UNE EXPERIENCE D'ENSEIGNEMENT DE LA RECURSIVITE EN LOGO

C. DUPUIS, M.-A. EGRET et D. GUIN

We present here an experiment in a class with thirteen-fifteen years old students . Our aim was to elaborate a teaching about recursion which points out and splits difficulties of this notion . This research required an analysis of students productions to underscore these difficulties . The definition of specific criteria seems to be an appropriate tool for such a research .

Introduction

Cette étude s'intègre dans un ensemble de recherches de l'Institut de Recherche sur l'Enseignement des Mathématiques de Strasbourg liées à l'introduction de l'informatique dans le système scolaire. Elle a pour cadre le Groupement de Recherches "Didactique et Acquisition des Connaissances Scientifiques" (du Centre National de la Recherche Scientifique) qui coordonne le travail de plusieurs équipes françaises de recherche sur la Didactique des Mathématiques et de l'Informatique[1].

Cette étude est le prolongement naturel d'une expérience d'enseignement de la programmation structurée en langage LOGO. La méthodologie est analogue, nous en rappelons succintement les idées principales[2].

Il n'existe pas actuellement d'enseignement d'informatique obligatoire au niveau secondaire. S'il n'y a pas d'enseignement, il n'y a pas de **difficultés** spécifiques connues, repérées, pas de **conceptions spontanées** identifiées. Puisque notre

3 (1990) (p. 143-162) IREM de Strasbourg

1 *On peut citer notamment C. Laborde , N. Balacheff, B. Mejias (1985), P. Mendelsohn (1985), J.. Rogalski (1988), R. Samurçay (1985).*

2 *Pour plus de détails , il est possible de consulter (C. Dupuis , M.-A. Egret et D.Guin 1987 , 1988 1 et 2).*

objectif est la mise au point d'un enseignement prenant en compte ces éléments, il nécessite une expérience d'enseignement et une analyse de **l'activité de programmation** des élèves, de leurs difficultés .

L'activité de programmation contraint les élèves à expliciter, par un programme écrit, leurs procédures de résolution. Cette explicitation reflète les connaissances acquises et les représentations qu'ont les élèves du fonctionnement du dispositif informatique. L'analyse des procédures de résolution implique donc la définition de **critères** d'analyse fondés, non seulement sur la ressemblance du résultat obtenu et du résultat demandé, mais encore sur les connaissances et les représentations sous-jacentes. Les résultats tirés de l'analyse permettront ensuite de modifier l'enseignement pour tenter de remédier aux difficultés apparues.

I Préexpérimentation [1]

a) Essais

Nous avons proposé à 8 élèves volontaires des procédures récursives que nous leur avons fournies. Nous avons mis à leur disposition, sur une idée de P.Mendelsohn, le schéma suivant :

```
POUR ESSAI : N
SI : N = 0 ALORS  [STOP]
SPA
ESSAI : N-10
SPB
FIN
```

Remarques : N est supposé multiple de 10.
SPA, SPB sont des sous-procédures ou de simples instructions.

Les élèves ont manipulé cette procédure ESSAI pour différentes sous-procédures SPA et SPB avec des variations systématiques (il s'agit ici d'une récursivité **linéaire**, c'est-à-dire que la procédure ESSAI n'a qu'un seul appel récursif). Pour chaque procédure

[1] *Cette partie de la préexpérimentation a duré 6 heures d'enseignement. Pour plus de détails, il est possible de consulter (C.Dupuis , M.-A. Egret et D.Guin 1985) .*

ESSAI, ils devaient noter par écrit une **prévision** de l'exécution. Si la prévision était fausse, ils essayaient d'expliquer l'erreur.

