IFT1015 Programmation 1

Dessins "tortue"

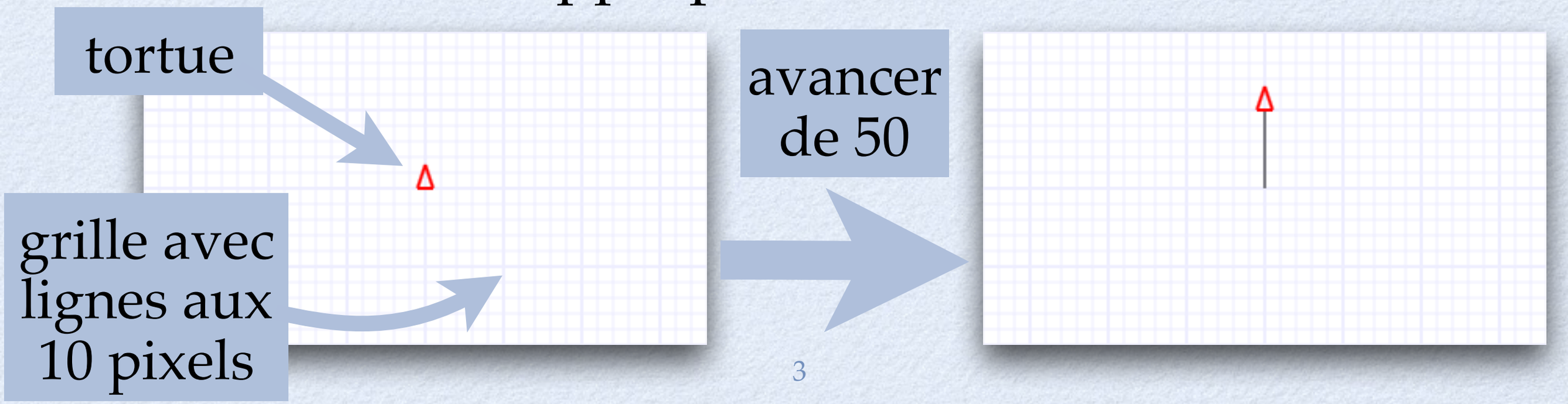
Marc Feeley

Université de Montréal

Dessins “tortue”

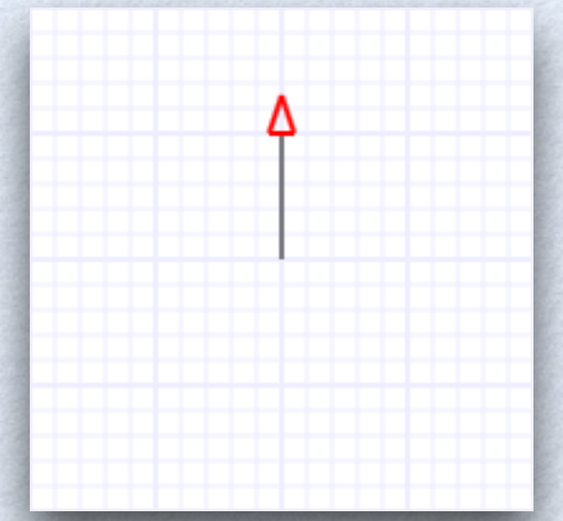
Dessins "tortue"

- codeBoot offre un environnement pour faire des dessins avec la métaphore de la **tortue**
- À l'aide de procédures prédéfinies, le programme peut faire **bouger une tortue** qui laisse un tracé sur son passage
- Un dessin complet s'obtient par une **séquence de mouvements** appropriés de la tortue

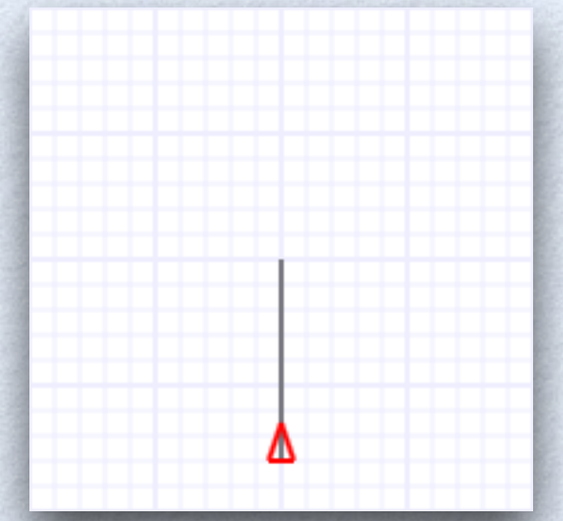


Mouvements prédéfinis

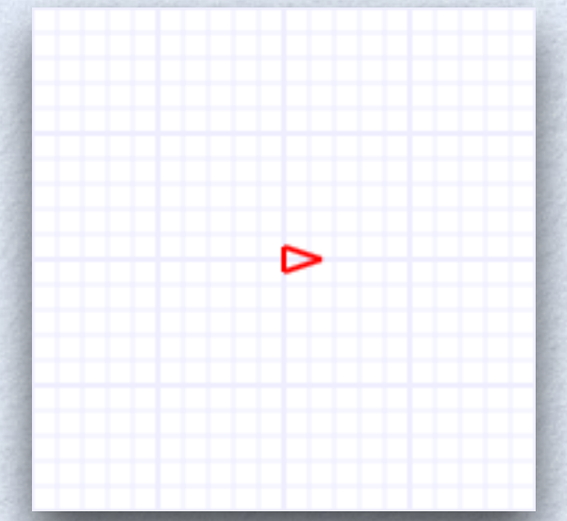
- Initialement la tortue est au centre de la fenêtre de dessin et pointe vers le nord
- **`fd(`***d*`)`** : avancer d'une distance *d* pixels ("forward")
- **`bk(`***d*`)`** : reculer d'une distance *d* pixels ("backward")
- **`rt(`***a*`)`** : pivoter à droite d'un angle *a* degrés ("right")
- **`lt(`***a*`)`** : pivoter à gauche d'un angle *a* degrés ("left")



