```
14.11.23
                                             antial.cc
#include <algorithm>
#include <cmath>
#include <iostream>
#include <numeric>
#include <vector>
#include <cppqt.h>
using namespace std;
void drawAntialiasedWideLine(Drawing& pic, DPoint2D p1, DPoint2D p2,
                           double w, int f = 4, int colour = 0)
// zeichnet eine Linie von p1 nach p2 der Dicke w, der Algorithmus verwendet
// zur Berechnung ein f-fach in x- und y-Richtung verfeinertes Raster
//<<<<
   if (p1.y > p2.y)
       swap(p1, p2);
                                          // dann ist p2 der obere Punkt
                                      // Eckpunkte des Linienrechtecks im
 DPoint2D p11, p12, p21, p22;
                                      // feineren Raster
 DPoint2D nv(p2.y - p1.y, p1.x - p2.x); // Normalenvektor (zeigt erstmal immer
                                       // nach rechts bzw. im Extremfall nach
                                       // oben oder unten)
 if (nv.y < 0)
   nv = -1.0 * nv:
                                       // dann zeigt nv nicht nach unten
 nv = nv / norm(nv);
                                       // normiere nv
                                                    file:///home/stiklas/Bildgen-Sessions/Excercises/03/antial.cc
```

```
// berechnen wir die Eckpunkte mittels Normalvektor
p11 = static cast < double > (f) * (p1 - 0.5 * w * nv);
p12 = static cast < double > (f) * (p1 + 0.5 * w * nv);
p21 = static cast < double > (f) * (p2 - 0.5 * w * nv);
p22 = static cast < double > (f) * (p2 + 0.5 * w * nv);
// nv zeigt immer nach oben, deshalb liegt p11 unterhalb von p12 und
// p21 unterhalb von p22.
//ymin un ymax in f-fach Raster
double yminf = p11.y;
double ymaxf = p22.y;
//x-Bereich im Original-Raster
int xmax = static_cast<int>(ceil(max(p12.x / f, p21.x / f)));
// Anzahl Zeilen im feinen Raster
// Selbst für ymaxf == 0 wird immernoch eine Zeile benötigt.
int numrows = static cast<int>(ceil(ymaxf)) + 1;
// Linker und rechter Rand der Linie im feinen Raster.<<<
//Initialisiere so, dass die Ränder sukzessive aktualisiert werden können,
//denn es ist nicht bekannt ob eine Seite des Rechtecks "links" bzw.
// "rechts" liegt.
vector<int> linkerrand(numrows, numeric limits<int>::max());  //aller werte=MAX INT
vector<int> rechterrand(numrows, -1); //vector mit Anzahl Zeilen Elementen, alle =-1
int v:
double x:
double einsdurchm;
                                                       file:///home/stiklas/Bildgen-Sessions/Excercises/03/antial.cc
```

antial.cc

14.11.23

antial.cc

// Bestimmen wir den x-Bereich für jede Bildzeile im feinen Raster den x-Bereich.

linkerrand[y] = min(linkerrand[y], static_cast<int>(round(x)));
rechterrand[y] = max(rechterrand[y], static_cast<int>(round(x)));

// Ob eine Seite des Rechtecks "links" bzw. "rechts" liegt, ist unbekannt.

// We will simply compute the edges of the big rectangle.

x += einsdurchm;

