Peut-on enseigner un langage que personne ne parle ?

Paul Delannoy,
Institut Universitaire de Formation des Maîtres (I.U.F.M.) des Pays de la Loire,
4 chemin de Launay-Violette,
44072 Nantes Cedex 03,
tél. 33-40163001
fac-similé 3040
et
Laboratoire d'Informatique de l'Université du Maine,
avenue Olivier Messiaen, B.P. 535,72017 Le Mans Cedex,
France.

Résumé :

De nos participations aux mises en place du langage Logo dans les écoles françaises jusqu'à nos actuels travaux en Robotique Pédagogique, nous avons sans cesse croisé, sous différentes formes, la question de l'apprentissage d'un langage de programmation.

Que ce soit avec Logo (cf. l'article publié dans le bulletin EPI n° 66 en avril 1992), avec la définition de formations d'enseignants «Découvrir l'ordinateur au travers d'un intégré», ou, plus récemment encore, avec le pilotage de robots pédagogiques sous Hypercard® ou Toolbook®, un aspect de cette question nous apparaît comme incontournable : c'est celui de la représentation implicite qu'a -ou que se fabrique- l'apprenant de l'ordinateur et de ses fonctionnalités de base, et des relations que cette représentation entretient : d'une part avec des apprentissages fondamentaux (mathématiques, langue maternelle, langues mortes ou langues vivantes,…), d'autre part avec ses apprentissages liés à l'utilisation de l'informatique (traitement de textes, base de données, robotique, logiciels éducatifs, multimédia,…).

Au point de départ, avec Logo, nous adhérions à l'idée, conséquence des affirmations piagétiennes sous-jacentes à la démarche proposée, que cette représentation se construirait «comme les autres» ; après avoir vu que les notions informatiques de test, de séquence, de variables, nécessaires à l'approche de tout langage de programmation, restaient confinés au rôle d'outils de travail, nous avons recherché dans le pilotage de robots, considéré comme généralisant l'idée de la tortue, une illustration directe du rôle du programme, et de ces notions, dans l'activité proposée à l'apprenant (Actes du colloque de Robotique Pédagogique de Liège juillet 1993). Notre point de vue actuel est qu'il existe deux raisons pour que ces notions soient aussi peu explicitement abordées : la plus évidente est que dans la majorité des cas où un ordinateur intervient en formation/apprentissage, la question de leur explicitation n'est pas pertinente en soi ; la seconde est que, si l'on se réfère à l'apprentissage d'un langage d'ordinateur, personne -mis à part quelques spécialistes- ne «parle» un tel langage.

1) La réponse est NON

Si la question est prise au sens strict, les éléments de réponse disponibles se trouvent auprès des enseignants de langues, maternelles ou étrangères, et ceux d'informatique et de programmation. Ceux-ci utilisent la phonétique, l'apprentissage lexical et l'analyse grammaticale, entre autres outils. Ceux-là n'ont au mieux que les deux derniers à leur disposition !

Dans le cas de l'apprentissage de la programmation les notions de lexique, de grammaire sont bien présentes, mais la relation au feed-back de l'ordinateur ne peut pas être directe, et nécessite, du fait de l'absence de forme parlée, l'intervention du formateur : pour traduire le bug, le message d'erreur,… Une situation paroxystique de cet état de fait est très bien décrite par Ch. Duchâteau lorsqu'il épingle les exemples d'exercices de programmation présentés à des débutants. (Actes du colloque de Liège cf. supra).

L'absence d'une représentation préalable de l'ordinateur rend ainsi caducs les efforts que l'on pourrait entreprendre pour se baser sur des descriptions de situations, comme on le fait en langue étrangère.

2) La réponse est "PEUT-ÊTRE“

Si l'on s'intéresse au double aspect d'un langage, informatique ou naturel, que constituent ses capacités de codage d'une part, et ses capacités d'analyse d'une situation donnée d'autre part, on verra poindre différemment des autres le cas où l'apprentissage du langage informatique est, volontairement ou non, lié au codage d'une situation pour laquelle un langage humain a pu (ou peut, ou doit) se constituer ; par exemple dans le cas du technicien abordant la programmation d'un contrôleur ou d'un processeur (langage machine) où le code-langage à apprendre est directement lié aux fonctions réalisables par l'électronique, et où les objectifs assignés à l'ensemble en cours de réalisation sont exprimés dans un (meta)-langage courant ; par exemple aussi dans toutes nos activités faisant intervenir des micro-robots, micro-mondes extérieurs dans lesquels la **description** du robot réalisé reste au centre de toutes les activités, y compris celle de la programmation des mouvements (Thèse de Pascal Leroux en cours au laboratoire : Roboteach, voir les actes du colloque de Liège déjà cités).

