Informatique et formation : réflexion à partir d'HyperCard de Apple

Gérard H. Gautier

Education Permanente, **numéro 93/94,**
"Que faire des nouveaux médias ?", juin 1988.

Gérard GAUTIER – ggautierk@online.fr

Gérard Gauthier

Informatique et formation : réflexion à partir d'Hypercard de Apple

Récemment, j'ai eu l'occasion de collaborer à la traduction de l'ouvrage de M. Goodman, *A complète Hypercard handbook*,* ce qui m'a permis de découvrir le logiciel dont il traitait. Comme formateur travaillant principalement dans le domaine de l'informatique, je me suis interrogé sur les utilisations possibles de ce nouveau genre d'outils dans les dispositifs de formation, ainsi que sur leurs limites.

Cet article n'est donc pas essentiellement technique. Il utilise plutôt le « prétexte Hypercard » pour réfléchir sur le thème « Informatique et formation ». Je ne cherche pas non plus à faire la publicité d'un produit : celui-ci ouvre, à mon avis, une nouvelle génération de logiciels, et sera sans aucun doute rejoint par des « cousins » d'ici peu de temps. Ceux-ci pourront sans doute fonctionner sur IBM-PC. C'est donc le **concept** que je discuterai plus que sa première réalisation.

Je situerai le logiciel Hypercard parmi les productions informatiques.

Gérard Gauthier est formateur en informatique, et animateur scientifique et technique.

* Hypercard est un logiciel Apple. L'ouvrage en question est publié chez Dunod dans une traduction d'I. Lévy.

Puis j'aborderai ses possibilités d'utilisation. Les concepts informatiques qui sous-tendent sa réalisation seront brièvement abordés. Ceci m'amènera à resituer la position des logiciels d'Enseignement assisté par ordinateur (EAO) face aux approches alternatives de l'usage de l'informatique en formation. Cette approche me permettra de revenir sur le débat concernant les formes d'apprentissage sous-jacent à la controverse entre les partisans de l'EAO et ceux des logiciels « ouverts », et qui le dépasse largement. Un tel rappel ne pourra ici qu'être effleuré. Il permettra cependant une approche « historique » : que sont, entre temps, devenues les pratiques « alternatives » à l'EAO ? La « prophétie » initiale s'est-elle ou non réalisée ? Enfin, je préciserai les limites de ce nouveau concept logiciel en formation, à la fois en termes des objectifs de formation eux-mêmes, et en termes du contexte dans lequel l'institution les assigne à ladite formation.

L'univers technique d'Hypercard par rapport au marché des logiciels didactiques

Une évolution rapide à rationalité essentiellement technicienne

Il n'est pas question, ici, de récrire une classification complète des productions didactiques « traditionnelles » à support informatique. Je serai donc très approximatif et les diviserai simplement en deux catégories, l'une dite « fermée », qui comprendra essentiellement les logiciels d'EAO (Enseignement assisté par ordinateur), et une autre dite « ouverte », recouvrant ce que je désignerai pour l'instant par le mot peu précis d'« environnements d'apprentissage ».

Intéressons-nous tout d'abord aux EAO, puisqu'aussi bien ils sont pour l'instant majoritaires dans les applications de formation. Que voit un utilisateur se servant d'un tel produit ? Dès les premières réalisations, un tel logiciel consistait en un exposé à l'écran d'éléments théoriques, suivi par quelques exercices d'application. Ceux-ci pouvaient consister en simples « questions de cours », ou en problèmes plus complexes. Mais même dans ce cas, le degré d'élaboration du produit restreignait ces problèmes à ceux de la catégorie « fermée »(exercices à une et une seule « bonne réponse »). Beaucoup de ces logiciels étaient rédigés en un langage de programmation « général » – je veux dire par là non orienté spécifiquement vers la production de supports didactiques –, comme Basic, ou Pascal. Le formateur les utilisant – quand il n'était pas lui-même le programmeur - ne pouvait donc que très difficilement intervenir pour en modifier le déroulement.

Puis l'idée est venue d'utiliser des langages de programmation spécialement adaptés à la réalisation de logiciels d'EAO, afin de permettre une programmation plus facile par des non-professionnels tout en leur fournissant des fonctions - notamment dans le domaine graphique – plus adaptées à l'objectif de réalisation spécifique. Une telle évolution des outils logiciels de développement répondait à l'évolution du rapport qualité-prix des micro-ordinateurs.

Le « langage-auteur », ou mieux, le « système-auteur » était né. Il était orienté vers la création de logiciels pédagogiques, comportait un langage de programmation produisant des « séquences didactiques » enchaînées, et souvent un éditeur graphique. En théorie tout au moins, le programmeur pouvait, dès lors, être le formateur lui-même. En fait, la réalisation de nombreux produits a continué à être le fait soit de quelques « mordus », soit de professionnels.

