easy.Filter

The Beauty of Bresenham's Algorithm

A simple implementation to plot lines, circles, ellipses and Bézier curves.

The Algorithm

This page introduces a compact and efficient implementation of Bresenham's algorithm to plot lines, circles, ellipses and Bézier curves.

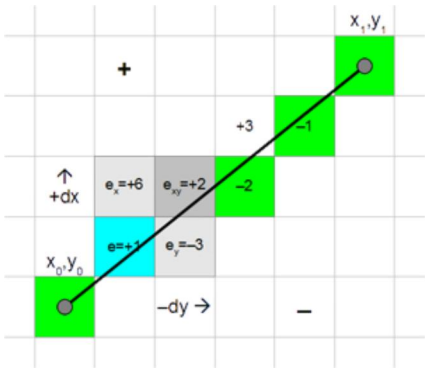
A detailed documentation of the algorithm is under development..

Four C-program examples of the document are listed below.

Line

A simple example of Bresenham's line algorithm.

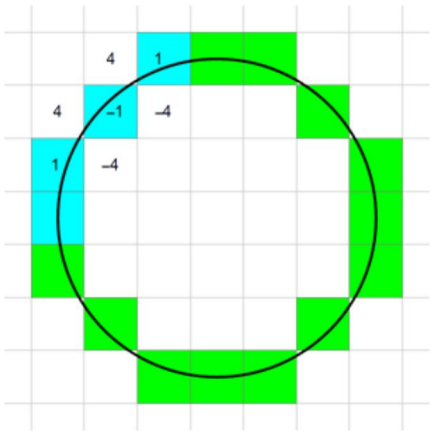
```
void plotLine(int x0, int y0, int x1, int y1)
{
  int dx =  abs(x1-x0), sx = x0<x1 ? 1 : -1;
  int dy = -abs(y1-y0), sy = y0<y1 ? 1 : -1;
  int err = dx+dy, e2; /* error value e_xy */

  for(;;){  /* loop */
    setPixel(x0,y0);
    if (x0==x1 && y0==y1) break;
    e2 = 2*err;
    if (e2 >= dy) { err += dy; x0 += sx; } /* e_xy+e_x > 0 */
    if (e2 <= dx) { err += dx; y0 += sy; } /* e_xy+e_y < 0 */
  }
}
```



Circle

This is an implementation of the circle algorithm.

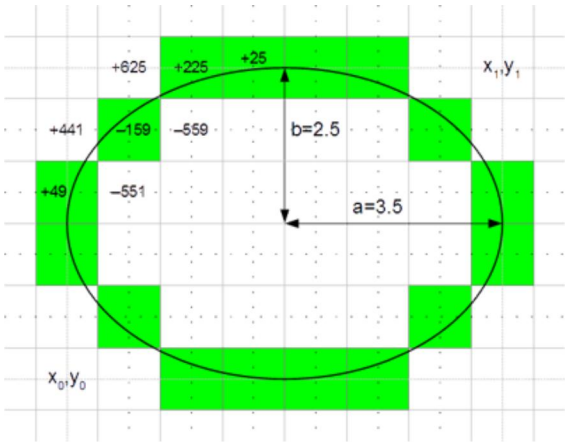
```
void plotCircle(int xm, int ym, int r)
{
   int x = -r, y = 0, err = 2-2*r; /* II. Quadrant */
   do {
      setPixel(xm-x, ym+y); /*   I. Quadrant */
      setPixel(xm-y, ym-x); /*  II. Quadrant */
      setPixel(xm+x, ym-y); /* III. Quadrant */
      setPixel(xm+y, ym+x); /*  IV. Quadrant */
      r = err;
      if (r >  x) err += ++x*2+1; /* e_xy+e_x > 0 */
      if (r <= y) err += ++y*2+1; /* e_xy+e_y < 0 */
   } while (x < 0);
}
```



Ellipse

This program example plots an ellipse inside a specified rectangle.

