LA "BOITE A OUTILS LOGO" POUR TO7, TO7-70, MO5
Les commandes COPIEDEF et CLONEDEF

Daniel ARGOT

Remarque : Les primitives DEFINIS et TEXTE, permettant d'ouvrir à deux battants les portes de LOGO à l'Intelligence Artificielle, n'existent pas dans le lot initial des primitives LOGO-Thomson. L'objectif de cet article est en lui-même une petite méditation sur DEFINIS et TEXTE - procédures que nous donnerons par la suite et que d'ailleurs de nombreux ouvrages pour TO7 ont mentionnées - méditation qui débouchera sur deux nouvelles "briques". les commandes COPIEDEF et CLONEDEF ayant pour fonctions respectives de dupliquer des définitions de procédures et d'en fabriquer de véritables clones.

Par égard pour le "tout-débutant" en LOGO (que l'amateur déjà chevronné et impatient m'excuse) supposons que nous n'ayons jamais entendu parler de DEFINIS et TEXTE à ce jour.

```
POUR TENTATIVE
EC [ BONJOUR]


POUR TEST
EC [REUSSI ou NON?]
FIN
EC [AU REVOIR !]
FIN
```

Afin de créer et modifier ses procédures le détenteur de TO7, MO5... n'a donc à sa disposition, pour l'instant, que les traditionnelles primitives POUR et ED !

Mais que se passe-t-il s'il est atteint par la très naturelle et irrésistible idée de *créer une procédure à l'intérieur d'une autre procédure ?*

D'où l'exemple ci-dessus de la procédure de nom TENTATIVE, un peu simpliste il est vrai, qui n'a d'autre objet que d'illustrer notre questionnement. Sortons de l'EDITEUR en validant par CNT Q :

```
VOUS VENEZ DE DÉFINIR TENTATIVE
AU REVOIR !
```

Notons sur l'écran l'interruption inattendue, à ce stade, du "AU REVOIR" Appelons alors, en mode direct, la procédure par son nom :

TENTATIVE	
BONJOUR	Le symbole ">" nous précise que nous venons de passer en "mode définition de procédure": le système attend.
> FIN	il nous faut donc taper FIN pour revenir en mode direct:
VOUS VENEZ DE DEFINIR TEST	il y a télescopage inopiné du "REUSSI OU NON?" sans mémorisation.
"REUSSI OU NON?	Cette non-mémorisation est montrée par l'impression, demandée par IM, de la procédure TEST.
IM "TEST	TEST est devenue une procédure qui ne
POUR TEST	fait rien. Rappelons maintenant
FIN	TENTATIVE pour en avoir le cœur net ;

```
? TENTATIVE
BONJOUR
TEST EXISTE DEJA DANS TENTATIVE
```

L'instruction EC [RÉUSSI OU NON ?] est bien irrémédiablement perdue ! La commande de définition POUR est en fin de compte une commande statique : elle n'autorise pas la création de procédure dans une autre procédure.

Pour remédier à cet inconvénient, il est aisé de créer sur TO7 ou MO5 une procédure nommée DEFINIS qui jouera le rôle de commande dynamique en autorisant la définition de nouvelles procédures au moment de l'exécution.

```
POUR DEFINIS :TITR :LIST
SORTIE 4 EC PH "POUR PH :TITR PREM :LIST AFFICHER SP :LIST
EC "FIN SORTIE 1 ENTRÉE 4
FIN
POUR AFFICHER :L
SI VIDE? :L [STOP]
EC PREM :L ARFFICHER SP :L
FIN
```

