MICRO-MONDES ET MACRO-PRIMITIVES

François BOULE

L'une des voies possibles pour une découverte significative de l'informatique consiste à aborder la programmation. Mais non pas d'un point de vue strictement technique et utilitaire car aucun langage de programmation ne peut sérieusement prétendre à un statut comparable à celui des langues naturelles, maternelle ou étrangères. La programmation n'est pas un but, pas plus que le stylo n'est la raison d'être de l'écrivain. Par contre elle met en jeu des modes de pensée particuliers mais non strictement propres à cette activité ; ceux-ci permettent d'éclairer – de l'intérieur – les modes d'action par l'informatique. lis retentissent aussi sur les représentations que l'on est conduit à se faire du réel, et des interactions multiples dont l'objet technique est le lieu et le signe.

LOGO, dans cette perspective, occupe une position singulière qui tient à son origine, aux circonstances de son emploi jusqu'à ce jour, et au fait que se révèlent à son propos, depuis peu, beaucoup de questions importantes jusqu'ici inaperçues ou éludées.

LOGO : DIFFÉRENTS ABORDS

LOGO est d'abord un instrument d'exploration du développement de la pensée de l'enfant : il a été conçu dans une perspective constructiviste. C'est à dire qu'un usage systématique, et en particulier dans le cadre scolaire excède un peu cette perspective originelle. Une seconde phase de développement a succédé à celle-là. LOGO a traversé les Grands Lacs, puis l'Atlantique et quelques continents, sous forme de système "clavier-écran". Pendant cette phase, centrée sur l'usage de la tortue-écran par des enfants d'une dizaine d'années, la démarche est relativement spontanéiste : l'adulte est "personne-ressource", observateur et intervenant discret. La tortue paraissant géométrique, une première dérive l'a conduite chez les enseignants de mathématiques, qui y ont vu un mode original d'appropriation de concepts mathématiques.

Une autre dérive, issue de celle-ci a mené vers des enfants de plus en plus jeunes, cette fois-ci sous la forme originelle (ou presque) d'un mobile programmable, se déplaçant sur le soi.

A l'heure actuelle, on peut distinguer dans la pratique de Logo avec des enfants la présence explicite ou non, mais rarement exclusive de trois attitudes :

La démarche "ascendante" consistant à partir de primitives (supposées sans équivoques), à construire des objets, à les conserver sous forme de procédure, à les réutiliser dans des objets plus complexes, etc.

La démarche "descendante" qui s'inspire, non des possibilités d'action à partir des outils disponibles, mais de la conception d'un projet ; en sacrifiant les détails superflus et en repérant une structure d'ensemble, en décomposant les tâches (éventuellement en les partageant), jusqu'à rencontrer des "objets simples" connus. On procède ainsi dans l'inconnu du complexe au simple. Ces deux démarches, les choix qu'elles impliquent, les difficultés qu'elles rencontrent seront examinées ailleurs. C'est la troisième qui est l'objet de la présente analyse.

macro-primitives

Il est très aventureux de prétendre que les primitives du langage ont quelque raison décisive (logique ou psychologique) d'être préférées à d'autres. L'argument de la simplicité, en particulier, n'est pas convaincant ; il y a longtemps que l'on a abandonné d'initier l'étude de la géométrie par le Point et la Droite. Un autre parallèle, avec la lecture cette fois-ci, est tentant : dans sa conquête de la parole, puis de la lecture, ce que l'enfant saisit, ce sont des unité de sens. Les primitives ne sont qu'un alphabet. Ce que l'on vise, c'est à dire l'activité organisatrice de la programmation risque d'être dilué, différé, pulvérisé si l'écart (donc la durée) est trop grand entre les "éléments" et les projets. De plus l' "espace de la tortue" est vide : cette absence de contexte suggestif nuit â la construction des significations. Le choix d'un projet parait gratuit et il advient que le cours de sa réalisation soit influencé par ce que l'on sait pouvoir faire ou ne pas faire. On restreint donc cette liberté désertique en peuplant l'espace d'objets. Cet ensemble cohérent (souvent baptisé MICRO-MONDE) suscite des contraintes et suggère des actions possibles. C'est la logique de ces actions qu'il s'agit de programmer. Ce sont alors des éléments de sens.

