DEUG MIAS 12
TP en solitaire - "Triangles en folie"

Frédéric Peschanski

25 octobre 2000

Vous devez répondre aux questions directement dans l'environnement DR SCHEME.

IMPORTANT 1 : Les fonctions Scheme que vous serez amenés à définir pour répondre à ce sujet doivent être correctement **commentées** et **indentées**.

IMPORTANT 2 : Les définitions de fonction et les exemples d'utilisation doivent être saisis dans la zone de définition en haut de l'écran.

Préliminaires

Dans l'environnement DR SCHEME, sélectionnez le mode de fonctionnement "*Advanced*" et vérifiez que la bibliothèque "*concabs.ss*" est bien chargée.

Vous devez enregistrer votre travail dans un fichier dont le nom est `VotreNom.sc` où `VotreNom` doit être remplacé par votre nom de famille. **Vous devez également indiquer votre nom en commentaire dans le fichier lui-même, sur la première ligne.**

Attention : le TP ne pourra être corrigé que si un fichier à votre nom est présent !
N'oubliez pas de sauvegarder régulièrement (bouton SAVE) votre travail dans ce même fichier.

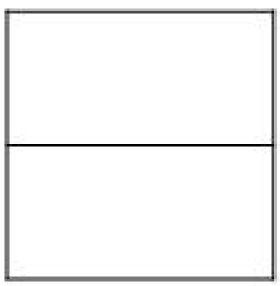
Question 1 : triangle isocèle vide

1.1 Fonction ligne

La bibliothèque "*concabs.ss*" fournit une fonction `line` permettant de dessiner des segments de droite. La spécification de cette fonction est la suivante :

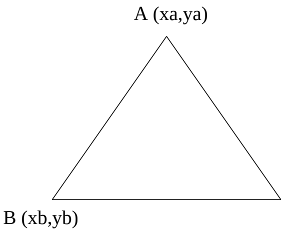
```
;; Dessin de segment de droite
;; float * float * float * float -> image
;; x1 y1 => coordonnées du premier point
;; x2 y2 => coordonnées du deuxième point
(define (line x1 y1 x2 y2) ... )
```

Utiliser cette fonction pour découper horizontalement la zone d'affichage, comme montré sur la figure suivante :



1.2 Triangle isocèle

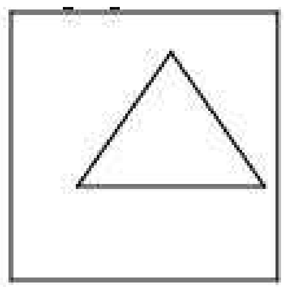
Définir une fonction (`triangle-isocele xa ya xb yb`) permettant de dessiner un triangle isocèle vide, selon la figure suivante :



En remarquant l'axe de symétrie (verticale passant par A) de ce triangle, il sera utile de définir la fonction intermédiaire suivante (déjà vue en TP) :

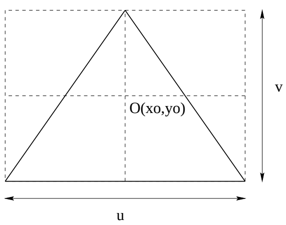
```
;; Fonction de symetrie
;; float*float -> float
(define (sym x xaxe)
   (- (* 2 xaxe) x))
```

Tester la fonction `triangle-isocele` avec l'expression `(triangle-isocele 0.2 0.7 -0.5 -0.3)`, le résultat attendu est le suivant :



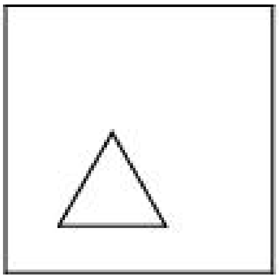
Question 2 : nouveau paramétrage

Ecrire une fonction `(triangle-isocele-2 xo yo u v)` permettant de dessiner une triangle isocèle avec un paramétrage différent de celui de la question 1.2. L'idée est de représenter le triangle à partir des informations concernant son rectangle englobant (coordonnées du centre O et largeur/hauteur du rectangle), comme indiqué sur la figure suivante :



Remarque : Le corps de la fonction `triangle-isocele-2` doit utiliser un appel à la fonction `triangle-isocele` définie à la question 1.2

Tester la fonction `triangle-isocele-2` avec l'expression `(triangle-isocele-2 -0.2 -0.3 0.8 0.7)`, le résultat obtenu doit ressembler à :

