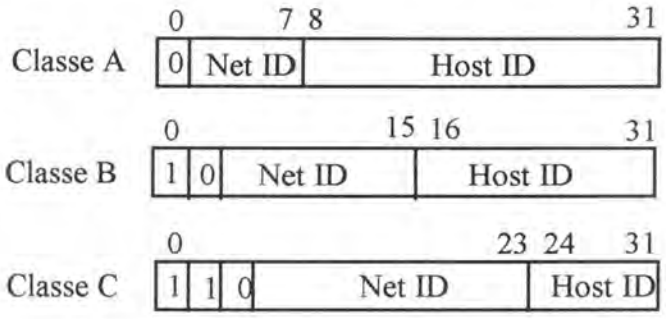


Figure 4.3. : Le format des adresses dans Internet

Si on reprend la notation de l'adresse IP sous forme de 4 nombres séparés par un point, par exemple a.b.c.d, on voit que la valeur des premiers bits impose des échelles de valeurs pour le premier nombre en fonction de la classe de l'adresse. Pour la classe A, comme le premier bit a une valeur imposée 0, le premier nombre ne peut prendre qu'une valeur entre 0 et 127. Pour la classe B, les deux premiers bits ayant respectivement les valeurs 1 et 0, le premier nombre ne peut prendre qu'une valeur entre 128 (128 + 0) et 191 (128 + 63). La même remarque est bien entendu d'application pour la classe C où le premier nombre ne peut prendre qu'une valeur entre 192 (192 + 0) et 223 (192 + 31).

Chaque classe admet un nombre limité de réseaux et un nombre limité d'hôtes sur chaque réseau. On trouvera dans le tableau 4.1. le nombre de sous-réseaux et le nombre d'hôtes acceptés par classe d'adresse[2].

[1] RFC : Request For Comments.

[2] Nous avons tenu compte dans ce tableau des adresses réservées, aussi bien pour le nombre d'hôtes que pour le nombre de réseaux.

Classe	Nombre de réseaux	Nombre d'hôtes
A	125	16777214
B	65534	65534
C	2097150	254

Tableau 4.1. : Le nombre de réseaux et d'hôtes par classe

Nous avons affirmé qu'une adresse IP est unique, mais nous n'avons pas dit qu'un hôte est identifié par une seule adresse IP. En effet, une même machine peut être connectée à plusieurs réseaux et peut donc posséder plusieurs adresses IP (c'est le cas des passerelles entre réseaux). Une adresse est en fait **l'identifiant d'une connexion** à un réseau.

Comment s'effectue le routage sur base de l'adresse IP ? Tout ce que les passerelles doivent extraire de l'adresse est le *network-id* ou identifiant du sous-réseau, qui est suffisant pour effectuer le routage. On peut remarquer que le *network-id* est toujours à la limite d'un octet, ce qui facilite une extraction efficace de cette information. Nous parlerons du routage plus en détail ultérieurement. On notera cependant que IP ne garantit pas une livraison correcte des datagrammes c'est-à-dire que IP ne procède à aucune vérification sur la bonne livraison d'un datagramme et sur l'exactitude des données du datagramme.

Les spécifications de IP définissent aussi une adresse de *broadcasting*. Le broadcasting est l'envoi d'un message simultanément à toutes les machines d'un même sous-réseau. La convention est alors de dire que tous les bits de l'*host-id* sont à 1. Il faut remarquer que le *broadcasting* peut se faire en une fois, comme sur les réseaux Ethernet (c'est-à-dire que Ethernet dispose d'une adresse destinée au *broadcasting*), ou alors hôte par hôte sur les réseaux qui ne bénéficient pas du *broadcasting*.

Le broadcasting peut se faire selon 2 formules : soit on connaît le *network-id*, soit on l'ignore. Dans le premier cas, on exécute un *directed broadcast* puisqu'on spécifie un réseau valide. Si on ne dispose pas du *network-id*, ce qui est par exemple le cas d'un hôte lors du startup, il reste à effectuer un *local network broadcast* en mettant tous les 32 bits à 1 dans l'adresse IP de destination. La règle générale qui prévaut est que TCP/IP limite le broadcasting au plus petit ensemble de machines possible.

Une autre particularité de l'adressage IP est que l'on peut interpréter un *host-id* ou un *network-id* égal à 0 comme signifiant "*this*" *network* ou *host*. Par exemple si une machine O veut envoyer un message à une machine D qu'elle sait être sur le même réseau mais que la machine O ne connaît pas l'adresse de son sous-réseau, il suffit que la machine O mette tous les bits de l'identifiant du sous-réseau à 0 pour spécifier que la destination est sur ce sous-réseau.

Une dernière particularité de d'adressage IP est que le *network-id* 127 est réservé au *loopback*, c'est-à-dire qu'une adresse de ce format référence l'hôte sur lequel il a été émis. Ce serait une erreur de trouver sur le réseau un paquet avec une adresse de destination de ce format.

Quelques faiblesses de IP sont :

- une machine qui change de sous-réseau doit aussi changer d'adresse IP

- les sous-réseaux qui doivent croître au-delà du nombre d'hôtes prévus doivent soit changer de classe (ce qui signifie une reconfiguration totale du réseau concerné), soit être coupé en 2 sous-réseaux (mais cela a des conséquences sur le routage).

- le point le plus critique dans IP vient du routage et de la constatation suivante : le chemin des paquets dépend de l'adresse utilisée. En particulier, on peut obtenir des choses "bizarres" pour les machines possédant plusieurs adresses IP.

- la taille actuelle des adresses, trop petite, ne permettra plus longtemps de faire face à l'expansion du réseau Internet.

C'est d'ailleurs quand les projections d'expansion du réseau Internet ont indiqué que l'espace d'adressage actuel deviendrait une contrainte limitative importante de l'expansion du réseau que l'IETF (Internet Engineering Task Force) a commencé à produire des efforts afin de sélectionner un successeur à IPv4. Ce successeur, en cours d'étude, est souvent appelé IPng pour "*IP next generation*" et maintenant IPv6.

Le comité chargé de l'étude de IPng a approuvé une série de critères techniques recueillis par RFC que IPng devra respecter. Nous nous bornerons ici à les décrire brièvement. Nous renvoyons le lecteur aux RFC 1726 [KASTEN94] et RFC 1752 [MANKIN95] pour une description plus détaillée.

Ces critères sont :

- un espace d'adressage suffisant : le protocole IPng devra pouvoir permettre au moins 10^9 sous-réseaux individuels.

- une topologie flexible : l'architecture de routage et les protocoles de IPng devront permettre l'utilisation de différentes topologies de réseaux.

- des performances acceptables : le trafic dans IPng devra se faire dans des vitesses au moins égales aux vitesses du trafic dans IPv4.

- un service robuste : le service de réseau et ses protocoles de routage et de contrôle devront être robustes.

- une bonne transition : IPng devra permettre une transition aisée et rapide depuis IPv4.

- une indépendance vis-à-vis des médias : le protocole devra être indépendant du type de sous-réseaux (LAN, MAN ou WAN) qui feront partie de l'interconnexion.

- un service de livraison de datagrammes non-assurés : le protocole, à l'image de IPv4, ne garantira pas une livraison correcte des datagrammes.

- une administration aisée : le protocole permettra de configurer un sous-réseau ou des hôtes ou d'effectuer des opérations sur ceux-ci facilement et de manière distribuée.

- des opérations sécurisées : le protocole devra fournir une couche réseau sécurisée.

- des noms uniques : IPng devra assigner à chaque objet situé au niveau de la couche IP un nom unique.

- un accès aisé à l'information : tout ce qui constituera la documentation de IPng et de ses protocoles devra être facilement accessible par les utilisateurs.. Cela se fera par la publication de cette documentation dans des RFC standards.

- le " multicast " : le protocole devra être capable de supporter le " multicast " aussi bien que " l'unicast ".

- l'extensibilité : le protocole devra être extensible afin de pouvoir faire face à des besoins futurs d'Internet. Cette extension devra pouvoir être faite sans devoir apporter de modifications aux applications existantes.

- un service réseau évolué : le protocole devra pouvoir associer des paquets avec des classes de service particulier et fournir les services spécifiés dans ces classes.

- un support à la mobilité : le protocole devra supporter des hôtes, des sous-réseaux ou des réseaux mobiles.

- un protocole de contrôle : le protocole devra fournir un support élémentaire afin de pouvoir tester et "debugger" des sous-réseaux ou des réseaux.

- la permission d'établir des réseaux privés : IPng devra permettre aux utilisateurs d'installer des réseaux, c'est-à-dire des interconnexions, privés basés sur l'infrastructure d'Internet.

Mais, en plus de respecter ces critères, IPng devra toujours vérifier les cinq principes de base suivants :

IPng devra présenter une architecture simple et laisser les fonctions de communication compliquées à la responsabilité d'autres niveaux de protocole.

IPng, à l'image d'Internet, sera un protocole de base sur lequel pourront s'appuyer toutes sortes de protocoles différents (existants ou non).

IPng devra avoir un long cycle de vie. Comme il n'est pas facile de changer un protocole qui est devenu un standard mondial, il est important de prévoir une longue période avant de devoir de nouveau changer complètement IPng; on estime que la vie de IPng devra s'étendre sur une période d'une vingtaine d'années.

En plus d'avoir une longue vie, IPng devra être prospère. Il n'est pas évident, en effet, que seuls l'agrandissement de l'espace d'adressage et une meilleure efficacité de routage suffiront à convaincre les utilisateurs de IPv4 à évoluer vers IPng. Il faudrait, en plus, que ce changement induise des bénéfices fonctionnels et opérationnels afin d'encourager les utilisateurs à changer.

Enfin, IPng devra conserver l'anarchie coopérative qui caractérise si bien Internet. Cette anarchie coopérative, qui n'est qu'une autre façon de parler de l'absence d'un point central de contrôle, permet en effet à l'ensemble des utilisateurs de coopérer efficacement entre eux et de répondre ainsi à des intérêts légitimes.

Depuis 1993, plusieurs successeurs à IPv4 ont été présentés mais actuellement, seules trois propositions ont été retenues par la "IPng area" de l'IETF :

- CATNIP ou Common Architecture for the Internet est une intégration d'un ensemble de propositions antérieures. Ce protocole fait particulièrement attention à la compatibilité avec les protocoles existant déjà.

- SIPP ou Simple Internet Protocol Plus est une nouvelle version de IP et n'est donc qu'une évolution de IPv4.

- TUBA ou The TCP/UDP over CLNP-Addressed Networks recherchent à minimiser le risque associé à la migration vers un nouvel espace d'adressage.

Nous renvoyons le lecteur à la RFC 1752 [MANKIN95] pour plus de détails.

4. Le routage dans Internet

Le routage des paquets IP consiste à choisir un chemin pour acheminer ceux-ci à leur destination. IP fournit un service de datagramme en mode non-connecté. L'algorithme de routage IP doit déterminer comment envoyer un datagramme au travers de plusieurs réseaux physiquement interconnectés.

Le routage dans IP est assez primitif dans le sens où il se base sur des hypothèses pour le chemin le plus court, et donc, qu'il ne tient pas compte de la charge réelle des liaisons mises en oeuvre, du type de service requis, de la longueur du datagramme, ... Pour bien percevoir le problème du routage, on doit revenir sur l'architecture des réseaux TCP/IP. On peut résumer la situation par la figure 4.4. Elle représente une configuration typique de plusieurs sous-réseaux interconnectés par des passerelles ou routeurs. Dans cette figure, les hôtes sont représentés par un H numéroté et les passerelles par un G numéroté.

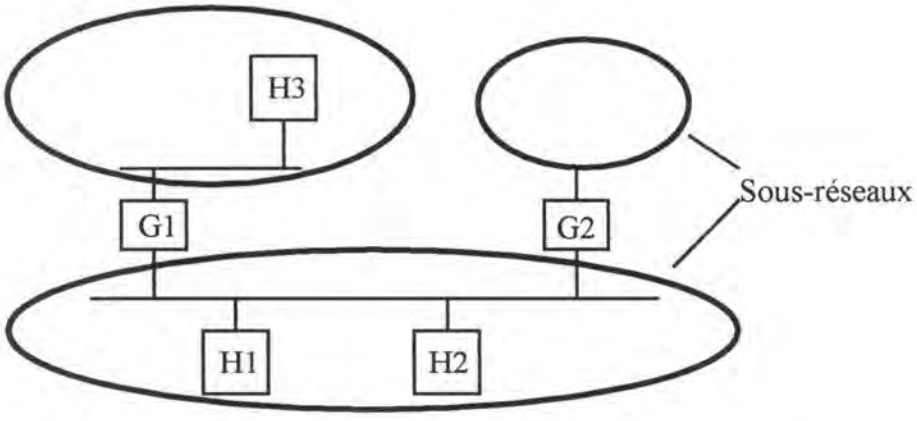


Figure 4.4. : Une architecture typique dans Internet

Dans le routage des datagrammes IP, les hôtes aussi bien que les passerelles ont un rôle spécifique. Les hôtes doivent déterminer où ils envoient leurs datagrammes, même s'ils n'ont qu'une seule connexion réseau (dans le cas où il y a deux passerelles sur leur réseau).

On peut diviser sommairement le routage en routage direct et routage indirect.

4.1. Le routage direct

Le routage direct est la base de la transmission de paquets dans TCP/IP. Il n'est possible que si les 2 entités communicantes sont sur le même sous-réseau et il ne fait pas intervenir les passerelles. Par exemple, les hôtes H1 et H2 de la figure 4.4 sont sur le même sous-réseau.

Il est important de signaler à cet endroit qu'un datagramme IP contient en particulier deux champs d'adresse : un pour l'adresse de l'origine et un pour l'adresse de la destination.

Le routage direct peut se résumer ainsi : le datagramme IP est encapsulé dans une trame correspondant au protocole propre au sous-réseau sous-jacent, l'adresse IP de destination est traduite en une adresse physique spécifique à ce sous-réseau et finalement, la trame ainsi formée est transmise directement par le sous-réseau sous-jacent au destinataire.

Comment un hôte peut-il savoir si la destination se trouve sur le même sous-réseau ? C'est très simple : un des composants de l'adresse IP est son *network-id*. Si un des *network-ids* de l'émetteur est le même que celui apparaissant dans l'adresse du destinataire, alors l'hôte se trouve sur un des sous-réseaux auxquels est connecté l'émetteur et on peut effectuer un routage direct.

4.2. Le routage indirect

Le routage indirect apparaît quand le destinataire n'est pas sur le même sous-réseau que l'émetteur et qu'il est donc nécessaire de passer par une ou plusieurs passerelles pour atteindre la destination.

Pour illustrer la technique mise en oeuvre par IP pour transmettre un datagramme par routage indirect, nous prendrons exemple du routage d'un paquet entre les hôtes H1 et H3 de la figure 4.4.

L'hôte émetteur H1 réalise que le destinataire ne se trouve pas sur le même sous-réseau. Il constate alors par consultation des tables de routage que l'hôte H3 est accessible par la passerelle G1. Il envoie, par routage direct, à la passerelle G1 le datagramme destiné à l'hôte H3.

La passerelle G1 recevant ce datagramme l'examine et détermine que le paquet IP ne lui est pas destiné, mais qu'il est destiné à l'hôte H3, connecté sur un des sous-réseaux auxquels elle a accès. Elle envoie alors le datagramme reçu par routage direct à l'hôte 3.

Dans le routage indirect, les datagrammes passent de passerelle en passerelle jusqu'à ce qu'ils puissent être transmis par routage direct. Le routage se fait en consultant une table de routage (*table-driven IP routing*). On trouve ce genre de tables aussi bien dans les hôtes que dans les passerelles. Que contiennent ces tables ?

Tout d'abord, on se rend compte qu'il est impossible de conserver une table avec le chemin complet menant à toutes les destinations possibles. On voudrait limiter la connaissance des hôtes au niveau local auquel ils appartiennent.

Le routage se fera sur base du *network-id* qui est commun à toutes les machines connectées à un même réseau. On conservera dans les tables de routage les *network-id* et les passerelles qui permettent de les atteindre. La table est organisée en paire **(N, G)** où N est le *network-id* et G est l'adresse IP de la passerelle intermédiaire. On désire aussi garder ces tables de routage assez petites sur les hôtes, préférant compter sur les passerelles pour assurer la plus grande partie du routage. Une technique utilisée est le routage par défaut vers une passerelle. Un routage spécifique par hôte est cependant toujours possible.

Ce routage sur base du *network-id* uniquement a plusieurs conséquences :

- dans la plupart des implémentations de IP, tout le trafic destiné à une machine passe toujours par le même chemin. Cela a pour corollaire, que même si plusieurs chemins existent, ils ne peuvent être utilisés simultanément. De plus, le routage se fait sans tenir compte de la charge des réseaux intermédiaires.

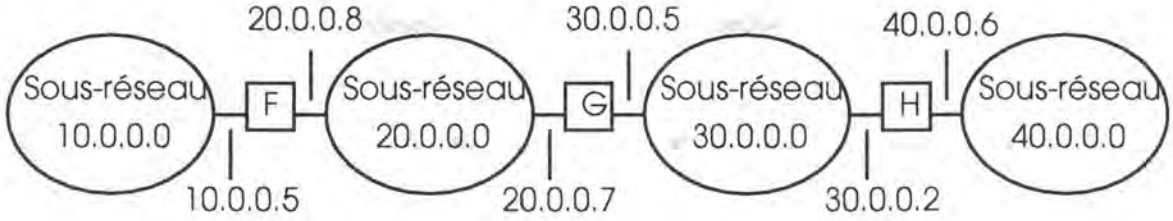
- comme seule la passerelle finale peut communiquer directement avec le destinataire, elle est aussi la seule à pouvoir dire si l'hôte est opérationnel ou non. On doit prévoir un mécanisme pour avertir l'émetteur en cas de difficulté.

- comme chaque passerelle effectue le routage indépendamment des autres passerelles, le chemin emprunté dans un sens ne sera peut-être pas le même dans l'autre sens. Il n'y a même pas de garantie qu'un chemin de retour existe !

Avant de présenter l'algorithme de routage, il est important d'expliciter la construction et la mise-à-jour des tables de routage ainsi que d'en donner un exemple.

Afin de construire ou de mettre à jour une table de routage, les passerelles s'échangent des informations quant à l'accessibilité de certains réseaux. Les informations sont échangées en respectant deux protocoles. Si ce sont deux passerelles situées sur deux sous-réseaux distincts qui s'échangent de l'information, c'est le protocole EGP (Exterior Gateway Protocol) qui est utilisé. Par contre, si les deux passerelles communicantes sont situées sur le même sous-réseau, c'est le protocole IGP (Interior Gateway Protocol) qui est mis en oeuvre. Les informations échangées sont alors utilisées par un algorithme afin de construire ou de mettre à jour les tables de routage.

Comme nous l'avons déjà précisé, une table de routage est formée de couples [N, G] où N est l'identifiant du sous-réseau à atteindre et G la prochaine passerelle à laquelle on doit transmettre le datagramme. La figure 4.5b représente un exemple simple d'une telle table de routage. La table présentée est celle que l'on retrouvera dans la passerelle G située entre les sous-réseaux 20.0.0.0 et 30.0.0.0 de la figure 4.5a.



(a)

Pour atteindre les sous-réseaux	Router vers cette adresse
20.0.0.0	Routage direct
30.0.0.0	Routage direct
10.0.0.0	20.0.0.8
40.0.0.0	30.0.0.2

(b)

Figure 4.5 : (a) une interconnexion de 4 sous-réseaux reliés par 3 passerelles et (b) la table de routage pour la passerelle G.

L'algorithme de routage est celui-ci :

<table>
<tr><td colspan="5">Extraction de l'adresse IP du destinataire, ID.</td></tr>
<tr><td colspan="5">Détermination du network-id du destinataire, IN.</td></tr>
<tr><td rowspan="5">ALORS
Encapsuler le datagramme dans une trame, transformer l'adresse IP de destination en adresse physique et transmettre la trame par le réseau sous-jacent.</td><td colspan="4">SI IN correspond à un réseau sur lequel on est directement connecté
SINON</td></tr>
<tr><td rowspan="4">ALORS
Effectuer le routage comme spécifié dans la table.</td><td colspan="3">SI il existe un routage spécifique à ID
SINON</td></tr>
<tr><td rowspan="3">ALORS
Transmettre le datagramme à la passerelle spécifiée.</td><td colspan="2">SI IN apparaît dans la table de routage
SINON</td></tr>
<tr><td colspan="2">SI une route par défaut est définie
SINON</td></tr>
<tr><td>Transmettre le datagramme à la passerelle par défaut.</td><td>Déclarer une erreur de routage.</td></tr>
</table>

5. Le didacticiel

Développer un didacticiel pour Internet représente un travail énorme tant les notions à voir sont importantes et pas toujours faciles à comprendre. C'est pourquoi le but que nous nous sommes fixés ne reprend pas toutes les notions d'Internet. Nous avons donc développé un didacticiel visant à faire comprendre à l'étudiant les principes fondamentaux d'adressage et de routage dans Internet.

Le but de ce chapitre n'étant que de donner une description sommaire d'Internet, nous renvoyons le lecteur au chapitre suivant pour une description complète du didacticiel.

Chapitre 5 :
Le didacticiel Tcplp

Après avoir introduit les notions principales propres à Internet dans le chapitre précédent, il nous reste maintenant à décrire en détail le didacticiel développé. Nous ferons pour cela largement appel aux différentes notions présentées dans les chapitres précédents. Ce chapitre est divisé en quatre sections.

La première section donne une description globale de l'application tant au niveau des notions abordées qu'à l'organisation de celles-ci pour faciliter l'assimilation de ces notions par l'étudiant. Cette première section peut donc être considérée comme la description de l'étape « Analyse pédagogique » de la démarche de conception du didacticiel. La deuxième section décrit ensuite en détail toutes les fonctionnalités que l'on rencontrera dans le logiciel ainsi que leur(s) effet(s) sur le fonctionnement du didacticiel. La troisième section quant à elle décrit l'interface homme-machine de l'application. Cette description sera accompagnée de la justification des choix faits lors du développement de cette interface. La quatrième section enfin fournit une description approfondie de l'architecture logicielle du didacticiel. On retrouvera notamment dans cette section une découpe en modules visant à faciliter la compréhension de l'organisation interne de l'application.

Bien que, comme nous l'avons déjà dit, les phases de réalisation de la maquette papier et de médiatisation aient été intimement liées lors du développement de l'application visée et, par conséquent, que la maquette papier n'ait pas été réalisée en une seule fois, ce chapitre peut être considéré comme la maquette papier de l'application terminée.

1. L'analyse pédagogique

Comme nous l'avons précisé plus haut, cette section peut être considérée comme la description de la phase « Analyse pédagogique » de la démarche de conception de notre didacticiel. Nous structurerons donc cette section de la même manière que la présentation de cette phase dans le premier chapitre de ce mémoire.

Rappelons que l'analyse pédagogique a pour but de déterminer deux choix : dire où on va et dire comment on y va [BESN88].

1.1. Dire où on va

1.1.1. L'objectif pédagogique

La première chose à déterminer pour dire où on va est l'objectif pédagogique à atteindre par l'apprenant. Dans notre cas, l'objectif pédagogique est la compréhension par l'étudiant du fonctionnement général du transit des données dans Internet.

Cet objectif principal sert, entre autres, à motiver l'élève et à évaluer l'efficacité de l'apprentissage car il représente le but à atteindre. Mais dans la phase de conception, dont l'analyse pédagogique est une étape, l'objectif pédagogique permet surtout au concepteur de déterminer un ensemble d'objectifs secondaires à maîtriser afin d'atteindre cet objectif pédagogique. Ceci permet également de structurer le contenu à transmettre.

Dans notre cas, les objectifs secondaires à maîtriser avant de comprendre le fonctionnement général d'Internet sont :

- **La compréhension de la notion d'interconnexion**
  Pour comprendre le fonctionnement du transit des données dans Internet, l'étudiant doit savoir et comprendre ce qu'est une interconnexion de sous-réseaux et, par conséquent, ce qu'est une passerelle.

- **La compréhension de la notion d'adressage**
  L'adressage de machines ou, pour être plus exact, l'adressage de connexion est une notion fondamentale dans le domaine des réseaux en général et dans le domaine d'Internet en particulier. C'est pourquoi une bonne compréhension de cette notion est indispensable pour l'atteinte de l'objectif pédagogique principal.

- **La compréhension de la notion de routage**
  Le routage est la notion principale du transit des données au sein d'Internet. L'étudiant devra donc maîtriser cette notion avec tout ce qu'elle implique (algorithme de routage, tables de routage, ...) pour assimiler de façon efficace les mécanismes de transit des données dans Internet.

- **La compréhension de la notion de datagramme**
  Le datagramme est la structure logique englobant toutes les données qui transitent dans Internet. Il paraît donc évident que c'est une notion critique à assimiler pour atteindre l'objectif principal. De plus, la notion

d'interconnexion de sous-réseaux implique des modifications (encapsulation, fragmentation,...) dans la structure du datagramme tout au long de son cheminement dans les sous-réseaux traversés. Nous pouvons donc ajouter à la notion de datagramme, la notion de modification des datagrammes durant leur transit dans Internet.

L'ensemble de ces objectifs secondaires est bien sûr loin de reprendre toutes les notions que l'on retrouve dans Internet. L'objectif du didacticiel n'étant pas de faire assimiler le fonctionnement d'Internet dans tous ses détails, on se limitera à ces notions de base qui suffiront amplement à faire comprendre le fonctionnement général du transit des données au sein d'Internet.

1.1.2. La population cible

Déterminer l'objectif pédagogique n'est pas tout, il faut également s'intéresser aux personnes auxquelles le didacticiel est destiné ainsi qu'à l'environnement dans lequel ces personnes évoluent. Dans le cas qui nous intéresse, la population cible est constituée d'étudiants de licence en informatique. L'âge des étudiants sera donc situé entre 20 et 24 ans et ces étudiants seront, du moins nous l'espérons, motivés dans l'apprentissage des notions proposées.

Cependant, ces informations ne sont pas suffisantes telles quelles afin de concevoir un didacticiel adapté de façon optimale aux caractéristiques de la population cible. On se basera donc sur la grille de facteurs utiles de la population cible [BESN88] fournie au chapitre 1 de ce mémoire afin d'obtenir des informations supplémentaires. On notera simplement que ne sont cités ici que les facteurs jugés disponibles et utiles à la conception de notre didacticiel.

- **Stade du développement cognitif**

  On fixera en pré-requis qu'un certain stade est atteint. On peut le faire sans prendre de grands risques étant donné qu'on a affaire dans notre cas à des étudiants de 20 à 24 ans ayant déjà réussi plusieurs épreuves d'études supérieures.

- **Capacité intellectuelle**

  On fixera également en pré-requis que ces capacités existent. La même remarque que pour le facteur précédent est également à faire à ce niveau.

- **Facteurs motivationnels et attitudes**

  La population cible étant constituée d'étudiants en informatique, ceux-ci sont en général motivés pour apprendre les notions abordées dans le didacticiel. Nous verrons toutefois plus loin que la forme d'interaction choisie (la simulation) attire l'attention d'étudiants moins motivés.

- **Facteurs didactiques au sens restreint**

  Le type de matériel disponible (que l'on précisera plus loin) et les préférences des étudiants pour les environnements graphiques couleurs ont résulté dans le choix d'une démarche pédagogique particulière qu'est le pilotage d'une simulation graphique du fonctionnement d'une interconnexion simple. On reparlera toutefois plus en détail de la démarche pédagogique choisie dans le point suivant.

- **Circonstances réelles (environnement)**
  L'apprentissage se faisant soit personnellement c'est-à-dire que l'étudiant est seul devant sa console, soit en auditoire c'est-à-dire que le professeur pilote la simulation en la commentant pour un groupe d'étudiants, une représentation simple et facile à comprendre doit être choisie afin que l'utilisateur du didacticiel ne soit pas tributaire du concepteur pour voir ce qui est représenté. Nous verrons lors de la description de l'interface homme-machine qu'une telle représentation a été choisie.

1.2. Dire comment on y va

On a parlé au premier chapitre de stratégie pédagogique pour préciser comment arriver à l'objectif principal [BESN88]. Dans cette stratégie, le concepteur essaie de combiner de façon optimale les ressources disponibles afin d'atteindre le but fixé. Les ressources sont divisées en quatre types : la ressource « matériel », la ressource « contenu », la ressource « élève » et la ressource « unité d'interaction » [BESN88].

1.2.1. La ressource « matériel »

La ressource « matériel » est l'environnement matériel dans lequel l'apprentissage se déroulera. Cet environnement peut être composé du tableau noir à l'ordinateur en passant par le rétroprojecteur, le projecteur de diapositives ou tout autre média. Les décisions à prendre à ce niveau sont donc la répartition des messages parmi les médias disponibles et la manière d'organiser les écrans dans le cas d'une utilisation de ce type de média.

