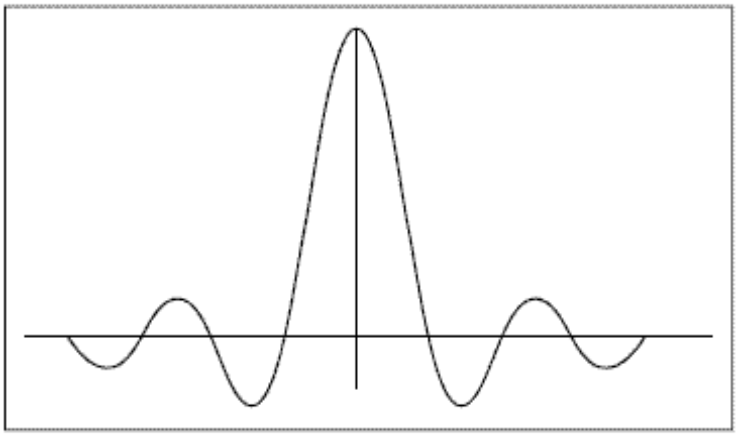
تبدیل پنجره به بندردید

مقدمه

ترسیمهائی که تاکنون نشان دادیم همگی در مختصات پنجرهٔ نمایشگر (بر حسب پیکسل) انجام گردید، که به آن **مختصات صفحه** یا **دستگاه** نیز گفته می شود. در دنیای واقعی با واحدهای مختلفی همانند متر، آنگستروم، میکرن، سال نوری، ... سروکار داریم. دستگاهی که در آن مختصات اشیاء جهان نسبت به آن بیان می شود **دستگاه مختصات جهان**[1] نام دارد. در یک کاربرد داده شده علاقمند هستیم که در دستگاه مختصات جهان فکر کنیم نه دستگاه متخصات پنجره. بطور مثال: نمایش رفتار یک منحنی بین -100<x<100 و -10<y<5.

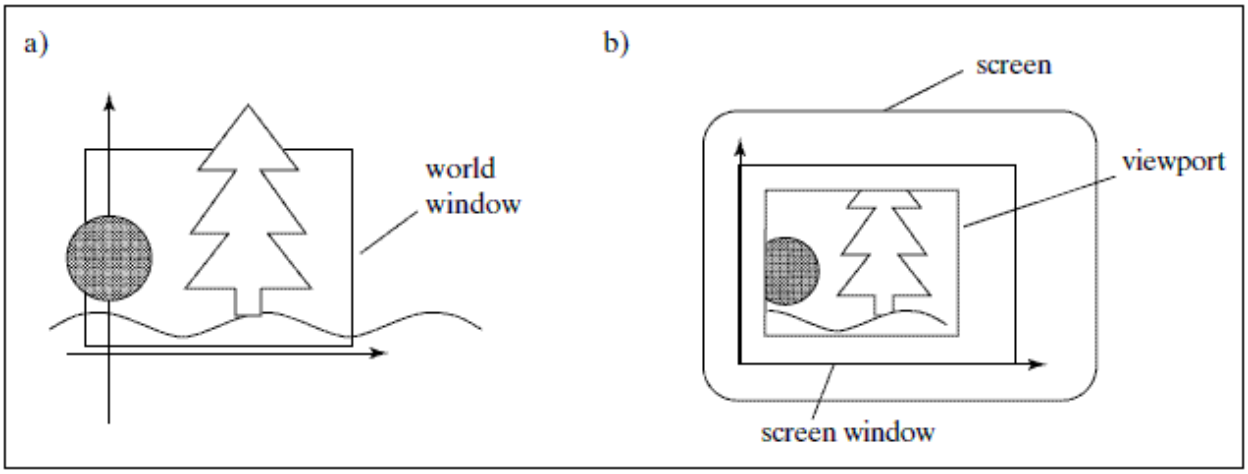
$$\sin c(x) = \frac{\sin(\pi x)}{\pi x} \qquad x \neq 0$$
$$\sin c(x) = 1 \qquad x = 0$$

نمودار این تابع در شکل مشاهده می شود. این تابع دارای مقادیر منفی است و حداکثر مقدار آن نیز یک می باشد. ضمناً مقادیر x نیز می توانند منفی باشد. اما در پنجرهٔ نمایش مختصات منفی وجود ندارد و تغییرات مقدار تابع بین صفر و یک نیز قابل نمایش نیست. بنابر این نگاشتی لازم است تا بتوان تابع را در محدودهٔ دلخواه در بخشی از پنجره که علاقمند به نمایش تابع در آن هستیم ترسیم نمائیم.



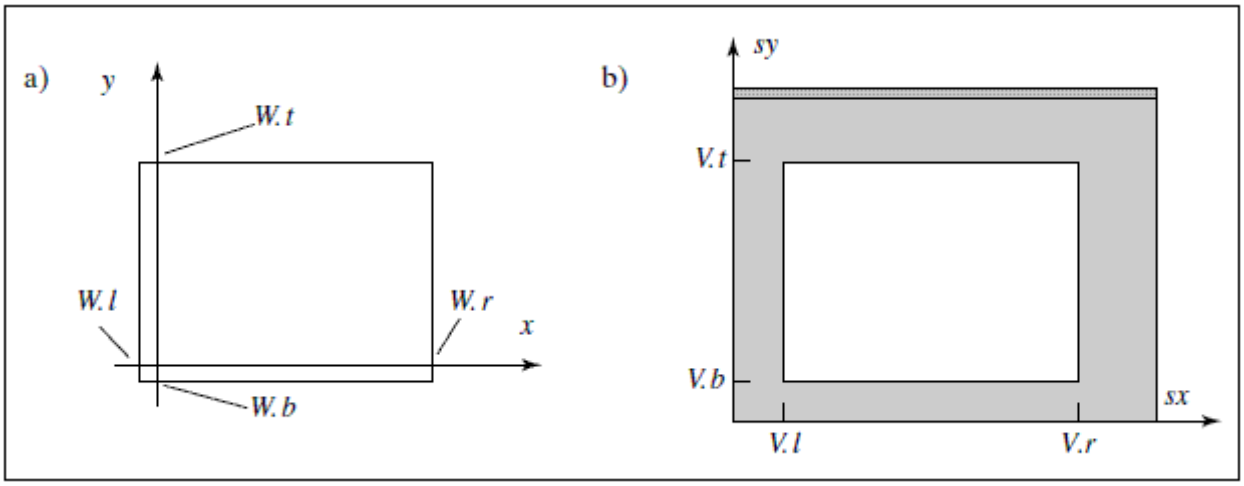
[1] world coordinates

جهت انجام این کار، معمولاً یک پنجره در جهان در نظر گرفته (پنجرهٔ جهان)، و یک پنجره در دستگاه نمایش (بندردید). پنجرهٔ جهان بخشی از صحنه ای که مایل به نمایش آن هستیم را در بر می گیرد. هر آنچه خارج از این پنجره باشد برش داده شده و ترسیم نخواهد شد. بندردید نیز بخشی از پنجرهٔ صفحهٔ نمایش است که اشیاء باید درون آن نمایش داده شوند.



