12

FORMATION EN EDUCATION N°1.6

RECHERCHE

CENTRE O.S.E.

LOGO : un langage, une théorie, un environnement...

Marc ROMAINVILLE

Jean-Pierre PETERS

DEPARTEMENT
EDUCATION & TECHNOLOGIE

Facultés Universitaires Notre-Dame
de la Paix B-5000 Namur

R. TUREL
oct. 87
3050 v

RÉSEAU O.S.E.

Ordinateur au Service de l'Éducation

LOGO:

un langage, une théorie, un environnement...

M. ROMAINVILLE
J. P. PETERS

Décembre 85

CENTRE O.S.E. NAMUR
Facultés Universitaires N.D. de la Paix
Unité de Pédagogie
61, rue de Bruxelles - 5000 NAMUR

On le dit souvent, LOGO n'est pas qu'un langage.
On parle d'ailleurs d'" environnement LOGO ", de " philosophie LOGO ".

Il n'est évidemment pas possible-ni facile!- de résumer en quelques pages, ni les théories sous-jacentes (le constructivisme de Piaget et la notion d'intelligence artificielle) ni les recherches qui ont abouti à la création du LOGO.

Ce modeste document n'est donc qu'une introduction à ces questions; une bibliographie vous permettra, si vous le désirez, de continuer la réflexion entamée ici.

Mais d'abord, un peu d'histoire ...

" LOGO est un nom utilisé d'abord par W. Feursig chez B.B.N, pour désigner un langage de programmation de la famille de LISP (LISts Processing language de John Mac Carthy). C'est ensuite un nom utilisé au Massachussets Institute of Technology, à partir de 1971, par l'équipe de Seymour Papert et Marvin Minsky, pour désigner un projet situé à la convergence des recherches en " Intelligence Artificielle " et en " Sciences de l'education " (1).

Il est intéressant de signaler que S. PAPERT a travaillé jusqu'en 1964 avec J. PIAGET à l'école d'épistémologie génétique de Genève. Ensuite, il rejoint le M.I.T. où il s'occupe essentiellement d'intelligence artificielle; ces deux domaines ne sont pas aussi éloignés qu'il n'y paraît: les recherches en intelligence artificielle qui visent à fabriquer des machines " douées de raison " passent d'abord par une connaissance approfondie de ce qu'est l'" apprentissage ", de " comment les humains pensent ".

En 68-69, les premiers essais du système LOGO ont lieu avec une douzaine d'élèves; le projet LOGO démarre réellement en 71.

Depuis lors, le LOGO qui au départ n'existait que sur gros systèmes a été implanté sur une gamme assez étendue de microordinateurs et de nombreuses équipes de recherches ont vu le jour aux quatre coins de la planète (l'équipe de BOSSUET en France, par exemple).

Mais qu'est-ce que LOGO ? C'est G. BOSSUET qui a donné, me semble-t-il, la définition la plus complète et la plus pertinente du "LOGO".

" LOGO est à la fois une théorie sur la construction par l'enfant de ses connaissances, un langage de communication et un ensemble d'unités matérielles permettant la mise en évidence des processus mentaux mis en jeu pour résoudre des problèmes que l'utilisateur se pose lui-même avant d'ESSAYER de leur apporter une solution.
LOGO n'est pas un nouveau langage pour programmer les ordinateurs.
LOGO est un outil qui permet de matérialiser nos images mentales, comme l'imprimerie permet, dans la pédagogie " Freinet ", de mieux maîtriser les rapports à l'écrit. LOGO a été conçu comme un outil adapté à la pensée, comme le marteau est adapté à la main ou l'avion au transport aérien ". (1)

---

(1) G. BOSSUET; Où en est LOGO, GREPACIFIC, mai 1984.

Passons en revue les 3 composantes de cette définition :

* la théorie

La théorie LOGO a été exposée par S. PAPERT dans son livre de base " Jaillissement de l'esprit ".
Comme nous l'avons déjà dit, S. PAPERT a été influencé par les théories Piagétiennes et principalement par son idée maîtresse : le constructivisme.

Ce dernier courant est souvent opposé - si l'on considère globalement l'histoire de la philosophie de l'éducation - au romantisme et à la transmission culturelle.

Pour le romantisme, les connaissances sont pour la plupart en germe dans l'individu; le rôle de l'éducation consiste à révéler ses potientalités. On peut citer à titre d'exemples, J-J, ROUSSEAU (l'Emile,1762), A.S. NEILL (Libres enfants de Sumerhil, 1960) ou dans tout autre registre, N. CHOMSKY.

Les tenants du second courant - la transmission culturelle - estiment eux que l'éducation est essentiellement une transmission d'informations, les matières décomposées en petites unités sont progressivement imprimées chez l'apprenant. Le behaviorisme est évidemment l'exemple type de ce deuxième courant.

Le constructivisme (J. PIAGET) tente de dépasser cette opposition en posant que"la connaissance n'est pas préformée dans l'individu; elle ne s'imprègne pas non plus dans l'esprit sous la pression des éléments extérieurs. Elle se construit au travers des interactions du sujet avec l'objet. Au départ, seules sont données quelques actions élémentaires (sucer, regarder, prendre, etc ..) qui, par généralisation, différenciation et coordination réciproques vont engendrer un répertoire plus vaste d'actions ou de chaînes d'actions, plus complexes et mieux ajustées.

Cette généralisation des actions initiales l'amène inévitablement à rencontrer des obstacles qui l'obligent à inventer de nouvelles formes d'action. L'objet, c'est à dire le monde physique, ne s'impose pas à un sujet passif; c'est l'enfant qui va à sa rencontre et s'efforce de l'intégrer, de l'assimiler à ses formes d'actions privilégiées.
Quant à l'objet, il peut se laisser faire; mais, parfois, il résiste ou réagit de manière surprenante: il contraint alors l'enfant à un réajustement, à une accomodation.
L'apport de l'interaction avec l'objet ne se réduit donc pas à un acquis qui se superposerait au potentiel de départ. En rencontrant l'objet, le sujet transforme ses schèmes d'assimilation initiaux."(1)

Le rôle de l'éducateur, pour ce troisième courant, sera de stimuler un processus d'interaction avec l'environnement pour amener l'enfant à construire de nouvelles structures mentales.

Pour PIAGET, ce processus de construction suit chez tous les enfants approximativement le même ordre : stade des opérations concrètes, stade des opérations formelles, etc ...

---

(1) M. CRAHAY.; Observer et réguler la construction des actions avec les objets, Thèse de Doctorat, Université de Liège, 1984, p.42.

Bien qu'appartenant au même courant, PIAGET et PAPERT divergent sur certains points précis, notamment sur l'importance des matériaux fournis par une culture donnée. Pour PIAGET, ils n'ont qu'une importance mineure : la succession des stades et leur chronologie sont immuables puisqu'elles dépendent de facteurs de maturation.
Pour PAPERT au contraire, un environnement riche favorisera une accélération du développement; pour lui, il importe de donner aux apprenants des métaphores, des moyens de concrétiser certaines opérations formelles et c'est parce que - jusqu'à présent - la culture environnante ne donnait pas de telles possibilités aux enfants que l'ordre était respecté.

Nous approchons ainsi progressivement du noyau central du système LOGO : ce dernier a été conçu de telle sorte qu'il offre à l'enfant un matériel riche, peu structuré, personnel, avec lequel il pourra essayer ses démarches de résolution de problèmes.

Le LOGO servira de "miroir à sa pensée". Il existe d'ailleurs un parallélisme entre la construction de programme en LOGO par procédures et la construction des connaissances par combinaison de structures de pensée. "L'évolution d'un programme LOGO par le développement et le testing de procédures simples puis l'incorporation de ces procédures dans de plus complexes offre un certain parallélisme avec les processus de pensée ... l'enfant construit ses structures de pensée en explorant le monde extérieur. Face à un problème, les structures mentales existantes sont combinées pour produire une nouvelle solution qui peut être utilisée dans d'autres occasions ou transférée à d'autres situations. L'exploration et la découverte sont des éléments clés de la théorie piagétienne de l'apprentissage. Ils sont également au coeur de toute expérience d'apprentissage LOGO : l'utilisateur explore des situations et résoud les problèmes en développant des programmes LOGO à partir des procédures qu'il a lui-même construites. (2)

Le LOGO offrira également des "métaphores" pour de nouveaux apprentissages plus complexes, tout comme les engrenages de l'enfance de PAPERT l'ont aidé à comprendre les équations différentielles. La loi fondamentale pour PAPERT est que tout apprentissage est facile dès l'instant où l'enfant peut le rapprocher de modèles déjà existants. Et c'est l'incapacité de la culture environnante à fournir à l'apprenant des référents concrets et manipulables qui rend certaines notions si difficiles.

Pour mieux percevoir la partie de cette théorie, prenons quelques exemples:

a. la pensée combinatoire

Pour PIAGET, elle est caractéristique du stade des opérations formelles, elle n'apparaît donc qu'à 11, 12 ans.
Ainsi, si l'on donne à des enfants plus jeunes un sac rempli de perles de différentes couleurs, ils seront incapables de former toutes les combinaisons possibles à deux termes (tous les couples possibles) de perles de couleurs différentes.

(2) A. MARTIN, Teaching and Learning with LOGO, London, Croom Helm, 1986, (p. 6)

Pour PAPERT, cette opération n'est pourtant pas plus abstraite qu'une autre mais les enfants n'ont que peu de référent : notre culture est pauvre en " modèles de procédures systématiques ".
Des enfants habitués à travailler sur l'ordinateur auraient à son avis moins de difficultés. Il s'agit d'un programme à boucles emboîtées : choisir la couleur 1 comme première couleur, former tous les couples possibles, idem avec couleur 2, éliminer les couples faisant double emploi.

b. le principe fondamental de la physique de Newton :

Un corps en mouvement continuera à se déplacer en l'absence de toute action contraire, indéfiniment, à vitesse constante et en ligne droite.

PAPERT explique que ce principe pose un problème, essentiellement parce qu'il vient en contradiction avec l'expérience familière, parce qu'il manque aux apprenants une expérience concrète et directe du mouvement newtonien.

Ce genre d'expérience peut être apportée par la manipulation des DYNATORTUES (tortues en mouvement).

c. la pensée autoréférentielle (la pensée sur la pensée)

Egalement caractéristique du stade des opérations formelles, elle est favorisée pour PAPERT par le système LOGO.: l'enfant qui doit apprendre de nouveaux mots à la tortue (= programmer) doit s'interroger intuitivement sur ce qu'est apprendre , comment lui-même apprend de "nouveaux mots ". De même, constatant une erreur dans son dessin, il doit " décortiquer " sa propre démarche de résolution de problème pour y repérer l'erreur.

Citons pour terminer quelques autres points importants de la théorie LOGO.

- A la différence de PIAGET, S. PAPERT insiste sur l'aspect affectif de tout apprentissage : l'assimilation de connaissances, si l'apprenant peut faire référence à du " déjà connu ", prend une coloration affective positive. De plus, le système LOGO permet à l'enfant de construire, selon ses intérêts, son projet personnel.
PAPERT reconnaît que tous les enfants ne peuvent s'intéresser aux engrenages; l'ordinateur par contre est un outil multiforme pouvant servir la construction de projets très différents suivant les intérêts de l'enfant.

- Par rapport à l'enseignement traditionnel, le statut de l'erreur est modifié.

" La réaction enfantine typique, en cours de maths traditionnel, est *d'oublier* le plus vite possible l'erreur qu'on vient de commettre.
En milieu LOGO, il en va autrement. D'abord l'erreur n'est pas un tort, mais un accident de programme, un " bug ". Par conséquent, pas de remontrance. Chercher à éliminer le " bug " fait partie intégrante du processus de compréhension d'un programme. L'enfant qui programme est donc encouragé à étudier de plus près ce " bug " plutôt qu'à l'effacer de sa mémoire au plus vite. Dans le monde des tortues, plus que nulle part ailleurs, il y a tout intérêt à examiner le " bug ". C'est un exercice fructueux." (1)

---

(1) S. PAPPERT, Jaillissement de l'esprit, Paris, Flammarion, p.82 (1980)

- L'apprentissage, en système LOGO, se veut fonctionnel; les notions nouvelles sont abordées quand elles deviennent nécessaires; des concepts sont introduits pour mener à bien un projet personnel de l'enfant. Ainsi, si un enfant désire dessiner des carrés de différentes dimensions, c'est fonctionnellement - c'est à dire en fonction de son but personnel précis - qu'il va devoir déterminer les caractéristiques formelles d'un " carré ".

  Par cet aspect, le système LOGO peut être rapproché de la pédagogie du projet.

Pour terminer cette première partie, il nous semble important d'insister sur les différences de philosophie sous-jacentes à l'E.A.O. classique d'une part et au LOGO d'autre part.
Si l'on peut rapprocher directement le LOGO au constructivisme, la philosophie sous-jacente à l'E.A.O. classique - du moins sous le mode tutoriel - s'apparente quant à elle plus au courant de la transmission culturelle : la matière est découpée en petites unités qui sont elles-mêmes présentées progressivement à l'apprenant; des renforcements accompagnent des réponses de plus en plus complexes. L'erreur est évitée au maximum.
Les connaisances sont déposées ainsi progressivement chez l'apprenant, elles s'additionnent au fur et à mesure de la présentation organisée de matières nouvelles.

* l'environnement LOGO

- l'univers matériel

Au départ, le dispositif comprend une tortue mécanique (tortue de plancher) commandée par des cartes perforées. Elle est capable de d'avancer, de reculer, de modifier sa position, de lever ou de baisser sa plume.

Ensuite, cette tortue s'est modifiée en tortue d'écran (un simple triangle sur écran).

Le matériel doit avoir une mémoire suffisante (64K) et il est intéressant de disposer d'une imprimante pour garder une trace du travail effectué.

- le rôle de l'éducateur

C'est un rôle délicat car il est assez différent du rôle classique :

" L'enseignant LOGO répondra aux questions, apportera son aide si on la lui demande, et s'assiéra parfois à côté d'un élève pour lui dire : " Laisse-moi te faire voir quelque chose." Ce qu'il va montrer ne lui est pas dicté par un programme strict. Parfois, c'est quelque chose dont l'élève va pouvoir se servir immédiatement dans l'entreprise en cours. Parfois, c'est quelque chose que l'enseignant lui-même vient d'apprendre, et dont il pense que la découverte va intéresser aussi son élève.
Parfois encore, tout simplement, l'enseignant agit spontanément, il improvise, comme on le fait souvent dans un groupe social en situation non structurée, quand on est pris ensemble par quelque chose de passionnant.". (1)

---

(1) S. PAPERT, idem, p.224.

* le langage LOGO

La caractéristique essentielle du LOGO est donc de donner à son utilisateur une structure bien adaptée à des formes nouvelles d'apprentissage.

Dans le cas de la "tortue d'écran", le microordinateur n'est que le moyen physique de l'expression du langage. Décrivons brièvement les structures internes de ce langage plus ou moins naturel, capable aussi de comprendre des mots français.

Comme tout langage, il est constitué de mots et de signes de ponctuation et est organisé grâce à un ensemble de règles syntaxiques. Le vocabulaire est riche, la ponctuation simple et les règles de syntaxe peu nombreuses. Ces caractéristiques générales permettent à un non-initié d'arriver rapidement à des résultats satisfaisants.

Une première caractérisation du LOGO est obtenue en distinguant ses composantes en procédures et objets.

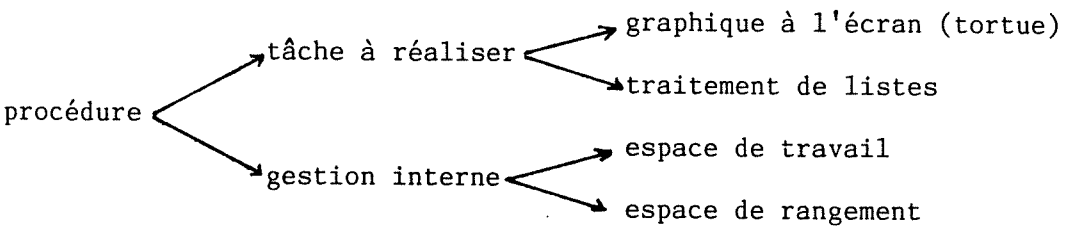
- En fonction de ce qu'elles produisent, les procédures peuvent être classées en commandes et en opérations : une commande produit soit une action directe sur l'environnement (par exemples : un déplacement de la tortue à l'écran, la mise en route d'un périphérique) soit une modification pour la suite,de l'état de cet environnement (par exemple en baissant le crayon ou en fixant la couleur du fond de l'écran). Une opération fournit ("rend", "retourne") une information sous forme d'un objet. Cette information est utilisable pour une autre opération ou pour une commande.
Dans son état "natif", la tortue comprend déjà plus de 150 procédures ce sont les primitives du langage; il est de plus très facile de lui apprendre de nouveaux mots de vocabulaire.

- Les objets peuvent être classés en caractères, mots, nombres et listes.
Les caractères isolés et les nombres peuvent être considérés comme des cas particuliers de mots. Les mots sont séparés les uns des autres par des délimiteurs qui peuvent être l'espacement, un signe d'opération arithmétique ou logique, ..., tandis que les listes sont constituées de mots compris entre crochets : [...] ou entre un début de ligne et l'entrée de la touche "Return" au clavier.
On peut enfin, à l'aide de procédures, comparer deux objets d'un même type et passer d'un type d'objet à l'autre : extraire des mots d'une liste ou des caractères d'un mot, construire des mots à partir de caractères ou des listes à partir de mots.

La variable est un objet qui a reçu un nom et qui peut dès lors être manipulé en rappelant ce nom. L'objet peut changer en cours de travail et on utilise ainsi toute une variété d'objets grâce à un seul nom. Ainsi s'introduit la notion de variable qui peut être étendue de plusieurs façons fort intéressantes si on veut aller au delà des procédures graphiques.

Une deuxième caractérisation du LOGO est établie en fonction des domaines dans lesquels les procédures produisent leurs effets. Cette distinction n'a de sens que dans une démarche d'analyse du langage : fonctionnellement les différences sont masquées au niveau de la syntaxe et du statut des mots; le "système opératoire" est intimement lié au langage de programmation proprement dit.

