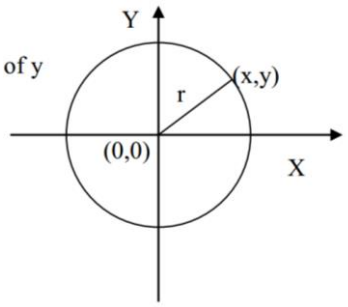
Тойрог зурах энгийн алгоритм

Координатын эх дээр төвлөрсөн тойргийн тэгшитгэл ба радиус дараах байдлаар өгөгдсөн.

$x^2 + y^2 = r^2$

$=> y = \pm\sqrt{x^2 - y^2}$



- Нэгж алхамаар x-ийг нэмэгдүүлж, дээрх тэгшитгэлээс харгалзах y-ийн утгыг тодорхойлно. Дараа нь (x, y) байрлалд пикселийг тохируулно.
- Алхамыг *–r* –ээс *+r* хүртэл авна.
- Компьютерийн графикт бид дэлгэцэнд дүрслэгдэх цэгийн координатын эхийг зүүн дээд булангаас авдаг. Тиймээс үзэгдэх зурсан тойргууд (0,0) цэгээс өөр цэг дээр төвлөрсөн байх болно. Хэрэв тойргийн төв нь *(xc,yc)* бол координатын төвөөс тооцоолсон цэг *(x+xc, y+yc)* цэгийн байрлалд шилжинэ.

Ерөнхийдөө *(xc,yc)*-д төвлөрсөн тойргийн томъёо ба радиус нь

$(x - xc)^2 + (y - yc)^2 = r^2$

$=> y = yc \pm \sqrt{r^2 - (x - xc^2}$ ……..(1)

Тойргийн цэгүүдийн байрлалыг тооцоолоход энэ тэгшитгэлийг ашигладаг. x утгийн хувьд xc-r -с *xc+r* хүртэл алхамуудыг гүйцэтгэх ба (x,y) цэгийн байрлалын хувьд y-байрлалын харгалзах утгуудыг тооцоолно. Энэ алгоритм нь энгийн боловч,

- Цаг хугацаа их шаарддаг – квадрат язгуур ба квадрат тооцоолол
- Муруйн налуугийн өөрчлөлтийн улмаас жигд бус зай үүсдэг. Хэрэв жигд бус зай үүссэн бол |m|>y налуугийн хувьд x ба y солихоос зайлсхийх хэрэгтэй, энэ нь маш их тооцоолол хийх шаардлага тулгардаг.

Доорхи програм нь дээрх тэгшитгэлийг (1) ашиглан тойргийн энгийн тооцооллыг харуулав.

```
//тойрог зурах програм
#include<stdio.h> #include<conio.h>
#include<stdlib.h> #include<math.h>
#include<graphics.h> #define
SQUARE(x) ((x)*(x))
void drawcircle(int ,int,int); void main()
{
        int gd,gm,err; int
        xc,yc,r;
        gd=DETECT;
        initgraph(&gd,&gm,"\\tc\\bgi"); err=graphresult();
        if(err!=0)
        {
                printf("ERROR:%s",grapherrormsg(err)); printf("\nPress a
                key..");
                getch();
                exit(1);
        }
        xc=getmaxx()/2;
        yc=getmaxy()/2; r=50;
        drawcircle(xc,yc,r); getch();
        closegraph();
```

```
}//end main
void drawcircle(int xc,int yc,int r)
{
        int i,x,y,y1;
        for(i=xc-r;i<=xc+r;i++)
        {
                x=i;
                y=yc+sqrt(SQUARE(r)-SQUARE(x-xc)); y1=yc-
                sqrt(SQUARE(r)-SQUARE(x-xc)); putpixel(x,y,1);
                putpixel(x,y1,1);
        }
}
```

Туйлийн тэгшитгэл (polar equations) ашиглан тойрог зурах

Хэрэв (x,y) нь (0,0) төв ба r радиустай тойргийн хязгаар дээрх цэг бол

$x = r\cos\theta$

$x = r\sin\theta$

Жнь $(x, y) = (r\cos\theta , r\sin\theta)$

Эдгээр координатын аргыг ашиглан тойрог зурахын тулд 0-ээс 360 хүртэлх өнцгийг нэмэгдүүлэх хэрэгтэй. Өсөлтийн өнцөгт харгалзах (x,y) байрлалыг тооцоолно. Эхлэлийн цэгийг хаана ч төвлөрүүлэн тойргийг зурж болох ч координатын эх дээр төвлөрсөн цэг дэлгэцэнд бүрэн харагдах боломжгүй болно. Хэрэв тойргийн төв (xc,yc) гэж өгвөл тойргийн зам дээрх (x,y) пикселийн байрлалыг дараах байдлаар тооцоолно.

