Tutoriels Dr. Geo
Educatec 2004

Hilaire Fernandes

CRDP Aquitaine/OFSET
Pôle de compétence logiciels libres du SCEREN

Novembre 2004

Qu'est ce que Dr.Geo ?

- Dr.Geo est un logiciel libre de géométrie interactive
- Il est utilisable en primaire et secondaire
- Il comprend des fonctionnalités originales comme :

Qu'est ce que Dr.Geo ?

- Dr.Geo est un logiciel libre de géométrie interactive
- Il est utilisable en primaire et secondaire
- Il comprend des fonctionnalités originales comme :
  - Les macro-constructions

Qu'est ce que Dr.Geo ?

- Dr.Geo est un logiciel libre de géométrie interactive
- Il est utilisable en primaire et secondaire
- Il comprend des fonctionnalités originales comme :
  - Les macro-constructions
  - Les scripts Scheme

Qu'est ce que Dr.Geo ?

- Dr.Geo est un logiciel libre de géométrie interactive
- Il est utilisable en primaire et secondaire
- Il comprend des fonctionnalités originales comme :
  - Les macro-constructions
  - Les scripts Scheme
  - Des figures programmées en Scheme

Où trouver Dr.Geo ?

- Son site Internet : `http ://www.ofset.org/drgeo`
- Dans le cédérom live Freeduc-cd d'OFSET : `http ://www.ofset.org/freeduc-cd`
- Dans le cédérom primaire logiciels libres édité par le SCEREN
- Pour plus d'information : espace SCEREN/CRDP Aquitaine dans ce salon

Qu'est ce qu'une macro-construction ?

Définition

1. *Une macro-construction est une série de constructions enregistrée sous la forme d'une seule instruction. C'est la phase de construction d'une macro.*
2. *Lors de son utilisation, toute la série de constructions est réalisée en une fois. C'est la phase d'exécution d'une macro.*
3. *Cet enregistrement se dissocie de la figure pour être sauvé sur fichier.*

Qu'est ce qu'une macro-construction ?

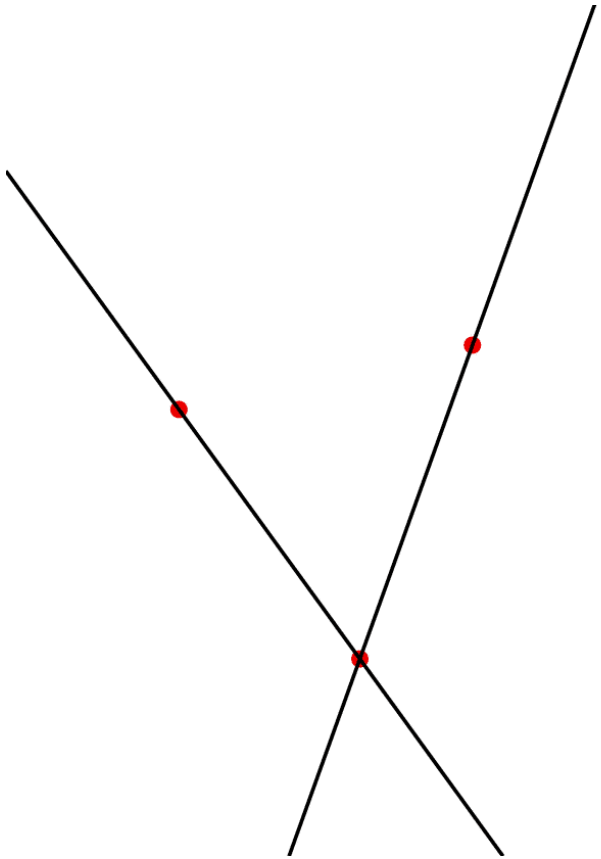
Définition

1. *Une macro-construction est une série de constructions enregistrée sous la forme d'une seule instruction. C'est la phase de construction d'une macro.*
2. *Lors de son utilisation, toute la série de constructions est réalisée en une fois. C'est la phase d'exécution d'une macro.*
3. *Cet enregistrement se dissocie de la figure pour être sauvé sur fichier.*

Définition

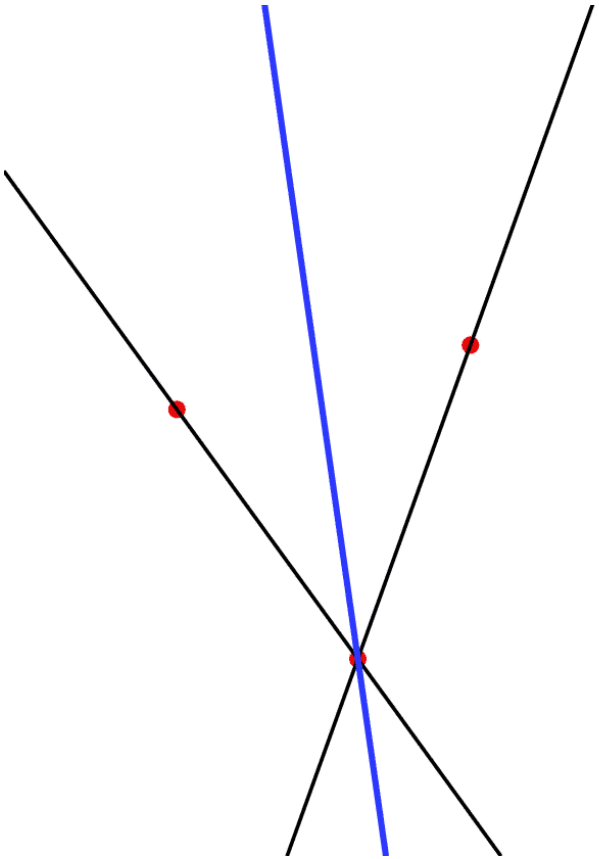
C'est un outil pour apprendre à Dr. Geo à faire des constructions complexes.

Une bissectrice, situation de départ



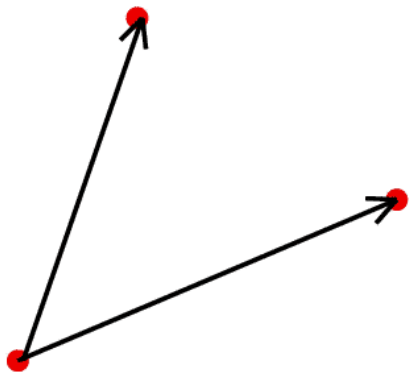
Situation de départ : trois points et deux droites.
Il s'agit donc d'apprendre au logiciel comment construire une bissectrice à partir de trois points.

Une bissectrice, résultat attendu



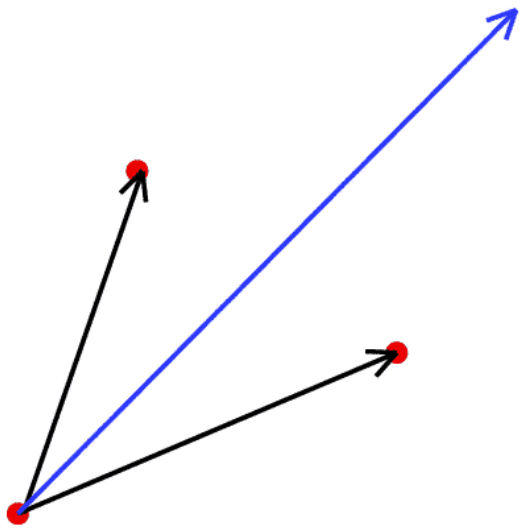
Exécution de la macro-construction sur les trois points.