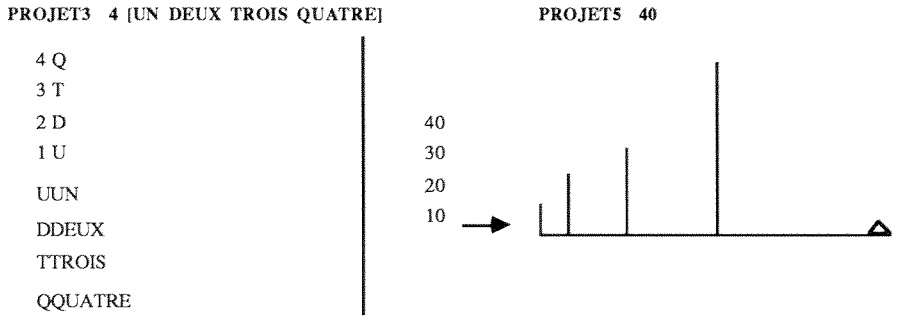
La manipulation de procédures récursives fournies à l'élève lui permet de se créer une représentation du **traitement d'une procédure récursive par le dispositif informatique**, à condition qu'on lui **propose des modèles** de fonctionnement de la récursivité lorsqu'il en ressent le besoin. Nous avons donc proposé aux élèves plusieurs **modèles** (il s'agissait de voir s'ils leur apportaient une aide sensible) : tout d'abord un modèle qui assimile l'appel récursif à une insertion de lignes, puis un tableau qui retrace l'ordre d'exécution des instructions de la procédure récursive.

La procédure récursive s'appelle elle-même, c'est ce qu'on désigne par le terme d'**auto-référence.** Dans un premier temps il faut convaincre l'élève que l'ordinateur accepte ce type de programme, c'est-à-dire qu'il peut l'exécuter. Une fois cette étape franchie, la **conception spontanée** de l'auto-référence est une forme de **retour au début** du programme. Cette conception spontanée (faire une action et recommencer) permet des prévisions d'exécutions correctes lorsque l'appel est **terminal** (situé à la fin de la procédure). Les procédures récursives fournies sont choisies de manière à remettre en question cette conception erronée en faisant apparaître des différences flagrantes entre les prévisions issues de cette conception et l'exécution de la procédure. De plus, les modèles de fonctionnement fournis mettent en évidence la **suspension** de l'exécution de la procédure appelante pour attendre la fin de la procédure appelée.

Remarquons que dans l'ensemble, à l'issue de cette phase, les prévisions sont correctes.

b) Projets

Nous avons ensuite demandé aux élèves d'écrire des procédures récursives en leur fournissant des exécutions de procédures dont voici deux exemples :



Les projets correspondaient aux réalisations de programmes récursifs à récursivité centrale, graphiques ou non graphiques, incluant parfois une procédure avant le STOP. Tous les élèves ont alors écrit au moins un programme correct, certains très rapidement.

Cette deuxième phase a permis de mettre en évidence l'**écart** existant entre la compréhension du **fonctionnement** d'une procédure récursive et son écriture. Celui-ci est lié au fait que l'écriture récursive nécessite la mise en place d'un **schéma fonctionnel statique** du type : pour construire l'objet de niveau n, on suppose construit l'objet de niveau n - 1. Les séances de travail que nous venons de décrire étaient centrées sur la compréhension du traitement d'une procédure récursive par le dispositif informatique, et non sur l'élaboration d'un tel schéma.

II Expérimentation *(15 h)*

Nous avons voulu élaborer des séquences d'enseignement sur la récursivité séparant les **différents types de difficultés** mises en évidence par la préexpérimentation. Nous avons donc dissocié :

- la recherche de l'**invariance** d'emboîtement de procédures (cf ci-dessous),
- l'exécution de procédures récursives fournies
- et l'écriture de procédures récursives.

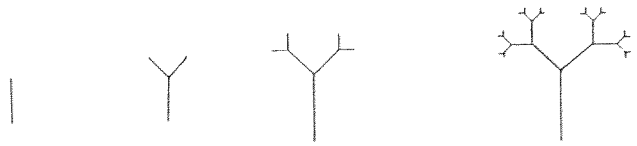
Nous avons travaillé avec une classe de 19 élèves (13-15 ans) de 4ème puis de 3ème. Les élèves suivaient une heure hebdomadaire d'informatique en plus de leur horaire de mathématique normal. Cette heure supplémentaire, mais obligatoire, était assurée par leur professeur de mathématique. L'activité décrite ici se place au début de la deuxième phase, après 23 heures de pratique active de la programmation en Logo. A l'issue de la première phase, un test individuel nous a convaincues que les élèves avaient acquis les éléments de base de la programmation structurée graphique (C. Dupuis, M. - A. Egret, D. Guin, 1987).

