

```

float xmag, ymag = 0;
float newXmag, newYmag = 0;
int count = 0;

float scNode, scLine, scAngle, scLabel, scScene, rndMoov; //Scales
int Background = 255;

String[] Tag = new String[14];
String Tag(String Prfx, String Sufx)
{
    Tag[ 0]=Prfx+"                o                " +Sufx;
    Tag[ 1]=Prfx+"                oooo                oo oo                " +Sufx;
    Tag[ 2]=Prfx+"                oooooooo oooo oooooooo oo oo                " +Sufx;
    Tag[ 3]=Prfx+"                oo oooo oooooooo oooo oo ooooo                " +Sufx;
    Tag[ 4]=Prfx+"                oo oooooo oooooooo oo ooooo                " +Sufx;
    Tag[ 5]=Prfx+"                ooo oooooo oooooooo ooo ooooo                " +Sufx;
    Tag[ 6]=Prfx+"                oooooooooo oooooooooooo oo oooo                " +Sufx;
    Tag[ 7]=Prfx+"                oooooo oooooooo oooo o oo oo                " +Sufx;
    Tag[ 8]=Prfx+"                oo oooo                " +Sufx;
    Tag[ 9]=Prfx+" --( Guillaume ooo LaBelle )-- goo.x@gmx.ch                " +Sufx;
    Tag[10]=Prfx+"                oo oo                " +Sufx;
    Tag[11]=Prfx+"                ooo ooo                " +Sufx;
    Tag[12]=Prfx+"                oooooooo                " +Sufx;
    Tag[13]=Prfx+"                oooo                " +Sufx;

    return join(Tag, "\n");
}

BFont LogicFont;
void setup()
{
    size(500, 550);
    initSlider();
    initButtons();

    smooth();

    ellipseMode(CENTER_DIAMETER);
    LogicFont = loadFont("SlideFont.vlw");

    initScale();
}
int tempMem = 6;

void loop()
{
    background(Background);
    updateButtons();
    updateSlider();

    push();
    updateOrbit();
    drawKnot();
    pop();
}

//-----
//-----
//--( VUES )--
//-----

void updateOrbit(){
    translate(width/2, width/2, -30);
    if(key=='1')
    {
        newXmag = mouseX/float(width) * TWO_PI;
        newYmag = mouseY/float(height) * TWO_PI;
        key='a';
    }
    float diff = xmag-newXmag;
    if (abs(diff) > 0.01) { xmag -= diff/4.0; }
    diff = ymag-newYmag;
    if (abs(diff) > 0.01) { ymag -= diff/4.0; }
    rotateX(-ymag);
    rotateY(-xmag);
}

```

```

    if(key=='2')
    {
        scScene*=0.875+(mouseY/float(height))/4;
        key='a';
    }
    scale(100*scScene);
}

void initScale(){
    scNode = 1;
    scLine = 1;
    scAngle = 3;
    scLabel = 0.75;
    scScene = 0.070;
    rndMoov = 0.01;
}

//-----
//-----
//--( Sliders )--
//-----
/*Global*/
BFont slideFont;
HScrollbar[] GooSlider = new HScrollbar[6];
////////////////////////////////////
//Sliders fn
////////////////////////////////////
void initSlider(){
    slideFont = loadFont("SlideFont.vlw");
    int co = 200;
    GooSlider[0] = new HScrollbar(10/*xPos*/,
height-10*1/*yPos*/,width-100/*width*/,8/*height*/,05/*friction*/,0.6/*init*/,0/*MIN*/,10/*MAX*/, "C",
color(co,co,co) );
    GooSlider[1] = new HScrollbar(10/*xPos*/,