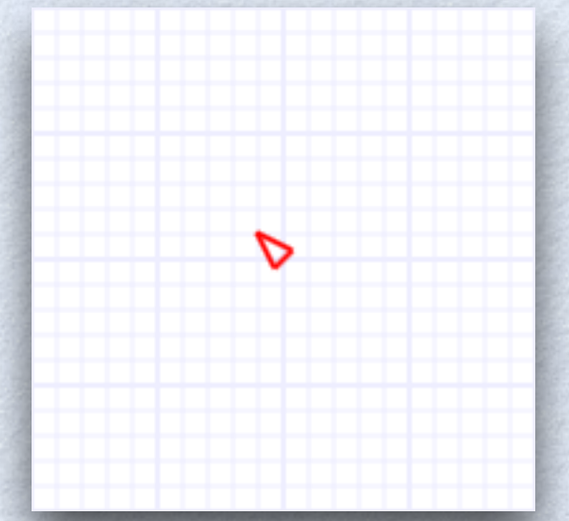
`fd(50)`



`bk(80)`



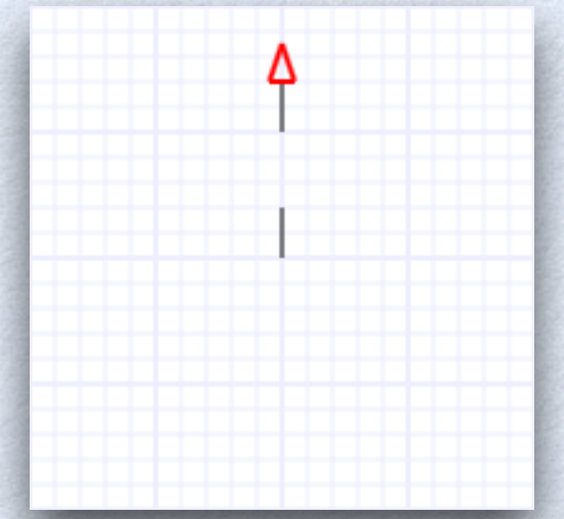
`rt(90)`



`lt(45)`

Pen up/down

- La tortue peut se déplacer sans laisser de tracé
- C'est utile pour les dessins déconnectés
- **`pu()`** : cesser de laisser un tracé ("pen up")
- **`pd()`** : commencer à laisser un tracé ("pen down")

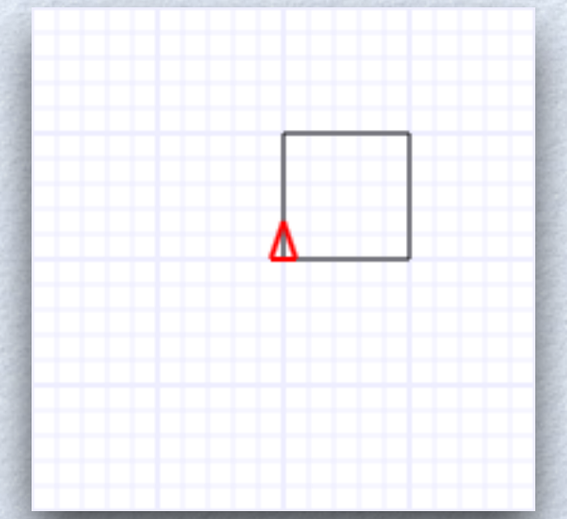


```
fd(20); pu(); fd(30); pd(); fd(20);
```

Dessiner un carré

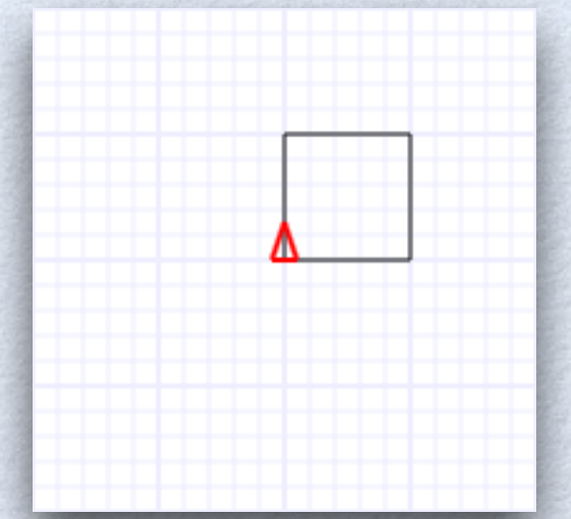
- Un carré est composé de 4 traits identiques à 90° :

```
// dessiner un carré de 50 pixels de largeur

fd(50); rt(90); // côté 1
fd(50); rt(90); // côté 2
fd(50); rt(90); // côté 3
fd(50); rt(90); // côté 4
```



- Avec boucle et abstraction procédurale :

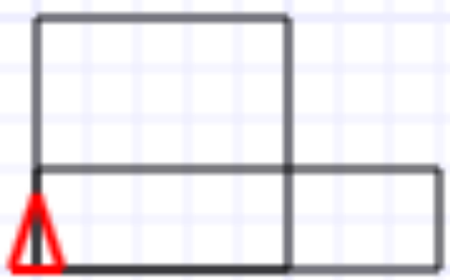
```
var carre = function (largeur) {
    for (var i=1; i<=4; i++) {
        fd(largeur);
        rt(90);
    }
};

carre(50); // dessiner un carré
```



Dessiner un carré

- Avec généralisation (rectangle) :