ÉCLUSE ET ASCENSEUR

Les deux exemples les plus connus (notamment par leur diffusion en 1985 dans les "valises" I.P.T.) sont "ECLUSE" et "ASCENSEUR". Il s'agit, d'après la documentation jointe, de "simulation graphique d'objets techniques". Les procédures de dessin et un certain nombre de transformations élémentaires sont fournies sous forme de "macro-primitives". Le logiciel doit permettre à l'enfant de construire l'algorithme de fonctionnement de l'objet technique (à l'aide de ces procédures) ; par exemple l'ouverture des portes (ou des vannes) amont, ou aval ; le remplissage. du bassin, le déplacement du bateau sont des procédures fournies. Il pourrait s'agir d'initiation à la programmation si l'usage de **ces** procédures ne supposait pas déjà un maniement relativement aisé du langage, et la nécessité de chercher dans la définition des procédures "cachées" leur liste, leurs paramètres et leur syntaxe. Cette démarche, et particulièrement les exemples ci-dessus soulèvent trois séries de difficultés qu'il importe d'analyser et d'essayer de franchir.

1. Simulation

S'agit-il d'une simulation ? Le document accompagnant ÉCLUSE mentionne que "le premier travail consiste à trouver une schématisation et non une modélisation". Un modèle est un objet abstrait (un ensemble de relations, éventuellement temporelles) dont on fait l'hypothèse de l'adéquation au phénomène réel. Un schéma est un mode de représentation, c'est à dire une situation intermédiaire entre la réalité et le modèle. Un schéma seul ne permet pas de tester une hypothèse. S'il s'agit pour les enfants de constituer une simulation, comment celle-ci sera-t-elle validée ? De deux choses l'une : ou bien le logiciel renseigne sur la validité, ou bien celle-ci s'assure par référence à une représentation que l'enfant a construit par ailleurs. On est bien sûr ici dans le second cas : il est nécessaire de connaître et de maîtriser le fonctionnement de l'objet, c'est à dire d'avoir une idée du modèle.

Une simulation est asservie par les contraintes du Réel (pesanteur, lois d'écoulement, flottaison, résistance des solides etc.). Ces contraintes sont-elles ou non prises en compte dans les procédures fournies ? S'il s'agit de simuler le fonctionnement, on est fondé à le croire : remplir, vider, ouvrir-porte, avancer-bateau paraissent signaler des actions réelles. Sinon il convient de simuler, non seulement le fonctionnement, mais aussi les lois de la physique. Malheureusement, on s'aperçoit que l'ouverture d'une porte entre deux niveaux différents n'entraîne pas de

conséquence spectaculaire. Dés lors, on peut imaginer de construire un prédicat qui teste les niveaux afin de "rendre" la possibilité d'ouvrir la porte. On lie ainsi les phénomènes dont les représentations sont, dans le logiciel, indépendantes.

S'il était ainsi, et s'il l'annonçait, le logiciel pourrait être cohérent. Mais ce n'est pas le cas : le bateau étant en amont, si l'on vide le bassin, le bateau perfore le bief amont et s'enfonce ; ici deux phénomènes indépendants sont mystérieusement liés. C'est la référence (incomplète) à une simulation qui est responsable de ces incohérences.

Si l'on veut faire construire l'algorithme du fonctionnement (qui est relativement simple), le logiciel doit prendre en charge les contraintes réelles (ce qui n'est pas tout à fait simple). Par contre si les objets fournis sont élémentaires (aides au dessin), ou bien on intègre les contraintes réelles (c'est à dire qu'on limite la validité de "avancer", "remplir" etc.) pour jouer de procédures synthétiques, ou bien on réalise une animation graphique en perdant la signification des procédures intermédiaires. En résumé : ou bien les procédures "remplir-", "vider-" ont une signification proche de celle qu'on leur connaît, ou bien les mots pour les désigner sont impropres. De cela l'utilisateur n'est pas averti. Sans doute aussi, s'il s'agit de procédures référent au réel, devrait-on souhaiter disposer des procédures inverses, afin de pouvoir corriger sans revenir au départ c'est précisément là qu'est le jeu (au sens de : marge de manoeuvre) qui fait l'intérêt d'une simulation. Enfin, une simulation, à moins d'être un exercice de renforcement, doit proposer plusieurs situations de départ (aléatoirement, ou au choix).