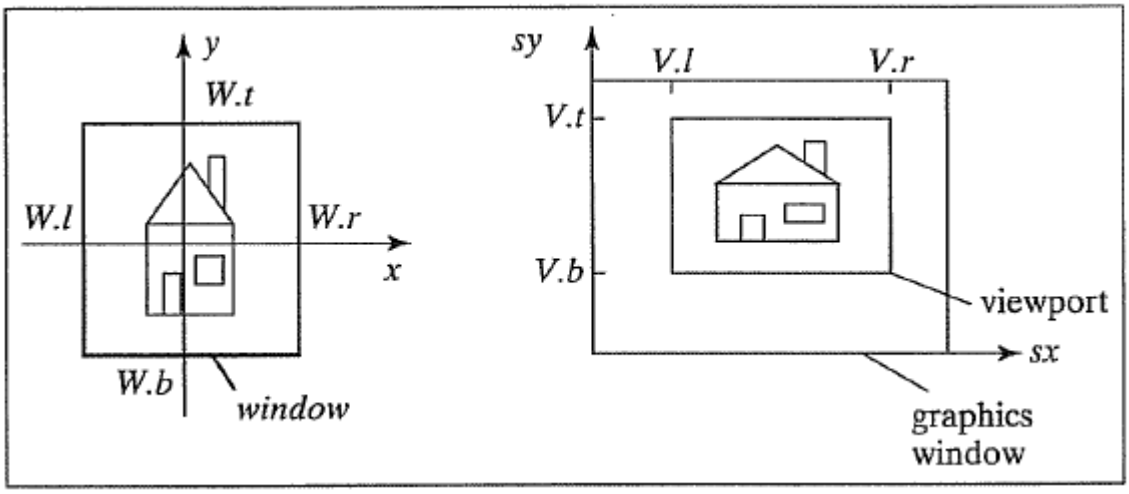
تبدیل پنجرهٔ جهان به بندردید

.

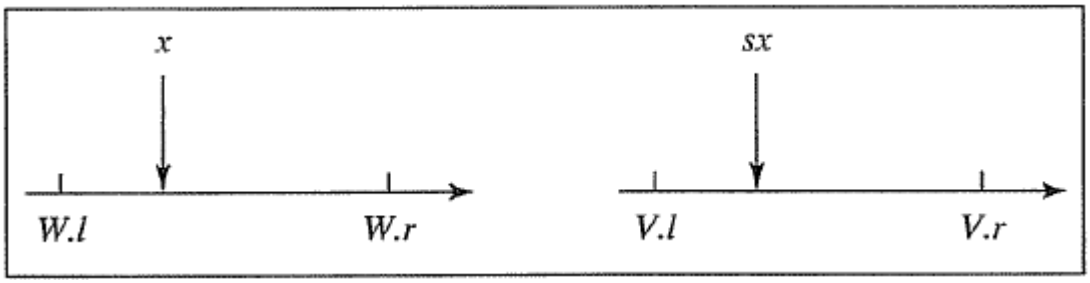


شکل 1

لازم نیست نسبت طول به عرض در پنجرهٔ جهان و در بندردید یکسان باشند. اگر نباشند مقداری اعوجاج حاصل می شود.



شکل 2 تبدیل پنجره به بندردید که داراي مقداري اعوجاج مي باشد.



شکل 3 تناسب تصویر x به sx.

می خواهیم بدانیم نقطهٔ (sx,sy) در بندردید متناظر کدام نقطهٔ (x,y) در پنجرهٔ جهان است. اگر x در فاصلهٔ 40٪ از سمت چپ، sx نیز در فاصلهٔ 40٪ از سمت چپ. بطور مشابه برای y و sy

نگاشت خطی بصورت:

$$sx = Ax + C$$
$$sy = By + D$$

می خواهیم:

$$\frac{sx - V.l}{V.r - V.l} = \frac{x - W.l}{W.r - W.l}$$

یا:

$$sx = \frac{V.r - V.l}{W.r - W.l}x + (V.l - \frac{V.r - V.l}{W.r - W.l}W.l)$$

بطور مشابه:

$$sy = \frac{V.t - V.b}{W.t - W.b} y - \frac{V.t - V.b}{W.t - W.b} W.b$$

این محاسبات را OpenGL بصورت خودکار انجام می دهد. برای ترسیم دوبعدی پنجرهٔ جهان با تابع gluOrtho2D()

gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);

پنجرهٔ دید با تابع glViewport()

glViewport(GLint x, GLint y, GLint width, GLint height);

دقت شود که در دستور قبل پهنا و ارتفاع آرگومانهای سوم و چهارم هستند

x و y نقطهٔ پائین- چپ بندردید

کد لازم در برنامه:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0, 2.0, 0.0, 1.0);
glViewPort(40, 60, 360, 240);
```

هرگاه دستور ترسیم همانند glVertex2*(x,y) صادر شود، (x,y) را در مختصات جهان داده و خود OpenGL تبدیلهای لازم را برای نمایش در بندردید انجام می دهد.

```
 //--------------setWindow-----------------
void setWindow (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top)
{
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(left, right, bottom, top);
}
//---------------setViewport ----------------
void setViewport(GLint left, GLint right, GLint bottom, GLint top)
{
      glViewport(left, bottom, right - left, top - bottom);
}
```

شکل 4

.

یادآوری برنامهٔ اوّل

در main()

```
glutInitWindowSize(640,480)
```

در myInit()

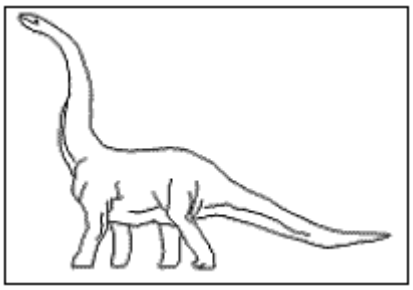
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0, 640.0, 0.0, 480.0);
```

چون دستور glViewport() استفاده نشده، بندردید پیش فرض استفاده می شود. بندردید پیش فرض کل پنجرهٔ نمایش است.

