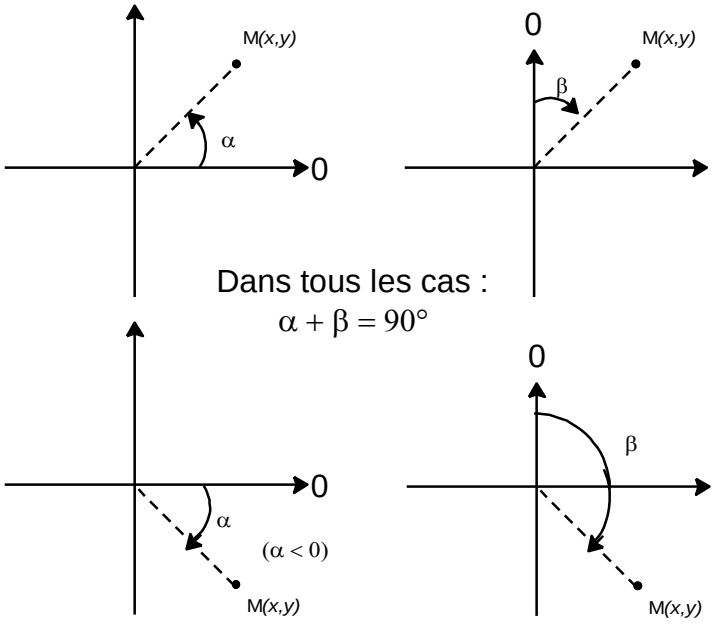
; CORRECTION *TP tortue Partie I*

```
; des polygones
;*****************************
(define (carre)
  (repeat 4 (av 100) (td 90)))

(define (carre_gene x)
  (repeat 4 (av x) (td 90)))

(define (triangle x)
  (repeat 3 (av x) (td 120)))

(define (hexagone x)
  (repeat 6 (av x) (td 60)))

(define (polygone x n)
  (repeat n (av x) (td (/ 360 n))))

(define (cercle)
  (repeat 36 (av 10) (td 10)))

(define (cercle_ge r)
  (repeat 36 (av (* r 10 (/ 3.14159265 180))) (td 10)))

(define (carre_tournant  x n)
  (repeat n (carre_gene x) (td (/ 360 n))))

; des spirales
;*****************************
(define (ecran?)
  (let ((x (car (pos))) (y (cadr (pos))))
  (and (<= (abs x) 320) (<= (abs y) 177))))

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

(define (spirale x inc angle)
  (cond ((ecran?) (av x) (td angle) (spirale (+ x inc) inc angle))))
;essayer (spirale 1 0.25 30)
```

```
; la spirale de Pythagore
(define (cap_point pos)
   (- 90 (radiantodegre (atan (cadr pos) (car pos)))))

(define (cap_centre pos)
   (- 90 (radiantodegre (atan (-(cadr pos)) (-(car pos))))))
```



Remarque : la fonction atan permet d'accéder à l'angle α Or dans la gestion de la tortue, le cap est associé à l'angle β

Il faut donc établir la relation entre les deux angles et , comme le montre les schéma ci-contre, dans toutes les situations possibles, on a la relation :

$$\alpha + \beta = 90°$$

```
(define (spi_pythagore a)
  (define (travail)
    (cond ((ecran?)
           (fcap (cap_centre (pos)))(tg 90)(av a)(travail))))
  (centre)
  (fcap 270) (av a) (travail))

(define (spi_pythagore2 a)
  (define (travail)
    (cond ((ecran?)
           ; un petit affichage de OMi/a au carre
           (newline)
           (write (/ (+ (* (car (pos)) (car (pos))) (* (cadr (pos))
(cadr (pos))))
                     (* a a)))
           (fcap (cap_centre (pos)))(tg 90)(av a)(travail))))
  (centre)
  (fcap 270) (av a) (travail))
; essayer avec a = 50
```

```
;*******************************
;                                    Arbre Pythagore
;*******************************
;préparation pour dessiner un arbre_pythagore

(define (prepa-arbre-pi)
  (begin (clear-graphics) (lc) (fxy -80 -140)(fcap 0) (bc)))

(define (arbre-pi x)
  (cond ((>= x 2)
         (carre_gene x) (av x) (tg 45)
         ; cet arbre sera parfaitement symetrique  (45 - 45)
         (let ((y (/ x (sqrt 2))))

           (arbre-pi y )
           (td 90) (av y)
           (arbre-pi y )
           (re y)(tg 45))
         (re x))))
; essayer avec x = 80

(define (arbre-pi2 x)
  (cond ((>= x 2)
         (carre_gene x) (av x) (tg 60)
         ; cet arbre sera  dissymetrique  (30 - 60)
         (arbre-pi2 (/ x 2 ))
         (td 90) (av (/ x 2))
         (arbre-pi2 (/ (* x (sqrt 3 )) 2))
         (re (/ x 2)) (tg 30)
         (re x))))
; essayer avec x = 70

; pb : y a t il deux carres identiques (même coté ?) dans tous les
carres dessinés ?
```

```
;*****************************
; des arbres
;*****************************

(define (prepa-arbre)
  (begin (clear-graphics) (lc) (fxy 0 -177)(fcap 0) (bc)))

(define (arbre niveau tronc)
    (cond ((> niveau 0)
                (av tronc)
                (tg 45)
                (arbre (1- niveau) (* tronc 0.667))
                (td 90)
                (arbre (1- niveau) (* tronc 0.667))
                (tg 45)
                (re tronc))))

(define (anim-arbre n) ; ca commence à 1
  (let ((i 1) (tronc 200))
  (repeat n
          (prepa-arbre)
          (msg 50 -100 (string-append "Arbre au niveau " (number->string i)))
          (arbre i tronc)
           (set! i (1+ i))
           (if (> tronc 125)(set! tronc (- tronc 15)))
           (delay-time 120)
           )))

;*******************************
;                                        VON KOCH
;*******************************
; pour bien positionner la tortue
(define (prepa-von-koch)
  (begin (clear-graphics) (lc) (fxy -300 -77)(fcap 90) (bc)))

; la procedure de von-koch
(define (von-koch prof cote)
    (if (zero? prof)
        (av cote)
        (begin (let ((cote (/ cote 3))
                     (prof (1- prof)))
                 (von-koch prof cote)
                 (tg 60)
                 (von-koch prof cote)
                 (td 120)
                 (von-koch prof cote)
                 (tg 60)
                 (von-koch prof cote)))))

; pour que ce soit bien positionner faire
; (von-koch prof 600)

; pour faire une animation avec les courbes de vonkoch qui se dessinent
; du niveau de profondeur 0 à n-1

(define (anim-von-koch n) ; ca commence à 0
  (let ((i 0))
  (repeat n
          (prepa-von-koch)
          (msg -70 150 (string-append "La courbe de Von Koch au niveau "
(number->string i)))
          (von-koch i 600)
           (set! i (1+ i))
```

```
           (delay-time 120)
           )))
;*******************************
;                                        FLOCON
;*******************************
;préparation pour dessiner un flocon

(define (prepa-flocon)
  (begin (clear-graphics) (lc) (fxy -150 85)(fcap 90) (bc)))

(define (flocon n)
  (repeat 3 (von-koch n 300) (td 120)))

; pour faire une animation de flocon de différents niveaux

(define (anim-flocon n) ; ca commence à 0
  (let ((i 0))
  (repeat n
          (prepa-flocon)
          (msg -50 50 (string-append "Flocon au niveau " (number->string i)))
          (flocon i)
           (set! i (1+ i))
           (delay-time 120)
           )))
```