Dans le cas qui nous intéresse on utilise un seul média, à savoir l'ordinateur, pour le déroulement de l'apprentissage. En effet, l'environnement matériel dans lequel le didacticiel a été développé et sera utilisé est composé de stations HP dotées d'écrans couleur haute-résolution avec éventuellement un projecteur permettant de visualiser l'image-écran sur un grand écran. Les caractéristiques techniques du matériel utilisé sont les suivantes :

Ordinateur :HP 9000/700.
Système d'exploitation : HP-UNIX.
Mémoire vive : 32 megabytes.
Mémoire disque : 2 gigabytes.
Périphériques d'entrée : souris, clavier.
Périphérique de sortie : écran graphique à haute résolution.
Langages de programmation : C et C++.

Notons simplement que l'utilisation de langages tels que le C et C++ pour la partie algorithmique et l'utilisation des librairies Xwindow pour la partie interface permet de porter l'application sur d'autres types de machines à la condition que ces machines soient dotées de ces langages et librairies.

La présence d'un média unique implique qu'un seul type de décision est à prendre à ce niveau; à savoir la façon de gérer l'écran, d'organiser l'utilisation des couleurs, etc. Bref, choisir la manière de concevoir l'interface homme-machine de l'application. La troisième section de ce chapitre étant consacrée entièrement à cet aspect, nous ne nous étendrons pas plus avant sur ce point ici.

1.2.2. La ressource « contenu »

Comme on l'a précisé au premier chapitre, le contenu est l'ensemble de la matière à transmettre afin d'atteindre l'objectif pédagogique. Dans le cas présent, l'objectif pédagogique est la compréhension du fonctionnement général du transit des données dans Internet. Pour atteindre cet objectif, l'étudiant doit maîtriser un certain nombre d'objectifs secondaires que nous avons déterminés au point 1.1.1. de ce chapitre; ces objectifs secondaires sont la compréhension des notions d'interconnexion, d'adressage, de routage et de datagramme.

Les objectifs secondaires permettent au concepteur de prendre les décisions adéquates en ce qui concerne la ressource contenu : on devra choisir certaines notions relatives aux objectifs secondaires retenus et on devra rejeter d'autres notions qui ne feraient qu'alourdir inutilement le didacticiel; une fois que les notions à maintenir sont choisies, il faut les organiser et les hiérarchiser afin de faciliter au maximum l'apprentissage.

Les objectifs secondaires cités plus haut permettent, dans notre cas, de choisir aisément les notions à présenter dans le didacticiel afin d'assurer un apprentissage efficace aux étudiants; Ces notions, décrites dans le chapitre précédent, sont :

- la notion d'interconnexion;
- la notion d'adressage;
- la notion de routage;
- la notion de datagramme.

Cette liste est loin de reprendre toutes les notions que l'on retrouve dans Internet, elle nous semble cependant largement suffisante pour assurer l'atteinte de l'objectif principal par les apprenants. Mais présenter ces notions ne suffit pas, il reste encore à les organiser afin d'optimiser l'apprentissage. Comme nous le verrons dans la suite, le type d'activité proposé à l'apprenant lui permet de gérer lui-même son apprentissage; nous avons toutefois organisé les notions abordées de telle manière que l'élève les rencontre dans l'ordre dans lequel elles ont été présentées ci-dessus : ces premières notions représentent en effet les bases nécessaires à l'assimilation de notions plus poussées.

1.2.3. La ressource « élève »

Dans le cadre du développement d'un didacticiel, cette ressource est très importante en ce sens que l'élève est le premier intéressé. C'est ici que l'analyse de la population cible est utilisée afin de définir une démarche pédagogique qui permettra à l'élève d'atteindre l'objectif pédagogique d'une part et le motivera d'autre part.

Les différents facteurs de la population cible que nous avons passés en revue au point 1.1.2. de ce chapitre nous ont amenés à choisir le pilotage d'une simulation graphique comme activité proposée à l'étudiant. Les premiers facteurs sur lesquels nous nous sommes basés pour choisir ce type d'activité sont les facteurs didactiques. C'est en effet en voyant le type de matériel dont nous disposons que l'idée de développer une simulation graphique nous est apparue opportune. Ensuite, d'autres facteurs tels que les facteurs motivationnels ou les circonstances réelles sont venus renforcer cette idée. Les facteurs motivationnels du fait que la forme d'interaction qu'est la simulation attire très fort l'attention des étudiants qui seraient éventuellement moins motivés par le sujet abordé; et les circonstances réelles du fait que la simulation permet de mettre en oeuvre facilement les deux types d'apprentissage envisagé :

l'apprentissage personnel (l'autoformation) ou l'apprentissage piloté par un enseignant devant un ensemble d'étudiants.

Après avoir choisi le type d'activité proposé aux étudiants, il reste à déterminer le déroulement des activités dans le didacticiel. Dans notre cas, le type d'activité proposé étant le pilotage d'une simulation graphique, le déroulement des activités sera directement fonction des choix faits par l'étudiant ou l'enseignant lors de leur manipulation du didacticiel. Ils pourront donc déterminer leur rythme de travail, privilégier certaines notions, etc., ce qui, d'une part, motivera d'autant plus l'étudiant dans son apprentissage dans le cas d'une situation d'autoformation ou, d'autre part, permettra à l'enseignant de structurer son cours comme il l'entend dans le cas d'une situation d'enseignement piloté par le professeur.

1.2.4. La ressource « unité d'interaction »

L'unité d'interaction est comme nous l'avons vu la structure de base de l'échange d'informations entre l'étudiant et l'ordinateur. Sa structure est représentée à la figure 1.1. du chapitre 1 et est composée de 4 phases : la sollicitation machine, l'entrée des données par l'élève en fonction de la sollicitation, la réaction de la machine aux données entrées par l'élève et la décision de branchement.

Les principales décisions que le concepteur doit prendre à ce niveau sont de choisir les formes d'interaction pour les activités proposées et d'articuler les unités d'interaction entre elles afin d'assurer un suivi logique de l'apprentissage. Dans le cas qui nous préoccupe, l'activité qui est proposée étant le pilotage d'une simulation graphique, la forme d'interaction qui s'impose est la forme à simulation dont le modèle sous-jacent est le modèle d'Internet. Le lecteur trouvera ci-dessous la description de l'établissement d'une simulation qui a pour but de montrer les articulations qui existent entre les différentes unités d'interaction présentes dans notre didacticiel.

Au départ de la manipulation du didacticiel, l'étudiant se trouve devant la représentation graphique d'une interconnexion simple (cfr. figure 5.5. de ce chapitre); l'ordinateur lui propose alors de sélectionner une machine qui sera la source de la communication; l'élève sélectionne une machine et celle-ci est mise en évidence à l'écran.

L'ordinateur passe ensuite à l'unité d'interaction suivante en proposant à l'étudiant de sélectionner une seconde machine qui sera la destination de la communication; l'élève la sélectionne et la machine est mise en évidence à l'écran.

L'ordinateur passe alors à la troisième unité d'interaction et propose à l'étudiant de lancer l'animation de la communication ou d'annuler ce cas de figure. Si l'étudiant décide de lancer l'animation, la réaction de l'ordinateur est de lancer l'animation de la communication; lorsque cette animation est terminée, l'ordinateur revient à la troisième unité d'interaction et propose à l'étudiant de relancer l'animation ou d'annuler le cas de figure. Si l'élève décide d'annuler le cas de figure, la réaction de l'ordinateur est de « désélectionner » les machines source et destination et de revenir à l'état de départ.

La structure de toutes les unités d'interaction décrites ci-dessus est considérablement enrichie par l'introduction entre les phases « sollicitation machine » et « entrée des données par l'élève en fonction de la sollicitation » d'un ensemble de fonctions supplémentaires permettant d'enrichir l'apprentissage. Ces différentes fonctions sont :

- **Le changement de vue**
  Cette fonction permet de changer la vue que l'on a de l'interconnexion. Deux vues sont possibles : une vue de l'interconnexion en tant que réseau de machines et une vue de l'interconnexion en tant que réseau de sous-réseaux.

- **L'appel d'une vue détaillée**
  Cette fonction permet d'obtenir une nouvelle fenêtre à l'écran dans laquelle est représentée une vue détaillée du sous-réseau dans lequel se trouve le datagramme.

- **L'appel de moniteurs**
  Cette fonction permet d'appeler deux fenêtres particulières. La première permet de suivre l'exécution de l'algorithme de routage lorsque celui-ci est exécuté et la deuxième fenêtre permet d'avoir une représentation de l'état courant du datagramme.

- **La « désélection »**
  Cette fonction permet de ne « désélectionner » qu'une des deux machines source ou destination.

On vient donc d'établir une stratégie pédagogique qui combine les quatre ressources que nous venons de passer en revue : elle propose un certain type d'activité (le pilotage d'une simulation) en se basant sur une forme d'interaction particulière (la forme à simulation) afin de faire assimiler un contenu (les notions retenues) à des élèves en utilisant un matériel bien défini (stations HP avec écran couleur). Notons cependant que le didacticiel développé ne constitue qu'un premier pas et qu'il servira de base pour le développement de didacticiels présentant des notions supplémentaires du domaine des télécommunications.

Maintenant que le contenu et la façon de le transmettre sont choisis, il reste à décrire en détail les fonctionnalités fournies par l'application et l'interface homme-machine développée dans le but de faciliter l'échange d'informations entre l'élève et l'ordinateur avant d'expliciter l'architecture logicielle proprement dite.

2. Les fonctionnalités présentes dans le didacticiel

Le didacticiel développé est un logiciel permettant la simulation du fonctionnement d'une interconnexion simple. Mais comme cette simulation est pilotée par l'étudiant, un certain nombre de fonctionnalités lui sont offertes afin qu'il se sente impliqué et surtout motivé dans l'apprentissage des diverses notions qui lui sont présentées.

Ces fonctionnalités vont être à présent passées en revue une à une afin que le lecteur puisse se faire une idée concrète de ce que propose le didacticiel. Sauf indication contraire, l'ordre dans lequel ces fonctionnalités sont présentées ne représente aucunement l'ordre dans lequel ces fonctionnalités doivent être appelées lors de l'utilisation du didacticiel; en effet, comme ce didacticiel a été développé dans un environnement Xwindow, celui-ci est piloté par des événements déclenchés par l'utilisateur, ce qui a pour conséquence que la plupart des fonctionnalités offertes peuvent être appelées à n'importe quel moment de l'exécution du programme. Chaque description est structurée en deux points : le premier point donne une description de la fonctionnalité et de son déclenchement tandis que le deuxième point

présentera les effets que la fonctionnalité provoque sur l'aspect et le fonctionnement du didacticiel.

2.1. La sélection d'une machine

Description

Avant de simuler le transit d'un datagramme entre deux machines, l'étudiant doit d'abord sélectionner les deux machines « actrices » de la communication : la machine source et la machine destination. Cette sélection se fait à l'aide de la souris en cliquant deux fois sur la représentation de la machine désirée. Un maximum de deux machines peut être sélectionné simultanément.

Effets induits

Le déclenchement de cette fonctionnalité a différents effets sur l'aspect et le fonctionnement de l'application. Le premier effet est la mise en évidence de la machine sélectionnée par un changement de couleur de sa représentation à l'écran. Le deuxième effet dépend de ce qu'on sélectionne la première machine ou la seconde. Si on sélectionne une première machine, le second effet est l'apparition d'une boîte de dialogue reprenant les principales caractéristiques de la machine (son nom, son adresse IP, et son type (hôte classique ou passerelle)) et présentant un bouton de « désélection » de la machine ainsi qu'un bouton d'annulation complète[1]. Par contre, si on sélectionne la machine destination, l'effet observé est l'agrandissement de la boîte de dialogue décrite ci-dessus afin de reprendre les caractéristiques de la machine destination et de présenter un nouveau bouton de « désélection » ainsi qu'un bouton supplémentaire permettant d'actionner l'animation graphique du transit du datagramme entre les deux machines.

Si l'utilisateur sélectionne, délibérément ou non, une passerelle comme machine première ou seconde, un effet supplémentaire est observé : c'est l'apparition d'une boîte de dialogue exclusive[2] lui demandant laquelle des deux connexions de la passerelle il désire utiliser lors de la simulation.

2.2. La « désélection » d'une machine

Description

On a parlé plus haut de boutons de « désélection » d'une machine, boutons qui se trouvent dans la boîte de dialogue apparaissant lors de la sélection des machines. Ces boutons, au nombre de deux, sont relatifs chacun à une seule des deux machines sélectionnées. Leur activation à l'aide de la souris permet de « désélectionner » la machine à laquelle ils font référence.

[1] Les fonctionnalités relatives à ces boutons seront décrites plus loin.
[2] c'est-à-dire que l'utilisateur doit absolument faire ce qui lui est indiqué dans cette boîte avant de pouvoir continuer la manipulation du didacticiel.

Effets induits

Cette fonctionnalité a également différentes conséquences sur l'aspect et le fonctionnement de l'application. Le premier effet est la neutralisation de la mise en évidence de la machine qui était sélectionnée par un retour à la couleur d'origine de sa représentation. Le second effet est l'effacement des caractéristiques de la machine « désélectionnée » dans la boîte de dialogue décrite plus haut.

2.3. L'annulation de toutes les sélections

Description

Cette fonctionnalité permet d'annuler la (ou les) sélection(s) faite(s). Il faut donc avoir sélectionner au moins une machine pour que la fonctionnalité soit accessible à l'utilisateur. Elle se déclenche par l'activation du bouton d'annulation situé dans la boîte de dialogue reprenant les caractéristiques des machines sélectionnées.

Effets induits

Les effets induits par le déclenchement de cette fonctionnalité sont nombreux : le premier est la neutralisation de la mise en évidence des machines encore sélectionnées (rappelons que celles-ci peuvent être « désélectionnées par le déclenchement de la fonctionnalité précédente); le second effet est la fermeture de la boîte de dialogue reprenant les caractéristiques des machines sélectionnées. Un dernier effet peut encore être induit si le déclenchement de cette fonctionnalité a lieu durant une animation; si c'est le cas, l'effet est l'arrêt immédiat de l'animation.

En conclusion, on peut dire que le déclenchement de cette fonctionnalité provoque un retour à l'état initial de l'application.

2.4. Le déclenchement de l'animation

Description

Si deux machines sont sélectionnées simultanément, un bouton supplémentaire apparaît dans la boîte de dialogue reprenant les caractéristiques des machines sélectionnées. L'activation de ce bouton à l'aide de la souris provoque le déclenchement de l'animation.

Effets induits

L'effet principal du déclenchement de cette fonctionnalité est le déroulement de l'animation graphique du transit d'un datagramme entre les deux machines sélectionnées en respectant les règles de fonctionnement d'Internet. Cela se traduit à l'écran par le mouvement d'un point de couleur sur les lignes reliant les deux machines sélectionnées en passant par les éventuelles passerelles intermédiaires. Un second effet est la mise en évidence et la neutralisation de la mise en évidence des éventuelles passerelles intermédiaires se trouvant sur la route que doit suivre le datagramme.

Lorsque l'animation est déclenchée, le bouton à l'origine de ce déclenchement se transforme en bouton « STOP ». Ce qui permet à l'utilisateur d'arrêter l'animation à tout moment. Si il arrête l'animation, le bouton « STOP » se transforme en bouton « RESTART » qui permet de redémarrer l'animation où elle s'était arrêtée et de retransformer le bouton « RESTART » en un bouton « STOP ».

2.5. Le changement de vue

Description

La vue est la façon dont est présentée l'interconnexion à l'écran; deux vues sont possibles : la vue de l'interconnexion comme un simple réseau de machine, cette vue ne montre aucun détail concernant les sous-réseaux; et la vue de l'interconnexion comme un réseau de sous-réseaux, cette vue donne une représentation de l'interconnexion sous forme d'un ensemble de sous-réseaux reliés entre eux par des passerelles. Il est important de noter qu'à ce niveau ne sont représentés que les machines et contours de réseaux, les lignes entre machines, elles, ne sont pas visibles ici. Le lecteur trouvera une représentation de ces deux vues aux figures 5.5. et 5.6. de ce chapitre.

Le déclenchement de cette fonctionnalité est provoqué par l'activation à l'aide de la souris d'un item de menu; deux items sont donc prévus : un item est présent pour la vue du réseau de machines et un autre item est présent pour la vue du réseau de sous-réseaux.

Effets induits

L'effet induit par le déclenchement de cette fonctionnalité est le basculement de la représentation de l'interconnexion vers la vue souhaitée si la vue demandée n'est pas celle qui est déjà représentée à l'écran. Si l'utilisateur demande la vue qui est déjà représentée à l'écran, aucun effet n'est à remarquer.

2.6. La demande d'une représentation détaillée d'un sous-réseau

Description

Il est possible d'obtenir une représentation détaillée du sous-réseau courant; le sous-réseau courant est soit le dernier sous-réseau dans lequel se trouvait le datagramme lors de la fin de la dernière animation; soit le sous-réseau dans lequel se trouve le datagramme durant une animation ou encore le sous-réseau comprenant la machine source lors de la sélection de celle-ci. La représentation détaillée d'un sous-réseau reprend la représentation de toutes les machines qui le composent et de toutes les lignes qui relient ces machines entre elles (cfr. figures 5.2., 5.3. et 5.4. de ce chapitre).

Le déclenchement de cette fonctionnalité est provoqué par l'activation à l'aide de la souris d'un item de menu.

Effets induits

L'effet direct du déclenchement de cette fonctionnalité est l'ouverture à l'écran d'une nouvelle fenêtre Xwindow dans laquelle est dessinée la représentation détaillée du sous-réseau

courant. L'utilisateur peut fermer cette fenêtre à tout moment par un « double-click » sur le coin supérieur gauche de la fenêtre ou par l'activation de l'item « CLOSE » du menu de la fenêtre Xwindow.

Si l'utilisateur déclenche cette fonctionnalité alors que la fenêtre est déjà présente à l'écran, aucun effet n'est à observer.

2.7. La demande d'un moniteur pour l'algorithme de routage

Description

L'utilisateur peut à tout moment demander qu'une fenêtre contenant la description écrite de l'algorithme de routage apparaisse à l'écran lorsque l'algorithme est exécuté dans le cadre du transit d'un datagramme dans l'interconnexion représentée. Cette fonctionnalité a un statut spécial du point de vue de son déclenchement : elle est en effet active ou non active; c'est-à-dire que si l'utilisateur a activé l'item de menu correspondant à cette fonctionnalité, il pourra observer le déroulement de l'algorithme de routage chaque fois que celui-ci sera exécuté; dans le cas contraire, si la fonctionnalité est désactivée, l'utilisateur ne verra pas l'algorithme de routage se dérouler.

Effets induits

Un premier effet sur l'aspect de l'item de menu est à observer : lorsque l'utilisateur active la fonctionnalité, la représentation en trois dimensions d'un petit bouton bascule vers une représentation d'un bouton enfoncé. La réciproque se produit lorsque l'utilisateur désactive la fonctionnalité.

Cependant l'effet principal de l'activation de cette fonctionnalité est que, lors de chaque animation qui aura lieu tant que la fonctionnalité restera active, chaque fois que l'algorithme de routage sera exécuté, la fenêtre contenant sa description écrite apparaîtra à l'écran et chacune des instructions exécutées sera mise en évidence par un changement de couleur.

Cette fonctionnalité peut être activée ou désactivée à tout moment. Si elle est activée, respectivement désactivée, durant l'exécution de l'algorithme, la fenêtre apparaîtra, respectivement disparaîtra, immédiatement.

2.8. La demande d'un moniteur pour le datagramme

Description

Le moniteur pour le datagramme est une fenêtre Xwindow que l'utilisateur peut appeler à tout moment pour étudier le format du datagramme et ses modifications lors de l'animation graphique de la simulation. Cette fonctionnalité est déclenchée par l'activation à l'aide de la souris d'un item de menu.

Effets induits

L'effet du déclenchement de cette fonctionnalité est l'apparition à l'écran d'une nouvelle fenêtre Xwindow dans laquelle se trouve une représentation graphique de l'état

courant du datagramme. L'utilisateur peut fermer cette fenêtre dès qu'il le désire. Si la fenêtre est déjà présente à l'écran lorsque l'utilisateur déclenche cette fonctionnalité, aucun effet n'est à remarquer.

Lorsque la fenêtre est ouverte, elle contient toujours une représentation de l'état courant du datagramme, c'est-à-dire que cette représentation change lorsque le format du datagramme change comme, par exemple, lors de l'encapsulation du datagramme IP dans une trame Ethernet. Dans ce cas, l'utilisateur voit une trame Ethernet dont la zone des données est occupée par le datagramme IP.

2.9. La demande d'un fond coloré pour la représentation des réseaux

Description

Cette fonctionnalité permet à l'utilisateur de demander une modification de la représentation des réseaux à l'écran. Quelle que soit la vue représentée, un réseau est toujours représenté par un cadre regroupant les machines que ce réseau contient; la modification engendrée par le déclenchement de cette fonctionnalité ne concerne pas le cadre mais la trame de fond de ce cadre.

Comme dans le cas de la demande d'un moniteur pour l'algorithme de routage, cette fonctionnalité est active ou non active. Si l'utilisateur active la fonctionnalité en « cliquant » sur l'item de menu correspondant, il obtiendra la représentation décrite par cet item. Si, par contre, il laisse désactivée ou s'il désactive cette fonctionnalité, il obtiendra une représentation de réseaux par défaut. Cette fonctionnalité est offerte afin que l'utilisateur puisse choisir la représentation qu'il préfère et ainsi travailler dans un meilleur environnement.

Effets induits

Le premier effet est identique à celui remarqué pour la demande du moniteur de l'algorithme de routage : selon que cette fonctionnalité est activée ou non, la représentation en trois dimensions d'un petit bouton au niveau de l'item de menu montre un bouton enfoncé ou non.

L'effet principal cependant est un effet sur l'aspect du didacticiel : si la fonctionnalité est activée, les cadres représentant les réseaux à l'écran auront un fond coloré tandis que si la fonctionnalité est désactivée les cadres représentant les réseaux auront un fond hachuré. Notons cependant que si la vue représentant l'interconnexion comme un réseau de sous-réseaux est la vue courante, chaque type de sous-réseau a un fond le différenciant des autres types de sous-réseaux présentés et ce quel que soit le type de fond (coloré ou hachuré) demandé par l'utilisateur.

2.10. La « désélection » automatique en fin d'animation

Description

Cette fonctionnalité permet à l'étudiant de demander que dès que l'animation qu'il a lancée se termine, les machines sélectionnées soient automatiquement « désélectionnées » et

que la boîte de dialogue reprenant les caractéristiques des machines source et destination soit automatiquement fermée.

Cette fonctionnalité est également soit activée, soit désactivée. Si elle est activée, après chaque animation, les machines sélectionnées sont désélectionnées et la boîte de dialogue décrite ci-dessus est fermée. Si elle est désactivée; l'utilisateur garde la main à la fin d'une quelconque animation.

Effets induits

Etant donné que cette fonctionnalité est active ou non, on a le même effet que pour la fonctionnalité précédente en ce qui concerne la représentation en trois dimensions du bouton se trouvant à côté de l'item de menu correspondant à la fonctionnalité : si la fonctionnalité est activée, on observe la représentation d'un bouton enfoncé et si elle est désactivée, on observe la représentation d'un bouton non enfoncé.

Si la fonctionnalité est activée, les effets principaux se produisent à la fin de chaque animation que lance l'utilisateur : les machines sélectionnées sont « désélectionnées » et la boîte de dialogue reprenant les caractéristiques des machines source et destination est fermée. Par contre, si la fonctionnalité est désactivée, aucun effet n'est à observer si ce n'est qu'il n'y ait aucun changement d'aspect et de fonctionnement du didacticiel lorsqu'une animation se termine.

2.11. La modification de la vitesse d'animation

Description

Cette fonctionnalité permet à l'utilisateur du didacticiel de modifier à tout moment la vitesse d'animation. Cette modification de vitesse se fait par la manipulation à l'aide de la souris ou du clavier d'un bouton glissant sur une échelle graduée de 1 à 100.

Effets induits

Le premier effet engendré par la manipulation du bouton décrit plus haut est le « glissement » de sa représentation à l'écran le long d'une échelle graduée. De plus, la valeur courante de la vitesse choisie est indiquée en permanence au dessus du bouton.

Si la manipulation de ce bouton se fait lorsqu'une animation est en cours, le changement de vitesse est répercuté directement dans l'animation, c'est-à-dire que le point de couleur représentant le datagramme se déplacera plus ou moins vite selon que l'utilisateur choisit une vitesse plus ou moins élevée. Si la fonctionnalité de demande du moniteur pour l'algorithme de routage est activée, la variation de vitesse est également répercutée sur la vitesse d'exécution de cet algorithme à l'écran.

Si l'utilisateur modifie la vitesse d'animation alors qu'aucune animation n'est en cours, le seul effet visible est l'effet de glissement du bouton décrit plus haut; mais la nouvelle valeur de vitesse choisie sera appliquée aux animations à venir.

2.12. L'appel de la fenêtre de logo

Description

Cette fonctionnalité permet à l'utilisateur d'obtenir une nouvelle fenêtre Xwindow dans laquelle se trouve le nom du didacticiel, sa date de production et l'endroit où il a été conçu. Cette fonctionnalité est déclenchée par l'activation à l'aide de la souris de l'item de menu correspondant.

Effets induits

Le seul effet engendré par le déclenchement de cette fonctionnalité est l'apparition à l'écran d'une nouvelle fenêtre Xwindow dans laquelle se trouve les renseignements cités plus haut. L'utilisateur peut fermer cette fenêtre à tout moment par l'activation du bouton « OK » se trouvant également dans la fenêtre Xwindow.

Le lecteur peut à présent se faire une idée précise de ce que l'étudiant peut faire en manipulant le didacticiel développé. Mais, comme nous l'avons déjà signalé, le didacticiel développé ne constitue pas un produit fini mais servira de base pour des extensions présentant des notions supplémentaires propres au domaine des télécommunications. Par conséquent, la liste des fonctionnalités présentée ci-dessus n'est constituée que des fonctionnalités présentes dans la version actuelle de notre produit. Il reste après cela à décrire en détail l'interface homme-machine et l'architecture logicielle. C'est l'objet des deux parties suivantes de ce chapitre.

3. Description de l'interface

Avant de décrire en détails l'architecture logicielle de notre didacticiel, il nous semble important de décrire l'interface homme-machine conçue pour celui-ci. Nous structurerons cette description en deux parties : la première décrira les représentations que nous avons adoptées et la seconde fournira une description des différentes fenêtres créées. Notons que pour le développement de l'interface, nous avons utilisé les notions présentées au chapitre 2 de ce mémoire.

3.1. Les représentations utilisées

L'objet du didacticiel étant de présenter une simulation graphique du fonctionnement d'une interconnexion simple, nous avons dû adopter une représentation pour les notions de machines, de sous-réseaux et d'interconnexions. Parmi les quatre types de représentation possible, nous avons choisi une représentation abstraite. Même si celle-ci nécessite un apprentissage préalable, elle nous semblait la plus valable d'un point de vue pédagogique. Le lecteur trouvera une explication plus détaillée de ce choix au chapitre 7 : choix et difficultés de conception du didacticiel.

3.1.1. La représentation d'une machine

Nous avons adopté deux représentations abstraites pour les machines : une pour les hôtes classiques et une pour les passerelles. La représentation d'une machine est en fait la traduction des couches des protocoles de communication qu'elle met en jeu. C'est pourquoi un hôte classique est représenté comme dans la figure 5.1. par un cylindre divisé en trois couches; couches qui représentent respectivement la couche IP, la couche transport et la couche application. Une passerelle, par contre, sera représentée comme dans la figure 5.1. par un cylindre d'une couche qui représente la couche IP qui est la seule mise en jeu par une passerelle, celle-ci ne s'occupant que de faire transiter des datagrammes IP d'un sous-réseau à l'autre.

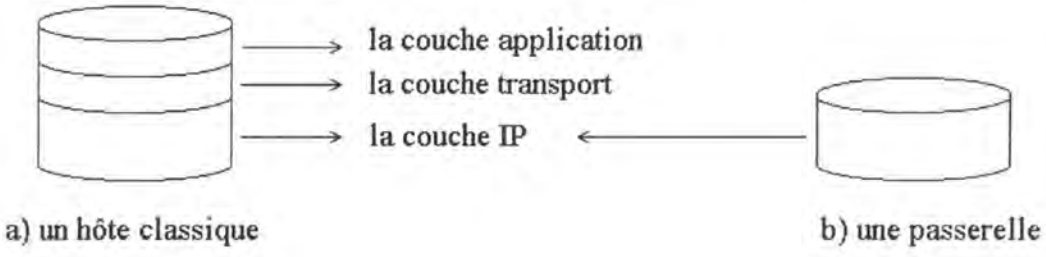


Figure 5.1. : La représentation d'un hôte classique et d'une passerelle.