Avec l'apparition de machines plus puissantes et sophistiquées, de périphériques nouveaux (« souris » et table à digitaliser sur le plan strictement informatique, vidéodisque en tant que système connexe), il était aussi possible d'augmenter à la fois l'attractivité des produits didactiques finis et les ressources des logiciels-auteurs utilisés pour les produire, par exemple en les dotant de fonctions de création graphique plus « conviviales ». Enfin, le vidéodisque permettait de réaliser de véritables films dont les séquences arborescentes s'enchaînaient au gré de l'utilisateur (« Si vous voulez que le méchant soit blessé, tapez 1, sinon tapez 2 »).

Parallèlement, les retombées des recherches sur les logiciels de simulation – en particulier les recherches militaires sur les simulateurs de vol – ont fait naître, outre les jeux vidéo des cafés, un type particulier de logiciel didactique, et pas seulement pour les sciences physiques. Certes, sur le plan technique, la production d'images synthétiques a fait un grand bon en avant, mais c'est aussi l'idée même de simulation qui a progressé. Par exemple, le « jeu d'entreprise », déjà utilisé avant l'ordinateur, a pu profiter du nouvel arrivant pour produire des séquences de formation collective pouvant durer jusqu'à plusieurs jours, et bâties également autour de l'idée de **simulation de situation.**

En bref, les techniques pouvant être mises au service de la réalisation de produits didactiques ont rapidement évolué en dix ans, aussi bien concernant les matériels que les logiciels de base. Les recherches cognitives et les nouvelles technologies de l'éducation ont suscité des débats inconnus auparavant. Bien sûr, ça n'était pas seulement pour des raisons techniques. La crise économique sans précédent que nous vivons depuis 1974 a généré un discours politique de la modernisation et des nouvelles technologies qui a mis l'informatique à l'ordre du jour de façon parfois assez « mystique ».

Mais, paradoxalement, en ce qui concerne les retombées pratiques sur la conception de l'EAO, on reste sur l'impression que les développeurs ont tellement été occupés à jouer avec les nouvelles techniques qui apparaissaient quotidiennement qu'ils en ont quelque peu oublié l'apprenant, à la fois sur le plan psychologique et sur le plan social ! L'évolution technique rapide du secteur ne bloque certes pas les développements d'applications, mais elle l'oriente fortement en fonction de critères obéissant à une rationalité esssentiellement... technicienne, même si le fait qu'on ait cherché ce type d'applications relève de l'idéologie.

Certaines recherches sur les moyens de perfectionner les logiciels d'EAO ont bien abordé le problème de l'apprenant lui-même : il y a eu des tentatives

utilisant l'« intelligence artificielle » pour développer une « interface utilisateur » (système de dialogue avec l'apprenant) qui connaisse les erreurs statistiquement le plus fréquemment commises, et utilise cette connaissance pour bâtir des hypothèses sur les représentations de l'apprenant et orienter la suite du processus (projet LIOPE à Jussieu). Mais cette connaissance à acquérir était centrée sur les aspects exclusivement en rapport avec le corpus à lui faire ingérer, d'un point de vue très « que n'a-t-il pas compris ? ». De plus, le coût de ces nouvelles technologies en a réservé l'usage aux secteurs de pointe (système PLATO, en aérospatial), puis à certains lieux muséologiques (Cité des sciences et de l'industrie, bientôt le Futuroscope), et... à la recherche pédagogique, ce qui ne débouche pas toujours sur des applications en « vraie grandeur ». On peut dire aussi que l'arrivée de ces nouvelles techniques n'a pas pour autant fait éclater le cadre somme toute assez fermé du « logiciel d'EAO ». J'y reviendrai.

Comment se situe Hypercard sur ce marché?

On peut considérer Hypercard comme un langage-auteur. Il permet en effet de construire des logiciels d'EAO. Mais on peut également dire qu'il remplit des fonctions particulières de simulation dans le domaine de l'informatique, et tout aussi bien le considérer comme un environnement de développement assez évolué. Comment est-ce possible?

• **L'environnement Macintosh :** ce logiciel « tourne » sur Macintosh, ce qui signifie qu'il utilise l'environnement de travail familier aux utilisateurs de ce dernier. Décrivons-le rapidement:

– la « souris » : ce périphérique est une petite boîte que l'utilisateur peut déplacer en la faisant glisser sur la surface du bureau où se trouve l'ordinateur ;

– un « pointeur », en général une petite flèche – mais ce peut être un autre dessin –, qui reproduit à l'écran les mouvements de la souris ;

– un écran de bonne qualité graphique, sur lequel les objets informatiques apparaissent sous forme de symboles évocateurs (« icônes ») : par exemple, un fichier se trouvant sur une disquette et contenant du texte est représenté par une pile de feuilles de papier; la disquette est un cadre (« fenêtre ») contenant une série de telles représentations. Point très important, une « icône » est **plus** qu'un symbole. Elle est une incarnation de l'objet représenté, un être **actif :** copier une icône, c'est copier le fichier qu'elle représente.