1) Les courbes (recherche de l'invariance d'emboîtement) :

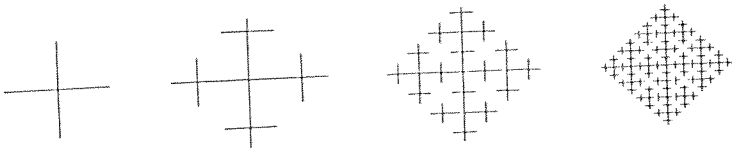
a) présentation de la situation :

Voici trois familles de courbes. Sur une même ligne sont dessinées quatre courbes de la même famille. Choisissez une famille, puis écrivez les programmes correspondant à chaque courbe, **en utilisant à chaque fois le programme précédent**.

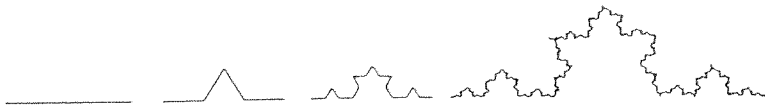
Les arbres



Les croix

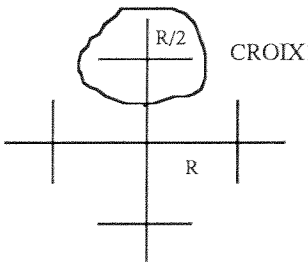


La courbe de Von Koch



Les élèves disposent des outils qui leur permettent d'écrire les programmes correspondant aux différentes courbes. Bien entendu, l'écriture ne sera pas récursive, étant donné qu'ils n'ont eu, à ce moment, aucun enseignement sur la récursivité. Il faut reconnaître qu'il s'agit pour eux d'une activité assez complexe comportant des difficultés de **structuration** et de **coordination** (cf. C. Dupuis, M-A. Egret et D.Guin 1987, 1988 1 et 2).

Tous les ingrédients nécessaires à la récursivité sont présents dans ces situations, sauf **l'auto-référence**. Chaque procédure peut faire appel au programme de la courbe précédente, toujours **de la même façon** : c'est ce que nous appelerons **invariance d'emboîtement**. Ce phénomène d'invariance est commun aux trois familles choisies. L'emboîtement de procédures différentes est le prolongement "naturel" des connaissances acquises en programmation structurée. Ce travail est donc consacré à la recherche de l'**invariance d'emboîtement**. Cette situation évite la difficulté liée à la **suspension de l'exécution** tout en préparant à la **représentation statique** du programme, nécessaire à l'écriture de procédures récursives : c'est un premier pas vers l'élaboration d'un schéma fonctionnel statique (J. Rogalski, G. Vergnaud, 1987). Prenons l'exemple des croix : le programme d'une croix quelconque s'écrit, en appelant **toujours de la même façon** le programme de la croix précédente :




```
POUR CROIX2 :R
REPETE 4 [AV :R CROIX1 :R/2 RE :R TD 90]
FIN
```

Pour les arbres et la courbe de Von Koch, les quatre étapes explicitées ne correspondent pas aux quatre premiers niveaux au sens récursif du terme. Les courbes représentées correspondent aux niveaux 1, 2, 3 et 5. Le niveau 4 a été volontairement omis ; la **courbe fantôme** qui lui correspond aurait sa place entre la troisième et la quatrième étape explicitée. L'écriture d'une procédure pour la courbe fantôme est indispensable pour que soit réalisée **l'invariance d'emboîtement** d'une courbe à la suivante dans la même famille.

La famille des arbres présente une particularité qui s'est révélée être une différence significative : **la présence de deux, trois puis cinq longueurs différentes** dans le même arbre. Aucune relation entre les longueurs n'était explicitée dans la consigne et nous n'avons donné aucune indication orale. Il faut reconnaître que nous n'avions pas prévu l'ampleur des **difficultés** rencontrées dans la **gestion des variables et des relations entre les variables.**

b) analyse de la situation :

Huit groupes d'élèves (sur neuf) ont choisi de programmer la famille des arbres. C'est donc les productions concernant la famille des arbres que nous avons analysées. Tous ces groupes ont écrit au moins une procédure par étape et, suivant la consigne, ils ont **tous emboîté** la procédure écrite à l'étape précédente dans la procédure de l'étape suivante. Comme nous l'avons vu précédemment, notre analyse repose sur la définition de critères. Chaque critère reflète un des aspects de l'activité. Nous ne définirons ici que le critère CODAGE révélant une des difficultés propres à l'emboîtement [1].