3.1.2. La représentation d'un sous-réseau

La représentation d'un sous-réseau est la reprise de la représentation de toutes les machines appartenant à ce sous-réseau reliées par la représentation de lignes de communication. La représentation de ces lignes de liaisons dépend du type de sous-réseau impliqué. Une passerelle est représentée de façon incomplète dans la représentation d'un sous-réseau afin de traduire le fait qu'elle est une connexion vers l'extérieur.

La figure 5.2. montre la représentation d'un sous-réseau Ethernet. On peut voir l'ensemble des machines qui sont reliées à une ligne horizontale représentant le « backbone » du sous-réseau.

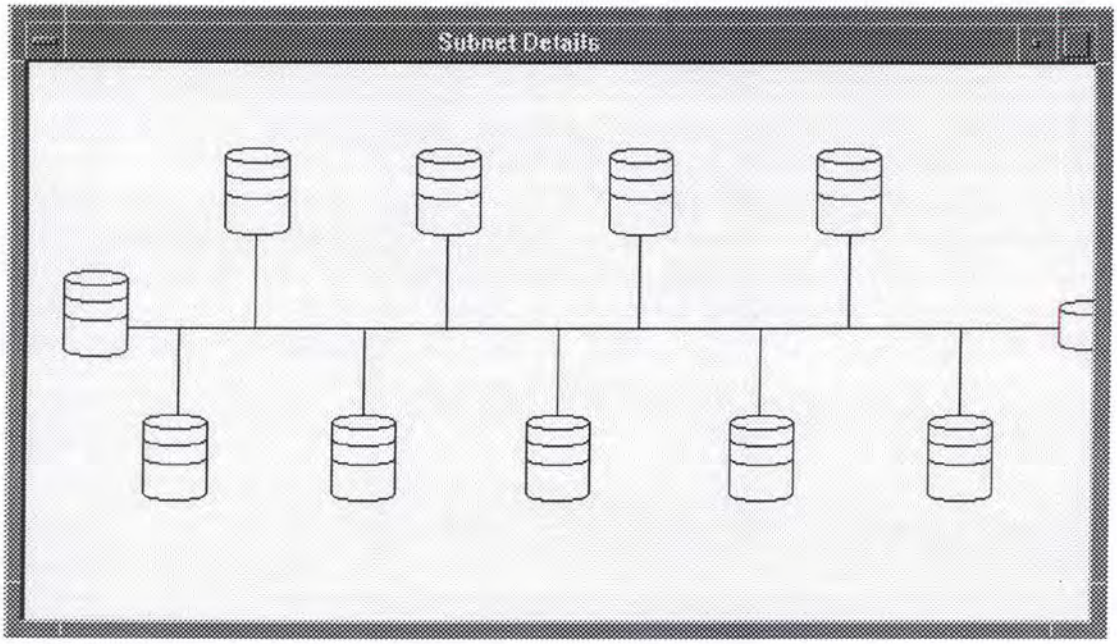


Figure 5.2. : La représentation d'un sous-réseau Ethernet.

La représentation d'un sous-réseau commuté, reprise dans la figure 5.3., comprend un ensemble de machines qui sont connectées au sous-réseau commuté lui-même représenté par la forme circulaire centrale.

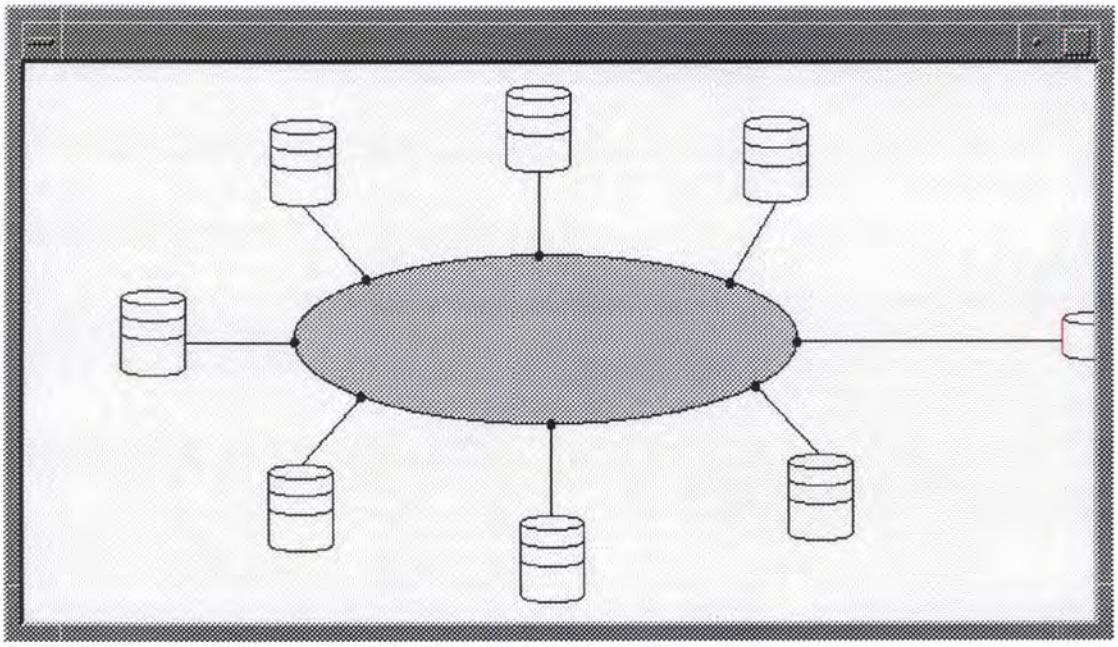


Figure 5.3. : La représentation d'un sous-réseau commuté.

La représentation d'une ligne louée ne reprend, comme dans la figure 5.4., que deux passerelles connectées entre elles par une simple ligne de liaison.

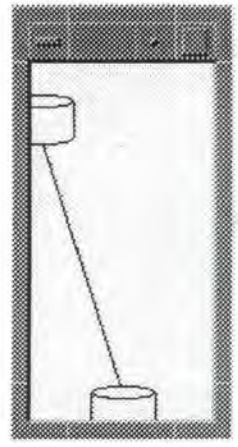


Figure 5.4. : La représentation d'une ligne de liaison.

3.1.3. La représentation d'une interconnexion

Deux représentations d'une interconnexion sont possibles en fonction du type de vue sélectionné par l'utilisateur. Si la vue de l'interconnexion comme un réseau de machines est sélectionnée, l'interconnexion est représentée, comme à la figure 5.5., sous la forme d'un ensemble de machines. Cette représentation permet à l'utilisateur de se rendre compte que lorsqu'on se trouve sur une machine reliée à Internet, on peut communiquer avec n'importe quelle autre machine connectée à Internet comme avec une machine située sur le même réseau physique.

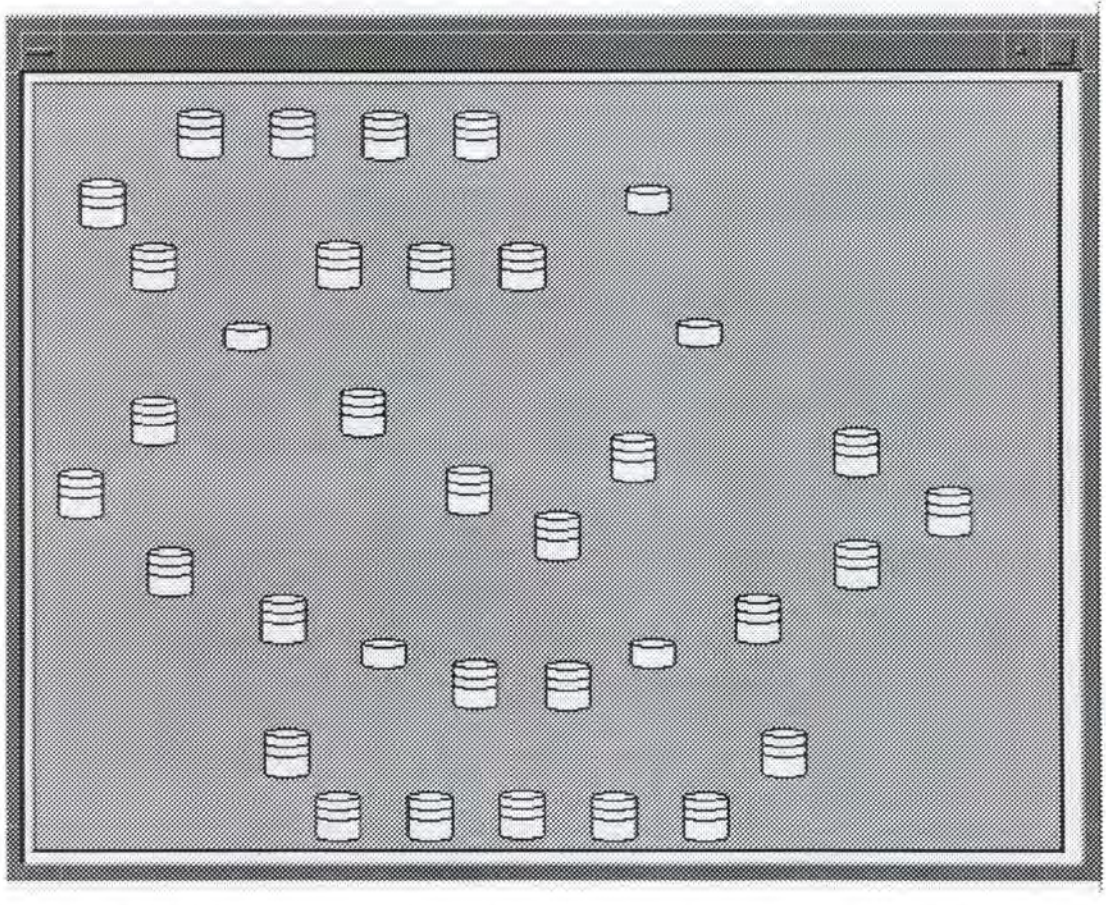


Figure 5.5. : La représentation d'une interconnexion comme un réseau de machines.

Par contre, si la vue de l'interconnexion comme un réseau de sous-réseaux est sélectionnée, l'interconnexion est représentée comme à la figure 5.6. sous la forme d'un ensemble de sous-réseaux de machines interconnectés entre eux par des passerelles.

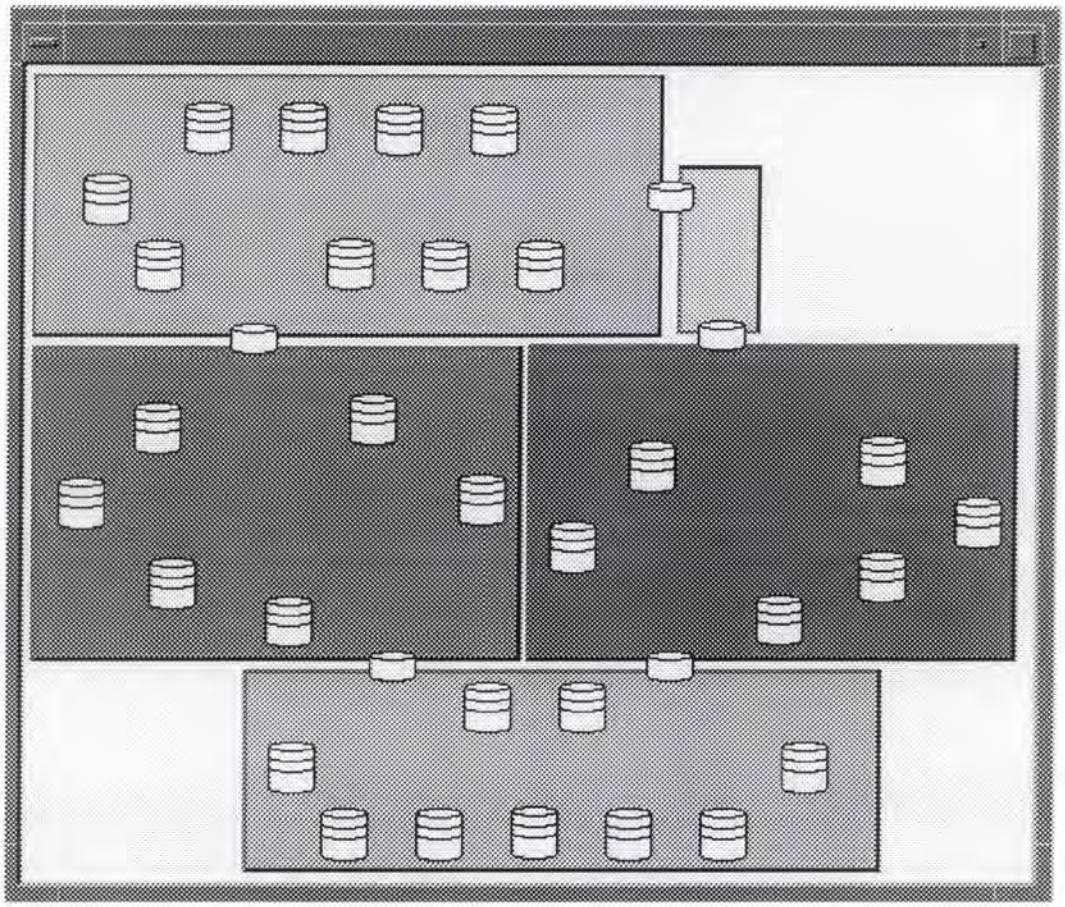


Figure 5.6. : La représentation d'une interconnexion comme un réseau de sous-réseaux.

3.2. La description des différentes fenêtres

Il nous reste à présent à décrire les différentes fenêtres créées pour notre interface. Nous avons en fait conçu une interface multi-fenêtrée; ce type d'interface nous permet de prévoir une fenêtre particulière pour chaque notion que l'apprenant désire aborder et ainsi de ne pas surcharger une fenêtre donnée.

3.2.1. La fenêtre principale

La fenêtre principale représentée à la figure 5.7., reprend les composantes principales du didacticiel. Cette fenêtre est découpée en quatre zones : la zone des menus, la zone de représentation de l'interconnexion, la zone des commentaires et la zone de contrôle de vitesse de la simulation. Ces zones resteront identiques tout au long de l'utilisation du didacticiel afin de ne pas perturber l'utilisateur.

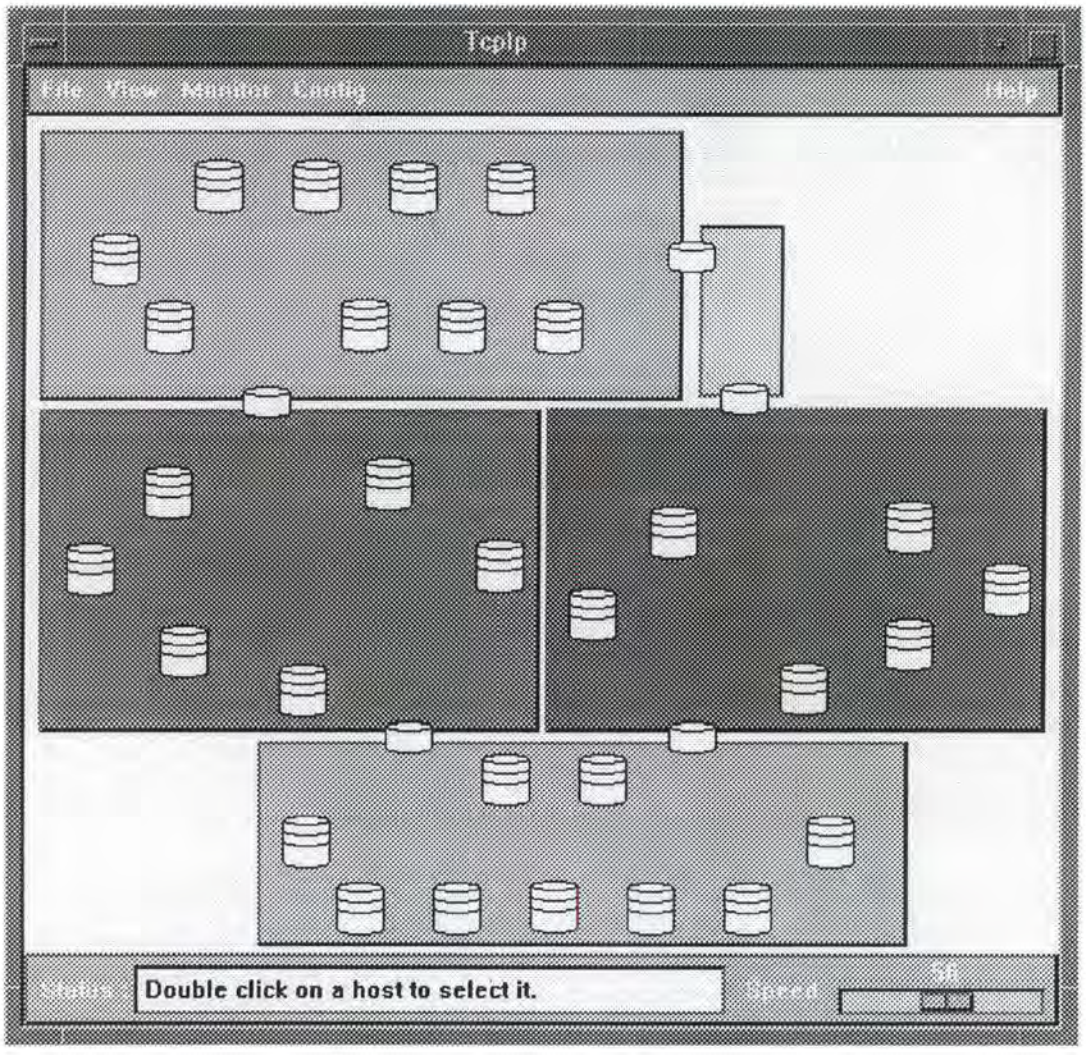


Figure 5.7. : La fenêtre principale.

A) **La zone des menus**

La zone des menus reprend la barre de menu principale. Les menus que l'on retrouve sur cette barre sont constitués d'items de menu correspondant à la plupart des fonctionnalités offertes par le didacticiel. Le menu « File » reprend l'item « Exit » qui permet de quitter le didacticiel. Le menu « View » reprend les différents items de menu qui permettent de changer la vue de l'interconnexion ou d'obtenir la vue détaillée d'un sous-réseau dans la fenêtre sous-réseaux décrite au point 3.2.3. de ce chapitre. Le menu « Monitor » est constitué des items de menu qui permettent d'obtenir les fenêtres « moniteurs » de l'algorithme de routage et du format du datagramme qui sont décrites respectivement aux points 3.2.4. et 3.2.5. de ce chapitre. Le menu « Config » reprend les items de menu permettant d'obtenir un fond coloré pour la représentation des sous-réseaux ou de sélectionner la fonctionnalité de « désélection » automatique des machines. Enfin, le menu « Help » contient l'item de menu permettant d'obtenir la fenêtre de logo décrite au point 3.2.6. de ce chapitre.

B) **La zone de représentation de l'interconnexion**

La zone de représentation de l'interconnexion reprend comme son nom l'indique une représentation de l'interconnexion. C'est dans cette zone que se déroulera l'animation relative

à la simulation graphique. C'est également dans cette zone que l'utilisateur devra sélectionner les machines qu'il désire voir communiquer lors de la simulation.

C) **La zone des commentaires**

C'est dans cette zone que le didacticiel indique en permanence à l'utilisateur ce qu'il peut faire ou dans quelle étape de la simulation il se trouve. Cette zone de commentaires a été placée dans le bas de l'écran en respect de règles ergonomiques qui concernent les messages. De plus, nous avons choisi une police de caractères bien lisible pour cette zone.

D) **La zone de contrôle de vitesse de la simulation**

Cette zone reprend l'échelle graduée qui permet à l'utilisateur de modifier à sa guise la vitesse de la simulation.

Du point de vue de l'utilisation des couleurs (cfr. chapitre 2, point 2.2.5.), n'étant pas des experts, nous avons prévu que toutes les couleurs utilisées dans le didacticiel puissent être personnalisées par l'utilisateur final. Enfin, le moyen d'interaction privilégié par ce didacticiel est la souris; c'est une conséquence directe de la dimension principalement graphique du didacticiel.

3.2.2. Les boîtes de dialogue

Deux boîtes de dialogues sont utilisées dans le didacticiel : une boite de dialogue reprend l'information des machines sélectionnées par l'utilisateur et est représentée à la figure 5.8.; une autre boîte de dialogue exige de la part de l'utilisateur qu'il choisisse une des deux connexions de la passerelle qu'il vient de sélectionner et est représentée à la figure 5.9.

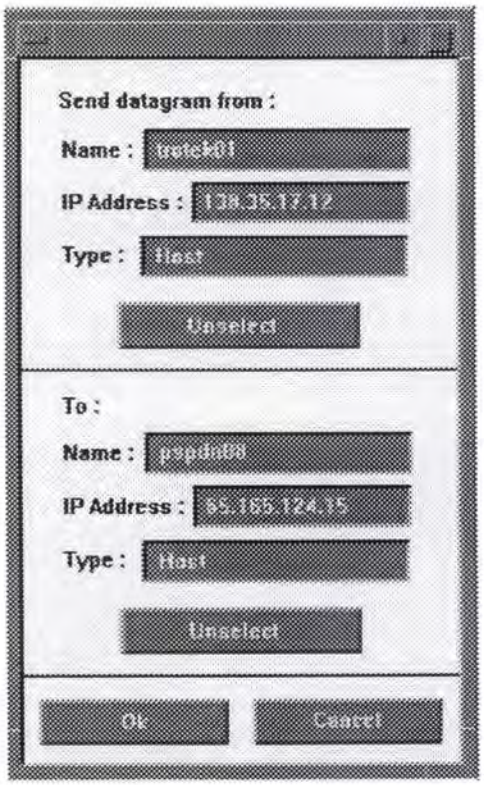


Figure 5.8. : La boîte de dialogue pour les machines sélectionnées.

C'est de la première boîte de dialogue que l'utilisateur lancera l'animation en activant le bouton « OK ». De plus, pour des raisons de cohérence intra-application, les renseignements d'une machine sélectionnée sont écrits dans la même couleur que la couleur de mise en évidence de cette machine.

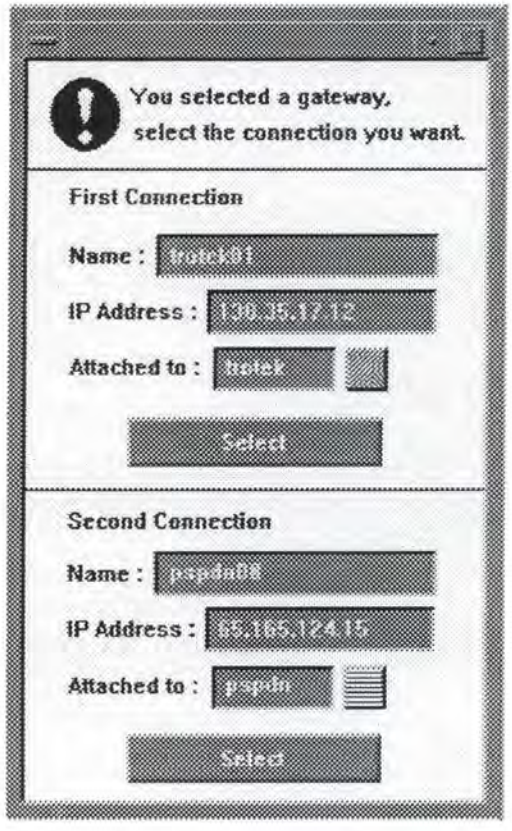


Figure 5.9. : La boîte de dialogue pour le choix d'une sélection.

Pour des raisons de respect de règles ergonomiques (cfr. chapitre 2), la taille des boutons correspondant au même type de fonctionnalités est identique pour chacun de ces boutons. Encore pour des raisons de cohérence intra-application (cfr. chapitre 2, point 1.1.2.), on remarque dans la seconde boîte de dialogue qu'à chacune des connexions est rattaché un témoin qui rappelle le fond du sous-réseau auquel la connexion appartient.

3.2.3. La fenêtre des sous-réseaux

La fenêtre des sous-réseaux qui est représentée à la figure 5.10. contient une représentation détaillée du sous-réseau courant. Cette fenêtre est ouverte lorsque l'utilisateur demande à voir cette vue particulière en activant l'item approprié du menu « View » de la fenêtre principale.

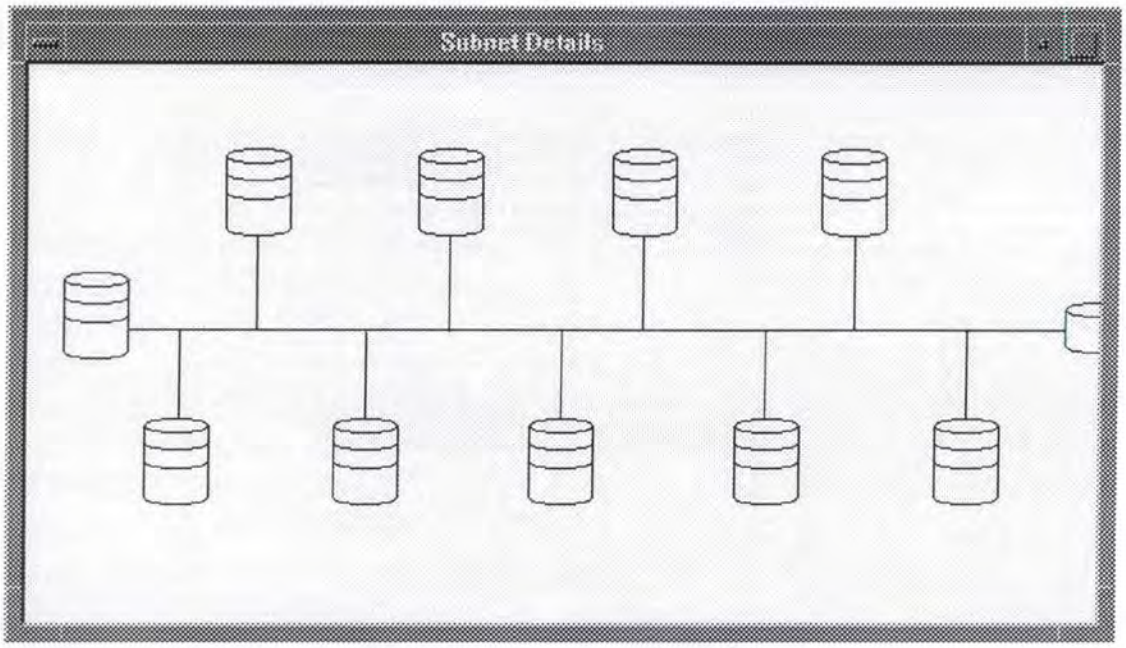


Figure 5.10. : La fenêtre des sous-réseaux.

Les seules actions que l'utilisateur peut exécuter sur cette fenêtre sont d'en modifier la taille ou de la fermer; cette fenêtre est en fait destinée à l'observation : lorsqu'une simulation est en cours, un point de couleur représentant le datagramme y est animé simultanément à l'animation de la fenêtre principale.

3.2.4. La fenêtre de l'algorithme de routage

La fenêtre de l'algorithme de routage qui est représentée à la figure 5.11. contient une description lexicographique de l'algorithme de routage. Cette fenêtre n'apparaît que si l'utilisateur a activé la fonctionnalité correspondante et si l'algorithme est en cours d'exécution lors d'une simulation. Cette technique est utilisée pour ne pas avoir une fenêtre perturbatrice en permanence à l'écran. Etant donné l'utilisation du texte de cette fenêtre, nous avons veillé à utiliser une police de caractères lisible.

