SCHEME TORTUE (Version 2011)

(***Version avec Racket Langage : « Assez gros Scheme »)***

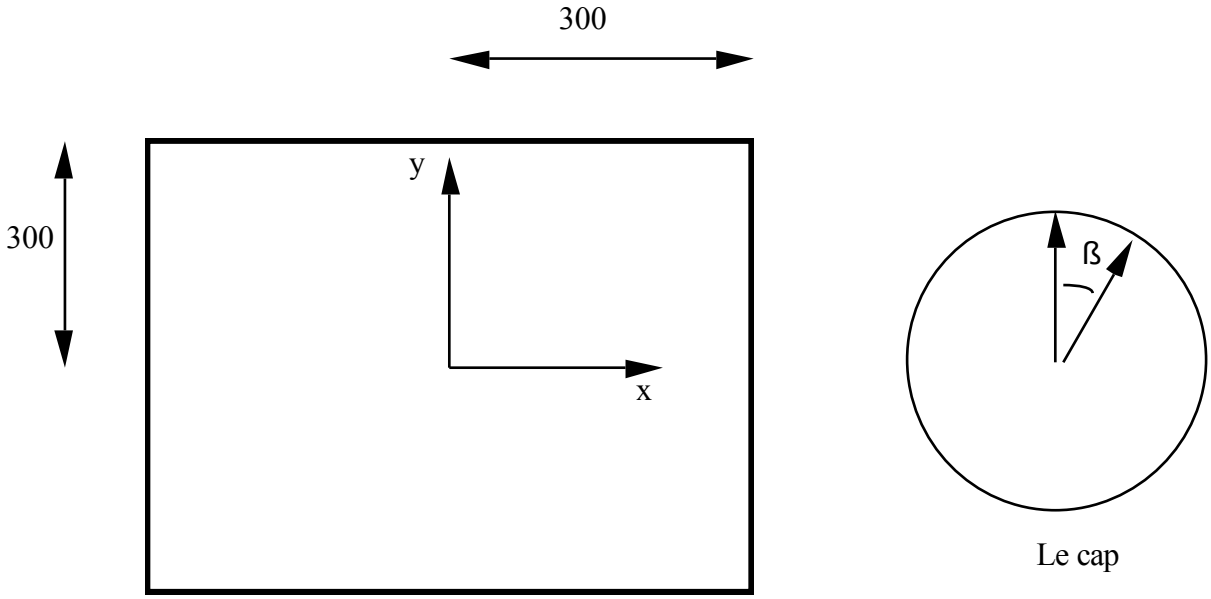
Le langage SCHEME comporte une couche graphique qui n'est pas tres facile à manipuler. Nous avons ajouté au langage une couche "tortue" , comme le fait le langage LOGO, qui permet de faire des dessins en promenant une tortue sur l'écran : la tortue est censée porter un crayon sur le ventre et si elle se déplace le crayon baissé, cela fait un trait, si le crayon est levé , il ne se passe rien.

Après avoir chargé DrScheme, charger le fichier "schemetortue.scm par :

```
>>> (load ......................."tortuegraphique.scm" )
; les pointillés représentant le chemin pour accèder au fichier
; le chemin doit apparaître sous la forme "C:\\....\\....\\tortuegraphique.scm"
; c'est à dire une chaîne de caractères (entre "...") chaque directory étant précédé
de 2 \\
```

Un truc est de se placer dans le dossier où se trouve le fichier tortuegraphique.scm, de copier le chemin dans l'Explorateur, de le coller dans DrScheme derrière load et de remplacer les simples \ par des doubles \\ et ne pas oublier les guillemets pour la chaîne de caractères.

Une fenêtre graphique s'ouvre et occupe une bonne partie de l'écran. Elle occupe (600 x 600) pixels et l'origine est au centre de la fenêtre soit :



La tortue est repérée par trois informations : ses coordonnées (x,y) et son cap qui est l'angle vers lequel elle se dirige : l'axe des y correspondant au cap = 0 et l'axe des x, au cap = 90 par exemple.