[1] *Pour plus de détails , notamment sur les modalités de gestion des relations entre longueurs , il est possible de se reporter à (C. Dupuis, D. Guin, 1989-2) .*

critère CODAGE :

Trois modalités de codage des longueurs sont apparues : le codage descriptif, le codage analytique et l'absence de codage.

- *descriptif* :

Il y a dans chaque procédure **autant de variables** d'entrée que de longueurs différentes dans l'arbre correspondant, ce qui permet d'éviter l'explicitation d'une relation entre ces longueurs. Mais, en plus, les noms des variables changent d'une procédure à l'autre ce qui permet d'éviter tout **conflit de désignation**. Le nombre de variables utilisées pourrait faire croire que ces élèves possèdent à fond la notion de variable en Logo. En fait, c'est une illusion car ils ne font ici que **désigner des objets différents par des noms différents.**

Le choix de la procédure de résolution par codage descriptif est un bon choix pour nos élèves dans cette situation, car il permet de **traiter successivement** deux problèmes :

- d'abord l'**écriture** et la **coordination** de procédures, où tous les objets qui ne sont pas reconnus comme égaux sont désignés par des variables différentes,

- puis, au moment de l'exécution, l'**affectation** de valeurs aux variables désignant les longueurs.

- *analytique* :

Une même variable est utilisée dans toutes les étapes ; des **relations entre les longueurs sont alors explicitées en termes de variables** (C et C/2, C et 2*C ou C et C-10). La procédure de résolution par codage analytique est plus complexe puisqu'il faut alors **traiter simultanément** les deux problèmes suivants :

- l'écriture et la coordination des procédures,

- l'expression, en termes de variables, d'une **relation fonctionnelle** entre les longueurs et la gestion de la composition (due à l'emboîtement) de cette relation avec elle-même.

Cette procédure de résolution présuppose l'**explicitation** d'une relation entre les longueurs.

- *sans* :

aucune variable d'entrée n'est utilisée. Les longueurs données a priori sont modifiées par approximations successives en fonction de l'aspect à l'écran.

La réussite complète est obtenue par les deux groupes ayant travaillé en **codage descriptif** et un groupe qui, après avoir écrit les procédures des trois premiers arbres en codage analytique, a écrit la dernière procédure en codage descriptif. Il semble bien que le codage descriptif soit la procédure de résolution la plus efficace, dans cette situation, pour nos élèves. La procédure de résolution par **codage analytique** est objectivement **plus difficile** que la procédure de résolution par **codage descriptif.** Les difficultés supplémentaires qu'implique la procédure de résolution par codage analytique peuvent expliquer le changement de procédure d'un groupe au dernier niveau.

Le **choix** d'un type de codage, descriptif ou analytique, dans la résolution de problèmes de programmation peut dépendre de la situation. Mais la possibilité de choisir dépend surtout de la **représentation** que l'élève s'est construite du fonctionnement du dispositif informatique. Ce choix est donc un **indice observable** de cette représentation.

c) lien avec les mathématiques

M. Kourkoulos (à paraître), dans une étude menée avec les mêmes élèves sur la mise en équation de problèmes, a observé une similitude des procédures de résolution en mathématique :

En ce qui concerne les problèmes qui peuvent être résolus par une équation du premier degré à une inconnue, la **réussite des élèves augmente** à partir du moment où ils

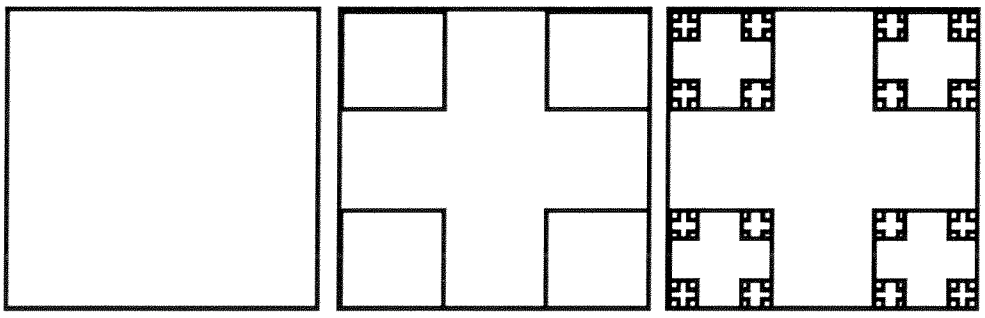
"disposent" de **plusieurs variables**, c'est-à-dire après un enseignement des systèmes d'équations. L'augmentation de la réussite est d'autant plus évidente que le problème est difficile. M. Kourkoulos explique cette augmentation du taux de réussite par la **possibilité** d'utiliser plusieurs variables.