Somme de vecteurs, situation de départ



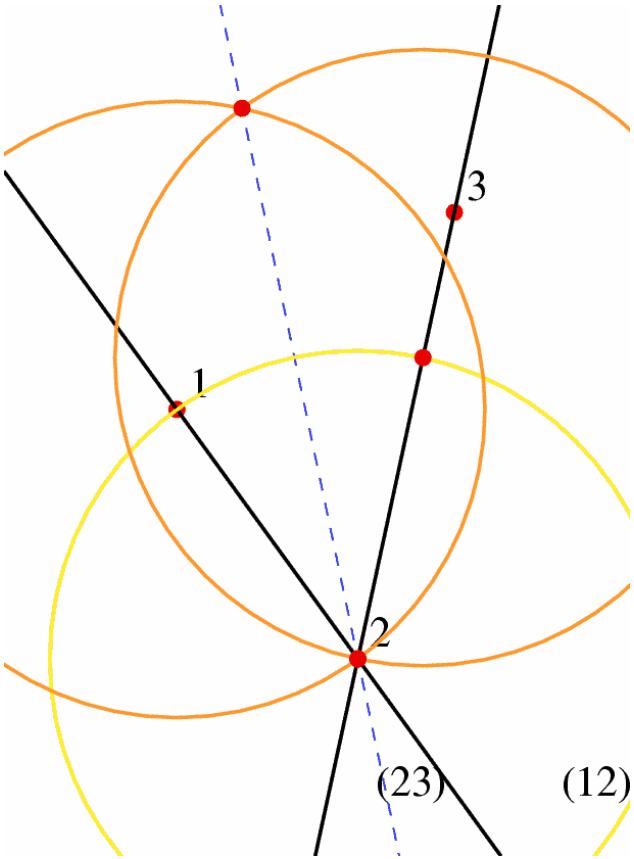
Situation de départ : trois points et deux vecteurs. Il s'agit donc d'apprendre au logiciel comment construire une somme de vecteurs à partir d'un point d'origine et de deux vecteurs.

Somme de vecteurs, résultat attendu



Exécution de la macro-construction sur un point d'origine et deux vecteurs.

Macro-construction d'une bissectrice

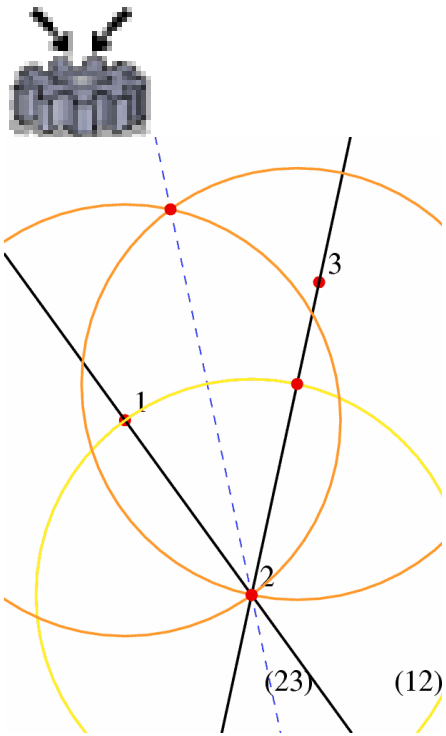


L'utilisateur résout le problème une fois.
Par exemple, en utilisant des cercles pour déterminer deux points de la bissectrice.

Macro-construction d'une bissectrice, construction

Choisir l'outil de construction de macro-construction

Macro-construction d'une bissectrice, construction



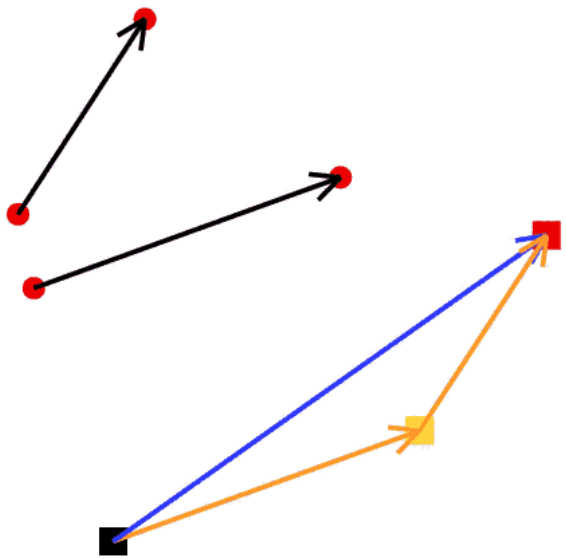
Choisir l'outil de construction de macro-construction

1. Sélectionner les trois points 1, 2 et 3 en objets d'entrée de la macro-construction
2. Sélectionner la droite en objet de sortie
3. Nommer et décrire notre macro
4. Valider la macro-construction

Macro-construction d'une bissectrice, utilisation

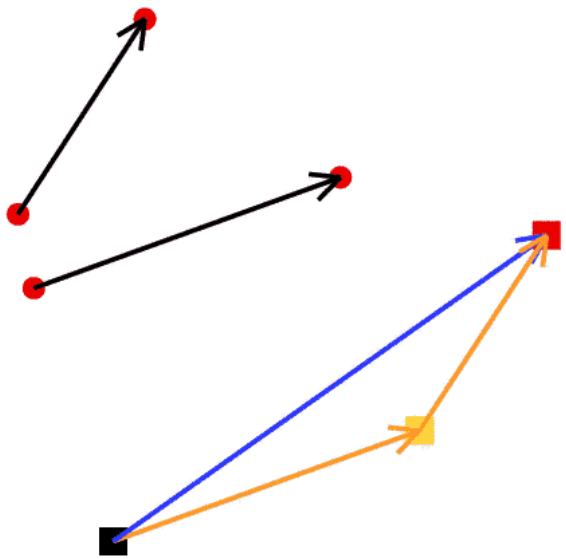
1. Choisir l'outil d'exécution de macro-construction
2. Sélectionner notre macro-construction
3. Cliquer sur trois points – les objets d'entrée
4. Résultat : la macro-construction crée immédiatement la bissectrice

Macro-construction d'une somme de vecteurs



L'utilisateur résout le problème en utilisant, par exemple, la translation.

Macro-construction d'une somme de vecteurs



L'utilisateur résout le problème en utilisant, par exemple, la translation.

1. les objets en entrée :
   - le point (noir et carré)
   - les deux vecteurs (noir)
2. l'objet de sortie, le vecteur (bleu)

Que retenir ?

- Créer une macro-construction, c'est apprendre à Dr. Geo à faire une construction donnée.

Que retenir ?

- Créer une macro-construction, c'est apprendre à Dr. Geo à faire une construction donnée.
- Une macro-construction attend des objets en entrée

Que retenir ?