En réalisant une telle classification,on obtient ce schéma :



Chaque flèche indique vers quel domaine portent les effets de la procédure.
La tâche à réaliser est l'objectif que s'est fixé l'utilisateur avant de débuter son travail au clavier (par exemple paysage ou forme géométrique à dessiner à l'écran, liste ou variables à manipuler pour répondre à telle question).

La gestion interne est celle de la machine elle-même. L'espace de travail (mémoire vive) contient tous les mots connus de la tortue : les procédures constituant l'ensemble du programme nécessaire à la réalisation de la tâche et les variables qu'elles utilisent. L'espace de rangement désigne les disquettes sur lesquelles sont réunies les procédures et les variables qui doivent être réutilisées lors d'une autre session de travail.

QUE PEUT- ON FAIRE AVEC LOGO ?

Très régulièrement,des enseignants se montrent intéressés par le système LOGO mais éprouvent des difficultés à déterminer ce qu'ils pourraient en faire dans leur classe.
Cette difficulté pourrait être expliquée,d'une part,par la raison suivante: comme nous le verrons ci-dessous, le système LOGO, à l'origine, visait à développer chez l'enfant plus des aptitudes cognitives de base que des contenus spécifiques.L'organisation scolaire étant basée précisément sur des cloisonnements de matières,il est difficile de préciser où interviendra exactement le système LOGO

D'autre part, l'environnement LOGO a été conçu en dehors de toute structure scolaire. On pourrait même dire, à la lecture du livre PAPERT, qu'il a été conçu contre toute structure scolaire en proposant la disparition de l'école comme institution d'apprentissage cloisonnée.
PAPERT prône ainsi une transformation totale des contenus et l'abolition des découpages en disciplines.

On comprend dès lors mieux le sentiment de malaise qu'éprouvent les enseignants à devoir introduire dans leur école un système nouveau dont un des objectifs, à l'origine,était précisément la disparition de l'école !

Dans cette introduction au LOGO, il ne nous semble pas utile de dresser une liste exhaustive des activités et des objectifs liés à l'environnement LOGO.

Nous avons préféré analyser sommairement les implications d'une utilisation du LOGO avec une classe. Ces implications sont de 4 types (A. MARTIN)

- **implications au niveau du programme**
- **implications au niveau du style d'enseignement**
- **implications au niveau des ressources**
- **implications au niveau de l'organisation de la classe**

A Implications au niveau du programme

Deux questions essentielles se posent ici :

1. Quel est le rapport entre les compétences que développe un environnement LOGO et les objectifs poursuivis actuellement par l'enseignant ?

2. Quels sont les nouveaux objectifs que l'environnement LOGO permettrait de poursuivre ?

La séparation entre ces deux questions est bien évidemment arbitraire, notons, par exemple, que l'utilisation du LOGO pourrait entraîner une compréhension plus profonde des contenus traditionnels (par exemple, la notion d'angle).Il y aurait dans ce cas de nouveaux apprentissages à propos de contenus anciens.

1. LOGO ET LES OBJECTIFS ACTUELS

1.1. Le lien le plus évident entre LOGO et les contenus scolaires est **mathématique** . L'utilisation de la tortue graphique permet à l'enfant d'explorer les notions de formes géométriques, d'angles, etc. Citons, par exemple, cette expérience de découverte de la symétrie axiale à partir de la réalisation de deux oreilles identiques pour le dessin d'un CHAT :

"Les réalisations de certains projets ont permis d'aborder tout naturellement la notion de symétrie axiale; de découvrir les propriétés pour les utiliser et rendre plus aisée l'exécution de certaines parties de dessins. Le problème s'est posé en particulier pour les oreilles du CHAT. le dessin de l'oreille gauche est d'abord tracé en mode pilotage, par approches successives; si l'instruction qui vient d'être tapée donne un résultat jugé acceptable par les enfants, ils la notent sur le cahier et continuent, sinon ils l'enlèvent de l'écran et recommencent en modifiant les données. Le dessin achevé, ils écrivent alors la procédure OREILLEG avec les instructions qu'ils ont retenues. Pour tracer l'oreille droite, ils hésitent à se lancer dans une recherche analogue et me demandent comment ils pourraient faire pour dessiner, selon leur expression : "la même chose à l'envers" ou de "l'autre côté".

Intuitivement, ils pressentent que les résultats qu'ils ont obtenus pour tracer la partie gauche, doivent pouvoir être utilisés pour la partie droite. Pour amener les élèves à comprendre et interpréter cette situation, j'utilise les écrans vidéo qui constituent d'excellents miroirs avec le contre-jour de la fenêtre.

Je leur suggère de comparer leurs attitudes à celles de leur image observée à travers ces miroirs. Ils constatent alors que les déplacements sont identiques : lorsqu'ils avancent (ou reculent) leur image avance (ou recule) de la même manière, par contre les rotations sont inversées, l'enfant qui se tourne vers la droite voit son image effectuer une rotation vers la gauche et inversément; comme Les MOUVEMENTS DE L'ENFANT SONT AUSSI CEUX DE LA TORTUE, les modifications à apporter dans la procédure OREILLEG pour obtenir une OREILLED apparaissent clairement.

Après les constatations faites sur les modifications à apporter à nos actions motrices, nous retournons au dessin sur la feuille de papier et je place deux tortues au point origine (ou au point de départ choisi pour la tracé précédent). Un élève déplace l'une d'elle en lisant à haute voix les instructions de la procédure, l'autre les exécute simultanément avec la seconde tortue en signalant au passage la modification qu'il doit apporter pour se déplacer et dessiner "de l'autre côté".

Les élèves sont rapidement convaincus que cette façon de faire apporte bien une solution au problème qu'ils se posaient : sans plus s'occuper du dessin, ils écrivent directement la procédure OREILLED à partir de celle de OREILLEG en effectuant au fur et à mesure la transformation".(1)

GA -----> DR
DR -----> GA

Ces tentatives de détournement du système LOGO à des fins mathématiques est actuellement assez fréquente et par ailleurs, assez évidente. Nous ne nous y attarderons donc pas et nous développerons plutôt d'autres expériences plus originales concernant d'autres contenus.

(1) Une géométrie de Tortue, une expérience en CM2, Pédagogie et Informatique, Fascicule 1, IREM, Paris NOrd, p. 24 (1982)

1.2. Le système LOGO permet en effet d'aborder d'autres contenus scolaires, le traitement de liste semble notamment favoriser chez les enfants une **réflexion sur les structures de la langue.**

SHARPLES (1) a ainsi testé en classe quelques programmes de traitement de liste au cours de ce qu'il a appelé un "langage workshop". Un premier programme GRAM 1 génère aléatoirement des phrases; il permet de démontrer aux enfants le fait élémentaire qu'une phrase est plus qu'une séquence aléatoire de mots.

On propose aux enfants d'améliorer ce programme en créant des "catégories de mots"; plusieurs catégories sont proposées et l'on essaie à nouveau la composition aléatoire; on observe que certaines catégories sont exclusives, etc.

1.3. Le traitement de liste permet également de traiter avec simplicité de petites **bases de données** . Ainsi une classe de 3ème et 4ème année (2) primaire a mis au point une base de données concernant les propriétés essentielles des aliments les plus courants; chaque nom d'aliments est relié à une liste de propriétés (substance organique, fonction, groupe, etc.) Les enfants peuvent modifier certaines propriétés, enrichir la base de données, la réduire et la consulter à partir de différents points d'entrée (ainsi, on peut analyser le repas de la cantine en vérifiant qu'un élément de chaque groupe y est présent).

"L'exemple de cette démarche - recherche de propriétés, constitution de la mini-base, utilisation de celle-ci - montre la relation qui s'établit entre l'outil informatique et les activités d'apprentissage. Par sa conception, la mini-base oblige l'enfant à organiser, à schématiser, donc à mieux comprendre. Il lui faut au préalable formuler clairement ses résultats afin de pouvoir les communiquer. En outre, ce travail lui permet de conserver les résultats sous une forme immédiatement accessible.

La mini-base de données constitue un patrimoine commun à une classe : elle est à sa disposition à tout moment. Puisqu'elle est évolutive, d'autres classes peuvent la consulter, la modifier ou la compléter". (2)

(1) Cf. MARTIN A., op.cit., p.164-175

(2) HELM J.G., GUIN D., LOGO et les mini Bases de Données, Education et Informatique, 28, 1985.

1.4. Enfin LOGO est un langage de programmation comme un autre, l'enseignant peut l'utiliser pour construire des produits d'**enseignement assisté par ordinateur** .

Citons, par exemple, les logiciels suivants :

ECLECT (1) : "L'idée de base est de récupérer les possibilités dynamiques du point lumineux colorié qu'est la tortue LOGO pour faciliter l'acquisition de l'écriture manuscrite, de la dynamique de la main". (1) Le logiciel permet à l'élève de percevoir les différentes formes entrant dans la constitution des lettres (ex. : O L pour le A), de suivre du doigt la formation des lettres à l'écran, de jouer avec ces formes et ces lettres, de reconnaître les lettres dans les mots et leur raccordement.
(Cf. ci-dessous)

écriture sans raccord.

écriture avec raccords.

(1) VIVET et al. n., ECLECT, Un logiciel pour faciliter l'acquisition de l'écriture manuelle et de la lecture, Actes du Colloque EAO 84, Lyon, 1984 (335-342).

PHRASES RELATIVES (1) : On propose à l'enfant une phrase incomplète. Il doit ajouter le pronom relatif, il n'y a pas une bonne réponse (ex. : La fille ... joue au ballon est ma voisine;dans ce cas les réponses acceptables sont: avec qui, avec laquelle, contre qui, contre laquelle), le programme "calcule" pour chauqe phrase les pronoms admissibles et les affiche à l'écran après une bonne réponse. (Cf. écran 1) Dans le cas d'une réponse incorrecte, il précise la ou les constructions verbales (Cf. écran 2)

ECRAN 1

```
Complète la phrase suivante par un pronom relatif:
   L'homme ... Serge parle est son oncle.

Réponse:   avec qui

Très bien. Tu aurais pu aussi employer:
   L'homme de qui Serge parle est son oncle.
   L'homme dont Serge parle est son oncle.
   L'homme duquel Serge parle est son oncle.
   L'homme avec lequel Serge parle est son oncle.
   L'homme à qui Serge parle est son oncle.
   L'homme auquel Serge parle est son oncle.
Tape "Esc" pour arrêter, ou n'importe quel autre caractère pour continuer
```

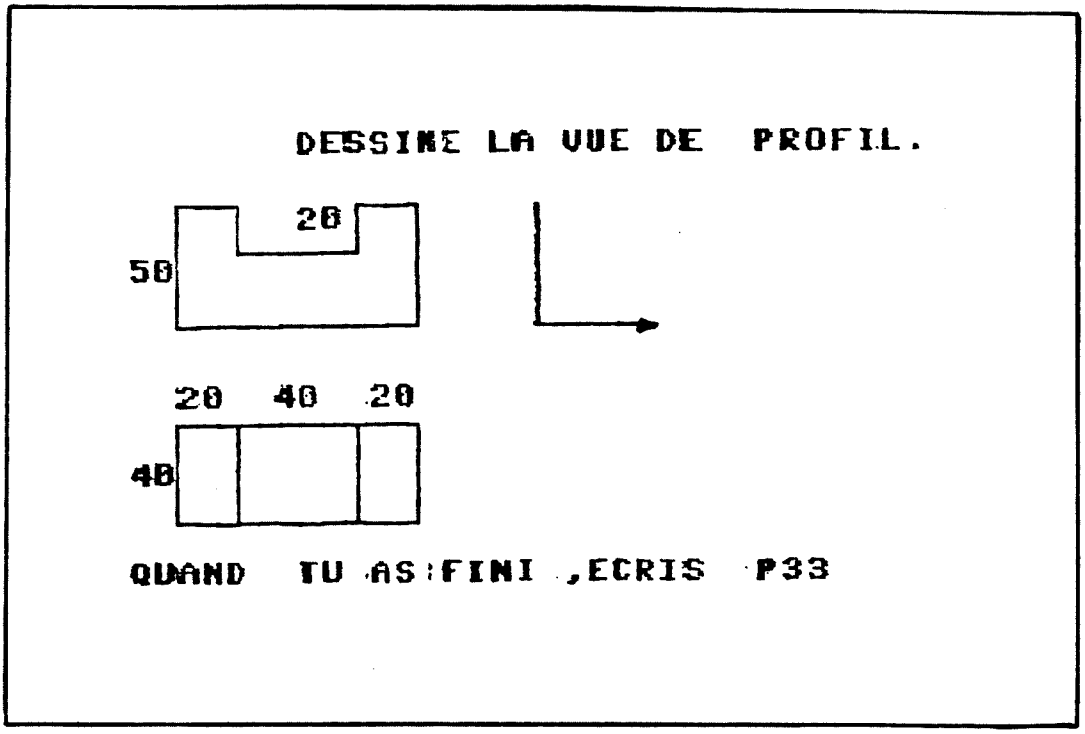
ECRAN 2

```
Complète la phrase suivante par un pronom relatif:
   L'homme ... Serge parle est son oncle.

Réponse:   que

ATTENTION!
     parler à ...
     parler avec ...
     parler de ...

Essaie encore.
```

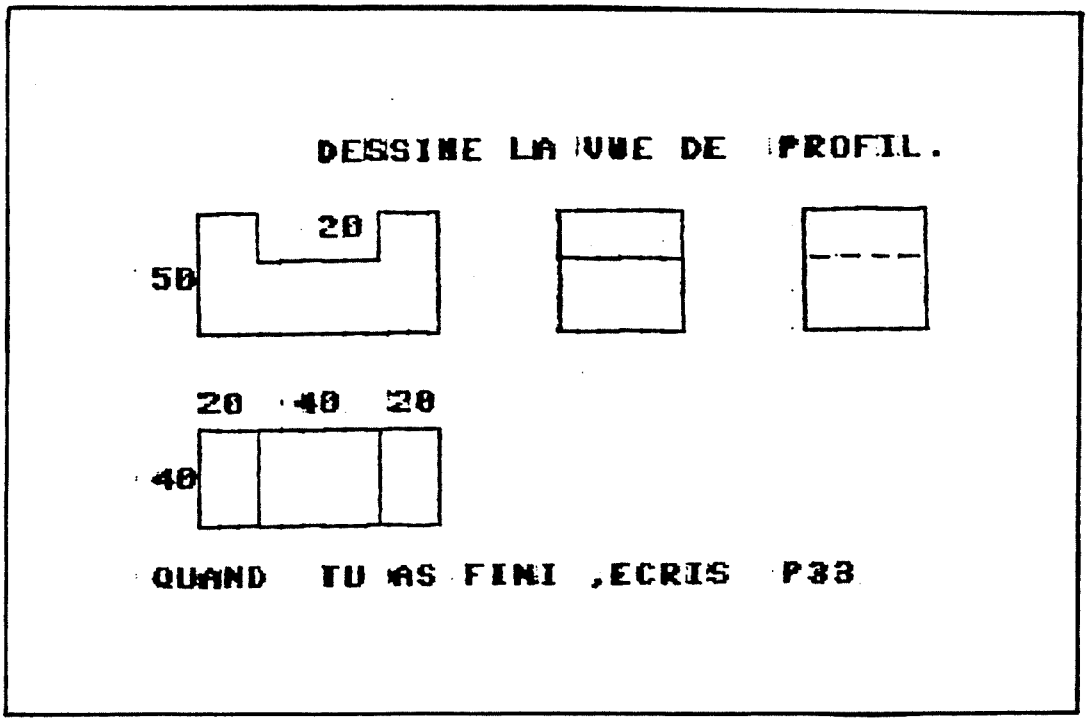
(1) EMIRKANIAN L. et BOUCHARD L.H., Utilisation de l'ordinateur pour l'apprentissage de la langue, Actes du Colloque EAO 84, Lyon, 1984 .(97-110).