height-10*2/*yPos*/,width-100/*width*/,8/*height*/,05/*friction*/,17/*init*/,1/*MIN*/,60/*MAX*/, "B",
color(co,co,co) );
    GooSlider[2] = new HScrollbar(10/*xPos*/,
height-10*3/*yPos*/,width-100/*width*/,8/*height*/,05/*friction*/,10/*init*/,1/*MIN*/,30/*MAX*/, "A",
color(co,co,co) );
    GooSlider[3] = new HScrollbar(10/*xPos*/,
height-10*4/*yPos*/,width-100/*width*/,8/*height*/,05/*friction*/,8.22/*init*/,0/*MIN*/,10/*MAX*/, "LW",
color(co,co/2,0) );
    GooSlider[4] = new HScrollbar(10/*xPos*/,
height-10*5/*yPos*/,width-100/*width*/,8/*height*/,05/*friction*/,1.28/*init*/,0/*MIN*/,10/*MAX*/, "LH",
color(co,co/2,0) );
    GooSlider[5] = new HScrollbar(10/*xPos*/,
height-10*6/*yPos*/,width-100/*width*/,8/*height*/,05/*friction*/,100/*init*/,1/*MIN*/,200/*MAX*/, "NUM",
color(co,co/2,0) );
    //GooSlider[2] = new HScrollbar(10/*xPos*/,
height-10*3/*yPos*/,width-100/*width*/,8/*height*/,35/*friction*/,0.5/*init*/,0/*MIN*/,1/*MAX*/, "PHY",
color(co,co,co) );
}

void updateSlider(){
    for(int i=0;i<GooSlider.length; i++)
    {
        GooSlider[i].update();
        GooSlider[i].draw();
    }
}

////////////////////////////////////
//Sliders
////////////////////////////////////
class HScrollbar
{
    int swidth, sheight; // width and height of bar
    int xPos, yPos; // x and y position of bar
    float spos, newspos; // x position of slider
    int sposMin, sposMax; // max and min values of slider
    int loose; // how loose/heavy
    boolean over; // is the mouse over the slider?
    boolean locked;
}

```

```

float ratio;
float value;
float valueMAX;
float valueMIN;

color couleur;
String name;

HScrollbar (int xPos, int yPos, int sw, int sh, int l, float v, float MIN, float MAX, String name, color couleur)
{
    swidth = sw;
    sheight = sh;
    int widthtoheight = sw - sh;
    ratio = (float)sw / (float)widthtoheight;
    this.xPos = xPos;
    this.yPos = yPos-sheight/2;

    spos = (swidth*v)/(MAX-MIN);
    newspos = spos;
    sposMin = xPos;
    sposMax = xPos + swidth - sheight;
    loose = 1;
    value = v;
    valueMAX = MAX;
    valueMIN = MIN;

    this.couleur=couleur;
    this.name = name;
}

void update() {
    if(over()) {
        over = true;
    } else {
        over = false;
    }
    if(mousePressed && over) {
        locked = true;
    }
    if(!mousePressed) {
        locked = false;
    }
    if(locked) {
        newspos = constrain(mouseX-sheight/2, sposMin, sposMax);
    }
    if(abs(newspos - spos) > 1) {
        spos = spos + (newspos-spos)/loose;
    }
}

int constrain(int val, int minv, int maxv) {
    return min(max(val, minv), maxv);
}

boolean over() {
    if(mouseX > xPos && mouseX < xPos+swidth &&
        mouseY > yPos && mouseY < yPos+sheight) {
        return true;
    } else {
        return false;
    }
}

void draw() {
    fill(red(couleur), green(couleur), blue(couleur), 150);