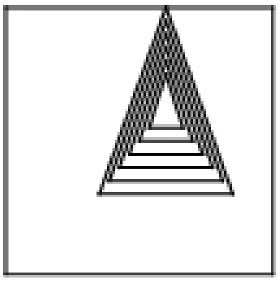


Question 3 : Dans l'hyper-espace

3.1. Hyper-espace, premier paramétrage

Définir une fonction `(hyper-espace-impl xo yo u du v dv nb)` qui dessine une superposition de `nb` triangles. Chaque triangle est centré en (`xo`, `yo`) et dispose d'une taille (`u`,`v`) (cf. question précédente). Le premier triangle à une largeur `u`, le deuxième `(u-du)`, et ainsi de suite. Il faut dessiner en tout `nb` triangles

Tester votre fonction avec l'expression `(hyper-espace-impl 0.2 0.3 1 0.15 1.4 0.2 6)`, le résultat à l'écran est le suivant :

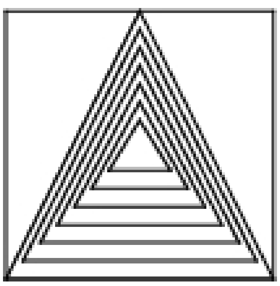


3.2. Hyper-espace simple

La fonction précédente n'est pas facile à utiliser car le paramétrage n'est pas évident à maitriser (notamment pour éviter les erreurs). Pour simplifier son utilisation, nous proposons la définition d'une fonction intermédiaire `(hyper-espace xo yo ug up vg vp nb)` dont le paramétrage est le suivant :

Paramètre	Rôle
xo, yo	Centre de l'hyper-espace
ug	largeur du plus grand triangle
up	largeur du plus petit triangle
vg	hauteur du plus grand triangle
vp	hauteur du plus petit triangle
nb	nombre de triangles dans la figure

Ainsi, en évaluant l'expression `(hyper-espace 0 0 2 0.2 2 0.1 7)`, nous obtenons la figure suivante :



(attention : lisez la remarque page suivante)

Remarque : Le corps de la fonction `hyper-espace` doit effectuer un appel à la fonction `hyper-espace-impl` avec les bons paramètres.

Question 4 ("bonus") : hyper-hyper-espace

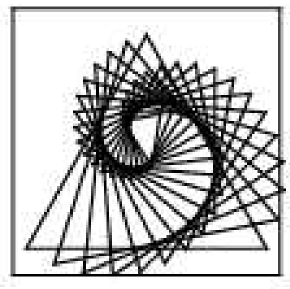
S'il vous reste du temps ou si vous voulez aller plus loin (chez vous), nous proposons une variante à notre dessin d'hyper-espace. L'idée est d'ajouter une rotation (centrée sur l'origine du repère) aux triangles successifs : le permier triangle subit une rotation de 0 radian, le deuxième de $d\theta$, le troisième de $2 * d\theta$, etc.

Comme pour la question 3, nous décomposerons le problème en deux fonctions : l'une, `hyper-hyper-espace-impl`, construisant notre figure et l'autre, `hyper-hyper-espace`, plus simple d'utilisation. Par rapport à l'exercice 3, les seuls paramètres à ajouter sont l'angle `theta` et la variation de l'angle `dtheta` pour la version "impl". On supposera pour la version "simple d'utilisation" que l'angle de départ est 0.

Les en-têtes de fonctions seront donc les suivants :

```
(define (hyper-hyper-espace-impl xo yo u du v dv theta dtheta nb) (...))
(define (hyper-hyper-espace xo yo ug up vg vp dtheta nb) (...))
```

Au final, l'expression `(hyper-hyper-espace 0 0 1.8 0.2 1.6 0.3 (/ pi 16) 18)` doit fournir l'affichage suivant :



Il sera utile de définir une fonction intermédiaire `triangle-rot` , proche de `triangle-isocele` mais avec un paramètre `theta` supplémentaire pour la rotation. Une version simple `triangle-rot-2` sera également intéressante.

Finalement, nous vous "offrons" les formules de rotation, implémentées par les fonctions suivantes (à saisir dans la zone de définitions) :

```
;; Fonction de rotation sur l'axe x , autour de l'origine
;; float * float * float -> float
(define (rotx x y theta)
    (- (* x (cos theta))
       (* y (sin theta))))

;; Fonction de rotation sur l'axe y, autour de l'origine
;; float * float * float -> float
(define (roty x y theta)
    (+ (* x (sin theta))
       (* y (cos theta))))
```