Vous disposez des primitives suivantes :

primitive complète	synonyme	signification
levecrayon	lc	lève le rayon
baissecrayon	bc	baisse le crayon
cap		demande le cap de la tortue, retourne un nombre n ($0 \leq n \leq 360$)
fcap		fixe le cap de la tortue
pos		demande la position de la tortue; retourne une liste de deux nombres
fxy		fixe la position de la tortue, les deux coordonnées en arguments
fpos	td	fixe la position de la tortue, les deux coordonnées en liste
tournedroite	tg	fait tourner à droite d'un certain angle le cap de la tortue
tournegauche	av	fait tourner à gauche d'un certain angle le cap de la tortue
avance	re	fait avancer la tortue d'un certain nombre de pixels
recule		fait reculer la tortue d'un certain nombre de pixels
nettoie		nettoie l'écran et place la tortue à l'origine, cap = 0
centre		place la tortue à l'origine, cap = 0 sans nettoyer
repeat		permet de répéter des opérations successives
radiantodegre		transforme un angle en radian vers sa valeur en degré
degretoradian		transforme un angle en degré vers sa valeur en radian

exemples :

```
>>> (fxy  100 200)
```

mais

```
>>> (fpos '(100 200))

>>>(repeat 4 (av 100) (td 90))
```

TP SCHEME TORTUE

TP n°1

Après avoir chargé le fichier "SCHEMETORTUE" , écrire les procédures suivantes :

Des polygones...

1 Ecrire la procédure qui permet de dessiner un carre de longueur 100 avec la primitive `repeat`

Pourriez vous écrire la même procédure mais de façon récursive, sans la primitive repeat - qui est impérative !- ?

2 Ecrire la procédure qui permet de dessiner un carre de longueur X, X étant un argument.

3 En procédant de la même façon, écrire les procédures qui permettent de dessiner un triangle équilateral et un hexagone puis un polygone dont on passera en argument X la longueur du coté et n , le nombre de côtés.

Que se passe-t-il quand avec la procédure polygone, on prend n grand (n = 18, 20, 36 par exemple !) ?

La tortue n'a qu'une vision locale des choses; pour faire un cercle il faut donc la faire avancer de X, puis tourner de ß, et repeter l'opération jusqu'à ce qu'elle retourne à son point départ.

4 En s'inspirant de la procédure `polygone`, écrire la procédure `cercle` qui la fait avancer de 10 pixels puis tourner de 10° jusqu'à retourner à son point de départ

5 Ecrire la procédure cercle dans laquelle le rayon est passé en argument : il vous faut choisir X et ß de telle façon que le cercle ait le rayon voulu. (Vous essaierez ensuite de passer les coordonnées du centre en arguments)

6 A partir de la procédure `carre`, écrire la procédure `carre_tournant` qui repete N fois le dessin d'un carre en tournant entre chaque dessin de ß= $\frac{360}{N}$ (X, la longueur du côté et N, étant passés en argument !)

Des spirales ....

7 Ecrire la procédure- test qui permet de savoir si la tortue est toujours dans la fenêtre (on l'appellera `ecran?`)

8 Soit la procédure suivante :

```
(define (spi_carre x inc)
  (cond ((ecran?) (av x) (td 90) (spi_carre (+ x inc) inc))))
```

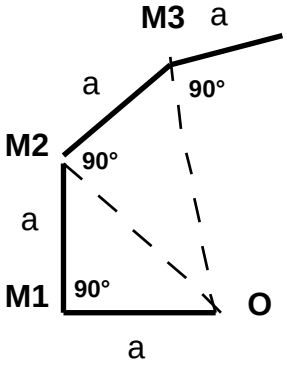
que fait cette procédure ? comment fonctionne-t-elle ? La tester en évaluant `(spi_carre 10 2)`

Ecrire la ou les procédures permettant de dessiner en surimpression plusieures spirales carré mais décalée d'un certain angle ß (par exemple ß = 20°)

9 En vous inspirant de `spi_carre` et de `cercle`, ecrire la procédure qui permet de dessiner une vraie spirale en tournant toujours du même angle ß que l'on passera en argument de la procédure.