تعریف بندردید در OpenGL (glViewport())

پیش فرض بندردید

در برنامهٔ رسم دایناسور ، مختصات خطوط در پروندهٔ همگی در محدودهٔ (0,0) و (640,480) تعریف شده بودند و مشکلی نبود.



با برخی تغییرات اشکال جالبی می توان بدست آورد. فرض تکرار دایناسور: کاشیکاری (tiling)

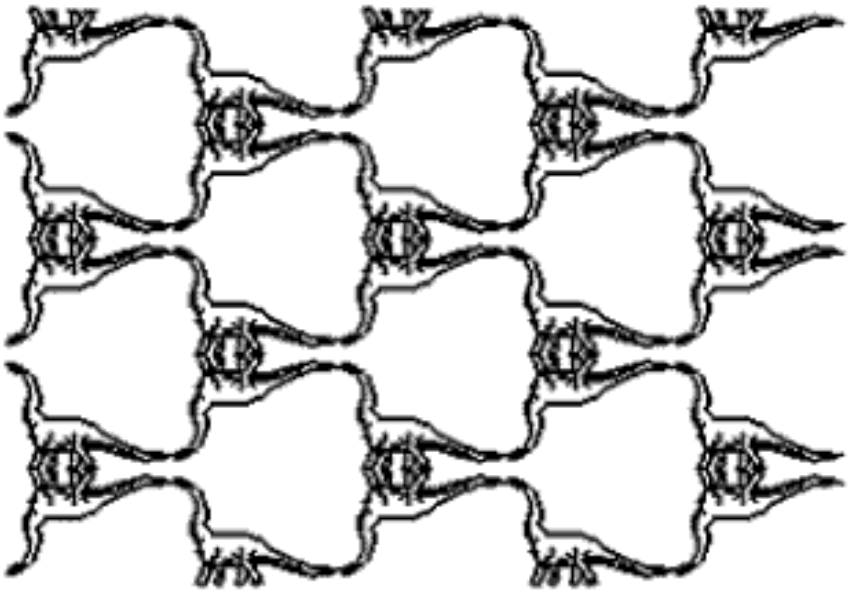
a)

```
setWindow(0.0, 640.0, 0.0, 440.0);
for (int i=0; i<5; i++)
        for (int j=0; j<5; j++){
                glViewport(i*64, j*44, 64, 44);
                  drawPolylineFile(“dino.dat”);
        }
```

.

.

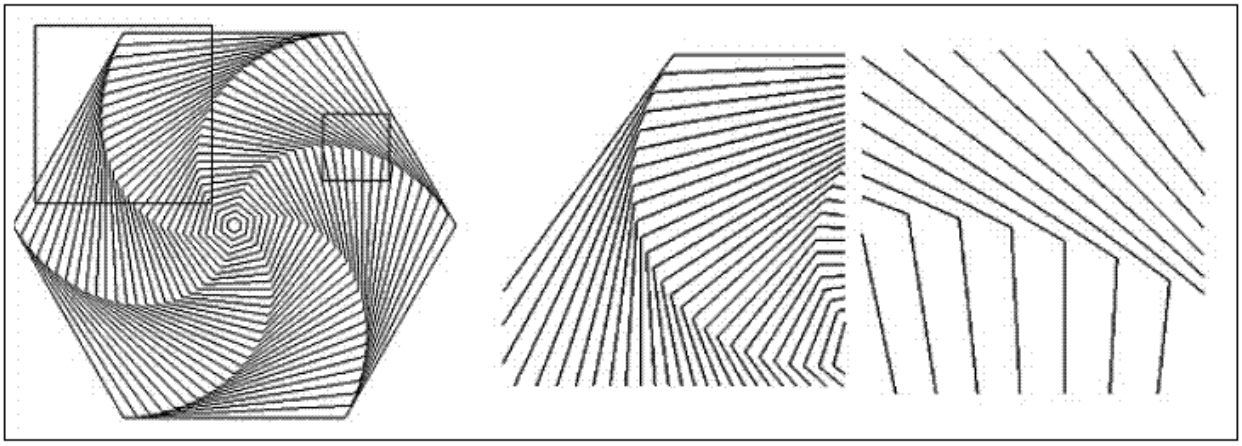
b)



```
for (int i=0; i<5; i++)
      for (int j=0; j<5; j++){
            if ((i+j)%2 ==1)
                  setWindow(0.0, 640.0, 0.0, 440.0);
            else
                   setWindow(0.0, 640.0, 440.0, 0.0);
             glViewport(i*64, j*44, 64, 44);
             drawPolylineFile("dino.dat");
      }
```

برش بخشی از شکل

OpenGL بصورت خودکار بخشهائی از اشیاء که خارج از پنجرهٔ جهان قرار می گیرند را برش می دهد.



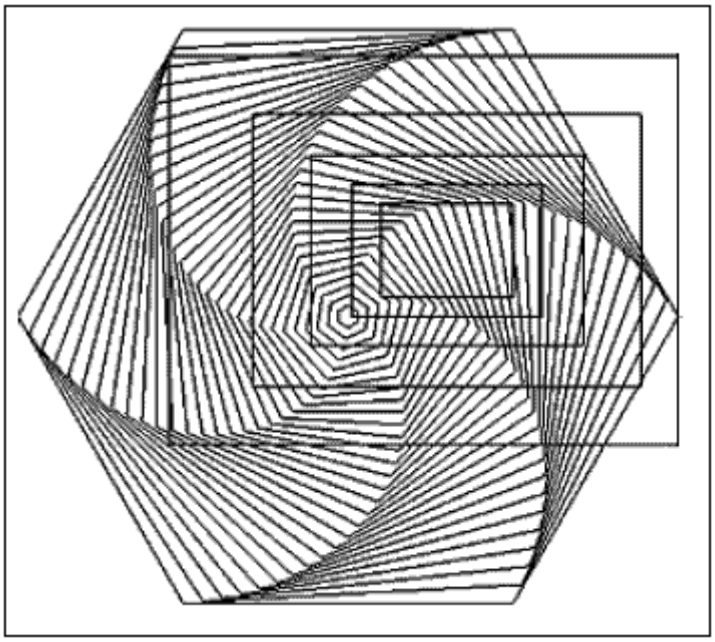
```
setWindow(…);
setViewport(…);
hexSwirl();
```

با ثابت گرفتن بندردید هر چه پنجره کوچکتر گرفته شود، شکل باید کشیده شود تا بندردید را بپوشاند. همانند zoom in کردن.