$x = xc + r\cos\theta$

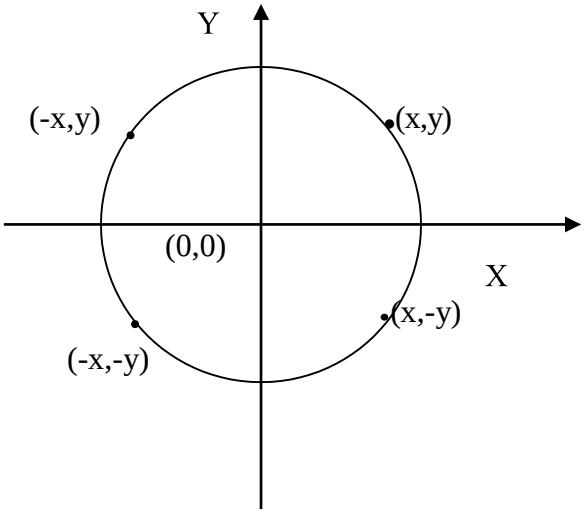
$y = yc + r\sin\theta$

Туйлын хувиргалт (polar transformation) ашиглан тойрог зурах тойргийн функц:

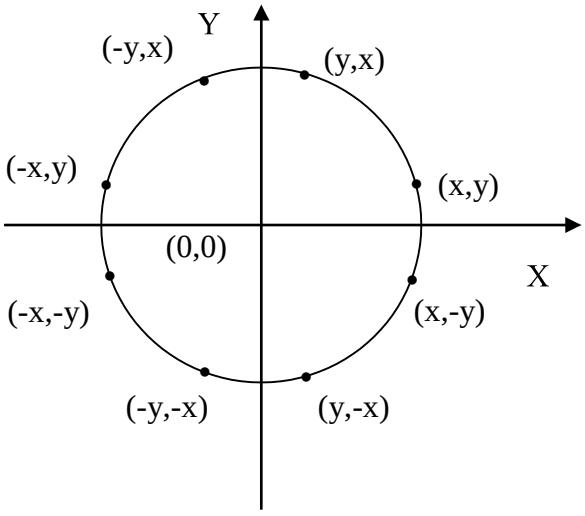
```
void drawcircle(int xc,int yc,int r)
{
        int x,y; float
        theta;
        const float PI=3.14;

        for(theta=0.0;theta<=360;theta+=1)
        {
                x=      xc+r*cos(theta*PI/180.0);       y=
                yc+r*sin(theta*PI/180.0); putpixel(x,y,1);

        }
}
```

Тойргийн скан хөрвүүлэлт (circle scan conversion)-ийн тэгш хэм:

Бид тойргийн тэгш хэмийг ашиглах тойрог үүсгэхэд шаардагдах хугацааг багасгаж чадна. Жишээ нь 4-чиглэлт эсвэл 8-чиглэлт тэш хэм. Иймд бид зөвхөн нэг квадрат буюу октантын хувьд цэгүүдийг үүсгэх ба бусад цэгүүдийг тодорхойлоход тэгш хэмийг ашиглана.



4-way symmetry



8-way symmetry

Дээрх функцүүдэд квадрат язгуур болон тригнометрийн функцууд арилаагүй байгаа тул тооцооллын асуудал тэгш хэмийг ашигласаар байна.

Дундаж цэгийн тойргийн алгоритм (Mid point circle Algorithm):

Дундаж цэгийн тойргийн алгоритмд бид нэгж интервал дээр түүвэрлэж, алхам тус бүрт тойргийн замд хамгийн ойр пикселийн байрлалыг тодорхойлно.

r радиус болон (*xc*, *yc*) дэлгэцийн төвийн байршил өгөгдсөн ба бид (0,0) дээр төвлөрсөн тойргийн эргэн тойрон дахь пикселүүдийг тооцоолох алгоритмаа тохируулж, дараа нь тооцоолсон пикселийн байршил (x, y)-г x дээр xc, y дээр yc нэмж байрлалыг шилжүүлнэ. Жишээ нь: $x = x + xc, y = y + yc$.

