إظهار النقاط والخطوط والمضلعات :

- ترسم النقاط افتراضياً كبكسل وحيد على الشاشة والخط يرسم بعرض بكسل واحد والمضلع يرسم بشكل ممتلئ ومصمت .
- سنقوم بدراسة كيفية تغيير هذه الافتراضيات .

1. تفاصيل النقطة :

- نستخدم الأمر glPointSize() للتحكم بحجم النقطة المنفذة، ونزود الحجم المرغوب بالبكسلات كمعامل للأمر.

الشكل العام للأمر :

```
Void glPointSize(GLfloat size);
```

- علماً أن حجم النقاط يجب أن يكون أكبر من الصفر والقيمة الافتراضية هي الواحد .

2. تفاصيل الخطوط :

- نستطيع مع openGL تعيين خطوط بعرض مختلف وبأنواع مختلفة (منقطة، خطوط مقطعة)

الشكل العام لأمر التحكم بعرض الخط :

```
Void glLineWidth(GLfloat Width);
```

- يحدد عرض الخطوط بالبكسلات و يجب أن يكون أكبر من الصفر والقيمة الافتراضية هي الواحد .

الشكل العام لأمر التحكم بنوع الخط :

```
Void glLineStipple(GLint factor , GLushort pattern);
```

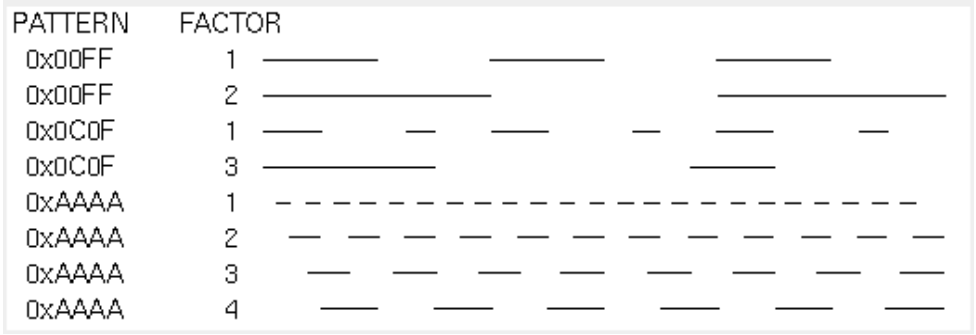
- المعامل pattern عبارة عن سلسلة بطول 16 bit من الأصفار والواحدات . وتكرر حسب الضرورة لتحديد نوع الخط المعطى . القيمة 1 تشير لحدوث رسم و 0 لا ترسم شيء . المعامل Factor يستخدم كعامل ضرب للسلسلة . فإذا تعاقبت ثلاثة واحدات فسوف تمتد إلى ستة واحدات إذا كان factor=2 . (قيمة factor بين 1 و 255). يلغى تفعيل الأمر السابق باستخدام ( )glDisable .

مثال:

```
glLineStipple(1,0x3f07);
glEnable(GL_LINE_STIPPLE);
```

الأمر الأول يحدد نوع الخط والأمر الثاني يفعل نوع الخط

- 0x3f07 ==> 0011111100000111 سيرسم الخط بثلاثة بكسلات on ثم خمسة بكسلات off ثم ستة بكسلات on واثنتين off .
- (البتات الأقل أهمية تستخدم أولاً) ، إذا كان factor=2 ستصبح لدينا ستة بكسلات on وعشرة بكسلات off .



القسم العملي

تطبيق 1: استخدام نماذج خطوط متنوعة

```
#include<GL/glut.h>
#include <stdlib.h>
#include<math.h>

static void redraw(void);
int main(int argc, char **argv);

int main(int argc, char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(400,400);
   glutCreateWindow("draw multi lines");
    glutDisplayFunc(redraw);
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45,1.0,10.0,200.0);
    glMatrixMode(GL_MODELVIEW);
    glutMainLoop();
    return 0;
}
static void redraw(void)
{
    float x1,x2,y1,y2;
    int i;
    //تعريف تابع لرسم الخطوط
    #define drawOneLine(x1,y1,x2,y2) glBegin(GL_LINES);\
    glVertex2f ((x1),(y1)); glVertex2f ((x2),(y2)); glEnd();
    glClearColor(1.0,1.0,1.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0.0f,0.0f,-100.0f);
    /*  رسم جميع الخطوط باللون الأسود  */
   glColor3f (0.0, 0.0, 0.0);
   glEnable (GL_LINE_STIPPLE);

    glTranslatef(0.0f,0.0f,0.0f);
    glLineWidth (5.0);//5 عرض الخط
      glLineStipple (1, 0x0101);
      drawOneLine (-5.0, -10.0, 10.0, -10.0);
```

```
    glLineStipple (1, 0x00FF);
    drawOneLine (-5.0, -5.0, 10.0, -5.0);
    glLineStipple (1, 0x1C47);
    drawOneLine (-5.0, 0.0, 10.0, 0.0);
    glLineWidth (1.0);
  glutSwapBuffers();
 }
```

