**public** **static** **void** BresenhamCircle(Graphics g, Color clr, **int** \_x, **int** \_y, **int** radius)

{

**int** x = 0, y = radius, gap = 0, delta = (2 - 2 \* radius);

**while** (y >= 0)

{

PutPixel(g, clr, \_x + x, \_y + y, 255);

PutPixel(g, clr, \_x + x, \_y - y, 255);

PutPixel(g, clr, \_x - x, \_y - y, 255);

PutPixel(g, clr, \_x - x, \_y + y, 255);

gap = 2 \* (delta + y) - 1;

**if** (delta < 0 && gap <= 0)

{

x++;

delta += 2 \* x + 1;

**continue**;

}

**if** (delta > 0 && gap > 0)

{

y--;

delta -= 2 \* y + 1;

**continue**;

}

x++;

delta += 2 \* (x - y);

y--;

}

}

В данном приложении помещены процедуры генерации окружностей по алгоритму Брезенхема и Мичнера, а также программа T\_Circle для тестирования данных процедур.

/\*--------------------------------------------------- V\_Circle

\* Подпрограммы для генерации окружности

\* Pixel\_circle - занесение пикселов с учетом симметрии

\* V\_BRcirc - генерирует окружность по алгоритму

\* Брезенхема.

\* V\_MIcirc - генерирует окружность по алгоритму

\* Мичнера.

\*/

#include <graphics.h>

/\*----------------------------------------------- Pixel\_circle

\* Заносит пикселы окружности по часовой стрелке

\*/

static void Pixel\_circle (xc, yc, x, y, pixel)

int xc, yc, x, y, pixel;

{

putpixel(xc+x, yc+y, pixel);

putpixel(xc+y, yc+x, pixel);

putpixel(xc+y, yc-x, pixel);

putpixel(xc+x, yc-y, pixel);

putpixel(xc-x, yc-y, pixel);

putpixel(xc-y, yc-x, pixel);

putpixel(xc-y, yc+x, pixel);

putpixel(xc-x, yc+y, pixel);

} /\* Pixel\_circle \*/

/\*--------------------------------------------------- V\_BRcirc

\* Генерирует 1/8 окружности по алгоритму Брезенхема

\*

\* Процедура может строить 1/4 окружности.

\* Для этого надо цикл while заменить на for (;;)

\* и после Pixel\_circle проверять достижение конца по условию

\* if (y <= end) break;

\* Где end устанавливается равным 0

\* В этом случае не нужен и последний оператор

\* if (x == y) Pixel\_circle (xc, yc, x, y, pixel);

\* Генерацию 1/8 можно обеспечить задав end = r / sqrt (2)

\*/

void V\_BRcirc (xc, yc, r, pixel)

int xc, yc, r, pixel;

{ int x, y, z, Dd;

x= 0; y= r; Dd= 2\*(1-r);

while (x < y) {

Pixel\_circle (xc, yc, x, y, pixel);

if (!Dd) goto Pd;

z= 2\*Dd - 1;

if (Dd > 0) {

if (z + 2\*x <= 0) goto Pd; else goto Pv;

}

if (z + 2\*y > 0) goto Pd;

Pg: ++x; Dd= Dd + 2\*x + 1; continue; /\* Горизонт \*/

Pd: ++x; --y; Dd= Dd + 2\*(x-y+1); continue; /\* Диагонал \*/

Pv: --y; Dd= Dd - 2\*y + 1; /\* Вертикал \*/

}

if (x == y) Pixel\_circle (xc, yc, x, y, pixel);

} /\* V\_BRcirc \*/

/\*--------------------------------------------------- V\_MIcirc

\* Генерирует 1/8 окружности по алгоритму Мичнера

\*/

void V\_MIcirc (xc, yc, r, pixel)

int xc, yc, r, pixel;

{ int x, y, d;

x= 0; y= r; d= 3 - 2\*r;

while (x < y) {

Pixel\_circle (xc, yc, x, y, pixel);

if (d < 0) d= d + 4\*x + 6; else {

d= d + 4\*(x-y) + 10; --y;

}

++x;

}

if (x == y) Pixel\_circle (xc, yc, x, y, pixel);

} /\* V\_MIcirc \*/

/\*=============================================== T\_CIRCLE.C

\*

\* ТЕСТ ГЕНЕРАЦИИ ОКРУЖНОСТЕЙ

\*

\* Запрашивает ввод четырех чисел - координат центра,

\* радиуса и цвета построения: Xc Yc R Pix

\*

\* Затем строит заданную окружность по алгоритму Брезенхема

\* и концентрично с ней с радиусом, уменьшенным на 2, и

\* номером цвета, уменьшенным на 1, выдает окружность по

\* алгоритму Мичнера.