Il y a à la fois **convergence** entre les raisons qui amènent à utiliser plusieurs variables en mathématique et en Logo et **coïncidence** entre les élèves qui le font en mathématique et en Logo. L'explicitation d'une relation entre variables est une tâche qui s'apparente à la mise en équations d'un problème de mathématique.

d) une nouvelle situation : les carrés

Nous avons voulu savoir si le codage **descriptif** était effectivement utilisé pour **contourner des difficultés** lors de la résolution d'un problème de programmation, et ce même lorsqu'une relation **fonctionnelle** est **connue**.

Nous avons donc décidé de proposer une nouvelle famille de courbes :



(étape 1) (étape 2) (étape 3)

en explicitant oralement une relation **fonctionnelle** (chaque carré a pour côté le tiers du côté du carré qui l'entoure). La détermination de la relation n'était donc plus à la charge de l'élève. Par contre, l'expression de cette relation en termes de variables devait être trouvée.

Nous avons constaté que tous les groupes ont utilisé une seule variable pour écrire les programmes de cette famille de courbes. L'analyse des différentes situations confirme bien que, lorsqu'une relation fonctionnelle entre les variables est **connue** des élèves, ils cherchent à l'exprimer par un **codage analytique** et à l'utiliser tant que cette procédure de résolution est **compatible** avec leur représentation du fonctionnement du dispositif.

2) Essais

Cette partie s'est déroulée de manière analogue à celle de la préexpérimentation (cf I.a).

3) Ecriture de procédures récursives

a) Ecriture récursive des courbes.

L'objectif de cette séance est alors d'écrire récursivement les programmes correspondant aux différentes familles de courbes. A titre d'exemple, le programme récursif de la croix de niveau N dont la branche a pour mesure L peut s'écrire :

```
POUR CROIX : L : N
SI : N = 0 ALORS [STOP]
REPETE 4 [AV : L  CROIX : L / 2 : N -1 RE : L  TD 90]
FIN
```

Le passage d'une écriture respectant l'invariance d'emboîtement à une écriture récursive se fait sans difficultés, bien qu'il s'agisse ici d'une récursivité non linéaire. L'invariance d'emboîtement mise en évidence auparavant a favorisé une représentation statique de la procédure, nécessaire à l'écriture récursive. Il est clair que cet objectif n'aurait pu être atteint sans le travail préalable sur la recherche de l'invariance d'emboîtement. Signalons que cette activité, quoique très dirigée, a beaucoup plu aux élèves.

b) Projets

Cette partie s'est déroulée de manière analogue à celle de la préexpérimentation (cf I.b).

III Test.

1) Construction du test.

Le but de ce test est de repérer les conceptions que les élèves se sont construites du **fonctionnement d'une procédure récursive**, plus particulièrement lorsque la récursivité est **centrale**. Pour cela, nous avons choisi de faire passer un test individuel papier-crayon (donc sans accès à l'ordinateur) en demandant, pour chaque exercice, une prévision d'exécution de la procédure fournie. Contrairement à ce qui peut se produire lorsque l'on demande l'écriture de procédures récursives, il n'est pas possible de contourner les difficultés liées au fonctionnement du dispositif informatique.

Nous avons choisi de ne pas nous limiter à des situations où la réussite est possible avec une règle d'action globale (lorsqu'il s'agit de procédures récursives avec appel central) de la forme : "avant l'appel, ça décroit et après l'appel, ça croit". Dans de telles situations, les conceptions des élèves sont difficilement identifiables. Cette règle d'action globale était cependant suffisante pour écrire bon nombre des procédures récursives que nous avions demandées dans les "projets" (cf. I b).

Ce test s'est déroulé en deux séances, espacées d'une semaine, sans corrigé entre les deux séances. Les élèves étaient invités à consulter leur cahier d'informatique contenant les fiches de cours et les programmes qu'ils avaient réalisés. Ils ont été un peu déroutés lors de la première séance par la forme des questions posées. Nous avons constaté que la première séance a eu un effet d'entraînement à ce type d'activité et que certains types de prévisions fausses ont disparu complètement lors de la deuxième séance.