Cet ensemble souris-pointeur-représentation graphique active permet ainsi de remplacer la frappe traditionnelle au clavier de toute une série de commandes, par des manipulations simples portant sur les icônes. La souris comporte un bouton sur lequel l'utilisateur peut appuyer pour « sélectionner » un objet à l'écran. Par exemple, recopier un fichier d'une disquette vers une autre consiste à :

– au moyen de la souris, déplacer à l'écran le pointeur sur l'icône du fichier à recopier ;

– appuyer sur le bouton de la souris pour « prendre » l'icône ;

– tout en maintenant enfoncé le bouton de la souris, déplacer celle-ci, ce qui fait que le pointeur entraîne dans son mouvement à l'écran une icône copie de celle du fichier choisi ;

– amener cette icône sur la fenêtre représentant la disquette cible ;

– « déposer » cette copie sur la disquette en relâchant alors le bouton de la souris.

De plus, en haut de l'écran se trouve une barre horizontale portant une demi-douzaine de titres. Cliquer sur l'un d'entre eux provoque le « déroulement » d'un menu lui correspondant, qui porte diverses commandes ne pouvant être accomplies directement par mouvements d'icônes. Si le menu est dans un environnement ordinaire un moyen d'action assez lourd, ici, l'usage de la souris transforme totalement la situation: l'action est très rapide, et les menus, tout comme les icônes, évitent l'apprentissage de syntaxes compliquées.

• **Reproduire soi-même l'environnement Macintosh :** les utilisateurs du Macintosh étaient habitués à cet environnement. Pourquoi ne pas leur fournir les outils leur permettant de créer eux-mêmes des logiciels personnels où des icônes se trouvant à l'écran pourraient être « activées » par la souris, où certaines opérations pourraient être accomplies de la même manière que la copie de fichier expliquée plus haut ? C'est l'une des idées fondamentales d'Hypercard : **ce que l'on peut fabriquer avec Hypercard fonctionne pour l'utilisateur de la même manière qu'Hypercard lui-même.** Vous pouvez récrire Hypercard avec Hypercard si vous en avez envie.

La seconde idée fondamentale est celle-ci : on doit pouvoir programmer au moins des applications courantes avec un langage réduit à des manipulations du genre de celle déjà décrite : le commun des utilisateurs n'ayant ni temps ni envie d'apprendre à utiliser un langage fait de commandes doit pouvoir fabriquer le logiciel qu'il vise à coups de « clics » de la souris.

Indépendamment de la nature de son « environnement », Hypercard constitue un système de stockage d'informations sur des « cartes » – reproduction à l'écran de fiches bristol – groupées en « piles ». Chaque carte affecte à l'écran une forme rappelant une carte réelle, et peut aussi porter des dessins. Elle comporte surtout des champs pouvant contenir des textes, ainsi que des « boutons ». Qu'est-ce qu'un bouton ? C'est l'équivalent de notre icône de fichier de tout à l'heure, c'est-à-dire un symbole réellement actif. Par exemple: une pile de cartes d'adresses est fournie avec Hypercard. Chaque carte contient un champ réservé pour y porter le numéro de téléphone de la personne, ainsi qu'une icône représentant un combiné. Amener le pointeur sur le combiné et y faire un « clic » de la souris provoque la prise de la ligne et la numérotation par l'ordinateur – à condition de posséder le système de liaison adéquat. Appuyer sur un bouton déclenche des actions.

Il reste à ajouter, pour terminer cette description, qu'Hypercard profite de la qualité graphique du Macintosh pour mettre à la disposition de l'utilisateur un logiciel complet de dessin sur écran, toujours, bien sûr, au moyen de la souris, et

qui peut être utilisé, ainsi qu'une banque de dessins préparés d'avance, pour enrichir considérablement l'environnement graphique des productions développées en Hypercard.

● Plusieurs niveaux d'utilisation :

– *Utilisation de piles déjà faites.* L'utilisateur peut se servir des piles de cartes fournies avec le logiciel – un agenda journalier pour noter rendez-vous et coups de téléphone, la pile d'adresses et la pile « téléphone » déjà mentionnées, etc. –, ou celles qui commencent à être vendues séparément : il place alors dans des cartes définies par d'autres les informations qu'il veut utiliser.

– *Création en « mode graphique ».* Il peut aussi créer ses propres applications, ou « piles », dans la terminologie Hypercard. Un premier niveau de création est totalement basé sur la souris : définir un nouveau format de carte, y placer des champs, des boutons, se fait à la « manière Macintosh ». Quand vous posez un bouton sur une carte, vous le créez à l'aide d'une commande de menu « nouveau bouton », et le « placez » au bon endroit de la carte à l'aide de la souris. Ensuite, vous pouvez choisir successivement son allure à l'écran, et sa fonction dans des fenêtres spéciales.

Le cas le plus simple est celui de la liaison entre deux informations : le bouton intitulé « carte suivante » – et qui porte en général une petite flèche orientée vers la droite – peut être simplement programmé par ce moyen pour « emmener » effectivement le futur utilisateur de la pile à la carte suivante. Un bouton peut vous emmener à une autre carte de la même pile, ou aussi d'une autre pile. Ainsi, faire un clic sur un jour de l'année dans la pile « calendrier » vous emmènera-t-il à la carte correspondant à ce jour dans la pile « agenda », pour que vous puissiez y noter vos rendez-vous...