برزگتر کردن پنجره مشابه zoom out کردن. با حرکت دادن پنجره می توان روی شکل لغزید.

بزرگنمائی روی تصویر با پویانمائی

نمایش یک سری از تصاویر با مرتباً مقداری کوچکتر کردن پنجره



```
float    cx=0.3, cy=0.2;                    //مرکز پنجره
float    H,W=1.2, aspect = 0.7;      //خواص پنجره
نشاندن بندردید
for (int frame=0; frame<NumFrames; fram++){
      پاک کردن صفحه       //پاک کردن شکل قبلي
      W *= 0.7;              //کاهش پهناي پنجره
      H = W/aspect;        //نگهداري نسبت مشابه
      setWindow(cx-W, cx+W, cy-H, cy+H);    //نشاندن پنجرة بعدي
      hexSwirl();
}
```

در روش قبل کاربر مرتباً مراحل زیر را مشاهده خواهد کرد:

- پاک کردن آنی شکل فعلی
- احتمالاً ترسیم یک شکل جدید

می تواند کند و آزاردهنده باشد.

علاقمند هستیم به:

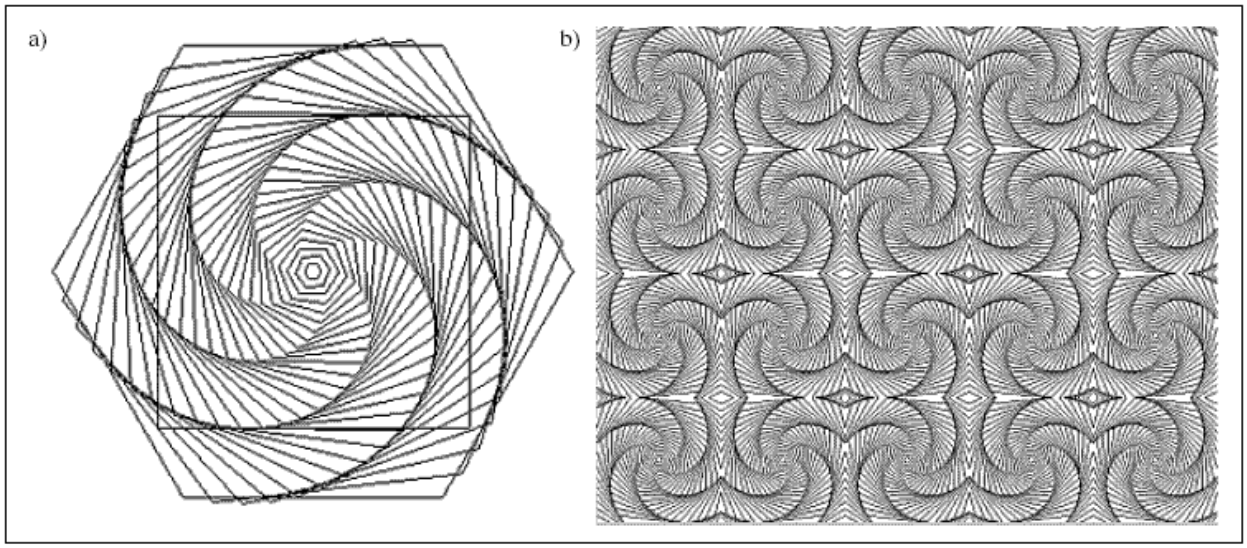
- نمایش دائم شکل فعلی
- جایگزینی یکبارهٔ شکل فعلی با یک شکل جدید

استفاده از روش میانگیری دوگانه. یک میانگیر نمایش داده می شود. ترسیم شکل جدید روی میانگیری که نمایش داده نمی شود

استفاده از دستور glutSwapBuffers() برای معاوضهٔ میانگیرها

برای اطلاع به OpenGL که میانگیر دیگری را رزرو نماید استفاده از GLUT_DOUBLE بجای GLUT_SINGLE

gultInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)

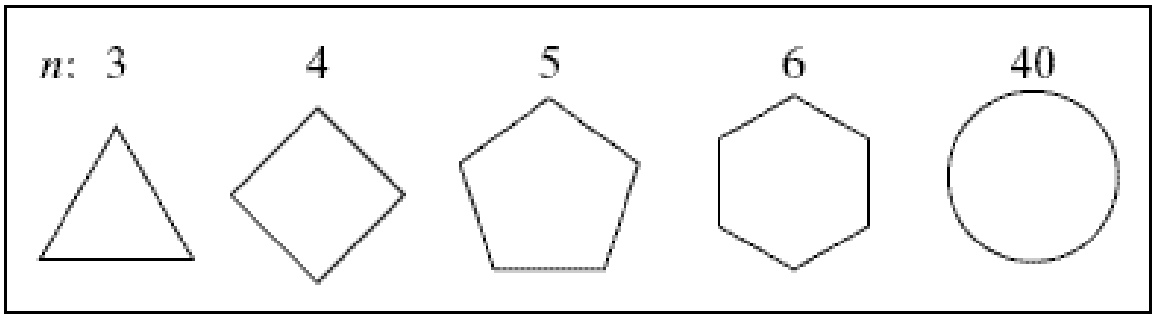


کد بدون قسمت معکوس شده

```
Void myDisplay(){
        پاک کردن پنجره
    setWindow(-0.6, 0.6, -0.6, 0.6);        // بخشي از شكل براي نمايش
     for (int i=0; i<5; i++)
            for (int j=0; j<4; j++){
                    int L=80;          //ميزان جابجائي هر بندرديد
                     setViewport(i*L, i*L+L, j*L, j*L+L);
                     hexSwirl();
```

چندضلعی منتظم

چندضلعی که همهٔ اضلاعش برابر و محدب است.