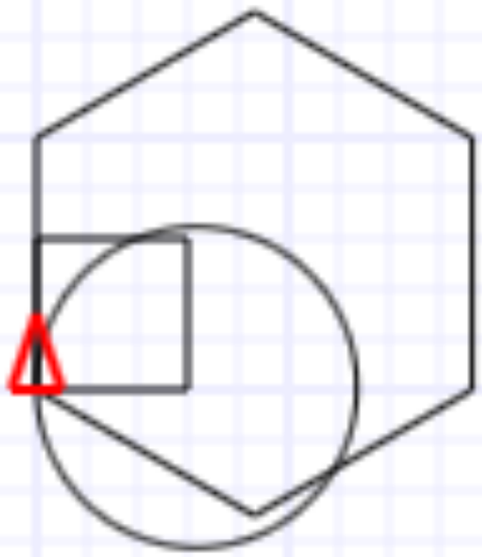
```
var rectangle = function (largeur, hauteur) {
    for (var i=1; i<=2; i++) {
        fd(hauteur);
        rt(90);
        fd(largeur);
        rt(90);
    }
};

var carre = function (largeur) {
    rectangle(largeur, largeur);
};

rectangle(80, 20); // dessiner un rectangle

carre(50); // dessiner un carré
```



Dessiner un carré

- Avec généralisation (polygone régulier) :

```
var polygoneReg = function (cote, nbCotes) {
    for (var i=1; i<=nbCotes; i++) {
        fd(cote);
        rt(360/nbCotes);
    }
};

var carre = function (largeur) {
    polygoneReg(largeur, 4);
};

polygoneReg(50, 6);    // dessiner un hexagone

carre(30);             // dessiner un carré

polygoneReg(1, 200);   // dessiner un cercle
```



Dessiner un carré

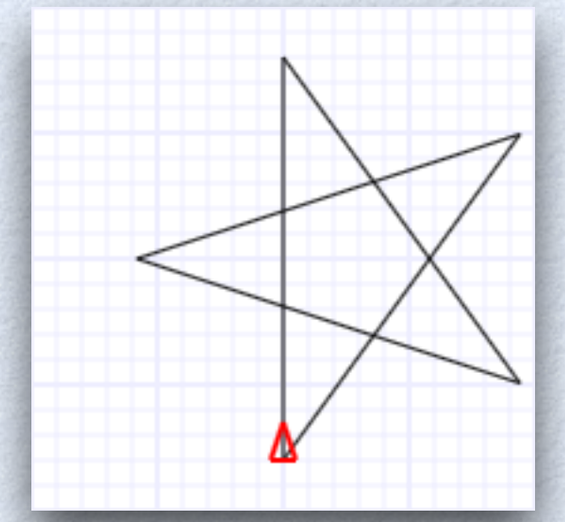
- Polygones centrés :

```
var polygoneReg = function (cote, nbCotes) {
    for (var i=1; i<=nbCotes; i++) {
        fd(cote);
        rt(360/nbCotes);
    }
};

var polygoneRegC = function (cote, nbCotes) {
    var r = cote / (2*Math.tan(Math.PI/nbCotes));
    pu(); lt(90); fd(r); rt(90); bk(cote/2); pd();
    polygoneReg(cote, nbCotes);
    pu(); lt(90); bk(r); rt(90); fd(cote/2); pd();
};

var carreC = function (largeur) {
    polygoneRegC(largeur, 4);
};

polygoneRegC(50, 6);    // dessiner un hexagone
carreC(30);             // dessiner un carré
polygoneRegC(1, 200);   // dessiner un cercle
```

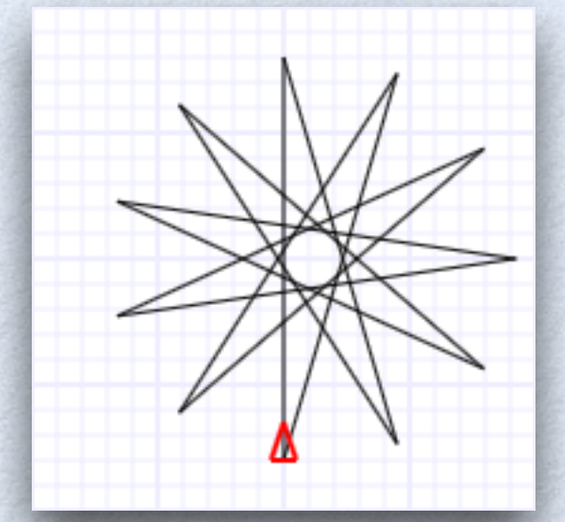
Dessiner une étoile

```
var polygoneRegPointu = function (cote, nbPointes) {
    for (var i=1; i<=nbPointes; i++) {
        fd(cote);
        rt(180-180/nbPointes);
    }
};

pu(); bk(80); pd();

polygoneRegPointu(160, 5); // dessiner une étoile
```