DESSIN TECHNIQUE : Mis au point par un enseignant de la région namuroise (Ph. BRANDT), ce programme propose à l'élève deux vues d'une pièce; celui-ci repasse alors au commande du langage LOGO et est invité à réaliser la 3ème vue de la même pièce (Cf. écran 1)

A la fin de l'exercice, l'élève peut comparer son résultat avec la réponse correcte enregistrée. (Cf. écran 2)



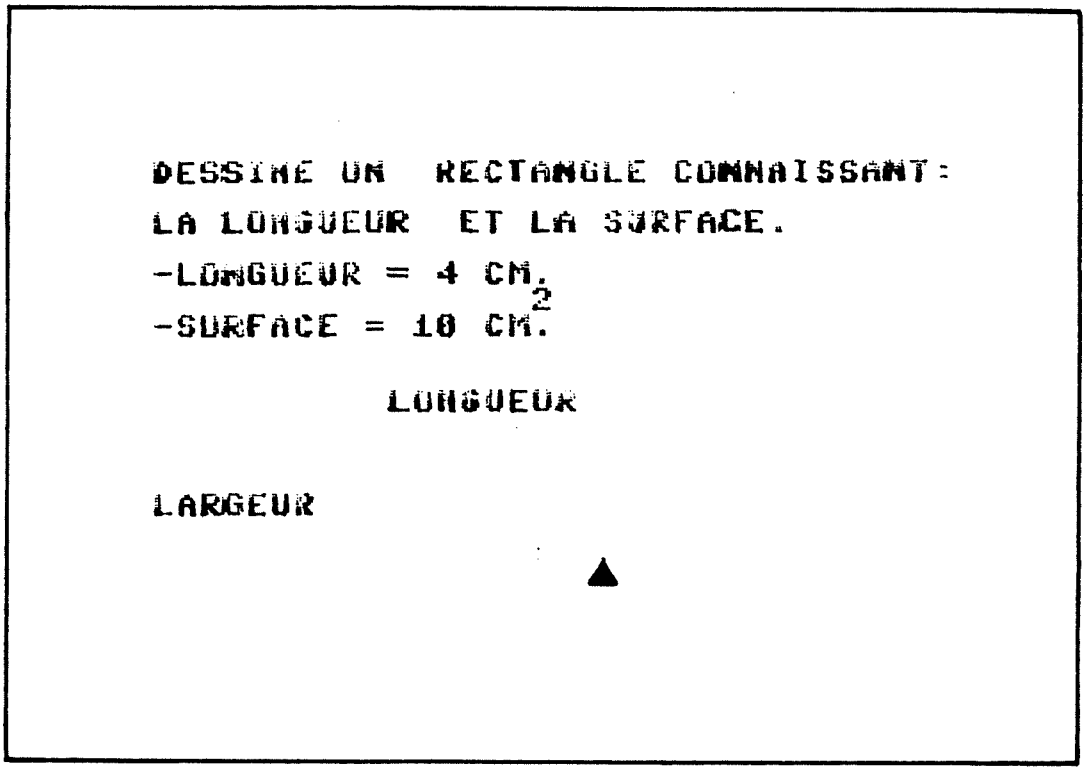
ECRAN 1



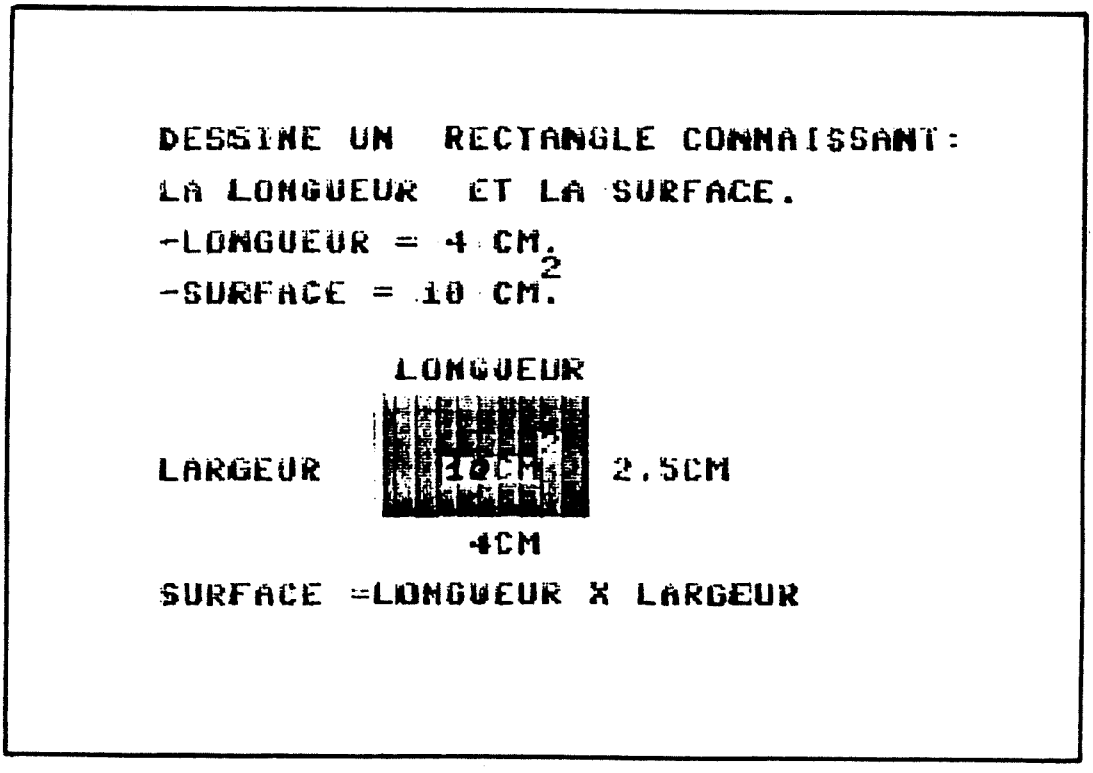
ECRAN 2

Le même principe a été utilisé pour construire un second didacticiel : FORMES GEOMETRIQUES. Un des éléments de la forme géométrique apparaît sur la gauche de l'écran (le côté, le périmètre, la surface, la médiane, etc. (Cf. écran 3)

L'élève est invité à construire la forme demandée à l'aide d'instructions LOGO. La bonne réponse apparaît également à la fin de l'exercice sur demande de l'élève (Cf. écran 4)



ECRAN 3



ECRAN 4

Notons qu'il est particulièrement intéressant pour des programmes simples écrits en LOGO, que les enfants puissent analyser comment le programme fonctionne.

L'avantage de didacticiels en LOGO est précisément cette relative transparence :

"Ces inventeurs ont voulu que les programmes LOGO soient facilement compréhensible. La programmation ne devait pas, d'après eux, rester un art secret réservé à un petit nombre. La construction d'un programme LOGO par procédures que l'on définit soi-même et auxquelles on peut attribuer des noms significatifs rend la structure d'un programme LOGO assez claire. Les didacticiels LOGO sont donc plus faciles à adapter et à comprendre. L'auteur d'un programme important en LOGO le développe en identifiant clairement des modules, chaque fonction intervenant dans le programme général est isolée dans une procédure ayant un nom significatif. La procédure générale présente les éléments principaux du programme et la fonction de chacun est reprise dans son nom. Il en va de même à l'intérieur de chacun de ses modules : le nom donne une bonne idée de ce que fait la procédure .L'adaptation et l'extension sont donc possibles".(1)

L'activité de la classe peut même consister essentiellement en une reconstruction ou un prolongement du programme analysé (Cf. Les programmes de SHARPLES ci- dessus)

2. LOGO ET LES ACTIVITES NOUVELLES

Les activités nouvelles développées dans un environnement LOGO peuvent être très grossièrement regroupées en deux ensembles :

a. **Eveil informatique** : le contact de l'enfant avec l'environnement informatique permet de réaliser un début d'alphabétisation informatique.

* Approche de la machine, de son fonctionnement, de son rôle exact (la tortue doit recevoir des ordres pour arriver à faire quelque chose) de ses potentialités et de ses limites.

(1) MARTIN A., op. cit., p. 197 .

• Approche de l'activité de programmation : qu'est-ce que programmer ?, l'approche descendante ?, les notions de procédure, de variables, etc.

Cetta activité de programmation en LOGO est souvent présentée comme favorisant des habiletés générales de résolution de problèmes ,ce qui nous amène au second point.

b. **Entraînement aux processus de résolution de problèmes**

"Un attrait important du LOGO réside dans sa potentialité à développer des habiletés métacognitives du type résolution de problème ou jugement critique. Parmi les nombreuses expériences d'apprentissage réalisées en LOGO, une place importante est occupée par celles qui visent à entraîner aux processus de résolution de problèmes.

1. Une expérience de résolution de problèmes dans laquelle ceux-ci doivent être formulés de manière hiérarchique donc favorisant une approche "descendante".

2. Un mode de pensée locale et par procédure.

Par opposition à l'approche "montante" typique en E.A.O. traditionnel (l'élève doit maîtriser une séquence d'étapes avant d'essayer de proposer une solution) LOGO favorise une meilleure approche vis-à-vis de problèmes complexes. L'attention est focalisée sur des buts de haut niveau, à l'opposé de la fragmentation des processus d'apprentissage en une série de tâches de bas niveau."(1).

Dans cette optique, on considère que la programmation permettrait l'acquisition de capacités générales de haut niveau, comme le font remarquer PEA et KURLAND (2), cette croyance est "une vieille idée dans un nouveau costume", en effet, de tels arguments ont déjà été utilisés dans le passé, concernant l'utilité des mathématiques, du latin ou du grec. Il convient donc d'être particulièrement prudent vis-à-vis de cette position "technoromantique".

(1) GALLINI J.K., Instructional conditions for Computer-Based Problem Solving Environments, Educational Technology, février 1985 (7-11)
(2) PEA R.D. and KURLAND D.M., On the cognitive effects of learning computer programming, New Ideas in Psychology, N.Y., Pergamon Press, vol. 2, 2, 1984 (p. 137-168).

Et d'abord quelles capacités cognitives générales pourrait-elle développer ?

FEURZEIG (1) résume en 7 points les changements deans les structures de pensée que pourrait entraîner cette nouvelle activité.

(1) La découverte et l'entraînement à l'explicitation des probèmes,à la pensée et à l'expression précise et rigoureuse.

(2) La compréhension de concepts généraux tels que des procédures, variables, transformation, etc.

(3) La familiarisation avec un certain nombre d'"heuristiques", c'est-à-dire des approches générales utiles pour résoudre des problèmes dans n'importe quel contexte (ex. : décomposer un problème en sous-problèmes)

(4) La découverte que le "debugging" des erreurs est une activité constructive et "planifiable", cette idée est, de nouveau, applicable dans de nombreux domaines.

(5) L'idée générale que l'on peut inventer des petites procédures pour ensuite construire des solutions à des problèmes plus importants

(6) La découverte et la réflexion sur les processus de résolution de problèmes que l'on a utilisés

(7) La prise de conscience qu'il existe rarement une bonne manière de résoudre un problème mais différents chemins qui ont chacun leurs avantages et inconvénients en fonction du but poursuivi.

Cette conception est centrale dans un certain nombre d'applications actuelles du système LOGO.

Pour illustrer cette position, prenons deux exemples de proposition d'applications du LOGO à l'école, l'une extraite d'un manuel, l'autre d'un curriculum.

Les écoles publiques de LINCOLN (Nebraska, U.S.A.) ont ainsi défini un "curriculum LOGO" allant de la maternelle jusqu'à la fin du secondaire. Ce programme prévoit un ensemble d'activités structurées axées sur le LOGO. A l'école primaire, le curriculum met surtout l'accent sur le "problem solving" et les mathématiques.

(1) Adapté de PEA R.D.. et KURLAND D.M., op. cit.

Plus précisément, citons l'objectif principal 500 :

"L'étudiant découvrira des stratégies de résolution de problème en utilisant le LOGO pour :

1. Développer des capacités d'estimation par déduction logique.

2. Développer une approche modulaire de la résolution d'un problème.

3. Prédire le résultat des procédures.

4. Répondre de manière constructive aux messages d'erreur.

5. Etablir une séquence logique.

6. Définir les procédures dans un langage spécifique.

7. Continuer de modifier les procédures existantes. "(1)

• Un second exemple éclairant est la classification proposée par REGGINI des objectifs généraux en rapport avec l'introduction du LOGO à l'école. Au sein des objectifs cognitifs, il insiste notamment sur les capacités de résolution de problèmes.

- Développer et approfondir les capacités pour résoudre des problèmes. Leur solution est un sujet capital dans l'enseignement. La plupart des disciplines peuvent être définies comme un ensemble de problèmes à résoudre. L'étude d'une discipline doit amener à comprendre la nature de ses problèmes. Elle inclut l'étude des problèmes déjà résolus, la façon de les résoudre, la solution des problèmes en suspens et les façons de présenter et d'aborder les nouveaux problèmes. LOGO comporte l'apprentissage d'un langage. Mais le plus important est qu'il comporte la recherche et la pratique des ressources pour résoudre des problèmes.

- Exercer l'analyse à travers la division des problèmes en petites parties.La segmentation des problèmes et la correction des procédures sont des idées très pratiquées dans le modèle LOGO.

(1) BURNETT J.T., FRIESEN C.D., Development of a K-12 computer science curriculum utilizing the LOGO computer language as a primary vehicle,Computers in Education, K. DUNCAN and D.HARRIS (eds.), Elsevier, IFIP, 1985 (p.603).

- Exercer la synthèse dans la construction de Macroprocédures de haut niveau de complexité, combinés aux Sous-procédures relativement élémentaires." (1)

Cette conception de la programmation comme outil de développement de la pensée pose néanmoins un certain nombre de problèmes.

Le problème le plus crucial est celui du transfert : les stratégies cognitives ainsi développées seront-elles également appliquées dans d'autres situations ou dans d'autres contextes ? Les recherches actuelles sont contradictoires et posent plus de problèmes méthodologiques qu'elles n'apportent de réponses. Il semble d'ailleurs que le problème soit mal posé et cela pour au moins deux raisons :

a. La première est clairement expliquée par NORTON

"Avec l'introduction de l'ordinateur, de nombreux chercheurs ont fait l'hypothèse que le travail avec cet outil avait un impact sur les capacités de résolution de problème. PAPERT et d'autres ont essayé de déterminer les effets spécifiques de l'utilisation de l'ordinateur sur la pensée. Jusqu'à présent pour les évaluations du LOGO, par exemple, il y a peu d'études qui établissent clairement des gains concernant les capacités de résolution de problèmes. Le problème avec ces recherches est qu'elles ont tendance à poser des vieilles questions à propos d'une nouvelle technologie. (questions qui ont trait à une culture de l'écrit) et ne trouvent ainsi que peu de résultats significatifs. (2)

L'idée de NORTON est qu'il est inutile de tenter de mesurer des gains quantitatifs car l'impact d'une utilisation de l'ordinateur de type LOGO se réalise précisément à propos des capacités nouvelles propres au médium rarement associées à des situations de résolution de problèmes classiques dans une culture de l'écrit.

(1) REGGINI H.C., "LOGO, des ailes pour l'esprit", Paris, Cedic Nathan, 1983 (p. 184).

(2) NORTON P., Problem-Solving Activities in a Computer Environment : A different Angle of Vision, Educational Technology, octobre 1985 (p. 36-41).

Ainsi alors qu'on encourage traditionnellement le raisonnement **déductif** et **logique**, une activité de type LOGO amène plutôt l'apprenant à développer un raisonnement **inductif** et **intuitif**.

"Les élèves qui veulent dessiner un cercle en LOGO commence par examiner leur connaissance personnelle des cercles, peut-être en traçant un cercle ou en se déplaçant en cercle et ensuite en analysant ce qu'ils ont fait. En appréhendant le concept de cercle globalement, les étudiants tentent d'identifier les variables qui interagissent d'une certaine manière pour créer une forme circulaire. Par l'expérience, il dégage deux éléments essentiels : le "avance" et l'angle. Alors ils tentent d'établir les règles selon lesquelles ces deux variables interagissent". (1)

b. Deuxièment, toute activité de programmation ne mène sans doute pas aux mêmes possibilités de transfert.

Il est important en effet de bien définir ce que l'on entend par "programmation", car plusieurs auteurs y englobent des activités somme toute fort différentes. "Programmer", sert de boîte noire, d'activité globale dont les effets sont supposés irradier ceux qui y sont exposés". (2)

C'est ainsi les effets vont notamment dépendre du type de programmation enseignée, des conditions matérielles dans lesquelles les étudiants devront travailler, la durée du cours, etc.

Par exemple, un enseignement de la programmation qui se réduit à l'apprentissage d'un code de communication avec une machine a peu de chance d'amener les effets souhaités.

De même, il semble que le transfert ne se réalisera qu'à partir d'un certain stade de développement de la capacité à programmer.

(1) NORTON P., op. cit., p. 38

(2) PEA and KURLAND, op. cit., p. 144

PEA et KURLAND ont en effet montré qu'il existait différents stades de développement pour l'apprenti-programmeur, plus le programmeur devient expert, plus il se centre sur les techniques mêmes de résolution de problèmes; le transfert, d'après eux, ne peut se réaliser qu'à ce niveau. Des recherches futures devraient donc tenter de déterminer **dans quelles conditions quel type de programmation pourrait entraîner certaines habiletés de résolution de problème transférables.**

B Implications au niveau du style d'enseignement.

Le système LOGO sera sans doute plus efficace avec certains styles d'enseignement qu'avec d'autres. Basé sur l'apprentissage par la découverte, il exige un équilibre judicieux entre des périodes d'essais et erreurs et des interventions de l'enseignant. Ce rôle de "l'animateur LOGO" est loin d'être évident : doit on tout laisser découvrir (PAPERT) sinon quand intervenir et comment ? Faut-il orienter les solutions des élèves et comment ?

Le LOGO développe d'ailleurs, d'après les enseignants qui l'ont testé, de nouvelles attitudes par rapport au savoir et par rapport à "l'apprenant" : impossibilité de se définir des micro-objectifs, nouveau statut de l'erreur (et donc, sans doute, de l'évaluation ?), recherche commune d'une solution, etc.

C Implications au niveau des ressources.

La possibilité d'implanter un système LOGO dépendra évidemment des ressources disponibles; rappelons que le matériel suffisant consite soit en une tortue de sol (environ (50.000 Frs),soit en une tortue d'écran (le langage LOGO est actuellement disponible sur une grande majorité de micro-ordinateur (Apple, Commodore, Sharp, BBC, IBM,Thomson, etc.)

Avec une organisation spécifique (Cf. ci-dessous) on peut déjà commencer à travailler, avec une classe, avec un seul micro-ordinateur (unité centrale - 1 lecteur de disque - une imprimante - un video - le langage LOGO :soit, au minimum, environ 45.000 Frs)

d. Implications au niveau de l'organisation de la classe

L'ordinateur doit s'intégrer au maximum à la vie de classe : il doit être considéré comme un outil supplémentaire disponible; les activités réalisées en LOGO doivent être en relation avec le centre d'intérêt du moment, d'autres activités peuvent s'harmoniser avec le travail sur micro-ordinateur à propos du même thème "par exemple, les manipulations de la tortue de sol peuvent faire partie d'une série d'activités concernant la notion de direction et de carte, on y trouverait en outre des exercices de lecture de carte ou de construction de cartes, de labyrinthe ou de rose des vents ainsi que des exercices de mesure et de calcul de distance.(1)

De nombreuses expériences ont montré que les interactions dans un groupe d'enfants travaillant en LOGO sont riches et nombreuses. Il s'agit d'ailleurs, d'après les enseignants qui l'ont utilisé dans leur classe, d'un aspect fondamental du LOGO.

La taille du groupe est un problème important. En effet, POTTER (2) a montrer que les interactions centrées sur la tâche diminuaient dans des groupes supérieurs à quatre élèves. Au delà de ce nombre, les élèves qui se situent aux extrémités se retirent de la discussion. Des groupes plus petits (2 ou 3) ont un taux d'interactions plus faible.

L'organisation de la classe se basera donc sur une rotation de petits groupes,l'ordinateur étant un des supports de travail diponibles.Dans tous les cas,la solution d'un "local informatique"est à rejeter.