    stroke(red(couleur), green(couleur), blue(couleur) );
    line(xPos, yPos+sheight/2, xPos+swidth, yPos+sheight/2);
    rect(spos, yPos, sheight, sheight);

    fill(red(couleur), green(couleur), blue(couleur), 30);
    noStroke();
    rect(xPos, yPos+sheight/4, swidth, sheight/2);

    fill(red(couleur), green(couleur), blue(couleur));
}

```

```

    value = getPos();
    textFont(slideFont, 18);
    text( " "+name+" "+nf(value, 0, 2), xPos+swidth, yPos+sheight/2+6);
}

float getPos() {
    return ((valueMAX-valueMIN)*(spos * ratio))/swidth;
}

}

////////////////////////////////////
//Buttons fn
////////////////////////////////////
int Bval=7;
RectButton[] Buttons = new RectButton[10];
void initButtons(){
    color base = color(255,225,150);
    color high = color(255,150,50);
    int w=15;
    int s=2;
    for(int i =0;i<Buttons.length;i++)
        Buttons[i] = new RectButton(w*i+s,s,w-s,base,high,i);
}

void updateButtons(){
    for(int i =0;i<Buttons.length;i++)
    {
        if(locked == false) Buttons[i].update();
        else locked = false;
        if(mousePressed) if(Buttons[i].pressed()) Bval = Buttons[i].value;
        Buttons[i].display();
    }
}

////////////////////////////////////
//Buttons
////////////////////////////////////
boolean locked =false;

class RectButton
{
    int x, y;
    int size;
    color basecolor, highlightcolor;
    color currentcolor;
    int value;
    boolean over = false;
    boolean pressed = false;

    void update()
    {
        if(over()) {
            currentcolor = highlightcolor;
        } else {
            currentcolor = basecolor;
        }
    }

    boolean pressed()
    {
        if(over) {
            locked = true;
            return true;
        } else {
            locked = false;
            return false;
        }
    }

    RectButton(int _x, int _y, int _size, color _color, color _highlight, int _value)
    {
        x = _x;
        y = _y;

```

```

size = _size;
basecolor = _color;
highlightcolor = _highlight;
currentcolor = basecolor;
value=_value;
}

boolean over()
{
  if( overRect(x, y, size, size) ) {
    over = true;
    return true;
  } else {
    over = false;
    return false;
  }
}

void display()
{
  stroke(255);
  fill(currentcolor);
  rect(x, y, size, size);
}
}
boolean overRect(int x, int y, int width, int height)
{
  if (mouseX >= x && mouseX <= x+width &&
  mouseY >= y && mouseY <= y+height) {
    return true;
  } else {
    return false;
  }
}

//-----
//-----
//--( KNOT )--
//-----

//Fonction du noeud Basée sur les Équations Paramétriques
gPoint KnotEq(float gPHz){
  float A = GooSlider[2].getPos(); //10
  float B = GooSlider[1].getPos(); //15
  float C = GooSlider[0].getPos(); //2

  gPoint Pt=new gPoint();
  float scale = 1;
  Pt.x = (scale*(-A*cos(gPHz) - C*cos(5*gPHz) + B*sin(2*gPHz)) );
  Pt.y = (scale*(-B*cos(2*gPHz) + A*sin(gPHz) -C*sin(5*gPHz)) );
  Pt.z = (scale*(A*cos(3*gPHz)) );