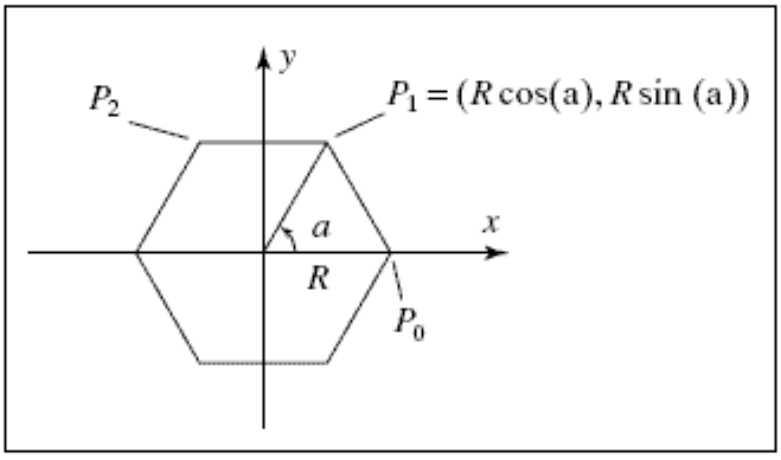


3- منتظم: مثلث

4- منتظم: مربع

n- منتظم

n اگر بزرگ باشد به دایره نزدیک می شویم.

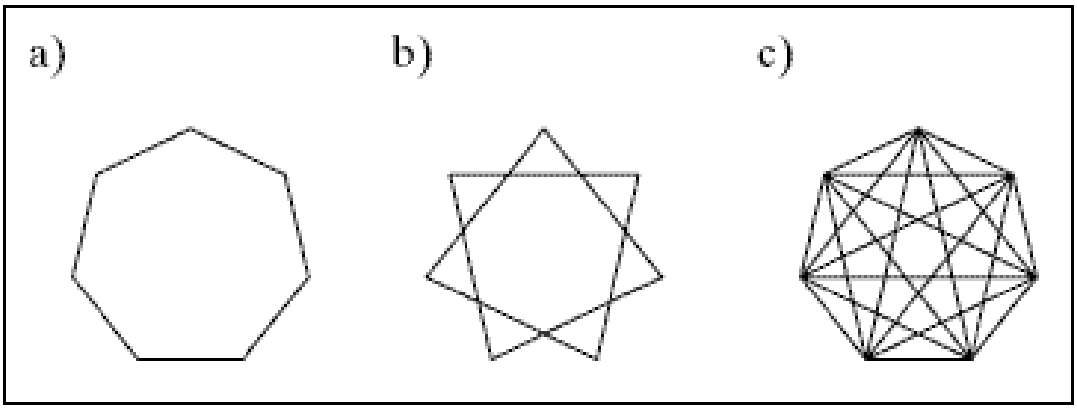


شکل 5 یافتن رئوس یک شش ضلعي منتظم

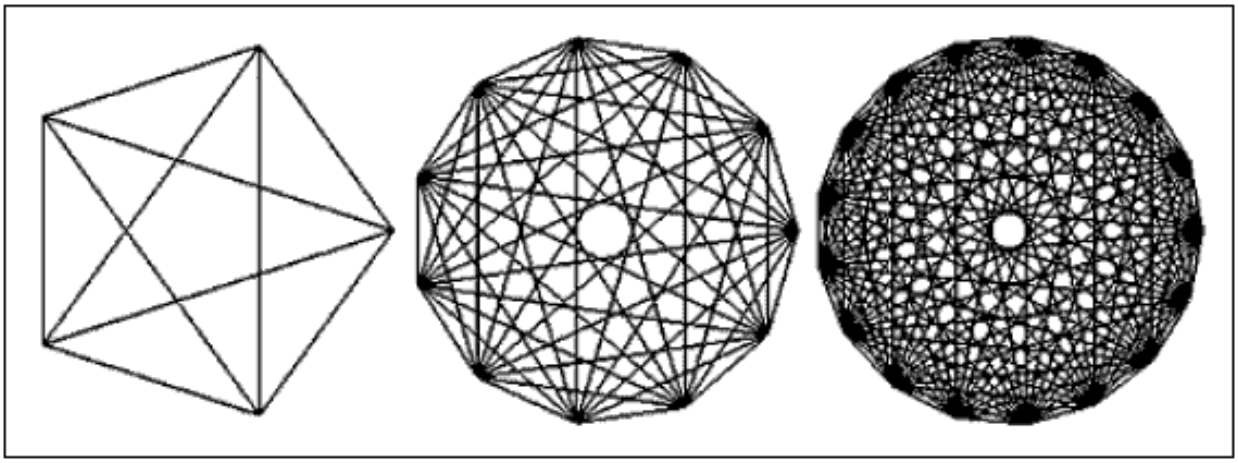
$$P_i = (R\cos(ia), R\sin(ia)) \quad i = 0,\cdots,5$$

R شعاع یک دایرهٔ فرضی

تنوعهای دیگر



Stellation: وصل کردن نقاط یکی در میان. چندضلعی کامل (rosette): وصل کردن هر نقطه به همهٔ نقاط دیگر



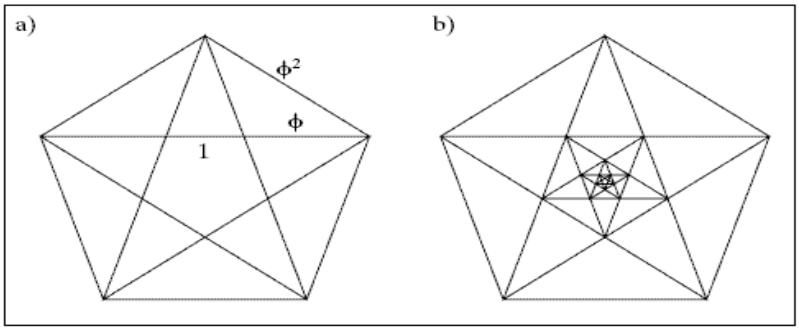
شکل 6 چندضلعیهاي کامل با 5، 11، و 17 رأس

```
// demo program to draw a rosette based on a 5-gon
#include <windows.h>
#include <gl/Gl.h>
#include <gl/Glu.h>
#include <gl/glut.h>
#include <iostream>
#include <math.h>

using namespace std;

// point 2 class
class Point2
{
   public:
      float x, y;
      void set (float dx, float dy){x = dx; y = dy;}
      void set(Point2& p) {x = p.x; y = p.y;}
      Point2(float xx, float yy){x=xx; y=yy;}
      Point2(){x=y=0;}
};

Point2 CP;

void moveTo(Point2 p)
{
      CP.set(p);
}

void lineTo(Point2 p)
{
      glBegin(GL_LINES);
            glVertex2f(CP.x, CP.y);
            glVertex2f(p.x, p.y);
      glEnd();
      glFlush();
      CP.set(p);
}

void myInit(void)
{
      glClear(GL_COLOR_BUFFER_BIT);
      glClearColor( 1.0, 0.0, 0.0, 0.0 );
      // background is red
      glColor3f( 0.0, 0.0, 1.0 );
      // drawing color is blue
}
```

```
void rosette(int N, float radius)
{
      Point2 * pointlist = new Point2[ N ];
      GLfloat theta = ( 2.0f * 3.1415926536 ) / N;

      for( int c = 0; c < N; c++ )
        pointlist[c].set(radius * sin(theta * c), radius * cos(theta * c)
);

      for( int i = 0; i < N; i++ ){
         for( int j = 0; j < N; j++ ){
            moveTo(pointlist[i]);
            lineTo(pointlist[j]);
         }
      }
}

void render()
{
      //this is the callback for displays
      glClear(GL_COLOR_BUFFER_BIT);
      glViewport(10, 10, 640, 480);
      rosette(5, .66f);
      glFlush();
}

void main(int argc, char** argv) {
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      glutInitWindowSize (640,480);
      glutCreateWindow("Rosette");
      glutDisplayFunc( render );//register the callback for display
function
      myInit();
```