10 La spirale de Pythagore.

soit le dessin construit de la manière suivante :



On part du centre O et on trace un trait de longueur a ; on a le point M_1

Puis à partir de M_1 et perpendiculairement à OM_1, on trace de nouveau un trait de longueur a; on a le point M_2

....à partir de M_2 et perpendiculairement à OM_2, on trace de nouveau un trait de longueur a; on a le point M_3 etc...

On se propose d'écrire la procédure `spi_pythagore` qui permet de réaliser ce dessin (on s'arrête quand la tortue sort de l'écran !)

a) Soit un point M de coordonnée (x,y), comment connaitre le cap de la tortue lorsque, étant au centre, elle se dirige vers le point M ? Ecrire la fonction correspondante `cap_point` qui prend comme argument la liste des coordonnées.

Exemples : `(cap_point '(100 100))` doit retourner `45` `(cap_point '(-100 -100)) --> 225`

Rem : le langage SCHEME possède une fonction identique à celle que vous avez sur vos calculettes - qui s'appelle généralement INV TAN ou TAN^{-1} sur les calculettes ! - qui permet à partir d'une valeur x de connaître l'angle ß tel que tan ß = x. En Scheme, cette fonction s'appelle ATAN et prend deux arguments y et x tels que tan ß = $\frac{y}{x}$

(le résultat est donné en radian, mesuré par rapport à l'axe Ox et signé !)

Exemples : `(atan 100 100)` = 0.7853981633974483 (soit 45° exprimé en radian !)

`(atan -100 -100)` = -2.356194490192345 (soit -135° exprimé en radian!)

b) Soit un point M de coordonnée (x,y), comment connaître le cap de la tortue lorsque, étant au point M, elle se dirige vers le centre O ? Ecrire la fonction correspondante `cap_centre` qui prend comme argument la liste des coordonnées.
Exemple : `(cap_centre '(100 100))` doit retourner 225 (le résultat en degré !)
`(cap_centre '(-100 -100))` doit retourner 45
Rem : il est recommandé de réfléchir au changement de repère que l'on fait quand on passe du problème vu du centre (question a) au même problème vu du point M(question b) !

c) en utilisant la fonction `cap_centre`, écrire la procédure `spi_pythagore` qui permet de dessiner la spirale de Pythagore.

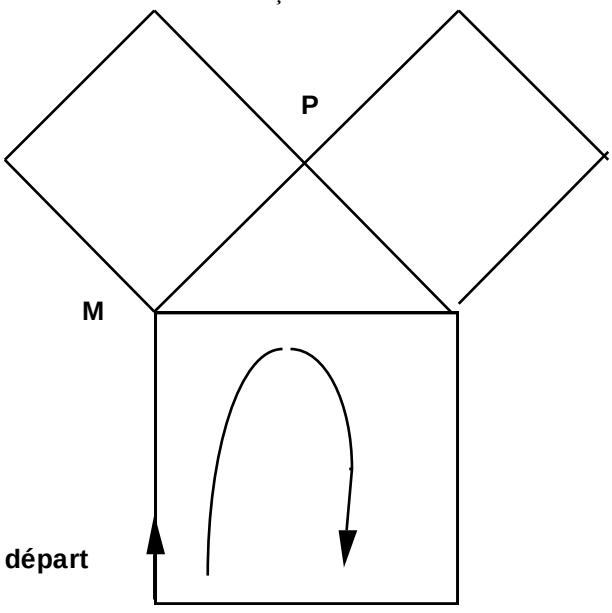
d) en supposant que la longueur a vale 1 (l'unité), que valent les différentes longueurs OM_1, OM_2, OM_3, etc ?
Voyez vous pourquoi cette spirale s'appele "la spirale de Pythagore" ?
Modifier la procédure `spi_pythagore` en faisant afficher $(\frac{OM_i}{a})^2$ à chaque étape de la construction du dessin.

