ENVIRONNEMENTS LOGICIELS POUR UNE INTEGRATION QUOTIDIENNE DE L'E.A.O. DANS L'ENSEIGNEMENT

V. Guéraud, J.-P. Peyrin, J.-P. David, J.-P. Pernin

Laboratoire de Génie Informatique
IMAG-Campus BP 53X
38041 Grenoble Cedex
France

Résumé : *Dans un premier temps, nous présentons le laboratoire Arcade, comme un exemple de produit hypermédia pour l'enseignement de l'algorithmique. Ce faisant, nous défendons un certain style d'EAO, offrant des ressources complémentaires intégrées dans les pratiques pédagogiques quotidiennes des enseignants. Pour permettre une telle intégration, il paraît nécessaire de donner aux enseignants les moyens d'adapter ces ressources à leurs cours, de les enrichir, et même d'en créer de nouvelles. Dans cet objectif, il nous faut améliorer la méthode et les outils de production, jusqu'ici assez lourds à mettre en œuvre.*

Nous analysons sous cet angle deux types d'environnements de production, en nous appuyant sur diverses expériences. Le premier (Smalltalk) minimise le travail de production en permettant une réutilisation maximum de composants existants (grâce à l'approche par objets qu'il propose), mais il impose un certain type d'interface et nécessite une solide compétence informatique. Le deuxième, constitué de logiciels hypermédias (HyperCard, ToolBook), permet à des auteurs ayant un minimum de connaissances informatiques de produire de petites applications ; nous proposons des éléments de méthodes autorisant une utilisation cohérente de ces logiciels.

Enfin, dans la perspective précise de produire rapidement de nouvelles activités au sein du laboratoire Arcade, ou de créer un nouveau laboratoire de même type, nous travaillons sur la définition de nouveaux environnements de production basés sur les deux approches ci-dessus. Des maquettes ont été réalisées dans ce sens.

I. UN PRODUIT HYPERMÉDIA POUR L'APPRENTISSAGE : LE LABORATOIRE ARCADE

Le laboratoire Arcade est un ensemble intégré de logiciels destiné à faciliter l'enseignement et l'apprentissage de l'algorithmique. Deux enseignants sont à l'origine de sa création. Les activités proposées dans le laboratoire ont été définies progressivement, en réponse aux problèmes concrets qui se posaient à ces enseignants dans leurs cours. Les problèmes rencontrés étaient en général liés à la présentation des phénomènes dynamiques inhérents à la programmation. Il est en

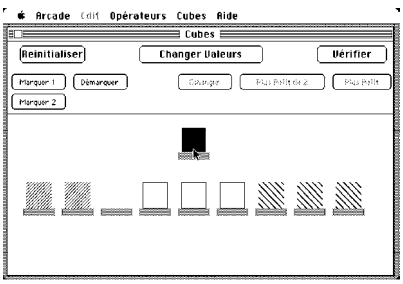
effet difficile de montrer au tableau noir les phénomènes liés à l'exécution de programmes, à l'évolution des structures de données... Mais pourquoi même tenter de le faire au tableau noir alors que l'ordinateur fournirait un support idéal pour le faire ? Le laboratoire Arcade a été progressivement construit autour de cette idée. Découvrons-le maintenant.

I.1. Description du laboratoire Arcade

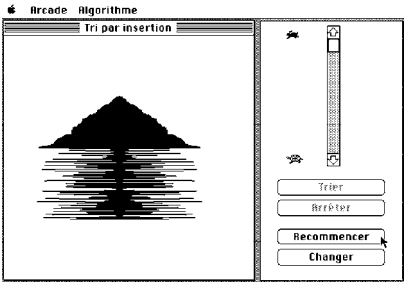
Conformément à la métaphore spatiale utilisée, le laboratoire se présente comme un ensemble de bâtiments, dans lesquels on doit se rendre pour accéder à diverses activités. Le nom de chaque bâtiment renseigne sur le sujet algorithmique qu'il traite (Tris, Récursivité, Graphes, Spécifications...). Des bâtiments spéciaux tels que "Accueil" ou "Bibliothèque" remplissent ici l'équivalent de leurs fonctions habituelles. A l'intérieur de chaque bâtiment se trouve une salle qui offre diverses activités. Les activités proposées visent une participation active de l'étudiant et reposent sur la manipulation, l'observation ou le jeu. Sans entrer dans le détail de chaque activité, illustrons la diversité de celles-ci par quelques exemples. Ceux-ci sont choisis dans le thème "tri".

Le premier exemple "Meccano de tri" met à la disposition de l'étudiant un ensemble d'outils permettant de réaliser des tris d'objets représentés à l'écran par des cubes (dont la valeur est invisible). Ces outils sont de divers niveaux : déplacer un cube vers une zone de travail, échanger deux cubes, rechercher le cube contenant la plus petite valeur, etc. L'étudiant peut choisir librement des outils parmi ceux qui lui sont proposés. Il doit ensuite travailler uniquement avec les outils choisis pour réaliser le tri de ces objets. Selon les choix faits, il devient possible ou non de faire des tris suivant tel ou tel algorithme (tri par minimums successifs, tri par bulles, tri par insertion...). Ce type de manipulation est également offert pour travailler sur d'autres thèmes algorithmiques : manipulation d'arbres binaires...

Le deuxième exemple "Tris internes" offre la visualisation de l'exécution des principaux algorithmes de tris internes (tri par minimums successifs, tri par insertion, tri par tas, tri de Shell...). Chaque tri est exécuté sur une représentation graphique : les données sont matérialisées par des traits de longueurs proportionnelles à leurs valeurs (Brown, 1985). L'étudiant choisit un algorithme de tri, définit les caractéristiques des données (aléatoires, ordonnées, inversées, égales), leur nombre, ou même définit leurs valeurs. Il choisit la vitesse d'exécution. Il peut à chaque instant recommencer le tri, choisir d'autres données ou un autre algorithme. Ce type d'observation active où l'étudiant agit sur les paramètres de visualisation, est également proposé dans d'autres activités : Graphes, Dessins récursifs... Plusieurs dessins animés sont aussi proposés.