Dans le livre *Logo Manuel de référence* (Cédic-Nathan) p. 54 ou dans l'article d'André Myx paru dans *Éducation et Informatique* n° 23, intitulé "les commandes TEXTE et DÉFINIS", vous trouverez des présentations assez voisines des deux procédures ci-dessus. Je vous les

propose donc sans autre commentaire puisqu'elles serviront de procédures "outils". Voici un premier exemple très simple :

```
   POUR EXEMPLE1
   EC [ Que pensez-vous de ce petit test ?]
   DEFINIS "ESSAI I [[ ][EC [un essai réussi ? ]]]
   EC [n'est-ce pas]
   FIN
VOUS VENEZ DE DEFINIR EXEMPLE1
? EXEMPLE1
Que pensez-vous de ce petit test ?
n'est-ce pas
VOUS VENEZ DE DÉFINIR
ESSAI1
IM "ESSAI1
POUR ESSAI1
EC [un essai réussi?]
FIN
```

DEFINIS demande 2 données ; La première donnée est un mot qui est le nom de la procédure à définir. La deuxième est une liste de liste (la 1° sous-liste étant la liste des arguments ; ici vide). On voit ici l'effet de la procédure récursive AFFICHER.

Encore plus fort ; on peut indenter les DEFINIS !

```
POUR EXEMPLE2
EC [ de plus en plus amusant: ]
DEFINIS "ESSAI2 [ [] [ EC [Ce n'est pas fini i] DEFINIS "ESSAI3 [[ ]
[EC [Bravo]]]]
EC [Vraiment]
FIN

? EXEMPLE2
de plus en plus amusant:
VOUS VENEZ DE DEFINIR ESSAI2
IM "ESSAI2
POUR ESSAI2
EC [Ce n'est pas fini !] DEFINIS "ESSAI3 [ ] [EC [Bravo]]]
FIN
? ESSAI2
Ce n'est pas fini 1
VOUS VENEZ DE DÉFINIR
ESSAI3
POUR ESSAI3
EC [Bravo]
FIN
```

DEFINIS, comme son nom l'indique, permet de DÉFINIR une procédure sous forme de LISTE de LISTES et pourra ainsi engendrer une forme d'auto-programmation.

Même si DEFINIS est une commande dynamique de définition de procédures, son utilisation ne suffit pas pour traiter et modifier des procédures déjà définies. Or, on le sait, LOGO ne fait pas de distinction entre "programmes" et "données". II suffit donc de s'octroyer un outil adéquat capable d'examiner une procédure déjà définie et d'en extraire le "texte" afin qu'il puisse être manipulé comme une liste.

Cet outil, c'est la procédure TEXTE que voici : (sa construction est plus délicate que celle de DEFINIS. En particulier la réalisation de la procédure auxiliaire TEXTAUX)

```
POUR TENTE :NOM
SORTIE 4 IM :NOM
ENTREE 4
RENDS (TEXTAUX LL [] ENTREE 1 SORTIE 1 )
FIN
POUR TEXTAUX :LIST :LAUX
SI EGAL? :LIST [FIN] [RENDS MP SP SP PREM :LAUX SP :LAUX] RENDS
TEXTAUX LL MD :LIST :LAUX
FIN
```

On en trouve une forme équivalente dans le livre *LOGO MANUEL de RÉFÉRENCE* déjà cité, à la page 55.

Par contre, l'article d'André Myx nous la présente comme une commande (et non comme une opération)

ICI, TEXTE demande une donnée ; un mot qui est le nom d'une procédure définie. En quelque sorte, TEXTE est une procédure dont le rôle est inverse de celui de DEFINIS :

TEXTE permet de transformer la procédure considérée en une liste de listes (la première sous-liste étant la liste des paramètres de la procédures). La plus simple est de montrer comment agit la procédure-opération TEXTE :

```
POUR CARRE :L
REPETE 4[AV :L TD 90]
FIN
```

L'exemple classique de la procédure CARRE

En vérité, outre la liste des paramètres (en fait, ici il n'y en a qu'un), TEXTE nous donne accès au "CORPS" de la procédure.

```
EC TEXTE "CARRE
[:L] [REPETE 4 [AU :L TD 90]]
```

Une petite remarque ; d'autres versions LOGO (Apple, Micral, etc.) nous fournissent la liste des paramètres sans les ":", ainsi [L] précisant par là que les "passages de paramètres en LOGO" s'effectuent "par valeurs".