Les énoncés sont construits suivant deux variations du schéma ESSAI (cf. I.a) :

Type de récursivité :

- récursivité terminale, c'est-à-dire absence de SPB et absence d'opérations sur l'appel
- récursivité avec appel au début, c'est-à-dire absence de SPA
- récursivité centrale, c'est-à-dire présence de SPA et SPB.

Type de sous-procédures SPA et SPB :

- écriture de valeurs de variables
- graphique (dessin dont les dimensions varient en fonction du niveau)
- liste (écriture de lettres d'un mot)

Pour permettre une meilleure observation, les énoncés choisis ne combinent pas les difficultés. SPA et SPB sont toujours du même type dans ces tests, sans être identiques. Voici trois exercices extraits du test :

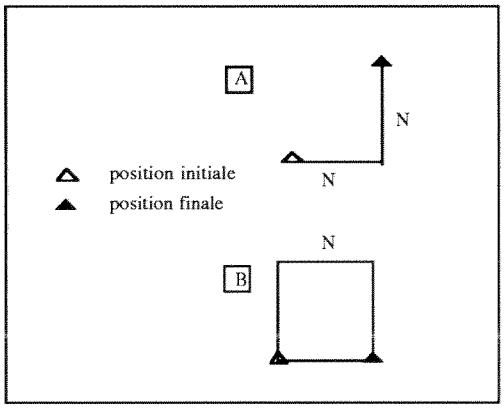
exercice a :

```
POUR TOTI   :MOT
SI VIDE? :MOT ALORS [STOP]
ECRIS DER   :MOT
TOTI SD     :MOT
ECRIS PREM  :MOT
FIN
```

La prévision d'exécution était demandée pour TOTI "SAC

exercice b :

La procédure ESSAI étant rappelée, la prévision d'exécution était demandée pour ESSAI 30, avec les sous-procédures SPA (dessin de A) et SPB (dessin de B) suivantes :



exercice c :

```
POUR    TOTO :N
SI      : N = 1   ALORS  [STOP]
ECRIS   :N - 1
TOTO    :N - 2
ECRIS   :N
FIN
```

La prévision d'exécution était demandée pour TOTO 9.

2) Analyse des productions des élèves .

a) Un critère d'analyse : prévision d'exécution en récursivité centrale[1].

S - exécution séquentielle

Les procédures sont exécutées dans l'ordre où elles sont écrites, une seule fois. C'est le type de prévision le plus "loin" de la récursivité. On n'y voit aucune trace de l'enseignement de la récursivité.

R - l'appel récursif central ne déclenche pas la suspension de l'exécution de la procédure appelante.

La conception spontanée de l'auto-référence est le retour au début du programme. On voit cette conception en œuvre dans les différentes modalités "R". Cette conception est d'autant plus difficile à mettre en échec qu'elle permet, par ailleurs, des prévisions d'exécution correctes pour les programmes en récursivité terminale.

R 0 et R 0 AT Les procédures situées après l'appel ne sont jamais exécutées,
- **R 0 AT :** et la prévision d'exécution de SPA est correcte
- **R 0 :** et la prévision d'exécution de SPA n'est pas correcte

R 1 et 2 Les procédures situées après l'appel sont prises en compte lorsque le test d'arrêt est vrai et sont alors exécutées :
- **R 1 :** soit pour la seule valeur initiale de la variable
- **R 2 :** soit pour la dernière valeur de la variable avant le STOP.

RFin 1, 2 et 3 Toutes les procédures sont exécutées dans l'ordre où elles sont écrites et pour toutes les valeurs de la variable. Mais la variable change de valeur :
- **RFin 1** soit après chaque exécution de SPB
- **RFin 2** soit avant chaque exécution de SPB
- **RFin 3** soit avant et après chaque exécution de SPB.

---

[1] *Nous ne présenterons ici qu'un seul des critères d'analyse. Pour plus de détails, on peut consulter (C. Dupuis, D. Guin, 1989-1).*

Les modalités classées sous "S" et "R" traduisent la **méconnaissance de l'aspect suspensif** dans l'exécution d'une procédure récursive.

D A T - La procédure fonctionne comme une succession de Deux programmes récursifs avec Appel Terminal.

