PROCESSING.PY: INTERACCIÓN BÁSICA E CONDICIONAL 2
Rotar, trasladar, escalar

translate(x,y): función que traslada o centro de debuxo ás coordenadas (x,y)

Se incluímos variables nas coordenadas podemos lograr unha modificación continua do debuxo.

```
def setup():
    size(120,120)

def draw():
    translate(mouseX, mouseY) #centra o debuxo nas coordenadas (mouseX,mouseY)
    rect(0,0,30,30)
    translate(35,10) #centra o debuxo nas coordenadas (mouseX+35,mouseY+10)
    rect(0,0,15,15)
```

Exercicio: averigua como facer para borrar a pantalla e desapareza o rastro.

rotate(ángulo): rota o sistema de coordenadas un número de radiáns (ou grados).

```
def setup():
    size(420,420)
def draw():
    rotate(mouseX/100.0) #IMPORTANTE: dividimos por 100.0, non por 100
    rect(40,30,160,20)
```

Como facer para rotar dende o centro da figura?

```
def setup():
    size(420,420)
def draw():
    rotate(mouseX/100.0) #IMPORTANTE: dividimos por 100.0, non por 100
    rect(-80,-10,160,20) #ao modificar a orixe de coordenadas modifica o xiro
```

Translación+rotación e viceversa. Fai os seguintes programas intentanto comprender o que fan.

```
angle = 0.0

def setup():
    size(420,420)
def draw():
    global angle
    translate(mouseX,mouseY)
    rotate(angle)
    rect(-15,-15,30,30)
    angle +=0.1
```

```
angle = 0.0

def setup():
    size(420,420)
def draw():
    global angle
    rotate(angle)
    translate(mouseX,mouseY)
    rect(-15,-15,30,30)
    angle +=0.1
```

Exercicio para nota na aula virtual: o seguinte programa é un brazo articulado. Identifica como se realiza o movemento de cada unha das partes. Serías capaz de debuxalo/animalo ao revés, coma nun espello?

```
angle = 0.0
angleDirection = 1
speed = 0.01 #determina a velocidade de xiro

def setup():
    size(120,120)

def draw():
    global angle,angleDirection
    background(204)
    translate(20,25) #posición inicial
    rotate(angle)
    strokeWeight(12)
    line(0,0,40,0) #liña grosa
    translate(40,0) #mover á seguinte posición, suma 40 á posición anterior
    rotate(angle*2.0) #rota o doble de rápido que a liña anterior
    strokeWeight(6)
    line(0,0,30,0) #liña do medio
    translate(30,0) #mover á seguinte posición, suma 30 á posición anterior
    rotate(angle*2.5) #rota máis rápido
    strokeWeight(3)
    line(0,0,20,0) #liñas máis fina
    angle += speed*angleDirection #incrementa o águlo
    if angle>QUARTER_PI or angle<0: #cando o ángulo é maior que 90º ou menor que 0ª
        angleDirection = -angleDirection #modifica sentido de xiro
```

scale(x): escala o debuxo nunha porcentaxe x. Exemplo: scale(1.5) aumentaría o 150%.

```
def setup():
    size(420,420)
def draw():
    translate(mouseX,mouseY)
    scale(mouseX/60.0)
    rect(-15,-15,30,30)
```

```
def setup():
    size(420,420)
def draw():
    translate(mouseX,mouseY)
    scalar = mouseX/60.0
    scale(scalar)
    if scalar > 0.0:
        strokeWeight(1.0/scalar)
    else: #isto evitar dividir entre 0
        strokeWeight(0.0)
    rect(-15,-15,30,30)
```

Como illar os efectos de translación, rotación...a un único obxecto?

pushMatrix(): garda unha copia das coordenadas do sistema
popMatrix(): restaura as coordenadas gardadas anteriormente con pushMatrix()
Estas 2 funcións van sempre en parella, primeiro unha e logo outra.
No seguinte programa o rectángulo pequeno non desaparece nunca. Explica como sucede iso.

```
def setup():
    size(420,420)
def draw():
    pushMatrix()
    translate(mouseX,mouseY)
    rect(0,0,30,30)
    popMatrix()
    translate(35,10)
    rect(0,0,15,15)
```