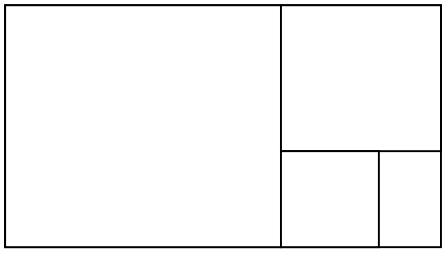
شکل 7 برنامهٔ ترسیم یک پنج ضلعی کامل

چندضلعی کامل 5 تائی شامل چندین نمونه از نسبت طلائی ϕ است.ϕ تقریباً برابر 1.618034 است. نسبت هر پاره خط به پاره خط کوچکتر بعد از خودش برابر ϕ است.



شکل 8

از خواص دیگر ϕ آن است که اگر نسبت اضلاع یک مستطیل برابر ϕ باشد، با جدا کردن یک مربع از آن مستطیل باز هم مستطیلی به نسبت ϕ باقی می ماند.



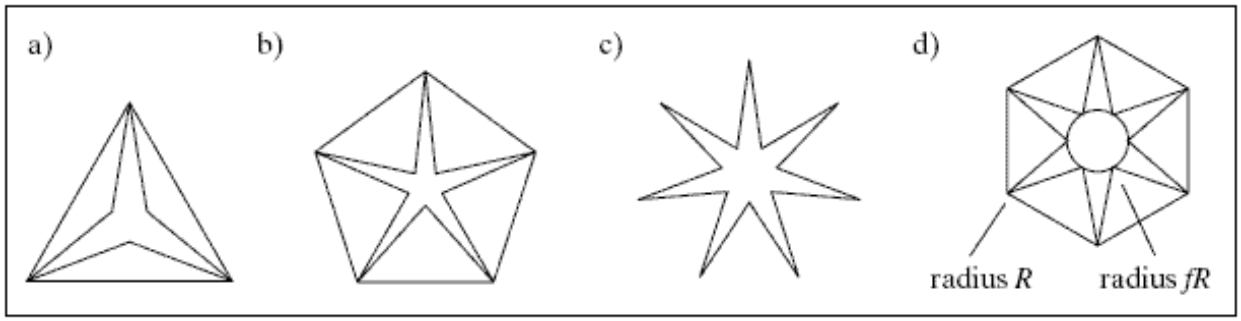
و

$$\phi = 1 + \frac{1}{\phi}$$

$$\phi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \ldots}}}$$

$$\phi = \sqrt{1 + \sqrt{1 + \sqrt{1 + \ldots}}}$$

دو چندضلعی کامل که دایرهٔ مادرشان هم مرکز ویکی دارای شعاع R و دیگری fR است. هر شکل متناوباً یک رأس از دایره کوچکتر و بعد یک رأس از دایرهٔ بزرگتر انتخاب می کند.



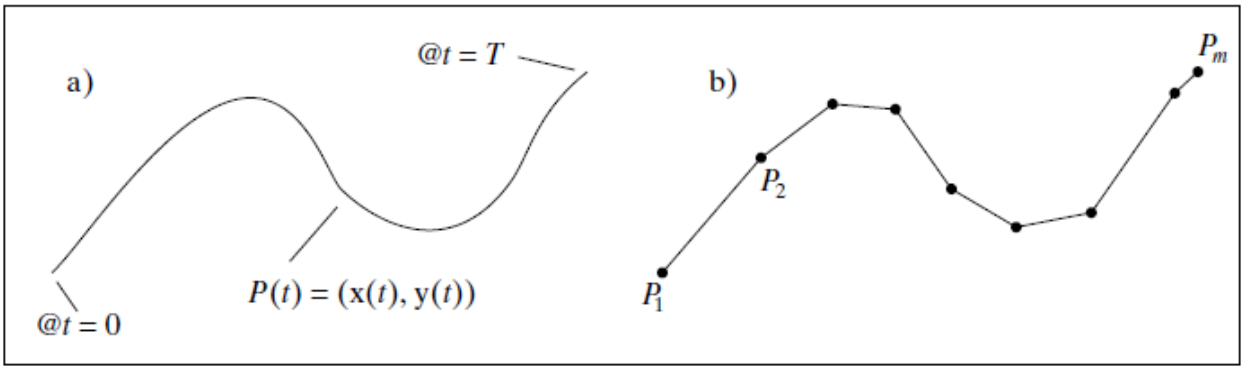
شکل 9

رسم منحنیها بصورت پارامتری

اگر نمایش پارامتری یک منحنی وجود داشته باشد رسم آن سرراست است.