Si TEXTE est présentée comme une procédure-fonction, donc qui "rend" un résultat, (et c'est le cas ici), il est immédiat que le "corps" de notre procédure devient manipulable à souhait par les primitives usuelles de traitement de liste (ou de mots) : PREM, SP, SD, PH, LISTE, MP, MD...

```
DEFINIS "CARREMOD MD [AU 50] TENTE "CARRE
VOUS VENEZ DE DEFINIR
CARREMOD
IM "CARREMOD
POUR CARREMOD :L
REPETE 4[AV :L TD 90]
AV 50
FIN
```

L'utilisation fine du tandem "DEFINIS-TEXTE" ouvre le champ à la définition, l'examen, la modification, l'évolution de procédures. N'est-ce pas prodigieux ?

Toutefois, si l'on veut jongler en toute liberté avec TEXTE et DEFINIS, il est préférable de se méfier de la première sous-liste des paramètres. Un essai convaincra les débutants

Supposer que dans la définition précédente, on veuille placer AU 50 comme première instruction. Donc à la place de MO, on écrit MP :

```
DEFINIS "CARREMOD MP [AU 50] TENTE "CARRE
POUR N'AIME PAS ( CARREMOD AU 50 1
```

Eh oui, il y a des interférences. IL faut donc prendre les précautions suivantes (que les lecteurs et lectrices décoderont aisément) :

```
DEFINIS "CARREMOD MP PREM TENTE "CARRE MP [AU 50] SP TENTE "CARRE
POUR CARREMOD :L
AV 50
REPETE 4 [AV :L TD 90]
FIN
CARREMOD 30 (suivi de COPIE pour la recopie de l'écran graphique)
```

Et CLONEDEF, et COPIEDEF dans tout cela ?

Nous venons de passer tout notre temps sur TEXTE et DEFINIS allez-vous dire, et le titre de cet article est : "les commandes COPIEDEF et CLONEDEF", alors. C'est que, voyez-vous, il ne s'agit nullement d'un détour de ma part, En réalité, COPIEDEF et CLONEDEF, si vous avez bien intégré TEXTE et DEFINIS vont apparaître naturellement ;

Occupons-nous d'abord de COPIEDEF,

Si l'on veut modifier et faire évoluer une procédure donnée, il peut s'avérer indispensable de **sauvegarder le nom** (présupposé connu) de cette procédure source afin de la comparer à ses clones dégénérés: Tel est le rôle de COPIEDEF.

COPIEDEF demande deux données. Les données doivent être des MOTS représentant des NOMS de procédures. COPIEDEF fait de la première donnée un synonyme de la seconde. La syntaxe de cette commande (il s'agit d'une commande de gestion de l'espace de travail) sera du type ;

```
COPIEDEF "NOMNOUVEAU "NOMANCIEN
```

Comment allons-nous PROCéder pour créer cette commande (qui existe sur d'autres matériels : Apple, Micral, Atari 520ST, IBM PC, etc.) ? Un jeu d'enfants : II suffit d'utiliser le prédicat PROC? (PROC? "nom rend VRAI si "nom est le titre d'une procédure, FAUX sinon) et de créer la procédure synonyme (que l'on nommera par exemple PROC2) de la procédure originale PROC1 en attribuant (grâce à DEFINIS) à PROC2 la liste rendue par TEXTE :PROC1

```
POUR COPIEDEF :PROC2 :PROC1
SI PROC? :PROC1 [DEFINIS :PROC2 TEXTE :PROC1]
FIN                              facile, non ?
COPIEDEF "CARREDIS "CARRE
VOUS VENEZ DE DEFINIR CARREDIS
IM "CARREDIS
POUR CARREDIS :L                 La procédure CARREBIS est bien une
REPETE 4 [AV :L TD 90]           copie conforme de CARRE.
FIN
```