Дундаж цэгийн аргийг хэрэгжүүлэхийн тулд бид тойргийн функцийг тодорхойлно:

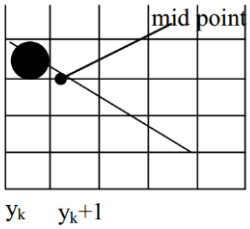
$f_{circle} = x^2 + y^2 - r^2$

(x, y)цэгт хамааралтай байрлалыг fcircle тэмдгээр шалган, дүгнэх,

$$f_{circle}(x, y) \begin{cases} <0, \text{ if } (x, y) \text{ тойргийн хязгаар дотор орших} \\ =0, \text{ if } (x, y) \text{ тойргийн хязгаар дээр орших} \\ >0, \text{ if } (x, y) \text{ тойргийн хязгаарын гадна орших.} \end{cases}$$

Тойргийн функцын туршилтыг түүврийн алхам бүрт тойргийн замын ойролцоох пикселүүдийн хоорондох дундаж байрлал дээр гүйцэтгэнэ. Иймд тойргийн функц нь дундаж цэгийн алгоритмын шийдвэрийн параметр юм.

Зурагт x_k □ 1 түүврийн байрлал дахь сонгон авсан 2 цэгийн хоорондох дундын цэгийг харуулав. Бид (x_k, y_k) пикселийг тэмдгэлсэн гэж үзээд, тойрогт ойр $(x_k \square 1, y_k)$ *эсвэл* $(x_k \square 1, y_k \square 1)$ пиксел байгаа эсэхийг тодорхойлох шаардлагатай.
Бидний шийдвэрийн параметр нь дундаж цэг дээр тооцоолол хийдэг тойргийн функц юм.



$$p_k = f_{circle}(x_k + 1, y_k - \frac{1}{2})$$
$$= (x_k + 1)^2 + (y_k - 1/2)^2 - r^2 = (x_k + 1)^2 + y_k^2 - y_k + \frac{1}{4} - r^2$$

Хэрэв p_k □ 0 бол mid-point тойргийн дотор оршидог тул y_k дээрх цэг тойргийн хязгаартай ойр байх ба дараагийн пикселийн байршлийг сонгоход y_k □1 ойр байна.

Шийдвэрийн дараалсан параметрүүдийг өсөн нэмэгдэх тооцооллоор гарган авдаг. Дараагийн байршилийг тооцоолох шийдвэрийн параметрийг $x_{k\square1}$ □ 1 түүврийн байрлал дахь тойргийн функцийг үнэлэх замаар тооцоолно. x_k □ 2 as

$$p_{k+1} = f_{circle}(x_{k+1} + 1, y_{k+1} - \frac{1}{2})$$
$$= \{(x_{k+1} + 1)\}^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$$
$$= (x_{k+1})^2 + 2x_{k+1} + 1 + (y_{k+1})^2 - (y_{k+1}) + \frac{1}{4} - r^2$$

Now, $p_{k+1} - p_k = 2x_{k+1} + (y^2_{k+1} - y_k^2) - (y_{k+1} - y_k) + 1$

i.e. $p_{k+1} = p_k + 2x_{k+1} + (y^2_{k+1} - y_k^2) - (y_{k+1} - y_k) + 1$

жишээ нь p_{k+1} □ p_k □ $2x_{k+1}$ □ ($y^2_{k\square1}$ □ y^2_k) □ (y_{k+1} □ y_k) □1

$y_{k\square1}$ нь y_k эсвэл y_k □1 байх эсэх нь p_k –ийн тэмдэгээс хамаарах ба $x_{k\square1}$ □ x_k □1 байна.

Хэрэв p_k сөрөг бол, $y_{k\square1} = y_k$ иймд бид

$p_{k\square1}$ □ p_k □ $2x_{k\square1}$ □1 авн

Хэрэв p_k эерэг бол,

$y_{k\square1} = y_k \square 1,$

$p_{k\square1} \square p_k \square 2x_{k\square1} \square 1 \square 2y_{k\square1}$

Where $2x_{k\square1} \square 2x_k \square$

$2\ 2\ y_{k\square1} \square 2y$

At the start position, (0,r), these two terms have the values 0 an 2r, respectively. Each successive values are obtained by adding 2 to the previous value of 2x and subtracting 2 from previous value of 2y.

