

# Géométrie et Programmation Scheme avec Dr. Geo

PAR ANDREA CENTOMO

Organization for Free Software in Education and Teaching

EMAIL : [acentomo@ofset.org](mailto:acentomo@ofset.org)

*13 juin 2004*

## Résumé

One of the most peculiar aspects of Dr. Geo is related to the presence of a powerful programming ambient strongly connected to its geometrical engine. In this paper, after a brief introduction to the concept of Scheme Figure, we present an easy example of iterative figure with some generalizations.

La forte intégration qu'on trouve dans le logiciel de géométrie interactive Dr. Geo entre les outils avancés de programmation - langage Guile/Scheme - et le moteur pour la manipulation d'objets géométriques, ouvre de nouvelles perspectives pour la didactique des mathématiques à différents niveaux, dans un horizon qui s'étend de l'école secondaire de premier degré jusqu'à l'école de deuxième degré.

Nous ne sommes pas encore à même d'offrir un développement systématique et exhaustif à propos des possibilités didactiques relatives à ce mariage entre programmation et géométrie et, nous procéderons, donc, en analysant seulement certains exemples d'emploi relatifs aux figures interactives.

L'idée d'exploiter des structures interactives mises à disposition par le langage de haut niveau Scheme représente l'exemple, le plus naturel, d'articulation entre géométrie et programmation. Cet exemple est certainement connu par tous ceux qui connaissent le langage Logo. D'autres possibles domaines d'application, moins évidents, pourraient être liés à l'étude de la théorie de la probabilité.

## 1 Figures Scheme de Dr. Geo

Les **Figures Scheme de Dr. Geo** – FSD – sont des figures écrites dans un langage relativement naturel. Il ne s'agit donc plus de construire une figure à l'aide de l'interface graphique de Dr. Geo mais plutôt de décrire une figure dans le langage Scheme. Nous avons apporté le plus grand soin afin que la syntaxe utilisée soit facile et légère. Aussi l'ensemble des mots clés utilisés pour décrire une figure simple sont adaptables dans différentes langues. Ainsi une figure pourra être décrite en Français, en Anglais, en Espagnol, etc. Un mélange de langues est même possible mais ce n'est pas souhaitable.


### 1.1 Exemples base

En lui-même Scheme est un langage de très haut niveau, lorsqu'une figure est définie dans ce langage, nous disposons également de toute sa puissance pour par exemple définir récursivement telle partie de la figure, ou bien pour placer aléatoirement certains objets de telle sorte qu'à chaque ouverture de la figure, celle-ci est légèrement différente. Bref, les FSD sont libérées du carcan de l'interface graphique tout en étant renforcées du langage Scheme. Une FSD est donc un fichier créé à l'aide d'un éditeur de texte, il est ensuite ouvert dans Dr. Geo à l'aide de la commande Fichier → Évaluer.

Commençons par étudier un exemple simple de FSD :

```
(new-figure "Figura")
```

---

 Ce document a été conçu avec GNU  $T_{E}X_{M}A_{C}S$  (voir <http://www.texmacs.org>).

```
(lets Point "A" free 2 -2)
```

Cette FSD définit une figure avec un point libre  $A$  de coordonnées initiales  $(2; -2)$ . Comme nous pouvons le voir la syntaxe de définition d'un objet géométrique est relativement agréable, d'autant plus qu'elle est exprimée dans une langue maternelle. Intéressons nous de plus près à la deuxième instruction, en effet celle-ci suit une syntaxe qui est commune à toutes les commandes de définition d'objet. Ce type de commande se décompose comme suit :

- a) Elle commence toujours par le mot-clé `lets`, il indique que nous souhaitons définir un nouvel objet.
- b) Il est immédiatement suivi de la catégorie de l'objet, ici `Point`.
- c) Le nom de l'objet vient ensuite, `A`, il doit toujours être entouré de `"`. Si nous ne souhaitons pas nommer l'objet, il faut tout de même donner un nom vide comme suit: `"`.
- d) Enfin, nous précisons le type de l'objet – le type de point dans notre exemple – ici `free`. Cela signifie que le point  $A$  est libre.
- e) Le type de l'objet est suivi d'une liste d'argument spécifique. Dans notre exemple cette liste est composée de deux nombres, les coordonnées du point libre  $A$ .

Poursuivons avec un autre exemple :

```
(define (triangle p1 p2 p3)
  (Segment "" extremities p1 p2)
  (Segment "" extremities p2 p3)
  (Segment "" extremities p1 p3))

(define (rand) (- 8 (* 16 (random:uniform))))

(new-figure "Exemple")
(lets Point "A" free (rand) 0)
(lets Point "B" free 5 0)
(lets Point "C" free (rand) 5)
(triangle A B C)
```

Cet exemple est particulièrement intéressant, il nous montre trois choses importantes :

1. L'introduction de construction de plus haut niveau, non prévue au départ par Dr. Geo. Ici nous avons défini la fonction `triangle` qui, à partir de trois points, construit le triangle passant par ces trois points. Nous pouvons comparer ceci avec les macro-constructions mais avec un degré de liberté beaucoup plus important.
2. La définition de fonctions associées, ici nous avons défini la fonction `rand` qui retourne un nombre décimal compris entre -8 et 8. Nous utilisons cette fonction pour placer au hasard certains points de notre figure, ainsi à chaque ouverture la figure est légèrement différente.
3. En fait l'utilisation du mot clé `lets` n'est pas obligatoire, nous l'utilisons lorsque nous souhaitons garder une référence de l'objet créé. Par exemple dans la fonction `triangle`, nous ne gardons pas de références des segments créés, en revanche lorsque nous définissons nos points  $A$ ,  $B$  et  $C$  nous avons besoin de garder une référence, ces références ont le même nom<sup>1</sup> sans guillemet : `A`, `B` et `C`. Dans la suite nous appellerons **symbole** ces références, c'est la terminologie exacte du langage Scheme. Ainsi, lors de l'appel de la fonction `triangle`, nous passons en paramètre les symboles `A`, `B` et `C` qui sont utilisés pour définir nos trois segments.