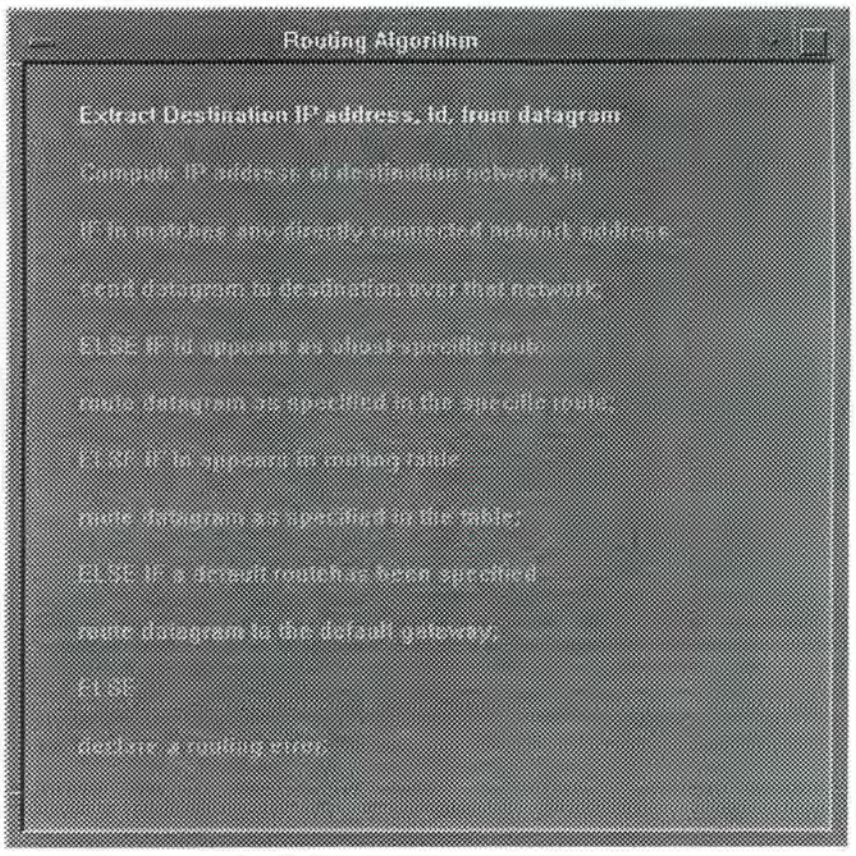


Figure 5.11. : La fenêtre de l'algorithme de routage.

Dans son état initial, cette fenêtre contient les instructions de l'algorithme de routage dans une couleur neutre vis-à-vis du fond de la fenêtre. Lors de l'exécution de l'algorithme, chacune des instructions qui est exécutée est mise en évidence par un changement de couleur vers une couleur « brillante » qui tranche vis-à-vis du fond de la fenêtre. Cette technique permet à l'utilisateur d'avoir son attention attirée directement vers les instructions exécutées, ce qui permet d'obtenir une meilleure représentativité du processus sous-jacent.

3.2.5. La fenêtre du datagramme

La fenêtre du datagramme représentée à la figure 5.12. contient une représentation de l'état courant du datagramme. Cette fenêtre est ouverte à l'écran lorsque l'utilisateur déclenche la fonctionnalité correspondante. La seule action que l'utilisateur puisse exécuter sur cette fenêtre est de la fermer; elle est, au même titre que la fenêtre des sous-réseaux, une fenêtre réservée exclusivement à l'observation.

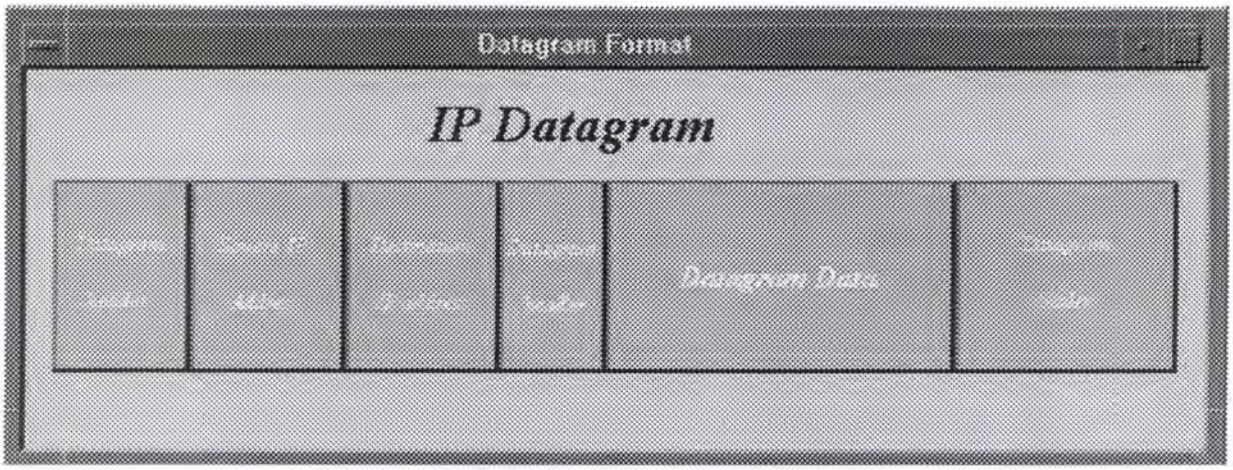


Figure 5.12. : La fenêtre du datagramme.

La représentation dessinée dans cette fenêtre varie en même temps que le format du datagramme lorsque celui-ci transite au sein de l'interconnexion représentée.

Pour des raisons de cohérence intra-application (cfr. chapitre é, point 1.1.2.), la partie IP du datagramme représenté dans cette fenêtre est représentée dans la même couleur que la couleur du point que représente le datagramme lors de la simulation graphique dans la fenêtre principale et dans la fenêtre des sous-réseaux.

3.2.6. La fenêtre logo

La fenêtre logo représentée à la figure 5.13., reprend les informations suivantes : le nom du didacticiel (TcpIp), le lieu de développement et la date de sortie de la version actuelle du didacticiel. Cette fenêtre apparaît lorsque l'utilisateur déclenche la fonctionnalité correspondant à l'activation de l'item « About » du menu « Help ».

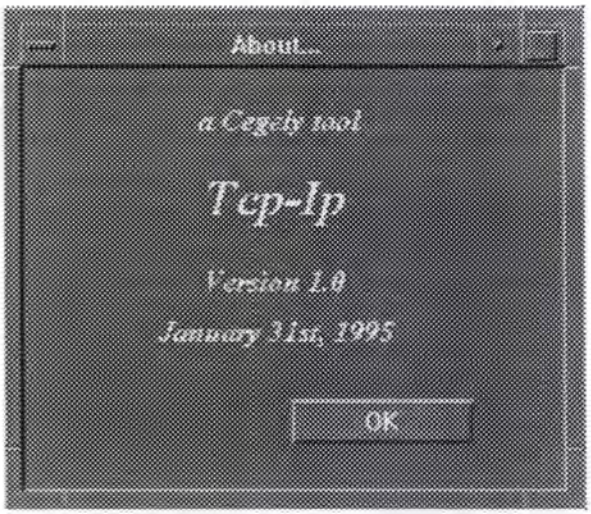


Figure 5.13. : La fenêtre logo.

Le seul but de cette fenêtre étant un but informationnel, la seule action que l'utilisateur puisse effectuer sur la fenêtre est de la fermer en activant le bouton « OK ».

4. L'architecture logicielle

Même si l'application développée est un didacticiel et que, par définition, son objet est l'enseignement, il n'en reste pas moins qu'un didacticiel est un logiciel et que le concepteur peut utiliser lors de son développement des techniques propres au monde des logiciels. C'est pourquoi cette partie fournit une description détaillée de l'architecture logicielle adoptée dans le cadre du développement du didacticiel.

Comme on l'a déjà précisé plusieurs fois, le problème est de créer un didacticiel visant à couvrir les principales notions du monde d'Internet. Au vu des processus à représenter, le coeur même de l'application est développé en langage orienté-objets; cette partie du logiciel sera donc écrite en C++ et est destinée à couvrir et à implémenter les notions de réseau, de sous-réseau, de lignes de communication entre machines, d'hôte et de passerelle. Cependant, une série d'outils[3] visant à faciliter la création d'interfaces ont été utilisés. Ces outils étant conçus pour générer des fichiers écrits en C, toute la partie interface et graphique du didacticiel est écrite en C s'appuyant sur les « Intrinsics » et sur « Xlib » de OSF Motif.

Bien que complexe, cette organisation est réalisable à condition de créer des liens logiques entre les parties C et C++ à l'aide du compilateur C++. Nous pouvons ainsi cumuler les avantages de l'environnement orienté-objets du C++ avec les facilités de création d'interface offertes par les outils écrits en C. On ne négligera pas non plus l'intérêt de la portabilité de l'application : le C et le C++ étant des langages réputés portables, les seules conditions pour que l'application soit portable vers d'autres plates-formes sont que le C, le C++ et OSF Motif soient installés sur cette plate-forme.

On trouvera plus de détails concernant ces notions de programmation dans les ouvrages suivants :

- [STROUST93] et [MEYERS92] en ce qui concerne la programmation sous C++.
- [YOUNG90a], [YOUNG90b], [OREILLY90a], [OREILLY89], [OREILLY90b], [OREILLY91] et [HPCOMP89] pour la programmation sous Xwindow et OSF Motif.

Au vue de l'organisation interne du didacticiel, la suite de cette section sera structurée en trois parties : la première partie décrira de façon détaillée les différents objets créés, la deuxième partie décrira ensuite les techniques d'implémentation de l'interface et la troisième partie fournira une découpe en modules de l'application visant à faciliter la compréhension de l'organisation globale du didacticiel.

4.1. Les objets

Comme il a été précisé plus haut, la partie de l'application développée en C++ vise à couvrir et à implémenter les notions de réseau, de sous-réseau, des lignes de communication entre machines, d'hôte et de passerelle; les objets créés dans ce but sont donc les objets « host », « link », « subnet » et « internet ». Comme on le verra dans la description détaillée de chacun d'eux, ces objets gèrent eux-mêmes leur tracé à l'écran ainsi que le calcul de leurs coordonnées en fonction de la grandeur de la fenêtre dans laquelle ils sont dessinés.

[3] Le lecteur trouvera une description détaillée de ces outils dans le chapitre suivant.

La description des différents objets sera composée d'une partie décrivant leur structure de données et d'une partie reprenant la description de leurs méthodes sous la forme « arguments-préconditions-postconditions ». La structure de cette description d'objet est basée sur les notions abordées au cours de MDL [DUBOIS93] du professeur E. Dubois.

4.1.1. L'objet « host »

L'objet « host » est appelé à représenter une machine : chacune de ses instances représentera donc une des machines présentes dans l'interconnexion dont on voit la représentation graphique à l'écran.

A) **Structure de données**

La structure de donnée de l'objet « host » se compose des variables suivantes :

- *IP_Add* : variable multivaluée représentant la ou les adresses IP d'une machine.
- *Net_Add* : variable multivaluée représentant la ou les adresses physiques d'une machine sur son ou ses sous-réseaux.
- *Name* : variable multivaluée représentant le ou les noms d'une machine.
- *Type* : variable monovaluée représentant le type d'une machine. Une machine peut être une passerelle ou un hôte classique.
- *x_sup, y_sup* : variables monovaluées représentant les positions relatives de la machine vis-à-vis des coordonnées graphiques du sous-réseau auquel elle appartient.
- *connexion* : variable multivaluée indiquant par quel(s) côté(s) la machine est reliée graphiquement à une ou des lignes de liaison entre machines.
- *Selected1, Selected2* : variables monovaluées indiquant si la machine est sélectionnée comme machine source ou comme machine destination.
- *h_width, h_height* : variables monovaluées indiquant respectivement la largeur et la hauteur absolue de la représentation de la machine à l'écran.
- *x_real, y_real* : variables monovaluées représentant les coordonnées absolues du coin supérieur gauche de la représentation de la machine à l'écran.
- *attached_to* : variable multivaluée pointant vers le ou les objets « link » représentant la ou les lignes auxquelles la machine est reliée.
- *belonged_to* : variable multivaluée pointant vers le ou les objets « subnet » représentant le ou les sous-réseaux auxquels la machine appartient.

On remarque que les quatre premiers paramètres correspondent aux caractéristiques IP de la machine tandis que les suivants sont des paramètres typiquement graphiques. On notera qu'une machine est toujours reliée à une ou deux lignes de communication et que toutes les variables multivaluées décrites ci-dessus auront une cardinalité strictement égale à ce nombre de liaisons.

B) **Spécifications des méthodes**

- La méthode « host » ou le constructeur

Arguments :	adresse IP, adresse physique, nom de la machine, type de machine, abscisse relative, ordonnée relative, connexion.
Préconditions :	aucune.

Postconditions : une nouvelle instance de l'objet « host » est créée et les paramètres correspondant aux arguments sont initialisés aux valeurs de ces arguments.

- La méthode « ~host » ou le destructeur

Arguments : aucun.
Préconditions : aucune.
Postconditions : l'instance de l'objet « host » à partir de laquelle le destructeur a été appelé n'existe plus. Tous les paramètres qui étaient des pointeurs ont été désalloués.

- La méthode « modify »

Arguments : abscisse relative, ordonnée relative, largeur, hauteur, connexion.
Préconditions : aucune.
Postconditions : les paramètres correspondant aux arguments sont modifiés ou initialisés aux valeurs de ces arguments.

- La méthode « modify »[4]

Arguments : type de machine, pointeur vers une autre instance de l'objet « host ».
Préconditions : aucune.
Postconditions : si l'instance de l'objet « host » n'est pas une passerelle, l'instance de cet objet est transformée en passerelle et tous les paramètres multivalués prennent comme deuxième valeur, les valeurs des paramètres de l'instance passée en argument. La valeur « TRUE » est retournée. Sinon rien ne se passe et la valeur « FALSE » est retournée.

- La méthode « trace »

Arguments : les caractéristiques d'une fenêtre graphique.
Préconditions : aucune.
Postconditions : la représentation de l'instance de l'objet est dessinée sur la fenêtre graphique dont les caractéristiques sont passées en arguments. La position et les dimensions de la représentation sont données par les paramètres de l'objet : x_real, y_real, h_width et h_height. Si l'instance de l'objet représente une machine sélectionnée, la représentation de cette instance est dessinée dans la couleur de mise en évidence correspondante.

- La méthode « recompute »

Arguments : la position et les dimensions absolues de la représentation du sous-réseau auquel la machine appartient; les dimensions de la fenêtre graphique dans laquelle l'instance de l'objet est représentée.
Préconditions : aucune.

[4] Les deux méthodes de même nom sont différenciées par le C++ en fonction de leurs arguments.

Postconditions : les dimensions absolues (h_width et h_height) de la représentation de l'instance de l'objet sont (re)calculées en fonction des dimensions de la fenêtre graphique dans laquelle l'instance de l'objet est représentée. La position absolue (x_real et y_real) de la représentation est calculée à partir de la position et des dimensions absolues de la représentation du sous-réseau passées en arguments et à partir des paramètres x_sup et y_sup donnant la position relative de la représentation de l'instance de l'objet vis-à-vis de la position et des dimensions du sous-réseau auquel elle appartient.

- La méthode « WhichParam »

Arguments : l'adresse IP d'une connexion.
Préconditions : aucune.
Postconditions : la méthode retourne un entier indiquant à laquelle de sa ou ses connexions correspond l'adresse IP passée en argument. Cela permet de choisir les paramètres désirés d'une passerelle.

- La méthode « IsXYInHost »

Arguments : des coordonnées de la fenêtre graphique dans laquelle est représentée l'instance de l'objet.
Préconditions : aucune
Postconditions : la méthode retourne un Booléen indiquant si les coordonnées passées en arguments se trouvent dans la surface délimitée par la représentation de l'objet. Cela permet de retrouver la machine que l'utilisateur veut sélectionner en appelant cette méthode à partir de toutes les instances existantes et en passant en arguments les coordonnées du « double-click ».

- La méthode « GetValue »

Arguments : une variable indiquant le paramètre désiré.
Préconditions : aucune.
Postconditions : cette procédure renvoie un pointeur vers le paramètre indiqué par la variable passée en argument.

On ne trouve jamais dans ces spécifications une précondition imposant que l'instance de l'objet existe. C'est une conséquence d'une caractéristique de la programmation orientée objets : ces fonctions étant des méthodes de l'objet, elles ne peuvent être appelées qu'à partir d'une instance de cet objet; et donc, le fait même que ces méthodes soient appelées impliquent directement l'existence d'une instance de l'objet.

4.1.2. L'objet « link »

L'objet « link » vise à représenter les lignes de liaison entre les machines. Cet objet est donc intimement lié à l'objet « host ».

A) **Structure de données**

La structure de données de cet objet est composée des variables suivantes :

- *IP_Add* : variable monovaluée représentant l'adresse IP de la ligne. Cette adresse est en fait l'adresse du sous-réseau auquel la ligne appartient.
- *x1, y1, x2, y2* : variables monovaluées représentant les coordonnées graphiques relatives des deux extrémités de la représentation de la ligne vis-à-vis des coordonnées graphiques de la représentation du sous-réseau auquel cette ligne appartient.
- *x1_real, y1_real, x2_real, y2_real* : variables monovaluées représentant les coordonnées absolues des deux extrémités de la représentation de la ligne à l'écran.
- *First_Host* : variable monovaluée pointant vers l'instance de l'objet « host » à laquelle est reliée la première extrémité de la ligne. Si cette extrémité n'est pas reliée à une machine, la variable vaut « NULL ».
- *Second_Host* : variable monovaluée pointant vers l'instance de l'objet « host » à laquelle est reliée la deuxième extrémité de la ligne. Si cette extrémité n'est pas reliée à une machine, la variable vaut « NULL ».

Le premier paramètre correspond à la caractéristique IP de la ligne de liaison alors que les suivants sont des paramètres graphiques.

B) **Spécifications des méthodes**

- La méthode « link » ou le constructeur

Arguments :	adresse IP, coordonnées relatives de la première extrémité, coordonnées relatives de la seconde extrémité, pointeur vers une instance de l'objet « host » reliée à la première extrémité, pointeur vers une instance de l'objet « host » reliée à la deuxième extrémité.
Préconditions :	aucune.
Postconditions :	une nouvelle instance de l'objet « link » est créée et les paramètres de l'objet correspondant aux arguments sont initialisés aux valeurs de ces arguments.

- La méthode « ~link » ou le destructeur

Arguments :	aucun.
Préconditions :	aucune.
Postconditions :	l'instance de l'objet « link » à partir de laquelle le destructeur a été appelé n'existe plus. Tous les paramètres qui étaient des pointeurs ont été désalloués.

- La méthode « modify »

Arguments :	coordonnées relatives des deux extrémités, pointeurs éventuels vers une instance de l'objet « host » reliée à la première ou à la deuxième extrémité de la ligne.
Préconditions :	aucune.

Postconditions : les paramètres correspondant aux arguments passés sont modifiés ou initialisés aux valeurs de ces arguments.

- La méthode « trace »

Arguments : les caractéristiques d'une fenêtre graphique.
Préconditions : aucune.
Postconditions : la représentation de l'instance de l'objet est dessinée sur la fenêtre graphique dont les caractéristiques sont passées en arguments. Les coordonnées des deux extrémités de la ligne sont données par les paramètres de l'objet x1_real, y1_real, x2_real et y2_real.

- La méthode « recompute »

Arguments : la position et les dimensions absolues de la représentation du sous-réseau dans lequel la ligne se trouve, le type de ce sous-réseau et éventuellement le numéro logique de la ligne.
Préconditions : aucune.
Postconditions : les coordonnées absolues des extrémités de la représentation de l'instance de l'objet ont été recalculées en fonction des paramètres passés en arguments.

- La méthode « GetValue »

Arguments : une variable indiquant le paramètre désiré.
Préconditions : aucune.
Postconditions : cette méthode renvoie un pointeur vers le paramètre indiqué par la variable passée en argument.

4.1.3. L'objet « subnet »

L'objet « subnet » représente l'implémentation de la notion de sous-réseau. En plus d'avoir ses caractéristiques propres, cet objet est intimement lié aux notions de machine implémentée par l'objet « host » et de ligne de liaison implémentée par l'objet « link ».

A) **Structure de données**

La structure de données de cet objet reprend les variables suivantes :

- *IP_Add* : variable monovaluée représentant l'adresse IP du sous-réseau. Si on se rappelle des notions présentées au chapitre 4, cette adresse permet de déduire la classe du sous-réseau grâce à son premier byte.
- *Protocol* : variable monovaluée représentant le protocole du sous-réseau. Dans l'état actuel du didacticiel, les protocoles supportés sont : Ethernet, Réseau à commutation de paquets, réseau à commutation de circuit et la simple ligne louée.
- *Routing_Table* : variable monovaluée organisée sous forme de tableau qui représente la table de routage propre au sous-réseau. On a placé la table de routage à ce niveau pour ne pas devoir la placer dans chacune des machines du sous-réseau et ainsi éviter une redondance de cette information en mémoire vive.

- *Host_Nbr* : variable monovaluée représentant le nombre de machines présentes sur le sous-réseau.
- *x_fact, y_fact* : variables monovaluées représentant les coordonnées relatives du coin supérieur gauche de la représentation du sous-réseau à l'écran.
- *s_width_fact, s_height_fact* : variables monovaluées représentant la largeur et la hauteur relatives de la représentation du sous-réseau à l'écran.
- *x_sup, y_sup* : variables monovaluées représentant les coordonnées absolues du coin supérieur de la représentation du sous-réseau à l'écran.
- *s_width, s_height* : variables monovaluées représentant la largeur et la hauteur absolues de la représentation du sous-réseau à l'écran.
- *sens* : variable monovaluée qui, dans le cas où le sous-réseau impliqué est une ligne louée indique si la représentation à l'écran est horizontale ou verticale.
- *First_Host, Last_Host* : variables monovaluées pointant respectivement vers le premier et le dernier élément d'une liste chaînée de pointeurs vers les instances de l'objet « host » appartenant au sous-réseau.
- *First_Link, Last_Link* : variables monovaluées pointant respectivement vers le premier et le dernier élément d'une liste chaînée de pointeurs vers les instances de l'objet « link » appartenant au sous-réseau.

Les quatre premiers paramètres de cette structure de données correspondent aux caractéristiques IP du sous-réseau alors que les paramètres suivants sont des paramètres graphiques.

B) **Spécifications des méthodes**

- La méthode « subnet » ou le constructeur

Arguments : adresse IP, protocole, nom standard des machines du réseau, sens de présentation, coordonnées relatives et dimensions relatives.
Préconditions : aucune.
Postconditions : une nouvelle instance de l'objet « subnet » est créée et les paramètres de l'objet correspondant aux arguments sont initialisés aux valeurs de ces arguments. De plus, toutes les instances des objets « host » et « link » composant le sous-réseau sont également créées et rattachées à l'instance de l'objet « subnet » par les listes chaînées représentées par les paramètres First_Host, Last_Host, First_Link et Last_Link.

- La méthode « ~subnet » ou le destructeur

Arguments : aucun.
Préconditions : aucune.
Postconditions : l'instance de l'objet « subnet » à partir de laquelle le destructeur a été appelé n'existe plus. Toutes les instances des objets « host » et « link » composant le sous-réseau ont également été détruites.

- La méthode « recompute »

Arguments : aucun.

Préconditions : aucune.
Postconditions : les coordonnées et dimensions absolues de la représentation de l'objet ont été recalculées à partir des coordonnées et dimensions relatives de celui-ci et à partir des dimensions absolues de la fenêtre graphique dans laquelle la représentation se trouve. Cette méthode a également appelé les méthodes « recompute » de toutes les instances des objets « host » et « link » reliées au sous-réseau.

- La méthode « trace »

Arguments : les caractéristiques d'une fenêtre graphique.
Préconditions : aucune.
Postconditions : les représentations de l'instance de l'objet « subnet » et des instances des objets « host » et « link » qui lui sont reliées sont dessinées sur la fenêtre graphique dont les caractéristiques sont passées en arguments. Le dessin des instances des objets « host » et « link » a été provoqué par l'appel de la méthode « trace » de ces instances.

- La méthode « host_search »

Arguments : une variable entière.
Préconditions : aucune.
Postconditions : la méthode retourne un pointeur vers l'instance de l'objet « host » dont le numéro logique correspond à la variable entière passée en argument. Si cette instance n'existe pas, la valeur « NULL » est retournée.

- La méthode « host_search »[5]

Arguments : une adresse IP.
Préconditions : aucune.
Postconditions : la méthode retourne un pointeur vers l'instance de l'objet « host » dont l'adresse IP correspond à l'adresse passée en argument. Si cette instance n'existe pas, la valeur « NULL » est retournée.

- La méthode « merge_host »

Arguments : une première variable entière, un pointeur vers une autre instance de l'objet « subnet », une deuxième variable entière.
Préconditions : l'instance de l'objet « host » qui est reliée à l'instance de l'objet « subnet » et dont le numéro logique correspond à la première variable entière passée en argument existe et n'est pas une passerelle. L'instance de l'objet « host » qui est reliée à l'instance de l'objet « subnet » passée en argument et dont le numéro logique correspond à la deuxième variable entière passée en argument existe et n'est pas une passerelle.

[5] Les deux méthodes de même nom sont différenciées par le C++ en fonction de leurs arguments.

Postconditions : une première instance de l'objet « host » (reliée à cette instance de l'objet « subnet » et dont le numéro logique correspond à la première variable entière des arguments) et une deuxième instance de l'objet « host » (reliée à l'instance de l'objet « subnet » passée en argument et dont le numéro logique correspond à la seconde variable entière des arguments) sont fusionnées en une seule instance qui devient de type « passerelle ».

- La méthode « construct_routing_table »

Arguments : une table de routage contenant le numéro logique des passerelles à atteindre lors du routage d'un datagramme.
Préconditions : aucune.
Postconditions : la table de routage de l'instance de l'objet « subnet » est construit. Dans la table de routage construite, les numéros logiques de la table de routage passée en argument ont été remplacés par les adresses IP des instances de l'objet « host » que ces numéros logiques désignaient.

- La méthode « GetValue »

Arguments : une variable indiquant le paramètre désiré.
Préconditions : aucune.
Postconditions : cette méthode renvoie un pointeur vers le paramètre indiqué par la variable passée en argument.

4.1.4. L'objet « Internet »

L'objet « Internet » est l'implémentation de l'entièreté de l'interconnexion présentée à l'écran ; c'est l'objet qui coordonne donc l'ensemble des instances représentées à l'écran.

A) **Structure de données**

La structure de données de cet objet est composée des variables suivantes :

- *SubNet_Nbr* : variable monovaluée représentant le nombre de sous-réseaux qui composent l'interconnexion.
- *First_Subnet, Last_Subnet* : variables monovaluées pointant respectivement vers le premier et le dernier élément d'une liste chaînée de pointeurs vers les instances de l'objet « Subnet » composant l'interconnexion.

Cet objet ne servant que de coordinateur, il est normal que l'on ne retrouve que des paramètres visant à pouvoir gérer l'ensemble de l'interconnexion.

B) **Spécification des méthodes**

- La méthode « Internet » ou le constructeur

Arguments : une liste chaînée de pointeurs vers des instances de l'objet « Subnet ».

Préconditions : les instances de l'objet « Subnet » ont été créées (cela se fait lors de l'initialisation).

Postconditions : une nouvelle instance de l'objet Internet est créée et les paramètres de l'objet correspondant aux arguments sont initialisés aux valeurs de ces arguments.

- La méthode « ~Internet » ou le destructeur

Arguments : aucun.

Préconditions : aucune.

Postconditions : l'instance de l'objet « Internet » à partir de laquelle le destructeur a été appelé n'existe plus. Toutes les instances de l'objet « Subnet » composant l'interconnexion ont également été détruites.

- La méthode « recompute »

Arguments : aucun.

Préconditions : aucune.

Postconditions : les coordonnées et dimensions absolues de toutes les instances des objets « Subnet », « post », « link » ont été calculées. Cette méthode ne fait qu'appeler les méthodes « recompute » des instances de l'objet « Subnet » composant l'interconnexion.

- La méthode « trace »

Arguments : les caractéristiques d'une fenêtre graphique.

Préconditions : aucune.

Postconditions : La représentation de l'interconnexion est dessinée à l'écran en respect de la vue souhaitée par l'utilisateur. Cette méthode appelle les méthodes « trace » des instances de l'objet « Subnet » composant l'interconnexion.

- La méthode « WhichHost »

Arguments : des coordonnées absolues situées dans la fenêtre graphique contenant la représentation de l'interconnexion.

Préconditions : aucune.

Postconditions : si les coordonnées passées en argument se trouvent dans la surface délimitée par la représentation d'une instance de l'objet « host », un pointeur vers cette instance est renvoyé par la méthode. Sinon la méthode renvoie la valeur « NULL ». Cette méthode permet d'obtenir un pointeur vers l'instance de l'objet « host » représentant la machine que l'utilisateur désire sélectionner : les coordonnées passées en arguments étant les coordonnées du point où l'utilisateur « double-clique » à l'aide de sa souris.

4.1.5. Les relations entre les objets.

Afin de terminer la description des objets créés, il est intéressant de montrer la relation qui existe entre ceux-ci. La figure 5.14. indique ces relations.