Un tel système entretient ainsi la confusion entre plusieurs niveaux :

- les éléments de la situation. Il s'agit par BATEAU-AMONT, PORTE-AMONT-FERMÉE... de réclamer des éléments du paysage (images) ;
- les opérations sur ces objets. Certaines sont licites, d'autres illicites, d'autres non-signifiantes. Avancer le bateau est licite si la porte est ouverte. Ces opérations peuvent être désignées par un infinitif. Par contre, dans une simulation, le remplissage du sas est une transformation conséquente et non commandée directement. Impératif et infinitif sont alors impropres ;
- parmi les opérations licites, quelques enchaînements constituent l'algorithme cherché.

2. Graphique

À vouloir surcharger un projet de programmation d'une intention technologique d'origine extérieure, on crée l'équivoque et on contrefait une simulation. Par contre il est certainement souhaitable de fournir des outils plus puissants que les primitives habituelles, quitte à les interroger dans un second temps, après s'y être accoutumé. L'exemple le plus simple, à destination des jeunes enfants est un °Mini-Logo" qui est une re-définition du clavier évacuant les paramètres numériques. Il ne s'agit là que d'adopter une notation abrégée pour un ensemble restreint d'actions élémentaires, et non d'objets plus significatifs. Mais si l'on fait intervenir des objets significatifs (fleurs, maison, carré, avion...) on se heurte à l'opposition local/global.

D'abord parce qu'une forme (par exemple un CARRE) n'est pas consciemment lue comme un parcours ; alors qu'une suite de procédures doit impérativement tenir compte des états (position et orientation) d'entrée et de sortie. La figure l, ci-dessous, est perçue globalement comme un carré, sans que rien n'appelle mention d'un trajet continu pour la parcourir, qui commence ici ou s'achève là. C'est pourquoi il est tout à fait impropre de baptiser CARRE une procédure telle que :

```
REPETE 4 [ AV 10 TD 90 ]
```

DESSINE-CARRE serait mieux dire, mais il en existe bien d'autres, et qui ne sont pas équivalentes, par exemple :

```
AV 10 TD90 AV10 TD90 AV10 TD90 AV10
REPETE 4 [TD 90 AV 10 ]
REPETE 6 [AV 10 TD 90]
```

sans compter toutes celles qui ne tracent pas un trait continu, celles dont le départ ou l'arrivée est au centre etc.

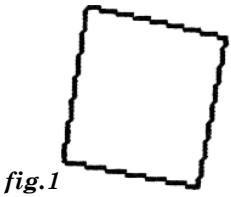


fig.1

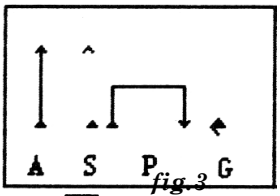
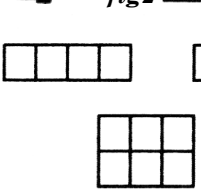


fig2

fig.3

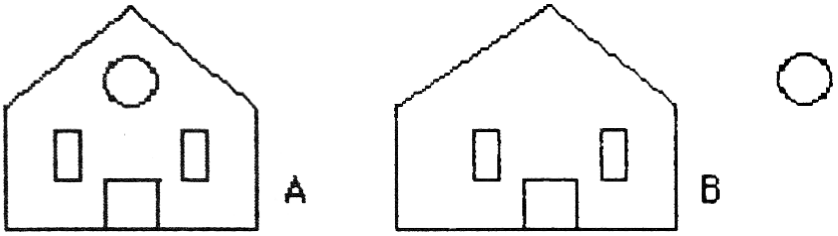


Le cadre de la figure 2 propose quatre "primitives" pour lesquelles sont indiqués les états de départ et d'arrivée. S est un "saut", P un "pont", G est un pivotement de 90° à gauche. Avec ce seul système, on peut se proposer de réaliser des dessins du type de ceux que présente la figure 3. Un tel système entraîne moins à une programmation structurée "descendante" qu'à un attention minutieuse aux "recollements" d'une procédure à la suivante.