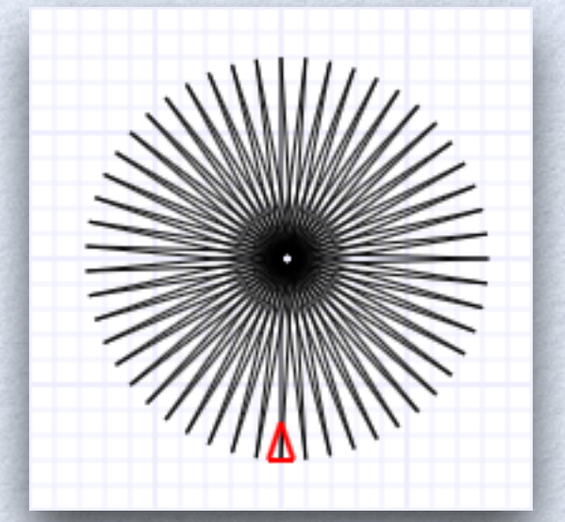
`polygoneRegPointu(160, 5);`



`polygoneRegPointu(160, 11);`

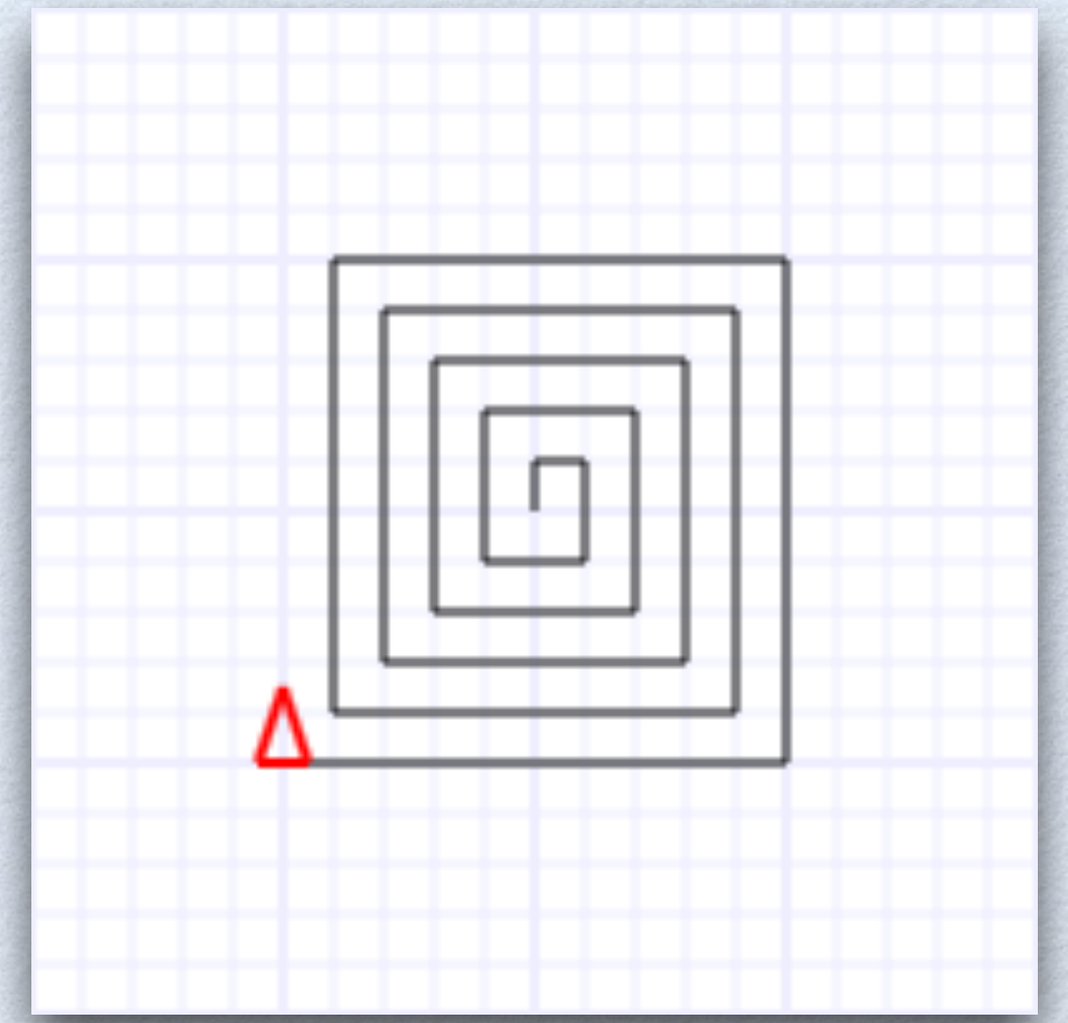


`polygoneRegPointu(160, 51);`



Dessiner une spirale

- Comment faire ce dessin?



```
// Fichier: spirale1.js

for (var i=1; i<=10; i++) {
    fd(i*10); rt(90);
    fd(i*10); rt(90);
}
```

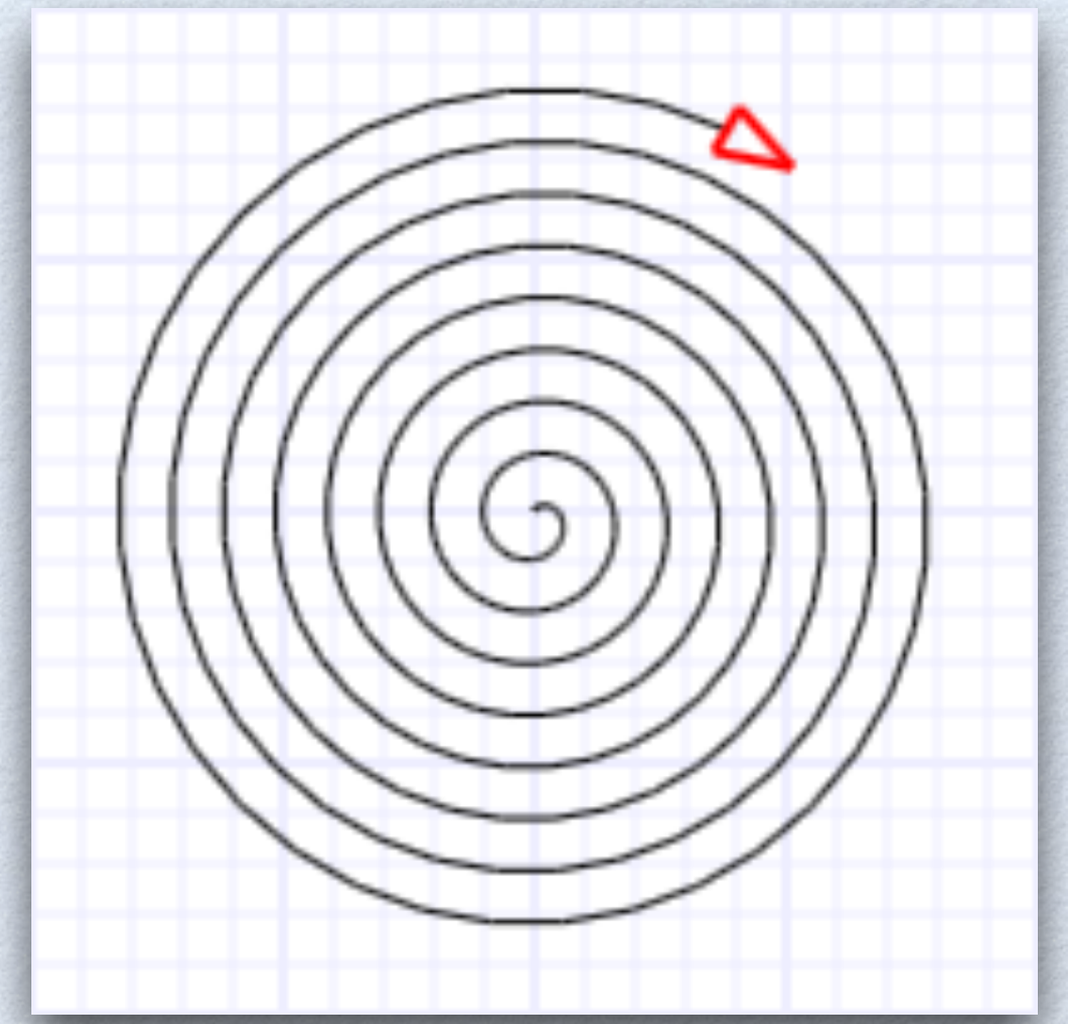
```
// Fichier: spirale2.js

for (var i=1; i<=10; i++) {
    for (var j=1; j<=2; j++) {
        fd(i*10); rt(90);
    }
}
```

```
// Fichier: spirale3.js

for (var i=1; i<=20; i++) {
    fd(((i+1)>>1)*10); rt(90);
}
```