14.11.23

```
// Bereich zwischen p11.y und p21.y
    einsdurchm = (p11.x - p21.x) / (p11.y - p21.y);
    x = p11.x + (ceil(p11.y) - p11.y) * einsdurchm;
    for (y = \text{static cast} < \text{int} > (\text{ceil}(p11.y)); y <= floor(p21.y); ++y)
        linkerrand[y] = min(linkerrand[y], static cast<int>(round(x)));
        rechterrand[y] = max(rechterrand[y], static cast<int>(round(x)));
        x += einsdurchm;
    // Bereich zwischen p12.y und p22.y
    einsdurchm = (p12.x - p22.x) / (p12.y - p22.y);
    x = p12.x + (ceil(p12.y) - p12.y) * einsdurchm;
    for (y = static cast<int>(ceil(p12.y)); y <= floor(p22.y); ++y)</pre>
        linkerrand[y] = min(linkerrand[y], static cast<int>(round(x)));
        rechterrand[y] = max(rechterrand[y], static cast<int>(round(x)));
        x += einsdurchm;
  }//endif
// Konvertiert y-Bereich von verfeinerten Raster im Original-Raster
int ymin = static_cast<int>(round(1.0 / f * yminf));
int ymax = static cast<int>(round(1.0 / f * ymaxf));
// Für jedes Pixel der ursprünglichen Pixelzeile, summiere die
// Intensitäten innerhalb dieser Zeile. // Ein Pixel mehr, für Rechenungenauigkeiten.
vector<int> xx(xmax + 1, 0);
```

file:///home/stiklas/Bildgen-Sessions/Excercises/03/antial.cc

antial.cc

14.11.23

if (p1.y != p2.y)

```
int xxmin, xxmax; //T. Var
int xi, xf, xfend, z;
// Schleife über die Zeilen des Originalbildes
for (y = ymin; y \le ymax; ++y)
    // Enthält später den linken bzw. rechten Rand der zugehörigen
    // echten Pixelzeile.
    xxmin = numeric limits<int>::max();
    xxmax = -1:
    // Schleife über die f Zeilen des feineren Rasters pro echter Zeile
    for (z = 0; z < f \&\& f * y + z <= ymaxf; ++z)
        // Finde den vorher berechneten Rand der Subpixelzeile
        xf = linkerrand[f * y + z];
        xfend = rechterrand[f * y + z];
        // Wenn mindestens ein Subpixel
        if (xf <= xfend)</pre>
            // bestimme kleinstes und größtes x in Subpixeln
            xxmin = min(xf, xxmin);
```

antial.cc

14.11.23

// Addiere die Pixel des feineren Bildes auf. Laufe dazu über
// die Subpixelzeile und addiere im Eintrag, der zum Originalpixel

xxmax = max(xfend, xxmax);

// im Vektor xx gehört.
for (; xf <= xfend; ++xf)</pre>

xx[xf / f] += 1;

```
14.11.23
                                                   antial.cc
      // Wenn Zeile nicht leer
      if (xxmin <= xxmax)</pre>
        // Schleife über den zu zeichnenden Bereich in Zeile y, in echten Pixeln
        for (xi = xxmin / f; xi <= xxmax / f; xi++)</pre>
          // Schleife über den zu zeichnenden Bereich in Zeile y
            // bestimme die Intensität des Pixels
            // 0 → 255 (weiß)
            // f<sup>2</sup> → col (die gewünschte Farbe der Linie)
            double n = 255;
            double m = (colour - 255.0) / (f * f);
            pic.drawPoint(xi, y, static cast<int>(round( m * xx[xi] + n )),
                           false);
            // lösche den Intensitätswert für die nächste Zeile
            xx[xi] = 0;
int maindraw()
 Drawing pic1(170, 170);
 pic1.show();
 DPoint2D p1, p2;
 double width;
 int f;
                                                           file:///home/stiklas/Bildgen-Sessions/Excercises/03/antial.cc
```

```
14.11.23
                                                   antial.cc
 cout << "Dicke, Raster: ";</pre>
 cin >> width >> f;
 for (int theta = 0; theta <= 90; theta += 9)
      p1 = DPoint2D(10, 10);
      p2 = p1 + 150.0 * DPoint2D(sin(theta * M PI / 180.0))
                                   cos(theta * M PI / 180.0));
      drawAntialiasedWideLine(pic1, p1, p2, width, f, 0);
 cin.get();
 cout << "Return zum Vergrößern" << endl;</pre>
 cin.get();
 pic1.setZoom(4);
 IOThread::waitForWindow(60);
 return 0;
                                                            file:///home/stiklas/Bildgen-Sessions/Excercises/03/antial.cc
```