- Créer une macro-construction, c'est apprendre à Dr. Geo à faire une construction donnée.
- Une macro-construction attend des objets en entrée
- Elle retourne des objets en sortie, résultat d'une construction déduite à partir des objets en entrée

Que retenir ?

- Créer une macro-construction, c'est apprendre à Dr. Geo à faire une construction donnée.
- Une macro-construction attend des objets en entrée
- Elle retourne des objets en sortie, résultat d'une construction déduite à partir des objets en entrée
- Tous les objets intermédiaires de la construction sont masqués

Que retenir ?

- Créer une macro-construction, c'est apprendre à Dr. Geo à faire une construction donnée.
- Une macro-construction attend des objets en entrée
- Elle retourne des objets en sortie, résultat d'une construction déduite à partir des objets en entrée
- Tous les objets intermédiaires de la construction sont masqués
- Les macro-constructions sont sauvées dans des fichiers, individuellement ou en groupe.

Qu'est ce qu'un script ?

Définition

1. *Un script est une fonction écrite dans le langage Scheme.*

Qu'est ce qu'un script ?

Définition

1. *Un script est une fonction écrite dans le langage Scheme.*
2. *Comme une fonction, il reçoit des arguments en entrée.*

Qu'est ce qu'un script ?

Définition

1. *Un script est une fonction écrite dans le langage Scheme.*
2. *Comme une fonction, il reçoit des arguments en entrée.*
3. *Toujours comme une fonction, il retourne une valeur numérique.*

Qu'est ce qu'un script ?

Définition

1. *Un script est une fonction écrite dans le langage Scheme.*
2. *Comme une fonction, il reçoit des arguments en entrée.*
3. *Toujours comme une fonction, il retourne une valeur numérique.*
4. *Cette valeur de retour est affichée dans la figure. Elle est utilisable par les constructions suivantes.*

Qu'est ce qu'un script ?

Définition

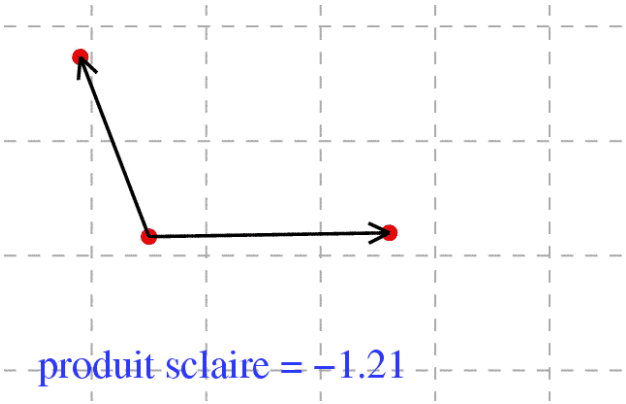
1. *Un script est une fonction écrite dans le langage Scheme.*
2. *Comme une fonction, il reçoit des arguments en entrée.*
3. *Toujours comme une fonction, il retourne une valeur numérique.*
4. *Cette valeur de retour est affichée dans la figure. Elle est utilisable par les constructions suivantes.*
5. *Les arguments en entrée sont des références vers d'autres objets de la figue courante.*

Qu'est ce qu'un script ?

Définition

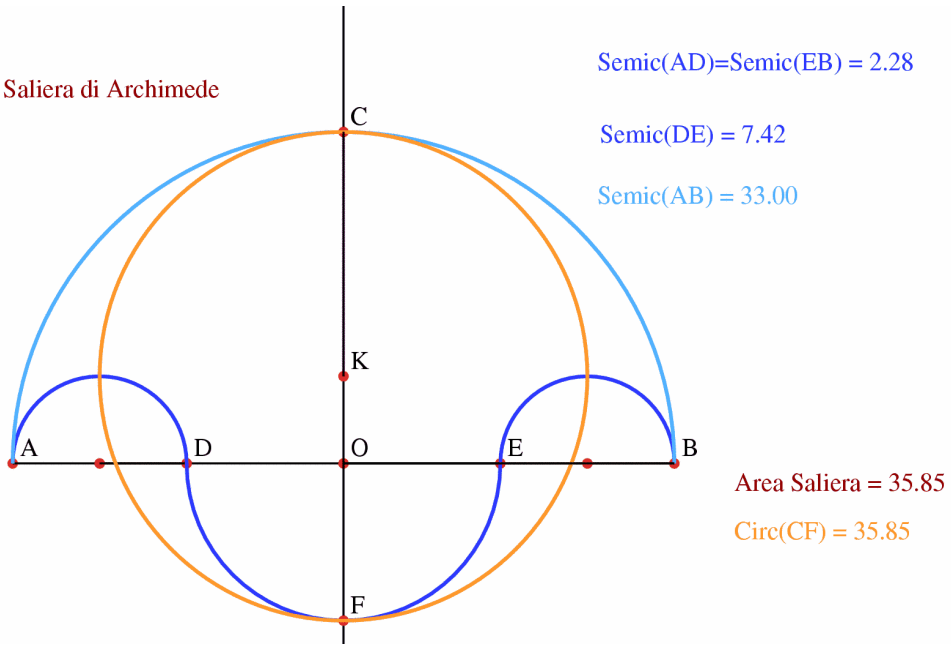
1. *Un script est une fonction écrite dans le langage Scheme.*
2. *Comme une fonction, il reçoit des arguments en entrée.*
3. *Toujours comme une fonction, il retourne une valeur numérique.*
4. *Cette valeur de retour est affichée dans la figure. Elle est utilisable par les constructions suivantes.*
5. *Les arguments en entrée sont des références vers d'autres objets de la figue courante.*
6. *Dr. Geo propose une API – interface de programmation – pour interroger ces objets.*

Calcul du produit scalaire



Situation de départ : deux vecteurs.
Nous souhaitons calculer le produit scalaire de ces deux vecteurs à l'aide d'un script.

Calculs d'aires et comparaison

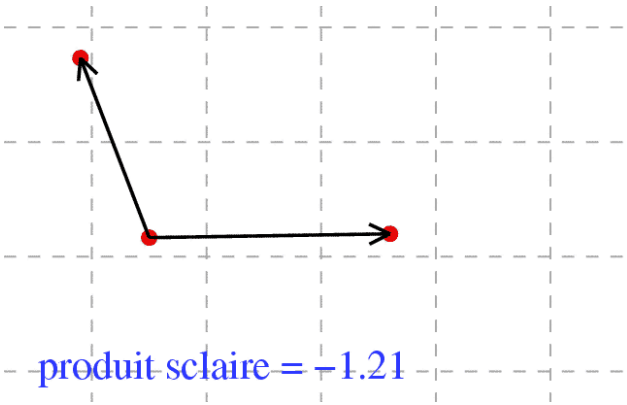


Situation de départ : La saliera d'Archimèdes.
Nous souhaitons calculer les aires des zones délimitées par la ligne bleu et le cercle.
Nous pouvons utiliser des scripts en cascade pour ces calculs.

Calcul du produit scalaire, créer le script

Choisir l'outil pour créer un script.

Calcul du produit scalaire, créer le script



Choisir l'outil pour créer un script.