Sauf précaution expressément prise les parties d'un programme tel que

```
POUR MAISON
FACADE TOIT PORTE FEN-DROITE FEN-GAUCHE LUCARNE
FIN
```

ont peu de chances d'être commutables. C'est donc dire que "TOIT" ne peut se contenter de faire le toit sans tenir compte de ce qu'elle est précédée par la façade et suivie par la fenêtre droite. C'est là que se rencontre encore l'opposition GLOBAL/LOCAL. L'utilisation de macro-primitives devrait libérer des contingences dilatoires et rendre ainsi un projet plus directement réalisable. La conception d'un projet (la perception même de l'image) s'accommode mal d'un point de vue strictement local. On a voulu voir dans cette contrainte vertu. Mais il ne devrait pas échapper que si cette contrainte a l'intérêt de susciter un autre regard sur l'organisation de l'espace, elle n'est formatrice (au mieux) qu'en tant qu'une règle du jeu parmi d'autres possibles.



Dans [A], il s'agit de dessiner une lucarne ; sa position est fortement liée aux autres éléments (symétrie). En [B], le cercle représente le soleil. Seul importe qu'il soit disjoint de la maison, assez loin. On le définit par rapport à l'image globale, et l'on aimerait pouvoir sauter de la maison à cet élément, chacun étant perçu dans sa relative indépendance. Au point de vue de la programmation, il est d'intérêt nul de se préoccuper du passage de la "sortie" de la maison, à l'"entrée" du "soleil".

La Tortue de sol, pour ce qui la concerne, est bien sûr strictement gouvernée dans une perspective locale, il est impossible d'échapper à la séquentialité de fait. Par contre, sur l'écran, on pourrait s'en tirer de deux façons : soit en utilisant le crayon optique pour "sauter" d'un point courant à un nouveau point de départ (c'est à dire en passant d'un repérage local, polaire, au repérage global, cartésien) ; soit en "déposant" de place en place des "noms de lieux" qui permettraient de repartir d'une position déjà atteinte sans rebrousser chemin. Mais l'on voit que réapparaît ici un problème déjà rencontré : s'il s'agit d'aide au dessin, il existe des logiciels "transparents" plus efficaces pour lesquels les tâches de programmation sont si réduites et peu structurées qu'on peut les dire absentes. L'écriture en Logo de telles aides peut contribuer à montrer le fonctionnement d'objets de cette sorte (simplifiés). Mais il ne s'agit plus alors d'initiation à la programmation : une certaine aisance dans le maniement du langage est requise pour découvrir ces objets inspirés de logiciels-outils ; on peut citer par exemple B.D.L0G0, ou DESSIN, produits à Reims.

Mais peut-on parler d'algorithme ou de programmation s'il ne s'agit que de répondre à une situation singulière (comme de réaliser un dessin) ? Ces concepts-là paraissent plutôt relever de méthodes visant à résoudre des classes de problèmes. C'est pourquoi l'emploi d'un Bigtrak ou d'une calculette peut difficilement passer pour une véritable activité de programmation : il n'existe pas de représentation indépendamment des valeurs numériques occurrentes ; il n'y a ni paramètres, ni tests.

L'activité n'est alors rien d'autre que le repérage d'une linéarisation (d'un objet ou d'un processus), et la reconnaissance d'un découpage de ce "parcours" en unités identifiables. On voit que c'est une activité praticable, moyennant quelques exigences, dans la langue naturelle, et qui relève essentiellement de la maîtrise de cette langue. C'est pourquoi il faut examiner maintenant quelques questions relatives au langage.

3. Langage

Ces questions pourraient se réduire à celle-ci : quel est l'intérêt d'utiliser une langue artificielle ? ou plutôt : cet intérêt n'est-il que de permettre (actuellement) de commander des machines, ou bien les langages de programmation permettent-ils plus commodément que les langues naturelles de développer certaines activités logiques ? On pourrait prendre ici l'exemple de l'algèbre (ou celui de la logique aristotélicienne) la traduction algébrique d'une situation permet d'opérer des transformations selon des règles formelles ; ces transformations "machinales" mettent temporairement en retrait la signification des expressions utilisées. C'est précisément grâce à ce retrait temporaire que s'exprime la puissance du calcul algébrique (ou de la logique formelle). Il faut donc examiner les aspects sémantiques et syntaxiques soulevés par l'emploi des "macro-primitives".