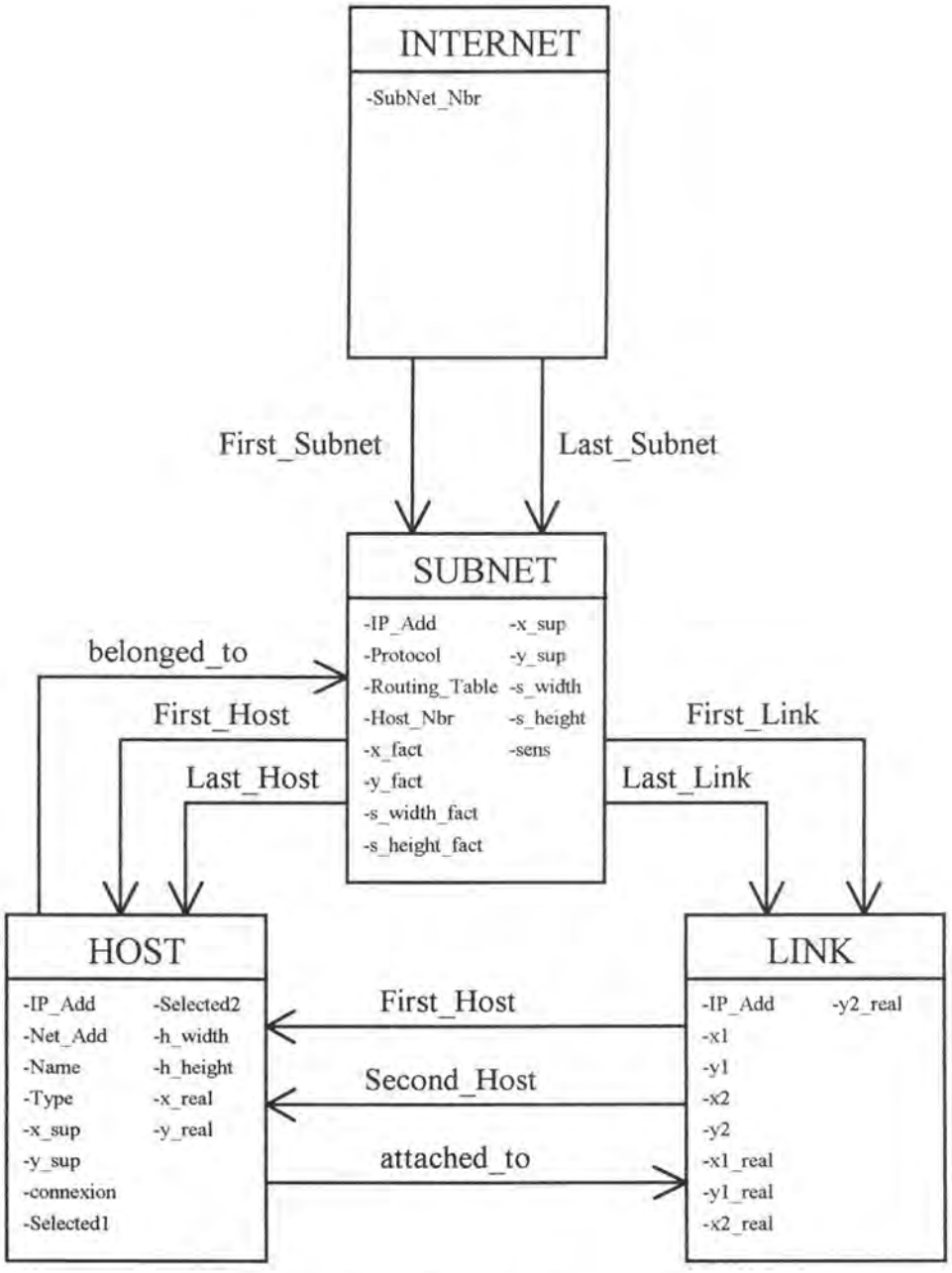


Figure 5.14. : Les relations entre les objets.

Cette figure reprend les différents objets créés avec leurs paramètres; les paramètres classiques sont placés dans les cadres représentant les objets tandis que les paramètres marquant une relation avec d'autres objets sont représentés par une flèche partant de l'objet auquel appartient le paramètre et pointant vers l'objet avec lequel une relation est établie.

On peut donc voir qu'une instance de l'objet « internet » représentant l'interconnexion est composée d'instances de l'objet « subnet ». Ensuite ces instances de l'objet « subnet » sont composées à leur tour d'instances des objets « host » et « link ». Les instances de l'objet

« host » appartiennent au moins à une instance de l'objet « subnet » et sont reliées à au moins une instance de l'objet « link ». Enfin, une instance de l'objet « link » est reliée à une ou deux instances de l'objet « host ».

Toutes ces considérations indiquent que la création de ces objets et de leurs relations implémente donc bien la notion d'interconnexion; ce qui était le but poursuivi.

4.2. L'implémentation de l'interface

Le didacticiel ayant été développé dans un environnement Xwindow, l'implémentation de l'interface a dû être réalisée en respect des règles instaurées dans ce cadre particulier. Le but de cette partie est de décrire la philosophie de programmation sous Xwindow et les techniques utilisées dans le cadre du développement du didacticiel plutôt que de décrire l'implémentation proprement dite de l'interface du didacticiel, ce qui se serait avéré long et sans intérêt. Nous nous baserons pour cette description sur le document « Interface Homme-Machine. Réalisation d'application avec Xt et Motif » [MULLER93] de Daniel Muller, Professeur agrégé à l'Ecole Centrale de Lyon.

4.2.1. La programmation sous Xwindow

La programmation sous Xwindow se base sur des bibliothèques écrites en C et appelées Xlib. Un programme basé sur Xlib est piloté par des événements c'est-à-dire que son fonctionnement n'est pas séquentiel mais que ce sont les événements de l'environnement qui déterminent quelles sont les fonctions à exécuter.

La structure standard d'une application basée sur Xlib est composée d'une boucle de gestion des événements qui permet d'exécuter les actions appropriées en fonction des événements qui se produisent. Si l'application utilise le multi-fenêtrage, le programme doit également préciser la fenêtre dans laquelle l'événement est survenu. Mais si le nombre de fenêtres augmente, la boucle de gestion peut vite devenir énorme et difficile à gérer.

La bibliothèque Xt encore appelée Intrinsics, rend ce problème transparent au concepteur d'application en fournissant une philosophie de type « action-réaction ». Le programmeur peut dès lors écrire de petites fonctions pour réagir à tel ou tel événement spécifique tel que l'activation d'un bouton par exemple.

4.2.2. Les Widgets

Le programmeur associe à chaque programme une interface utilisateur à l'aide d'un jeu de Widgets[6]. Un Widget est un composant de l'interface comme un bouton, un ascenseur ou encore un élément d'un menu. Les Intrinsics ne fournissent que quelques Widgets de base ne suffisant pas en tant que tels pour construire une interface évoluée, mais fournissent également les mécanismes qui permettent de construire d'autres Widgets à partir de Widgets existants. C'est ainsi qu'existent des bibliothèques de Widgets plus évolués permettant la création d'interfaces sophistiquées. Les deux bibliothèques de Widgets les plus répandues sont Xaw (Athena Widgets) développée par le MIT qui est du domaine publique et Xm (Motif Widgets)

[6] Widget signifie Window Object.

développée par OSF et fournissant des Widgets qui ont un aspect tridimensionnel. Notre didacticiel utilise la bibliothèque OSF - Motif pour son interface.

Les Widgets composant l'interface d'une application sont organisés en une hiérarchie arborescente correspondant à la hiérarchie des fenêtres X qui leur sont associées. Une particularité est que le Widget constituant la racine de l'arborescence est toujours du type « Shell ». Ce sont ces Widgets de type « Shell » qui gèrent entre autres les interactions avec le Window manager. Chaque Widget est géré par son parent notamment en ce qui concerne sa position et sa taille, le fait qu'il soit vu ou caché et le contrôle des événements qui peuvent lui être envoyés ou non.

4.2.3. Association des événements aux Widgets

Xt offre également deux mécanismes pour associer les événements aux Widgets : les Callbacks et les Event Handlers qui sont en fait des fonctions qui sont déclenchées suite aux actions de l'utilisateur ou aux événements correspondant à chaque Widget. L'utilisateur peut donc associer une fonction à tous les types d'événements qu'il désire gérer.

Par exemple, lorsque l'utilisateur agrandit la fenêtre principale de notre didacticiel, il provoque un événement (l'agrandissement de la fenêtre) auquel est associée la Callback « TcpIpRecomputeCB ». Cette fonction est alors exécutée et appelle la méthode « recompute » de l'instance de l'objet « Internet » qui est représentée dans la fenêtre agrandie.

4.2.4. La structure standard d'une application Xwindow

La structure standard d'une application X est constituée des 5 étapes suivantes :

- La première étape consiste à initialiser la couche des Intrinsics, ce qui établit la connexion avec le serveur X et alloue les différentes ressources (dont nous parlerons plus loin).
- La deuxième étape est la création de l'interface à l'aide d'une bibliothèque ou l'autre de Widgets.
- La troisième étape consiste à établir les associations « action - réaction ». C'est à ce niveau que le concepteur introduit les Callbacks et les Event Handlers.
- La quatrième étape est la réalisation de tous les Widgets, c'est-à-dire la création des fenêtres X qui leur sont associées. La réalisation d'un Widget entraîne automatiquement la réalisation de tous ses enfants.
- La dernière étape, enfin, est l'entrée dans la boucle des événements qui est entièrement gérée par Xt.

4.2.5. Les Ressources

La ressource est à un Widget ce que la variable d'instance est à un objet. Le comportement d'un Widget est déterminé à un instant donné par la valeur de ses ressources. Les ressources sont, par exemple, la couleur, la taille ou la position d'un Widget.

Il existe différentes méthodes pour affecter ou modifier la valeur des différentes ressources d'un Widget. Le programmeur peut affecter une valeur à ces différentes ressources au moment de la création du Widget; il doit pour cela initialiser un tableau contenant le nom

des ressources et la valeur qu'il désire affecter à ces ressources et passer ensuite ce tableau comme un argument de la fonction de création du Widget. Par exemple, si le programmeur désire créer un Widget de type « label » ayant une taille de 200 par 40, il peut écrire le code suivant :

```
#include <X11/StringDefs.h>

Arg       args[2];
int       n;
Widget          label;

n = 0;
XtSetArg (args[n], XtNwidth, 200); n++;
XtSetArg (args[n], XtNheight, 40); n++;

label = XtCreateManageWidget("label", xmLabelWidgetClass,
                        toplevel, args, XtNumber(args));
```

où XtNwidth et XtNheight sont des constantes définies dans le fichier « X11/StringDefs.h » comme étant respectivement égales aux chaînes de caractères « width » et « height »; ces deux valeurs correspondent aux noms des ressources dont on fixe la valeur. Et où XtSetArg ( ) est une macro permettant d'initialiser le tableau des ressources « args ».

Le programmeur peut également retrouver et modifier la valeur des ressources d'un Widget existant déjà. Ce type de manipulation peut être utile par exemple si le programmeur veut changer le libellé d'un bouton lorsqu'on appuie sur celui-ci.

Les méthodes d'affectation et de modification des valeurs de ressources présentées jusqu'ici permettent au concepteur de déterminer lui-même les valeurs de ces ressources; il est pourtant souvent souhaitable que ces ressources ne soient pas imposées par le concepteur mais par l'utilisateur final. Les couleurs, en particulier, sont un exemple de ressource que l'utilisateur final devra pouvoir personnaliser à sa guise afin d'intégrer harmonieusement l'application dans son environnement de travail habituel.

Pour permettre cela, Xwindow maintient une base de données contenant les préférences des utilisateurs ainsi que les valeurs par défaut conseillées pour chaque application. Cette base de données est construite à chaque exécution de l'application lors de la phase d'initialisation de la couche Xt et se base pour cela sur un fichier de ressources que l'utilisateur peut personnaliser à sa guise.

Afin d'accéder à une ressource particulière d'un Widget donné à partir de ce fichier, il faut spécifier l'entièreté du chemin à suivre dans l'arborescence des Widgets depuis le nom de l'application jusqu'au Widget considéré. Par exemple, si l'utilisateur veut que le fond du bouton « quitter » dont le parent est le Widget « panel » situé dans la fenêtre principale « main_window » de l'application « TcpIp » soit jaune, il devra spécifier la ligne suivante dans le fichier des ressources :

```
TcpIp.main_window.panel.quitter.background : yellow
```

Cette ligne spécifie le nom de la ressource, des Widgets et de l'application qui sont touchés. Mais une ressource, un Widget ou une application appartiennent également à une classe; et ces classes peuvent être aussi spécifiées dans le fichier des ressources. Par exemple, la ligne

```
TcpIp.main_window.XmForm.XmPushButton.Background : yellow
```

indique que toutes les ressources de type « Background » de tous les Widgets de type « XmPushButton » situé dans un Widget de type « XmForm » placé dans la fenêtre principale « main_window » de l'application « TcpIp » doivent avoir la couleur jaune.

Enfin, il également possible de ne pas spécifier complètement le chemin aboutissant à un Widget particulier en utilisant l'astérisque (*). Par exemple, la ligne :

```
TcpIp*XmPushButton.background : gray
```

indique que le fond de tous les Widget de type « XmPushButton » (c'est-à-dire de tous les boutons) de l'application « TcpIp » doit être gris. Si plusieurs spécifications conduisent à des résultats différents, c'est la spécification définissant le Widget de la manière la plus précise qui est prise en compte.

Toutes les ressources spécifiées ici sont des ressources prédéfinies dans les librairies de Xwindow. Xt fournit cependant des mécanismes permettant au concepteur de créer de nouvelles ressources propres à l'application développée. Le didacticiel développé dans le cadre de ce travail est pourvu de telles ressources. Par exemple, les ressources suivantes ont été créées :

- *host1SelectionColor* : ressource permettant de spécifier la couleur de mise en évidence de la première machine sélectionnée par l'utilisateur.
- *DatagramColor* : ressource précisant la couleur du point représentant le datagramme lors de l'animation.
- ...

Ces ressources sont également définissables à partir du fichier de ressources.

Si une ressource est définie explicitement par le programmeur au moment de la création du Widget ou de l'application, l'utilisateur ne pourra pas modifier ces valeurs à l'aide du fichier de ressource. Le développement d'une application utilisant cette notion de ressources demande donc d'établir un équilibre entre les ressources imposées par le programmeur afin de fournir un certain style à son application et les ressources pouvant être personnalisées par l'utilisateur. C'est dans le but d'arriver à un tel équilibre que notre didacticiel peut être personnalisé du point de vue des couleurs, des types des caractères et de la langue utilisée alors que d'autres ressources telles que la taille minimum des fenêtres ou les libellés des menus ont été imposées lors du développement.

4.2.6. Les outils de génération automatique de code

Une fois que toutes ces notions sont maîtrisées, l'implémentation d'une interface sous Xwindow devient une tâche purement mécanique et fastidieuse. C'est pourquoi les membres de COLOS qui travaillent principalement dans cet environnement ont créé des outils générant automatiquement du code à partir de fichier descriptif à syntaxe simple.

Quelques uns de ces outils sont :

- *MakePixAppl* : outil générant le code nécessaire à la création d'une fenêtre à l'écran. Cette fenêtre est pourvue d'un menu et d'une surface graphique à affichage rapide.

- *Xweb* : logiciel permettant de générer du code C à partir d'une création interactive de l'interface ou à partir d'un texte de syntaxe simple décrivant l'arborescence des Widgets.

- *xmMenuGen* : outil générant soit un texte C, soit un texte Xweb à partir d'une description d'un ou plusieurs menus propres à l'application développée.

- *ui2c* : outil écrit sous forme de « script » dans le langage « PERL » permettant de générer du code C à partir d'un texte de syntaxe simple décrivant l'arborescence des Widgets (qu'il s'agisse de menus ou de fenêtres classiques).

La description détaillée de ces outils que nous avons utilisés faisant l'objet du chapitre suivant, nous ne nous étendrons pas plus sur cette description ici.

4.3. La découpe en modules

La découpe en modules représente le coeur même de l'architecture logicielle. Comme il est précisé dans le cours de MDL [DUBOIS93] du Professeur E. Dubois et rappelé dans le chapitre 1 de ce mémoire, deux approches sont possibles à ce niveau. Le concepteur peut procéder à une découpe en modules pour arriver soit à une architecture physique dans laquelle le module constitue « *une unité de travail pour la machine* » [DUBOIS93], soit à une architecture logique dans laquelle le module constitue « *une unité de travail pour le concepteur* » [DUBOIS93].

Lors du développement de notre didacticiel, nous avons adopté la deuxième approche qui facilite par définition la division du travail, la maintenance et la documentation.

4.3.1. La découpe adoptée

La figure 5.15. montre la découpe en modules que nous avons adoptée.

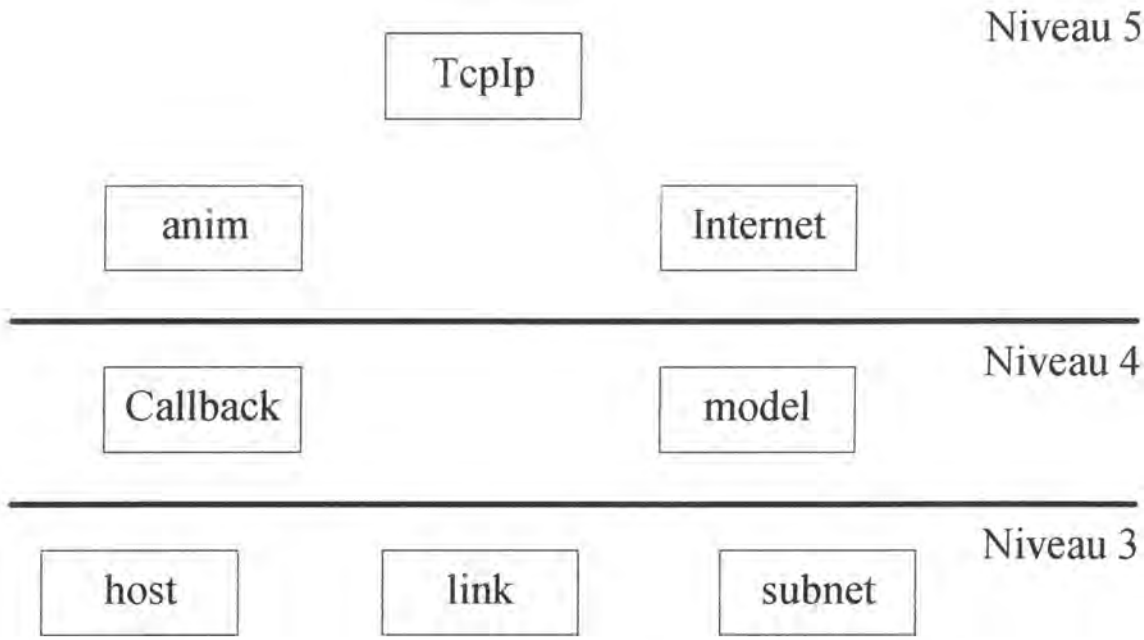


Figure 5.15. : La découpe en modules logiques du didacticiel.

Les modules sont représentés par des cadres contenant leur nom. On remarque que tous les modules ne sont pas situés au même niveau, ces niveaux représentent les caractéristiques des fonctions se trouvant dans les différents modules. Le niveau 5 correspond aux modules de coordination et de traitement. Le niveau 4 reprend les modules dont les services sont des fonctions d'entrées-sorties (que ce soit du point de vue écran, clavier, souris ou fichier). Le niveau 3 est constitué des modules contenant les fonctions travaillant sur les objets persistants en mémoire (centrale ou secondaire). Les niveau 1 et 2, non représentés, correspondent aux services et primitives du système d'exploitation.

Le module « TcpIp » constitue le module coordinateur de l'application. Le module « anim » reprend toutes les fonctions de traitement relatives à la gestion de l'animation. Le module « Internet » qui correspond à l'objet « internet » est placé également au niveau 5 puisqu'il assure la coordination des autres objets créés.

Le module « Callback » reprend la plupart des Callbacks définies dans l'application; les Callbacks étant les fonctions réagissant aux événements de l'environnement, nous avons décidé de placer ce module au niveau 4. Le module « model » reprend les fonctions gérant la représentation à l'écran des différentes fenêtres. On retrouvera notamment dans ce module les Callbacks appelées lors de la manipulation des fenêtres par l'utilisateur.

Enfin, les modules « host », « link » et « subnet » sont les modules correspondant aux objets « host », « link » et « subnet ». Ces objets gérant leur tracé à l'écran, nous aurions pu les placer également au niveau 4; mais la plupart des méthodes travaillant sur les paramètres des instances de ces objets en mémoire centrale, nous avons décidé de les placer au niveau 3.

L'application étant pilotée par les événements, certaines fonctions telles que les Callbacks ou les Event Handlers ne sont pas appelées explicitement à partir d'autres modules; c'est la raison pour laquelle les relations « utilise » ne sont pas représentées à la figure 5.15.

Maintenant que la découpe en modules et la répartition en niveaux ont été précisées, il reste à décrire les différents modules avec leurs services. La description des services se fera de façon littérale afin de faciliter la compréhension du lecteur.

4.3.2. Le module « TcpIp »

Le module « TcpIp » est le module de base de l'application; il contient toutes les fonctions d'initialisation de l'application ainsi que la fonction de gestion de la boîte de dialogue contrôlant l'animation. Ce module reprend les services suivants :

- *customize* : fonction initialisant les écrans graphiques et les ressources propres à l'application.
- *initInterface* : fonction créant les bitmaps, pixmaps et curseurs spéciaux nécessaires à l'exécution de l'application.
- *initDrawArea, initSubnetDrawArea, initDatagramDrawArea* : fonctions initialisant les caractéristiques graphiques des trois fenêtres graphiques.
- *addBehavior* : fonction initialisant le comportement de l'application en associant des Callbacks à différents Widgets et en appelant la fonction de création de l'interconnexion présentée à l'étudiant.
- *FillHostSelBox* : fonction qui gère la boîte de dialogue contrôlant l'animation et reprenant les caractéristiques des machines sélectionnées. Cette fonction est appelée par la Callback déclenchée par l'événement « sélection d'une machine ».
- *WhichParam* : fonction qui gère la boîte de dialogue qui demande à l'utilisateur de choisir une connexion lorsqu'il a sélectionné une passerelle.
- *create_default_internet* : fonction créant l'interconnexion par défaut présentée à l'écran en appelant les constructeurs des objets concernés.
- *round* : fonction retournant l'arrondi d'un réel passé en argument.
- *delay* : fonction provoquant une pose d'une durée égale à un nombre de secondes passé en argument.
- *CleanAll* : fonction « nettoyant » toutes les variables et tous les pointeurs utilisés lors de l'animation du transit d'un datagramme.

4.3.3. Le module « anim »

Le module « anim » est le module chargé de la gestion de l'animation du transit des datagrammes dans l'interconnexion représentée à l'écran. Pour ce faire, ce module offre les services suivants :

- *animate* : fonction initialisant toutes les variables nécessaires au déroulement de l'animation et appelant la fonction chargée d'exécuter l'algorithme de routage.

- *RoutingTimerProc* : fonction assurant les traitements relatifs à l'exécution de l'algorithme de routage. Si l'algorithme de routage aboutit sans erreur, elle appelle la fonction « move_datagram » chargée de l'animation. Dans le cas contraire, elle rend la main à l'utilisateur.

- *Route_datagram* : fonction chargée de la gestion de l'exécution de l'algorithme de routage aussi bien de manière interne que de manière graphique dans la fenêtre reprenant le moniteur pour l'algorithme de routage.

- *compute_graphic_way, compute_subnet_graphic_way* : fonctions calculant le chemin graphique que devra suivre le point de couleur représentant le datagramme lors de l'animation. Ces fonctionnalités calculent respectivement le chemin graphique dans la fenêtre principale et dans la fenêtre donnant une vue détaillée du sous-réseau courant.

- *move_datagram, MoveDatagramTimer* : fonctions assurant, en collaboration, la représentation graphique de l'animation, ce qui consiste à faire déplacer le point de couleur représentant le datagramme le long du chemin graphique calculé par les deux fonctions précédentes.

4.3.4. Le module « Internet »

Le module « Internet » est le module correspondant à l'objet « internet » qui est l'objet qui coordonne les autres objets créés. Le lecteur trouvera la description des services de ce module dans la description des méthodes de l'objet « internet » située au point 4.1.4. de ce chapitre.

4.3.5. Le module « Callback »

Le module « Callback » reprend la plupart des Callbacks créées dans l'application. Les services de ce module sont les suivants :

- *quitApp* : fonction appelée par l'événement produit lors de l'activation de l'item de menu « exit » du menu « File »; son effet est la terminaison de l'exécution du didacticiel.

- *destroyCB* : Callback associée à tous les Widgets représentant une fenêtre que l'utilisateur peut fermer. Lorsque l'événement produit par la fermeture d'une de ces fenêtres survient, cette fonction détruit le Widget représentant la fenêtre fermée.

- *viewCB* : Callback associée aux items de menu du menu « View ». Lorsque l'utilisateur active un des ces items de menu, cette fonction est appelée et bascule la représentation de l'interconnexion vers la vue souhaitée.

- *HostSelectionBoxCB* : Callback associée aux boutons situés dans la boîte de dialogue contrôlant l'animation. Lorsqu'un de ces boutons est activé par l'utilisateur, cette fonction offre la réaction relative à la fonctionnalité représentée par le bouton activé. Ces fonctionnalités sont la « désélection » d'une machine, l'annulation de toutes les sélections, le déclenchement de l'animation ou la sélection d'une connexion lorsqu'une passerelle a été sélectionnée.

- *monitorCB* : Callback associée aux items du menu « Monitor ». Lorsque l'utilisateur active un de ces items de menu, la fonction est appelée et fournit le moniteur demandé c'est-à-dire qu'elle crée la fenêtre correspondante.

- *configCB* : Callback associée aux items du menu « Config ». Lorsqu'un de ces items est activé par l'utilisateur, la fonction est appelée et fournit la configuration demandée. Les deux fonctionnalités offertes par ces items de menu sont « la demande d'un fond coloré pour la représentation d'un réseau » et « la demande de *désélection* automatique en fin d'animation ».

- *helpCB* : Callback associée à l'item du menu « Help ». Lorsque cet item est activé, la fonction réagit en ouvrant la fenêtre de logo à l'écran.

- *scaleCB* : Callback associée à l'ascenseur permettant de gérer la vitesse de l'animation. Lorsque l'utilisateur manipule cet ascenseur, cette fonction est appelée et enregistre la nouvelle vitesse désirée.

4.3.6. Le module « model »

Le module « model » est chargé de gérer la représentation des fenêtres à l'écran. On retrouve notamment dans ce module les Callbacks appelées lorsqu'un événement de type « manipulation d'une fenêtre » survient. Ce module reprend les services suivants :

- *initState* : fonction initialisant toutes les variables représentant des paramètres de représentation de divers Widgets. Cette fonction est appelée une seule fois au début de l'exécution du didacticiel.

- *TcpIpRecomputeCB* : Callback appelée lorsque l'utilisateur modifie la taille de la fenêtre principale du didacticiel. Cette fonction appelle la méthode « recompute » de l'objet « internet » représenté dans la fenêtre principale et, si une animation est en cours, elle recalcule toutes les coordonnées du chemin graphique que parcourt la représentation du datagramme.

- *TcpIpRedrawCB* : Callback appelée lorsqu'un événement nécessite de redessiner ce qui est représenté dans la fenêtre principale. Les événements pouvant provoquer l'appel de cette fonction sont la modification de la taille de la fenêtre et le fait de masquer ou démasquer une partie de cette fenêtre avec d'autres fenêtres présentes à l'écran. Cette fonction appelle la méthode « trace » de l'objet « internet » représenté dans la fenêtre principale.

- *TcpIpMultiClickEH* : Event Handler réagissant à un « double-click » de l'utilisateur dans la fenêtre graphique principale. Cette fonction sélectionne alors la machine dont la surface de représentation reprend les coordonnées du point de « double-click ». Si aucune machine n'est représentée à cet endroit, la fonction ne fait rien.
  Notons qu'il n'existe pas d'événement « double-click » pour une fenêtre graphique. Cet inconvénient a toutefois été contourné par une astuce que nous décrivons en détail au chapitre 7 (point 3.1.1.).

- *SingleClickHandler* : fonction représentant un Timer[7] utilisé par la fonction « TcpIpMultiClickEH ».

- *SubnetRecomputeCB* : Callback similaire à la Callback « TcpIpRecomputeCB » sauf que la fenêtre impliquée ici est la fenêtre dans laquelle est représentée la vue détaillée du sous-réseau courant.

- *SubnetRedrawCB* : Callback similaire à la Callback « TcpIpRedrawCB » sauf que la fenêtre impliquée ici est la fenêtre dans laquelle est représentée la vue détaillée du sous-réseau courant.

- *GiveRightSize* : fonction dont le but est de donner la taille par défaut correcte à la fenêtre contenant la représentation détaillée du sous-réseau courant. Cette fonction est appelée chaque fois que le sous-réseau courant change.

- *DatagramRedrawCB* : Callback appelée lorsqu'un événement nécessite de redessiner ce qui est représenté dans la fenêtre graphique du moniteur pour le datagramme. A part lorsque ces événements surviennent, cette fonction est appelée chaque fois que le format du datagramme change et que la fenêtre est présente à l'écran.