Un exemple d'application (pour l'instant imaginaire, mais qui sait ?) : vous voulez réaliser un didacticiel sur la faune et la flore du milieu désertique. Un premier dessin apparaîtra, représentant un paysage typique. Un clic sur le cocotier, on obtient un menu proposant de choisir entre des informations sur sa culture, ses besoins en eau, etc. Un clic sur le dessin d'un dromadaire vous envoie sur une série de cartes donnant des informations sur ses moeurs, sa physiologie, etc. Il peut aussi y avoir une carte des lieux, également active : un clic sur un oued provoque l'affichage du paysage correspondant, commenté... On voit apparaître des possibilités d'interaction finalement très proches de celles d'un vidéodisque, pour un coût moindre.

– *Création en « mode langage ».* L'exemple précédent concernait ce qu'il y a de plus simple: une action sur un dessin à l'écran (« bouton ») provoque l'affichage d'autres objets, eux-mêmes reliés à d'autres... Mais si l'on désire, par exemple, obtenir la composition d'un numéro de téléphone comme dans l'un des exemples cités plus haut, il faut accomplir un travail différent d'une simple création de liaison entre cartes ou entre une pile et une autre. C'est un travail de **programmation,** utilisant un langage propre à Hypercard – appelé Hypertalk – et dans ce cas précis, à l'un de ses ordres appelé *dial* (« composer »).

Quelques mots sur Hypertalk : les concepteurs de ce langage ont cherché à le rapprocher le plus possible de l'anglais standard. Ainsi, une phrase comme: *« get the number of this card, put it into N » («* prendre le numéro de cette carte, le placer dans N ») constitue une séquence de deux ordres Hypertalk tout à fait standard. D'une certaine manière, cette tentative tourne le dos à toute une évolution des langages de programmation. Mais le succès est indéniable, même si les versions non anglaises du logiciel ne comportent pas (encore ?) de Hypertalk non anglais. Il semble que le problème des différences syntaxiques entre diverses langues naturelles ait été sous-estimé au départ, quand il fut question de munir chaque Hypercard d'un traducteur intégré. L'auteur voulant mettre à la disposition de l'utilisateur des fonctionnalités plus « évoluées » que la simple liaison entre deux informations stockées en des endroits différents peut donc se servir de Hypertalk.

– *Les concepts informatiques sous-tendant la conception.* On a donc en apparence, en Hypercard, un système-auteur classique, mais utilisant à un point jamais vu auparavant les capacités de la machine qui l'héberge. Ce qui est réellement nouveau dans ce système, c'est la **structure** du logiciel et de son langage. Un programme dans n'importe quel langage prend en général l'allure d'un listing, d'un « texte » unique. Ce n'est pas le cas dans Hypercard, où chaque objet du système possède un programme qui lui est propre, et qui est exécuté quand cet objet est « activé ».

Par exemple, « cliquer » avec la souris sur un bouton, c'est envoyer à ce bouton le message « la souris a été enfoncée » *(mousedown* en anglais et en Hypertalk). Le bouton tente de traiter le message et vérifie si son programme personnel – ou « script » en terminologie Hypercard – possède une procédure de gestion adaptée. Si c'est le cas, le bouton exécute les ordres contenus dans cette procédure, sinon le message est transmis à d'autres (une carte, une pile, un autre bouton...). Dans le cas du bouton « combiné téléphonique » d'une carte « adresse », mentionné plus haut, la procédure activée lors du clic de la souris pourrait déclencher les actions suivantes (description très simplifiée) : récupérer le numéro de téléphone inscrit dans le champ « Numérotéléphone » de la carte ; appeler la fonction *dial* (« composition ») en lui donnant à composer le numéro en question ; ce qui s'écrirait: *« get field "numérotéléphone", dial it ».*

Programmer en Hypertalk c'est donc attacher à des objets Hypercard des procédures dont chacune sera exécutée sur réception d'un message particulier. La souris est, bien entendu, une source de choix pour de tels messages ! Cette structure « décentralisée » et ce mode de fonctionnement, où des « objets » échangent des messages, sont caractéristiques de ce que l'on appelle en informatique un « langage-objet ». Hypertalk est en fait un « smalltalk » (un de ces langages). Aborder ici la manière dont est organisée la transmission des messages (à qui, dans quel ordre sont-ils envoyés?) me mènerait hors du sujet de cet article. Je vais plutôt discuter le problème des **applications réelles** d'un tel système.

Quel usage pour Hypercard ?

On voit que si, par certains côtés, Hypercard est un langage-auteur traditionnel, par d'autres, en particulier sa structuration particulière, l'usage qu'il fait et l'accès qu'il permet à l'environnement cohérent du Macintosh, il peut être qualifié de véritable **environnement de développement.** Les exemples que j'ai mentionnés, même ceux qui sont du domaine de l'imaginaire, me semblent montrer clairement qu'un outil comme Hypercard peut être utilisé pour fabriquer des logiciels didactiques. Mais il reste encore à préciser quels usages réels il peut recevoir. La question est en gros : voici une perfection technique certaine, mais pour quoi faire ?