On voit ici aussi en œuvre la conception spontanée d'un retour à un début des procédures, plus sans doute aussi le "souvenir" qu'en récursivité centrale il y a autant d'exécutions de la procédure SPA que de la procédure SPB. Un pas est fait dans la bonne direction. Une partie de **l'aspect suspensif** de la récursivité a été perçue.

C - prévision correcte.

Même dans le cas d'une prévision correcte isolée, il n'est pas certain que le modèle de fonctionnement que l'élève s'est construit soit correct. Un modèle global "miroir", tel qu'il est décrit ci-après, permet des prévisions correctes dans de nombreuses situations.

Prévisions pour ESSAI 30

S	R0	R1	R2	Rfin1	RFin2	RFin3	DAT	Correct
SPA 30	SPA 30	SPA 30	SPA 30	SPA 30	SPA 30	SPA 30	SPA 30	SPA 30
SPB 30	SPA 20	SPA 20	SPA 20	SPB 30	SPB 20	SPB 20	SPA 20	SPA 20
	SPA 10	SPA 10	SPA 10	SPA 20	SPA 20	SPA 10	SPA 10	SPA 10
		SPB 30	SPB 10	SPB 20	SPB 10		SPB 30	SPB 10
				SPA 10	SPA 10		SPB 20	SPB 20
				SPB 10			SPB 10	SPB 30

b) Modèle global "miroir"

Nous avons mis en évidence pour des procédures spécifiques SPA et SPB un modèle erroné qui conduit parfois à des prévisions correctes : nous l'avons appelé modèle global **"miroir"**. Prenons l'exemple de la procédure TOTI : MOT (exercice a, cf.III.1) La prévision d'exécution était demandée pour le mot SAC. La prévision correcte était :

C A S S S S

Suivant un modèle global **" miroir "**, l'élève fournit la réponse :

C A S S A C

qui provient d'une prévision correcte de l'exécution de la procédure **avant** l'appel (**C A S**). La prévision est complétée par symétrie, comme dans un miroir, bien que les instructions avant et après l'appel ne soient pas identiques. On peut aussi obtenir d'autres réponses "miroir" résultant de prévisions fausses de la procédure avant l'appel.

3) Résultats et commentaires

Les phénomènes les plus marquants sont la **grande variété des erreurs et l'instabilité des comportements de réponses.** Ainsi, par exemple, les prévisions de la majorité des élèves sont correctes pour l'exercice b (cf. III.1) : 16 réussites sur 18 élèves présents. En revanche, ils ne sont que 6 à faire une prévision correcte pour l'exercice a et 2 pour l'exercice c.

La différence de réussite entre les trois exercices a, b et c s'explique par la conjonction de caractéristiques qui rendent l'exercice b plus facile que les deux autres (analogie avec des exemples programmés, le test d'arrêt est à 0 et un éventuel dessin de taille 0 est invisible). De plus, un modèle global "miroir" donne des prévisions correctes. En effet, l'exercice b est le seul des trois à être "symétrique" et la gestion des valeurs de la variable y est beaucoup plus simple (on constate 3 prévisions correspondant à un modèle global "miroir" pour a et 6 pour c ; elles sont toutes associées à une prévision correcte pour l'exercice b) .

On observe aussi **un phénomène de retour à des conceptions antérieures ou plus primitives** lorsque la difficulté s'accroît. Ce phénomène de retour à une conception antérieure "dans la difficulté" a déjà été observé précédemment avec les mêmes élèves. Ainsi, à la fin de la phase d'enseignement de la programmation structurée, nous avions écrit : <<on peut affirmer (...) que les éléments de base d'une programmation graphique structurée sont en place chez ces élèves. La structuration des programmes est devenue une méthode efficace de résolution de problème.>> (Dupuis, Egret, Guin, 1987). Cependant, nous avons constaté lors de l'écriture des programmes pour les arbres (cf II.1 a) que deux groupes d'élèves avaient écrit des procédures graphiques sans structurer leurs programmes (Dupuis, Guin, 1989-2).

Conclusion

En ce qui concerne la phase de recherche de l'invariance d'emboîtement, les observations faites sur les choix de codage, la gestion des variables et des relations fonctionnelles confirment l'existence d'un **lien** étroit entre, d'une part, le **codage** des variables et leur gestion et, d'autre part, la **représentation** du traitement d'une procédure par le dispositif informatique :

- représentation exclusivement en termes d'**exécution,** dont un indice observable est le codage descriptif,

- représentation **statique** en termes d'états et de relations entre ces états, dont un indice observable est le codage analytique.