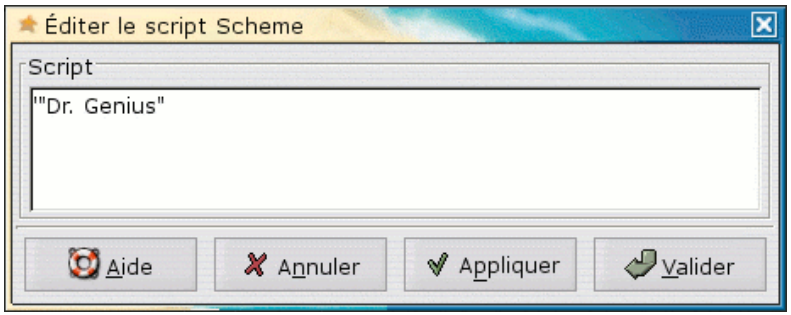
1. Dans la figure, sélectionner les deux vecteurs en arguments d'entrée du script.
2. Cliquer dans le fond de la figure afin d'y placer le script.
3. Le message *Dr. Genius* s'affiche à cet emplacement, c'est ce sur quoi est ancré le script.

Calcul du produit scalaire, éditer le script

Choisir l'outil de propriétés des objets et cliquer sur le script.

Calcul du produit scalaire, éditer le script

Choisir l'outil de propriétés des objets et cliquer sur le script.



Remplacer le contenu du script par :

```
(let  ((u (getCoordinates a1))
       (v (getCoordinates a2)))
   (+ (* (car u) (car v))
      (* (cadr u) (cadr v))))
```

et appliquer les modifications.

Calcul du produit scalaire, explications 1/2

```
(let  ((u (getCoordinates a1))
       (v (getCoordinates a2)))
```

- a1 et a2 sont les deux arguments du script. Ce sont des références vers nos deux vecteurs.
- `getCoordinates` est une fonction qui retourne une liste des coordonnées d'un vecteur.

Calcul du produit scalaire, explications 2/2

```
(+ (* (car u) (car v))
   (* (cadr u) (cadr v))))
```

`car` et `cadr` permettent de récupérer l'abscisse et l'ordonnée de la liste des coordonnées

Associer script et macro-construction

Cela permet d'étendre Dr.Geo par des *macro-scripts* pour :

- calculer le produit scalaire de deux vecteurs
- calculer une mesure algébrique
- construire un polygone
- et bien d'autres choses...

Que retenir ?

- Un script est une fonction qui retourne une valeur affichée dans la figure.

Que retenir ?

- Un script est une fonction qui retourne une valeur affichée dans la figure.
- Il reçoit des arguments en entrée, références vers d'autres objets de la figure.

Que retenir ?

- Un script est une fonction qui retourne une valeur affichée dans la figure.
- Il reçoit des arguments en entrée, références vers d'autres objets de la figure.
- Dr. Geo propose une API pour interroger depuis un script des objets de la figure.

Que retenir ?

- Un script est une fonction qui retourne une valeur affichée dans la figure.
- Il reçoit des arguments en entrée, références vers d'autres objets de la figure.
- Dr. Geo propose une API pour interroger depuis un script des objets de la figure.
- Les scripts peuvent étendre les possibilités des macro-constructions : les *macro-scripts*.

Qu'est ce qu'une figure programmée ?

Définition

1. *C'est une figure décrite dans un langage semi-naturel.*

Qu'est ce qu'une figure programmée ?

Définition

1. *C'est une figure décrite dans un langage semi-naturel.*
2. *Le langage informatique est Scheme, une variante de Lisp.*

Qu'est ce qu'une figure programmée ?

Définition

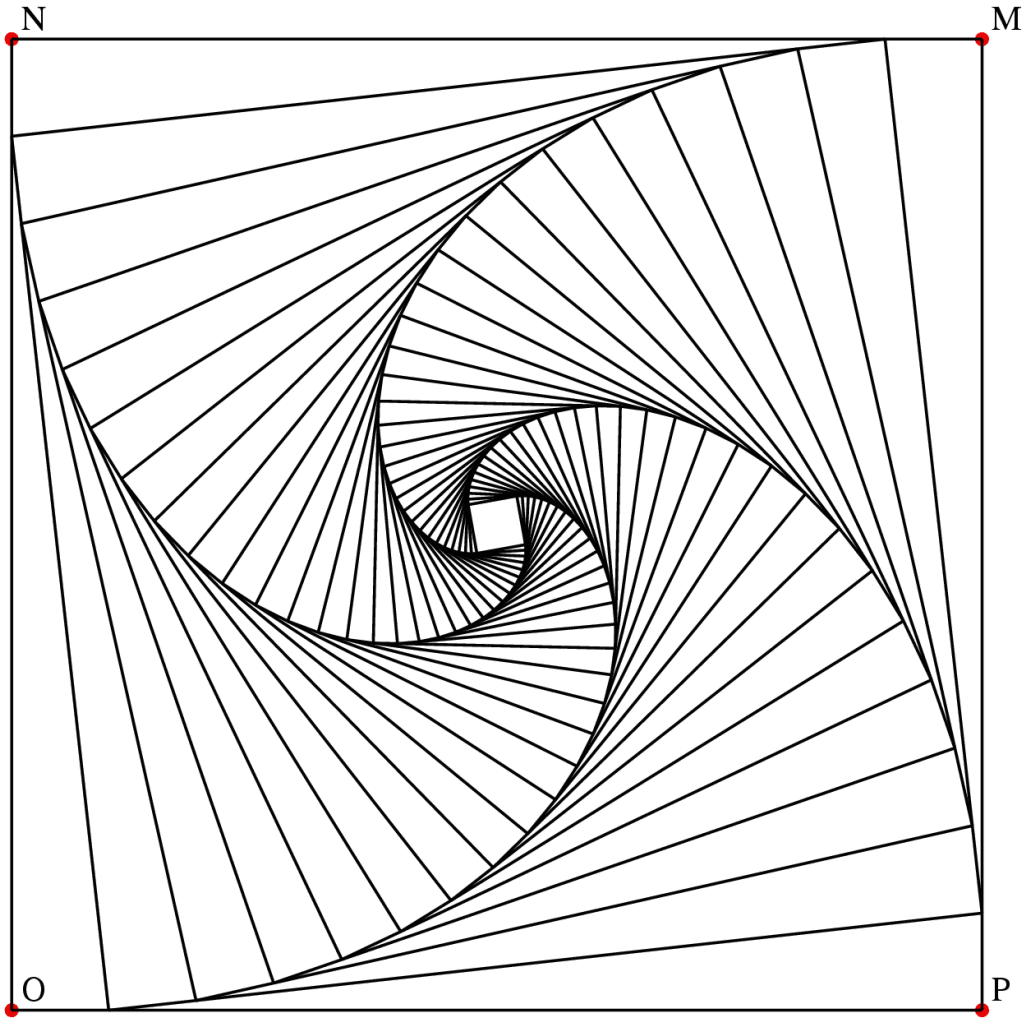
1. *C'est une figure décrite dans un langage semi-naturel.*
2. *Le langage informatique est Scheme, une variante de Lisp.*
3. *Le langage permet une construction itérative ou récursive d'une figure.*

Qu'est ce qu'une figure programmée ?