  return Pt;
}

float cont = 0;
float step = 10;
void drawKnot(){
  float lenW =GooSlider[3].getPos();
  float lenH =GooSlider[4].getPos();
  float num =GooSlider[5].getPos();
  gPoint[][] coo = new gPoint[(int)num+1][4];

  cont++;
  if(cont>step-1) cont=0;
  float cf = ((cont/step))/num;

  for(float i=0; i<num; i++)
  {
    float phz1 = (i/num+cf)*((float)2*PI);
    float phz2 = ((i+1)/num+cf)*((float)2*PI);

    gPoint A1 = new gPoint(KnotEq(phz1));
    gPoint A2 = new gPoint(KnotEq(phz2));

```

```

FrenetFrame Z = new FrenetFrame(A1,A2);

if(Bval==0){Z.setScalar(lenW);Z.draw();}
if(Bval==1) Z.drawFace(lenW,lenH);
if(Bval==2){Z.setScalar(lenW);Z.drawRGB();}
if(Bval==3) Z.drawStep(lenW,lenH);
if(Bval==4) Z.drawStep2(lenW,lenH);
if(Bval==5) Z.drawStep3(lenW/2,lenH);
if(Bval==6) Z.drawStep4(lenW/2,lenH);
if(Bval==7) Z.drawMainCourantes(lenW/2,lenH,coo,(int)i);
if(Bval==8) Z.drawMainCourantes2(lenW/2,lenH,coo,(int)i);
}

if(Bval==7||Bval==8)
{
for(int j=0;j<4;j++)
{
beginShape(LINE_STRIP);
curveVertex(coo[0][j].x, coo[0][j].y, coo[0][j].z);
for(int i=0; i<coo.length; i++)
curveVertex(coo[i][j].x, coo[i][j].y, coo[i][j].z);

curveVertex(coo[0][j].x, coo[0][j].y, coo[0][j].z);
curveVertex(coo[1][j].x, coo[1][j].y, coo[1][j].z);
endShape();
}
}
}

//-----
//-----
//--( Vec3D )--
//-----
// Vec3D - simple 3D vector class
// processing.unlekker.net

class Vec3D {
float x,y,z;

Vec3D(){}
Vec3D(float _x,float _y,float _z) {
x=_x;
y=_y;
z=_z;
}

Vec3D(Vec3D v) {
x=v.x;
y=v.y;
z=v.z;
}

void set(float _x,float _y,float _z) {
x=_x;
y=_y;
z=_z;
}

void set(Vec3D v) {
x=v.x;
y=v.y;
z=v.z;
}

void add(float _x,float _y,float _z) {
x+=_x;
y+=_y;
z+=_z;
}

void add(Vec3D v) {

```

```

    x+=v.x;
    y+=v.y;
    z+=v.z;
}

void sub(float _x,float _y,float _z) {
    x-=_x;
    y-=_y;
    z-=_z;
}

void sub(Vec3D v) {
    x-=v.x;
    y-=v.y;
    z-=v.z;
}

void mult(float m) {
    x*=m;
    y*=m;
    z*=m;
}

void div(float m) {
    x/=m;
    y/=m;
    z/=m;
}

float length() {
    return sqrt(x*x+y*y+z*z);
}

void normalise() {
    float l=length();
    if(l!=0) {
        x/=l;
        y/=l;
        z/=l;
    }
}
}

//-----
//-----
//--( gPoint )--
//-----

class gPoint extends Vec3D {

gPoint(){}
gPoint(float x,float y,float z) {this.x=x;this.y=y;this.z=z;}
gPoint(Vec3D v) {x=v.x;y=v.y;z=v.z;}

////////////////////////////////////
float lengthRapid() {
    return x*x+y*y+z*z;
}
////////////////////////////////////
void normalize() {
    float l=length();
    if(l!=0) {
        x/=l;
        y/=l;
        z/=l;
    }
}
////////////////////////////////////
void transform(float[][] M){
    x = M[0][0]*x + M[1][0]*y + M[2][0]*z + M[3][0];
    y = M[0][1]*x + M[1][1]*y + M[2][1]*z + M[3][1];
    z = M[0][2]*x + M[1][2]*y + M[2][2]*z + M[3][2];
}
}

```

```

void translate(gPoint V){
    float[][] M =
    {
        {1,0,0},
        {0,1,0},