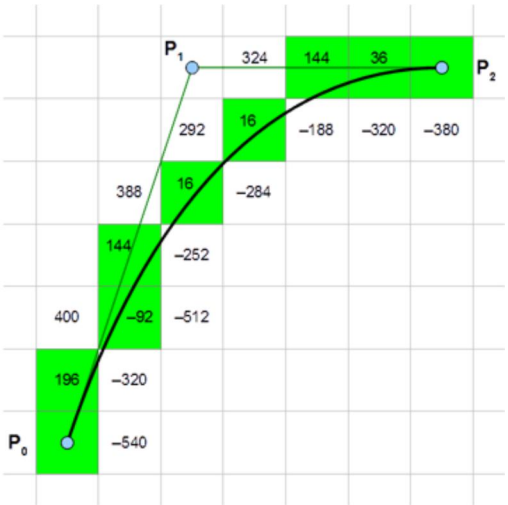
```
void plotEllipseRect(int x0, int y0, int x1, int y1)
{
   int a = abs(x1-x0), b = abs(y1-y0), b1 = b&1; /* values of diameter */
   long dx = 4*(1-a)*b*b, dy = 4*(b1+1)*a*a; /* error increment */
   long err = dx+dy+b1*a*a, e2; /* error of 1.step */

   if (x0 > x1) { x0 = x1; x1 += a; } /* if called with swapped points */
   if (y0 > y1) y0 = y1; /* .. exchange them */
   y0 += (b+1)/2; y1 = y0-b1;   /* starting pixel */
   a *= 8*a; b1 = 8*b*b;

   do {
       setPixel(x1, y0); /*   I. Quadrant */
       setPixel(x0, y0); /*  II. Quadrant */
       setPixel(x0, y1); /* III. Quadrant */
       setPixel(x1, y1); /*  IV. Quadrant */
       e2 = 2*err;
       if (e2 >= dx) { x0++; x1--; err += dx += b1; } /* x step */
       if (e2 <= dy) { y0++; y1--; err += dy += a; }  /* y step */
   } while (x0 <= x1);

   while (y0-y1 < b) {  /* too early stop of flat ellipses a=1 */
       setPixel(x0-1, y0); /* -> finish tip of ellipse */
       setPixel(x1+1, y0++);
       setPixel(x0-1, y1);
       setPixel(x1+1, y1--);
   }
}
```



Bézier curve

This program example plots a quadratic Bézier curve limited to gradients without sign change.

```
void plotBasicBezier(int x0, int y0, int x1, int y1, int x2, int y2)
{
  int sx = x0<x2 ? 1 : -1, sy = y0<y2 ? 1 : -1; /* step direction */
  int cur = sx*sy*((x0-x1)*(y2-y1)-(x2-x1)*(y0-y1)); /* curvature */
  int x = x0-2*x1+x2, y = y0-2*y1+y2, xy = 2*x*y*sx*sy;
                              /* compute error increments of P0 */
  long dx = (1-2*abs(x0-x1))*y*y+abs(y0-y1)*xy-2*cur*abs(y0-y2);
  long dy = (1-2*abs(y0-y1))*x*x+abs(x0-x1)*xy+2*cur*abs(x0-x2);
                              /* compute error increments of P2 */
  long ex = (1-2*abs(x2-x1))*y*y+abs(y2-y1)*xy+2*cur*abs(y0-y2);
  long ey = (1-2*abs(y2-y1))*x*x+abs(x2-x1)*xy-2*cur*abs(x0-x2);

                            /* sign of gradient must not change */
  assert((x0-x1)*(x2-x1) <= 0 && (y0-y1)*(y2-y1) <= 0);

  if (cur==0) { plotLine(x0,y0,x2,y2); return; } /* straight line */

  x *= 2*x; y *= 2*y;
  if (cur < 0) {                            /* negated curvature */
    x = -x; dx = -dx; ex = -ex; xy = -xy;
    y = -y; dy = -dy; ey = -ey;
  }
  /* algorithm fails for almost straight line, check error values */
  if (dx >= -y || dy <= -x || ex <= -y || ey >= -x) {
    plotLine(x0,y0,x1,y1);                /* simple approximation */
    plotLine(x1,y1,x2,y2);
    return;
  }
  dx -= xy; ex = dx+dy; dy -= xy;             /* error of 1.step */

  for(;;) {                                         /* plot curve */
    setPixel(x0,y0);
    ey = 2*ex-dy;                /* save value for test of y step */
    if (2*ex >= dx) {                                 /* x step */
      if (x0 == x2) break;
      x0 += sx; dy -= xy; ex += dx += y;
    }
    if (ey <= 0) {                                    /* y step */
      if (y0 == y2) break;
      y0 += sy; dx -= xy; ex += dy += x;
    }
  }
```



```
}
```