Définition

1. *C'est une figure décrite dans un langage semi-naturel.*
2. *Le langage informatique est Scheme, une variante de Lisp.*
3. *Le langage permet une construction itérative ou récursive d'une figure.*
4. *Dr. Geo propose une API – interface de programmation – entièrement documentée.*

Spirale – Figure



Spirale – Description

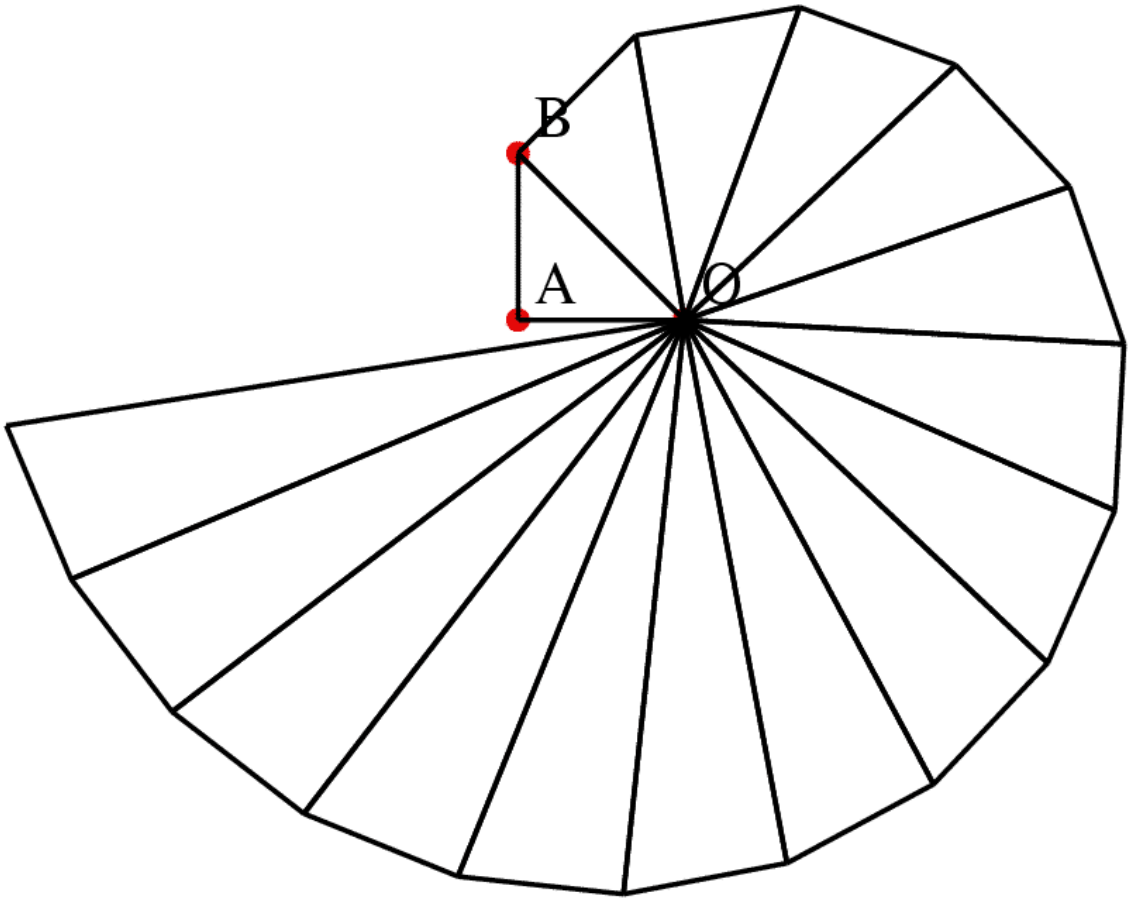
```
(nouvelle-figure "Spirale")

(define (square 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   "" sur-ligne    s1   1/10))
          (B   (Point   "" sur-ligne    s2   1/10))
          (C   (Point   "" sur-ligne    s3   1/10))
          (D   (Point   "" sur-ligne    s4   1/10)))
     (envoi A masquer)
     (envoi B masquer)
     (envoi C masquer)
     (envoi D masquer)
 (if (> n 0)
     (square  A B C D (- n 1)))))

(soit Point "M" libre  5  5)
(soit Point "N" libre -5  5)
(soit Point "O" libre -5 -5)
(soit Point "P" libre  5 -5)

(square M N O P 30)
```

Construction des nombres premiers – Figure



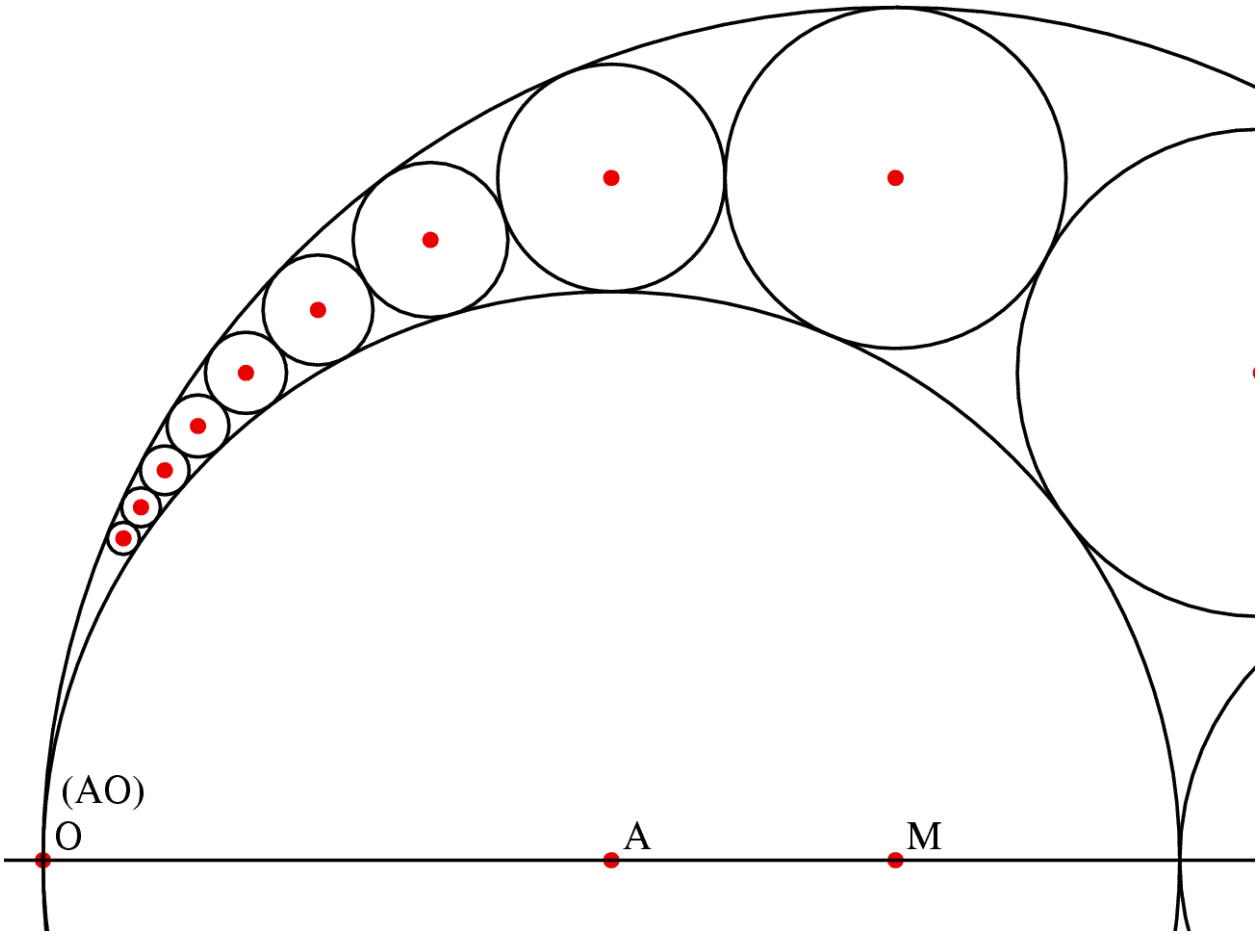
Construction des nombres premiers – Description

```
(nouvelle-figure "Triangle")

(define (triangle p1 p2 p3 n)
  (let* ((s1 (Segment "" extremities    p1 p2))
         (s2 (Segment "" extremities    p2 p3))
         (s3 (Segment "" extremities    p3 p1))
         (pe (Line    "" perpendiculaire  p3 s3))
         (ci (Circle  "" centre-segment p3 s2))
         (p4 (Point   "" intersection2  pe ci)))
    (envoi pe masquer)
    (envoi ci masquer)
    (envoi p4 masquer)
    (if (> n 0)
       (triangle p1 p3 p4 (- n 1)))))

(soit Point "O" libre  0  0)
(soit Point "A" libre -1  0)
(soit Point "B" libre -1  1)
(triangle O A B 15)
```