A. L'aspect le plus superficiel concerne la syntaxe "résidante" dans le langage. Si les macro-primitives doivent constituer une aide pour l'utilisateur, il n'est pas admissible qu'elle s'expriment en des termes et selon des expressions qui bousculent la langue française. Ainsi peut-on difficilement admettre des expressions telles que `PORTE [AVAL OUVERTE]` ou `BASSIN [VIDER]` ou `BATEAU [AVANCE ECLUSE AMONT]`. On ne peut espérer ainsi ni clarifier la pensée, ni établir le bon usage de la langue. Le rôle des crochets n'est aperçu que si l'on connaît LOGO, et devrait pouvoir être évité ; l'action est signalée tantôt par un infinitif, tantôt par un indicatif, ou encore par un participe ; l'ordre des mots qui pourrait paraître arbitraire, n'est quand même pas indifférent... Il est possible d'éviter de telles incohérences moyennant, il est vrai, un "coût" assez élevé de programmation préalable. Mais s'il s'agit de proposer un système d'initiation à l'algorithmique (ou à la programmation), et qu'on ne veuille supposer aucune familiarité préalable avec Logo, il faut permettre une expression plus proche de la langue naturelle que ne le sont les langages de programmation. C'est alors le système qui doit prendre en compte la plasticité d'expression qu'un langage évolué ne

tolère pas. Mais n'est-on pas ramené à un écueil classique de l'E.A.O., celui de l'interprétation des réponses "naturelles" ?

On sait assez que la critique la plus légitime à l'égard d'Apprentissages Assistés par Ordinateurs (sous leur forme actuelle), concerne non tant la rigidité du scénario ou l'absence d'initiative de l'utilisateur (assez facilement contournables) mais le filtrage des réponses : ou bien les réponses admises sont en petit nombre (QCM), lapidaires, dépourvues de connotations et de variantes ; ou bien leur examen est réduit à un prélèvement d'unités de sens (mots-clés) avec un risque non négligeable d'erreurs d'interprétation. Il n'est pas certain qu'une initiation à la programmation (dans une perspective logistique) échappe à cette critique. Les langages de programmation ont d'excellentes raisons de ne pas ressembler à des langues naturelles : ce qui est commodité pour l'Interpréteur est une contrainte pour l'interprète humain et réciproquement. La notation "infixe" en est l'exemple le plus simple ; l'expression PH PREM SP :A PH ITEM :B :C :D facilement analysable syntaxiquement n'est pas immédiatement saisie comme unité de sens. La raison en est qu'un codage de type algébrique supporte très bien un nombre fini quelconque de niveaux coexistants (ici 4 enchâssements et 3 niveaux) alors que la pratique langagière et l'intuition recourent dans ce cas à des "procédures refermées" de façon à ne pas saturer l'espace de travail (mémoire à court terme). Ainsi reste-t-il trois possibilités :

- ou bien les macro-primitives sont intégrées dans une structure tout à fait "plate" (type récit) et l'algorithme est dépourvu d'intérêt,
- ou bien la structure est plus riche et s'inspire de l'expression logique de la langue naturelle, mais le système d'interprétation est excessivement touffu,
- ou encore (c'est la solution implicitement retenue dans les exemples actuels) on adopte les structures logiques telles qu'elles sont dans le langage, et l'on en suppose la connaissance préalable (ou "naturelle", hypothèse encore moins acceptable, mais peut-être pas exclue).