The initial decision parameter is obtained by evaluating the circle function at starting position $(x_0, y_0) \square (0, r)$.

$$p_0 = f_{circle}(1, r - 1/2)$$
$$= 1 + (r - 1/2)^2 - r^2$$
$$= 1 + r^2 - r + \frac{1}{4} - r^2$$
$$= \frac{5}{4} - r$$

p_0 бүхэл тоогоор тодорхойлсон бол,

$$p_0 = 1 - r$$

Алгоритм

1. Радиус r болон тойргийн төв (x_c, y_c) оролтод өгөх ба координатын эх дээр төвлөрсөн тойргийн анхны цэгийг гарган авна.

   $(x_0, y_0) \square (0, r).$
2. Анхдагч шийдвэрийн параметрийг тооцоолно.

   $p_0 = 5/4\, r$
3. x_k байршил бүрт $k \square 0$ эхлэх ба дараах туршилтыг гүйцэтгэнэ:

   Хэрэв $p_k \square 0$ тойргийн төв (0,0) дагуух дараагийн цэг нь $(x_k \square 1, y_k)$ байна.

   $p_{k\square1} \square p_k \square 2x_{k\square1} \square 1)$

   Эсрэгээр тойргийн дагуух дараагийн цэг $(x_k \square 1, y_k \square 1)$ байна.

   $p_{k\square1} \square p_k \square 2x_{k\square1} \square 1 \square 2y_{k\square1}$

   $2x_{k\square1} \square 2x_k \square 2$ ба $2y_{k\square1} \square 2y_k \square 2.$
4. Бусад долоон октант дээр тэгш хэмийг тодорхойлно.
5. (x_c, y_c) дээр төвлөрсөн тойргийн зам дагуух цэгийн байрлал тус бүрт шилжиж, тооцоолол хийнэ

   $x \square x \square x_c, y \square y \square y_c$
6. $x \square y$ хүртэл 3-с 5 алхамыг давтана.

С хэл дээрх дундаж цэгийн тойргийн алгоритмын програм

//mid point тойргийн алгоритм

```
#include <graphics.h> #include
<stdlib.h> #include <stdio.h>
#include <conio.h>
void drawpoints(int,int,int,int); void
drawcircle(int,int,int);

void main(void)
{
      /* request auto detection */
      int gdriver = DETECT, gmode, errorcode; int xc,yc,r;
    /* initialize graphics and local variables */
      initgraph(&gdriver, &gmode, "\\tc\\bgi");

      /* read result of initialization */ errorcode =
      graphresult();
      if (errorcode != grOk)                    /* an error
            occurred */
      {
            printf("Graphics error: %s\n", grapherrormsg(errorcode)); printf("Press any key to
            halt:");
            getch();
            exit(1); /* terminate with an error code */
      }
    printf("Enter the center co-ordinates:");
    scanf("%d%d",&xc,&yc);
    printf("Enter the radius");
    scanf("%d",&r); drawcircle(xc,yc,r);
    getch();
    closegraph();
}
void drawpoints(int x,int y, int xc,int yc)
{
            putpixel(xc+x,yc+y,1);
            putpixel(xc-x,yc+y,1);
            putpixel(xc+x,yc-y,1);
            putpixel(xc-x,yc-y,1);
            putpixel(xc+y,yc+x,1);
            putpixel(xc-y,yc+x,1);
            putpixel(xc+y,yc-x,1);
            putpixel(xc-y,yc-x,1);
}
void drawcircle(int xc,int yc,int r)
{
            int p,x,y; x=0;
            y=r; drawpoints(x,y,xc,yc);
```

```
        p=1-r;
        while(x<y)
        {
                if(p<0)
                {
                        x=x+1; p=p+2*x+1;

                }
                else
                {       x=x+1;
                        y=y-1;
                        p=p+2*(x-y)+1;

                }
                drawpoints(x,y,xc,yc);
        }
}
```

Даалгавар:

1. Тойрог зурах алгоритмыг судлах, тойрог ашиглан объект байгуулна уу.
2. Алгоритм хоорондын ялгааг тайлбарлана уу.