Pappo – Figure

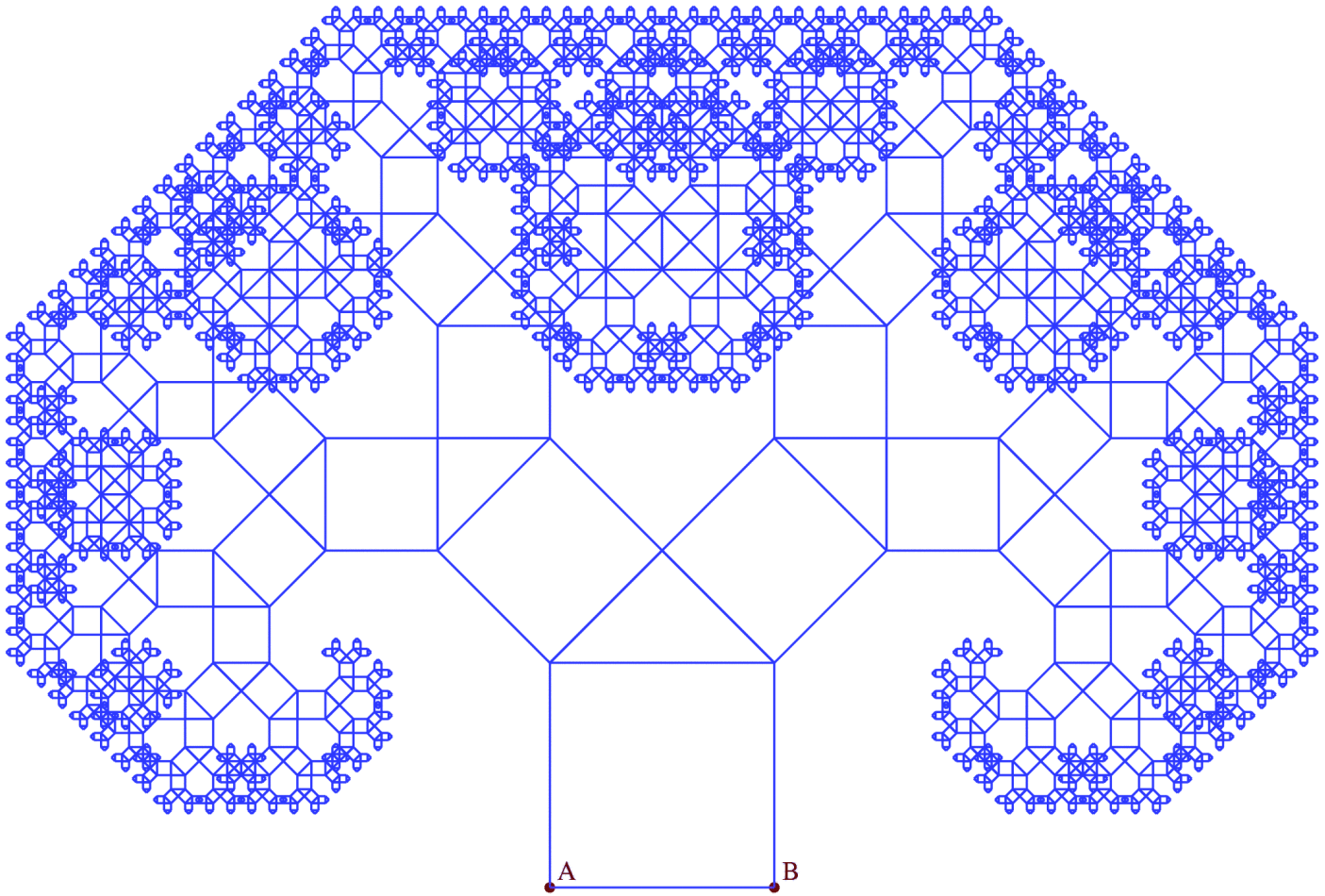


Pappo – Description

```
(nouvelle-figure "Pappo")
(define (circle n)
(let*((r (Numeric "" libre 0 0 (/ 15 ( + 6 (* n n )))))
      (c (Point   "" libre (* 5 (/ 15 ( + 6 (* n n ))))
                           (* 2 (* n (/ 15 ( + 6 (* n n )))))))
      (p (Circle "" centre-rayon c r )))
  (envoi r masquer)
  (if (> n 0)
    (circle (- n 1)))))

(circle 10)
(soit Point "A" libre 5 0)
(soit Point "O" libre 0   0)
(soit Point "B" libre 15 0)
(soit Point "M" milieu-2pts B O)
(soit Circle "" 2points M O)
(soit Circle "" 2points A O)
(soit Line   "" 2points A O)
```