"Meccano de tri"



"Tris internes"

Dans l'état actuel, le laboratoire Arcade concerne une vingtaine de thèmes algorithmiques et offre matière à une cinquantaine d'heures de travail étudiant. Le laboratoire, disponible sur du matériel Apple Macintosh, est constitué d'une vingtaine de programmes Pascal (environnement de production Think Pascal) et de quatre dessins animés (environnement VideoWorks). Tous ces composants sont intégrés sous le logiciel HyperCard. Le développement du laboratoire a commencé en 1985, il représente un travail d'environ 10 hommes-année et le logiciel occupe environ 5 MégaOctets (Cagnat, 1990).

I.2. Utilisations pédagogiques du laboratoire

Une des richesses de ce logiciel est d'être utilisable aussi bien par l'enseignant durant son cours que par les étudiants durant leur travail individuel. En effet, l'enseignant utilise certains composants pendant son cours, pour démontrer, expliquer ou illustrer des concepts. L'étudiant peut ensuite utiliser les mêmes composants, ou d'autres, pour revoir à son rythme ce qu'a fait l'enseignant, pour travailler sur des points difficiles, pour vérifier sa compréhension par des exercices. Dans un autre cours, l'enseignant se sert de nouveau du laboratoire pour faire une synthèse, insister sur des points de vue particuliers, ou inciter les étudiants à en découvrir un peu plus tout seuls (les logiciels contiennent des suggestions d'utilisation pour chaque activité du laboratoire)... L'enseignant peut également s'appuyer sur une pratique préalable par les étudiants de certains composants pour introduire de nouveaux concepts.

Pour illustrer ces possibilités, développons un scénario pédagogique parmi ceux que nous avons repérés (Guéraud, 1991). Ce scénario pédagogique est adapté à un public universitaire débutant en informatique.

L'enseignant peut grâce au thème "tri" :

- montrer l'existence d'une variété de solutions pour un même problème, le tri ;
- mettre en évidence les différences de performances des algorithmes solutions ;
- illustrer le fait que les performances des algorithmes sont liées aux conditions d'utilisation ;
- proposer une étude détaillée de nombreux algorithmes de tris.

Comment l'enseignant peut-il se servir du laboratoire Arcade dans ce contexte pédagogique ? L'enseignant propose tout d'abord aux étudiants de découvrir des méthodes de tris. Pour cela il leur demande d'utiliser le logiciel "Meccano de tri", avec comme seuls outils le déplacement d'un cube vers un socle de travail et la recherche du minimum de 2 valeurs. Les étudiants vont ainsi progressivement découvrir le mécanisme du tri par bulles et l'expliciter avec l'aide de l'enseignant. Si l'enseignant propose ensuite d'utiliser comme outil la recherche du minimum de n valeurs, les étudiants seront en mesure de découvrir la méthode du tri par minimum successifs.

L'enseignant montre, grâce au logiciel "Tris internes", l'exécution de ces deux algorithmes sur la représentation graphique du tableau. L'effet visuel obtenu est bien sûr différent selon l'algorithme et il renforce la compréhension de chaque algorithme. L'enseignant peut maintenant proposer d'observer l'exécution d'un autre

algorithme de tri : le tri par insertion. Les étudiants doivent cette fois comprendre le mécanisme de ce tri en observant son exécution. L'enseignant propose alors aux étudiants l'exercice consistant à vérifier leur compréhension de cet algorithme en essayant de reproduire cette exécution sur les cubes de "Meccano de tri". Si les étudiants éprouvent quelques difficultés, ils peuvent observer de nouveau l'exécution de l'algorithme dans "Tris internes" mais cette fois avec un petit nombre d'éléments et en vitesse réduite.

L'observation de l'exécution d'autres algorithmes de tris tels que le tri par tas, le tri de Shell,... "plus rapides" va orienter la discussion vers une comparaison des performances des différents algorithmes. Un exemple spectaculaire, l'exécution du tri par tas sur des données déjà ordonnées, qui se traduit par une destruction complète de l'ordre des données avant le rétablissement de cet ordre, montre qu'un algorithme n'est en général pas plus performant qu'un autre dans l'absolu, mais que pour chacun il existe des conditions optimales d'utilisation (caractéristiques de données,...).

Des expérimentations de divers scénarios pédagogiques ont été réalisées à un niveau universitaire (en particulier Université Joseph Fourier - Grenoble I et Université Pierre Mendès France - Grenoble II), dans des formations d'enseignants du secondaire, ainsi qu'en lycée pour l'enseignement optionnel d'informatique.

I.3. Pour une véritable intégration de l'EAO

A travers le logiciel Arcade, c'est une certaine vision de l'utilisation de l'ordinateur dans l'enseignement que nous illustrons. Indépendamment de la discipline enseignée, nous souhaitons que soient proposées sur ordinateur des activités complémentaires à un enseignement de base. Ces activités doivent viser à combler les insuffisances des formes traditionnelles d'enseignement (il est inutile dans ce contexte, de faire sur ordinateur des choses que l'enseignant sait très bien faire par ailleurs, comme il est inutile de reproduire un livre à l'écran).

De plus, nous souhaitons que l'EAO permette une exploration et un apprentissage actifs en proposant aux étudiants une multitude de parcours possibles, sans vouloir contrôler leur progression.

Enfin, nous pensons que les mêmes activités sur ordinateur doivent pouvoir servir aux enseignants pendant leurs cours, comme aux étudiants en dehors de la classe, afin de permettre une utilisation pédagogique de l'EAO parfaitement intégrée à une formation.