Le fantasmatique et le réel au niveau du public

Le mot « utilisateur » est ambigu pour un logiciel où l'on passe de façon aussi continue, selon ses concepteurs, de l'usage des produits fabriqués avec lui à la création de produits. Désigne-t-on par ce mot le formateur ou le programmeur ? Dans l'esprit des concepteurs – et de l'auteur de *The complète Hypercard handbook* –, n'importe qui ou presque peut se débrouiller avec le logiciel, et presque n'importe qui avec le langage Hypertalk. La réalité sera peut-être plus modeste. Pour plusieurs raisons indépendantes.

D'abord, on ne peut pas ne pas remarquer le saut de difficulté important entre la première partie du livre de Goodman, consacrée essentiellement à l'utilisation – ce que l'on appelle en jargon Hypercard le mode « navigation » – et à la création en « mode souris », et la seconde, où sont expliqués les ordres Hypertalk.

Ensuite, le public visé par les concepteurs aux USA n'est pas forcément celui qui sera touché en France, en raison non seulement de la différence relative de prix du Macintosh entre les deux marchés, mais aussi de celle entre les deux manières de concevoir la production... En France, cette machine est typiquement un outil de cadre supérieur voire de chef d'entreprise : le directeur a **son** Macintosh – la secrétaire de direction éventuellement –, alors qu'on commence à trouver des IBM-PC dans des pools de dactylos..., parfois de la même entreprise. (Je ne parle pas des applications spécifiques pour lesquelles le « Mac » a une longueur d'avance sur l'IBM et où son usage s'impose quel que soit le poste de l'opérateur dans l'entreprise.) Qu'on le veuille ou non, votre ordinateur vous dénote hiérarchiquement. Notons que cette remarque ne s'appliquera pas totalement aux « imitations » de Hypercard qui verront sans doute le jour pour le PC d'IBM.

Les problèmes posés par la traduction des matériels didactiques se rapportant au logiciel sont des reflets de cette différence de public potentiel..., et des usages qui les intéresseront, ainsi que du fait que les concepteurs ont parfois « fantasmé » sur certains usages de leur « bébé », qui ont peu de chance de se développer à grande échelle.

Enfin, l'expérience, vieille de plusieurs années à présent, du développement de logiciels d'EAO pour et dans l'éducation nationale devrait amener à se montrer prudent quant aux nouvelles possibilités introduites par un nouvel outil, si « génial » fût-il techniquement. Savoir programmer n'est pas équivalent à savoir développer des produits didactiques, savoir développer des outils didactiques n'est pas en avoir le temps dans son institution, et en avoir le temps n'est pas en être chargé...

Il faudra donc distinguer parmi les usages possibles techniquement, le fantasmatique et le réel, le plus probable à court et à moyen terme. En fait, derrière l'angélisme de façade, de mise dans les textes (« tout le monde pourra développer ses propres applications »), la politique d'Apple est soigneusement calculée pour permettre aux développeurs indépendants de s'insérer sur le marché en vendant des piles à leur compte.

Le fantasmatique et le réel au niveau des usages

La tentation est grande pour l'instant de privilégier les applications didactiques portant sur d'autres logiciels fonctionnant sur Macintosh. Comme Hypercard peut reproduire facilement l'environnement classique de cet ordinateur, il peut être utilisé pour **simuler** des séquences à l'intérieur d'un autre logiciel du marché. Par exemple, on peut récrire un morceau d'un tableur (feuille de calcul automatique) en Hypertalk. L'intérêt n'est, bien sûr, pas de substituer la version Hypercard du produit à l'original, mais bien plutôt de modifier les messages envoyés à l'écran par le « simulateur » par rapport au logiciel simulé. On obtient ainsi un didacticiel du produit qui peut passer par des phases « fermées » – menus fournissant des renseignements, exercices à faire... – et des phases « ouvertes » : tandis qu'il réalise l'exercice, l'apprenant ne voit aucune différence entre la simulation sur laquelle il travaille et le logiciel réel..., sauf quand il commet une « erreur ».

C'est au niveau des messages envoyés lors d'actions inadéquates de l'apprenant que celui qui a développé le simulateur peut intervenir : – en **rendant plus explicites** des messages fort sybillins dans la version originale du logiciel à apprendre (par exemple, remplacer « accès impossible » par « le traitement de texte n'arrive pas à charger en mémoire le texte que vous lui avez demandé » ;

– **en distinguant des erreurs** auxquelles le logiciel original réagit de la même manière : remplacer le message « commande inconnue », soit par « le tableur ne connaît pas la commande Gllark », soit par « pour utiliser la commande Assign, vous devez vous placer d'abord dans le menu Fichier », ne peut qu'aider l'apprenant à comprendre mieux quelle fut son erreur – ce n'est malheureusement pas le cas de la plupart des logiciels... ;

– en tenant compte de la situation présente, soit de l'exercice que l'utilisateur doit réaliser, soit de la démarche précédente de l'utilisateur (dernières commandes employées), et en tirer une analyse simple de l'erreur commise...