Oui ... mais (biensûr, il y a un mais), c'est trop beau pour marcher dans tous les cas ! Vous avez deviné ; et si l'on veut "cloner" une procédure récursive ? Allez, pour simplifier, prenons même une "fausse récursivité" .

```
POUR ROND
AV 1 TD 1 ROND
FIN
COPIEDEF "CERCLE "ROND           essayons de "cloner"
VOUS VENEZ DE DEFINIR CERCLE
IM "CERCLE
POUR CERCLE                      raté ! je voulais "cercle", pas "rond"
AV1 TDIROND                      Une petite consolation ; ce serait aussi
```

```
FIN
```

raté avec les COPIEDEF des autres matériels:

Réfléchissons : comment modifier ce COPIEDEF en CLONEDEF pour "cloner" de façon parfaite une procédure, qu'elle soit courte ou longue, récursive ou non ? II va de soi que l'on va réutiliser DEFINIS et TEXTE. Cependant, cela ne suffit pas : dans le cas d'une procédure récursive (qui se rappelle elle-même) il faut remplacer d chaque fois l'ancien nom de la procédure.

```
POUR REMPLACE :OBJ1 :OBJ2 :LIST
SI VIDE? :LIST [RENDS[]]
SI LISTE? PREM :LIST [RENDS MP REMPLACE :OBJI OBJ2 PREM :LIST
REMPLACE :OBJ1 :OBJ2 SP :LIST]
SI EGAL? PREM :LIST :OBJ1 [RENDS MP :OBJ2 REMPLACE :OBJ1 :OBJ2 SP
:LIST 1
RENDS MP PREM :LIST REMPLACE :OBJ1 0,J2 SP :LIST
FIN
```

D'où cette procédure outil REMPLACE (bien connue). Exemple :

```
EC REMPLACE "LA "UNE [LA CIGALE ET LA FOURMI]
UNE CIGALE ET UNE FOURMI
```

Mous obtenons maintenant une véritable commande, que j'ai nommée CLONEDEF, fabriquant d'authentiques "clones" :

```
POUR CLONEDEF :PROC2 :PROC1
SI PROC? :PROC1 [DEFINIS :PROC2 REMPLACE :PROC1 :PROC2 TEXTE :PROC1]

FIN
CLONEDEF "CERCLE "ROND
VOUS VENEZ DE DEFINIR CERCLE
IM "CERCLE
POUR CERCLE
AV1 TD 1 CERCLE
FIN
```

Cette fois, on a cloné pour de vrai !

Un test ultime s'impose avec la procédure REMPLACE :

```
CLONEDEF "CHANGE "REMPLACE
VOUS VENEZ DE DEFINIR CHANGE
IM "CHANGE

POUR CHANGE :OBJ1 :OBJ2 :LIST
```

```
SI VIDE? :LIST [RENDS 1]]
SI LISTE? PREM :LIST [RENDS MP CHANGE :OBJ1 :OBJ2 PREM :LIST CHRNGE
:OBJ1 :OBJ2 SP :LIST]
SI EGAL? PREM :LIST :OBJ1 [RENDS MP :OBJ2 CHANGE :OBJ1 :OBJ2 SP
:LIST ] RENDS MP PREM :LIST CHANGE :OBJ1 :OBJ2 SP :LIST
FIN
```

Ça marche ! Une petite question maintenant : juste pour voir si vous avez assimilé les DEFINIS, TEXTE etc. Quel est l'effet de la procédure CLOWNDEF suivante (et ne dites pas que c'est clownesque): POUR CLOWNDEF :PROC2 :PROC1

```
SI PROC? :PROC1 [DEFINIS :PROC2 TEXTE ;PROC1 DEFINIS ;PROC2 REMPLACE
:PROCI :PROC2 TEXTE ;PROC1]
FIN
```

Daniel ARGOT