Pour viser une telle intégration de l'EAO dans les pratiques pédagogiques quotidiennes des enseignants, il est nécessaire de permettre aux enseignants d'adapter des applications existantes à leurs cours, de leur permettre de les enrichir, de les modifier, voire d'en créer de nouvelles. Bien sûr, il ne s'agira pas pour eux de créer des applications importantes couvrant un grand domaine de connaissances, mais de fabriquer de petites applications illustrant des points très précis pour remplacer ce qu'ils savent mal faire au tableau noir. Le laboratoire Arcade, n'est finalement que la réalisation de nombreuses petites idées...

Dans cet objectif, il nous faut réfléchir aux moyens de production de telles applications. Une production "lourde" nécessitant beaucoup de temps et d'effort ne

peut être prise en charge que par des équipes d'informaticiens. Elle ne répond pas au besoin que nous avons défini, besoin pour l'enseignant de produire rapidement une petite application pour son prochain cours.

II. ETUDE D'ENVIRONNEMENTS DE PRODUCTION

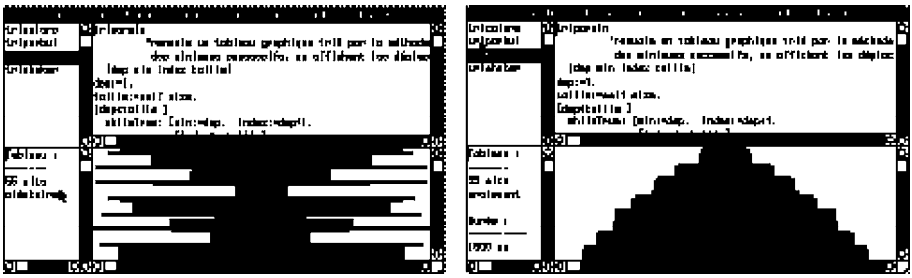
Nous nous intéressons donc à des environnements qui permettraient une production rapide d'applications. Nous analysons ici deux types d'environnements qui nous ont paru pertinents.

II.1. Un environnement de conception par objets

Dans un premier temps, afin de minimiser le travail de production, nous nous sommes intéressés à des environnements de développement qui permettent la réutilisation de composants existants. Nous avons choisi Smalltalk, un environnement de programmation objet complet disponible sur Macintosh (Clavel, 1991).

Une première expérience a consisté à reproduire l'activité "Tris internes" du laboratoire Arcade qui avait été initialement réalisée dans l'environnement Think Pascal. Le prototypage d'une telle application est assez rapide en Smalltalk car elle s'appuie sur des classes existantes de la façon suivante. Nous définissons une classe "tableaux-graphiques", qui hérite des propriétés et des comportements de la classe "Array" prédéfinie dans Smalltalk. Dans cette nouvelle classe nous ajoutons des caractéristiques (l'apparence graphique d'un tableau) et des "méthodes", qui décrivent les comportements spécifiques de ces nouveaux objets (c'est ici que nous pouvons définir les méthodes de tris). Nous redéfinissons également des méthodes de la classe "Array" (par surcharge) : à la méthode d'affectation d'une valeur à une case de tableau, nous rajoutons le déplacement graphique du trait correspondant. Ainsi, lors de l'exécution d'un algorithme, l'objet tableau-graphique recevant le message d'affectation d'une valeur à une case, réagit en déplaçant graphiquement le trait correspondant. Notons que nous n'avons ici qu'à travailler sur la classe "tableaux-graphiques", sans toucher aux classes utilisées (notion d'encapsulation des classes). D'autre part, l'environnement utilisateur de cette application a pu être créé rapidement en réutilisant des classes avec lesquelles est construite l'interface utilisateur de Smalltalk lui-même.

L'environnement de développement utilisé a donc bien répondu à notre attente. Il nous a également permis de rajouter une fonctionnalité à l'activité : nous pouvons maintenant proposer à l'étudiant d'observer l'exécution de ses propres algorithmes de tri (écrits dans le langage Smalltalk) sur la représentation graphique des données. Les algorithmes fournis par l'utilisateur sont vus comme des méthodes de la classe "Tableaux-graphiques" ; leur interprétation fournit la visualisation de leur exécution. Cette propriété fait de Smalltalk un excellent outil de prototypage, car elle ajoute à la puissance du modèle objet la souplesse de la création dynamique. Ceci aurait nécessité dans un langage de type Pascal la réécriture d'un interpréteur, c'est à dire la définition d'une nouvelle couche de langage qui interprète le langage algorithmique écrit par l'utilisateur.

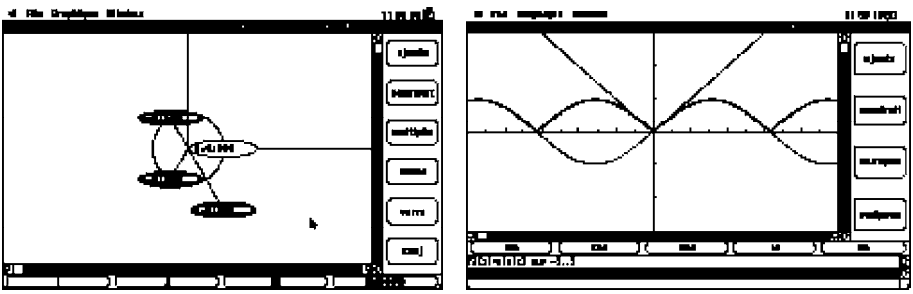


Un éditeur d'algorithmes de tris de tableaux
avant et après l'exécution de la méthode triparmin sur un tableau de 55 éléments)

Nous avons également réalisé dans cet environnement plusieurs activités de même nature mais destinées à un autre domaine, celui des mathématiques. Chacune de ces activités propose de visualiser l'effet d'actions sur des objets mathématiques abstraits à l'aide d'une représentation graphique.