- *Draw3DRectangle* : fonction dessinant dans la fenêtre graphique dont les caractéristiques sont passées en arguments la représentation « en relief » d'un rectangle dont la position, les dimensions et la couleur sont également passées en arguments. Cette fonction sert principalement à la fonction « DatagramRedrawCB » afin de tracer la représentation du datagramme dans la fenêtre du datagramme.

4.3.7. Les modules « host », « link » et « subnet »

Les modules « host », « link » et « subnet » sont les modules correspondant aux objets de même nom. Le lecteur trouvera la description des services de ce module dans la description des méthodes de ces objets aux points 4.1.1., 4.1.2. et 4.1.3.

[7] Un Timer est une fonction qui est appelée après un certain temps.

Chapitre 6 :
Les outils utilisés pour le développement du didacticiel

Comme nous l'avons déjà précisé dans le chapitre précédent, l'implémentation d'une interface dans un environnement Xwindow, bien que mécanique, est une tâche longue et fastidieuse pour le concepteur. C'est cette caractéristique d'être mécanique qui a donné l'idée à différents membres du consortium COLOS de développer plusieurs outils destinés à faciliter la tâche du concepteur. Ces outils vont de « shell script » permettant de créer de nouvelles applications Xwindow à des programmes sophistiqués permettant de créer une interface homme-machine de manière interactive.

L'équipe de COLOS a également créé toute une série d'outils à caractère plus logistique permettant, par exemple, de fusionner deux fichiers sources en un seul ou encore permettant d'installer correctement la plate-forme standard de COLOS qui reprend tous les programmes et librairies utiles lors de la création d'une nouvelle application, etc.

Nous nous limiterons dans ce chapitre à décrire les outils d'aide à l'implémentation d'une interface que nous avons utilisés de façon soutenue tout au long du développement de notre didacticiel; il s'agit des outils « makePixAppl », « xweb », « xmMenuGen » et « ui2c ». La description de ces outils expliquera la façon de les utiliser, leurs principales caractéristiques et, bien entendu, les résultats liés à leur utilisation. Notons que tous les outils décrits dans ce chapitre nécessitent l'installation préalable de la plate-forme standard de COLOS dont nous avons parlé plus haut.

1. L'outil « makePixAppl »

L'outil « makePixAppl » est un « script » permettant d'obtenir une nouvelle application. Nous nous baserons pour la description de cet outil sur les documents référencés [MULLER92] et [MULLER94] dans la bibliographie.

La première chose à faire pour construire une nouvelle application à partir de cet outil est de se placer dans le répertoire où l'on désire que l'arborescence de sous-répertoires de la nouvelle application soit créée. La nouvelle application peut alors être créée en tapant simplement la ligne de commande :

```
makePixAppl [-ui2c] TcpIp
```

si l'on désire que la nouvelle application s'appelle « TcpIp ». L'option « -ui2c » est à spécifier si on désire que la description de l'interface utilisateur de l'application réponde à la syntaxe de l'outil « ui2c » que l'on décrira plus loin. Si cette option n'est pas spécifiée, la description de l'interface utilisateur respectera la syntaxe imposée par l'outil « xweb ». La nouvelle application créée est similaire à une application « patron » se trouvant dans le répertoire « PIXAPPL » de la plate-forme standard de COLOS.

Les effets pratiques de l'exécution de la ligne de commande présentée plus haut sont :

- la création d'un répertoire dont le nom est le nom de l'application créée en majuscules. Tous les fichiers nécessaires au bon fonctionnement de l'application sont placés dans ce répertoire.

- la création de tous les fichiers nécessaires au bon fonctionnement de l'application. Ces fichiers sont les fichiers de description de l'interface utilisateur, les fichiers contenant le code C de l'application (ce sont les fichiers source « .c » et les fichiers à inclure « .h ») et un fichier ressources (TcpIp_def dans le cas de l'application TcpIp) qui est placé dans un sous-répertoire « RESOURCES » du répertoire principal de l'application.

- la création d'un « MakeFile » qui est un « script » reprenant toutes les commandes nécessaires à la compilation complète et correcte de l'application.

L'interface utilisateur de l'application créée est composée d'une fenêtre Xwindow contenant les éléments suivants : un Widget XcLogoForm et un Widget XcDBDrawArea.

Le Widget XcLogoForm est un Widget de type XmForm qui est capable de représenter à l'écran les logos de COLOS. Ce Widget est en fait un Widget dans lequel sont représentés le nom de l'application, le logo COLOS et cinq lignes horizontales que l'on retrouve dans le haut de la fenêtre Xwindow. Ces différents éléments sont tous des Widgets (respectivement deux XmLabel et cinq XmSeparatorGadget) dont le parent est le Widget XcLogoForm.

Le Widget XcDBDrawArea représente une fenêtre graphique à affichage rapide; il utilise pour cela la méthode du « double buffer » : c'est-à-dire que l'image représentée à l'écran est d'abord composée dans un buffer situé en mémoire centrale avant que celui-ci ne soit envoyé en une seule fois vers le buffer de l'affichage. Cette technique permet d'avoir un affichage très rapide puisque le temps éventuellement perdu lors du calcul des coordonnées des points à afficher est en fait perdu lors de la composition de l'image dans le premier buffer et non pas lors de l'affichage proprement dit. Deux Callbacks sont associées à ce Widget afin de

réagir aux événements survenant lorsque l'on modifie la taille de la fenêtre ou lorsque l'on modifie son exposition à l'écran.

En supposant que c'est l'application « TcpIp » qui vient d'être créée, le concepteur peut définir sa propre interface utilisateur; il doit pour cela soit éditer manuellement les fichiers « TcpIp.ui » ou « TcpIp.wtr » (selon qu'il ait passé l'option « -ui2c » ou non lors de l'exécution de makePixAppl) afin de décrire l'arborescence des Widgets qu'il désire créer et le fichier « RESOURCES/TcpIp_def » pour ajouter de nouvelles ressources, soit utiliser interactivement l'outil « xweb » pour modifier l'interface utilisateur produite par défaut. Le concepteur peut également spécifier le comportement de l'application en éditant les fichiers « TcpIp.c » et « TcpIp.h » afin de créer toutes les fonctions C qu'il désire. Si le concepteur manipule des Widgets dans ses fonctions, il peut utiliser les noms définis pour ces Widgets dans le fichier « TcpIp.ui » ou « TcpIp.wtr » car un fichier « widgets.h » contenant la déclaration de ces Widgets comme variables globales est généré automatiquement par « ui2c » ou « makewidgeth » selon que l'utilisateur utilise l'outil « ui2c » ou « xweb ».

Quand le concepteur a défini l'interface utilisateur et le comportement de son application, il ne lui reste qu'à la compiler. Pour ce faire, il utilise le « MakeFile » créé lors de l'exécution de « makePixAppl »; sans oublier toutefois de personnaliser le « MakeFile » si il désire utiliser d'autres bibliothèques C que les bibliothèques « X11 » et « Motif » relatives à l'interface utilisateur. Quand le « MakeFile » est définitivement spécifié, le concepteur peut alors compiler son application en tapant la commande « make ». Tous les fichiers sources, générés automatiquement ou non, sont alors compilés et « linkés » avec les bibliothèques adéquates pour produire un fichier exécutable appelé « TcpIp » correspondant à l'application développée par le concepteur.

2. L'outil « xweb »

L'outil « xweb » a été le premier outil d'aide à la création d'une interface utilisateur créé et utilisé par les membres de COLOS. Ceux-ci se sont en effet très vite rendus compte que de développer manuellement ces interfaces était une tâche longue et fastidieuse qui n'enrichissait en rien les connaissances acquises jusque là.

« xweb » permet de créer ou de modifier un interface utilisateur de deux manières différentes : soit de manière complètement interactive en manipulant directement à l'aide de la souris la représentation de l'interface en cours de création ou de modification; soit de façon indirecte en éditant un fichier reprenant la description textuelle de l'arborescence des Widgets composant l'interface.

2.1. L'utilisation interactive

L'utilisation interactive de « xweb » pour créer ou modifier l'interface utilisateur nécessite « d'exécuter xweb »; cela se fait en tapant la ligne de commande :

```
xweb [-f filename.wtr]
```

l'option « -f filename.wtr » signifie que l'on désire modifier interactivement l'interface utilisateur dont l'arborescence des Widgets est décrite dans le fichier « filename.wtr ».

L'exécution de cette ligne de commande provoque l'apparition à l'écran d'une fenêtre contenant le tableau de bord de « xweb ».

Les fonctions présentées dans ce tableau de bord permettent de créer ou de modifier une interface utilisateur. Passons en revue les principales fonctionnalités offertes :

- **Créer un nouveau Widget**

  Pour créer un nouveau Widget, il suffit de « cliquer » sur le bouton « Show Make Panel » du tableau de bord; une boîte de dialogue apparaît alors à l'écran et demande les caractéristiques (la classe, le nom et le parent) du Widget à créer. Une fois ces caractéristiques fournies, il suffit de cliquer sur le bouton « makeIt » pour créer effectivement le Widget. Si on clique sur le bouton « make+Place », le concepteur peut alors placer interactivement le Widget à l'endroit où il désire dans son parent.

- **Spécifier les ressources d'un Widget**

  Le concepteur peut également spécifier les ressources d'un Widget créé dans la boîte de dialogue ouverte lors de l'activation du bouton « Show Make Panel » du tableau de bord.

- **Voir les ressources d'un Widget**

  Le concepteur peut visualiser les différentes ressources d'un Widget à l'aide de la fonction « Edit Widget » du tableau de bord.

- **Modifier les ressources d'un Widget**

  Il est également possible de modifier les ressources d'un Widget existant à l'aide de la fonction « Edit Widget » du tableau de bord.

Les autres fonctionnalités interactives qu'offre « xweb » sont la destruction d'un Widget, l'affichage de l'arborescence des Widgets, l'affichage d'un Widget particulier, l'association de Callbacks et d'Event Handlers à un Widget, etc. Le lecteur trouvera une description détaillée de toutes ces fonctionnalités dans le document « Quick Reference to Common Xweb Operations » de C.Y. Young [YOUNG95].

Les dernières étapes de la création interactive d'interface sont la création d'un fichier décrivant l'arborescence des Widgets de l'interface, fichier qui pourra être utilisé lors de l'utilisation indirecte de « xweb » et la création du fichier source C correspondant à l'implémentation de l'interface. Ces deux fonctionnalités sont respectivement déclenchées par l'activation des boutons « writeTree » et « writeSource » du tableau de bord de « xweb ».

2.2. L'utilisation indirecte

L'expérience de l'utilisation de « xweb » par les membres de COLOS a bientôt montré que les programmeurs experts préfèrent ne pas utiliser les possibilités interactives de « xweb » mais bien d'écrire directement des fichiers décrivant l'arborescence des Widgets composant

l'interface utilisateur et d'ensuite les faire traduire en C par « xweb » qui est alors utilisé comme une sorte de compilateur; c'est ce que nous appelons l'utilisation indirecte de « xweb ».

Les fichiers descriptifs de l'arborescence des Widgets sont les fichiers présentant l'extension « .wtr »[1]et respectant un certain nombre de règles syntaxiques. Il n'en reste cependant pas moins que la syntaxe imposée par « xweb » est beaucoup plus simple que la syntaxe imposée par le C.

Toutes les fonctionnalités présentées dans le cas de l'utilisation interactive de « xweb » sont également disponibles lors de son utilisation indirecte, chacune de ces fonctionnalités devant simplement respecter une syntaxe propre. Passons en revue les principales instructions utilisées pour la description d'une interface utilisateur :

- **La création d'un Widget**

  L'instruction de création d'un Widget est composée de deux champs : le premier contient le nom de la classe du Widget à créer et le deuxième contient une construction de la forme « <parent_name>.<widget_name> ». Si le nom du Widget parent n'est pas spécifié, « xweb » assume que le parent est le « shell » de base de l'application.

  Par exemple, si le concepteur veut créer un bouton « CANCEL » situé dans le Widget « dialog_box », il doit spécifier dans le fichier descriptif la ligne suivante :

  ```
  XmPushButton    dialog_box.CANCEL
  ```

- **La spécification et la manipulation des ressources**

  L'instruction de spécification d'une ressource d'un Widget donné doit se trouver avant l'instruction de création du Widget et est structurée en trois champs : le premier contient invariablement le littéral « Arg » qui a pour but de préciser que c'est une instruction de spécification de ressource qui suit; le deuxième contient le nom de la ressource spécifiée et le troisième champ contient la valeur de cette ressource.

  Par exemple, si le concepteur désire que le bouton « CANCEL » créé plus haut ait un fond rouge, il doit écrire l'instruction suivante avant l'instruction de création du Widget :

  ```
  Arg background  red
  ```

  Le concepteur peut bien sûr manipuler ces ressources par la suite; il devra simplement « recompiler » le fichier descriptif à l'aide de « xweb » pour rendre effectives ses manipulations.

- **La destruction d'un Widget**

  Pour détruire un Widget le concepteur n'aura qu'à effacer les instructions de spécification des ressources et de création du Widget qu'il désire détruire; il faut

[1] wtr signifie Widgets' tree.

alors recompiler le fichier descriptif à l'aide de « xweb » pour rendre effective cette destruction.

- **L'association d'une Callback à un Widget**

L'instruction permettant d'associer une Callback à un Widget a le format suivant :

```
widgetName callbackName functionName [D fileName] [[iws] funcCallArg]
```

où les trois premiers champs sont obligatoires et les deux derniers sont optionnels. Le premier champ indique le nom du Widget auquel la Callback est associée; le deuxième champ indique le type de la Callback; et le troisième champ contient le nom de la fonction à appeler. Le quatrième champ, s'il est spécifié, indique le nom d'un fichier qui doit être dynamiquement « linké » au code C généré; le cinquième champ, enfin, indique éventuellement l'argument qui doit être passé à la fonction appelée. Cet argument peut être de trois types : de type entier si l'option i est spécifiée, de type Widget si l'option w est spécifiée ou de type « string » si l'option s est spécifiée.

Par exemple, si le concepteur veut associer une Callback « PushButtonCB » appelée lorsque le bouton « CANCEL » est activé et qu'il ne désire « linker » aucun fichier ni passer aucun argument à la Callback, il devra écrire la ligne suivante :

```
CANCEL    activateCallBack  PushButtonCB
```

En plus de ces fonctionnalités, « xweb » propose un mécanisme permettant d'introduire directement dans les fichiers descriptifs de l'arborescence des Widgets des instructions C qui seront placées dans le code C généré. Il suffit pour cela de faire précéder l'instruction C du mot réservé « LITERAL ». Si cette instruction n'est pas située dans une zone de spécifications de ressources, elle sera placée dans le code source généré avant toute création de Widgets. Si, par contre, l'instruction est située dans une zone de spécification de ressources, elle sera placée dans le code source juste avant la spécification de ces ressources.

Le lecteur trouvera une description plus poussée et des exemples de ces fonctionnalités dans les documents référencés [YOUNG95] et [MULLER94] dans la bibliographie.

3. L'outil « xmMenuGen »

L'implémentation ou la description « xweb » d'un menu résultent généralement en un texte peu clair car la création d'un menu unique nécessite la création de deux Widgets (un bouton « cascade » et un menu) connectés ensembles par la ressource « XmNsubMenuId ». C'est pourquoi, poursuivant dans la philosophie de rendre la tâche du concepteur la plus agréable et la plus facile possible, les membres de COLOS ont créé un outil permettant de simplifier la spécification des menus déroulants et des menus « popup »[2]: c'est l'outil « xmMenuGen ». Nous nous baserons sur le document référencé [MENU94] dans la bibliographie pour décrire cet outil.

[2] Un menu « popup » est un menu apparaissant à l 'écran lorsque l'utilisateur clique sur une zone spéciale de l'écran définie par le concepteur.

3.1. Description de « xmMenuGen »

« xmMenuGen » est un programme qui permet de générer soit du code C, soit une description « xweb » de l'arborescence des Widgets à partir d'un fichier contenant une description des menus écrite dans une syntaxe simplifiée. Si, par exemple, le concepteur a écrit la description d'un menu dans le fichier « un.menu », il peut générer le code C implémentant cette description en exécutant la ligne de commande :

```
xmMenuGen un.menu menu.c [menu.h]
```

Le résultat de l'exécution de cette commande est le fichier source C « menu.c ». Si l'option « menu.h » est spécifiée, « xmMenuGen » génère en plus le fichier « include » « menu.h » contenant la déclaration comme variables globales de tous les Widgets composant le menu ainsi que les prototypes des fonctions de création de ce menu.

Le concepteur peut également générer un fichier « xweb » contenant la description de l'arborescence des Widgets composant le menu décrit dans le fichier « un.menu ». Il doit pour cela exécuter la ligne de commande :

```
xmMenuGen un.menu -wtree menu.wtr
```

Le résultat de l'exécution de cette commande est le fichier « menu.wtr » contenant la description de l'arborescence des Widgets composant le menu dans la syntaxe de « xweb ».

3.2. La syntaxe de « xmMenuGen »

La syntaxe de « xmMenuGen » est très simple, elle reprend les règles décrites ci-dessous. Un fichier est utilisé pour spécifier une seule hiérarchie de menu; si le concepteur désire créer plusieurs hiérarchies, il devra créer autant de fichier que de hiérarchies désirées.

Le premier constituant d'un menu est toujours soit une barre de menu, soit un menu « popup » ou encore un sous-menu. Le concepteur peut respectivement spécifier ces trois types de menu par les lignes de code suivante :

```
MenuBar menuSymbolicName [FuncName aFuncName]
PopUpMenu menuSymbolicName [FuncName aFuncName]
Menu menuSymbolicName
```

où MenuBar, PopUpMenu, Menu et FuncName sont des mots réservés de la syntaxe de « xmMenuGen ». « menuSymbolicName » est le nom symbolique du menu spécifié par le concepteur. Si l'option « FuncName » est spécifiée, l'utilisateur doit également fournir « aFuncName » qui correspond au nom de la fonction de création du menu dans le code C généré.

Lorsque la barre de menu ou le menu « popup » ont été précisés, il faut définir les items du menu. La syntaxe de l'instruction de création d'un item de menu est la suivante :

```
MenuItem symName ItemType labelString
```

où « MenuItem » est un mot réservé, « symName » est le nom symbolique donné par le concepteur à l'item, « ItemType » est le type de l'item de menu et « labelString » est le libellé de l'item de menu à l'écran.

Il existe plusieurs types d'items de menu. L'ensemble de ces types d'item est constitué des types suivants :

- *pushButton, buttonGadget* : c'est le plus simple des types d'item de menu, il s'agit d'un bouton classique.

- *toggleButton, toggleButtonGadget* : c'est un type d'item de menu pouvant basculer entre deux états. L'état courant est indiqué par la représentation tridimensionnelle d'un petit bouton en position enfoncée ou non.

- *separator, separatorGadget* : ce type d'item de menu représente simplement un séparateur dans le menu : c'est une simple ligne permettant de séparer deux groupes d'items différents. Dans le cas où le concepteur crée un séparateur, il ne doit pas spécifier le champ « labelString » de l'instruction de création d'un item de menu.

- *label, labelGadget* : ce type d'item de menu représente simplement une zone de texte inaccessible à l'utilisateur. Cela peut servir au concepteur si il veut placer un commentaire dans son menu.

- *subMenu, subMenuGadget, helpSubMenu, helpSubMenuGadget* : ce type d'item de menu représente un bouton qui, lorsqu'il est activé, fait apparaître un sous-menu. L'item de menu créé peut être distingué d'un bouton classique grâce à une petite flèche qui est représentée à la droite du libellé du bouton. L'instruction de création de ce type d'item de menu nécessite un cinquième champ qui reprend le nom symbolique du sous-menu. Ce nom symbolique permettra de faire le lien entre l'item de menu et le sous-menu grâce à la ressource « XmNsubMenuId ».

Enfin, « xmMenuGen » permet également d'associer des Callbacks aux items de menu représentant un bouton. Il suffit pour cela que l'utilisateur définisse cette Callback juste après l'instruction de création de l'item de menu.

Nous terminerons cette description par la présentation d'un exemple de spécification d'un menu simple. Ce menu est composé d'une barre de menu « mBar » contenant un item de menu unique « _File ». Cet item de menu est un sous-menu qui contient également un item unique qui est le bouton « _quit » auquel est associée la Callback « exit ». Le fichier descriptif d'un tel menu serait le suivant :

```
MenuBar mBar
    MenuItem _File subMenuGadget "File" menuFile

Menu menuFile
    MenuItem _quit buttonGadget "Quit"
          activateCallback exit
```

4. L'outil « ui2c »

Sachant que l'outil « xweb » est une application importante comptant quelques 20 000 lignes de code (et est donc particulièrement difficile à maintenir) et que la plus grande partie de cette complexité est due aux fonctions interactives qui ne sont pratiquement pas utilisées par les programmeurs experts, l'équipe de COLOS Lyon a décidé de créer un nouveau traducteur de description d'interface vers le C : traducteur beaucoup plus simple dans sa conception mais

qui accepte le même genre de syntaxe que « xweb » pour la description de l'arborescence des Widgets composant l'interface utilisateur d'une application.

Cet outil, appelé « ui2c »[3], est écrit en « PERL » et contient dans sa version actuelle quelques 1 000 lignes de code (dont le tiers de commentaires). Cette taille réduite a pour conséquence directe de permettre une maintenance plus facile ainsi qu'une installation sur de nouveaux sites plus simple que pour l'outil « xweb ». De plus, les capacités de « ui2c » couvrent également les possibilités de « xmMenuGen ». Le concepteur d'interface ne devra donc plus utiliser qu'un outil unique (« ui2c ») lors de la conception au lieu de devoir jongler avec deux outils (« xweb » et « xmMenuGen »).

Etant donné que « ui2c » a été développé durant la période de développement de notre didacticiel, nous avons pu utiliser les outils « xweb », « xmMenuGen » et « ui2c » successivement. Cette expérience a permis à l'équipe de COLOS Lyon de constater que l'utilisation de « ui2c » s'avère plus simple et donc plus efficace que l'utilisation combinée des outils « xweb » et « xmMenuGen ». Remarquons néanmoins que ces derniers outils sont encore nécessaires pour modifier et recompiler les applications créées avec l'aide de ceux-ci.

4.1. Principes de fonctionnement de « ui2c »

Nous nous baserons pour la description de cet outil sur le document « ui2c : a User Interface to C language translator » [MULLER94a] de Daniel Muller, professeur agrégé à l'Ecole Centrale de Lyon.

L'outil « ui2c » analyse un fichier de description d'interface utilisateur et génère un fichier source en C contenant l'implémentation de l'interface utilisateur décrite et un fichier « include » contenant la déclaration comme variables globales de tous les Widgets utilisés. « ui2c » utilise pour cela deux fichiers d'information : « ui.allWidgets » et « ui.specResources ».

Le fichier « ui.allWidgets » spécifie, pour chaque type de Widgets, le fichier de Xwindow à inclure dans le code généré et la méthode d'instanciation relative aux types de Widgets concernés. Cela permet à « ui2c » de retrouver pour chaque type de Widget la syntaxe C correcte pour implémenter l'interface décrite.

Le fichier « ui.specResources » quant à lui reprend la liste de toutes les ressources nécessitant une attention particulière. C'est le cas par exemple des ressources dont la valeur est une chaîne de caractères. Pour de telles ressources, il est en effet d'abord nécessaire de transformer la chaîne de caractères fournies par le concepteur en une variable de type « XmString » afin de l'affecter à la ressource concernée. Ce fichier permet également à « ui2c » de générer du code C syntaxiquement correct.

Le concepteur devra pour générer du code C à partir d'un fichier descriptif « file.ui » exécuter la ligne de commande suivante :

```
ui2c file.ui [-s generated.c] [-includeFile generated.h]
```

dont l'effet sera la génération du fichier source C « generated.c » et du fichier « include » « generated.h » à partir du fichier descriptif « file.ui ». Si l'option « -s » n'est pas

[3] « ui2c » signifie User Interface TO C.

spécifiée, le fichier « source.c » sera généré automatiquement et si l'option « -includeFile » n'est pas spécifiée, le fichier « Widget.h » sera automatiquement généré. Une série d'autres options sont également disponibles afin de spécifier, par exemple, la version de l'application générée, le fait que l'on veuille que « ui2c » génère automatiquement des commentaires, etc. Nous renvoyons le lecteur au document référencé [MULLER94a] dans la bibliographie pour plus de détails sur ces options.

4.2. La syntaxe de « ui2c »

La syntaxe de « ui2c » étant très proche des syntaxes de « xweb » et « xmMenuGen », nous la décrirons brièvement en insistant toutefois sur les changements vis-à-vis des syntaxes de « xweb » et « xmMenuGen ».

La syntaxe de « ui2c » permet d'obtenir les fonctionnalités suivantes :

- **La création d'un Widget**

  La syntaxe de l'instruction de création d'un Widget avec « ui2c » est identique à la syntaxe de « xweb » : l'instruction est composée de deux champs, le premier contient le nom de la classe du Widget à créer et le second contient une construction de la forme : « <parent_name>.<widget_name> »

- **La spécification et la manipulation des ressources**

  L'instruction « ui2c » correspondant à cette fonctionnalité a une syntaxe identique à la syntaxe de « xweb ».

- **La destruction d'un Widget**

  La méthode de destruction des Widgets avec « ui2c » est identique à celle employée avec « xweb ».

- **L'association d'une Callback à un Widget**

  La syntaxe de « xweb » pour l'ajout des Callbacks est acceptée par « ui2c ». Cependant, une nouvelle syntaxe plus formelle est proposée; dans cette syntaxe, l'instruction d'association d'une Callback à un Widget est constituée des cinq champs suivants :

  ```
  awidget Callbacktype CB function [client_data]
  ```

  Le premier champ correspond au nom du Widget auquel est associée la Callback; le deuxième champ donne le type de la Callback; le troisième champ est caractérisé par le mot réservé « CB » indiquant que c'est une Callback qui est associée au Widget; le quatrième champ correspond au nom de la fonction et le cinquième champ, optionnel, indique l'éventuel argument à passer à la Callback. De plus, « ui2c » propose une instruction supplémentaire de syntaxe similaire permettant d'associer un Event Handler à un Widget. La syntaxe de cette instruction est la suivante :

  ```
  awidget EventHandlertype EH function [client_data [flag]]
  ```

- **La création d'un menu**

  Les instructions de création de menu ont une syntaxe strictement identique à la syntaxe de l'instruction de création d'un Widget. La seule différence se situe dans la syntaxe de l'instruction de création d'un sous-menu où un troisième champ spécifiant le nom symbolique du sous-menu est nécessaire.

Les dernières différences de syntaxe avec « xweb » sont que les lignes de codes précédées du mot réservé « LITERAL » sont placées dans le fichier C exactement à l'endroit où elles se trouvent dans le fichier descriptif. Un nouveau mot réservé, « HEADER », permet d'insérer des lignes de code qui seront placées en tête du fichier source écrit en C. Enfin, « ui2c » permet également de traduire un fichier descriptif unique en un fichier source contenant deux fonctions C distinctes. Le concepteur doit simplement utiliser dans le fichier descriptif l'instruction réservée à cet effet :

```
NEW_FUNC anotherCreateFunc
```

où « NEW_FUNC » est un mot réservé et « anotherCreateFunc » est le nom de la deuxième fonction C que l'utilisateur veut créer.

Chapitre 7 : Choix et difficultés de conception du didacticiel

Peu importe le type de l'application impliqué, le processus de développement d'une application voit toujours le concepteur de celle-ci confronté à des choix ou à des difficultés qu'il doit surmonter de façon efficace s'il veut que son application soit de bonne qualité. Le but de ce chapitre est de présenter au lecteur les choix principaux que nous avons dû faire, ainsi que les principales difficultés auxquelles nous avons été confrontés tout au long du processus de développement de notre didacticiel. Le lecteur pourra ainsi se faire une idée des choix et difficultés typiques rencontrés lors du développement d'un didacticiel dans un environnement Xwindow.

1. Le choix d'un type de représentation pour un réseau de sous-réseaux

Une des principales difficultés rencontrées lors du développement du didacticiel a été de faire le choix d'un type de représentation graphique d'un réseau de sous-réseaux à l'écran. Avant d'arriver à la représentation graphique définitive, nous avions choisi un type de représentation qui, après concertation avec le commanditaire du didacticiel, s'est avérée incorrecte vis-à-vis de l'objet même de l'application qui est l'enseignement.