Dessiner une spirale

- Comment faire ce dessin?

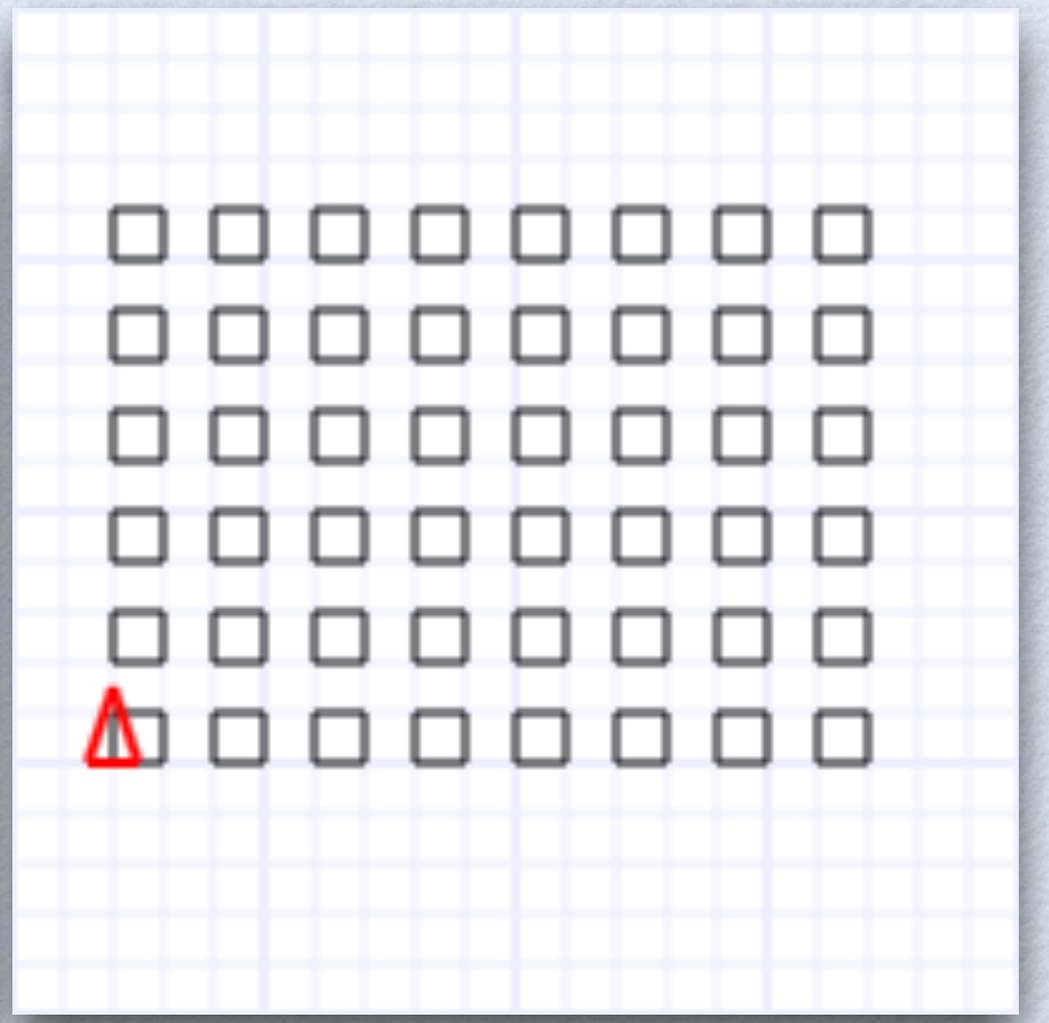
```
// Fichier: spirale4.js

for (var i=1; i<=300; i++) {
    fd(i*0.05); rt(10);
}
```



Dessiner une grille

- Comment faire ce dessin?