On voit qu'un tel travail met à la fois en jeu des compétences de formateur et d'informaticien... Il me semble qu'il y a peu de chances pour que ces deux

fonctions soient accomplies par la même personne. Les utilisations que l'on voit pour l'instant sont plus modestes : « logiciels pour cadres » comme ceux mentionnés plus haut (agenda, planning...), ou EAO classique, accompagnant éventuellement un support de cours sur papier. D'autres utilisent l'aspect « belle présentation » du produit: telle cette entreprise de formation qui, au dernier salon de la Publication assistée par ordinateur, présentait son programme de stages sur une disquette Hypercard. On peut imaginer toute une série d'utilisations « gadget », voire un peu « frime », comme par exemple un rapport d'activité d'un service sur écran géant dans une réunion de direction...

Cela ne signifie pas que d'autres applications n'apparaîtront pas, mais on peut déjà relativiser cette idée pas si nouvelle - mais qui réapparaît à chaque nouveau support technique! - qu'un support technique peut avoir une action quelconque **en soi** pour modifier les pratiques. C'est un peu sur l'histoire récente de cette idée que se centrera la section suivante.

L'univers pédagogique : formation par EAO ou formation par logiciel ouvert ?

Pas plus que les autres évolutions techniques, Hypercard ne remet en cause le cadre fermé de l'EAO. Il est effectivement lui-même un outil de développement fort ouvert au sens informatique : on peut le considérer comme un langage muni d'un grand nombre de fonctions préprogrammées. En ce sens, il est ouvert, et il est significatif que, dans la section précédente, la liste des utilisations possibles en soit restée elle aussi... ouverte: Peut-on définir par avance **toutes** les utilisations d'un langage quel qu'il soit ? Mais il ne s'ensuit pas que ce que l'on réalisera avec cet outil soit ouvert au sens pédagogique – même si on a vu qu'il est possible de réaliser des didacticiels à la fois plus « léchés » et plus « intelligents » qu'avec des outils plus classiques. Examiner ce que l'on entend pas ce terme d'« ouvert » en ce sens non informatique nécessite d'élargir le champ pour resituer la position de l'enseignement assisté par ordinateur face aux approches alternatives de l'usage de l'informatique en formation. Il nous faut réintroduire l'outil technique dans le débat pédagogique.

La controverse EAO-logiciels « ouverts »

Au moment où l'informatique se révélait comme moyen pédagogique, une alternative aux logiciels d'EAO s'esquissait avec les propositions de S. Papert, et son langage Logo. Dans son livre, *Le jaillissement de l'esprit,* ce chercheur américain proposait un nouvel « environnement d'apprentissage », centré sur l'usage d'un mobile se déplaçant à l'écran (« tortue »). A l'origine conçu avec de véritables mobiles reliés à l'ordinateur et roulant sur le sol, Logo servait à la latéralisation des très jeunes enfants. Mais il évolue très vite vers des ambitions plus larges: Papert travaille au MIT en face d'un laboratoire d'intelligence artificielle, et Logo en hérite des caractéristiques du langage Lisp sur lequel ce laboratoire travaille.

L'intelligence artificielle, qui cherche à reproduire informatiquement des modes de raisonnement propres à l'être humain, est traversée par des interrogations psychologiques sur le fonctionnement du cerveau. Ainsi, Papert qui est un piagétien, se retrouve-t-il en pays de connaissance. Logo peut devenir un outil de recherche cognitive permettant de mettre en action le cerveau dont on cherche à comprendre le mode de raisonnement. C'est à cela que s'emploient en effet les premières expérimentations.

Mais Logo est aussi au départ un outil d'apprentissage par expérimentation – ce qui fait son intérêt –, et visant le domaine des mathématiques. Il va échapper partiellement à son inventeur suivant un processus d'« emballement » – à tous les sens du mot. On va s'emballer pour Logo, et sa diffusion va connaître la fortune que l'on sait. Il finira comme le langage par excellence.

La confusion qui subsistera sera toujours la même : quels objectifs pour l'informatique ? Apprendre **par** l'informatique ou apprendre l'informatique ? Sans aller plus avant dans l'exploration des problèmes posés par la polysémie du mot « informatique », on retrouve les prémisses d'une question qui concerne toujours Hypercard : Est-ce un outil pour apprendre spécifiquement certains produits informatiques ? Ou un support de formation plus général ?

Au travers de la fortune du langage Logo, on voit déjà poindre tous les dangers de la « mystique de l'informatique » ou de la mystique de tel ou tel produit : l'attribution à un outil technico-pédagogique particulier d'une vertu allant au-delà de son rôle d'outil. Je ne veux pas déconsidérer le travail fait avec ce langage, ou dire par là que la seule voie est celle de l'EAO. Mais il importe de se rappeler, par exemple, les déclarations d'un Jean-Jacques Servan-Schreiber, après l'ouverture du Centre mondial de l'informatique: « L'informatique peut réduire les différences sociales ».