Fractale – Figure

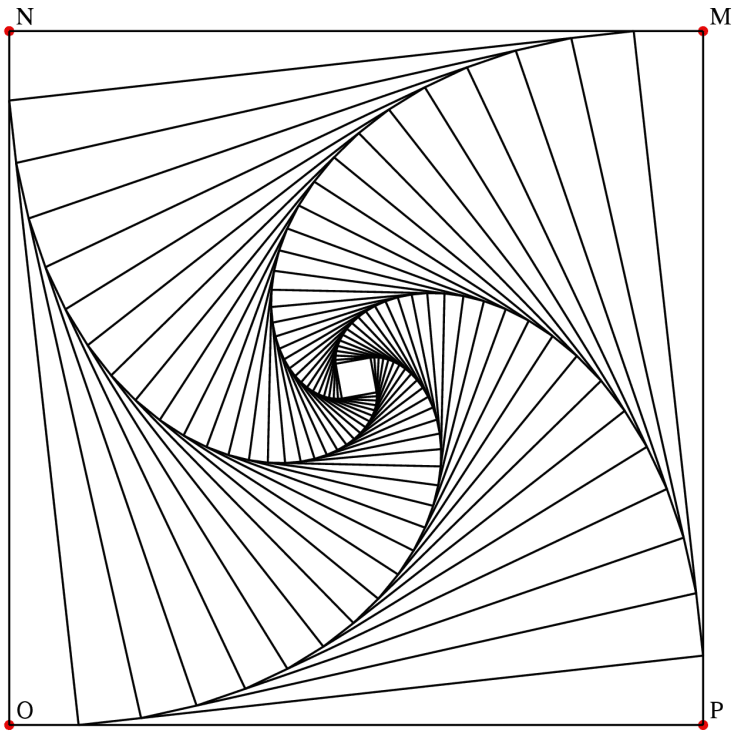


Fractale – Description

```
(nouvelle-figure "Une belle Fractale")
(soit Numeric "a1" libre 1 2 (acos 0))
(soit Numeric "a2" libre 1 3 (- 0 (acos 0)))
(envoi a1 masquer)
(envoi a2 masquer)
(define (pythagore p1 p2 n)
   (envoi p1 masquer)
   (envoi p2 masquer)
   (let* (
      (p4 (Point "" rotation p2 p1 a1))
      (p3 (Point "" rotation p1 p2 a2))
      (s1 (Segment ""extremities p1 p2))
      (s2 (Segment "" extremities p2 p3))
      (s3 (Segment "" extremities p3 p4))
      (s4 (Segment "" extremities p4 p1))
      (O (Point "" milieu-2pts p3 p4))
      (M (Point "" rotation p3 O a1))
      (s5 (Segment "" extremities p4 M))
      (s6 (Segment "" extremities M p3)))
      (envoi p1 masquer)
      (envoi p2 masquer)
      (envoi p3 masquer)
      (envoi p4 masquer)
      (envoi O masquer)
      (envoi M masquer)
      (if (> n 0)
         (begin
            (pythagore p4 M (- n 1))
            (pythagore M p3 (- n 1))))))

(soit Point "A" libre -5 0)
(soit Point "B" libre -2 0)
(pythagore A B 10)
```

Spirale – Explications



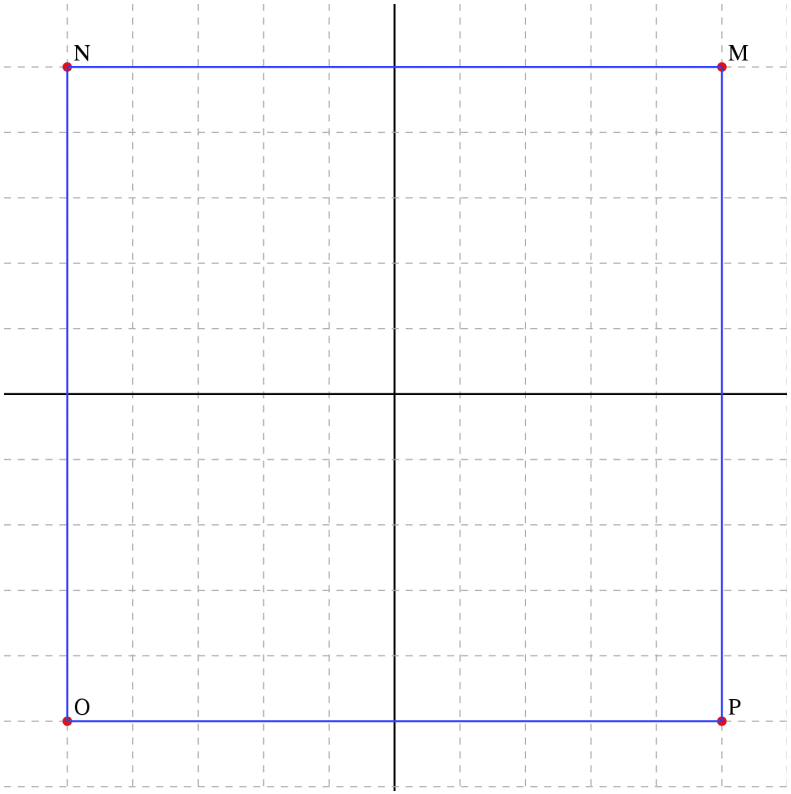

```
(nouvelle-figure "Spirale")

(define (square 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   "" sur-ligne    s1   1/10))
          (B   (Point   "" sur-ligne    s2   1/10))
          (C   (Point   "" sur-ligne    s3   1/10))
          (D   (Point   "" sur-ligne    s4   1/10)))
     (envoi A masquer)
     (envoi B masquer)
     (envoi C masquer)
     (envoi D masquer)
 (if (> n 0)
     (square  A B C D (- n 1)))))

(soit Point "M" libre  5  5)
(soit Point "N" libre -5  5)
(soit Point "O" libre -5 -5)
(soit Point "P" libre  5 -5)

(square M N O P 30)
```

Spirale – Récursivité de profondeur 0



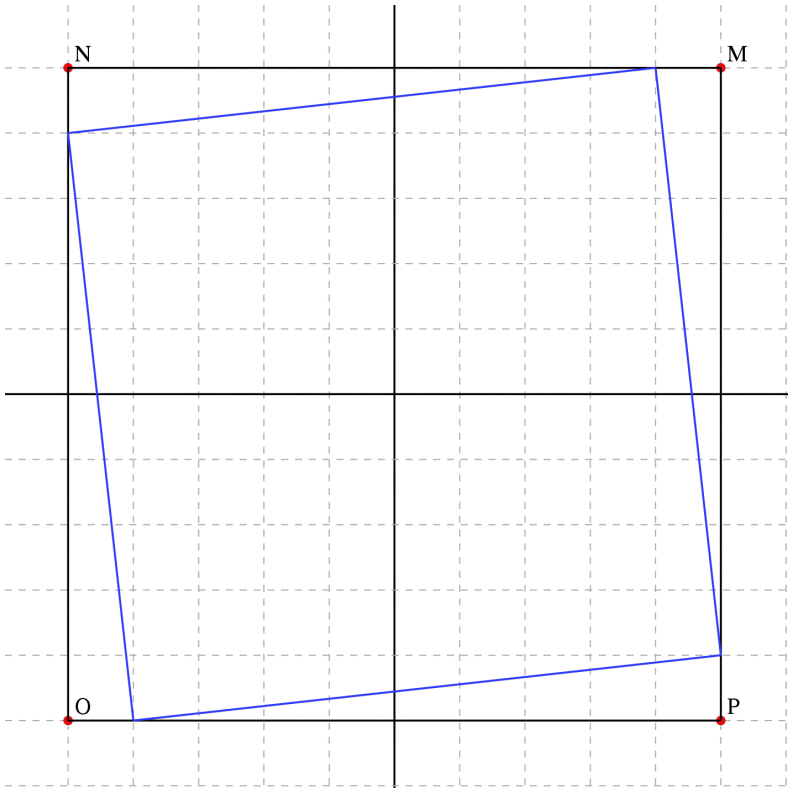

```
(nouvelle-figure "Spirale")

(define (square 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   "" sur-ligne    s1   1/10))
          (B   (Point   "" sur-ligne    s2   1/10))
          (C   (Point   "" sur-ligne    s3   1/10))
          (D   (Point   "" sur-ligne    s4   1/10)))
     (envoi A masquer)
     (envoi B masquer)
     (envoi C masquer)
     (envoi D masquer)
 (if (> n 0)
     (square  A B C D (- n 1)))))

(soit Point "M" libre  5  5)
(soit Point "N" libre -5  5)
(soit Point "O" libre -5 -5)
(soit Point "P" libre  5 -5)

(square M N O P 0)
```