La première de ces activités simule ce que serait une "calculette pour nombres complexes", utilisant une représentation graphique. Pour la réaliser, nous avons créé 2 classes. La classe "Complexe" comporte les attributs et les méthodes caractérisant les nombres complexes. La classe "LaboComplexes" modélise l'activité offerte à l'utilisateur sur ces nombres complexes : les actions possibles et la représentation graphique du résultat de ces actions. Nous avons réalisé ensuite très rapidement un jeu de "nombre complexe caché", que le joueur doit atteindre à l'aide des opérateurs de la calculette. Pour cela il suffit de spécialiser l'interface du LaboComplexes en créant une sous-classe "JeuComplexes", contenant quelques méthodes supplémentaires.

Nous avons pu appliquer la même méthode de conception pour une "calculette pour les fonctions numériques" fournissant une représentation graphique des résultats. La puissance d'expression du langage a permis d'implémenter le calcul formel sur les fonctions. Un enseignant peut alors facilement créer sur cette base une nouvelle application en ajoutant une sous-classe à celles existantes, par exemple la classe "JeuFonction", dans laquelle il définira les opérations disponibles à l'utilisateur et le but à atteindre.



Calculette pour nombres complexes Calculette pour fonctions numériques

Les facilités de réutilisation qui ont été appréciées pour la création de l'interface utilisateur ont pour contrepartie l'acceptation du mode de fenêtrage proposé

dans l'environnement Smalltalk. La personnalisation des écrans dans le style Arcade demanderait la spécialisation des classes existantes, ce qui alourdirait le travail de prototypage. Mais la réutilisation de ces nouvelles classes résoudrait définitivement ce problème pour les développements futurs. D'autre part, l'utilisation de ce type d'environnement nécessite une connaissance informatique solide et n'est pas à la portée de tout enseignant. Nous nous sommes également intéressés à un autre type d'environnement de développement : celui des logiciels hypermédias.

II.2. Des logiciels hypermédias comme outils de production

Les logiciels hypermédias tels que HyperCard (HyperCard, 1987) dans le monde Macintosh, ToolBook (ToolBook, 1991) sous Windows dans le monde PC... proposent à l'utilisateur de multiples exemples, d'aspect attrayant et de nature conviviale. Ils vont plus loin en permettant à l'utilisateur de modifier ces exemples, ou d'en créer lui-même de nouveaux répondant à ses propres besoins.

Alors que sans ce type d'outils, il ne serait pas venu à l'idée d'un utilisateur X de créer lui-même une application, le voici transformé progressivement en auteur. Le passage de la fonction d'utilisateur à la fonction d'auteur est relativement naturel. En effet, dans un premier temps, l'auteur crée interactivement les écrans qu'il destine à l'utilisateur en manipulant directement les objets qui les composent ; il décrit interactivement les comportements souhaités de ces objets (passage à un autre écran, animations visuelle et sonore...), le tout sans nécessairement écrire de "code" informatique. S'il veut aller plus loin, il dispose d'un langage lui permettant de spécifier plus librement, plus complètement le comportement des objets.

Ce type de logiciel donne à des enseignants l'envie de créer eux-mêmes de petites applications répondant à leurs besoins ponctuels, et leur offre la possibilité de le faire, rapidement et sans trop d'efforts. Nous pensons qu'il faut définir plus précisément les connaissances minimales en informatique à acquérir pour cela, et proposer en I.U.F.M. sur cette base, une formation aux futurs enseignants.

Développons ces aspects en nous appuyant sur des exemples de petits logiciels EAO réalisés à l'aide de logiciels hypermédias.

II.2.1. Intérêt des logiciels hypermédias

A titre d'exemple, situons-nous à l'école primaire. L'apprentissage de l'addition au cours préparatoire nécessite une pratique par les enfants d'un grand nombre d'additions. Très souvent, l'instituteur propose comme devoir aux enfants de compléter une page d'additions qu'il a au préalable écrite à la main et tirée en 30 exemplaires à l'aide de la machine à alcool. Or il serait possible à l'instituteur, avec une connaissance minimale de concepts informatiques, de réaliser par exemple une pile HyperCard, proposant ce même exercice.

Pour cela, il est suffisant de savoir appeler une fonction générant des nombres aléatoires, vérifier un résultat, et proposer un décor (créé sous logiciel graphique simple ou plus simplement encore copié). Il ne lui faudrait pas plus de temps pour créer cette application que pour créer la page décrite plus haut. Or cette application serait réutilisable à volonté, fournirait un nombre illimité d'additions et permettrait à moindre effort de nombreuses variantes (trouver l'un des 2 opérandes connaissant l'autre et le résultat de l'addition...). De plus, l'exercice motiverait alors

davantage les enfants sous cette forme plus attrayante. Bien sûr, cela suppose que des ordinateurs soient disponibles à l'école...

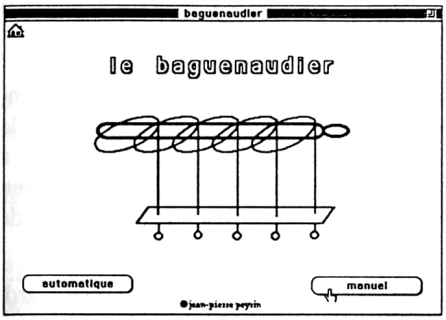
D'autres exemples de même nature peuvent être cités dans le domaine du calcul (apprentissage des tables de multiplication) comme dans le domaine du français (connaissance du vocabulaire par la pratique du jeu du pendu). Ces exemples se prêtent bien à la réalisation de petites applications, réalisations qui ne requièrent que peu de connaissances informatiques et qui peuvent être rapidement menées à bien grâce à un logiciel hypermédia.

A un tout autre niveau, intéressons-nous maintenant à des formations universitaires et plus précisément à l'enseignement de l'algorithmique à l'Université. Le baguenaudier est un casse-tête classique reposant sur un problème algorithmique. Ce casse-tête est formé de deux parties : d'une part, des anneaux imbriqués et reliés à une réglette par des tiges coulissantes et d'autre part une barre mobile appelée navette. Au début du jeu, la navette est enchevêtrée dans les anneaux. Le jeu consiste à séparer les deux parties.