---

1. D'un point de vue interne au langage Scheme, ces références sont des symboles pointant sur une structure interne de l'objet – un prototype – alors que les noms sont des chaînes de caractères.

Noter que lors de la définition des segments, nous ne donnons pas de nom, dans ce cas Dr. Geo va attribuer un nom par défaut défini à partir du nom des extrémités. Nos trois segments auront donc comme nom [AB], [BC] et [AC].

Pour clure cette section, voici un dernier exemple :

```
(new-figure "Send")

(lets Point "A" free 1 0)
(lets Point "B" free 2 0)
(lets Point "C" free 2 0)

(lets Line "d1" 2points A B)

(send A color yellow)
(send A shape round)
(send A size large)

(send B color green)

(send C masked)

(send d1 thickness dashed)
```

Les trois premières commandes créent deux points et une droite. La partie qui nous intéresse plus particulièrement est la série de commande `send`. Cette commande permet de communiquer avec un objet dont nous avons gardé un symbole, ici nous avons les symboles `A`, `B`, `C` et `d1`. Elle consiste à envoyer un message à un objet, son premier argument est l'objet avec lequel nous communiquons, le deuxième argument le message, le troisième et les suivants sont déterminés par la nature du message. Par exemple `(send A color yellow)` envoie le message `color` avec comme paramètre `yellow`, le point `A` est peint en jaune. Il est assez facile de comprendre le sens des autres commandes `send`.

## 1.2 Figures Scheme Iteratives

Un premier exemple est représenté par la figure :

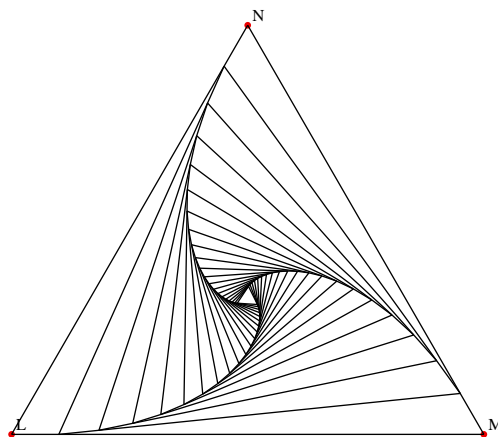


Figure 1. Base

Le code Scheme qui permet de la fixer est le suivant :

```

(new-figure "Base")

(define (triangle p1 p2 p3 n)

  (let* ((s1 (Segment "" extremities p1 p2))
         (s2 (Segment "" extremities p2 p3))
         (s3 (Segment "" extremities p3 p1))
         (A (Point "" on-curve s1 1/10))
         (B (Point "" on-curve s2 1/10))
         (C (Point "" on-curve s3 1/10)))

    (send A masked)
    (send B masked)
    (send C masked)

    (if (> n 0)

        (triangle A B C (- n 1))))

(lets Point "L" free -5 0)
(lets Point "M" free 5 0)
(lets Point "N" free 0 (* 5 (sqrt 3)))

(triangle L M N 20)

```

Ici nous avons utilisés la fonction `triangle` qui, à partir de trois points, construit le triangle passant par ces trois points. Un petit cycle est suffisant pour obtenir la figure à partir de  $LMN$ .

**Problème 1.** Construire une analoge FSD à partir de ABCD où le polygone ABCD est un carré.

La solution est donnée par le code Scheme :

```

(new-figure "General")

(define (carré p1 p2 p3 p4 n)

  (let* ((s1 (Segment "" extremities p1 p2))
         (s2 (Segment "" extremities p2 p3))
         (s3 (Segment "" extremities p3 p4))
         (s4 (Segment "" extremities p4 p1))
         (A (Point "" on-curve s1 1/10))
         (B (Point "" on-curve s2 1/10))
         (C (Point "" on-curve s3 1/10))
         (D (Point "" on-curve s4 1/10)))

    (send A masked)
    (send B masked)
    (send C masked)
    (send D masked)

    (if (> n 0)

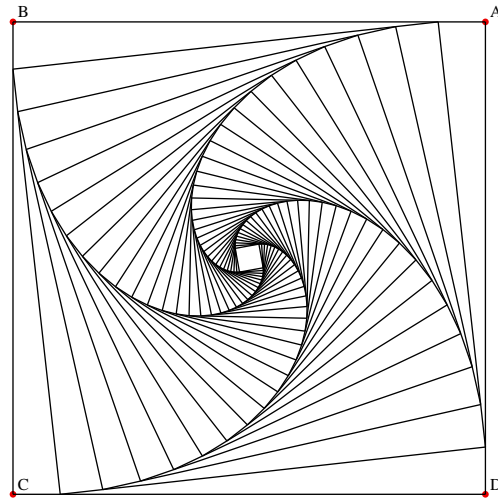
        (carré A B C D (- n 1))))

(lets Point "A" free 5 5)
(lets Point "B" free -5 5)
(lets Point "C" free -5 -5)
(lets Point "D" free 5 -5)

```

(carre A B C D 30)

La figure est :



**Figure 2.** Generalization

## Bibliographie

[1] H. Fernandes, *Manuel utilisateur de Dr. Geo*, Ofset, 2004.