(1) MARTIN A., p. 24

(1) POTTER F.N., Classroom organisation and group - discussion : the role of the microcomputer, the role of the teacher, Université d'Ete, C.E.E., Liège, 1985.

En guise de conclusion, il nous semble important de rassembler quelques réflexions critiques par rapport au système LOGO. En effet, les discours sur le LOGO sont, en général, triomphalistes et manquent parfois de nuance; ainsi il n'est pas rare de se voir présenter le LOGO comme une "révolution dans l'enseignement" ou encore une "clinique intellectuelle". L'utilisation du LOGO par les enfants est présentée par certains auteurs comme naturelle, spontanée et non soumise à l'effet de la fatigue.

Nous avons déjà parlé du problème du *transfert* qui est loin d'être résolu.
Nous avons également bien montré comment le LOGO est lié à un certain type d'apprentissage - apprentissage par la découverte - essai et erreurs - et l'on sait bien maintenant qu'il existe plusieurs manières d'apprendre, le LOGO risque donc, comme toute démarche pédagogique, de *ne pas convenir à certains apprenants.*

On constate d'ailleurs que tous les enfants ne sont pas tous *spontanément* intéressés par le LOGO. A titre d'exemple, citons les conclusions d'une expérience d'utilisation du LOGO à l'occasion de stage d'insertion sociale avec des adolescents de 16 à 18 ans :

"La première constatation que cette intervention nous amène à faire est triviale : l'ordinateur, ce n'est pas si simple !

Face aux délires technocratiques qui ont cours sur la micro-informatique, nous avons pu voir qu'il ne suffit pas - quoiqu'en disent les augures - de placer un individu devant un clavier pour qu'il s'approprie l'outil, même avec un "gentil-éducateur".

Les difficultés rencontrées par les stagiaires pour mettre en oeuvre les fonctions les plus simples, les plus directes, et en conséquence, l'absence de réalisations motivantes au bout de plusieurs séances, posent sur un plan cette fois méthodologique, la question de l'adaptation de l'outil micro-informatique aux personnes en recherche d'insertion ...

... Un autre questionnement a surgi également qui devrait ébranler sérieusement les tenants de la pédagogie par le jeu; dans ce monde d'adolescents en butte à l'échec scolaire, puis aux difficultés de l'entrée dans le monde du travail, les réactions sont fréquentes du genre : "Le dessin, c'est bon pour les enfants", ou "On n'est pas là pour s'amuser, on est là pour trouver du boulot !" "(1)

De plus, on a constaté à de nombreuses reprises que pour certains enfants une période de *"stagnation"* suit l'enthousiasme du début : après avoir découvert et maîtriser les primitives de base, soit ils définissent à nouveau des projets semblables qui ne représentent pour eux plus aucun défi ,soit ils éprouvent des difficultés à trouver de nouveaux projets et se tournent vers l'animateur.

Enfin si LOGO se révèle, de fait, être un outil puissant d'apprentissage de méthodes de résolution de problèmes, il faut bien remarquer qu'il s'agit toujours d'une classe de problèmes bien spécifiques.

"Dans ces bénéfices, rien qui ne soit de l'ordre de la formalisation logique avec comme clé de voûte, évidemment les mathématiques. Même lorsque S. Papert propose de tenir compte, à la différence de Piaget, de l'affect en matière de processus d'apprentissage, il s'agit toujours de l'utiliser pour le mettre au service de la pensée procédurale. Son souci majeur me paraît pouvoir être formulé ainsi : comment capter, mobiliser l'investissement affectif et créatif pour en irriguer les plus beaux fleurons de la création technologique que sont les ordinateurs ? Et au bout du compte, en obtenir des changements libératoires en matière de processus d'apprentissage, de rigueur intellectuelle (voir l'importance qu'il attribue aux mathématiques) et de mutations institutionnelles. Mais ces effets espérés s'inscrivent toujours dans les phantasmes d'une maîtrise accentuée d'outils techniques complexes et surtout, des processus intellectuels de leurs mises en oeuvre. Programmation des ordinateurs par et pour la transparence des chemins de la connaissance : voilà l'objectif. Dans ce cadre, la performance, la logique formelle, la rationalité technique règnent en maîtres. Ainsi ce que S. Papert désigne par pensée abstraite n'est le plus souvent qu'une catégorie intimement liée à une intelligence pratique-technique des modes de fonctionnement machiniques.

(1) CHAPPAZ G., LOGO chez les ados, Education Permanente, 70-71, 1983 ,(p. 139-150)

Selon un modèle très au goût du jour, la complexité de l'abstraction doit être traitée par la concrétisation, la manipulation d'objects et de procédures. Ainsi lorsqu'un utilisateur se propose de dessiner une fleur avec le langage LOGO (ou un autre), il se situe toujours dans un registre de concrétisations de capacités assez délimité : celui de l'activité rationnelle par rapport à une fin, de la gestion univoque des moyens, d'une logique non ambiguë, etc." (1)

WEISSBERG J.L. La maladie infantile de l'informatique,Education Permanente,70-71,1983 (pp 17-26).

BIBLIOGRAPHIE

ABELSON, H.; Le LOGO sur APPLE, Paris, Nathan, 1983.
Manuel d'utilisation du LOGO sur APLLE.

BOSSUET, G.; L'ordinateur à l'école, Paris, P.U.F., 1982.
Compte-rendu des expériences françaises de LOGO avec des enfants d'écoles primaires.
Qu'est-ce que " l'univers Logo " ?: un langage, une théorie de l'apprentissage, un matériel informatique, etc ... Comment introduire des activités dans une classe ?

BOSSUET, G.; Où en est LOGO, GREPACIFIC,mai 1984.

HARDY, J.L.; LOGO : un système informatique évalué au service d'une pédagogie d'une pédagogie évaluée, U.L.G., L.P.E., 1982.
Analyse des caractéristiques du langage " LOGO ".
Propose différents types d'utilisations et différents types de contenus liés à ces utilisations (géométrie, physique, cybernétique).

MEYNARD, F.: Enquête sur LOGO, projet d'utilisation de l'ordinateur en Pédagogie, Montréal, Ministère de l'Education, Direction de la Technologie Educative, 1975.
Compte rendu d'une expérience d'implentation d'un "laboratoire LOGO" dans un collège.

MYX, A.; et SUBTIL, P.. LOGO votre langage de programmation, Paris, Cedic Nathan, 1985.

MYX, A.; Plus loin avec LOGO, Paris, Cedic-Nathan, 1984.
Excellente introduction au traitement de liste en LOGO. Le livre explore les nombreuses possibilités de cette ensemble moins bien connu du LOGO et propose quelques programmes déjà complexes.

PAPERT, S.; Jaillissement de l'esprit, Paris, Flammarion, 1981.
L'inventeur du LOGO en donne la philosophie générale.

REGGINI, H.C.; LOGO, des ailes pour l'esprit, Paris, F. nathan, 1983.
Loin des discours abstraits, cet ouvrage présente le langage LOGO à travers de nombreux exemples de programmes.
L'auteur présente les objectifs généraux d'activités LOGO au primaire : objectifs cognitifs, affectifs et sociaux.

SIMON, J.C.; L'éducation et l'informatisation de la société, annaexe 1, Paris, La Documentation Française, 1980.
Une partie de cette annexe est consacrée à " l'apprentissage autonome ": quels types d'apprentissages les enfants réalisent-ils dans un environnement LOGO ?

TAYLOR, R.T.; (editor) The computer in the school : tutor, tool, tutee, N.Y, Teachers College Press, 1980.
Le livre retrace différents types d'applications pédagogiques de l'ordinateur. Les articles proposés tournent autour de 3 thèmes : l'ordinateur comme "tuteur", "outil" ou "élève". Dans ce dernier cas, l'objectif est d'apprendre quelque chose à l'ordinateur. Cette dernière partie est présentée par S. PAPERT.

WEIDENFELD, G.; MATHIEU, F.; PEROLAT, Y.D.; LOGO, Paris, Eyrolles, 1984.
Introduction complète et illustrée de nombreux exemples au langage LOGO.
L'ouvrage aborde de plus, les possibilités pédagogiques du langage par le compte-rendu d'une expérience scolaire.

6

FORMATION EN EDUCATION N°1.12

RECHERCHE

CENTRE O.S.E.

Introduction aux logiciels auteur

Marc Romainville

DEPARTEMENT
EDUCATION & TECHNOLOGIE

Facultés Universitaires Notre-Dame
de la Paix B-5000 Namur

R. MOREL
oct. 87
3043 ✓

Mon objectif est de proposer quelques pistes de réflexion permettant de mieux comprendre l'origine des logiciels auteur, leurs fonctions, leurs avantages et leurs limites.

I ORIGINE

Pourquoi à un moment donné de l'histoire (courte) de l'informatique certaines personnes ont-elles imaginé de construire ces outils ?

Rappelons que fondamentalement l'ordinateur transforme les informations en signaux simples (les octets : combinaisons de 8 signaux positif ou négatif (oui/non) dans des ordres divers). Dialoguer avec la machine à ce niveau n'est pas une sinécure; c'est pour cette raison que sont apparus les langages informatiques. Ceux-ci permettent de commander l'exécutant en terme d'opérations élémentaires, d'instructions d'action (lire au clavier, afficher, etc.). Ils se sont d'ailleurs différenciés en fonction de leur objectif.

Ex. : . FORTRAN : calcul scientifique
. COBOL : gestion
. BASIC : accessible aux débutants
. ...

Dans les années 60 donc, l'enseignant qui se lançait dans la construction d'une séquence d'E.A.O. - c'est-à-dire commander l'exécutant dans un but de formation - ne disposait que de ces langages informatiques classiques.

Pour comprendre la naissance des logiciels auteur, analysons les obstacles qui s'opposaient alors au développement de l'E.A.O. :

soit une partie de programme réalisé en BASIC; il s'agit d'une sous-routine vérifiant la réponse de l'élève :

```
1000    REM    Sous-routine de vérification de la réponse
1010    L = 0
1020    INPUT  A$
1030    IF A$ = "réponse correcte" THEN 1080
1040    L = L + 1
1050    IF L < 3 THEN PRINT "Essaie encore" : GOTO 1020
1060    PRINT CHR$ (4); OPEN "fichier remédiation"
1070    FOR I = 1 TO fin : READ "fichier de remédiation" : NEXT I
1080    REM Fin de la boucle réponse : GOTO question suivante
```

... ou en LOGO pour, au passage, réaffirmer qu'il s'agit aussi d'un langage de programmation :

```
Pour VERIF. REPONSE
DONNE "L 0
REPONSE
FIN

        POUR REPONSE
        DONNE  "réponse PR LL
        SI : réponse = "réponse-correcte [Question suivante]
        DONNE "L  L + 1
        SI L > 3 [remédiation]
        EC [Essaie encore]
        REPONSE
        FIN
```

Les objectifs de l'enseignant sont simples et compréhensibles : vérifier si la réponse est correcte, laisser deux essais à l'apprenant puis lui proposer une remédiation. Cependant, pour exprimer ces consignes élémentaires à un exécutant borné, il doit avoir recours à un langage ésotérique, complexe, à la syntaxe chatouilleuse; il aboutira ainsi à 10 lignes d'ordres qui pour les non-initiés ressemblerait plus à une formule magique qu'à une explicitation d'une démarche pédagogique bien élémentaire.

Quelles sont les inconvénients d'une telle programmation ?

1. Les ordres donnés à l'exécutant le sont en terme d'<u>opérations logiques propres au fonctionnement de la machine ou au langage</u>.

   Ex. : `L = 0` remise à zéro du compteur
   `PR LL` prendre le premier élément de la liste introduite au clavier

   Les enseignants quant à eux conçoivent leurs leçons en terme pédagogique, c'est-à-dire en terme de moyen à utiliser pour faire apprendre telle notion, pour réviser telle autre, etc.

   Ex. : lui dire ceci
   vérifier s'il comprend bien
   lui donner 3 essais
   éviter telle mauvaise interprétation
   ...

Les langages et systèmes auteur ont tenté de remédier à ce problème en permettant à l'enseignant de communiquer sa compétence au sujet des méthodes d'enseignement en terme de stratégies, d'opérations élémentaires de l'acte pédagogique; les logiciels se chargent eux-mêmes de la conversion en logique d'ordinateur.

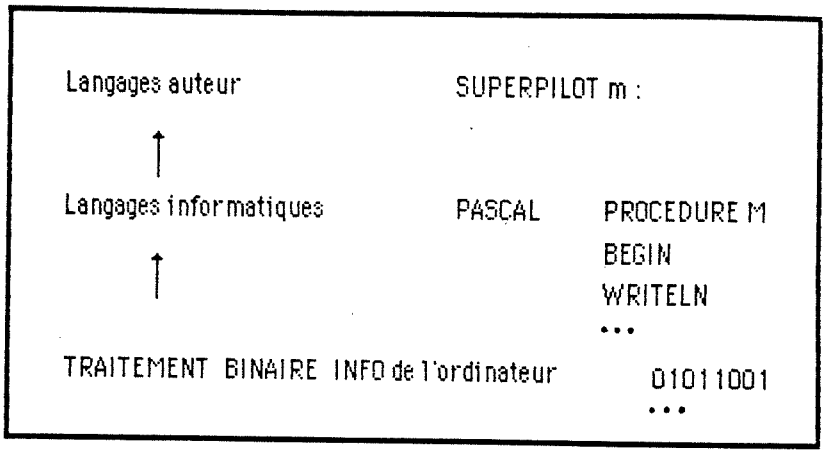
2. Dans cette optique, les enseignants doivent s'initier aux méthodes de la programmation et dans la suite à un langage particulier. Il faut compter un minimum de 120 heures de formation. Celle-ci prend donc du temps et se révèle par ailleurs insuffisante : seule une longue pratique sur des problèmes divers permet d'atteindre une maîtrise efficace du langage choisi. De plus, l'adaptation de tels programmes n'est également possible qu'à des enseignants maîtrisant la programmation.

   Deux solutions ont été apportées à ce problème par les logiciels auteur :

   - la syntaxe des langages auteur a été réduite et simplifiée. Le nombre d'instructions a par ailleurs été réduit. Un gain de temps peut donc être réalisé dans l'apprentissage du langage. Une initiation aux principes de la programmation structurée est cependant encore nécessaire.

   - les systèmes auteur ne supposent plus aucune connaissance informatique de la part de l'utilisateur, du moins en ce qui concerne la programmation; nous en reparlerons plus loin.

3. Il faut compter 300 à 500 heures de travail pour mettre au point 1 heure d'E.A.O.. Le coût est donc très élevé. Dans certains cas, cela remet en cause l'efficacité de ce nouveau médium. La simplification de la syntaxe et l'utilisation de fonctions pré-programmées pour les langages auteur et l'utilisation de menus pour les systèmes auteur permettent de gagner du temps pour la mise au point d'un programme. Les instructions du SUPERPILOT, par exemple, sont en fait des petites procédures PASCAL. L'utilisation récupère donc en quelque sorte le temps qu'il a fallu aux concepteurs du SUPERPILOT pour élaborer et réaliser ces procédures spécifiques aux problèmes d'enseignement.



4. Avec un langage classique, l'enseignant passe un temps considérable au niveau du "faire faire"; c'est-à-dire de la procédure : comment faire pour que la machine fasse ceci ou cela.
Les logiciels auteur ont tenté de dépasser cette procéduralité dans la description des didacticiels; l'enseignant ne s'occupe plus de savoir comment la machine fera ce qu'elle doit faire mais lui dit - ou mieux choisit par un système de menu - ce qu'elle doit faire.
Cette distinction permettrait d'éviter le phénomène de réduction qui consiste à renoncer à de nombreuses ambitions pédagogiques devant les difficultés de réalisation propres au langage.

Mais comment réalise-t-on une séquence à l'aide d'un logiciel auteur ?
Revenons au programme de la page 2. Celui-ci s'écrira en SUPERPILOT (langage auteur) de la manière suivante :

a :	(pour accepter la réponse de l'élève)
m : réponse correcte	(pour rechercher la réponse correcte)
jn3 : remédiation	(si elle ne s'y trouve pas après 3 essais, l'envoyer vers une remédiation)
jn:@ a	(boucle vers le dernier a:)
* question suivante	(passe à la question suivante si la réponse est correcte)

Enfin, simulons le dialogue qui nous permettrait sur un système auteur (EURIDIS) de construire cette séquence :

Les données introduites par l'utilisateur sont en caractères gras :

Les données introduites par l'utilisateur sont en caractères gras :

```
----------------------
        CREER
----------------------
Indiquer le n° du bloc à créer 1
```

```
      réponse correcte
--------------------
BLOC 1 * Réponse.type n° 1

              ENTREZ LE TEXTE
```

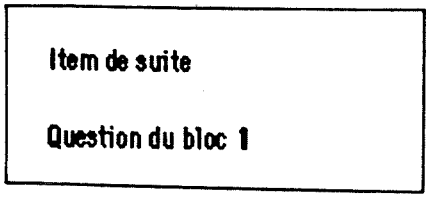
```
      <RETURN>

BLOC 1 * COMMENTAIRE N° 1
```

aucun commentaire

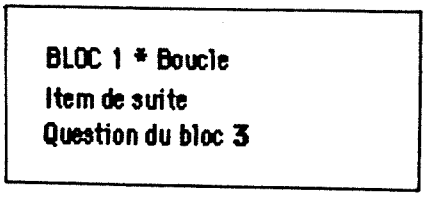
```
Item de suite
Question du bloc n° 2
```

au bloc 2, la question suivante est posée

```
      <RETURN>

BLOC 1 * Commentaire Réponse Imprévue
```



le renvoit à la même question



Au bloc 3 se trouve la séquence de remédiation

II DEFINITION ET CLASSIFICATION

Les logiciels auteur sont donc des outils informatiques spécialisés dans l'écriture de programmes éducatifs; ils ont pour but de faciliter l'utilisation de l'informatique dans l'enseignement et sont des moyens de commander l'ordinateur afin d'organiser son fonctionnement de telle sorte qu'il puisse servir à la formation.