Spirale – Récursivité de profondeur 1



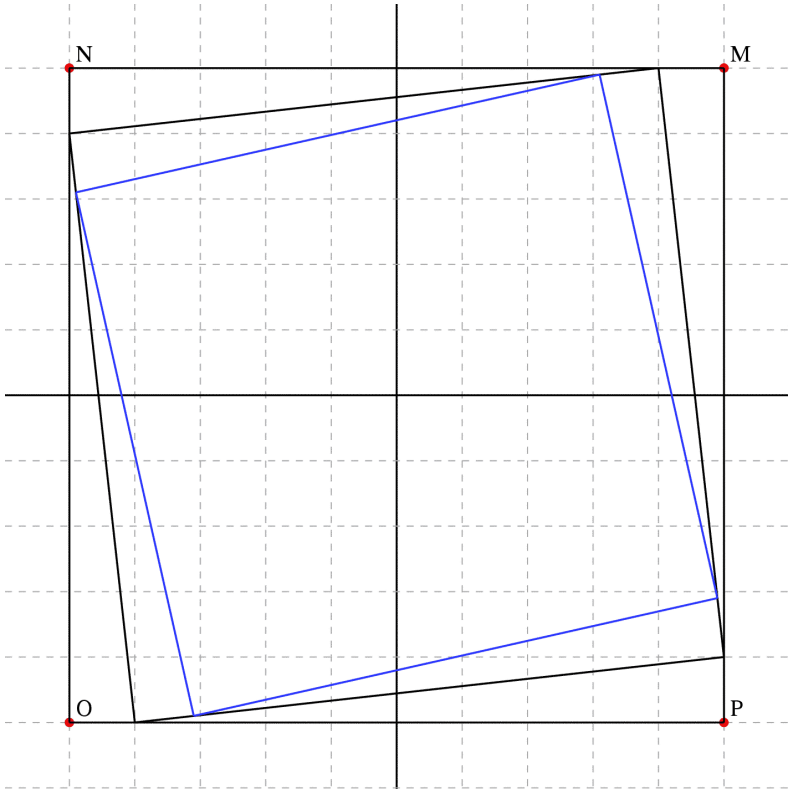

```
(nouvelle-figure "Spirale")

(define (square 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   "" sur-ligne    s1   1/10))
          (B   (Point   "" sur-ligne    s2   1/10))
          (C   (Point   "" sur-ligne    s3   1/10))
          (D   (Point   "" sur-ligne    s4   1/10)))
     (envoi A masquer)
     (envoi B masquer)
     (envoi C masquer)
     (envoi D masquer)
 (if (> n 0)
     (square  A B C D (- n 1)))))

(soit Point "M" libre  5  5)
(soit Point "N" libre -5  5)
(soit Point "O" libre -5 -5)
(soit Point "P" libre  5 -5)

(square M N O P 1)
```

Spirale – Récursivité de profondeur 2




```
(nouvelle-figure "Spirale")

(define (square 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   "" sur-ligne    s1   1/10))
          (B   (Point   "" sur-ligne    s2   1/10))
          (C   (Point   "" sur-ligne    s3   1/10))
          (D   (Point   "" sur-ligne    s4   1/10)))
     (envoi A masquer)
     (envoi B masquer)
     (envoi C masquer)
     (envoi D masquer)
 (if (> n 0)
     (square  A B C D (- n 1)))))

(soit Point "M" libre  5  5)
(soit Point "N" libre -5  5)
(soit Point "O" libre -5 -5)
(soit Point "P" libre  5 -5)

(square M N O P 2)
```

Spirale – Principales commandes 1/

Extrait de code

```
(soit Point "M" libre 5 5)
```

Déclaration d'un point libre, nommé M et de coordonnées (5 ;5). `soit` est un mot-clé pour créer un objet, l'objet créé est accessible depuis la variable `M`.

Spirale – Principales commandes 2/

Extrait de code

```
(let* ((s1  (Segment "" extremities p1 p2))
       (A   (Point   "" sur-ligne  s1   1/10))
```

`let*` contient un bloc de déclaration de variables locales à la fonction `square`.
Au lieu du mot-clé `soit`, des appels de fonctions permettent de créer les objets géométriques.
Le résultat de ces appels est affecté aux variables `s1`, `A`, etc.

Spirale – Principales commandes 3/

Extrait de code

```
(Point   ""  sur-ligne  s1  1/10)
```

Cet appel crée un point sur la ligne `s1` avec comme abscisse curviligne 1/10.
Quelque soit le type de ligne (droite, demi-droite, cercle, etc), les points qui y sont placés ont une abscisse curviligne comprise dans [0 ;1].

Figure programmée – Quelques conseils

- Uitiliser un éditeur de texte adapté au langage Scheme – ici nous avons utilisé l'éditeur SciTe. Il doit gérer au minimum :

Figure programmée – Quelques conseils

- Uitiliser un éditeur de texte adapté au langage Scheme – ici nous avons utilisé l'éditeur SciTe. Il doit gérer au minimum :
  - la colorisation syntaxique

Figure programmée – Quelques conseils

- Uitiliser un éditeur de texte adapté au langage Scheme – ici nous avons utilisé l'éditeur SciTe. Il doit gérer au minimum :
  - la colorisation syntaxique
  - l'appareillement des parenthèses – elles sont nombreuses en Scheme et il est suicidaire de ne pas avoir un outil adapté.

Figure programmée – Quelques conseils

- Uitiliser un éditeur de texte adapté au langage Scheme – ici nous avons utilisé l'éditeur SciTe. Il doit gérer au minimum :
  - la colorisation syntaxique
  - l'appareillement des parenthèses – elles sont nombreuses en Scheme et il est suicidaire de ne pas avoir un outil adapté.
- Bien étudier la documentation de Dr. Geo et les exemples fournis.

Figure programmée – Quelques conseils

- Uitiliser un éditeur de texte adapté au langage Scheme – ici nous avons utilisé l'éditeur SciTe. Il doit gérer au minimum :
  - la colorisation syntaxique
  - l'appareillement des parenthèses – elles sont nombreuses en Scheme et il est suicidaire de ne pas avoir un outil adapté.
- Bien étudier la documentation de Dr. Geo et les exemples fournis.
- Se familiariser avec le langage Scheme – Dr. Scheme est un environnement libre pour son apprentissage.

Merci de votre attention

http ://www.ofset.org/drgeo