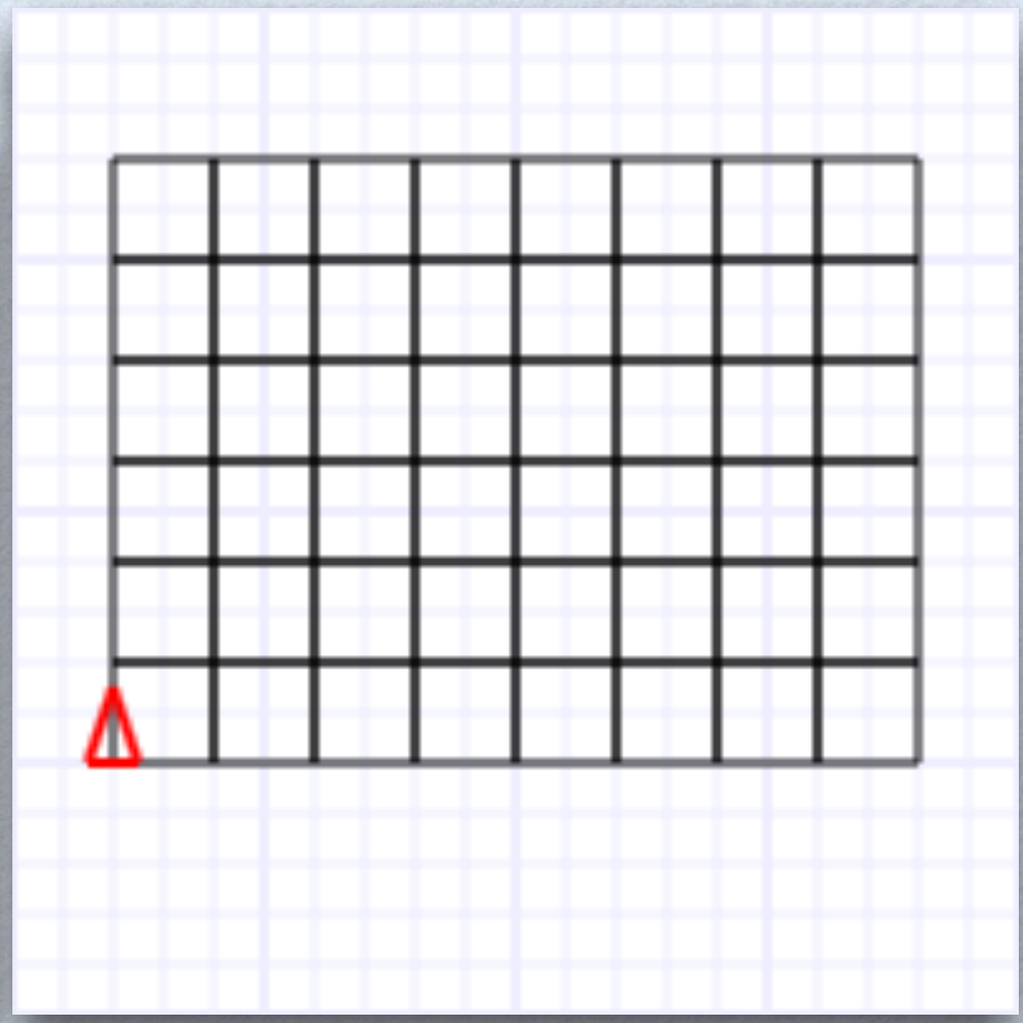
```
var carre = function (largeur) {
    for (var i=1; i<=4; i++) {
        fd(largeur);
        rt(90);
    }
};

var positionner = function (x, y) {
    pu(); rt(90); fd(x); lt(90); fd(y); pd();
};

var grille = function (nx, ny, pas, largeur) {
    for (var x=0; x<nx; x++) {
        for (var y=0; y<ny; y++) {
            positionner(x*pas, y*pas);
            carre(largeur);
            positionner(-x*pas, -y*pas);
        }
    }
};

positionner(-80, -50);

grille(8, 6, 20, 10); // dessiner la grille
```



Dessiner une grille

- Comment faire ce dessin?