Une hiérarchie peut être établie dans ces logiciels allant des plus lourds à utiliser aux plus aisés (mais aussi des plus flexibles aux plus contraignants).

Les langages auteur sont des langages de programmation, disposant cependant d'un nombre d'instructions limité et d'une syntaxe plus simple, qui au moyen de fonctions proches de la pédagogie permet au concepteur d'écrire plus facilement des didacticiels.

Les systèmes auteur ne sont pas des langages de programmation mais des programmes permettant au concepteur de réaliser des didacticiels en lui proposant une trame à l'intérieur de laquelle il insère ses données : textes, questions, branchements, etc. Aucune connaissance informatique n'est alors requise, ils sont en effet conversationnels : toutes les phases de création sont présentées en clair au moyen de menus, question, bordereau de création; l'enseignant n'ayant qu'à choisir, à répondre ou à compléter la trame.

La différence entre langage et système auteur peut être illustrée par l'exemple suivant.

Il s'agit d'indiquer à l'ordinateur de réaliser un "délai d'une seconde".

a. langage de programmation classique

en Basic : 10 FOR N = 0 TO 1000 : NEXT N

en Pascal (1) :

```
PROCEDURE ATTENDRE (T : INTEGRER) :
    VAR  X : intégrer;
    Begin
    FOR X = 1 TO * 1000 do;
    end;
```

b. langage auteur en Superpilot : DELAY : 1

c. système auteur avec DIGIT-DACT = Q : "quel délai désirez-vous ?"
R : 1

Citons encore deux catégories bien que nous n'en parlerons guère dans ce document centré sur les langages et systèmes auteur.

Les logiciels générateurs d'exercices, de test :

Ces logiciels génèrent et souvent impriment des exercices, des feuilles de travail, des devoirs à réaliser à domicile, des interrogations, etc.

PEDAGO : Ce logiciel de création d'exercices possède un champ d'application assez large. Les exercices sont générés aléatoirement parmi les données proposées par l'enseignant.

---

(1) Ce point de vue, l'avantage d'un langage procédural est de pouvoir définir soi-même une procédure qui exécutera un délai voulu; aussi pour la réaliser en LOGO en seconde il suffit de se construire la procédure PAUSE

```
POUR PAUSE : sec
ATTENDS 60 X : sec
```

Celle-ci peut alors être appelée dans d'autres procédures - éventuellement pour d'autres valeurs - et le travail de codage n'a été réalisé qu'une seule fois.

Supposons la construction d'un exercice à partir du schéma suivant : une personne part faire des courses avec une certaine somme d'argent. Elle achète un ensemble de produits et revient avec la somme restante. Voilà le scénario d'un exercice sur les 4 opérations, la question à poser est donc la suivante : "Avec combien d'argent la personne revient-elle de course ?" La première phase consiste à noter sur papier les paramètres du problème :

NOMS DES NOTIONS (choisi par le formateur)	**VALEUR POSSIBLES**
PERSO	maman, tante Jeanne, tante Lise, grand-mère
FRUIT	mandarines, organes, pommes, cerises
KILFR (nombre de kilo de fruits)	2, 3, ..., 10
PRIFR (prix au kilo des fruits)	8, 9, 10, ..., 13
EPICERIE	sucre, café, sel, farine
PRIEP (prix du paquet d'épicerie)	5, 6, 7, ..., 10
PAQEP (nombre de paquets d'épicerie)	2, 3, 4, ..., 8
POMON (valeur dans le porte-monnaie)	120, 121, ..., 250
PXTFR (prix total des fruits)	$ PXTFR $ = $ KLFR $ X $ PRIFR $
PXTEP (prix total épicerie)	$ PXTER $ = $ PAQEP $ X $ PRIEP $
DEPTO (dépense totale)	$ DEPTO $ = $ PXTFR $ + $ PXTEP $
RESUL (résultat final)	$ RESUL $ = $ POMON $ - $ DEPTO $

L'enseignant introduit ensuite les données en utilisant l'écran de dialogue suivant :

Grille de saisie des NOTIONS :
NOTIONS : Avez-vous d'autres notions à introduire ? O/N
En cas de réponse positive (O pour OUI) le formateur remplit alors la grille suivante :
nom :
numérique entier : ? numérique réel : ? alpha-numérique : ?
intrinsèque : ? calculée : ?
(indépendante d'autres notions)
fixe : ? aléatoire : ?
(dont la valeur change à chaque exécution)

En cas de notion calculée
notions intervenant pour le calcul ou la détermination : ?
formule ou graphe : ?
Si formule : introduction de la formule
Si graphe : introduction du graphe définissant la notion en cours de saisie
En cas de notion intrinsèque aléatoire numérique
plage de valeurs : ? valeurs exclues : ?
En cas de notion intrinsèque aléatoire alpha-numérique
liste de valeurs : ?
(parmi lesquelles la machine choisit aléatoirement une valeur à chaque exécution)

Puis introduit le texte du problème ($ PERSO $ va au marché ...) et de la question.

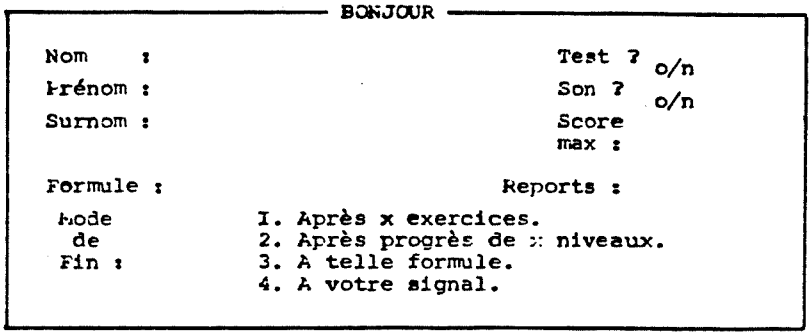
CALCUL FRACTIONNAIRE : Produit par le CEFIS, ce logiciel permet de créer une grande variété d'exercices sur un domaine particulier : la simplification, la comparaison et opérations arithmétiques sur les fractions. L'utilisateur choisit le nombre d'exercices, le type, le niveau de difficulté, la grandeur des nombres manipulés. L'ordinateur génère, dans les conditions prédéfinies, les exercices au hasard et fournit un solutionnaire.

Les logiciels ouverts : la coquille du programme existe mais l'utilisateur doit fournir, en début d'exécution, les données, les informations sur lesquelles l'étudiant va travailler. Il s'agit le plus souvent de leçons de drill. Le carcan pédagogique est évidemment important.

Citons à titre d'exemple le logiciel suivant :

ADDITION : Il permet à l'enseignant de créer des exercices sur l'addition en fonction des problèmes et du niveau de chaque élève.

Avant de faire travailler un élève sur le logiciel, l'enseignant est invité à préciser un certain nombre de renseignements visant à adapter les exercices.
L'écran suivant lui est présenté :



Ex. : FORMULE : si l'enseignant introduit la formule U + U, les exercices d'addition se réaliseront uniquement sur des unités avec report à la dizaine s'il choisit d dans Report.

formule : U + U report d ---> 6 + 7
8 + 3
9 + 2
...

formule cdu + du report cd --->143 + 17
137 + 28
...

Etc.

Nous ne nous attarderons pas sur les deux dernières catégories car il s'agit de logiciels très spécifiques, assez peu flexibles et qui ne permettent pas la création d'une séquence pédagogique complète.

III QUE PEUT-ON ATTENDRE D'UN LOGICIEL AUTEUR ?

3.1. Les langages auteur

Dans une étude sur les langages auteur, des chercheurs ont mis au point la grille ci-dessous; elle permet de comparer et de classer différents logiciels auteur en observant comment chacun d'eux s'acquitte des différentes tâches proposées: pour cela une leçon type (8 écrans) a été construite à partir de ces critères. On demande à des observateurs neutres de l'implémenter à l'aide des différents logiciels : on mesure également le temps de réalisation pour juger de leur efficacité. [voir GILLINGHAM]

1. Capacités Texte

   1.1. Afficher du texte
   1.2. Minuscules et majuscules
   1.3. Effacement du texte
   1.4. Effacement du texte en-dessous d'un graphique
   1.5. Mise en évidence d'une partie de texte
   1.6. Possibilité surbrillance

2. Capacités Graphique et Son

   2.1. Mise en couleur du texte
   2.2. Texte dans le graphique
   2.3. Emission de son
   2.4. Dessiner un cercle
   2.5. Le diviser
   2.6. Mettre en évidence des fractions
   2.7. Programmer touches pour déplacer le curseur
   2.8. Afficher 3 caractères standard en graphique

3. Capacités concernant le jeu de caractères

   3.1. Afficher
   3.2. Dessiner les lettres
   3.3. Dessiner les flèches
   3.4. Afficher un exposant
   3.5. Afficher une racine carrée

3.6. Afficher une note de musique
3.7. Afficher le signe intégral
3.8. Afficher le signe inégalité
3.9. Afficher la clef de sol

4. Capacités d'animation

4.1. Mouvement d'un graphique, d'un caractère
4.2. Positionner le curseur
4.3. Connaître la position du curseur
4.4. Faire effectuer une rotation à une figure
4.5. Faire effectuer un déplacement
4.6. Augmenter ou diminuer la taille d'une figure

5. Capacités de calcul et d'affectation

5.1. Utilisation d'une variable chaîne de caractères (stockage et sortie)
5.2. Utilisation d'un tableau (idem)
5.3. Générateur de nombre au hasard
5.4. Générateur de nombre au hasard dans un intervalle (ex. : - 3 + 3)
5.5. Calcul d'une fonction
5.6. Afficher une variable chaîne de caractères tirée au hasard d'un tableau

6. Capacités de gestion

6.1. Branchement
6.2. Contrôle du temps entre deux entrées
6.3. Message d'attente entre deux écrans
6.4. Compteur du nombre d'essais
6.5. Différencier les réponses pour des branchements différents
6.6. Pause
6.7. Analyse de la réponse
6.8. Feed-back adapté en fonction de la réponse et d'un compteur
6.9. Enregistrement des réponses dans un fichier

3. 2. Les systèmes auteur

Généralement, ceux-ci sont composés de 4 parties : passons en revue les qualités attendues de chacune de ces parties.

MODE AUTEUR : On devrait y trouver un éditeur de texte avec les facilités d'insertion, d'effacement, de centrage, de manipulation des attributs du texte (couleur, vidéoinverse). L'éditeur graphique doit permettre de créer facilement des affichages graphiques. L'animation est souhaitable. Enfin, l'éditeur de dialogue permet d'agencer les différentes séquences ainsi mises au point. Et cela en fonction des réponses de l'élève, de ses choix, de son taux de réussite, etc.

MODE ELEVE : La convivialité sera améliorée par l'existence d'un certain nombre de fonctions telles que : possibilités de défilement arrière, de suspension/reprise du travail, d'accès à une calculatrice, à des aides, etc.

MODE GESTION DES APPRENTISSAGES :
Il doit être possible d'enregistrer les données d'un élève dans un dossier personnalisé et de les adapter au fur et à mesure de sa progression.

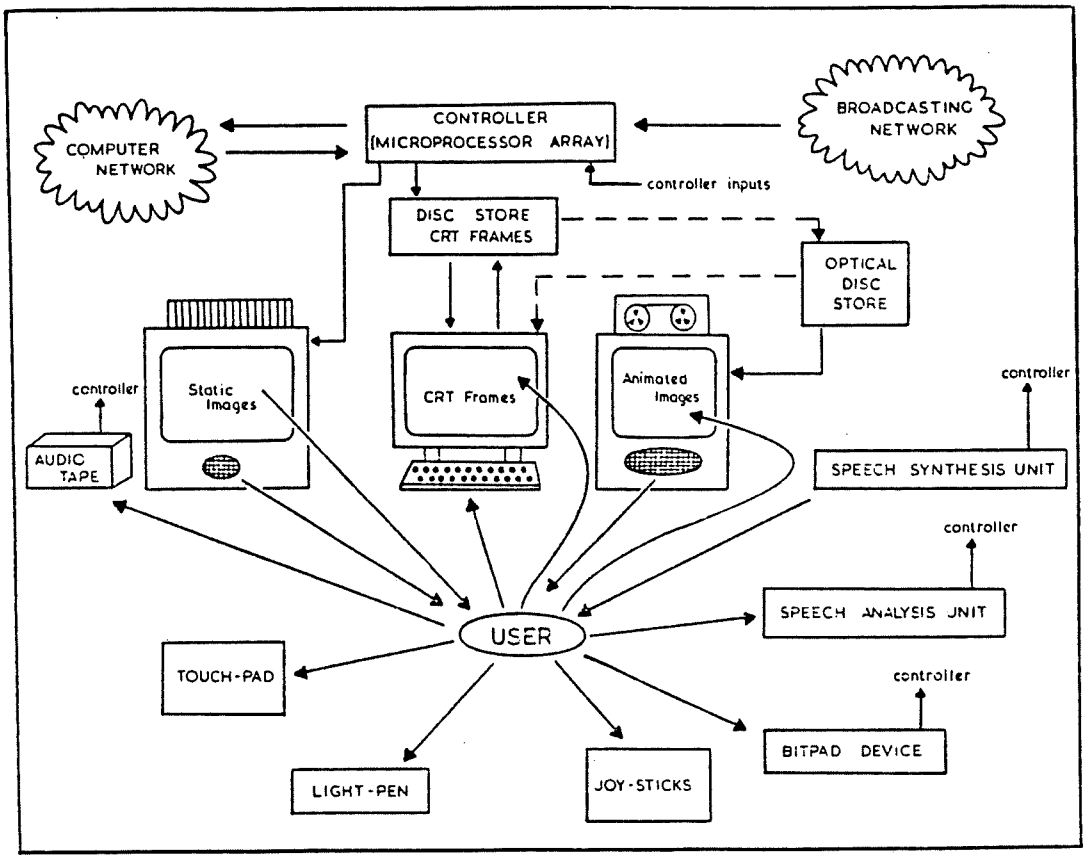
UTILITAIRES : Tels que l'installation automatique pour l'imprimante, etc.

Notons également qu'il est souhaitable que le logiciel auteur dispose de commandes permettant des communications avec divers périphériques.

SUPERPILOT : Il existe actuellement un interface permettant de piloter un magnétoscope ou un vidéodisque à partir de ce langage auteur. Des commandes supplémentaires permettent de

- chercher un passage — V : find 1234
- lire le segment trouvé — V : plyb 5678
- lire uniquement le son — V : phya 5678
- lire uniquement l'image — V : phyv 5678

NUMEDALA : Ce <u>langage auteur</u> a été expressément conçu dans le but de contrôler un environnement interactif d'apprentissage sophistiqué. Les communications prévues sont schématisées dans la figure suivante :



EGO : Ce <u>système auteur</u> possède un module de pilotage de périphériques audio-visuels; il est ainsi capable de gérer un projecteur dia, un magnétoscope ("taper" la bande, rechercher un enregistrement, le présenter), un vidéodisque (positionner le bras de lecture sur une vue, regarder image par image, appeler des séquences animées, avec ou sans son, à vitesse accélérée ou au ralenti, en avant ou en marche arrière.)

Certains logiciels auteur permettent également de faire appel à des programmes rédigés dans des langages de programmation classiques; des procédures PASCAL peuvent ainsi être appellées à partir du SUPERPILOT, le système EURIDIS permet d'établir des connexions avec des programmes LSE, Pascal ou Basic.

Enfin dans les prochaines années des améliorations importantes pourraient être apportées à ces logiciels dans les domaines suivants :

1. L'analyse de réponse par l'utilisation de systèmes d'intelligence artificielle.

2. L'élargissement de l'éventail des canevas pédagogiques. Des logiciels auteur spécialisés dans l'élaboration d'environnement de simulation pourraient ainsi voir le jour.

3. La disposition d'écran avec l'apparition des zones de travail multi-fenêtres, multi-fonctions.

IV UN POINT DE VUE CIRITIQUE SUR LES LOGICIELS AUTEUR

Un certain nombre de critiques ont été émises à l'encontre des logiciels auteur; de plus, il faut bien constater que peu de didacticiels mis sur le marché sont écrits en langage ou en système auteur. Devant ces faits, certains prétendent que ce que l'on présentait comme une solution par excellence au problème de l'EAO a fait long feu et que ces logiciels auteur sont amenés à disparaître. Mais, que leur reproche-t-on au juste ?

1. Contrairement à ce qui est annoncé, un logiciel auteur ne s'apprend pas sans difficultés.
   L'apprentissage d'un langage auteur exige une initiation à la méthode algorithmique, à la syntaxe et aux spécificités du système de codage. Un nombre impressionant de commandes doivent être mémorisées; le langage SUPERPILOT par exemple dispose, si l'on prend en compte les commandes des différents éditeurs, de bien plus d'instructions que le BASIC.
   De plus, on découvre actuellement que même l'utilisation d'un système auteur suppose une certaine alphabétisation informatique de base de la part du concepteur. Les notions de programme, enregistrement, variable, etc. doivent notamment lui être familières pour comprendre comment le système auteur fonctionne (ex. : dans PEDAGO p. 7 : le nom des notions et les valeurs possibles représentent l'étiquette et le contenu de la variable).
   Leur durée d'apprentissage peut également être très élevé (cf. EURIDIS, IMG, etc.).

2. La programmation à l'aide d'un langage auteur se révèle parfois moins efficace. Elle est moins économique dans son utilisation de la machine (ex. : lenteur d'exécution et travail incessant sur les disques en SUPERPILOT) et favorise plus une programmation linéaire que structurée.

3. Il reste - et dans ce sens l'apparition des logiciels auteur ne peut être une solution à l'EAO - que la réussite d'un didacticiel dépend au moins à 90 % de la phase de conception (scénario pédagogique, etc.) et seulement à 10 % de la phase de codage. Il convient donc de resituer à leur juste place ces outils nouveaux de codage; comme l'indique POLLOCK, les logiciels auteur ne sont pas la panacée qui nous sortirait de l'indigence actuelle des didacticiels; la création facile d'un mauvais programme n'est pas un progrès.