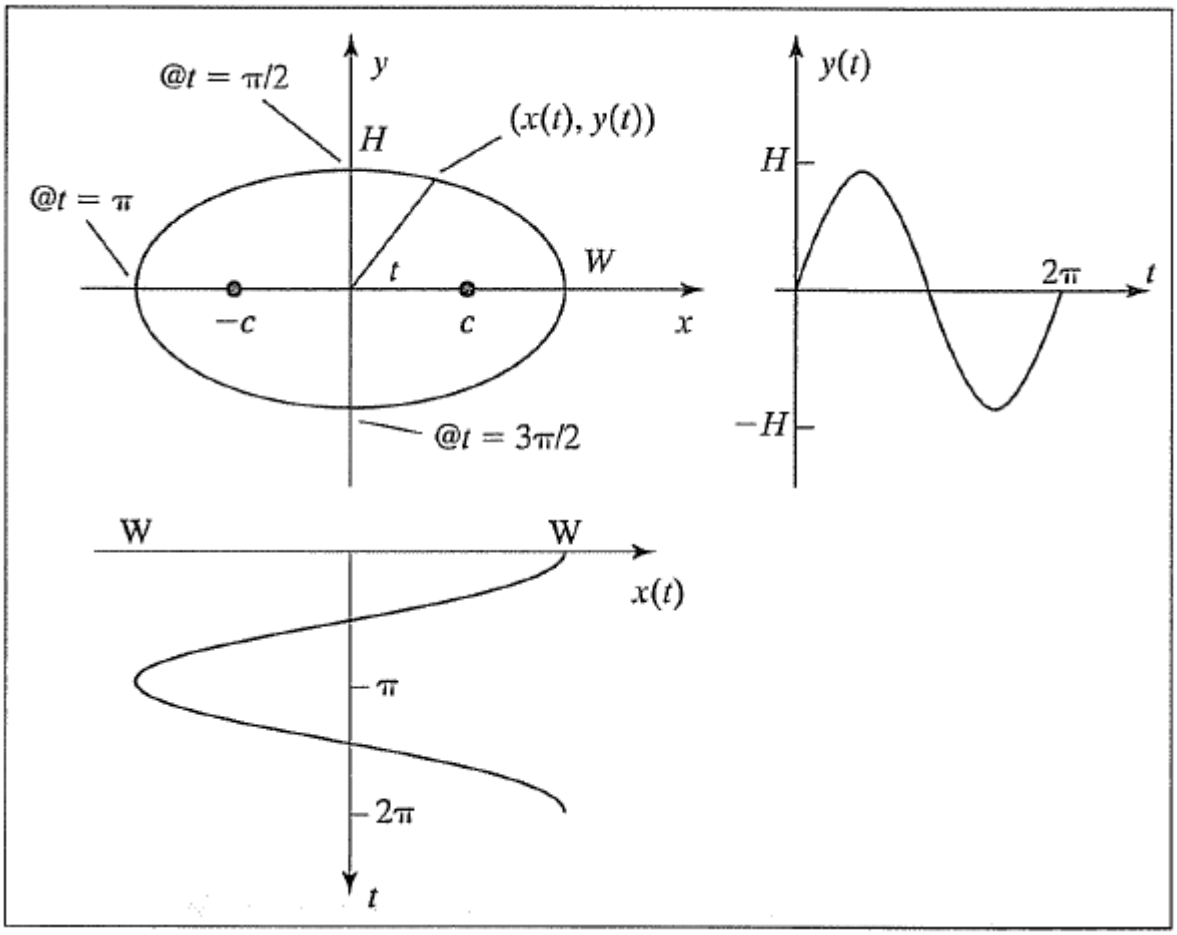
$$P(t)=(x(t),y(t))$$

t بین صفر و T تغییر داده شده و با پاره خط تقریب زده می شود.



شکل 10

.



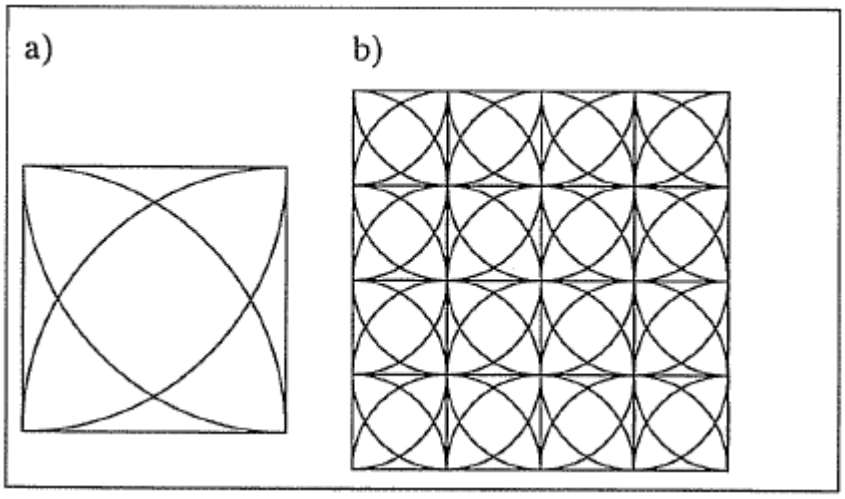
شکل 11 تعریف بیضی بصورت پارامتری.

کد رسم بیضی بصورت پارامتری:

```
#define TWOPI 2*3.14159265
glBegin(GL_LINES);
        for (double t=0; t<=TWOPI; t+=TWOPI/n)
                glVertex2f(W*cos(t), H*sin(t));
glEnd();
```

