

# La Tortue

## Algo & Prog avec R

---

A. Malapert, B. Martin, M. Pelleau, et J.-P. Roy

10 septembre 2021

Université Côte d'Azur, CNRS, I3S, France  
`firstname.lastname@univ-cotedazur.fr`

# Les deux types de graphisme dans le plan I

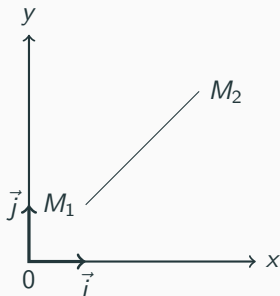
Il y a deux types de graphisme 2D, mathématiquement parlant :

## Le graphisme CARTESIEN (global)

Le plan est rapporté à un repère orthonormé direct  $(0, \vec{i}, \vec{j})$ .

### Une seule opération essentielle

Tracer un segment du point  $M_1(x_1, y_1)$  au point  $M_2(x_2, y_2)$ .



# Les deux types de graphisme dans le plan II

## Le graphisme POLAIRE (local)

Aucune notion de coordonnées.

### Deux opérations essentielles

- ▶ **Tourner** à droite ou à gauche sur place d'un angle  $a$ .
- ▶ **Avancer** dans la direction courante d'une distance  $d$ .



L'animal traceur porte un repère mobile orthonormé avec une notion de droite et de gauche.

la tortue va tourner à gauche

- ▶ Opérateurs de translation et de rotation plane, qui engendrent le groupe des déplacements. La tortue se déplace dans le plan !
- ▶ Graphisme moins mathématique, plus intuitif. Inutile de calculer les coordonnées des points ...
- ▶ Une trajectoire qui semble lisse sera en fait un polygone !

# Le module TurtleGraphics de R

Le **graphisme de la tortue** a été inventé au Laboratoire d'Intelligence Artificielle du MIT vers 1968 avec le langage LOGO.

- ▶ Il est disponible dans quasiment tous les langages de programmation qui offrent des facilités graphiques.
- ▶ Et en particulier en R avec le module **TurtleGraphics**.

## Installation et chargement

Ce module n'est pas livré avec la distribution R standard.

```
install.packages("TurtleGraphics")
```

Il faut en importer les noms pour pouvoir les utiliser.

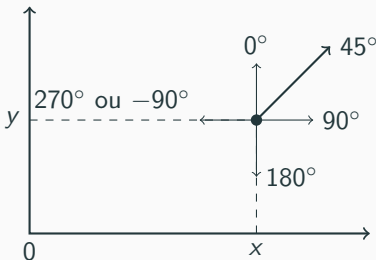
```
library(TurtleGraphics)
```

# Graphisme cartésien

C'est celui des matheux dans la mesure où il faut calculer les coordonnées des points à relier.

## Représentation de la tortue

- ▶ une flèche qui indique son **cap** en degrés ;
- ▶ une **position** : une abscisse et une ordonnée ;
- ▶ un **crayon** (*pen*) qui peut être baissé (*down*) ou levé (*up*). Si le crayon est baissé, la tortue laisse une trace en se déplaçant. On peut choisir la couleur du crayon ainsi que le type et l'épaisseur de la ligne.



# État et opération de la tortue

Une tortue a donc un ETAT représenté mathématiquement par trois données : position ; cap ; crayon.

## Position

```
turtle_getpos()  
turtle_setpos(x,y)
```

## Cap

```
turtle_getangle()  
turtle_setangle(a)
```

## Crayon (état)

```
turtle_down()  
turtle_up()
```

## Crayon (style)

```
turtle_param(col, lwd ,lty)  
turtle_col(col)  
turtle_lwd(lwd)  
turtle_lty(lty)
```

## Tracer un segment

```
turtle_goto(x, y)
```

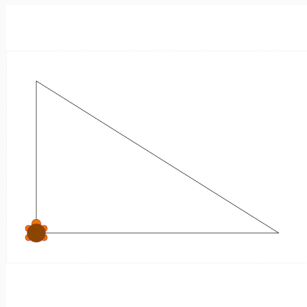
# Dessin d'un triangle rectangle

## Agir sur le bac à sable (canevas)

```
turtle_init(width = 100, height = 100,  
            mode = c("error", "clip", "cycle"))  
turtle_reset()
```

```
TriRect <- function(a, b, c = 10) {  
  turtle_up()  
  turtle_goto(c, c);  
  turtle_down()  
  turtle_goto(a + c, c)  
  turtle_goto(c, b + c)  
  turtle_goto(c, c)  
}
```

```
turtle_init(width = 100, height = 70)  
turtle_do(TriRect(80, 50))
```