        {0,0,1},
        {V.x,V.y,V.z}
    };
    transform(M);
}
void scale(float _x,float _y,float _z){
    float[][] M =
    {
        {_x,0,0},
        {0,_y,0},
        {0,0,_z},
        {0,0,0}
    };
    transform(M);
}
void scale(float _x){
    float[][] M =
    {
        {_x,0,0},
        {0,_x,0},
        {0,0,_x},
        {0,0,0}
    };
    transform(M);
}
void draw(){
    line(0,0,0,x,y,z);
}
void setScalar(float Sc){
    float OldLength = length();
    float ratio = Sc/OldLength;
    scale(ratio);
}
}

gPoint vcross(gPoint A, gPoint B){
    double Cx = (double)A.y*(double)B.z - (double)A.z*(double)B.y;
    double Cy = (double)A.z*(double)B.x - (double)A.x*(double)B.z;
    double Cz = (double)A.x*(double)B.y - (double)A.y*(double)B.x;
    return new gPoint((float)Cx, (float)Cy, (float)Cz);
}

class FrenetFrame {
    //Major Vector is Herited

    gPoint v0;    //The Vector at the origin [Tangent]
    gPoint v0n;
    gPoint v1,v2; //Points
    gPoint T0,T;  //Tangent Normalized
    gPoint B0,B;
    gPoint N0,N;

    FrenetFrame(gPoint v1,gPoint v2){
        this.v1=new gPoint(v1);
        this.v2=new gPoint(v2);
        v0 = new gPoint(v2); v0.sub(v1);
        T0 = new gPoint(v0); T0.normalize(); //T.add(v1);
        B0 = new gPoint(vcross(T0,v2));
        N0 = new gPoint(vcross(B0,T0));
        B0.normalize();
        N0.normalize();
        setTBN();
    }
    void setTBN(){
        T = new gPoint(T0); T.add(v1);
        B = new gPoint(B0); B.add(v1);
        N = new gPoint(N0); N.add(v1);
    }
    void setScalar(float cf){

```

```

T = new gPoint(T0); T.setScalar(cf); T.add(v1);
B = new gPoint(B0); B.setScalar(cf); B.add(v1);
N = new gPoint(N0); N.setScalar(cf); N.add(v1);
}
void draw(){
stroke(255,225,150); line(v1.x,v1.y,v1.z,T.x,T.y,T.z);
stroke(255,225,150); line(v1.x,v1.y,v1.z,B.x,B.y,B.z);
stroke(255,225,150); line(v1.x,v1.y,v1.z,N.x,N.y,N.z);
}
void drawRGB(){
stroke(255,0,0); line(v1.x,v1.y,v1.z,T.x,T.y,T.z);
stroke(0,255,0); line(v1.x,v1.y,v1.z,B.x,B.y,B.z);
stroke(0,0,255); line(v1.x,v1.y,v1.z,N.x,N.y,N.z);
}
void drawFace(float w,float h){
B = new gPoint(B0); B.setScalar(w); B.add(v1);
N = new gPoint(N0); N.setScalar(h); N.add(v1);
gPoint Bx = new gPoint(B0); Bx.setScalar(-w); Bx.add(v1);
gPoint Nx = new gPoint(N0); Nx.setScalar(-h); Nx.add(v1);
stroke(0);
fill(255,150,0,50);
beginShape(QUADS);
vertex(B.x,B.y,B.z);
vertex(N.x,N.y,N.z);
vertex(Bx.x,Bx.y,Bx.z);
vertex(Nx.x,Nx.y,Nx.z);
endShape();
}
void drawStep2(float w,float h){
gPoint Bx = new gPoint(B0); Bx.setScalar(h);
gPoint Tx = new gPoint(T0); Tx.setScalar(h);
gPoint W0 = new gPoint(B0); W0.setScalar(w);
gPoint W1 = new gPoint(W0); W1.add(Bx); W1.add(Tx);
gPoint W2 = new gPoint(B0); W2.setScalar(-w);
gPoint W3 = new gPoint(W2); W3.add(Bx); W3.add(Tx);
W0.add(v1);
W1.add(v1);
W2.add(v1);
W3.add(v1);

stroke(0);
fill(255,150,0,50);
beginShape(QUADS);
vertex(W0.x,W0.y,W0.z);