Les procédures qui définissent le dessin des spirales sont récursives (plus exactement récursives terminales pour les exemples ci-dessus!). Nous allons appliquer cette démarche pour des dessins un peu plus compliqués qui vont vous faire découvrir un monde merveilleux, celui des courbes fractales
(référence "Chaos and Fractals : new frontier of Science" Springer-Verlag de Peitgen, Jürgens and Saupe)

Des arbres ....

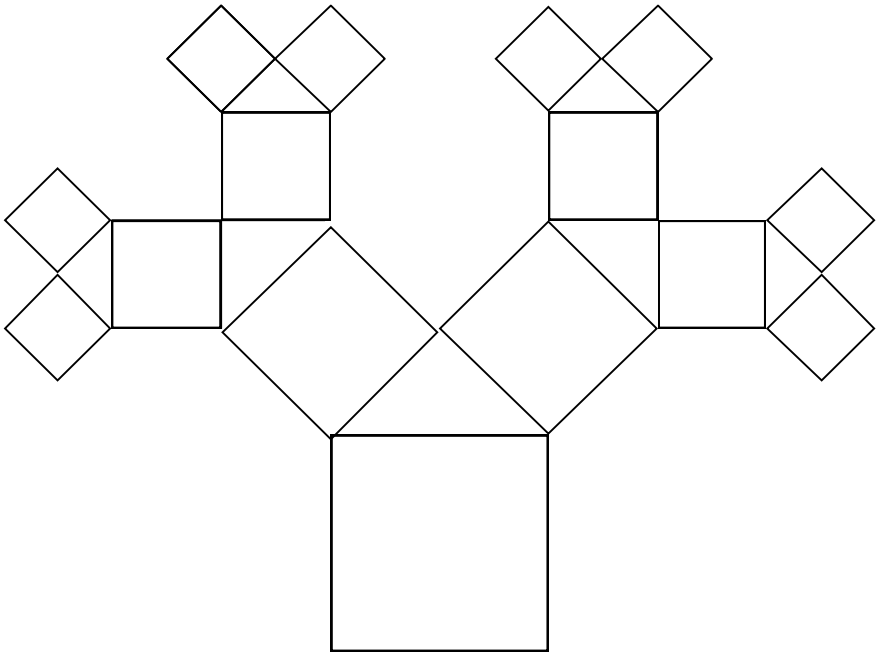
12 L'arbre de Pythagore (encore lui !)

Soit le dessin défini de la façon suivante :



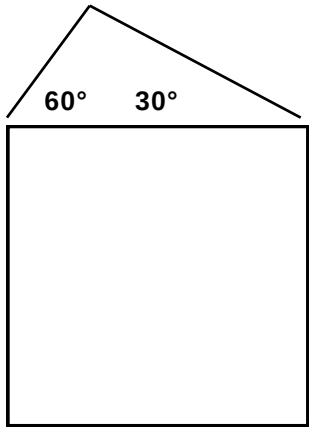
On dessine un carre de côté a, puis on va se placer en M, on tourne à gauche de 45°, on relance récursivement en modifiant l'argument a de telle façon que les carrés successifs se disposent comme ci-contre, puis on vient se placer en P et on relance encore récursivement....
et on s'arrête dès que a est trop petit (a < 5 par exemple)

Ecrire la procédure `arbre_pythagore`.



Voilà l'arbre que vous devez obtenir après 4 appels récursifs !

13 Modifier la procédure de construction de l'arbre de Pythagore de telle façon que les deux branches soient dissymétriques pour avoir par exemple le dessin ci-dessous:



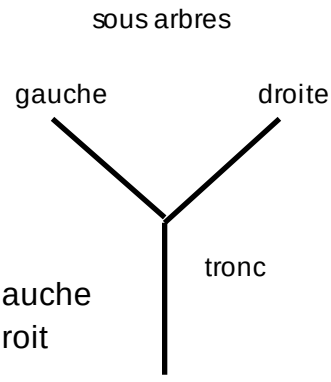
Dans ce nouvel arbre, y a t il deux carrés identiques (deux carrés qui ont le même côté) ? Essayez de <u>démontrer</u> votre réponse !