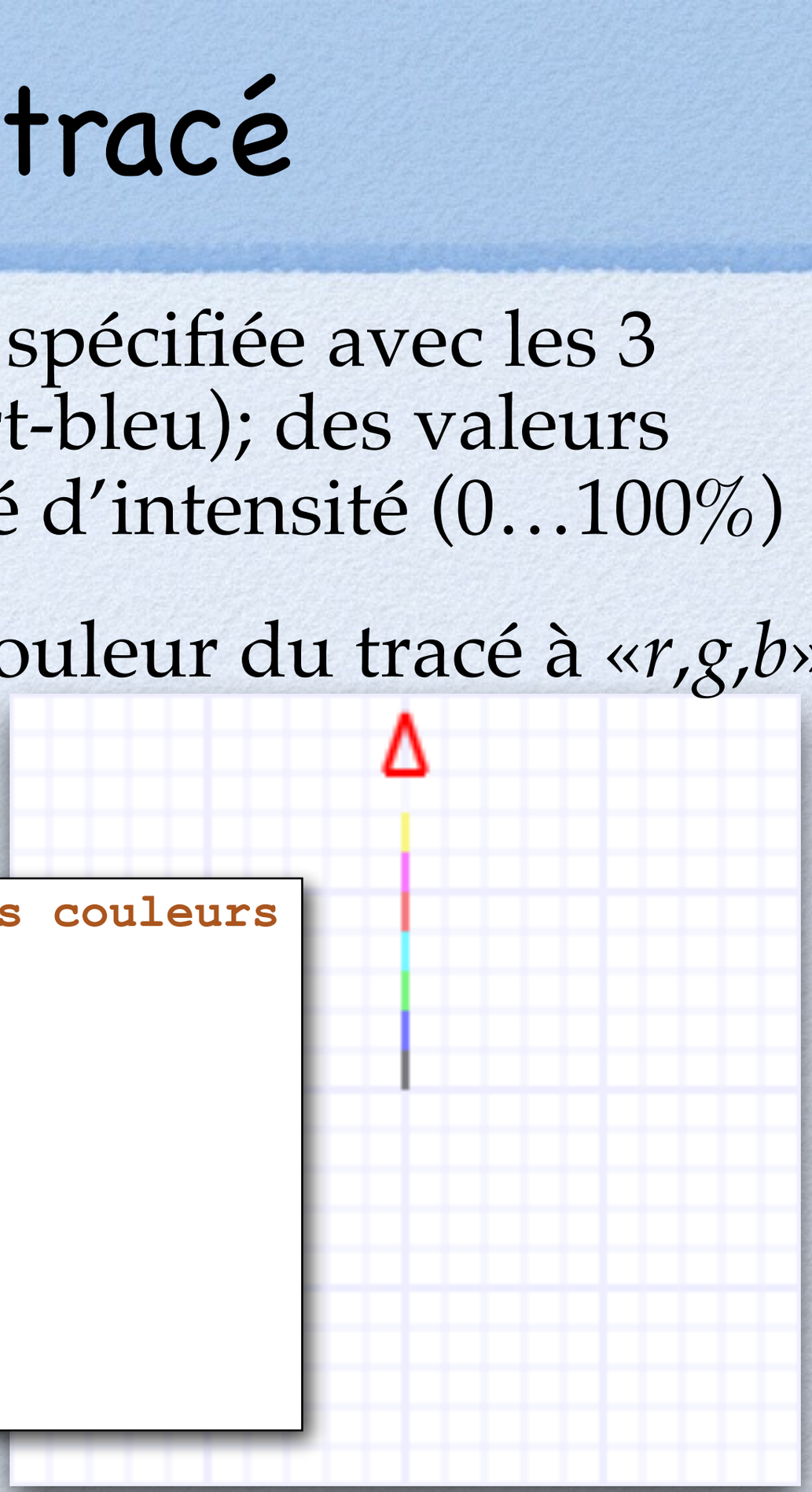
```
var carre = function (largeur) {
    for (var i=1; i<=4; i++) {
        fd(largeur);
        rt(90);
    }
};

var positionner = function (x, y) {
    pu(); rt(90); fd(x); lt(90); fd(y); pd();
};

var grille = function (nx, ny, pas, largeur) {
    for (var x=0; x<nx; x++) {
        for (var y=0; y<ny; y++) {
            positionner(x*pas, y*pas);
            carre(largeur);
            positionner(-x*pas, -y*pas);
        }
    }
};

positionner(-80, -50);

grille(8, 6, 20, 20); // dessiner la grille
```



Couleur du tracé

- La couleur du tracé peut être spécifiée avec les 3 composantes RGB (rouge-vert-bleu); des valeurs entre 0 et 1 indiquant le degré d'intensité (0…100%)
- **`setpc(`***r*`,`*g*`,`*b*`)`** : changer la couleur du tracé à «*r*,*g*,*b*» (“set pen color”)

```
// dessiner des traits de différentes couleurs

setpc(0,0,0); fd(10); // trait noir
setpc(0,0,1); fd(10); // trait bleu
setpc(0,1,0); fd(10); // trait vert
setpc(0,1,1); fd(10); // trait cyan
setpc(1,0,0); fd(10); // trait rouge
setpc(1,0,1); fd(10); // trait mauve
setpc(1,1,0); fd(10); // trait jaune
setpc(1,1,1); fd(10); // trait blanc
```