4. Certains prétendent également que la facilité d'utilisation du système de codage amène les concepteurs à fondre les deux phases décrites ci-dessus. On constaterait ainsi que le travail aux machines se fait beaucoup plus rapidement - et parfois même dès le début - avec les logiciels auteur, la phase d'analyse du problème sur papier disparaissant progressivement.
   Cette fusion provoque un rétrécissement continuel de l'idée de départ en fonction des possibilités du système sans que l'utilisateur ne se rende plus compte de cette influence de la technique sur la pédagogie. Ces deux phases devraient donc être dissociées, voire même réalisées par des personnes différentes.

5. Le logiciel auteur risque d'enfermer le concepteur dans un paradigme pédagogique; ces logiciels sont en effet construits pour aider l'enseignant à élaborer certains types de séquences pédagogiques, certains choix méthodologiques sont donc réalisés au départ; le carcan varie évidemment suivant la flexibilité du logiciel mais il reste qu'ils fonctionnent dans les limites de leur programme; chacun d'eux n'exploitant qu'un certain nombre d'activités d'enseignement seulement.

ANNEXE 1

GRILLE D'EVALUATION DE LOGICIEL AUTEUR
(d'après GILLINGHAM)

TITRE :
AUTEUR :
EDITEUR :
COUT :
LANGAGE :

A. LE LOGICIEL

	OUI	NON
1. Présence d'une documentation ?	...	...
Présence d'une séquence E.A.O. ?	...	...
Présence d'exercices ?	...	...
Possibilité de formation par la firme productrice ?	...	...
2. L'auteur peut-il accéder à		
carrousel dia ?	...	...
magnétoscope ?	...	...
vidéodisque ?	...	...
synthétiseur de voix ?	...	...
procédures écrites dans d'autres langages évolués ?	...	...
manettes de jeux ?	...	...
écran tactile ?	...	...
crayon optique ?	...	...
table graphique ?	...	...
souris ?	...	...
calculatrice ?	...	...
3. Peut-on stocker des données ?	...	...
4. Existe-t-il des aides à l'intérieur du logiciel ?	...	...
5. Les possibilités de présentation de texte sont-elles présentes		
- affichage ?	...	...
- modification des jeux de caractères ?	...	...
- modification des styles de caractères ?	...	...
6. Ces possibilités de présentation graphique sont-elles présentes		
- jeux de caractères ? combien ?	...	...

- couleur ? ... ...
- animation ? ... ...
- modification caractères (taille, couleur) ? ... ...
- haute résolution ? ... ...
- basse résolution ? ... ...
- édition de figures ? ... ...
- édition de graphes ? ... ...
- lutins ? ... ...
- rotation ? ... ...

7. Dans l'éditeur de leçons, y a-t-il
   - une possibilité d'exécution immédiate ? ... ...
   - des possibilités des traitements de texte (chercher, remplacer, copier, etc.) ... ...

8. Dans l'éditeur de questions, peut-on
   - formuler ces types de questions
     - QCM ? ... ...
     - lacunaires ? ... ...
     - appariement ? ... ...
     - question ouverte ? ... ...
     - classement ? ... ...
     - sélection ? ... ...
   - tirer au hasard le nombre de question ? ... ...
   - tirer au hasard le type de question ? ... ...
   - pondérer les questions pour le calcul du score ? ... ...

9. Dans la gestion des apprentissages, que fournit le logiciel
   - les scores ? ... ...
   - un classement des élèves ? ... ...
   - le nombre d'erreurs ? ... ...
   - le pourcentage d'erreurs ? ... ...
   - le nombre d'essais par item ? ... ...

10. L'analyse prévoit-elle
    - les corrections de fautes d'orthographe ? ... ...
    - les transformations minuscules/majuscules ? ... ...
    - la reconnaissance de synonymes ? ... ...

11. Le branchement peut-il être
    - direct ? ... ...
    - linéaire ? ... ...
    - dynamique ? ... ...
    - sous forme de boucle ? ... ...

B. LA CREATION DE SEQUENCE

1. Le logiciel est-il liée à une fonction particulière ?

	OUI	NON
drill	...	...
tutoriel	...	...
simulation	...	...
jeux	...	...
autres	...	...

2. Indiquez la facilité d'utilisation pour les points suivants

	Très facile	Facile	Difficile	Très difficile	Pas applicable
Utilisation de touches fonctions	1	2	3	4	5
Effacement de portions d'écran	1	2	3	4	5
Texte et graphique sur le même écran	1	2	3	4	5
Mouvement du texte sur écran	1	2	3	4	5
Modification du style des caractères	1	2	3	4	5
Modification du jeu de caractères	1	2	3	4	5
Soulignement	1	2	3	4	5
Couleurs	1	2	3	4	5
Figure en trois dimensions	1	2	3	4	5
Rotation de figures	1	2	3	4	5
Animation	1	2	3	4	5
Graphique haute résolution	1	2	3	4	5
Graphique basse résolution	1	2	3	4	5
Utilisation des lutins	1	2	3	4	5
Fichiers graphiques	1	2	3	4	5
Editer une figure	1	2	3	4	5
Editer un graphique	1	2	3	4	5
Test rapide	1	2	3	4	5
Modification du texte	1	2	3	4	5
Manipulation graphique	1	2	3	4	5
Manipulation fichiers	1	2	3	4	5
Copie de leçons	1	2	3	4	5
QCM	1	2	3	4	5
Lacunaires	1	2	3	4	5
Appariement	1	2	3	4	5
Question ouverte	1	2	3	4	5
Classement	1	2	3	4	5
Sélection	1	2	3	4	5

Cotation	1	2	3	4	5
Utilisation du tirage au hasard	1	2	3	4	5
Boucle	1	2	3	4	5
Branchement linéaire	1	2	3	4	5
Branchement direct	1	2	3	4	5
Branchement dynamique	1	2	3	4	5

3. Evaluez la flexibilité du logiciel auteur dans les domaines suivants :

	Très flexible	Flexible	Peu flexible	Inflexible	Pas applicable
Affichage de texte	1	2	3	4	5
Graphiques	1	2	3	4	5
Sons	1	2	3	4	5
Création de séquence	1	2	3	4	5
Branchement	1	2	3	4	5

C. LA SEQUENCE CONSTRUITE

Evaluez la qualité des aspects suivant de la séquence.

	Excellent	Bon	Faible	Pauvre	NA
Système d'aide	1	2	3	4	5
Affichage de texte	1	2	3	4	5
Affichage de graphique	1	2	3	4	5

Evaluez le temps requis pour

	Bon	Raisonnable	Passable	Excessif	NA
Apprendre le logiciel	1	2	3	4	5
Faire une leçon	1	2	3	4	5
Organiser affichage de texte	1	2	3	4	5
Organiser affichage graphique	1	2	3	4	5
Organiser présentation de son	1	2	3	4	5
Organiser le branchement	1	2	3	4	5
Entrer les questions	1	2	3	4	5

BIBLIOGRAPHIE

ALBERTINI J.M., Vers une informatique de formation, Education Permanente, 70-71, 1983.

BARKER P.G., Mumedala : an approach to multi-media authoring, Comput. Educ. Vol. 8 n° 4.

BIDEAU R., Les systèmes auteur, BIP, n° 37, 1985.

DUBUC L., A l'encontre des logiciels auteur, BIP, n° 36, 1984.

GILLINGHAM et al., An evaluation of computer courseware authoring tools, Educational Technology, septembre 1986.

HARDY A., ROMAINVILLE M., Introduction au langage auteur SUPERPILOT, Département Education et Technologie, Doc 1.8, 1985.

JOMASSEN D. H., The real case for using authoring systems to develop courseware, Education Technology, février 85.

KOZMA R. B., Present and Future Computer Courseware Authoring Systems, Educational Technology, juin 1986.

PAUL L., P.E.N., Progiciel éducatif Nathan, E et I, 15, 1983.

POLLOCK Y., Authoring Courseware : no skills necessary ?, Educational Technology, novembre 85.

YOUNG J. L., The Case of Using Authoring Systems to Develop Courseware, Educational Technology, octobre 84.

———, PEDAGO, logiciel de générations d'exercices, E et I, n° 13, 1982.

LISTE DES LOGICIELS

ADDITION, P. Dillembourg, Centre OSE Mons (Apple II)

APPLE II SUPERPILOT, editor's manual, Cupertino, 1980. (Apple) Le PILOT existe également sur d'autres machines. (PC, C64, BBC, TRS)

ARBRE, aide à la construction de séquences d'enseignement assisté par ordinateur, Province de Liège, Centre des Méthodes, 1985. (Apple)

CALCUL FRACTIONNAIRE, CEFIS Namur (Apple II)

DIGIT-DACT, SOFI, Montréal Québec (Apple II)

DUO, Système auteur, DDTEC, Paris, (PC HP 150)

EGO, 3 P informatique, Paris (PC ou compatibles)

EURIDIS, Système auteur, Hachette informatique, 1984. (Thomson)

EVA, système auteur, Eduvision, Paris, (PC).

MACDIDACTIC, NTI, Reims, (Macintosh).

MICROSCOPE-5, Jacques Ste Marie, Guérin, Montréal, Québec.

PEDAGO, logiciel de génération d'exercice, Edumicro, Paris (Micral)

PEN, système auteur, Nathan, Paris, (PC, Apple, Thomson)

7

FORMATION EN EDUCATION N°1.11

RECHERCHE

CENTRE O.S.E.

Les leurres et les problèmes de l'initiation à l'informatique

Marc ROMAINVILLE

DEPARTEMENT
EDUCATION & TECHNOLOGIE

Facultés Universitaires Notre-Dame
de la Paix B-5000 Namur

R. MOREL
oct. 87

3044

INTRODUCTION

Ce document a pour point de départ la constatation suivante : les discours sur l'E.A.A.O. se multiplient actuellement à une allure vertigineuse alors que, d'une part, son emploi et son efficacité paraissent très souvent critiqués et remis en cause et que, d'autre part, sa pratique est relativement rare compte tenu du manque évident de matériel et de didacticiel de qualité.

A l'inverse, on parle très peu de l'initiation à l'informatique; or, cette activité est relativement répandue et son bien-fondé est en général reconnu par tous. Il existe désormais des cours à option d'initiation à l'informatique dans le secondaire; le décret RISOPOULOS recommande la mise au point d'une alphabétisation informatique dès l'enseignement primaire. Les parents, soucieux de ne pas rater le "train du progrès" recommandent une formation à l'informatique : ainsi une des résolutions de la C.N.A.P., présentée au Congrès du 1er mars 1986, concerne la formation en informatique. Voici, à titre d'exemple, le texte de cette résolution :

> Une initiation générale pour tous les élèves
>
> Devant l'importance que revêtira l'informatique dans le monde de demain, l'école se doit de lui faire une place de choix dans tous les niveaux d'enseignement et dans toutes les disciplines.
>
> Les parents demandent :
> une initiation générale pour tous les élènes comme activité d'éveil dans l'enseignement primaire, et comme matière à apprendre dans l'enseignement secondaire. (1)

Les parents ne sont pas, bien sûr, les seuls à réclamer cette initiation. En France, une note du Directeur des Ecoles confirme cette tendance. Voici un extrait de l'introduction :

> "Ainsi envisagée, l'informatique, par son développement, affecte notablement le milieu où vivent les enfants et, vraisemblablement, son importance ira en grandissant dans les années à venir.
> Or, c'est une des finalités générales de l'école de préparer les enfants à vivre dans leur milieu (actuel et futur) en exerçant sur lui une maîtrise pratique autant qu'intellectuelle. Les activités d'éveil, à l'école élémentaire, sont précisément celles qui doivent permettre à l'enfant d'être tourné vers son milieu environnant, et de construire les pratiques et les représentations qui le lui rendent plus familier et plus clair. C'est donc dans un tel cadre qu'il convient de faire entrer l'informatique à l'école."

(1) paru dans Les parents et l'école, n° 2, 1986.

Enfin, les enseignants eux-mêmes semblent se rallier à cet avis : une enquête réalisée auprès des enseignants du fondamental de l'Etat montre, par exemple, que 82 % d'entre eux marquent leur accord pour la proposition suivante " Promouvoir l'informatique dans l'enseignement fondamental est indispensable.(1)

Et l'on pourrait ainsi multiplier les exemples; l'essentiel est de montrer qu'il se réalise une certaine unanimité quant au bien-fondé de cette initiation générale.

Nous avons voulu rassembler dans ce document quelques réflexions critiques concernant cette vague d'alphabétisation informatique. L'objectif est d'en montrer les leurres et les difficultés et de tenter d'aller plus loin dans la définition des contenus et des méthodes liés à cette initiation.

QUELS OBJECTIFS ?

Analysons de plus près ce phénomène en essayant de repérer les motifs invoqués par ces ardents défenseurs d'une initiation à l'informatique.

Dans la plupart des documents que nous avons récoltés (2), nous retrouvons le même type d'exposé des motifs :

(1) M.E.N., Organisation des Etudes, Informatique et enseignement fondamental de l'Etat.

(2) Résolutions de la C.N.A.P. (a);
Orientations fondamentales du Conseil Central de l'Enseignement Primaire Catholique (b);
Conclusions du colloque Education et Société du 3ème type, Club Athéna, Technologies-Education (c);
Proposition de développement - Ministère de l'Education, Québec(d);
Note du Directeur des Ecoles en France (e);
Programme d'études : Introduction à la science de l'informatique (Ministère de l'Education du Québec, septembre 1982);
Commission Bündlander sur la planification de l'éducation et l'aide à la recherche, décembre 1985. (Projet pour l'introduction d'une formation aux techniques de l'information à l'école)

dans un premier temps, les auteurs insistent sur un objectif d'"éveil au phénomène socio-culturel et technologique que constitue l'informatique"(e). Il s'agit donc essentiellement, dans le cadre des activités d'éveil, de montrer aux enfants l'importance que l'informatique a prise dans notre société, son rôle, ses effets positifs et négatifs, de les familiariser avec la machine, ses composantes et sa structure.

Dès cette première étape, on pourrait se poser un certain nombre de questions :

1. Si les activités d'éveil ont pour objectif d'aider les enfants à appréhender un certain nombre de données ou de dimensions de leur environnement, il est de fait assez logique qu'actuellement la dimension informatique soit prise en compte. Mais pourquoi pas d'autres ? En effet, d'autres révolutions technologiques ont profondément influencé nos sociétés : le téléphone et l'audiovisuel de masse pour ne citer que deux exemples. Existe-t-il, dans les activités d'éveil, des séances d'alphabétisation téléphonique ou audiovisuelle ?

2. Plus sérieusement, quel sera le contenu de l'éveil technologique ? En effet, si le contenu de l'éveil socio-culturel peut aisément être délimité (analyse et observation d'une société informatisée), l'éveil technologique pose plus de problèmes :

- jusqu'où aller ? Il est impossible de vouloir réellement faire comprendre le fonctionnement de base d'une machine à des jeunes enfants . S'agit-il alors simplement de savoir utiliser -voire simplement de brancher - l'appareil ? Mais alors la question réapparaît : donne-t-on également des cours sur le maniement du magnétoscope ou de la caméra ? J. WEIZEN BAUM, professeur au M.I.T., nous a habitué depuis bien longtemps à de pareilles critiques :

> "Savez-vous comment fonctionne un téléphone ou un réfrigérateur ? Non. Et pourtant vous vous en servez tout le temps. Demain, quand vous utiliserez le computer ou la machine à traitement de texte, vous ne saurez pas non plus comment ça marche. Il suffira d'appuyer sur le bouton."(1)

- à quelle technologie initier ? Celle-ci ne sera-t-elle pas complètement dépassée une fois le cours terminé ?

(1) Extrait d'une interview de J. WEIZENBAUM : L'ordinateur à l'école : une plaisanterie, Le Nouvel Observateur, n°2228, décembre 1983.

Mais assez rapidement, les auteurs passent, dans un second temps, à un motif qu'ils jugent souvent plus important : au-delà de cette première alphabétisation, on découvre alors un certain nombre d'objectifs plus ambitieux concernant le développement de la pensée logique - voire de la pensée tout court - à travers la programmation. Prenons quelques exemples : le premier est tiré des conclusions du congrès de la C.N.A.P.

> "La rigueur, résultant de la démarche algorithmique, fondement de l'informatique constitue un apport indispensable à la formation des élèves de l'an 2000."(a)

On retrouve la même proposition dans le document du C.C.E.P.C.(b)

> Développer la pensée algorithmique de l'enfant, l'amener à programmer, c'est-à-dire à organiser une "marche à suivre" pour résoudre une situation-problème rencontrée dans son environnement réel et imaginaire; considérer l'enfant en apprentissage en le plaçant dans des situations-problèmes propres à son milieu de vie afin de l'engager dans un ensemble de démarches universelles qui développent chez lui des comportements intellectuels de recherche. En apprenant à chercher, l'enfant apprend à penser.

Enfin cette même insistance sur la programmation peut également être isolée dans le document français (e).

> Si l'introduction d'ordinateur à l'école peut apporter une innovation pédagogique véritable et quelque chose de proprement inédit, c'est certainement par la possibilité de la programmation. Il faut que les enfants programment eux-mêmes pour entrer en relation véritable avec l'informatique et se l'approprier dans l'autonomie.

L'ensemble des documents mettent donc bien l'accent sur ce que l'on pourrait appeler la "programmation-prétexte". Car partout, il est bien rappelé que ce n'est pas l'apprentissage d'un langage qui est visé, mais bien le développement des capacités congitives logiques supposées être la base de tout travail de programmation structurée. Ces "skills" de résolution de problèmes sont, entre autres, les capacités à poser un problème, à le décomposer en sous-problèmes, à planifier des actions, à anticiper des résultats (formuler des hypothèses) etc.

Un des relevés des plus complets des effets cognitifs attendus de l'activité de programmation a été réalisé par FEURZEIG (cité dans [5] p.143).

1) la rigueur de la pensée et la précision de l'expression;
2) la compréhension de concepts généraux tels que variable, fonction;

3) une plus grande facilité dans l'utilisation d'heuristiques, c'est-à-dire d'approches générales de résolution de problèmes telles que la planification, la décomposition en sous-problèmes;
4) la conception du debugging (recherche des erreurs) comme une étape constructive et planifiable;
5) l'attitude générale de chercher une solution à un problème important en construisant petit à petit des solutions partielles;
6) la réflexion et la prise de conscience des processus de résolution de problèmes;
7) la prise de conscience du fait qu'il n'y a que rarement une seule "bonne" solution mais plus souvent différents chemins avec chacun leurs avantages et inconvévients.