14 Soit la définition récursive suivante d'un arbre :

Un arbre est :

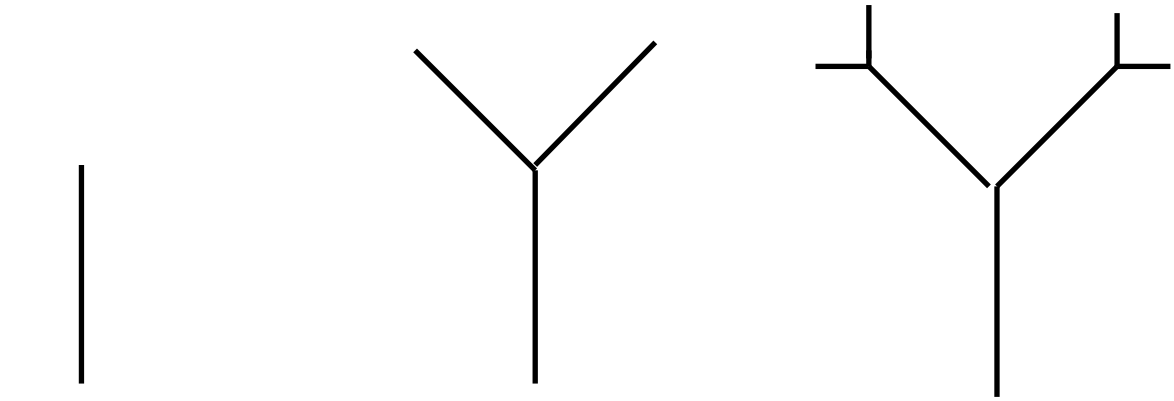
soit un arbre vide

soit
- un tronc
- un sous arbre gauche
- un sous arbre droit



la définition étant récursive, les sous arbres sont eux-mêmes des arbres, etc..

On peut associer, à chaque arbre, le niveau maximum de profondeur de la récursion ainsi :



niveau 1 niveau 2 niveau 3

Les sous arbres sont décalés de 45° par rapport au tronc et le longueur des troncs des sous arbres est diminuée d'un tiers à chaque niveau.

Ecrire la procédure récursive `(arbre niveau tronc)` , niveau étant le niveau maximum ou descendra la récursion, tronc étant la longueur du tronc initial.

Comme dans toute procédure récursive, il faut réfléchir :

- à la condition d'arrêt. et ce que l'on fait alors
- à la définition du problème au niveau i en fonction du même problème au niveau (i-1)

Une fois écrite cette procédure, écrire une procédure une procédure `(prepa-arbre)` qui nettoie l'écran et positionne la tortue de telle façon à exécuter le dessin de l'arbre sur toute la surface de la fenêtre de dessin.

Pour pouvoir utiliser la fonction sleep décrite ci-dessous, il faut se placer en Langage « Assez gros Scheme »
Pour sélectionner ce niveau de langage : Menu Langage =>Langage Professionnel => PLT => Assez gros Scheme

15 En utilisant la primitive DrScheme `(sleep N)` qui met le système en attente pendant Nsecondes (par exemple `(sleep 3)` met le système en attente pendant 3 s), écrire une procédure `(anim-arbre n)` qui fasse apparaître successivement les arbres au niveau 1, 2, jusqu'à n. en laissant 3 s entre les apparitions de chaque niveau d'arbre

16 Modifier la procédure `arbre` de telle façon que les sous arbres soient écartés d'un angle ß que l'on passera en argument.

17 Modifier de nouveau la procédure `arbre` pour que les deux sous arbres ne soient pas symétriques par rapport au tronc : vous choisirez le nouvel argument qui vous convient le mieux pour provoquer cette dissymétrie.