Compléments à l'article *Lois de Kepler et simulations numériques*

de Pierre Causeret, Cahiers Clairaut n° 177, p 32 à 34

Les programmes qui suivent sont réalisés pour le langage Processing, téléchargeable gratuitement ici : https://processing.org/
Vous pouvez les adapter à d'autres langages.

1. Démonstration de la première loi

Exemple 1, tracé d'orbite avec le langage Processing et une loi en $1/R^2$

(orbite tracée sans animation)

```
float XS,YS,X1,Y1,X2,Y2,Vx,Vy,d,A,Ax,Ay,Dt ;
int nbpas ;

void setup() {
  size (800, 800) ;
  nbpas=150 ;
  background(255) ;
  XS=400 ; YS=400 ;
  X1 = 400 ; Y1 = 100 ;
  Vx = -4 ; Vy = 0 ;
  Dt=1 ;
  stroke(0) ; fill(255,255,0) ; ellipse(XS,YS,5,5) ;// Dessin du Soleil
  for (int n=0 ; n<nbpas ; n = n+1) {
    X2=X1+Vx*Dt ; Y2=Y1+Vy*Dt ;
    line(X1,Y1,X2,Y2) ;
    d=sqrt((X2-XS)*(X2-XS)+(Y2-YS)*(Y2-YS));
    A=10000/(d*d) ; // Loi en 1/R²
    Ax=-A*(X2-XS)/d ; Ay=-A*(Y2-YS)/d; // acceleration en 1/r
    Vx=Vx+Ax*Dt ; Vy = Vy+Ay*Dt ;
    X1=X2 ; Y1=Y2 ;
  }
}
```

Exemples 2 et 3, tracé d'orbite avec le langage Processing et une loi en $1/R$ ou $1/R^3$

(orbite tracée sans animation)
Il suffit de reprendre le programme précédent en remplaçant la ligne
A=10000/(d*d) ; // Loi en 1/R²
par
A=200/d ; // Loi en 1/R
ou par
A=1700000/(d*d*d) ; // Loi en 1/R3

Si jamais la planète se rapproche trop du Soleil (en particulier avec la loi en 1/R3), la vitesse augmente beaucoup et on est obligé de diminuer le pas si on veut que l'orbite corresponde bien à la loi.

Exemple 4, tracé d'orbite animée avec une loi en $1/R^2$

(orbite tracée point par point)

```
float XS, YS, X, Y, Vx, Vy, A, Ax, Ay, d, Dt ;

void setup() {
 size(800,800) ;
```

```
 background(220,220,240) ;
 XS = 400 ; YS = 400 ;
 X = 400 ; Y = 100 ;
 Vx = -4 ; Vy = 0 ;
 Dt = 0.5 ; //Pas (temps)
 stroke(0) ; fill(250,250,100) ; ellipse(XS,YS,10,10) ; // Soleil
 fill(0) ;
 ellipse(X, Y, 1, 1) ; // Planete
}

void draw() {
 X=X+Vx*Dt ; Y=Y+Vy*Dt ;
 ellipse (X,Y,1,1) ;// place la planete au point (X,Y)
 d=sqrt((X-XS)*(X-XS)+(Y-YS)*(Y-YS));
 A=10000/(d*d) ; // Loi en 1/R²
 Ax=-A*(X-XS)/d ; Ay=-A*(Y-YS)/d ;
 Vx=Vx+Ax*Dt ; Vy = Vy+Ay*Dt ;
}
```

2. Tracé de l'orbite d'une planète

Tracé des orbites de Mercure à Cérès

```
float b, c, X0, Y0, echelle ;

void  setup() {
  size (600, 600) ; background(255) ; stroke(0) ; noFill() ;
  echelle = 100 ;
  fill(255,255,200) ; ellipse(300, 300, 10, 10) ;// Soleil
  noFill() ; strokeWeight(2) ;
  orbite(0.387, 0.206, 77.5) ;// Mercure
  orbite(0.723,0.007,131.8);//Venus
  orbite(1,0.017,102.9);//Terre
  orbite(1.523,0.093,336.1);//Mars
  orbite(2.768,0.076,153.1);//Cérès
  save ("DessinsPNG/orbites.png") ;
}

void orbite(float a, float e, float l) { //trace l'orbite connaisssant a, e et lp
   c=e*a ; b=sqrt(a*a - c*c) ;
   pushMatrix();
   translate(300,300) ;
   rotate(-l*PI/180) ;
   ellipse(-c*echelle, 0, 2*a*echelle, 2*b*echelle) ;
   popMatrix();
}
```

3. Mouvement d'une planète (animation)

Animation, la Terre se déplaçant sur son orbite

Les valeurs des paramètres a, e et T sont ceux de la Terre, où l'on voit que l'orbite est proche d'un cercle. Ils peuvent être modifiés pour montrer des orbites plus excentriques, par exemple pour Mercure :
a = 58 ; e = 0.206 ; T = 88 ; // parametres pour Mercure.

```
float a, b, c, e, T, ep ; //demi-grand axe, demi petit axe, distance foyer centre, excentricite, periode en j
float M, E ; // anomalie moyenne et anomalie excentrique
float r, s ; // rayon vecteur et anomalie vraie
float X0, Y0 ; // Coordonnees du centre
float XP, YP ; // Coordonnees de la planete
float t0, t ; // t0 au depart

void setup() {
  a = 150 ; e = 0.017 ; T = 365 ; // parametres pour la Terre
  c = a*e ; b = sqrt(a*a-c*c) ; ep=sqrt((1+e)/(1-e)) ;
  X0 = 400 ; Y0 = 300 ;
  size(800,600) ;
  background(255) ;
  orbite() ;
  fill(100,100,255) ;
  t0=millis();
}

void orbite() {
  //efface et retrace ellipse
  fill(255) ; noStroke() ;
  rect (0,0,800,600) ;
  noFill() ; stroke(0) ;
  ellipse(X0-c,Y0,a*2,b*2) ; // Trace l'ellipse
  fill(255, 255, 150) ;
  ellipse(X0, Y0, 20, 20) ; // Trace le foyer
}

void draw() {
    t = millis()-t0 ;
    M = t*0.23/T ;// anomalie moyenne avec 1 an = 10 s
    TrouveE(M,e) ;
    r=a*(1-e*cos(E)); s = 2*atan(ep*tan(E/2)) ; // Calcul des coordonnees polaires
    XP=X0+r*cos(s) ; YP = Y0-r*sin(s) ; // Calcul des coordonnees cartesiennes
    orbite() ;
    stroke(0) ; fill(100,100,255) ; ellipse(XP,YP,10,10) ; // place la planète bleue
}

void TrouveE(float M, float e) { // Resolution de l'equation de Kepler
    E=M ;
    for (int i =0; i<6 ; i=i+1) {
      E=M+e*sin(E);
    }
}
```