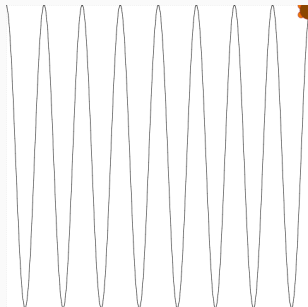


**ATTENTION**, les points du canevas ont des coordonnées positives.  
L'origine du repère est donc en bas à gauche.

# Tracé de la courbe du cosinus

```
TraceFunction <- function(f, a, b, n)
{
  turtle_up()
  turtle_goto(a, f(a))
  turtle_down()
  for(x in seq(a,b, length.out=n)) {
    turtle_goto(x, f(x))
  }
}
```

```
b <- 50
n <- 1000
turtle_init(width= b, height= b)
f <- function(x) b * (cos(x)+1) / 2
turtle_do(TraceFunction(f, 0, b, n))
```



Comme `turtle_goto` ou `TraceFunction`, la plupart des fonctions de dessin n'ont pas de résultat, seulement des effets.



# Courbes en coordonnées paramétriques

## Cinématique (étude du mouvement)

La cinématique s'intéresse à la trajectoire d'un corps dont les coordonnées  $(x, y)$  sont fonction d'un paramètre  $t$ . Autrement dit :

$$x = x(t) \text{ et } y = y(t)$$

Ces courbes englobent les courbes  $y = f(x)$  mais sont plus générales !

### Le segment

Le segment AB joignant le point  $A(x_A, y_A)$  au point  $B(x_B, y_B)$  est la trajectoire d'un mobile  $M$  paramétrée par  $t \in [0, 1]$  :

$$x(t) = tx_A + (1 - t)x_B$$

$$y(t) = ty_A + (1 - t)y_B$$

De manière vectorielle :  $\overrightarrow{MB} = t\overrightarrow{AB}$



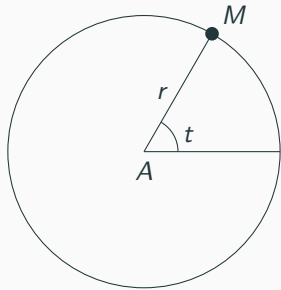
# Animation de la tortue parcourant un cercle

## Le cercle

le cercle de centre  $A(x, y)$  et de rayon  $r$  n'est autre que la trajectoire d'un mobile  $M$  dont les coordonnées sont paramétrés par  $t \in [0, 2\pi]$  :

$$x(t) = x + r \cos(t)$$

$$y(t) = y + r \sin(t)$$

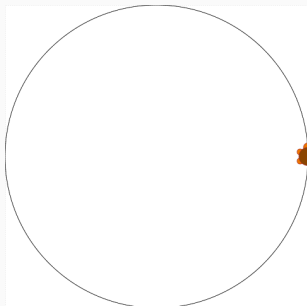


```
Cercle <- function(r, n) {  
  turtle_up()  
  turtle_goto(2*r, r)  
  turtle_down()  
  for(x in seq(0,2*pi, length.out=n)) {  
    turtle_goto(r + r*cos(x), r + r*sin(x))  
  }  
}
```

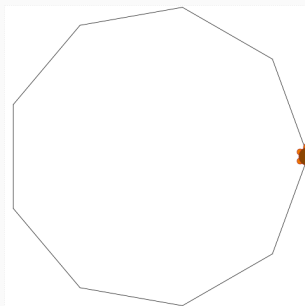
# Le caractère continu du mouvement est une illusion d'optique

En fait, il est **discrétisé**. Le paramètre  $t$  avance chaque fois de  $\frac{2\pi}{n}$ .

```
turtle_init()  
turtle_do(  
    Cercle(r = 50, n = 1000)  
)
```



```
turtle_init()  
Cercle(r = 50, n = 10)
```



Le choix de  $n$  peut être empirique, guidé par l'esthétique de la simulation.

Questions?

Retrouvez ce cours sur le site web

[`www.i3s.unice.fr/~malapert/R`](http://www.i3s.unice.fr/~malapert/R)