L'arrière-plan théorique

Ceci dit, un débat plus « sérieux » que celui auquel peuvent mener de semblables déclarations est sous-jacent à celui autour d'environnements pédagogiques tels que Logo. Il porte sur les **formes d'apprentissage.** Un récent article de Marcel Crahay dans la *Revue française de pédagogie* en trace l'historique. Sans reprendre l'article en question, je voudrais en tirer trois questions théoriques.

- **Question théorique 1 : apprend-on des procédures ?** Logo est un langage qui décompose les actions à accomplir en procédures composées chacune de plusieurs instructions simples. L'hypothèse sur laquelle repose l'efficacité d'environnements comme celui-là est que le cerveau mémorise des procédures – des suites d'instructions. Ces environnements visent donc à faire apprendre des savoir-faire, conçus comme des méthodes de construction d'enchaînement d'actions efficaces, plutôt que des savoirs. Jean Lave cite dans un numéro de *Culture technique* (« La qualité de la quantité », n° 14, 1985), une curieuse expérience de psychosociologie qui pose de façon inhabituelle le

problème de la manière dont nous mémorisons et mettons en œuvre nos méthodes d'actions quotidiennes. Je l'appellerai « l'expérience des tortillas » : « Une acheteuse et l'observateur se dirigent vers le rayon des surgelés du supermarché. L'acheteuse veut acheter un paquet de tortillas surgelées. Avant d'être arrivée en face du rayon, c'est comme si elle était à une distance non seulement physique, mais aussi cognitive, des tortillas. En revanche, lorsque l'acheteuse est mise en présence des tortillas, l'activité acquiert une qualité entièrement différente. *L'acheteuse:* (s'exprimant avec hésitation et cherchant des yeux le rayon des tortillas): « Actuellement, ces tortillas coûtent à peu près 55 cents. C'était le prix quand je les ai achetées la dernière fois, mais chaque fois que je reviens, elles augmentent ». *L'observateur: «* Y a-t-il une sorte que vous aimez plus que les autres ? *» L'acheteuse: « Oui,* c'est celles de... euh, je ne sais pas, je ne me souviens pas quelle marque les fait. En plus, ils changent toujours les produits de place. Je ne sais pas ». Quelques minutes plus tard, ayant découvert le rayon des dîners mexicains : « Ah, les voilà! (Parlant avec fermeté et conviction.) Elles étaient à 65 cents la dernière fois que j'en ai achetées. Maintenant, elles sont à 69. C'est terrible, hein ? » Lave tire de cet épisode la conclusion que nos savoirs sont **reconstitués en situation,** et non mémorisés en tant que tels. (Avez-vous déjà tenté de décrire **précisément** à un ami le chemin qui vous mène de chez vous à votre travail tous les jours?)

• **Question théorique 2 : les acquis procéduraux sont-ils réinvestissables ?** En admettant que les acquis permettant l'action quotidienne soient bien des procédures, indépendamment de leur mode de stockage, le problème reste posé de leur généralité. Crahay montre fort bien dans son texte que les expériences tentant d'établir ce fait pèchent quant au fond. L'expérience citée par Lave, qui se pose aussi le problème de la manière dont nous nous y prenons pour agir quotidiennement, ne répond à aucune question de ce genre, puisqu'elle tend au contraire à démontrer la dépendance de nos actes par rapport au contexte.

• **Et le contexte social de l'apprentissage ?** Par contre, au niveau de l'apprentissage lui-même (et non de l'action l'utilisant), les piagétiens dissidents, telle Anne-Nelly Perret-Clermont, ont mis en avant l'importance de l'élaboration de méthodes de résolution de problèmes au cours de « conflits cognitifs », au sens de désaccords dans un groupe d'apprenants quant à la méthode à utiliser pour résoudre un problème.

On voit qu'aucune de ces trois questions n'est résolue par un choix de support. En fait, on peut même dire que l'usage de supports « ouverts » comme Logo a permis d'actualiser dans la pratique plus de questions qu'il n'en a résolu – ce qui n'est pas forcément une critique, bien sûr!

L'arrière-plan « pratiques alternatives »

Mais le débat réel autour de l'usage de l'informatique que recouvrait le débat sur l'EAO, même s'il ne s'est pas exprimé de manière aussi formelle, était bien plus celui du choix entre: utiliser l'informatique comme un nouveau support pour des méthodes traditionnelles; ou utiliser l'informatique comme moyen de faire éclater les méthodes traditionnelles. On peut à présent se poser la question de savoir ce que sont devenues entre temps les pratiques « alternatives ». La « prophétie » initiale des opposants à l'EAO s'est-elle ou non réalisée ? Mon sentiment est qu'il y a eu, pour une large part, institutionnalisation des prophéties les plus radicales par l'usage jouissif de la technique : Logo, de langage de recherche cognitive, est devenu le langage d'initiation à la programmation: il s'est « intégré ». Les cercles qui étaient « radicaux » le sont restés, parfois en s'emparant des nouvelles techniques, parfois en les récusant justement comme fausses solutions. Les autres, soit ont pris soin de ne pas toucher à ce qui sentait le soufre, soit l'ont totalement intégré dans leur pratique antérieure. En quelque sorte, le prochain cours en amphi sur les outils techniques permettant de se passer de cours en amphi a déjà eu lieu.

