```
String [] pos2d;
String [] pos3d;
String [] boundb;
import signal.library.*;

// Create the filter

SignalFilter Filterkx0;
SignalFilter Filterkx1;
SignalFilter Filterky0;
SignalFilter Filterky1;
SignalFilter Filterb1;
SignalFilter Filterb2;
SignalFilter Filterl;
SignalFilter Filtera;
SignalFilter Filterd;

int index = 38575;
int end= 43800;
float ang;



min
import org.philhosoft.p8g.svg.P8gGraphicsSVG;

void setup ()
{
  frameRate(25);
  Filterkx0 = new SignalFilter(this);
  Filterkx0.setMinCutoff(minCutoff);
  Filterkx0.setBeta(beta);
  Filterkx1 = new SignalFilter(this);
  Filterkx1.setMinCutoff(minCutoff);
  Filterkx1.setBeta(beta);
  Filterky0 = new SignalFilter(this);
  Filterky0.setMinCutoff(minCutoff);
  Filterky0.setBeta(beta);
  Filterky1 = new SignalFilter(this);
  Filterky1.setMinCutoff(minCutoff);
  Filterky1.setBeta(beta);
  Filterb1 = new SignalFilter(this);
  Filterb2 = new SignalFilter(this);
  Filterl = new SignalFilter(this);
  Filtera = new SignalFilter(this);
  Filterd = new SignalFilter(this);
  minim = new Minim(this);
  player = minim.loadFile("ros.mp3");

  size (1000, 1000);
```

```
  background (255);
  frameRate(25);

  pos2d = loadStrings("Tracked2DPos_CamCenter_25fps.txt");
  // println("pos2d hat " + pos2d.length+ " Zeilen");

  pos3d = loadStrings("Tracked3DPosition_25fps.txt");
  //println("pos3d hat " + pos3d.length+ " Zeilen");

  boundb = loadStrings("BoundingBox_CamCenter_25fps.txt" );
  //println("boundb hat " + boundb.length+" Zeilen");
}

void draw ()

{
  if (index<end) {
    beginRecord(P8gGraphicsSVG.SVG, "test"+index+".svg");
    player.play();
    float level = player.mix.level()*10000;



    // ang=36;
    //println(level);
    stroke(0);
    fill(0);
    //noFill();

    float Kx0= Filterkx0.filterUnitFloat(float(pos3d[index].split(" ")[0]));
    float Ky0= Filterky0.filterUnitFloat(float(pos3d[index].split(" ")[1]));
    PVector K0 = new PVector(Kx0, Ky0);

    float Kx1= Filterkx1.filterUnitFloat(float(pos3d[index+1].split(" ")[0]));
    float Ky1= Filterky1.filterUnitFloat(float(pos3d[index+1].split(" ")[1]));
    PVector K1 = new PVector(Kx1, Ky1);

    int Kx2d= int(pos2d[index].split(" ")[0]);
    int Ky2d= int(pos2d[index].split(" ")[1]);

    float d = Filterd.filterUnitFloat(K0.dist(K1)); //Distanz bzw. Geschwindigkeit

    float deltaY = Ky1 - Ky0;
    float deltaX = Kx1 - Kx0;
    float a = Filtera.filterUnitFloat(abs(atan2(deltaY, deltaX))); //Richtungsänderung


    float a2d = map(Kx2d, 0, 1920, 0, PI/2.0);
```

```
    float b1 = Filterb1.filterUnitFloat(int(boundb[index].split(" ")[0]));
    float b2 = Filterb2.filterUnitFloat(int(boundb[index].split(" ")[2]));
    float b= b2-b1;
    //ang= map(mouseX, 0, width, 0.0, 2*PI);
    ang= Filterl.filterUnitFloat(map(b, 70, 200, 0.0, PI/7));
    //ang= Filterl.filterUnitFloat(map(level, 0, 1000, PI/10, PI));

    background (255);
    translate(width/2, height/2);
    // beginShape(TRIANGLE);

    // println(d);

    /*
pushMatrix();

     ellipse(0, 0, 10, 10);
     rotate(-a0);
     translate(0, d*2000);
     ellipse(0, 0, 10, 10);
     popMatrix();
     */

    rotate(radians(-180));
    pushMatrix();



    float a0 = 0*ang;
    pushMatrix();
    int dx= int(sin(a0)*(d*2000+40));
    int dy= int(cos(a0)*(d*2000+40));

    ellipse(dx, dy, 7, 7);
    line(0, 0, dx, dy);
    popMatrix();

    float a1 = a0+ang;
    pushMatrix();
    int bx= int(sin(a1)*(Kx0*20));//+10 damit mindestens 10 ist
    int by= int(cos(a1)*(Kx0*20));

    ellipse(bx, by, 7, 7);
    line(0, 0, bx, by);
    popMatrix();




    float a3 = a0-ang;
```

```
 pushMatrix();
 int cx= int(sin(a3)*(10+Ky0*50));//+10 damit mindestens 10 ist
 int cy= int(cos(a3)*(10+Ky0*50));

 ellipse(cx, cy, 7, 7);
 line(0, 0, cx, cy);
 popMatrix();


 float a4= a3-ang;
 pushMatrix();
 int ex= int(sin(a4)*(10+a*100));//+10 damit mindestens 10 ist
 int ey= int(cos(a4)*(10+a*100));
 ellipse(ex, ey, 7, 7);
 line(0, 0, ex, ey);
 popMatrix();

 float a5 = a1+ang;
 pushMatrix();
 int ax= int(sin(a5)*(b));//+10 damit mindestens 10 ist
 int ay= int(cos(a5)*(b));

 ellipse(ax, ay, 7, 7);
 line(0, 0, ax, ay);

 popMatrix();


 line (dx, dy, cx, cy);
 line (dx, dy, bx, by);
 line(ax, ay, bx, by);
 line(cx, cy, ex, ey);









 /*
int P1y= int(sin(0)*Kx);
  int P1x= int(cos(0)*Kx);
  println("KOPFX "+ P1x + " " +P1y);

  int PYy= int(sin(ang)*Ky);
  int PYx= int(cos(ang)*Ky);
  println("KOPFY " +PYx + " " +PYy);
  //rotate(radians(-ang));
  ellipse(P1x, P1y, 10, 10);
  ellipse(PYx, PYy, 10, 10);
```

```
      line(P1x, P1y, PYx, PYy);
      //line(P1y, P1x, PYy, PYx);
      */

     popMatrix();
     index=index+1;
     println(ang);
     endRecord();
   } else {
     endRecord();
     println("Done");
   }
   if (index == end) {
   }
}
```