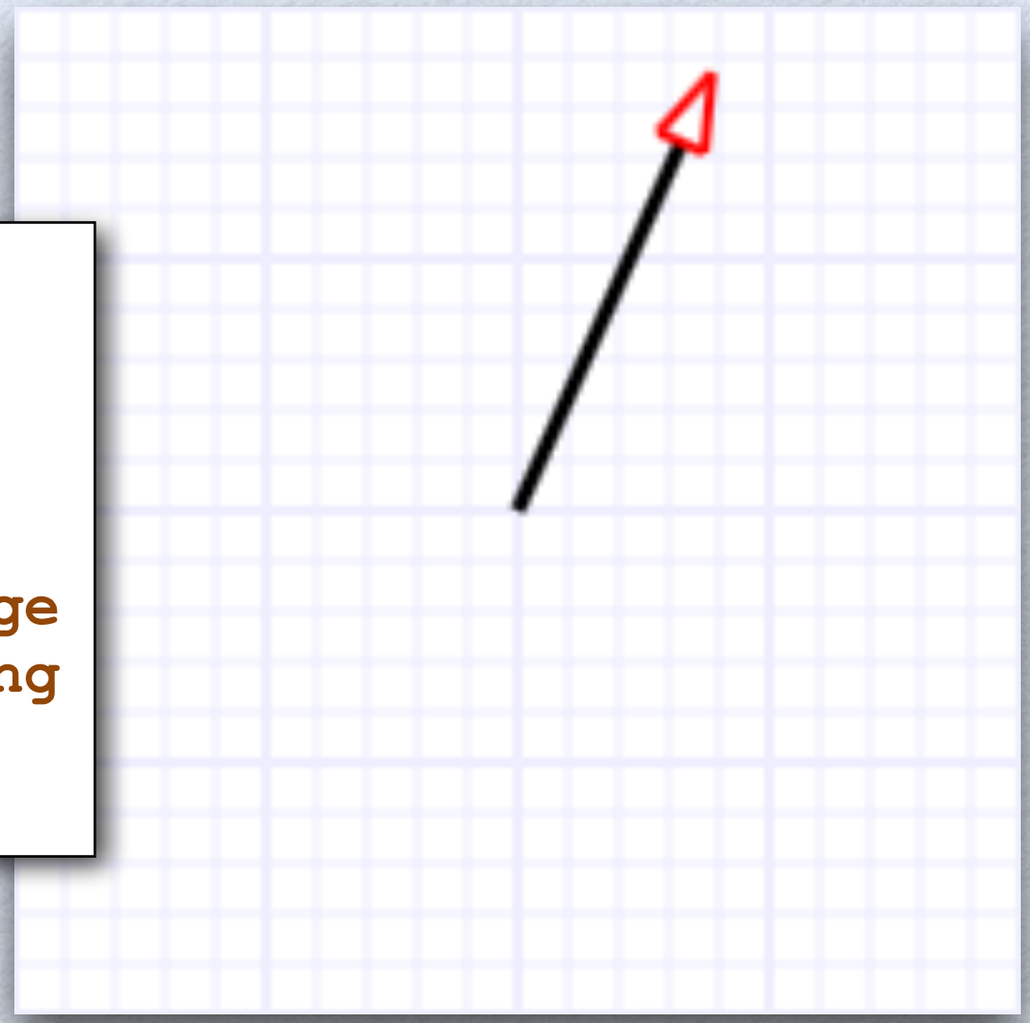
Largeur du tracé

- La largeur du tracé peut être spécifiée avec la fonction **`setpw`**
- **`setpw(`***largeur***`)`** : changer la largeur du tracé à *largeur* pixels (“set pen width”)

Animation

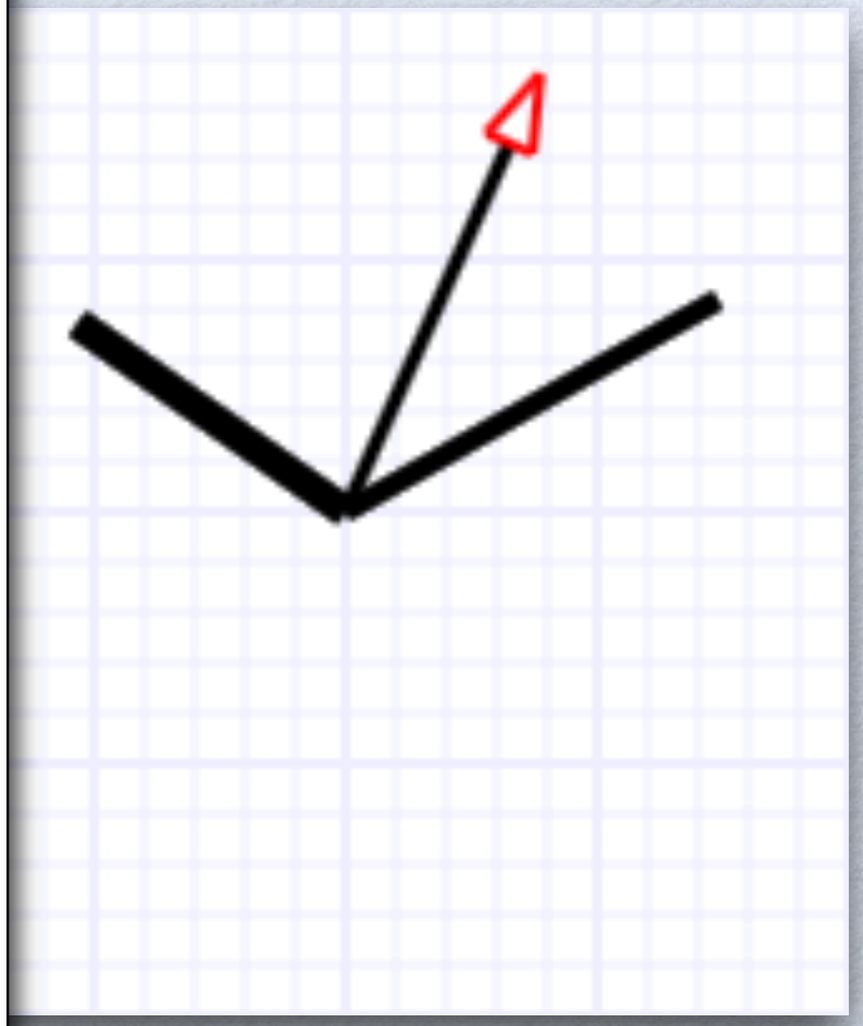
- Des fonctions utiles pour l'animation :
  - **`pause(`***délai***`)`** : rien faire pendant *délai* secondes
  - **`cs()`** : effacer le dessin et centrer la tortue ("clear screen")

```
// Fichier : horloge1.js

for (var secs=0; secs<60; secs++) {
    cs();           // effacer dessin
    rt(secs*6);     // tracer l'aiguille
    setpw(3);       // de 3 pixels de large
    fd(80);         // et 80 pixels de long
    pause(1);       // attendre 1 seconde
}
```



Animation

```
// Fichier : horloge2.js

var aiguille = function (largeur,
                         longueur,
                         angle) {
    // dessiner une aiguille
    rt(angle);
    setpw(largeur);
    fd(longueur);
    if (largeur > 3) { // laisser la tortue au
        bk(longueur);  // bout de l'aiguille
        lt(angle);     // des secondes
    }
};

var secs = (10*60+10)*60; // il est 10:10

while (true) {
    cs();
    aiguille(6, 65, secs/3600*30); // heures
    aiguille(4, 85, secs/60*6);    // minutes
    aiguille(3, 80, secs*6);       // secondes
    pause(1); // attendre 1 seconde
    secs++;
}
```



```
// Fichier: rebondir1.js

var py = 0; // position en y
var vy = 1; // vitesse en y

while (true) {
    cs();
    pu();
    fd(py); // positionner la tortue
    pause(0.01);
    if (py < -99 || py > 83) { // rebondir
        vy = -vy;
    }
   py += vy; // avancer la tortue
}
```

```
// Fichier: rebondir2.js

var px = 0; // position en x
var py = 0; // position en y
var vx = 2; // vitesse en x
var vy = 2; // vitesse en y

while (true) {
    cs();
    pu();
    rt(90); // positionner la tortue
    fd(px);
    lt(90);
    fd(py);
    pause(0.01);
    if (px < -173 || px > 173) { // rebondir
        vx = -vx;
    }
    if (py < -99 || py > 83) { // rebondir
        vy = -vy;
    }
    px += vx; // avancer la tortue
    py += vy;
}
```