vertex(W1.x,W1.y,W1.z);
vertex(W2.x,W2.y,W2.z);
vertex(W3.x,W3.y,W3.z);
endShape();
}
void drawStep(float w,float h){
gPoint Bx = new gPoint(B0); Bx.setScalar(h);
gPoint Tx = new gPoint(T0); Tx.setScalar(h);
gPoint W0 = new gPoint(N0); W0.setScalar(w);
gPoint W1 = new gPoint(W0); W1.add(Bx); W1.add(Tx);
gPoint W2 = new gPoint(N0); W2.setScalar(-w);
gPoint W3 = new gPoint(W2); W3.add(Bx); W3.add(Tx);
W0.add(v1);
W1.add(v1);
W2.add(v1);
W3.add(v1);

stroke(0);
fill(255,150,0,50);
beginShape(QUADS);
vertex(W0.x,W0.y,W0.z);
vertex(W1.x,W1.y,W1.z);
vertex(W3.x,W3.y,W3.z);
vertex(W2.x,W2.y,W2.z);
endShape();
}
void drawStep3(float w,float h){
gPoint Bx = new gPoint(B0); Bx.setScalar(h);
gPoint Tx = new gPoint(T0); Tx.setScalar(h);
gPoint W0 = new gPoint(N0); W0.setScalar(w);

```

```

    gPoint W1 = new gPoint(W0); W1.add(Bx); W1.add(Tx);
    gPoint W2 = new gPoint(N0); W2.setScalar(-w);
    gPoint W3 = new gPoint(W2); W3.add(Bx); W3.add(Tx);
);xd1ntint M0 = new gPoint(W0); M0.add(Bx);
    gPoint M1 = new gPoint(W2); M1.add(Bx);
    gPoint M2 = new gPoint(M0); Tx.setScalar(h*5); M2.add(Tx); Bx.setScalar(-h*5); M2.add(Bx);
    gPoint M3 = new gPoint(M1); M3.add(Tx); M3.add(Bx);
    gPoint W4 = new gPoint(W0); Tx.setScalar(-h/3); W4.add(Tx); Bx.setScalar(-h/3); W4.add(Bx);
    gPoint W5 = new gPoint(W2); W5.add(Tx); W5.add(Bx);

W0.add(v1);
W1.add(v1);
W2.add(v1);
W3.add(v1);
W4.add(v1);
W5.add(v1);
M0.add(v1);
M1.add(v1);
M2.add(v1);
M3.add(v1);

stroke(0);
fill(255, 150, 0, 50);
beginShape(QUADS);
vertex(W0.x, W0.y, W0.z);
vertex(W1.x, W1.y, W1.z);
vertex(W3.x, W3.y, W3.z);
vertex(W2.x, W2.y, W2.z);
vertex(W0.x, W0.y, W0.z);
vertex(W2.x, W2.y, W2.z);
vertex(W5.x, W5.y, W5.z);
vertex(W4.x, W4.y, W4.z);
endShape();

line(M0.x, M0.y, M0.z, M2.x, M2.y, M2.z);
line(M1.x, M1.y, M1.z, M3.x, M3.y, M3.z);
}
void drawStep4(float w, float h){
    gPoint Bx = new gPoint(B0); Bx.setScalar(h);
    gPoint Tx = new gPoint(T0); Tx.setScalar(h);
    gPoint W0 = new gPoint(N0); W0.setScalar(w);
    gPoint W1 = new gPoint(W0); W1.add(Bx); W1.add(Tx);
    gPoint W2 = new gPoint(N0); W2.setScalar(-w);
    gPoint W3 = new gPoint(W2); W3.add(Bx); W3.add(Tx);
    gPoint M0 = new gPoint(W0); M0.add(Bx);
    gPoint M1 = new gPoint(W2); M1.add(Bx);
    gPoint M2 = new gPoint(M0); Tx.setScalar(h*5); M2.add(Tx); Bx.setScalar(-h*5); M2.add(Bx);
    gPoint M3 = new gPoint(M1); M3.add(Tx); M3.add(Bx);
    gPoint W4 = new gPoint(W0); Tx.setScalar(-h/3); W4.add(Tx); Bx.setScalar(-h/3); W4.add(Bx);
    gPoint W5 = new gPoint(W2); W5.add(Tx); W5.add(Bx);

W0.add(v1);
W1.add(v1);
W2.add(v1);
W3.add(v1);
W4.add(v1);
W5.add(v1);
M0.add(v1);
M1.add(v1);
M2.add(v1);
M3.add(v1);

stroke(0);
fill(255, 225, 150);
beginShape(QUADS);
vertex(W0.x, W0.y, W0.z);
vertex(W1.x, W1.y, W1.z);
vertex(W3.x, W3.y, W3.z);
vertex(W2.x, W2.y, W2.z);
vertex(W0.x, W0.y, W0.z);
vertex(W2.x, W2.y, W2.z);