Limites de ce type d'outils

Le débat pédagogique débouche donc sur un débat cognitif, mais on a vu que cet élargissement du champ n'est pas suffisant pour nous permettre d'apprécier la position réelle de ces nouveaux moyens techniques. Il nous a fallu franchir une étape encore, et réintroduire ces débats eux-mêmes dans l'environnement institutionnel de la formation pour montrer qu'en aucun cas, quelle que soit l'attirance exercée par des outils de développement de ce genre, ils ne constituent une solution. Le danger est bien plutôt qu'ils **n'apparaissent** illégitimement comme une solution. En forme de conclusion, je voudrais citer quelques problèmes que les formateurs reconnaîtront à coup sûr comme produits par cette attirance.

Quelques perversions tentantes

– La tentation de cesser de réfléchir en termes d'objectifs de formation pour se laisser aller au plaisir de la technique: ce texte lui-même oscille entre les deux positions, et j'en suis conscient. (Il me semble cependant que la première solution à cette perversion est bien de tenter au moins d'osciller ainsi, c'est-à-dire d'être capable de changer de point de vue.)

– La tentation de « remplacer » une mauvaise pédagogie par une bonne technique : c'est une conséquence directe de la précédente, l'enfermement dans le niveau de réflexion technique.

Les perversions induites par la « mystique de l'informatique »

Elles ne sont pas propres à ces nouveaux logiciels, mais plus l'outil semble techniquement au point et « facile à utiliser », plus elles sont tentantes. Comme si la « mystique de l'informatique » avait pour conséquences l'abandon des modes de pensée rationnels applicables dans les autres domaines. J'en ai donné la base dans la section qui précède. Voici quelques exemples vécus :

– Choisir matériels et logiciels sans conseil préalable, sans consultation extérieure sur les aspects informatiques, sans consultation interne du personnel qui devra ensuite utiliser les matériels et les logiciels. On aboutit ainsi à des produits incompatibles et inadaptés.

– Démarrer un plan de formation sans conseil préalable, conséquence directe de la précédente : plus on a choisi seul au départ, plus il est difficile d'admettre son erreur éventuelle ensuite...

– L'illusion que savoir se servir d'un logiciel dispense de savoir faire correctement ce à quoi le logiciel aide. Par exemple, la conception de didacticiels demande un savoir-faire non exclusivement informatique. Aucun environnement de développement ne vous apprendra ça ! De même que savoir utiliser un logiciel de mise en page ne vous donne pas en soi une compétence dans le domaine de la mise en page.

– L'illusion que puisque c'est « facile à employer », les problèmes non techniques (filière de production) seront résolus du même coup. Exemple du chef de service qui vient d'être informatisé, qui n'a pas participé lui-même aux formations qu'il a fait organiser pour le personnel. Au bout de quinze jours, il demande fort en colère: « Pourquoi les étiquettes adresses n'ont-elles pas été faites à temps, alors qu'à présent vous avez un logiciel qui les sort automatiquement ? ». Il n'a pas pensé une seconde au problème **non informatique** du choix d'organisation des fichiers, qui seul permet de répondre en fait à sa question.

L'informatisation fait donc apparaître les besoins en formation des cadres. C'est un lieu commun de dire que l'apparition de l'informatique oblige à mettre au net, à rationaliser des pratiques hors de son domaine strict. Indirectement, la perfection technique réelle des nouveaux outils aide à oublier ces problèmes. Les personnes qu'on met devant un écran, après des formations qui sont souvent aussi exclusivement techniques, elles, ne peuvent guère les oublier !

Ainsi, on revient de manière plus « classique » au rôle de révélateur du non-dit que peut jouer toute organisation de formation. Dans le cas d'une formation pour l'utilisation de nouveaux produits informatiques, c'est au travers de ces outils eux-mêmes, et éventuellement des moyens de formation retenus, didacticiels ou non, que peut passer l'analyse spontanée. Depuis le problème du lieu où on doit installer le stage, jusqu'à la réflexion sur les usages ultérieurs du produit, en passant par la négociation de qui doit aller en formation, tout peut jouer le rôle de révélateur des carences du processus de décision, et surtout des réseaux de domination, subis depuis si longtemps qu'ils en sont devenus psychologiquement « invisibles » tout en étant paradoxalement trop bien connus. La situation exceptionnelle de la formation peut les faire réémerger, et provoquer la reprise de la parole. La rationalisation informatique aussi, en ce qu'elle les réorganise: untel devient nouveau détenteur de l'information technique, un autre s'en fait exclure, la division du travail peut devenir plus rigide, ou au contraire s'« ouvrir ». Mais si la technique est, d'une certaine manière, le lieu de cette émergence, elle n'en est pas porteuse.