1.1. Le premier type de représentation choisi pour un réseau de sous-réseaux

Le premier type de représentation choisi pour un réseau de sous-réseaux était une représentation d'un ensemble de sous-réseaux disséminés sur une carte géographique et reliés entre eux par des lignes de communication. L'image d'une telle représentation aurait eu l'apparence de la figure 7.1. dans laquelle les sous-réseaux sont représentés par des cercles reliés entre eux par des lignes figurant les lignes de communication. Un tel type de représentation avait été choisi afin d'obtenir une représentation la plus réaliste possible. Cela aurait en fait constitué une représentation figurative du modèle sous-jacent.

Figure 7.1. : Le premier type de représentation envisagé pour un réseau de sous-réseaux

L'étudiant utilisant le didacticiel aurait dû, pour sélectionner les machines communicantes, d'abord sélectionner un des sous-réseaux représentés afin d'en obtenir une vue détaillée et pouvoir alors sélectionner la machine voulue. Le problème à ce niveau était de choisir une représentation graphique de la vue détaillée d'un sous-réseau. Toujours dans le but d'obtenir une représentation la plus réaliste possible, la première représentation de la vue détaillée d'un sous-réseau était constituée d'un ensemble de représentations figuratives d'ordinateurs reliées entre elles par de simples lignes représentant le câblage du sous-réseau. Dans le cas de la représentation de la vue détaillée d'un sous-réseau Ethernet, l'image obtenue à l'écran aurait eu l'aspect de la figure 7.2.

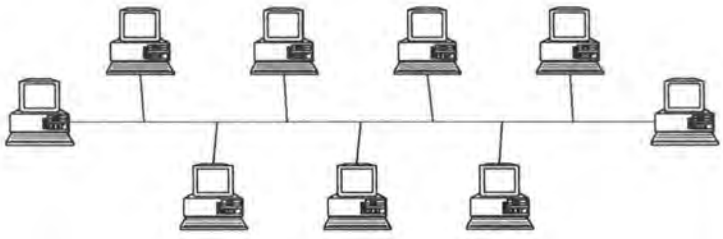


Figure 7.2. : La représentation de la vue détaillée d'un sous-réseau Ethernet

On voit dans la figure 7.2. la représentation d'un ensemble d'ordinateurs reliés à une ligne horizontale représentant le « backbone » du sous-réseau Ethernet. Une telle représentation permet à l'étudiant de deviner directement ce qui est représenté; sauf sur un point, il est impossible à l'étudiant de déterminer quelle machine joue le rôle de passerelle vers l'extérieur. C'est pourquoi une première modification de cette représentation a été décidée : afin de pouvoir déterminer quelle machine est une passerelle, toutes les représentations de passerelles sont l'origine d'une deuxième ligne représentant le câble de sortie vers le domaine extérieur. Si on assume le fait que la machine représentée à l'extrême gauche de la figure 7.2. est une passerelle, la nouvelle représentation de la vue détaillée du sous-réseau Ethernet est telle présentée dans la figure 7.3.

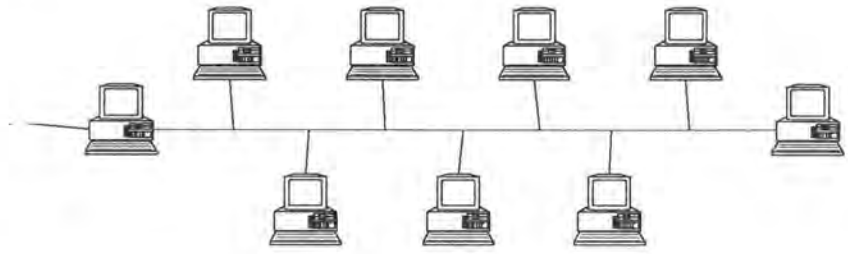


Figure 7.3. : La nouvelle représentation de la vue détaillée d'un sous-réseau Ethernet.

L'étudiant voit ainsi directement quelle machine est une passerelle et quelle machine est un simple hôte.

1.2. Le type de représentation définitif pour un réseau de sous-réseaux

Après concertation avec le commanditaire du didacticiel, il a été décidé que cette première représentation d'un réseau de sous-réseaux n'était pas correcte du point de vue de sa valeur pédagogique. En effet, bien que réaliste, cette représentation ne traduit pas les notions et principes de fonctionnement d'Internet. C'est pourquoi un nouveau type de représentation moins réaliste mais plus pédagogique a été conçu et implémenté pour notre didacticiel.

1.2.1. La représentation d'une machine

La première chose à avoir été repensée est la représentation graphique des machines constituant l'interconnexion. Le concept de machine dans le domaine des télécommunications est organisé en terme de couches représentant les divers protocoles utilisés en fonction du réseau utilisé. Dans le cas d'Internet, ces couches sont au nombre de trois : la couche IP, la couche transport (mettant généralement en oeuvre le protocole TCP) et la couche application reprenant les diverses applications utilisant les deux couches inférieures.

Au vu de cette organisation, il a été décidé qu'une machine serait désormais représentée sous forme d'un cylindre divisé en trois parties représentant chacune une des couches présentées ci-dessus. De plus, une passerelle n'assurant que le transit des datagrammes IP d'un sous-réseau à l'autre, elle n'est concernée que par la couche IP. La représentation d'une passerelle doit donc être différente de la représentation d'un hôte classique; cette représentation a la forme d'un cylindre dont la hauteur est égale à la hauteur de la partie représentant la couche IP dans la représentation d'une machine classique. La figure 7.4. montre l'aspect de la représentation d'un hôte classique et d'une passerelle.

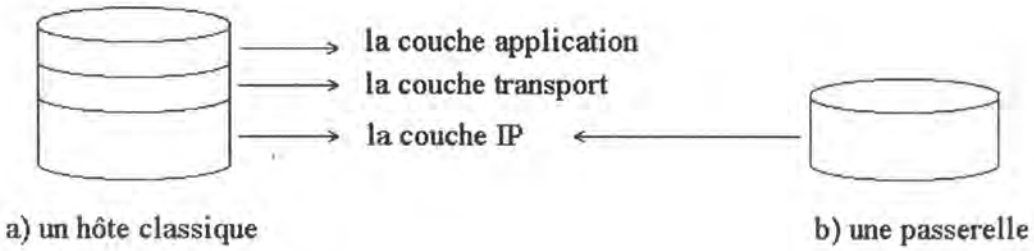


Figure 7.4. : La représentation d'un hôte classique et d'une passerelle.

1.2.2. La représentation de la vue détaillée d'un sous-réseau

La représentation de la vue détaillée d'un sous-réseau a également été modifiée. Tout d'abord, la représentation des machines correspond à la représentation décrite plus haut; ces représentations de machines sont toujours reliées entre elles par des lignes figurant le câblage du sous-réseau. Ensuite, les passerelles ne sont dessinées que partiellement dans la représentation détaillée d'un sous-réseau; ce mécanisme permet de traduire le fait qu'une passerelle est en fait une connexion vers le domaine extérieur.

La représentation de la vue détaillée d'un sous-réseau Ethernet est constituée d'un ensemble d'hôtes représentés par des cylindres à trois couches, reliés à un « backbone » représenté par la ligne horizontal. Un cylindre partiellement dessiné est la représentation d'une passerelle vers le domaine extérieur. Cette représentation est reprise dans la figure 5.2. du chapitre 5.

1.2.3. La représentation d'un réseau de sous-réseaux

Comme nous l'avons précisé au chapitre 5 de ce mémoire, notre didacticiel offre la possibilité d'obtenir deux vues différentes de l'interconnexion : une vue de l'interconnexion en tant que réseau de machines et une vue de l'interconnexion en tant que réseau de sous-réseaux. Quelle que soit la vue sélectionnée par l'utilisateur, la représentation définitive de l'interconnexion est totalement différente de la première représentation envisagée.

La nouvelle représentation ne reprend que la représentation de l'ensemble des machines appartenant aux sous-réseaux qui constituent l'interconnexion. La représentation de ces machines est organisée en sous-réseaux si la vue de l'interconnexion en tant que réseau de sous-réseaux est demandée. Cette représentation, plus simple que la première envisagée, permettra, à notre avis, à l'utilisateur de comprendre réellement ce qu'est une interconnexion plutôt que de voir, comme c'était le cas dans la première représentation, un ensemble de sous-réseaux reliés par des lignes de communication sans pour autant comprendre le principe de l'interconnexion. On se rend immédiatement compte de « l'efficacité pédagogique » de ce type de représentation vis-à-vis du premier type de représentation envisagé.

Les sous-réseaux sont représentés par des cadres contenant des représentations de machines et les passerelles ont une représentation qui couvre à chaque fois les deux sous-réseaux qu'elles connectent ensemble. Le lecteur trouvera une représentation d'interconnexion à la figure 5.6. du chapitre 5.

Enfin, il est important de noter que l'action de sélectionner une machine qui aurait été particulièrement complexe si le premier type de représentation avait été adopté est triviale dans le cas de ce second type de représentation.

2. L'utilisation des ressources

Une autre difficulté à laquelle nous avons été confrontés lors du développement de notre didacticiel est liée à l'environnement de programmation Xwindow; il s'agit de l'utilisation des ressources des Widgets. Comme nous l'avons précisé au chapitre 5, la ressource est à un Widget ce qu'une variable d'instance est à un objet. Le comportement d'un Widget est déterminé à un moment donné par la valeur de toutes ses ressources. Il est possible d'affecter une valeur à une ressource de différentes manières : lors de la création du Widget, lorsque le Widget est déjà créé ou encore à l'aide d'un fichier de ressources qui est également accessible à l'utilisateur final. Si le concepteur initialise explicitement une ressource au moment de la création d'un Widget ou d'une application, l'utilisateur final ne pourra pas modifier la valeur de cette ressource à l'aide du fichier de ressources.

« Ainsi, l'écriture d'une application est un exercice basé sur l'équilibre délicat des ressources imposées par le programmeur pour garantir un certain style à son application et des ressources pouvant être personnalisées par l'utilisateur final. » [MULLER93]

2.1. L'initialisation des ressources

C'est cet équilibre délicat que nous avons essayé de trouver lors du développement de notre didacticiel. Nous avons choisi pour cela de faire le plus d'initialisations de ressources possibles à partir du fichier de ressources. Deux raisons sont à l'origine de ce choix :

- la première raison est le fait qu'initialiser une ressource à partir du fichier de ressources permet au concepteur de modifier les valeurs de ces ressources et de visualiser les effets consécutifs de cette modification sans devoir recompiler l'application. Ce genre de pratique peut être utile lorsque, par exemple, le concepteur doit trouver une taille correcte pour une fenêtre ou une police de caractères qui lui plaisent. Le fait qu'il y ait un maximum de ressources reprises dans le fichier permet donc de modifier un maximum de paramètres de l'interface sans devoir recompiler l'application.

- la deuxième raison est que cela permet à un utilisateur, expert ou non dans le domaine de Xwindow, de personnaliser au maximum le didacticiel. Cela permettra donc de recréer un environnement à l'utilisateur et d'augmenter la motivation de celui-ci.

La conséquence directe de ce choix est que le fichier de ressources de notre didacticiel est assez conséquent; nous l'avons donc structuré afin que l'utilisateur final puisse retrouver rapidement les ressources qu'il désire éventuellement personnaliser.

2.2. La structure du fichier de ressources

Le fichier de ressources est divisé en différentes sections qui correspondent chacune à une fenêtre spécifique de l'interface. Chacune de ces sections est alors subdivisée en sous-sections qui reprennent les ressources spécifiques à un aspect particulier de l'interface. L'ensemble de ces sous-sections est constitué des éléments suivants :

- **Le comportement général**

  Cette sous-section reprend toutes les ressources caractérisant le comportement général de la fenêtre Xwindow impliquée. Un exemple d'une ressource appartenant à cette sous-section est la ressource « doubleBuffer » qui indique si l'utilisateur veut utiliser ou non la technique de « double buffer » pour une fenêtre graphique à affichage rapide.

- **L'apparence générale**

  Cette sous-section contient toutes les ressources spécifiant l'apparence générale de la fenêtre Xwindow référencée. La ressource « FontList » indiquant la police de caractères à utiliser pour le texte apparaissant dans la fenêtre est un exemple d'une ressource de cette sous-section.

- **Les positions**

  Cette sous-section reprend principalement les ressources indiquant des contraintes de position de la fenêtre référencée ou des contraintes de position pour les Widgets appartenant à la fenêtre référencée. La ressource « alignment » qui spécifie l'alignement (gauche, droite ou centrée) d'un texte dans une zone de texte est une des ressources que l'on retrouvera dans cette sous-section.

- **Les couleurs**

  Cette sous-section est constituée des ressources spécifiant les couleurs des différents éléments constituants de la fenêtre référencée. Les ressources généralement indiquées à cet endroit sont les couleurs de fond (background), les couleurs de texte (foreground) ou encore des ressources propres à notre didacticiel telles que, par exemple, la couleur de mise en évidence d'une machine sélectionnée (host1SelectionColor et host2SelectionColor).

- **Les « strings »**

  Cette sous-section contient l'ensemble des chaînes de caractères qui constitue le libellé des Widgets qui sont pourvus de cette ressource. La ressource indiquée ici est la ressource « labelString ». Par exemple, si l'utilisateur veut que le bouton implémenté par le Widget « mConfig » soit présenté par le libellé « Configuration » à l'écran, il peut écrire la ligne suivante dans le fichier de ressources :

  ```
  TcpIp*mConfig.labelString : Configuration
  ```

  où TcpIp est le nom de l'application dans l'interface de laquelle le bouton apparaîtra. C'est l'ensemble des ressources de cette sous-section qui permet de prévoir

plusieurs versions linguistiques d'une même application : il suffit pour cela de prévoir plusieurs fichiers de ressources contenant chacun les spécifications de « strings » dans une langue différente. L'utilisateur ne doit alors plus que copier le fichier contenant la version linguistique désirée comme fichier de ressources principal pour obtenir l'application dans la langue voulue.

- **Les tailles**

  Cette sous-section reprend les différentes ressources spécifiant des contraintes de taille à la fenêtre référencée ou aux Widgets enfants de cette fenêtre. Un exemple d'une ressource spécifiée dans cette sous-section est la ressource « minWidth » qui spécifie la largeur minimum qu'une fenêtre peut avoir. Ce genre de spécification peut être utile pour spécifier la taille minimale que peut prendre une fenêtre graphique avant que son contenu devienne illisible.

Bien que l'utilisateur final du didacticiel puisse modifier à sa guise toutes les ressources reprises dans le fichier ressources, cette pratique n'est pas à conseiller : il vaut mieux que l'utilisateur se contente de personnaliser des ressources telles que les couleurs ou les « strings »; s'il « s'amusait » à modifier d'autres types de ressource, le résultat obtenu pourrait s'avérer surprenant.

Même si beaucoup de ressources sont initialisées à l'aide du fichier de ressources, il n'en reste pas moins que certaines d'entre elles sont encore initialisées directement dans le code afin d'assurer une certaine personnalité au didacticiel. Ces différentes ressources sont, entre autres, des contraintes de position, le nom (autrement dit le libellé) de l'application ou encore la taille par défaut des différentes fenêtres.

3. Les difficultés de programmation

Nous avons également rencontré des difficultés de programmation lors de la partie « médiatisation » (cfr. chapitre 1) du processus de développement de notre didacticiel. Quelques-unes de ces difficultés sont directement liées à l'environnement Xwindow alors que d'autres sont consécutives à l'utilisation mixte du C et du C++. Il nous semble intéressant de préciser ici comment nous avons surmonté ces difficultés.

3.1. Les difficultés de programmation avec Xwindow

Un certain nombre de difficultés de programmation directement liées à l'environnement Xwindow ont ponctué la démarche de développement de notre didacticiel. Une difficulté importante provient du fait de devoir gérer un événement non prévu par Xwindow alors qu'une autre difficulté émane de la gestion graphique de l'animation.

3.1.1. La gestion d'un événement non prévu par Xwindow

Tous les événements n'ont pas été explicitement prévus par les concepteurs de Xwindow; c'est le cas par exemple de l'événement « MultiClick » dans une fenêtre graphique. Or, nous avions besoin de cet événement pour gérer la sélection d'une machine à l'aide de la

souris; cette sélection se fait effectivement en cliquant deux fois sur la représentation de la machine à sélectionner.

L'événement « Simple Click dans une fenêtre graphique » est toutefois explicitement prévu par Xwindow et c'est lui qui permet de gérer l'événement précédent via l'utilisation d'une petite astuce. Cette astuce consiste à associer une fonction de type Event Handler au Widget représentant la fenêtre graphique. Cette fonction sera appelée chaque fois que l'événement « Simple Click » surviendra dans la fenêtre graphique.

Quand la fonction est appelée, elle vérifie à l'aide d'un témoin si c'est le premier « click » qu'elle enregistre. Si c'est le premier « click », elle crée un Timer[1] avec un délai d'appel dont la durée est égale au temps maximum autorisé entre deux « simple click » pour que ceux-ci soient considérés comme un « double click ».

Si la fonction est de nouveau appelée avant que le délai du Timer ne se soit écoulé, donc avant que le Timer ne soit appelé, c'est qu'un nouvel événement « simple click » est survenu dans la fenêtre graphique dans le délai autorisé pour que ce « simple click » soit considéré comme le second « click » d'un « double click ».

La fonction se sert ici du Timer comme témoin pour se rendre compte s'il s'agit d'un premier « click » ou d'un second. Si le Timer n'existe pas, c'est qu'il s'agit d'un premier « click » et la fonction reprend toute la procédure à partir de la création du Timer. Par contre, si le Timer existe, c'est qu'il s'agit du second « click » d'un « double click », la fonction supprime alors le Timer qui n'est plus nécessaire et entame la procédure de gestion de l'événement « DoubleClick dans une fenêtre graphique ».

Par contre, si la fonction Timer est appelée avant que la fonction Event Handler ne soit appelée une seconde fois, c'est que le délai maximum autorisé entre deux « clicks » pour que ceux-ci soient considérés comme un « double click » s'est écoulé. Dans ce cas, le Timer se détruit lui-même pour ne plus pouvoir servir de témoin et déclenche éventuellement la procédure de gestion de l'événement « Simple Click dans une fenêtre graphique ».

3.1.2. La gestion graphique de l'animation

Aucune fonction n'est prévue dans Xwindow pour gérer une animation. Nous avons donc dû gérer nous-mêmes l'animation graphique du transit des datagrammes dans l'interconnexion représentée. Pour rappel, cette animation graphique est le déplacement d'un point de couleur le long d'un chemin graphique calculé par les fonctions « compute_graphic_way » et « compute_subnet_graphic_way » du module « anim » (cfr. chapitre 5).

La principale difficulté provient du fait de faire bouger le point de couleur à l'écran; nous avons utilisé pour cela la méthode suivante : le principe est de tracer le point de couleur à l'origine du chemin graphique et ensuite d'exécuter en boucle la procédure suivante jusqu'à ce que tout le chemin graphique soit parcouru :

- attendre un court délai;
- effacer la représentation du point de couleur;
- retracer le point de couleur un peu plus loin sur la route graphique.

[1] Un Timer est une fonction qui est appelée après un certain temps.

Une attention toute particulière doit être accordée à deux choses :

- le délai entre l'affichage du point de couleur à l'écran et son effacement doit être, d'une part, suffisamment court pour que l'oeil humain ne puisse pas percevoir ses tracés et effacements successifs et avoir ainsi l'impression d'un mouvement continu et, d'autre part, suffisamment long pour que l'ordinateur ait le temps d'afficher le point avant que celui-ci ne soit effacé. Notre expérience a montré que ce délai devait avoir une valeur comprise entre 15 et 100 millisecondes.

- les tracés et les effacements successifs des points de couleur ne doivent pas altérer ce qui se trouve déjà à l'écran. Xwindow a prévu pour cela une règle particulière de tracé : la règle du « ou exclusif ». Lorsque cette règle est appliquée, Xwindow opère un « ou exclusif » entre ce qu'on veut tracer et ce qui est déjà tracé; Xwindow trace alors le résultat de cette opération. De cette façon, il suffit de retracer la même chose au même endroit pour retrouver l'état initial de l'écran. Donc, l'étape d'effacement du point de couleur consiste simplement à retracer le point au même endroit.

3.2. La programmation mixte en C et C++

Comme nous l'avons déjà précisé, l'implémentation du didacticiel s'est faite simultanément dans deux langages : en C++ en ce qui concerne l'implémentation des objets recouvrant les notions d'Internet et en C pour tout ce qui concerne l'implémentation de l'interface et de la gestion des événements. Cette organisation complexe demande une attention particulière au niveau du compilateur et au niveau de l'implémentation des fonctions d'interface entre les deux langages.

C'est le compilateur C++ qui autorise l'utilisation simultanée du C et du C++. Il suffit pour cela de compiler les parties écrites en C avec le compilateur C et les parties écrites en C++ avec le compilateur C++. Une fois que tout est compilé, il faut créer les liens entre ces différentes parties à l'aide du compilateur C++.

Cependant, comme la notion d'objet n'existe pas en C, il est impossible d'appeler directement une méthode d'un objet C++ à partir du C. Il est nécessaire pour cela de créer dans la partie C++ des fonctions d'interface entre les deux langages. Une telle fonction d'interface a les particularités suivantes :

- elle a un prototype[2] dans les fichiers « include » du C et du C++. De plus, ce prototype sera précédé du code « extern C » dans le fichier « include » du C++. Cela indique simplement au compilateur C++ que cette fonction peut être appelée à partir du code C.

- elle a pour arguments tous les arguments de la méthode dont elle est la fonction d'interface et un argument supplémentaire qui est un pointeur vers l'objet ou, autrement dit, une référence à l'objet qui comprend la méthode dont elle est la fonction d'interface.

[2] Un prototype est une entête de fonction définissant la syntaxe à utiliser lors de l'appel de cette fonction.

La fonction d'interface peut alors à partir de la référence à l'objet qu'elle a reçue par argument appeler la méthode de l'objet désiré. Lorsque l'on veut dès lors appeler une méthode d'un objet particulier à partir du C, il suffit d'appeler la fonction d'interface correspondante en lui passant la référence de l'objet et les arguments de la méthode.

Enfin, pour que la référence à l'objet soit disponible à partir du C, il faut que l'objet soit déclaré dans la zone C++ comme un pointeur vers une classe et qu'une variable de même nom soit déclarée dans la zone C comme un pointeur vers une structure. Une classe étant une structure particulière, il n'y a aucun problème de compatibilité entre les déclarations C et C++.

Chapitre 8 :
Evaluation, critique et perspectives

L'objet particulier d'un didacticiel qu'est l'enseignement nécessite, si les concepteurs sont rigoureux, de terminer le processus de développement du didacticiel par une phase d'évaluation. Cette phase consiste en un contrôle de la qualité du produit fini. Une telle tâche n'est pas facile à mener de façon critique (cfr. chapitre 1); elle est toutefois facilitée par l'utilisation de grille d'analyse standard créée par des spécialistes du domaine de l'enseignement assisté par ordinateur.

Le but de ce chapitre est de fournir une évaluation du didacticiel basée sur une de ces grilles d'analyse. Cette évaluation sera suivie d'une critique de notre travail. Il nous semble naturel de fournir ensuite quelques pistes pour une continuation du développement du didacticiel : c'est ce que nous appellerons perspectives.

1. L'évaluation

Nous nous baserons pour l'évaluation de notre didacticiel sur la grille d'analyse fournie dans l'ouvrage « Evaluer les Logiciels de Formation » [MEDA90] qui nous a été recommandé par Jean Donnay, Directeur de la Cellule Education et Technologie des Facultés Universitaires Notre Dame de la Paix.

Tous les points repris dans cette grille d'analyse n'ont pas été repris. Nous avons suivi la démarche proposée par l'auteur et qui est scindée en plusieurs étapes :

a) la première étape consiste à déterminer le pôle d'intérêt auquel on s'intéresse lors de l'évaluation. Les trois pôles d'intérêt envisageables sont la conception, l'utilisation et la diffusion;

b) la seconde étape consiste à choisir les préoccupations d'évaluation en fonction des pôles d'intérêt retenus lors de la première étape;

c) la troisième étape consiste ensuite à retenir certains aspects relatifs aux préoccupations d'évaluation choisies à la seconde étape;

d) la quatrième étape consiste alors à se référer aux fiches de question indiquées par les aspects retenus lors de la troisième étape. Ce sont ces questions qui permettent d'évaluer le didacticiel;

e) la dernière étape, enfin, consiste à faire une synthèse des réponses aux questions de la quatrième étape et de conclure l'évaluation.

Dans le but d'être clair et concis, nous nous contenterons de donner les observations auxquelles nous sommes arrivés en répondant aux questions retenues à l'aboutissement des quatre premières étapes. Chacune de ces observations sera donc relative à un aspect choisi en fonction des préoccupations d'évaluation relatives au pôle d'intérêt retenu.

Nous avons privilégié la conception du didacticiel comme pôle d'intérêt lors de notre évaluation. La raison de ce choix est que le didacticiel que nous avons développé n'est pas encore au stade d'utilisation et de diffusion, ce qui ne nous permet pas de privilégier ces pôles d'intérêt pour l'évaluation de celui-ci. Les préoccupations principales qui ont retenu notre intérêt sont la maîtrise des étapes de création, les tests techniques et la validation pédagogique du produit.

1.1. La maîtrise des étapes de création

L'analyse de la maîtrise des étapes de création nous permettra de vérifier entre autres le bien-fondé de la raison d'être du didacticiel, l'adéquation du produit aux caractéristiques de la population cible ou encore l'exactitude de la définition des différents paramètres propres à la création d'un didacticiel.

La raison d'être du produit

« La raison d'être du produit concerne le besoin ou le problème qui s'est trouvé à l'origine de la création du produit. » [MEDA90]

Les besoins qui sont à la base de la réalisation du produit sont représentés par le manque de moyens didactiques pour l'enseignement de notions des télécommunications. C'est en effet ce manque de moyens qui est à la base de la demande de création de notre didacticiel. De plus, un tel didacticiel permet aux étudiants de surmonter certaines difficultés de compréhension qui ne sont pas toujours faciles à surmonter grâce à un enseignement classique.

Enfin, ce didacticiel n'est pas un produit fini en ce sens qu'il constitue une base pour la continuation du projet et pour le développement de nouvelles fonctionnalités visant à présenter des notions supplémentaires du domaine des télécommunications.

Les caractéristiques de la population cible

Un certain nombre de caractéristiques de la population cible étaient disponibles lors de la création du didacticiel; ces caractéristiques sont l'âge, le niveau de qualification, les pré-requis de la population cible ou encore l'attitude de la population cible par rapport aux objectifs. Nous renvoyons le lecteur à la première partie du chapitre 5 pour une description complète de ces caractéristiques dans notre cas.

Toutes ces informations nous ont permis de créer un didacticiel qui correspond le plus possible aux attentes des utilisateurs finaux.

L'insertion dans le contexte de formation

Le didacticiel a été conçu pour répondre à un besoin précis du formateur, il sera intégré dans le programme des cours de télécommunications et, plus précisément, dans le cadre de l'enseignement des notions propres à Internet. De plus, ce didacticiel pourra être utilisé de deux manières : soit par le professeur afin d'illustrer son cours, soit par un étudiant qui utilisera alors le didacticiel dans une situation « d'auto-apprentissage ».

Les critères de production

« Les critères de production sont ceux qui permettent de juger de l'opportunité de développer un produit. » [MEDA90]

Le produit n'est pas nouveau, il existe en effet au moins une autre application (OpNet) qui permet de simuler graphiquement le fonctionnement de réseaux en général et d'Internet en particulier. Cependant, cette application coûtant particulièrement cher à l'achat, l'intérêt d'avoir créé un produit équivalent pour un coût beaucoup plus faible est évident. De plus, la production de notre didacticiel entraînera, nous l'espérons, des gains de formation tels que, par exemple, la diminution du temps de compréhension des principes de fonctionnement d'Internet.

L'implémentation

La question principale que l'on se pose à ce niveau est de savoir si on dispose de renseignements permettant de comprendre la structure logicielle du didacticiel.

Ce mémoire constitue en lui-même une source de documentation relative au didacticiel développé. Plus précisément, la quatrième partie du chapitre 5 fournit toute l'information nécessaire en ce qui concerne les langages utilisés (le C et le C++) et l'architecture logicielle de notre didacticiel (la découpe en modules). De plus, de nombreux commentaires ont été insérés dans le code même pour assurer une meilleure compréhension de celui-ci.

Enfin, l'implémentation a été faite de telle manière à rendre le didacticiel portable vers d'autres sites et un certain nombre de paramètres du didacticiel sont directement accessibles à l'utilisateur final via le fichier de ressources.

En conclusion de ce point, nous pouvons déduire de l'analyse de la maîtrise des étapes de création que notre didacticiel a été créé en réponse à un besoin réel d'outils didactiques pour l'enseignement de notions des télécommunications, que le produit correspond aux caractéristiques de la population cible, qu'il s'insère directement dans un contexte de formation et que son implémentation est suffisamment documentée. Enfin, nous espérons qu'il est suffisamment structuré pour servir de base dans le cas d'une continuation du projet.