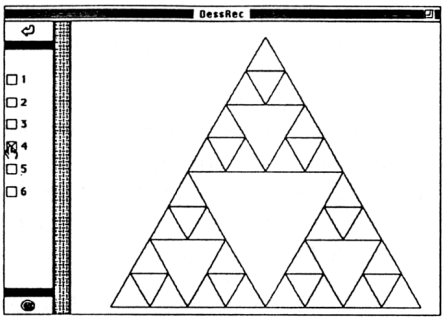
Le travail de modélisation du jeu et la recherche des solutions sont d'autant plus intéressants que les étudiants peuvent manipuler le jeu. Dans ce but, une simulation proche de la manipulation réelle a été réalisée en HyperCard en moins de 2 heures de travail et est maintenant à la disposition des étudiants. Remarquons que les enseignants d'algorithmique auraient pu réaliser une simulation du jeu en utilisant un quelconque langage de programmation, mais pour le faire aussi rapidement, ils auraient choisi un mode de représentation et de manipulation plus éloigné de la réalité et donc moins motivant pour les étudiants.

Toujours dans le domaine de l'algorithmique, considérons maintenant l'apprentissage de la récursivité. L'exemple des dessins récursifs permet une étude de ce thème, en offrant l'avantage de visualiser les problèmes de programmation rencontrés par l'étudiant : les erreurs de conception d'un algorithme récursif se traduisent ici concrètement par l'écart entre le dessin réalisé à l'écran et le dessin à obtenir. Un logiciel a été réalisé sur cette base pour Arcade en Think Pascal. Il permet d'observer les tracés de certains dessins récursifs, d'analyser la construction de ces dessins, et de vérifier sa compréhension. Notons que le temps initial de réalisation n'était pas négligeable et que l'ajout d'un nouveau dessin n'est pas immédiat.

Un autre logiciel a été réalisé sur ce même thème en utilisant HyperCard. Sa réalisation a été assez rapide, la seule difficulté étant liée à la connaissance des algorithmes de dessins récursifs, connaissance inhérente à l'enseignement visé. L'avantage de ce logiciel est que l'ajout d'un nouveau dessin récursif est maintenant facile : ajout d'un bouton permettant le choix du nouveau dessin, ajout d'une carte liée au tracé du dessin et frappe de l'algorithme du nouveau dessin. Notons toutefois que l'exécution de ces tracés est plus lente dans ce logiciel que dans celui réalisé en Think Pascal : ceci est techniquement dû au fait que les algorithmes sont interprétés dans HyperCard alors qu'ils sont compilés en Pascal.



"Le Baguenaudier"



"Dessins récursifs"

Ces divers exemples illustrent l'intérêt que présentent des logiciels hypermédias pour produire rapidement de petites applications EAO. Il est à noter que les applications ainsi réalisées sont en général plaisantes. Ceci est sans doute dû aux possibilités graphiques, sonores... offertes par les logiciels hypermédias, mais aussi au fait que l'auteur, de par les outils de création d'interface qui lui sont offerts, est sans cesse confronté au point de vue utilisateur.

La production dans un tel environnement d'applications importantes en taille est plus difficile. L'auteur, s'il ne se crée pas une organisation de travail rigoureuse, risque d'avoir de la difficulté à se repérer dans les nombreux écrans créés, à maintenir la cohérence des enchaînements... De même, si l'application à créer nécessite beaucoup de programmation, l'auteur devra suppléer aux manques du langage associé. Analysons de plus près ces problèmes.

II.2.2. Quelques critiques

Les métaphores de la "pile" ou du "livre" proposées aux utilisateurs respectifs d'HyperCard et de ToolBook sont contestables. Elles renforcent en effet l'idée d'un ordre séquentiel de lecture alors que les logiciels hypermédias offrent au contraire à l'utilisateur la possibilité de créer son propre parcours de l'information. Elles insistent en fait sur une organisation physique des cartes ou pages, ce qui n'est significatif ni pour l'utilisateur ni pour l'auteur.

En ce qui concerne la gestion des liens entre les cartes (ou pages), l'auteur peut référencer une carte par :

- son emplacement physique (carte suivante, carte précédente, première carte...) ; Or l'emplacement physique d'une carte est en général peu significatif et il peut être modifié en particulier par l'insertion d'autres cartes. Par prudence, l'auteur ne devrait procéder de cette façon que pour gérer les liens d'un sous-ensemble de cartes dont la lecture est effectivement fortement linéaire ;
- son identificateur numérique (carte d'ID 4056...). Cet identificateur, unique, est totalement géré par le logiciel, mais n'a aucun sens pour l'auteur. Toute modification devient alors pénible ;
- son nom logique. L'auteur a effectivement la possibilité de nommer chaque objet et de faire référence à l'objet par son nom. Un auteur averti en fera un usage systématique afin de pouvoir relire facilement les scripts et les modi-

fier. L'inconvénient est qu'il n'y a aucune vérification par le logiciel de l'unicité d'un nom logique ; en cas de noms définis plusieurs fois, c'est le premier objet rencontré portant ce nom qui est effectivement désigné!

De plus, il est à noter qu'aucune vérification de l'existence des objets référencés ne peut être demandée au logiciel. Rien n'indique par exemple, lors de la suppression d'une carte, que des scripts faisaient référence à cette carte. C'est à l'auteur de rester vigilant pour maintenir la cohérence des liens. Il n'a comme moyen de vérification de cette cohérence qu'une exploration complète des parcours de lecture offerts à l'utilisateur.