La situation micro robotique nous permet, en plus, grâce à l'utilisation de systèmes comme Hypercard® ou Toolbook®, d'intégrer une représentation de l'objet programme à l'ensemble de l'activité. Cette représentation constitue d'une part une aide à l'apprenant dans sa découverte et ses essais-erreurs, mais aussi une initiation à la programmation, voire un apprentissage d'un langage (si la représentation en utilise volontairement un particulier), et possède sur la situation précédente -le langage machine- l'avantage de mettre en évidence la structuration des programmes, plutôt réduite au rôle de facilité disponible dans l'initiation à la programmation à partir de circuits électroniques.

L'utilisation dans un tel contexte de langages algorithmiques assurera sans doute un passage efficace et harmonieux de la représentation par codage (quel capteur, quel moteur,…) de l'objet fabriqué à la représentation analytique de ses mouvements par la découverte de l'efficacité de l'analyse algorithmique.

C'est pourquoi, dans ces situations, la réponse à la question posée reste ambiguë, et le seul intérêt que représente le choix de tel ou tel langage de programmation réside dans sa "transparence" et sa capacité à s'intégrer comme une des «briques» de la situation d'apprentissage, en ne laissant voir éventuellement que les difficultés conceptuelles (modes de fonctionnement : éditeur, exécuteur, par exemple) et non les seules difficultés langagières (lexique, syntaxe) ; ainsi, comme nous nous y attendions un peu, Logo utilisé dans nos formations de robotique pédagogique n'y prend pas de valeur pédagogique particulière : seules l'existence du micro-monde (robot) à piloter et l'adéquation de ce langage à une telle situation en font un outil de choix. Mais dès lors que nous cherchons à intégrer dans un système multimédia un support à une telle formation (second objectif de Roboteach, cf. supra), les objets-programme sont décrits, a priori, dans un langage général algorithmique, et les difficultés évoquées ci-dessus peuvent être également aplanies, puisque le programme et le langage deviennent clairement les objets que manipule l'ordinateur, par construction (Cf. l'atelier "Apple et micro robotique" présenté à Liège en juillet)

2) La réponse est "OUI“

C'est à ce stade de la réflexion que ce travail rejoint le discours développé par Charles Duchâteau : il est tentant, et à l'expérience efficace, de généraliser la situation évoquée ci-dessus pour tout utilisateur de logiciel, traitement de textes, tableur, etc.… Bref, d'apprendre un langage informatique à l'utilisateur.

En effet, la situation de l'utilisateur débutant se révèle très vite paradoxale : les systèmes d'interface à fenêtres et souris tendent à réduire l'apprentissage langagier au lexique des commandes disponibles - voire encore souvent à celui des "raccourcis-claviers" (toujours plus efficaces sur des claviers … étendus) - et la syntaxe à l'apprentissage de deux ou trois gestes ; la demande de l'apprenant est souvent centrée sur la facilité de réalisation attendue par lui de la part d'un tel système. Mais, c'est le cas particulièrement avec les systèmes de dessin, la syntaxe utilisée et l'analyse effectuée par le programmeur des situations obtenues par l'apprenant, ainsi chassée a priori, revient au galop : comprendre comment le programme comprend les commandes permet d'assembler logiquement celles-ci, de les répéter efficacement, bref de les apprendre plus totalement (et, sans doute, de façon translatable à un logiciel pas trop différent dans ses analyses de situation) ; pensez, par exemple, au publipostage en traitement de textes.

Prenons le cas d'un dessin composé de deux éléments qu'il faut aligner : le code manipulé par l'ordinateur, qui représente pour lui ces objets, s'articule suivant une analyse à deux

entrées : celle de l'utilisateur, qui voudra sans doute essayer de réaliser l'alignement visuellement, et celle du programme, qui "réussira" mieux si l'utilisateur apprend à faire appel à lui par sélection et déclenchement de la bonne fonction. Nous prétendons que l'utilisateur qui possède la notion d'**analyse logique** (au sens que l'on donne à ce terme dans l'apprentissage de sa propre langue) comprend mieux et plus vite l'efficacité de la démarche passant par cette étape : un langage artificiel que personne ne parle, hormis l'ordinateur, est «caché» derrière chaque situation de ce type.

CONCLUSION

Nous ne proposons pas de faire un programmeur de chaque utilisateur, mais, en abordant ainsi la question, chacun pourra au travers de ses rencontres avec l'ordinateur, se forger une représentation efficace de son fonctionnement, être plus apte à choisir l'outil logiciel le plus adapté à ses besoins, etc.…

On peut illustrer fortement cette position par la négative : supposons qu'un linguiste intéressé par la programmation se persuade qu'aucun traitement de textes disponible n'est apte à servir de support à ses enseignements de langue, et qu'il pense qu'en programmant lui-même sur la base de sa propre analyse, il puisse réaliser LE traitement de textes qu'il lui faut. Nous pensons d'une part que son programme sera deux fois plus difficile à aborder que ceux qu'il refuse pour quelqu'un qui ne connaîtra ni la linguistique ni l'informatique, et d'autre part qu'un tel programme renforcera plus la compréhension des règles de cette discipline que celles qui régissent l'analyse informatique.