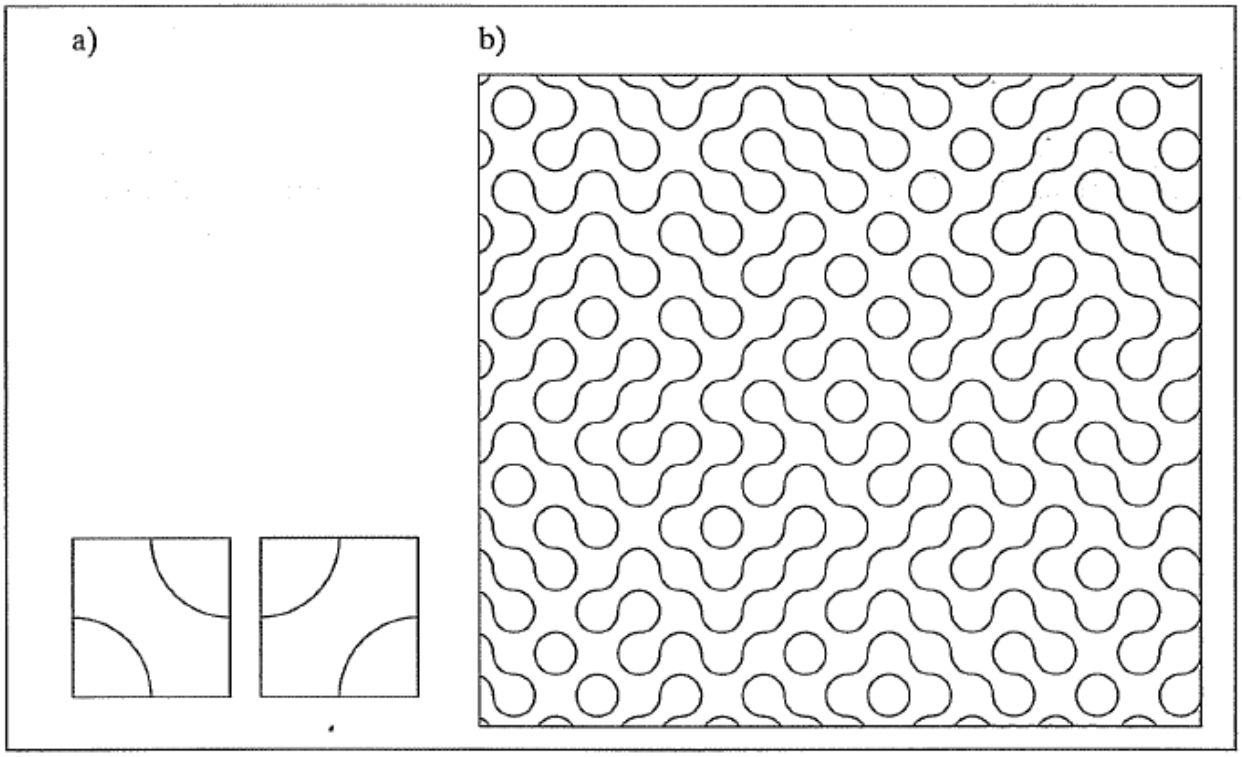
تمرین:

1- برنامه ای بنویسید که الگوی نشان داده شده در شکل 12 الف را بوجود آورده و با استفاده از آن کاشیکاری شکل 12 ب را بوجود آورید.



شکل 12

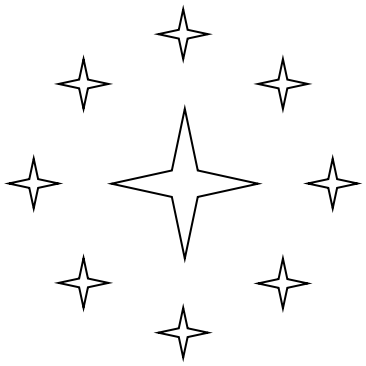
2- روش دیگری برای کاشیکاری روش کاشیکاری تراچت[2] می باشد. در این روش دو الگو وجود دارند که بصورت تصادفی انتخاب شده و در کنار همدیگر قرار می گیرند. برنامه ای بنویسید که با استفاده از کاشیهای نمایش داده شده در شکل 13 الف که از ربع دایره بوجود آمده اند کاشیکاری شکل 13 ب را نمایش دهد.



شکل 13

[2] Truchet

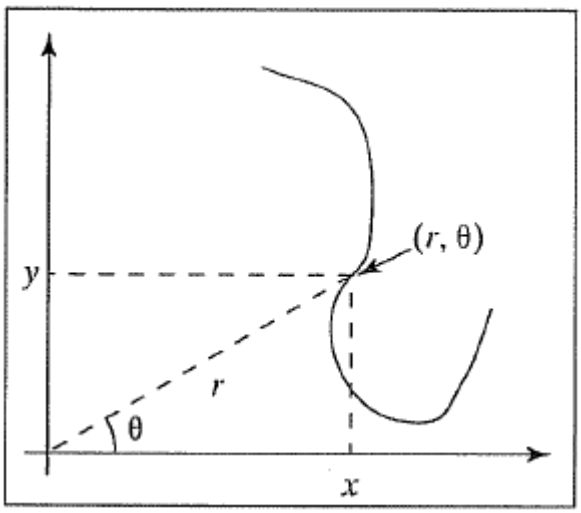
3- با استفاده از نشاندن بندر دید به مکانهای مناسب شکلی همانند شکل 14 را ترسیم نمائید.



شکل 14

4- بسیاری از منحنیها را می توان با استفاده از دستگاه مختصات قطبی ترسیم نمود. هر نقطه از منحنی توسط یک زاویه θ (نسبت به جهت مثبت محور x) و یک فاصله از مبدأ r نمایش داده می شود. اگر زاویه و فاصله هر دو تابعی از پارامتر t می باشند چنانکه t تغییر می کند منحنی $(r(t), \theta(t))$ ترسیم می گردد. در این حالت منحنی دارای مختصات دکارتی بصورت زیر می باشد:

$$x(t) = r(t)\cos(\theta(t))$$
$$y(t) = r(t)\sin(\theta(t))$$

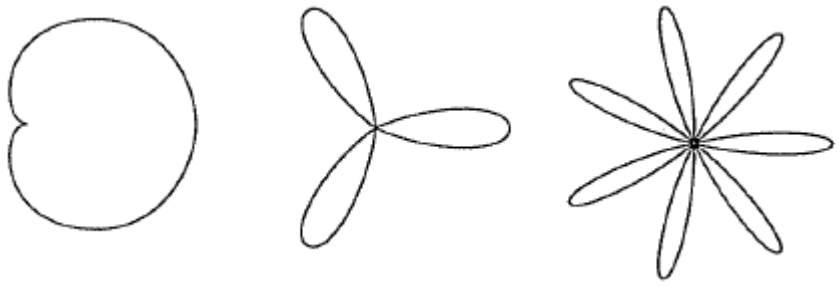


اشکال زیبای بسیاری را می توان در حالتی که r تابعی از θ باشد بدست آورد. در این وضعیت برای هر نقطهٔ (r,θ) مختصات دکارتی آن عبارت است از:

$$x(t) = f(\theta)\cos(\theta)$$
$$y(t) = f(\theta)\cos(\theta)$$

ساده ترین حالت هنگامی است که $f(\theta)=K$ که یک دایره بدست می آید. از دیگر اشکال جالب عبارت است از:

- قلب: $f(\theta) = K(1+\cos(\theta))$
- رز: $f(\theta) = K(\cos(n\theta))$ که n تعداد گلبرگها را نشان می دهد.



برنامه هائی بنوسید که بتوانند اشکال قلب و رز را ترسیم نمایند.