Le langage associé à ces logiciels (HyperTalk pour le logiciel HyperCard, OpenScript pour ToolBook) présente lui aussi des limites, limites de puissance d'expression et limites de représentation :

- Paramétrage uniquement par valeurs ;
- Variables locales ou globales : Dans HyperTalk, une variable est soit locale à une procédure soit globale à toutes les piles actives. Il serait intéressant de pouvoir définir des variables locales à tout le script d'un objet, caractérisant l'état de l'objet. Ceci est mieux géré dans ToolBook ;
- Absence de toute structuration aisée de l'information. Les structures de tableau, de listes chaînées, d'enregistrement ne sont pas explicitement disponibles pour l'auteur. Il peut plus ou moins facilement et efficacement les simuler en utilisant des objets invisibles mais il semble que cela ne soit pas très naturel pour l'utilisateur ;
- Absence de la notion de classe et sous-classe d'objets. On aimerait par exemple pouvoir définir une classe de boutons ayant une apparence et un rôle similaires, et créer des objets instances de cette classe. Or la façon de réaliser cet objectif en HyperCard par exemple sera de créer un représentant de cette classe logique puis de le dupliquer pour créer à moindres frais d'autres représentants de cette même classe. Si cela paraît acceptable pour la création d'objets similaires, cela ne l'est plus lorsqu'on pense à leur modification. En effet, pour modifier une caractéristique (propriété ou comportement) de la classe logique, il faudra la modifier sur toutes les instances déjà créées. En ce qui concerne le comportement, il est toutefois possible de simuler la notion de méthodes de classes en appelant une même routine depuis toutes les "instances". Cette routine est définie à un seul endroit (accessible depuis chaque instance) et donc modifiable aisément à cet endroit.

Sans aller jusqu'à la définition d'un véritable langage à objets (inutile à ce niveau d'introduire explicitement les notions de types abstraits, d'héritage, de polymorphisme et généricité), nous souhaiterions disposer d'un système plus cohérent, qui intègre les concepts tout en les rendant transparents, naturels à l'auteur. Plusieurs solutions peuvent être envisagées pour remédier aux défauts évoqués ci-dessus. La première consiste à fournir aux auteurs des éléments de méthode de production qui leur éviteraient quelques problèmes. Ces éléments de méthode pourraient par la suite être intégrés dans le logiciel. Un deuxième type de solution est abordé au paragraphe suivant.

III. VERS LA DÉFINITION DE NOUVEAUX ENVIRONNEMENTS DE PRODUCTION

Nous travaillons à présent sur la définition de nouveaux environnements de production, basés sur les environnements analysés ci-dessus, mais qui n'en présenteraient pas les inconvénients.

III.1. Vers la spécification d'une surcouche hypermédia pour Smalltalk

Considérons l'environnement Smalltalk : la création de nouvelles activités EAO passe actuellement par l'environnement Smalltalk intégral et nécessite une bonne connaissance informatique. Notre projet est donc de concevoir un environnement hypermédia pour l'auteur qui lui masque les aspects purement langages de Smalltalk, tout en lui donnant la possibilité d'y accéder si besoin est. De plus, nous souhaitons permettre à l'auteur une production facile d'interfaces, tout en l'autorisant à s'écarter du modèle d'interfaces Smalltalk.

Dans tous les exemples que nous avons réalisés, le principe est toujours calqué sur le mécanisme "Model View Controller" : créer une interface de visualisation d'objets abstraits, objets abstraits dont les caractéristiques et les comportements sont décrits dans une classe Smalltalk. Pour chaque application une classe décrit l'interface en regroupant la vue (présentation graphique) et le contrôle sur cette vue (boutons). C'est la gestion de l'activité hypermédia. L'enseignant voulant créer une activité a donc actuellement deux types de choses à réaliser : la description des objets abstraits qu'il manipule, et la description de l'interface qu'il propose.

En ce qui concerne la description des objets abstraits sur lesquels faire travailler des apprenants, l'enseignant dispose de certaines classes Smalltalk (les nombres entiers, les fractions, les nombres décimaux). D'autres classes que nous avons créées sont disponibles dans le même but (les nombres complexes, les fonctions numériques, les lignes brisées). Pour cette création, nous avons utilisé le point fort d'un environnement objet, c'est-à-dire la grande facilité de réutilisation des classes existantes (par héritage ou composition) ainsi que la puissance d'expression et la cohérence du langage pour lequel tout est objet. Ainsi, il sera possible de fournir aux auteurs des bibliothèques de classes (bibliothèques spécialisées par discipline enseignée et niveau visé par exemple) décrivant certains concepts abstraits qu'ils enseignent.

L'auteur ayant choisi dans la liste que nous lui proposons la classe sur laquelle il veut travailler, il lui reste à créer les interfaces de manipulation. Actuellement, nous avons créé la super-classe "Laboratoire", contenant les caractéristiques et les méthodes pour la présentation et le contrôle par boutons, dont pourront hériter toutes les sous-classes laboratoire rajoutées par la suite. Mais pour cela, l'auteur doit encore écrire quelques lignes de code. C'est la différence avec les environnements auteurs hypermédias tels que HyperCard ou ToolBook : les interfaces de visualisation ne se créent pas interactivement par des désignations graphiques. Nous voudrions, permettre à l'auteur de créer lui même les interfaces pour ses utilisateurs, sans qu'il soit obligé de recourir à la programmation. Pour cela nous voulons réaliser en Smalltalk un environnement interactif de spécification d'interfaces de type Laboratoire.

III.2. Surcouche pour logiciel hypermédia

Nous souhaitons pallier les inconvénients que présentent les logiciels hypermédias comme outils de production. Afin de concrétiser ce souhait, considérons la production d'une nouvelle activité au sein du laboratoire Arcade. Une telle production peut être décomposée en deux types de travaux différents :

- la création de l'activité proprement dite ;
- l'intégration de cette activité au sein du laboratoire.