B. La comparaison, évoquée au début, avec la conquête de la parole ou de la lecture est probablement une illusion. Ce que l'enfant distingue et lit, ce sont d'abord des unités de sens. C'est à dire la première articulation. Il découvre la seconde plus tardivement ainsi que la "superstructure" grammaticale. Rien de semblable avec un langage artificiel (algèbre, logique ou langage de programmation) ; la structure est première. C'est précisément parce que la syntaxe est rigide que l'on est

autorisé à congédier temporairement le sens, ce qui est à l'origine de la possibilité et de la puissance du "calcul". Il semble dès lors paradoxal que l'on s'efforce de le maintenir ou de le restaurer par l'emploi de macroprimitives.

Deux causes au moins sont à l'origine des difficultés spécifiques dans l'apprentissage des mathématiques ; l'une est l'exigence de preuves, c'est à dire d'une justification a priori et générale à l'exclusion de toute autre validation ordinaire ; l'autre est cette mise à l'écart temporaire des significations pour opérer sur des signes. Il n'est donc pas tout à fait surprenant qu'on retrouve ici ces deux difficultés. Il se peut que les macroprimitives aient un emploi, mais probablement pas comme support d'une initiation algorithmique autonome. En effet leur usage requiert alors la familiarité d'un langage dont elles intègrent les contraintes syntaxiques. Dans toute activité de résolution de problème, ou de programmation, il convient de distinguer :

- le type des objets avec lesquels on travaille,
- les types d'action que l'on peut exercer sur ces objets,
- les énonces que l'on peut produire sur ces objets,
- les articulations logiques des éléments entre eux.

Ces distinctions peuvent être opérées dans la langue d'usage ; c'est même le point de départ et le but de cette activité, si son objectif n'est pas simplement technique. Les macroprimitives sont alors les RELAIS (points d'ancrage, étiquettes...) qui vont garantir la règle du jeu : la validation (automatique) par la machine.

Il peut sembler que ces distinctions soient simplistes ou sommaires ; il est utile en effet qu'une théorie parvienne à les dépasser (cf. les "catégories" en mathématiques) et octroie aux "objets" un statut plus général. Mais elles sont un bon support pour l'intuition et de bonne propédeutique. En définitive elles déterminent le niveau inférieur (plancher) de l'analyse descendante ; à tout moment de cette analyse, on se demande quels objets interviennent (sans devoir en fixer par avance les paramètres), quelles procédures et quels prédicats peuvent leur être appliqués. C'est alors, et à la demande que ces constructions sont produites.

Exemple 1 : s'agissant de créer un pavage, 3 types interviennent :

- Le motif [M] C'est un objet géométrique qui peut être fourni ou donné à construire.

- L'engendrement, défini à l'aide d'un couple de vecteurs [U,V]
- Un test d'arrêt signalant que le motif 1 [M(XU+YV)] est entièrement hors de l'écran.

Exemple 2, s'agissant de productions ou de modifications de texte, interviennent :

- Les objets terminaux (mots ou phrases), stockés dans le programme et tirés au sort, ou fournis par l'utilisateur.
- Les règles d'engendrement ou de transformation (productions de C-grammaires, substitutions, conjugaison, mise au pluriel...).
- Les tests concernant genre, nombre, personne, désinence, exception, voire grammaticalité.

Exemple 3. Productions sonores. Les motifs sont constitués d'une ou plusieurs notes. Les transformations sont des engendrement à une ou 2 dimensions, transpositions, inversion, modification de rythme etc. On voit que si l'élaboration du cahier des charges concernant les procédures est la première phase de l'activité, elle peut être :

- orientée par les objets (disponibles ou projetés)
- orientée par les actions : quelle est la "grammaire" des actions possibles ? [approche constructive ou procédurale],
- orientée par les énonces : que peut-on dire des objets (donnés ou à construire) ? [approche déductive ou structurale].

En conclusion, ce qui est réellement visé est l'organisation par l'enfant d'un certain champ d'action (non vide). Si cette organisation lui échappe, soit par une référence subreptice à une simulation, soit par un dépouillement de ses moyens d'intervention, la tâche est égarée ou absente. Le champ graphique, s'il s'agit de construire des objets (et non de s'orienter vers la logique d'une situation) oppose des obstacles embarrassants. Il ne peut s'agir d'une véritable initiation à la programmation, mais plutôt d'un auxiliaire d'exploration logique, complémentaire et conséquent de l'analyse algorithmique en langue naturelle.

François BOULE