\*

\* При вводе Xc < 0 программа прекращает работу

\*/

#include <graphics.h>

#include <stdio.h>

#include "V\_CIRCLE.C"

/\*-------------------------------------------- MAIN T\_CIRCLE.C

\*/

void main (void)

{

int ii, Xc=300, Yc=240, R=238, Pix=14;

int gdriver = DETECT, gmode;

initgraph(&gdriver, &gmode, "c:\tc\bgi");

if ((ii= graphresult()) != grOk) {

printf ("Err=%d\n", ii); goto all;

}

setbkcolor(0);

cleardevice();

for (;;) {

gotoxy (1,1);

printf(" \r");

printf("Xc, Yc, R, Pix= (%d %d %d %d) ? ", Xc,Yc,R,Pix);

scanf ("%d%d%d%d", &Xc, &Yc, &R, &Pix);

if (Xc < 0) break;

V\_BRcirc (Xc, Yc, R, Pix);

V\_MIcirc (Xc, Yc, R-2, Pix-1);

}

all:

closegraph();

}

Исходники этого алгоритма, взятые из других источников

void Circle(int x, int y, int r,unsigned char color)

{

int x1,y1,yk = 0;

int sigma,delta,f;

x1 = 0;

y1 = r;

delta = 2\*(1-r);

do

{

PutPixel(x+x1,y+y1,color);

PutPixel(x-x1,y+y1,color);

PutPixel(x+x1,y-y1,color);

PutPixel(x-x1,y-y1,color);

f = 0;

if (y1 < yk)

break;

if (delta < 0)

{

sigma = 2\*(delta+y1)-1;

if (sigma <= 0)

{

x1++;

delta += 2\*x1+1;

f = 1;

}

}

else

if (delta > 0)

{

sigma = 2\*(delta-x1)-1;

if (sigma > 0)

{

y1--;

delta += 1-2\*y1;

f = 1;

}

}

if (!f)

{

x1++;

y1--;

delta += 2\*(x1-y1-1);

}

}

while(1);

}

На Паскале:

Procedure Circle(x,y,rr:integer);

var xi,yi,r,di,lim,s,ss:integer;

label 1,2,3,4,10,20,30;

Begin

r:=rr;

xi:=0; yi:=r; di:=2\*(1-r); lim:=0;

1: SetPixel(xi+x,yi+y);

SetPixel(xi+x,-yi+y);

SetPixel(-xi+x,yi+y);

SetPixel(-xi+x,-yi+y);

if yi<limthen goto 4;

if di<0then goto 2;

if di>0then goto 3;

if di=0 then goto 20;

2: s:=2\*di+2\*yi-1;

if s<=0then goto 10;

if s>0then goto 20;

3: s:=2\*di+2\*xi-1;

if s<=0then goto 20;

if s>0then goto 30;

10:xi:=xi+1;

di:=di+2\*xi+1;

goto 1;

20:xi:=xi+1;

yi:=yi-1;

di:=di+2\*xi-2\*yi+2;

goto 1;

30:yi:=yi-1;

di:=di-2\*yi+1;

goto 1;4:

end;

=================== ВИКИПЕДИЯ =======================

Рисование окружностей

Также существует алгоритм Брезенхэма для рисования окружностей. По методу построения он похож на рисование линии. В этом алгоритме строится дуга окружности для первого квадранта, а координаты точек окружности для остальных квадрантов получаются симметрично. На каждом шаге алгоритма рассматриваются три пикселя, и из них выбирается наиболее подходящий путём сравнения расстояний от центра до выбранного пикселя с радиусом окружности.

Разложение окружности в растр

// R - радиус, X1, Y1 - координаты центра

**int** x := 0

**int** y := R

**int** delta := 1 - 2 \* R

**int** error := 0

**while** (y >= 0)

drawpixel(X1 + x, Y1 + y)

drawpixel(X1 + x, Y1 - y)

drawpixel(X1 - x, Y1 + y)

drawpixel(X1 - x, Y1 - y)

error = 2 \* (delta + y) - 1

**if** ((delta < 0) && (error <= 0))

delta += 2 \* ++x + 1

**continue**

error = 2 \* (delta - x) - 1

**if** ((delta > 0) && (error > 0))

delta += 1 - 2 \* --y

**continue**

x++

delta += 2 \* (x - y)

y--

Литература

* *Роджерс Д.* Алгоритмические основы машинной графики. — М.: Мир, 1989. — С. 54-63. — [ISBN 5-03-000476-9](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/5030004769).
* *Шилдт Г.* "Си" для профессиональных программистов. — М., 1989.