Voilà donc bien le vrai motif et l'on se rend compte alors qu'il n'est pas spécifiquement lié à la technique informatique mais qu'il vise ces fameuses capacités cognitives de base. Il est aussi évident que la haute qualité de ces objectifs provoquent l'unanimité et ... malheureusement clôturent les documents comme si le seul énoncé des objectifs entraînait inexorablement leur réalisation.

Or, il me semble - et c'est là l'objectif de ce document - qu'il reste en certain nombre de points d'interrogations importants; et que ces objectifs ne sont que des points de départ, des lignes de force à partir desquelles nous devons développer des moyens, des outils, des méthodes, des contenus précis.

Si ce travail ne se réalise pas, le risque est grand de voir l'initiation à l'informatique, justifiée par l'annonce d'objectifs de hauts niveaux, se réduire à un cours mi-historique (les premiers ordinateurs,etc...), mi-technique (qu'est-ce q'une RAM, ROM,etc...).

QUELQUES PROBLEMES ?

En simplifiant un peu, le motif fondamental est donc le suivant : l'enfant, en apprenant la programmation, appprendra et exercera des techniques de résolution de problèmes (1). On peut déceler deux niveaux de questions dans cette proposition : premièrement on suppose que l'enfant apprendra sans trop de difficultés la programmation, deuxièmement on suppose que, cela étant fait, il aura appris des techniques transférables à d'autres situations.

Un certain nombre de faits vont cependant relativiser très fortement ces deux affirmations.

1) Les problèmes rencontrés dans l'apprentissage de la programmation

Un certain nombre d'études montrent que - contrairement au mythe de l'"enfant programmeur" - les débutants en programmation éprouvent un certain nombre de difficultés importantes. Celles-ci peuvent être regroupées en trois moments [1] :

a. la planification : les erreurs ont trait, dans ce cas, au découpement logique et préalable de la tâche . Ex. : sous-spécifier le problème : ne pas prévoir ce qui se passe quand telles conditions ne sont pas remplies.
b. le codage : il s'agit des erreurs concernant le langage utilisé (sa syntaxe, sa logique interne, etc...). Ex. : erreur de sens : lire un fichier sans l'avoir ouvert.
c. le debugging : ces erreurs concernent la recherche des "bugs".

En LOGO, par exemple, des enfants de 9 à 15 ans éprouvent des difficultés dans les domaines suivants [1] :

(1) Ce qui est d'ailleurs, fondamentalement, la thèse de S. PAPERT.

1) manque de spécifications;
2) confusion des rôles de la machine :
   - chargeur de LOGO;
   - exécution;
   - édition;
   - utilisation d'un programme LOGO.
3) croyance que le système stocke automatiquement un certain nombre de données;
4) distinction nom/valeur d'une variable, d'une procédure. Ex. : croire qu'un nom de procédure doit obligatoirement signifier ce qu'elle fait;
5) distinction répétition/récursivité;

etc...

Remarquons au passage qu'il ne s'agit pas souvent d'erreurs de syntaxe ou de règles sémantiques.

Les auteurs notent également que les enfants ne décomposent pas assez leurs problèmes, construisent des procédures peu élégantes, n'utilisent pas les solutions apprises dans de nouvelles situations. Enfin ces problèmes semblent être d'une importance très variable d'un enfant à l'autre.

Bref, il n'est pas sûr qu'un certain nombre de concepts riches (variable - recursivité) soient réellement maîtrisé par la plupart des enfants.

Un bel exemple de représentation erronnée et systématique d'un concept de base est analysé dans un article de KURLAND et PEA (cité dans [5] p. 147). Il concerne la manière dont se déroule une procédure récursive en LOGO.

> "La plupart des enfants croient que le fait de placer le nom d'une procédure à l'intérieur d'elle-même provoque l'exécution d'une boucle à l'intérieur de la procédure, alors qu'en réalité, le contrôle passe à une copie de la procédure. Cette procédure est exécutée et, quand le processus est terminé, elle repasse la main à la procédure qui l'a appelée en dernier lieu.
> Les enfants adoptent des modèles mentaux de déroulement du programme qui marchent pour des cas simples, telle que des programmes constitués d'une seule procédure, mais qui se révèlent inadéquats quand le projet requiert une constitution de procédures plus complexes".

Une étude intéressante [2] tente d'ailleurs de faire un parallèle entre les difficultés rencontrées par des enfants de 8 à 13 ans et leur niveau opératoire (en terme piagétien, leur niveau de développement intellectuel) : on donne aux enfants (après un entraînement de 10 heures au LOGO) un quadrillage représentant l'écran et un énoncé de programme que l'enfant est invité à exécuter lui-même en dessinant le trajet de la tortue. Deux types d'erreurs sont isolés :

- les erreurs d'interprétation : erreur sur la signification d'une instruction;
- les erreurs de coordination : les instructions isolées sont correctement interprétées mais mal agencées; ex. :certains appliquent à la répétition une sortie de règle de distributivité : l'instruction (répète 4 [av 20 av 90 av20] est "exécutée" de la manière suivante (répète 4 fois av 20, puis répète 4 fois av 90, etc.

Ces mêmes enfants passent une épreuve de l'échelle de Développement de la pensée logique permettant de répartir les enfants en 3 groupes : niveau opératoire, niveau préformel, niveau formel. L'étude montre qu'il existe des différences significatives entre les moyennes des erreurs en fonction du niveau opératoire.

Voici les conclusions de l'auteur concernant ces difficultés logiques :

> La compréhension de programmes suppose que l'enfant construise, en plus d'une représentation fonctionnelle de la signification des primitives, une aptitude à mobiliser mentalement des opérations qui rendent compte de la structure des programmes. Nous avons montré que cette aptitude est reliée à la capacité du sujet à pouvoir coordonner plusieurs opérations élémentaires. Ce phénomène explique les raisons pour lesquelles la capacité à manier des programmes quelque peu complexes, c'est-à-dire qui dépassent la simple structure séquentielle, n'est guère possible chez l'enfant avant 10-11 ans (au moment de la mise en place des opérations formelles). Mais, cela signifie en même temps, que l'activité de programmation est une bonne situation d'apprentissage pour l'exercice de ce qui est un des moteurs les plus puissants de la logique formelle, à savoir la construction d'opérations sur des opérations.

D'autres recherches chez des enfants plus âgés mettent également en évidence des difficultés d'appropriation de certains concepts : l'itération pour des enfants de 11 à 15 ans [3], la planification et la mise en évidence des implicites chez des élèves de 15 à 16 ans [4].

Enfin le GRAI (groupe de recherche sur l'apprentissage et l'informatique (1)) poursuit depuis plus de deux ans des recherches exploratoires visant à mettre en évidence les difficultés rencontrées par des adultes lors de leurs premières initiations à l'informatique. Les observations sont réalisées d'une part à l'occasion de nos formations d'enseignants et d'autre part en situation individuelle avec enregistrement des commentaires et des ordres donnés au clavier, l'adulte étant dans ce cas confronté à une première situation problème LOGO. Les premiers résultats permettent d'épingler un certain nombre de difficultés :

a) La découverte des caractéristiques du dialogue homme-machine semble poser problème. Les adultes observés éprouvent notamment des difficultés à :
   - cerner le type de langage : mode, temps, personne, degrés de simplicité, de complexité de la syntaxe,etc;
   - cerner les caractéristiques de la tortue : elle a une orientation, etc;
   - accepter l'impossibilité de projeter sur la tortue une compétence linguistique; contrairement à ce qui ce passe dans le dialogue humain, les adultes ne peuvent supposer à aucun moment un "background" chez la tortue leur permettant de laisser les implicites habituels dans leurs discours.

b) Un autre problème important est la difficulté de cerner la fonction en cours de la machine : ils confondent ainsi l'exécution et l'édition, l'utilisation du programme et sa construction.

Les comportements indicateurs de ces problèmes sont les suivants :
- tenter d'exécuter un programme dans l'édition;
- croire que les ordres donnés en mode direct sont enregistrés;
- en faisant exécuter un programme, ne pas "jouer le rôle " de l'utilisateur;
- attribuer des contenus de variables dans l'éditeur;
- etc.

(1) Département Education et Technologie, FNDP, 61, rue de Bruxelles, Namur

c) La confusion entre le nom de la variable et son contenu est également fréquente : pour certains, il est difficile de savoir quand on la définit et quand on lui donne une valeur; le problème étant résolu, dans certaines cas, en faisant les deux en même temps !
ex. : POUR CARRE :20
Il est à noter que ce problème est en rapport avec celui évoqué au point a) : dans ce cas,l'adulte confond la définition de la procédure et son appel.
Ce problème se retrouve quand il s'agit de manipuler les variables.
ex. : ECRIS "X pour écrire le contenu de X

d) La distinction n'est pas toujours bien établie entre ce qui est conventionnel et ce qui ne l'est pas. Il n'est ainsi pas rare qu'on nous demande si le nom de telle ou telle procédure ou variable pourrait être modifié.
Tout se passe comme si l'adulte, inhibé par les inévitables erreurs de syntaxe réalisées en début d'apprentissage, semble s'en remettre le plus possible à "ce qui a marché", y compris des désignations purement conventionnelles.

e) Les nombreuses informations contenues dans les messages d'erreurs sont fort peu utilisées; l'attitude face au "bug" est essentiellement passive (éditer la procédure, la relire), c'est surtout la formulation d'hypothèses et l'organisation de tests qui font défaut.

2) Les problèmes de transfert

En supposant le premier point réalisé, il reste, nous l'avons vu, une question fondamentale : les élèves ont-ils ainsi acquis des techniques générales de résolution de problèmes.

PEA et KURLAND ont réalisé une revue de la question intéressante dans ce domaine. Ils font d'abord remarquer qu'il s'agit d'une "vieille idée dans un nouveau costume" :

> "Dans la forme extrême, cette idée repose sur un présupposé concernant l'apprentissage qui veut qu'une expérience spontanée d'interaction avec un système symbolique riche a des conséquences cognitives bénéfiques, essentiellement pour les "skills cognitifs" de haut niveau. Des arguments semblables ont déjà été utilisés dans les siècles passés pour les mathématiques, la logique, la grammaire et le latin."[5 p.138]

Passant ensuite en revue une série d'études visant à mesurer le transfert réalisé par les élèves des activités de programmation à d'autres types d'activités, ils concluent qu'il n'existe encore que très peu de preuves de bénéfices cognitifs importants :

> "Premièrement, il n'y a pas d'études sérieuses qui supportent l'idée que la programmation favorise la rigueur mathématique.(...) Deuxièmement, aucun rapport de recherche ne démontre que la programmation aide les enfants à explorer les mathématiques.(...) Troisièmement, il n'est pas clair non plus qu'elle les aide à acquérir certains concepts mathématiques.
> Enfin nous nous demandons s'il est vrai que la programmation peut être considérée comme un contexte et un langage qui favorise le "problem solving"... Les résultats indiquent que les expériences LOGO n'ont pas d'effets significatifs sur les performances de planification..."[3 p.160]

Ces résultats peuvent paraître décourageants mais il faut cependant insister sur la nécessité d'affiner ce type d'étude dans deux directions au moins :

1) la finesse des mesures de transfert;
2) la spécification du contexte de l'activité de programmation.

1) Les mesures réalisées sont en effet souvent trop globales (ex. : 5 résolutions de problèmes mesurés en termes d'échec ou de réussite) et trop éloignées des situations LOGO. Des mesures plus fines peuvent faire apparaître des différences significatives parfois dans des domaines à priori peu en rapport avec l'activité de programmation. Un bon exemple nous est donné par l'étude de BIDEAULT visant à mesurer l'impact d'une pratique LOGO sur une épreuve de construction de parcours.

> "A partir d'un grand nombre d'observations sur le terrain au cours des séances LOGO, chez les enfants de 8 à 10 ans (CE2), l'hypothèse principale était la suivante : les stratégies de construction de parcours, utilisées lors de l'apprentissage LOGO, sont transférables à d'autres situations non LOGO ... Une seconde hypothèse concernait la verbalisation :
> L'apprentissage du LOGO oblige à une verbalisation précise d'un déplacement dans un espace bidimensionnel. En effet, les observations effectuées incitent à penser que la pratique du LOGO doit permettre à l'enfant d'accéder à un codage verbal du parcours plus précis (utilisation d'une direction et d'une mesure) qu'un enfant n'ayant pas fait de LOGO."[6 p.201]

Deux groupes équivalents sont constitués; le premier seulement prend part, une matinée par semaine, à des activités LOGO. Les deux subissent en fin d'année une épreuve sur feuille de papier : l'enfant dispose d'un plan et d'un bonhomme en plastique; il lui est demandé de construire un tracé de manière à rejoindre deux maisons sans passer par un certains nombres de points, de rédiger une marche à suivre destinée au bonhomme et enfin de trouver le tracé le plus court.

L'analyse des résultats montre que :

a) il n'y a pas de différence entre les groupes au niveau du type de stratégie employée.

> "Les enfants du groupe contrôle et des groupes expérimentaux arrivent à des résultats équivalents, ils ont trouvé dans leur expérience des sources d'enrichissement diverses; une diversité de stratégies apparaît dans la population. Par des voies différentes, les enfants parviennent au même résultat, ceci suggère l'existence de processus vicariants.
> Si LOGO n'apporte pas de nouvelles stratégies, du moins il ne limite pas les enfants à un seul type de réponse stéréotypée, LOGO n'a pas d'effet rigidifiant."[6 p.208]

b) le groupe expérimental est plus précis dans ses ordres en employant plus souvent la mesure.
c) le groupe expérimental utilise des stratégies de vérification du codage énoncé en utilisant le bonhomme en plastique.

> "Ces stratégies de vérification semblent liées à une certaine capacité d'introspection chez le sujet, qui a été favorisée par la pédagogie LOGO pratiquée avec ces deux groupes expérimentaux, comme le montrent de nombreuses observations sur le terrain."[6 p.209]

d) le groupe expérimental se montre davantage capable d'expliquer le comment de leurs actions. Les expressions verbales utilisées font appel à des comparaisons, à des mesures, etc.

Les bénéfices ne se mesurent donc pas globalement en terme de réussite ou d'échec mais se manifestent par l'acquisition d'attitudes telles que l'utilisation d'un outil de vérification, la nécessité d'expliciter aux autres ses actions, de prouver ses résultats.

> "Le rôle du conflit cognitif lors des séances d'apprentissage est probablement déterminant pour amener le besoin de prouver. Il est permis de penser que l'environnement LOGO, par une confrontation avec les autres, et avec la machine, incite l'enfant à perdre l'impression trompeuse d'être toujours compris, d'où l'apparition du besoin de prouver."[6 p.210)]

2) La question de savoir quels sont les bénéfices cognitifs de la programmation est également trop générale pour une autre raison. En effet, qu'entend-on par programmation ? Qu'a-t-on enseigné aux élèves ? Quel langage ? Combien de temps ? Dans quelles conditions ? Etc...

> "La programmation sert de boîte noire, d'activité globale dont les effets sont supposés irradier ceux qui s'y exercent."[5 p.144]

Il est en effet important de préciser le contexte dans lequel les étudiants ont eu l'occasion d'exercer des activités de programmation car celles-ci vont influencer fortement les possibilités de transfert :

> "La question du rôle du contexte dans l'apprentissage de la programmation est complexe parce qu'il ne s'agit pas d'une compétence unique. Tout comme la lecture elle englobe un grand nombre d'habiletés qui interagissent avec les connaissances antérieures de l'élève, sa mémoire, ses capacités de traitement de l'information, ses stratégies générales de résolution de problèmes telle que l'inférence, la génération d'hypothèses, etc... L'entraînement à la lecture suppose une gamme étendue d'expériences avec différents genres (narrations, essais, poésie, débat) et avec différents buts. Tout comme la lecture est souvent assimilée à l'habileté de décodage, l'apprentissage de la programmation est souvent équivalente à l'apprentissage d'une syntaxe et d'un vocabulaire d'un langage; alors que l'entraînement à la programmation, comme la lecture, est complexe et fortement dépendante du contexte. Nous devons donc commencer par préciser le contexte dans lequel la programmation a été mise en pratique et apprise."[5 p.144]

On se rendra compte ainsi que les probabilités de transfert vont fortement varier d'une situation à l'autre. Il est notamment important de savoir [5] :

a) Quel est l'environnement informatique ? Quel langage ? Existe-t-il un éditeur facilitant l'écriture du programme, des aides de debugging, de la documentation, etc...?

Ces facilités permettent en effet de rendre le codage plus aisé et de se concentrer ainsi sur les questions plus intéressantes de design, de structuration, d'élégance du programme.

b) Quelles sont les conditions d'enseignement ? Quelles sont les modalités d'accès aux machines ? Les possibilités de sauvetage du travail de l'élève ? Les cours sont -ils centrés sur la syntaxe, le vocabulaire ? A-t-on bien vérifié l'assimilation de concepts fondamentaux (variables -variable globale/locale, etc) ? A-t-on imaginé des exercices autres que l'entrée des données (ex. : prévoir les contenus des variables à chaque étape de l'exécution - choisir diverses données et prévoir le déroulement du programme en conséquence (avec les "si")) ? Quelle est la part le l'enseignement et des exercices ? Quel est le degré de directivité ? A-t-on enseigné des stratégies de debugging ? Etc.

PEA et KURLAND montrent ainsi qu'en fonction de ces différentes variables du contexte, les étudiants peuvent se situer à différents niveaux de maîtrise de la programmation. A chacun de ces niveaux, les habiletés transférables vont être différentes.

Niveau 1"Utilisateur"

L'étudiant est capable d'utiliser des programmes; on ne peut espérer aucun transfert à d'autres situations mais bien sur son niveau d'alphabétisation informatique. (Exemple : en utilisant différents programmes, il peut distinguer progressivement les fonctions qu'un ordinateur peut prendre en charge de celles qu'il lui est impossible d'assumer.)