Nous souhaitons donc proposer un environnement permettant de réaliser ces deux types de travaux, que ce soit pour étendre un laboratoire existant ou pour créer un nouveau laboratoire de toutes pièces. Nous résumons ici quelle a été la démarche utilisée pour la définition de cet environnement et les difficultés rencontrées lors de chaque étape.

La première étape a consisté à modéliser la notion de laboratoire sous-jacente au logiciel Arcade. En nous inspirant largement de la Cité des Sciences et de l'Industrie de la Villette, nous avons défini en terme d'objets ce que pouvait être un Centre d'Exposition à Caractère Scientifique. C'est ainsi que nous avons défini les notions de centre, de bâtiment, de salles et d'activités, ainsi que celles de responsable de centre, de concepteur d'activité et de visiteur, modélisant les différentes personnes concernées par un tel centre.

Une fois cette analyse effectuée, nous avons défini les fonctionnalités que doit rendre aux différentes catégories d'utilisateurs un logiciel d'EAO basé sur cette métaphore :

- le responsable de centre (c'est à dire le responsable du logiciel) doit pouvoir créer un centre d'exposition, y ajouter ou supprimer des bâtiments, organiser ses bâtiments en salles et affecter à chaque salle des activités. Sa tâche se rapproche de celle d'un architecte ;
- le concepteur d'activités (c'est à dire l'enseignant ou l'expert d'un domaine) doit pouvoir rapidement développer une nouvelle activité servant de support à son enseignement ;
- le visiteur (c'est à dire l'élève) doit pouvoir se mouvoir facilement dans le centre, entrer dans un bâtiment, dans une salle, s'intéresser à une activité, etc.

Naturellement, un enseignant pourra tour à tour remplir les rôles de concepteur d'activités pour créer une nouvelle activité, et de responsable de centre pour intégrer cette activité à un centre existant.

Nous avons résolument choisi de nous orienter vers une interface graphique permettant la manipulation directe des objets définis lors de l'analyse. Une possibilité pour implémenter cette solution consiste à utiliser un langage objet classique (Smalltalk, Eiffel, C++). Nous décrivons ici une autre implémentation qui consiste à nous appuyer sur un logiciel hypermédia existant (HyperCard, ToolBook). Si cette implémentation exige la réécriture d'une surcouche à objets (surcouche définissant les notions de classes, de méthodes et d'attributs pour les objets "centre", "bâtiment", etc.), elle permet d'exploiter toute la richesse du système multimédia (graphisme, son, simplicité d'utilisation). Une maquette a été réalisée à l'aide de ToolBook système plus complet et plus cohérent que HyperCard.

La première réalisation a concerné le développement de la partie du logiciel permettant au responsable de centre de gérer un centre, des bâtiments, des salles et des activités par manipulation directe, l'accès aux objets de base du système hypermédia lui étant masqué (le logiciel générant automatiquement les pages, boutons, objets graphiques...).

Le second volet de la maquette concernant la définition d'une activité par un "responsable d'activité" est beaucoup plus délicat. En effet, l'idée est ici de proposer à l'auteur (donc a priori à un enseignant non-spécialiste en informatique) une démarche lui permettant de définir son activité. La méthode proposée est la suivante : conduire l'auteur à séparer clairement l'abstraction (qu'est ce que je veux montrer ?) et la présentation qu'il va faire de cette abstraction (comment je vais le montrer ?). L'interface définie dans la maquette interdit le mélange de ces deux aspects (écrans différents), et force l'auteur à une certaine méthodologie dans la conception de son activité : définition de l'abstraction, puis de la présentation et enfin du contrôle (interactions entre l'abstraction et la présentation). Nous faisons ici référence au modèle PAC (Présentation, Abstraction, Contrôle) voir (Coutaz, 1990).

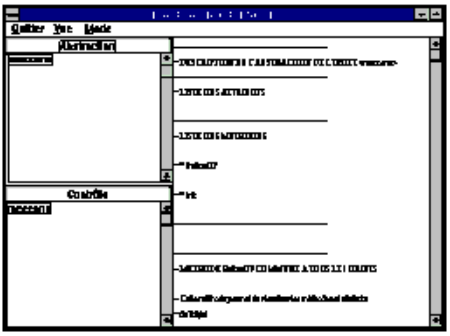
D'autre part, nous nous sommes également posés la question de savoir comment l'auteur pouvait spécifier ces différents aspects. Les systèmes hypermédias de style ToolBook ou HyperCard sont étroitement liés à la notion d'objets de présentation (boutons, champs, objets graphiques...). Cette notion d'objet est très restreinte et en fait très éloignée de celle proposée par l'approche objets (classe, méthode, attribut, instanciation, héritage, généricité, etc.). Elle se résume essentiellement à la copie d'un objet modèle.

Dans notre maquette, nous avons donc voulu d'une part enrichir cette notion et d'autre part la généraliser à la spécification des différents aspects d'une activité (abstraction, présentation, contrôle).

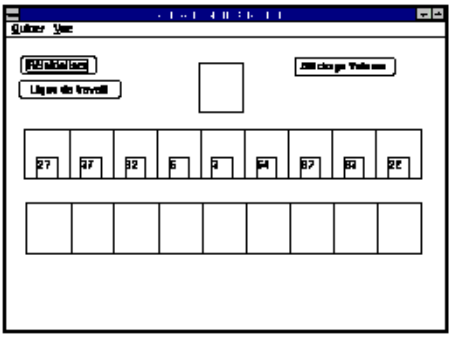
Reprenons l'activité "Meccano de Tri" décrite précédemment et essayons de décrire quelle pourrait être la démarche de l'auteur :