vertex(W5.x, W5.y, W5.z);
vertex(W4.x, W4.y, W4.z);
endShape();

```

```

    line(M0.x,M0.y,M0.z,M2.x,M2.y,M2.z);
    line(M1.x,M1.y,M1.z,M3.x,M3.y,M3.z);
}
void drawMainCourantes(float w,float h, gPoint[][] coo, int i){
    gPoint Bx = new gPoint(B0); Bx.setScalar(h);
    gPoint Tx = new gPoint(T0); Tx.setScalar(h);
    gPoint W0 = new gPoint(N0); W0.setScalar(w);
    gPoint W1 = new gPoint(W0); W1.add(Bx); W1.add(Tx);
    gPoint W2 = new gPoint(N0); W2.setScalar(-w);
    gPoint W3 = new gPoint(W2); W3.add(Bx); W3.add(Tx);
    gPoint M0 = new gPoint(W0); M0.add(Bx);
    gPoint M1 = new gPoint(W2); M1.add(Bx);
    gPoint M2 = new gPoint(M0); Tx.setScalar(h*5); M2.add(Tx); Bx.setScalar(-h*5); M2.add(Bx);
    gPoint M3 = new gPoint(M1); M3.add(Tx); M3.add(Bx);
    gPoint W4 = new gPoint(W0); Tx.setScalar(-h/3); W4.add(Tx); Bx.setScalar(-h/3); W4.add(Bx);
    gPoint W5 = new gPoint(W2); W5.add(Tx); W5.add(Bx);

    W0.add(v1);
    W1.add(v1);
    W2.add(v1);
    W3.add(v1);
    W4.add(v1);
    W5.add(v1);
    M0.add(v1);
    M1.add(v1);
    M2.add(v1);
    M3.add(v1);

    stroke(0);
    fill(255,225,150);
    beginShape(QUADS);
    vertex(W0.x,W0.y,W0.z);
    vertex(W1.x,W1.y,W1.z);
    vertex(W3.x,W3.y,W3.z);
    vertex(W2.x,W2.y,W2.z);
    vertex(W0.x,W0.y,W0.z);
    vertex(W2.x,W2.y,W2.z);
    vertex(W5.x,W5.y,W5.z);
    vertex(W4.x,W4.y,W4.z);
    endShape();

    line(M0.x,M0.y,M0.z,M2.x,M2.y,M2.z);
    line(M1.x,M1.y,M1.z,M3.x,M3.y,M3.z);

    coo[i][0]=M0;
    coo[i][1]=M1;
    coo[i][2]=M2;
    coo[i][3]=M3;
}
void drawMainCourantes2(float w,float h, gPoint[][] coo, int i){
    gPoint Bx = new gPoint(B0); Bx.setScalar(h);
    gPoint Tx = new gPoint(T0); Tx.setScalar(h);
    gPoint W0 = new gPoint(N0); W0.setScalar(w);
    gPoint W1 = new gPoint(W0); W1.add(Bx); W1.add(Tx);
    gPoint W2 = new gPoint(N0); W2.setScalar(-w);
    gPoint W3 = new gPoint(W2); W3.add(Bx); W3.add(Tx);
    gPoint M0 = new gPoint(W0); M0.add(Bx);
    gPoint M1 = new gPoint(W2); M1.add(Bx);
    gPoint M2 = new gPoint(M0); Tx.setScalar(h*5); M2.add(Tx); Bx.setScalar(-h*5); M2.add(Bx);
    gPoint M3 = new gPoint(M1); M3.add(Tx); M3.add(Bx);
    gPoint W4 = new gPoint(W0); Tx.setScalar(-h/3); W4.add(Tx); Bx.setScalar(-h/3); W4.add(Bx);
    gPoint W5 = new gPoint(W2); W5.add(Tx); W5.add(Bx);

    W0.add(v1);
    W1.add(v1);
    W2.add(v1);
    W3.add(v1);
    W4.add(v1);
    W5.add(v1);
    M0.add(v1);
    M1.add(v1);
    M2.add(v1);
    M3.add(v1);

```

```
stroke(0);
fill(255, 150, 0, 50);
beginShape(QUADS);
vertex(W0.x, W0.y, W0.z);
vertex(W1.x, W1.y, W1.z);
vertex(W3.x, W3.y, W3.z);
vertex(W2.x, W2.y, W2.z);
vertex(W0.x, W0.y, W0.z);
vertex(W2.x, W2.y, W2.z);
vertex(W5.x, W5.y, W5.z);
vertex(W4.x, W4.y, W4.z);
endShape();

line(M0.x, M0.y, M0.z, M2.x, M2.y, M2.z);
line(M1.x, M1.y, M1.z, M3.x, M3.y, M3.z);

coo[i][0]=M0;
coo[i][1]=M1;
coo[i][2]=M2;
coo[i][3]=M3;
}
}
```