Niveau 2 "Encodeur"

L'étudiant connaît la syntaxe et la sémantique des principales instructions d'un langage; il peut créer des programmes courts sur le modèle de ceux qu'il a déjà rencontré. L'attention est tellement portée sur l'exécution (on cherche à ce que le programme marche), que la probabilité de transfert de stratégies générales est faible.

Niveau 3 "Créateur de programme"

A ce niveau, l'étudiant devient capable de raisonner en termes

d'unité de haut niveau; il connaît des séquences d'instructions réalisant certains sous-objectifs fréquents (lire au clavier,etc). Il peut écrire de longs programmes mais peu "conviviaux". Il devient conscient des processus mis en oeuvre dans l'écriture d'un programme et cela semble une des conditions pour d'un transfert se réalise.

Niveau 4 "Créateur de software"

L'étudiant est alors capable de concevoir des programmes non seulement complexes et structurés mais aussi tournés vers l'utilisateur. Il commente ses programmes, en fait une analyse préalable, est capable de "simuler" son exécution. C'est à ce niveau que le transfert est sans doute le plus probable : il prend de la distance par rapport au code utilisé et se concentre sur les phases et les processus de résolution du problème que constitue la construction du programme. Par exemple, la nécessité à ce niveau d'être conscient de l'éventail des souhaits des utilisateurs le force à accorder toute son attention à ces derniers. Cette habileté se révèle utile dans beaucoup d'autres situations, en particulier dans les activités d'écriture.

Laissons à PEA et KURLAND le soin de conclure cette partie.

> "Derrière cette distinction entre différents niveaux de maîtrise de la programmation et leurs implications au niveau du type de transfert possible, il y a implicitement une théorie en désaccord avec la théorie "technoromantique"(1) qui prévaut actuellement dans le domaine de l'informatique éducative.
> Bien qu'il soit concevable que même les niveaux les plus bas de maîtrise produisent un transfert cognitif mesurable dans des domaines qui ne concernent pas la programmation, nous prétendons que, sur base des preuves disponibles limitées, cela est peu probable. Les élèves qui décodent et comprennent à peine un texte seront peu probablement de bons écrivains. De la même manière, nous doutons que des étudiants qui ne possèdent qu'un niveau assez bas de compréhension de ce qu'est la programmation et des habiletés qu'elle entraîne puissent écrire des programmes fonctionnels et s'améliorer ainsi dans d'autres domaines sur base de leur capacité limitée à programmer."[5 p. 155]

(1) Selon PEA et KURLAND, il y a actuellement dans le milieu scolaire un optimisme sans limite concernant les bénéfices cognitifs de la programmation qui a pour résultat que cette hypothèse est devenue une affirmation sans aucun recours à la vérification empirique.

Implications au niveau d'une initiation à l'informatique

Après avoir passé en revue quelques problèmes liés à l'apprentissage de la programmation et au transfert, il nous faut en conclure certaines conséquences pratiques au niveau des activités organisées en classe. Que nous disent ces recherches au niveau des contenus et des méthodes d'une initiation à l'informatique ?

1. La première conséquence importante est d'adopter une **position plus modeste et plus réaliste par rapport aux bénéfices escomptés de la programmation**. Plusieurs expériences mettent un terme à certaines illusions pédagogiques concernant l'ordinateur et mettent l'accent sur l'importance de l'environnement pédagogique.

> "Il semble que dans ce type de recherche, il faut se garder de tomber dans la mystique d'un instrument, aussi attrayant qu'il soit, mais qu'il faut plutôt appréhender l'ordinateur dans un ensemble où le rôle de la situation pédagogique, dans sa complexité, intervient au premier plan."[6 p.208]

Dans le même ordre d'idée, il faut bien resituer le type de "pensée" engagée dans la programmation parmi l'ensemble des processus cognitifs à faire acquérir.

> "Programmation des ordinateurs par et pour la transparence des chemins de la connaissance : voilà l'objectif. Dans ce cadre, la performance, la logique formelle, la rationalité technique règnent en maîtres. Ainsi ce que S. PAPERT désigne par la pensée abstraite n'est le plus souvent qu'une catégorie intimement liée à une intelligence pratique-technique des modes de fonctionnement machiniques. Selon un modèle très au goût du jour, la complexité de l'abstraction doit être traitée par la concrétisation, la manipulation d'objets et de procédures. Ainsi lorsqu'un utilisateur se propose de dessiner une fleur avec le langage LOGO (ou un autre), il se situe toujours dans un registre de concrétisations de capacités assez délimité : celui de l'activité rationnelle par rapport à une fin, de la gestion univoque des moyens, d'une logique non ambigüe, etc."[7 p.21]

2. L'objectif étant ainsi plus délimité, la seconde conséquence pratique est **la mise en oeuvre de toute une série de moyens pour que les élèves atteignent une maîtrise suffisante de la programmation** pour qu'un transfert soit possible (niveaux 3 et 4).

2.1. Une des premières conditions semble être la **limitation de l'âge** à partir duquel il est raisonnable d'envisager une initiation de ce type. En effet, il semble que certaines opérations nécessaires relèvent du stade de développement formel et que, dès lors, ces activités ne seraient accessibles qu'à des enfants de 10-11 ans. HOC rappelle que :

> "Si la programmation partage les propriétés générales des activités de résolution de problèmes, elle se distingue par le fait qu'elle exige du sujet qu'il exprime les procédures qu'il élabore.
> Il est utile d'opposer ces situations de programmation aux situations qui ont été qualifiées de "productions de résultats", en définissant ainsi deux pôles extrêmes, entre lesquels il faut voir un continuum. Dans les situations de production de résultats, l'objectif de l'élaboration de la procédure se réduit à l'exécution. L'élaboration peut se dérouler conjointement à l'éxécution, à mesure que les données particulières se présentent, sans qu'il soit nécessaire d'expliciter la procédure, notamment sa structure de contrôle. Les situations de programmation nécessitent, pour le sujet, un niveau de prise de conscience plus élevé que les situations de production de résultats, par ailleurs les plus courantes."[8 p.386](1)

L'étudiant - comme nous l'avons vu plus haut - organise ses instructions, dispose de plans, de schémas pour certains buts spécifiques, "intériorise" les stratégies utilisées pour construire les programmes antérieurs. Ces capacités -parfois appelées métacognitives - se développent en effet à l'entrée de l'adolescence (cfr BROWN).

2.2. **Les cours doivent porter autant sinon plus sur les méthodes que sur les contenus**. Nous avons vu ci-dessus qu'une énumération des instructions et de la syntaxe du langage était loin de suffire. Par exemple, une période de travail sur papier doit précéder ou accompagner le travail sur machine : elle consistera par exemple en un exercice de décomposition de problèmes, en une élaboration, à partir des expériences de chaque étudiant, d'une stratégie de recherche d'erreur, etc.

(1) CH. DUCHATEAU établit une distinction similaire entre le "faire" (résolution de la tâche) et le "faire faire" (faire résoudre le problème par un exécutant, ce qui nécessite une explicitation de la procédure).[13]

Il serait aussi intéressant de rassembler les règles, "trucs" et stratégies que les experts en programmation utilisent couramment (ex. : insérer dans un programme une instruction qui permette d'afficher le contenu des variables à certains points stratégiques du programme).

2.3. **Les cours devraient être centrés sur la <u>résolution de problèmes</u>** - puisque c'est fondamentalement cela qu'il s'agit de transférer - par les élèves plutôt que d'être un enseignement de contenus. Il semble dès lors primordial d'**<u>intégrer cette initiation informatique</u> aux autres disciplines** plutôt que d'en faire une matière à part, que ce soit dans l'enseignement fondamental ou dans l'enseignement secondaire.

Les problèmes choisis seraient ainsi en rapport avec leur vie quotidienne ou avec des questions apparues à l'occasion d'autres cours.

L'intégration de la programmation au cours de mathématique semble, par exemple, enrichissante (1).

> "A côté du problème qui consiste à essayer de rendre le matériel accessible à tous les étudiants de mathématique et non à une élite, il y a une autre question importante : comment incorporer cette initiation à la programmation dans des programmes déjà détaillés et complets.
> En sélectionnant des activités qui ont un rapport avec des parties du programme, on peut utiliser l'ordinateur pour enseigner ces parties. Par exemple, pour découvrir le concept de"premier", les étudiants peuvent écrire un programme qui génère des nombres premiers. Non seulement, l'étudiant doit bien comprendre le concept, mais il doit aussi communiquer l'algorithme à l'ordinateur. Une compréhension de surface ne peut suffire. L'étudiant maîtrise réellement le concept quand il est capable de le verbaliser, dans ce cas pour un ordinateur.

(1) Mais rappelons qu'il n'y a aucun lien privilégié : on peut ainsi l'intégrer au primaire, dans les activités d'éveil de manière à traiter, par exemple, les résultats d'une enquête; l'essentiel étant de résoudre par la programmation, ou plus généralement par l'informatique, comme nous le verrons dans la remarque finale, un problème posé dans la classe.

> L'utilisation de l'ordinateur comme moyen de résolution de problèmes entraîne diverses habiletés chez les étudiants. Leur ardeur au travail n'est pas inhibée par l'ordinateur. Au contraire, ils entrent plus profondément dans les problèmes qui étaient autrefois trop longs et trop ennuyeux pour être traités en classe.
> Ils utilisent leur temps et leur énergie pour étendre leur connaissance de concepts mathématiques plutôt que pour étudier par coeur des matières liées à l'informatique.
>
> Par exemple, quand on introduit, en statistiques le concept de la probabilité, il était impossible de réunir suffisamment de données pour faire la démonstration de choses telles que le lancer de dés ou le tirage de cartes et leurs représentations graphiques sous forme de courbes en cloches, d'histogrammes ou de graphiques en tarte. Il est relativement simple d'écrire un programme de moins de 15 lignes qui réalise un million de jets de dés et imprime la courbe des résultats sur l'écran..."

2.4. Une dernière conséquence porterait d'une part sur l'intensification des recherches sur les processus sous-jacents à la programmation et d'autre part sur **la mise au point d'outils d'aide à la conceptualisation** de notions que ces recherches auraient isolées comme des points cruciaux dans la maîtrise de la programmation.

Plusieurs chercheurs ont ainsi montré que la source de certaines difficultés résidait dans une mauvaise représentation du fonctionnement de la machine (CANNARA, cité dans [1] p.177, MAYER [10]).

MAYER, par exemple, fait l'hypothèse que les étudiants apprennent un langage de programmation en reliant le rôle des instructions à des modèles physiques ou mécaniques. D'où l'idée de fournir aux étudiants un modèle concret du fonctionnement de la machine (au niveau des instructions du langage seulement(1)) leur permettant d'intégrer et de mieux assimiler ces notions, en se représentant, à l'intérieur d'un modèle simple et visuel, le rôle des diverses instructions. Les modèles utilisés pour initier, d'une part à des instructions de type BASIC et d'autre part à la gestion des fichiers ainsi que l'explication qui en est donnée aux étudiants sont présentés en annexe 1.

(1) Il ne s'agit pas de décrire le fonctionnement du hardware mais de proposer une "notional machine", c'est-à-dire une machine simplifiée idéalisée dont les propriétés dépendent des concepts utilisés par le langage.

MAYER montre que les étudiants qui disposent de ces modèles assimilent mieux les diverses instructions vues dans la suite; leur score à des tests de transfert (à d'autres domaines informatiques proches) est également plus élevé.

DU BOULAY a également mis au point un modèle semblable à celui présenté ci-dessus pour l'apprentissage du LOGO; ce modèle permet à l'apprenant de "jouer à la machine" (placer des données dans l'espace de travail, lire au clavier,etc.). Il oppose ainsi deux approches de l'apprentissage de la programmation.

> "La première peut être appelée "l'approche boîte noire"; dans celle-ci, l'utilisateur considère la machine comme une boîte noire dans laquelle il peut introduire des données et instructions et en recevoir la réponse comme par magie. Les mécanismes utilisés par l'ordinateur restent cachés à l'utilisateur.
> De tels utilisateurs sont capables d'utiliser des algorithmes qui marchent, c'est-à-dire qui fournissent les résultats escomptés. Cependant, ils ne sont pas capables de relier les commandes à une compréhension de ce qui se passe à l'intérieur de la boîte noire.
> Dans la seconde appelée "approche boîte de verre", l'étudiant tente de comprendre ce qui se passe à l'intérieur de la machine. Chaque instruction aboutit à des modifications dans la machine; celles-ci peuvent être décrites et comprises. Le niveau de description ne doit pas être celui des entrailles de la machine; l'utilisateur ne doit pas devenir un expert en électronique mais il y a un niveau spécifique de description que MAYER appelle celui des transactions."[10 p. 126]

De nombreuses autres stratégies et dispositifs ont déjà été définis et testés; citons part exemple :

- l'utilisation de métaphores (cf métaphore de "l'exécutant comme gestonnaire de casier"[13]);
- la mise au point de minilangages rudimentaires et adaptés aux apprentissages visés (ex. : définition de compétences limitées d'un robot pour débutants en LOGO [3], ELOGO, [11]);
- illutration du fonctionnement d'un programme : un pointeur montre le déroulement de l'exécution, les contenus actuels des variables apparaissent à tout moment (BIP de BARR, BEARP et ATKINSON cités dans [11] p.242; le jeu de primitives chargeables "DEBUG" dans le LOGO LCSI sur le Macintosh permet une représentation similaire);
- livre ordinateur (MUPY; exercice, b.d., petit ordinateur simplifié);

- la mise au point d'interventions explicites pour favoriser les activités de planification : DALBEY [14] a ainsi construit et testé une série de cinq leçons visant à encourager les étudiants à donner plus d'importance à l'étape intermédiaire entre la définition des spécifications et le codage du programme. Il utilise à cette fin une méthode de représentation graphique de l'organisation logique du problème ("structure diagramming");
- le jeu de rôle dans lequel les différents enfants prennent en charge des fonctions spécifiques de la machine de manière à se représenter ce qui se passe dans la machine lors de la programmation. Cf annexe 2;
- etc.

D'autres moyens restent à inventer pour faciliter la maîtrise de ces outils informatiques.

Une dernière remarque : nous avons essentiellement parlé jusqu'à présent de programmation; or, il apparaît de plus en plus clairement que d'autres activités informatiques sont également enrichissantes au point de vue cognitif. L'utilisation de progiciels (traitement de texte, gestion de fichiers,etc.) notamment semble se révéler un excellent moyen pédagogique en tant qu'outils d'entraînement au traitement d'information (cueillette , mise en commun, organisation, analyse et communication des informations).

Un exemple de ce type d'applications est le projet de l'équipe de JAN PALKIEWICZ de developper par l'utilisation de banque de données la pensée formelle d'adolescents. La consultation future les encourage à :

1. bien analyser leur problème, le transformer en concept clé et établir des relations entre eux; établir des hypothèses;
2. comparer leur résultat avec le thésaurus; améliorer leur définition conceptuelle;
3. définir des stratégies de recherche .

La consultation est suivie des deux dernières étapes : l'analyse des documents et la communication des résultats.

Nous reproduisons ci-dessous un extrait des objectifs de ce groupe de travail car il est frappant de constater qu'il ne soit pas éloigné des objectifs fondamentaux assignés à l'apprentissage de la programmation dans les programmes que nous avons parcourus.

> "Ce projet vise le développement mental par l'utilisation systématique des banques de données. Il s'agit moins de la technique proprement dite d'accès aux données par l'utilisation d'une procédure déterminée, que de la conception d'une méthode de travail intellectuel fondée sur l'utilisation de banque de données. Ce projet cherche donc à donner aux élèves de niveau secondaire des habitudes d'un véritable travail intellectuel, par la confection d'un champ conceptuel touchant une problématique particulière, l'investigation systématique de diverses sources d'information et l'analyse de documents recueillis."[12]

BIBLIOGRAPHIE

[1] B. DU BOULAY and T. O'SHEA, Teaching novices programming, in Computing Skills and the User Interface édited by M.J. COOMBS and J.L. ALTY, Academic Press, 1981.

[2] P. MENDELSOHN, L'analyse psychologique des activités de programmation chez l'enfant de CM1 et CM2, Enfance, n° 2-3,1985.

[3] C. LABORDE, N. BALACHEFF, B. MEJIAS, Genèse du concept d'itération : une approche expérimentale, Enfance, n° 2-3,1985.

[4] R. SAMURCAY, A. ROUCHER, De "faire" à faire faire" : planification d'actions dans la situation de programmation, Enfance, n° 2-3, 1985.

[5] PEA R.D. and KURLAND D.M., On the cognitive effects of learning computer programming, New Ideas in Psychology, N. Y., Pegamon Press, Vol. II, 2, 1984.

[6] BIDEAULT A., Procédures d'enfants de CE2 dans une tâche de construction de parcours, Enfance, 2-3, 1985.

[7] WEISSBERG J.L., La maladie infantile de l'informatique, Education Permanente, 70-71, 1983.

[8] HOC J.M., L'étude psychologique de l'activité de programmation, Technique et Science Informatique, Vol. I, n°5, 1982.

[9] JONES K.L. et LAMB C.E.,Programming in mathematics education : ce essential component in the developpment of reasoning skills, in Computers in education, K. DUNCAN and D. HARRIS (eds) Elsevier Science Publishers B.V., 1985.

[10] MAYER R.E., The Psychology of How Novices Learn Computer Programming, Computing Surveys, Vol. 13, n°1, mars 1981.

[11] DU BOULAY B., O'SHEA T., MONK J., The black box inside the glass box : presenting computing concepts to novices, Int. I. Man-Machines Studies, 14, 1981.

[12] PALKIEWICZ J., Développement de la pensée formelle des adolescents par l'utilisation systématique de banques de données, Congrès Cognitiva 85, Paris, 4-7 juin 1985.

[13] DUCHATEAU CH., Programmer !, Wesmael Charlier, 1983.

[14] DALBEY J., TOURNIAIRE F. et LINN M.C. Making programming instruction cognitively demanding : an intervention study, Journal of Research in Science Teaching, vol. 23, n°5 (pp.427-436)