- spécification de l'abstraction. L'auteur définit son abstraction dans le langage de programmation du système : une liste de $2n + 1$ emplacements dont les n premiers sont occupés par des valeurs aléatoires, les autres étant libres ; l'état final souhaité étant que, après permutations, les n premiers emplacements soit occupés par des valeurs croissantes. Il spécifie donc par programmation l'initialisation des données, la vérification de l'ordonnancement après chaque permutation...
- spécification de la représentation. Ensuite, grâce à des outils graphiques (donc sans programmation), l'auteur définit d'une part les éléments de dialogue avec l'utilisateur et d'autre part les objets qui représenteront son abstraction à l'écran. Un emplacement sera symbolisé par une zone rectangulaire, et une valeur par un cube que l'utilisateur pourra manipuler d'une zone à l'autre. Pour définir ces objets dont le comportement est complexe, l'auteur va consulter le catalogue d'objets pour voir si des objets identiques ou approchants n'ont pas déjà été définis. Il trouve dans ce catalogue les objets "aimant rectangulaire" et "rectangle d'attraction" qui répondent à ses besoins, les transfère dans son application et modifie éventuellement quelques détails afin de les adapter à son application. Enfin, il organise ces différents composants pour obtenir l'aspect visuel souhaité.

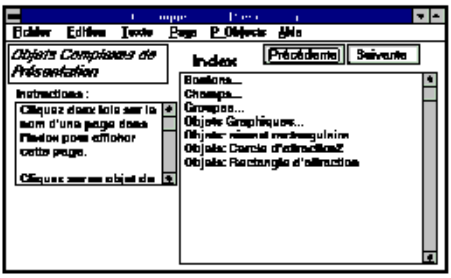
- spécification du contrôle. Pour terminer, l'auteur définit les interactions entre l'abstraction et la représentation grâce au langage de programmation. Par exemple, il spécifie par programmation le comportement du bouton "Réinitialiser" qui provoquera la réinitialisation de l'abstraction. L'intérêt de la notion de contrôle est de pouvoir définir une nouvelle présentation sans avoir à modifier l'abstraction.



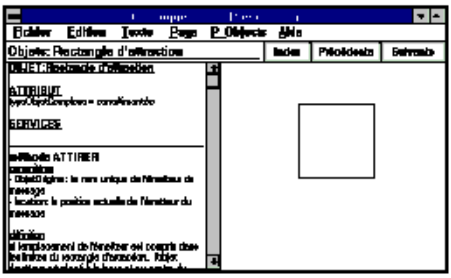
La fenêtre "Abstraction et Contrôle"



La fenêtre "Présentation"



Le catalogue des objets de présentation



Choix d'un objet dans le catalogue

L'utilisation du catalogue d'objets favorise un enrichissement progressif des outils disponibles pour l'auteur et de son savoir-faire mais est indissociable d'une maîtrise suffisante du modèle à objets (en particulier de la notion d'encapsulation).

En conclusion, cette maquette nous montre les directions vers lesquelles nous devons porter aujourd'hui nos efforts :

- quelle culture "à objets" minimale devons-nous demander à un responsable d'activité ?
- le modèle PAC (Présentation, Abstraction, Contrôle) présenté plus haut est-il le plus adéquat ? En particulier la notion de contrôle doit-elle être explicite pour l'utilisateur ?
- quel type d'interface devons-nous offrir, sachant que seule une interface simple et attrayante pourra rendre possible l'utilisation quotidienne d'un tel outil ?

Sur la base des résultats de cette réflexion, nous envisageons la spécification d'un nouvel environnement hypermédia adapté à notre besoin c'est-à-dire à la production rapide et méthodique, par les enseignants eux-mêmes, d'applications à intégrer dans leur pratique pédagogique.

CONCLUSION

Notre expérience dans le domaine de l'enseignement de l'algorithmique nous a conduit à définir plus généralement un certain style d'EAO, intégré de façon quotidienne dans les pratiques pédagogiques des enseignants. Pour permettre une telle intégration, il est apparu nécessaire de disposer d'environnements de production adaptés à une réalisation rapide et facile de petites applications, par les enseignants eux-mêmes. A partir de l'étude de deux types d'environnements existants, nous avons dégagé les caractéristiques des environnements dont nous rêvons. Il semble que ce rêve débouche sur deux outils : l'un (environnement hypermédia étendu), serait destiné à des enseignants ayant une connaissance minimale en informatique, l'autre (environnement objet étendu), plus puissant, s'adresserait à des auteurs professionnels.

RÉFÉRENCES

Baron (G.L.), De La Passardière (B.), 1991 - "Médias, multi et hypermédias pour l'apprentissage". *Hypermédias et Apprentissages*, Actes des premières journées scientifiques, INRP - MASI.

Booch (G.), 1992 - *Conception Orientée Objets et Applications*. Addison-Wesley.

Brown (M.H.), SEDGEWICK (R.), 1985 - "Techniques for Algorithm Animation". *IEEE Software*, vol. 2, n° 1, pp. 28 - 39.

Cagnat (J.M.), Guéraud (V.), Peyrin (J.P.), 1990 - "The Arcade Laboratory : an environment to help teach algorithms". *ACM SIGCSE Bulletin*, Vol. 22 N° 4, pp 37 -42.

Clavel (G.), Veillon (L.), 1991 - *Découvrir la programmation orientée objet avec Smalltalk V*. Masson, Mai 1991.

Coutaz (J.), 1990 - *Interfaces homme-ordinateur : Conception et réalisation*. Dunod Informatique.

Guéraud (V.), Cagnat (J.M.), Peyrin (J.P.), 1991 - "Teaching and Learning made easier by the Arcade Laboratory". *International Conference on Computer Aided Learning And Instruction In Science And Engineering*, Lausanne.

HyperCard, 1987 - *Guide de l'utilisateur HyperCard* et *Guide du langage Hyper-Talk*. Apple Computer.

Rhéaume (J.), 1991 - "Hypermédias et stratégies pédagogiques". *Hypermédias et Apprentissages*, Actes des premières journées scientifiques, INRP - MASI.

ToolBook, 1991 - *Guide d'utilisation de ToolBook* et *Guide d'utilisation d'Open-Script*. Asymetrix.