Le seul point faible à ce niveau est qu'il existe déjà au moins une autre application équivalente. Ce défaut est néanmoins amoindri par le fait que le coût de développement de notre didacticiel est très bas vis-à-vis du coût d'achat de l'application existante.

1.2. Les tests techniques

Cette partie de l'évaluation nous permettra de déterminer les caractéristiques techniques de l'environnement dans lequel le didacticiel est utilisé. Les caractéristiques techniques importantes sont, d'une part, celles du matériel sur lequel le didacticiel peut tourner et, d'autre part, celles de la manipulation même du didacticiel par l'utilisateur final. Cette partie nous permettra également de juger de l'autonomie d'un utilisateur ainsi que de la souplesse d'utilisation du didacticiel.

Les caractéristiques techniques du matériel

Le didacticiel a été développé sur stations HP dotées d'écran à haute résolution graphique. Les caractéristiques techniques de ce matériel sont les suivantes :

Ordinateur :HP 9000/700.
Système d'exploitation : HP-UNIX.
Mémoire vive : 32 megabytes.
Mémoire disque : 2 gigabytes.
Périphériques d'entrée : souris, clavier.
Périphérique de sortie : écran graphique à haute résolution.
Langages de programmation : C et C++.

L'Institut d'Informatique des FUNDP étant propriétaire d'une telle station de travail, le didacticiel est directement utilisable sur le site de formation.

Les caractéristiques techniques de la manipulation

Les différentes commandes que peut exécuter l'utilisateur sont toutes les fonctionnalités déclenchées par l'activation de boutons graphiques à l'aide de la souris. Une telle organisation facilite grandement la manipulation du didacticiel par l'apprenant.

Dans tous les cas, l'information dispensée décrit directement soit ce qui se passe au niveau de la simulation, soit ce que l'utilisateur doit faire par la suite. L'utilisateur peut donc déduire à tout moment où il en est dans l'utilisation du didacticiel. Cependant, par manque de temps, aucune fonction d'aide en ligne n'a été prévue dans le didacticiel. Cela constitue un des gros défauts de notre produit.

L'autonomie de l'utilisateur

L'autonomie d'un utilisateur faisant partie de la population cible est réalisée lorsque celui-ci peut interagir avec le didacticiel sans avoir recours à aucune aide technique extérieure.

Dans son état actuel, le didacticiel peut être utilisé par un utilisateur unique n'ayant pas de compétences particulières en informatique. Mis à part l'apprentissage des principes d'utilisation de l'environnement Xwindow, l'utilisateur n'a besoin de connaître aucune notion particulière. Ce mémoire constitue une source de documentation largement suffisante pour

utiliser le didacticiel. Le degré de solidité du didacticiel est satisfaisant, c'est-à-dire que le didacticiel ne s'interrompt pas lorsque l'utilisateur effectue une action non prévue.

La souplesse d'utilisation du didacticiel

Etant donné le type d'activité qui est proposé à l'utilisateur, à savoir le pilotage d'une simulation, l'utilisation du didacticiel est relativement souple, l'utilisateur pouvant décider lui-même de la prochaine action qu'il va exécuter.

De plus l'utilisation de la souris pour manipuler le didacticiel constitue un facteur supplémentaire en faveur de la souplesse d'utilisation de l'application.

La structure des menus du didacticiel correspond à la structure de menus que l'on retrouve dans la plupart des applications existantes et qui fournissent également des menus. Cela permet à l'utilisateur déjà familiarisé avec ce type d'environnement de ne pas être perdu face à une présentation marginale.

Comme nous l'avons déjà précisé plus haut, un des gros défauts de notre didacticiel est de ne fournir aucune fonction d'aide si ce ne sont les messages d'assistance situés dans le bas de l'écran. Ces fonctions étaient initialement prévues, mais nous n'avons pas eu le temps matériel suffisant pour les implémenter.

En conclusion, nous pouvons dire que l'analyse de ces différents points nous a permis de préciser que le didacticiel a été développé sur du matériel performant; mais également que les caractéristiques de manipulation sont satisfaisantes en ce qui concerne les caractéristiques techniques de manipulation, l'autonomie de l'utilisateur et la souplesse d'utilisation. Il faut également noter que les tests techniques nous ont permis de déceler une lacune important de notre didacticiel : l'absence de fonction d'aide à l'utilisateur. C'est donc la première chose à corriger par les personnes qui poursuivront éventuellement le développement de ce produit.

1.3. La validation pédagogique du produit

De par son objet particulier qu'est l'enseignement, un didacticiel doit avoir un contenu pédagogique solide et correct. C'est pourquoi il est important, lors de son évaluation, d'apporter une attention toute particulière à ce contenu pédagogique. La validation pédagogique du produit nous permettra entre autres de vérifier la définition des objectifs, la démarche pédagogique adoptée, les activités sollicitées chez l'apprenant mais également de vérifier l'adéquation de la conception de l'interface vis-à-vis de ces facteurs pédagogiques.

La définition des objectifs

L'objectif pédagogique de notre didacticiel est que l'étudiant apprenne et comprenne les principales notions du fonctionnement d'une interconnexion respectant le modèle d'Internet. Cet objectif principal, ainsi que sa décomposition en objectifs secondaires sont décrits en détail dans la première partie du chapitre 5 (point 1.1.1.).

La démarche pédagogique

La démarche pédagogique est la stratégie mise en oeuvre dans le didacticiel pour atteindre les objectifs pédagogiques. La stratégie mise en oeuvre par notre didacticiel est la simulation. La simulation est le moyen idéal pour motiver les étudiants faisant partie de la

population cible car elle place celui-ci devant une situation ou un environnement réel et l'invite alors à piloter la simulation du fonctionnement du modèle représenté par cet environnement.

De plus, une telle stratégie permet à l'étudiant d'avancer à son propre rythme dans son apprentissage et à revoir autant de fois qu'il le désire une partie qu'il ne comprend pas bien. Nous renvoyons le lecteur au chapitre 5 (point 1.2.) pour une description détaillée de la stratégie mise en oeuvre dans notre didacticiel.

Les activités sollicitées chez l'apprenant

Les activités sollicitées chez l'apprenant sont directement fonction de la stratégie mise en oeuvre dans le didacticiel. Dans notre cas, ces activités sont les actions liées au pilotage de la simulation du fonctionnement de l'interconnexion représentée. A tout moment, le didacticiel indique l'action que l'utilisateur peut exécuter pour mener à bien la simulation. L'utilisateur est toutefois toujours libre de choisir une autre fonctionnalité proposée par le didacticiel.

Un point faible de notre didacticiel est qu'il ne fournit aucune fonction permettant de vérifier l'atteinte de l'objectif par l'apprenant; aucune activité de contrôle de l'assimilation n'est donc sollicitée chez l'apprenant par le didacticiel. Le professeur devra donc vérifier lui-même l'assimilation des notions présentées auprès des étudiants.

L'interactivité

« L'interactivité entre deux partenaires suppose la possibilité d'action et de réaction de chacun d'eux. Un logiciel de formation sera dit interactif s'il réagit de manière diversifiée et adaptée aux différentes réponses des utilisateurs et s'il permet à ces derniers d'agir sur le déroulement du logiciel de formation. » [MEDA90]

Notre didacticiel peut être qualifié d'interactif : il permet à l'utilisateur de faire à peu près ce qu'il veut quand il le veut. L'utilisateur peut en effet à n'importe quel moment déclencher n'importe quelle fonctionnalité offerte par le didacticiel dans la limite des contraintes du modèle sous-jacent. Les contraintes du modèle imposent en effet quelques exceptions telles que, par exemple, la fonctionnalité de lancement de l'animation qui ne peut être lancée que lorsque deux machines sont sélectionnées; mais dans ce cas, le didacticiel ne permet pas à l'utilisateur de déclencher l'animation. En règle générale, l'utilisateur peut déclencher à tout moment n'importe quelle fonctionnalité rendue accessible par le didacticiel. De son côté, le didacticiel adapte **directement** son interface en fonction des actions exécutées par l'utilisateur. Cela traduit bien l'interactivité qui existe entre les deux acteurs que sont le didacticiel d'une part et l'apprenant d'autre part.

Notons encore que l'interactivité est un des facteurs auquel les membres du consortium COLOS accorde le plus d'importance lors du développement de leurs didacticiels. C'est donc tout naturellement que nous avons également privilégié ce facteur.

Cependant, comme l'apprenant pilote lui-même la simulation, il est de son initiative de revenir éventuellement sur des notions importantes et il n'aura peut-être pas le discernement nécessaire pour le faire de lui-même. C'est encore là un point faible de notre didacticiel.

La gestion de l'écran

La gestion de l'écran concerne la disposition spatiale et l'organisation des informations à l'écran. On peut distinguer la gestion du texte, la gestion du graphisme, l'utilisation des couleurs et l'organisation de la page écran.

a) *La gestion du texte*

Notre didacticiel fait peu usage du texte. Notons toutefois que les messages d'assistance indiquant ce que l'utilisateur peut faire ou ce qui est en train de se passer dans la simulation sont situés dans le bas de l'écran de façon lisible. En règle générale, le texte est placé dans les différentes fenêtres en respect des règles ergonomiques et utilise des polices de caractères lisibles. Nous avons veillé à ce que le texte soit explicite chaque fois qu'il est utilisé.

b) *La gestion du graphisme*

Le didacticiel présentant une simulation graphique, nous avons porté une attention toute particulière à cette dimension. La scène représentée, bien qu'évolutive, est constamment affichée au même endroit; ce qui permet d'assurer une cohérence spatiale à l'image.

Dans la mesure où l'utilisateur peut arrêter et redémarrer l'animation quand il le désire, le temps de présentation du graphisme est adéquat pour tous les utilisateurs. L'image d'une machine telle qu'elle est représentée est ambiguë; cette ambiguïté est toutefois levée par le fait que cette représentation est la seule pouvant correspondre à une machine.

Enfin, les mouvements sur l'écran en animation ne suivent évidemment pas le mouvement habituel des yeux pendant la lecture, cela provient du fait que l'objet qui est animé suit une trajectoire aléatoire qui dépend des machines sélectionnées par l'utilisateur. Néanmoins, l'utilisateur se rendra vite compte par lui-même de la dépendance de la trajectoire vis-à-vis de ses manipulations.

c) *L'utilisation des couleurs*

L'environnement graphique de Xwindow permet d'utiliser un nombre important de couleurs simultanément à l'écran. Nous avons cependant utilisé un nombre restreint de couleurs afin de ne pas surcharger la représentation graphique. De plus, la plupart des couleurs utilisées sont personnalisables par l'utilisateur à l'aide du fichier de ressources. Nous avons toutefois prévu d'utiliser des teintes proches les unes des autres pour des objets proches les uns des autres. Pour marquer une différence, nous avons utilisé une couleur différente. Par exemple, la couleur de mise en évidence d'une machine sélectionnée tranche nettement vis-à-vis de la couleur d'une machine non sélectionnée.

d) *L'organisation de la page écran*

L'organisation de la page écran que nous avons adoptée respecte les règles ergonomiques de base. Les messages destinés à l'utilisateur sont situés dans le bas de l'écran pour ne pas créer de rupture dans la cohérence spatiale de la fenêtre. De plus, la structure des menus correspond aux standards utilisés par la plupart des applications existantes. Enfin, le multi-fenêtrage est utilisé afin de représenter différents concepts simultanément à l'écran de façon structurée.

Le contenu

Le contenu est la matière sur laquelle porte le produit. Dans notre cas, le contenu est le modèle sur lequel repose la simulation c'est-à-dire le modèle d'Internet. Le lecteur trouvera une description détaillée du contenu de notre didacticiel aux chapitres 4 et 5 (point 1.2.2.). Nous pouvons dire que ce contenu est valide par rapport à la situation de formation, qu'il est à jour et qu'il est pertinent par rapport au niveau des apprenants.

Le feed-back

Le feed-back représente les informations fournies à l'utilisateur suite à ses actions. On peut considérer que notre didacticiel fournit un feed-back approprié en fonction des actions des utilisateurs. En effet, le didacticiel fournit à tout moment une information sur l'état courant de l'utilisation; dès que l'utilisateur exécute une action modifiant cet état, le didacticiel fournit directement l'information adéquate. De plus, le déroulement de la simulation se poursuit comme un feed-back continu pour l'apprenant.

L'adaptabilité pédagogique

L'adaptabilité pédagogique est la faculté d'un produit d'être adapté en vue de le rendre le plus performant pour réaliser les objectifs. Comme le didacticiel propose à l'apprenant de piloter lui-même la simulation, celui-ci peut décider lui-même de son rythme de travail, du temps qu'il désire passer à manipuler le didacticiel ou encore de l'endroit où il désire arrêter son apprentissage. Ces constatations dénotent une adaptabilité de notre didacticiel.

Un certain nombre de points faibles sont cependant à remarquer dans cette faculté d'adaptabilité : d'une part, le didacticiel ne permet pas à un utilisateur de reprendre la simulation là où il s'était arrêté et, d'autre part, il n'est pas possible au formateur de déterminer un degré de liberté d'utilisation en fonction du niveau de formation atteint par les apprenants.

La documentation d'accompagnement

Ce mémoire constitue une source de documentation relative au didacticiel largement suffisante. En ce qui concerne l'utilisation du didacticiel, mis à part l'apprentissage des principes d'utilisation de l'environnement Xwindow, il nous semble que le logiciel réalisé se suffit à lui-même. Vouloir concevoir une documentation reviendrait à éditer une brochure de présentation du didacticiel.

En conclusion, nous pouvons dire que notre didacticiel « tient relativement bien la route » en ce qui concerne ses différentes caractéristiques pédagogiques. Un certain nombre de points faibles ont cependant été soulevés : l'absence de fonctions d'aide, l'absence de fonction de contrôle vis-à-vis de l'atteinte des objectifs par les apprenants, l'impossibilité du formateur de déterminer un degré de liberté d'utilisation ou encore l'impossibilité pour un utilisateur de reprendre la simulation là où il s'était arrêté. Les trois premières lacunes pourront être éliminées assez facilement lors d'une continuation éventuelle du projet. Le dernier défaut par contre s'avère très compliqué à éliminer; la solution serait de sauvegarder totalement l'état du didacticiel au moment où l'utilisateur le quitte mais le caractère de simulation de l'application rend ce type de solution très compliqué étant donné le nombre de paramètres et de variables utilisés.

1.4. Les résultats de l'évaluation

Nous pouvons conclure cette évaluation de manière positive. L'analyse des diverses préoccupations que nous avions sélectionnées nous a permis de nous rendre compte que notre didacticiel, sans être parfait, respecte relativement bien la plupart des caractéristiques déterminées par les spécialistes de l'enseignement assisté par ordinateur comme étant les principales qualités que doit présenter un didacticiel. Notons cependant que le didacticiel reste très simple et ne présente qu'un nombre réduit de notions à l'apprenant. Il ne constitue en fait qu'un début de projet et pourra servir de base pour une continuation de ce projet.

2. Critique

Pour réaliser notre travail, nous avons dû acquérir de nombreuses connaissances dans de multiples domaines. Ainsi nous n'avons pu, dans la mesure de nos possibilités, qu'effleurer les différents sujets présentés. C'est pourquoi nous ne pouvons nullement prétendre que ce mémoire constitue une référence en la matière. Tout au plus constitue-t-il une contribution à l'état actuel du domaine.

Bien que nous l'ayons déjà précisé, il nous semble important de rappeler ici que dans notre travail, le didacticiel développé ne présente qu'une partie des notions que l'on retrouve dans Internet. Cependant, comme notre travail constitue une première expérience dans ce domaine, il nous a semblé plus sage de nous limiter aux notions qui représentent le coeur même du monde d'Internet.

La partie la plus difficile à juger du travail est sans aucun doute la partie de présentation de notre didacticiel. Nous ne pouvons pas prétendre, en effet, avoir réussi à décrire les différentes dimensions de notre application de la façon la plus claire qui soit. Nous pensons cependant que l'architecture que nous avons développée est relativement intéressante. Les objets que nous avons créés nous semblent se prêter assez naturellement aux notions que nous voulions implémenter. La principale difficulté est apparue alors avec l'obligation d'implémenter toute l'interface à l'aide d'un langage ne supportant pas le concept d'objet. C'est pourquoi la description que nous avons donnée de l'architecture logicielle de notre didacticiel peut paraître quelque peu inhabituelle. Il ne faut pas perdre de vue non plus que nous nous sommes occupés de toutes les étapes de développement du didacticiel; aussi, certains éléments qui nous paraissaient évidents n'ont pas été exprimés. De plus, nous avons également souligné le fait que nous avons développé notre didacticiel par « morceaux » ou, autrement dit, par prototypage. Ainsi chaque petite étape du développement a pu être discutée avant de continuer le développement; ce genre de pratique permet d'assurer une qualité accrue au produit fini mais ne facilite sûrement pas la rédaction de l'architecture globale de l'application terminée.

Nous pouvons regretter de n'avoir pas eu suffisamment de temps matériel pour parfaire notre didacticiel; il nous paraît en effet dommage de décrire un produit ne présentant pas de fonctions aussi importantes pour un didacticiel que sont les fonctions d'aide. Malgré cela, nous pensons que la partie du didacticiel que nous avons développée est intéressante. Elle reprend les fonctionnalités principales nécessaires au fonctionnement global de la simulation. Il reste donc à nos successeurs la tâche d'ajouter les dernières fonctionnalités si ceux-ci envisagent de

poursuivre notre projet. Nous espérons que la documentation que constitue ce mémoire les aidera dans cette tâche.

L'implémentation d'une interface dans l'environnement Xwindow, bien que facilitée par les différents outils que nous avons décrits au chapitre 6, est rendue plus difficile par le fait que les bibliothèques de Widgets sont très nombreuses et nécessite donc un apprentissage assez long. Toutefois, la grande expérience de l'équipe COLOS Lyon dans ce domaine nous a aidé de façon significative tout au long du développement du didacticiel.

Nous regrettons de ne pas avoir eu matériellement le temps de procéder à une évaluation d'utilisation de notre didacticiel grâce à un test sur la population cible. Une telle évaluation aurait très certainement pu enrichir notre expérience dans le domaine de l'E.A.O. C'est lorsque le didacticiel sera véritablement utilisé à l'Institut d'Informatique qu'une telle évaluation pourra se dérouler.

Nous regrettons enfin de n'avoir pas eu le temps de mener à bien l'étape de diffusion du didacticiel au sein de l'Institut d'Informatique. Cette étape nous aurait permis de déterminer les éventuels problèmes de portabilité de l'application.

3. Perspectives

Il nous semble également important de fournir à nos successeurs quelques pistes de continuation de notre projet; c'est ce que nous appelons perspectives. L'ensemble des perspectives qui nous viennent à l'esprit peut se diviser en trois catégories : les perspectives au niveau d'Internet, les perspectives au dessus d'Internet et les perspectives en dessous d'Internet.

3.1. Les perspectives au niveau d'Internet

Ce que nous appelons les perspectives au niveau d'Internet, ce sont le éventualités de continuation de développement du didacticiel pour de nouvelles notions propres à Internet. Le didacticiel, dans son état actuel, ne présente que les notions de base d'Internet (cfr. chapitre 5). Il est dès lors possible de pousser plus loin le développement de ce didacticiel pour qu'il englobe de nouvelles notions non encore abordées.

Nous pensons entre autres à la notion de fragmentation des datagrammes. Il arrive, lors du transit d'un datagramme dans une interconnexion, que la taille de ce datagramme soit supérieure à la taille maximale autorisée pour une trame sur un sous-réseau que le datagramme doit franchir. La solution à ce problème est que le datagramme est fragmenté en parties dont la taille est inférieure ou égale à la taille maximale autorisée sur le sous-réseau : c'est la fragmentation. Cette notion est couramment utilisée dans le monde Internet, c'est pourquoi il nous semble intéressant de prévoir son implémentation dans le didacticiel au cas où le développement de celui-ci serait poursuivi.

D'autres notions intéressantes sont les protocoles EGP et IGP qui réglementent le transit des datagrammes d'information entre les passerelles pour la mise à jour et le maintien des tables de routage, la notion de routage par défaut ou encore le protocole ICMP qui gère toutes les erreurs ou les pertes survenues lors du fonctionnement d'Internet.

Un dernier type d'amélioration auquel nous pensons est l'implémentation d'une représentation tridimensionnelle pour les différentes vues existant déjà.

3.2. Les perspectives au dessus d'Internet

Les perspectives au dessus d'Internet sont les éventualités de continuation de développement du didacticiel afin qu'il englobe des notions relatives à certains protocoles s'appuyant sur le protocole IP.

Dans ce cas, le didacticiel servirait de base pour l'implémentation de nouveaux services utilisant Internet. Ici aussi, le nombre de notions à couvrir est volumineux. Ces notions vont du protocole de transport TCP au protocole HTML bien connu des utilisateurs de fichiers HTTP, en passant par des services tels que le courrier électronique, le « remote login », le FTP et encore bien d'autres.

3.3. Les perspectives en dessous d'Internet

Par opposition aux perspectives au dessus d'Internet, les perspectives en dessous d'Internet sont les éventualités de continuation de développement du didacticiel pour que celui-ci reprenne des notions relatives aux protocoles des sous-réseaux sur lesquels Internet s'appuie.

Le didacticiel dans ce cas présenterait des extensions qui reprendraient la présentation, sous forme de simulation ou non, des protocoles des sous-réseaux constituant l'interconnexion représentée. L'ensemble des protocoles auxquels on peut s'intéresser est conséquent : il reprend en fait tous les types de réseaux LAN, MAN et WAN. Quelques exemples de ces réseaux sont les réseaux Ethernet, les réseaux DQDB, les réseaux X25, les réseaux ATM et encore bien d'autres.

Nous avons essayé, dans la mesure de nos moyens, de créer une architecture logicielle structurée et bien définie afin de permettre à nos éventuels successeurs de pouvoir se baser facilement sur le produit actuel pour leurs prochains développements. C'est dans cette perspective que nous avons choisi une découpe en modules logiques (cfr. chapitre 5, point 4.3.) pour cette architecture plutôt qu'une découpe en modules physiques.

Les perspectives de continuation de ce projet ne manquent donc pas. Nous terminerons ce chapitre en souhaitant « Bon Courage » à nos successeurs éventuels.

Conclusion

Le développement d'un didacticiel constitue une tâche particulièrement complexe car elle nécessite la connaissance de plusieurs notions telles que des notions pédagogiques, ergonomiques et informatiques sans oublier la connaissance des notions relatives à la matière qui est abordée dans le didacticiel. Nous espérons que ce mémoire aborde de façon suffisamment précise ces diverses dimensions afin de fournir une base solide à nos éventuels successeurs. Nous avons pour cela préféré scinder les dimensions théorique et pratique de la phase de développement d'un didacticiel; cette division semble assez intéressante dans une première approche mais son efficacité ne pourra être jugée que lorsque nos éventuels successeurs en tireront profit.

Nous avons essayé, dans la mesure de nos possibilités, d'appliquer au maximum les notions abordées dans la partie théorique pour la mise en pratique du développement de notre didacticiel. Nous avons essayé pour cela d'être méthodique et de ne rien laisser au hasard. Cette tâche n'étant pas aisée, nous espérons que la méthode utilisée est acceptable et qu'elle pourra être utilisée à l'avenir.

Le travail au sein d'une équipe COLOS nous a permis d'utiliser au maximum les divers outils et techniques utilisés par ses membres. Nous espérons que notre mémoire est suffisamment détaillé pour leur servir à son tour dans leur travail.

Nous n'avons malheureusement pas eu le temps matériel de réaliser toutes les étapes de la phase de développement de notre didacticiel. Les prochaines étapes à accomplir sont d'effectuer des tests auprès de la population cible afin d'évaluer le didacticiel d'un point de vue utilisation et de procéder à la diffusion du produit au sein de l'Institut d'Informatique, ce qui permettra de déceler les éventuels problèmes de portabilité de l'application.

Bibliographie

[BESN88] BESNAINOU R., MULLER C., THOUIN T. (1988), *Concevoir et Utiliser un Didacticiel*, Les Editions d'Organisation, Paris (France).

[BODART92] BODART F. (1992), *Interface Homme/Machine. Cours de seconde licence et maîtrise en Informatique*, Institut d'Informatique, Facultés Universitaires Notre-Dame de la Paix, Namur.

[COMER90] COMER D. E. (1990), *Internetworking With TCP/IP. Vol 1 : Principles, Protocols, and Architecture*, Second Edition, Prentice-Hall International, Inc., USA.

[DUBOIS93] DUBOIS E. (1993), *Méthodologie de Développement de Logiciel (MDL). Cours de seconde licence et maîtrise en Informatique*, Institut d'Informatique, Facultés Universitaires Notre-Dame de la Paix, Namur.

[GARCIA94] GARCIA-SUAREZ P., RIMLINGER A. (1994), *Réseau local d'entreprise*, EFREI I1, Paris (France).

[HPCOMP89] HP Company (1989), *HP OSF/Motif Programmer's Reference. HP 9000 Series 300/800 Computers*, Hewlett-Packard Company, USA.

[INTEL94] INTEL Corporation (1994), *Le guide Intel pour les réseaux*, INTEL Corporation, USA.

[ISIUSC81] INFORMATION SCIENCES INSTITUTE, UNIVERSITY OF SOUTHERN CALIFORNIA (1981), *Internet Protocol. DARPA Internet Program Protocol Specification*, RFC 791, USA.

[KASTEN94] KASTENHOLZ F., PARTRIDGE C. (1994), *Technical Criteria for Choosing IP The Next Generation (Ipng)*, RFC 1726, USA.

[MANKIN95] MANKIN A., BRADNER S. (1995), *The Recommendation for the IP Next Generation Protocol*, RFC 1752, USA.

[MEDA90] MEDA J. (1990), *Evaluer les Logiciels de Formation*, Les Editions d'Organisation, Paris (France).

[MENU94] COLOS MEMBERS (1994), *xmMenuGen. Read me File*, CEGELY, Ecole Centrale de Lyon, Lyon (France).

[MEYERS92] MEYERS S. (1992), *Effective C++. 50 Specific Ways to Improve Your Programs and Designs*, Addison-Wesley Publishing Company, USA.

[MULLER92] MULLER D. (1992), *makePixAppl. Man pages*, CEGELY, Ecole Centrale de Lyon, Lyon (France).

[MULLER93] MULLER D. (1993), *Interface Homme-Machine. Réalisation d'applications avec Xt et Motif*, CEGELY, Ecole Centrale de Lyon, Lyon (France).

[MULLER94] MULLER D. (1994), *The Standard COLOS platform*, 3rd edition, CEGELY, Ecole Centrale de Lyon, Lyon (France).

[MULLER94a] MULLER D. (1994), *UI2C : a User Interface to C language translator*, CEGELY, Ecole Centrale de Lyon, Lyon (France).

[MULLER95] MULLER D. (1995), *The COLOS Project*,
URL : http://www.colos.ec-lyon.fr/colosHp/, CEGELY, Ecole Centrale de Lyon, Lyon (France).

[OREILLY89] NYE A. (1989), *The Definitive Guides to the X Window System. Volume two : Xlib Reference Manual for Version 11*, 3rd edition, O'Reilly & Associates, Inc., USA.

[OREILLY90a] NYE A. (1990), *The Definitive Guides to the X Window System. Volume one : Xlib Programming Manual for Version 11*, 2nd edition, O'Reilly & Associates, Inc., USA.

[OREILLY90b] NYE A., O'REILLY T. (1990), *The Definitive Guides to the X Window System. Volume four : X Toolkit Intrinsics Programming Manual. OSF/Motif Edition for X11R4*, Motif edition, O'Reilly & Associates, Inc., USA.

[OREILLY91] O'REILLY AND ASSOCIATES, INC. STAFF (1990), *The Definitive Guides to the X Window System. Volume five : X Toolkit Intrinsics Reference Manual for X11 Release 4*, 2nd edition, O'Reilly & Associates, Inc., USA.

[STROUST93] STROUSTRUP B. (1993), *The C++ Programming Language*, 2nd edition, Addison-Wesley Publishing Company, USA.

[VANBAST93] 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.

[VANDERD94] VANDERDONCKT J. (1994), *Guide ergonomique des interfaces homme-machine*, Les Presses Universitaires de Namur, Namur.

[YOUNG90a] YOUNG D. A. (1990), *The X window system. Programming and Applications with Xt. OSF/Motif edition*, Prentice-Hall, Inc., USA.

[YOUNG90b] YOUNG D. A. (1990), *OSF/MOTIF Reference Guide*, Prentice-Hall, Inc., USA.

[YOUNG95] YOUNG C. Y. (1995), *Quick Reference to Common Xweb Operations*, 2nd revision, CEGELY, Ecole Centrale de Lyon, Lyon (France).