Durant la phase d'écriture de programmes récursifs, l'écriture récursive des familles de courbes a été bien réussie. De plus, tous les élèves ont été capables d'écrire au moins une procédure récursive correcte pour un "projet", en interaction avec l'ordinateur.

Cependant, l'étude de leurs prévisions d'exécution montre une grande diversité d'erreurs (si l'on considère l'ensemble des exercices que nous leur avions proposés), des procédures incorrectes et des erreurs instables pour la plupart d'entre eux, l'accroissement des difficultés induisant souvent un **retour** à des **conceptions antérieures**.

Nous avons constaté que les élèves se dégagent **progressivement** d'une représentation d'une procédure exclusivement en termes d'exécution, et accèdent à une **représentation statique** nécessaire à l'écriture récursive au fur et à mesure qu'ils intègrent les caractéristiques du traitement par le dispositif informatique. Toutefois, une représentation même erronée, tel le modèle global "miroir", est suffisante pour écrire correctement bon nombre de procédures récursives (possédant une certaine symétrie).

Ainsi, si l'objectif de l'enseignement est une bonne compréhension du traitement d'une procédure récursive par le dispositif informatique, l'enseignement devra être modifié de manière à casser cette conception fausse. Par contre, si l'objectif est plutôt l'écriture récursive, on peut émettre l'hypothèse qu'une conception partielle, même erronée, est suffisante dans un premier temps.

Références :

DUPUIS C. , EGRET M. - A. , GUIN D. (1985) : *Récursivité et Logo - 1 : Préexpérimentation,* I.R.E.M de Strasbourg .

DUPUIS C., EGRET M.- A., GUIN D. (1987) : *Logo 3 . Programmation structurée: Présentation et Analyse de situations* , I.R.E.M. de Strasbourg, F-67084 Strasbourg.

DUPUIS C., EGRET M. - A. , GUIN D. (1988-1) : Pour une analyse multi-critères de l'activité de programmation en Logo, *Annales de Didactique et de Sciences Cognitives* , vol .1, pp 111-130 , IREM de Strasbourg .

DUPUIS C, EGRET M - A , GUIN D. (1988-2) : Présentation et analyse d'activités de programmation en LOGO , *Petit X* , n° 18, pp. 47-69, I.R.E.M de Grenoble .

DUPUIS C. , EGRET M.- A. , GUIN D. : (à paraître), *Récursivité et Logo 2 : Présentation et Analyse de situations* , I.R.E.M. de Strasbourg, F-67084 Strasbourg.

DUPUIS C., GUIN D. (1989-1) : Représentations du fonctionnement d'une procédure récursive en Logo, *Psychology of Mathematics Education,* Actes de la 13e conférence internationale, vol. 1, pp. 220-227, Paris, Juillet 1989.

DUPUIS C., GUIN D. (1989-2) : Gestion des relations entre variables dans un environnement de programmation LOGO, *Educational Studies in Mathematics*, vol 20 n° 3, Information Technology and Mathematics Education, pp. 293-316.

KOURKOULOS M. (à paraître) :*Thèse de doctorat* , I.R.E.M. de Strasbourg, Université Louis Pasteur, F-67084 Strasbourg.

LABORDE C., BALACHEFF N., MEJIAS B.(1985) : "Genèse du concept d'itération une approche expérimentale" , *Enfance* , Vol. 2 / 3 , pp. 223 - 239 .

MENDELSOHN P.(1985) : "L'analyse psychologique des activités de programmation chez l'enfant de CM 1 et CM 2" *Enfance* , Vol. 2 / 3 , pp. 213 - 221.

ROGALSKI J. (1988) : "Acquisition de structures conditionnelles : effet des prérequis logiques et des représentations du dispositif informatique", *Annales de Didactique et de Sciences Cognitives* , N°1, pp 131 - 152.

ROGALSKI J., VERGNAUD G.(1987) : Didactique de l'Informatique et acquisitions cognitives en programmation , *Psychologie française* , vol 32-4, pp. 267-273.

SAMURCAY R.(1985) : "Signification et fonctionnement du concept de variable informatique chez des élèves débutants", *Educational Studies in Mathematics*, N° 16, pp.143-161.