- **تعديل: يطلب تعديل ألوان وأنماط الخطوط**

تخصيص نمط تظليل

يمكن رسم الخط أو المضلع المعبأ بلون واحد(flat shading) ، أو باستخدام عدة ألوان مختلفة (smooth shading/Gouraud)

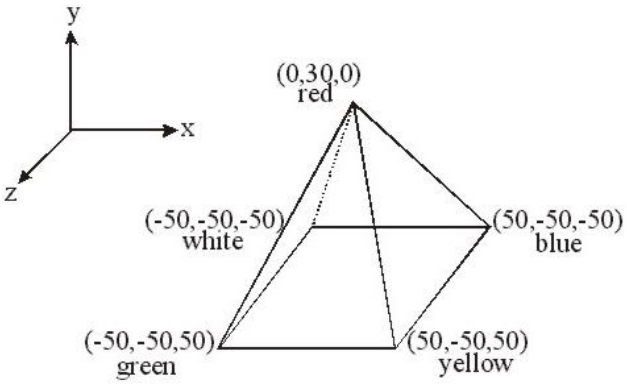
نقوم بتحديد تقنية التظليل المرغوبة باستخدام الأمر:

```
Void glShadeModel (GLenum mode);
```

با رامتر النمط يمكن أن يكون GL_SMOOTH (الافتراضي) أو GL_FLAT.

تطبيق 2: رسم هرم

```
glBegin(GL_TRIANGLE_FAN);  // draw triangle
      glVertex3f(  0.0f,  30.0f, 0.0f);
      glVertex3f(-50.0f, -50.0f, 50.0f);
      glVertex3f( 50.0f, -50.0f, 50.0f);
      glVertex3f( 50.0f, -50.0f, -50.0f);
      glVertex3f( -50.0f, -50.0f, -50.0f);
      glVertex3f(-50.0f, -50.0f, 50.0f);
glEnd();
```



تعديلات:

- اجعل كل نقطة من نقاط الهرم السابق بلون مختلف.
- استخدم التعليمة الخاصة بتخصيص نمط التظليل باستخدام الثابت GL_SMOOTH ثم باستخدام الثابت GL_FLAT .
- استخدم الثابت GL_DEPTH_TEST لإخفاء النقاط التي يجب أن لا تظهر.

تطبيق 3: خوارزمية Midpoint لرسم مستقيم ودائرة وقطع ناقص:

رسم مستقيم باستخدام خوارزمية midpoint

1- Input **(***x0,y0,x1,y1)*
2- Calculate
*dx = x1-x0 , dy = y1-y0 , d= 2*dy-dx ,*
*ΔE = 2*dy ,* ΔNE *= 2*(dy-dx)*
x= x0 , y=y0
3- Drawpixel **(***x,y)*
4- If *d < 0 then d=d+ ΔE , x = x+1 ,*
else *d=d+* ΔN*E , x=x+1, y=y+1*
5- Repeat 3-4 until *x>= x0*

رسم دائرة باستخدام خوارزمية midpoint

1. Input radius r and circle centre (x_c,y_c), and obtain the first point on the circumference of a circle centered on the origin as $(x_o,y_o) = (0, r)$.
2. Calculate the initial value of the decision parameter as $p_o = 5/4 - r$
3. At each x_i position, starting at i=0, perform the following test: if $p_i < 0$, the next point along the circle centered on (0,0) is (x_i+1, y_i) and

$$p_{i+1} = p_i + 2x_{i+1} + 1$$

otherwise, the next point along the circle is (x_i+1, y_i-1) and

$$p_{i+1} = p_i + 2x_{i+1} + 1 - 2y_{i+1}$$

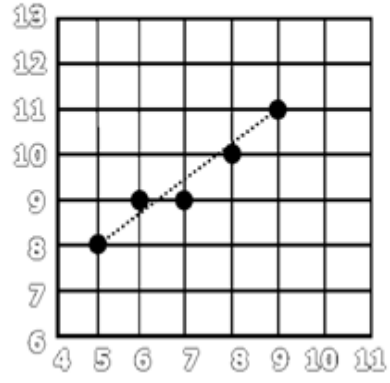
4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x,y) onto the circular path centered on (x_c,y_c) and plot the coordinate values. $x = x + x_c$, $y = y + y_c$
6. Repeat steps 3 through 5 until x >= y.

Example:

Starting point: (5, 8)
Ending point: (9, 11)

INIT: dy = 3; dx = 4; d_{start}=2;
$(\Delta d)_E$ = 6; $(\Delta d)_{NE}$ = -2;

- **d=2, (6, 9)**
- **d=0, (7, 9)**
- **d=6, (8, 10)**
- **d=4, (9, 11)**



مثال:

رسم قطع ناقص باستخدام خوارزمية midpoint

1. **Input r_x, r_y, and ellipse center (x_c, y_c), and obtain the first point on an ellipse centered on the origin as $(x_0, y_0) = (0, r_y)$**
2. **Calculate the initial parameter in region 1 as** $p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$
3. **At each x_i position, starting at $i = 0$, if $p1_i < 0$, the next point along the ellipse centered on (0, 0) is $(x_i + 1, y_i)$ and** $p1_{i+1} = p1_i + 2r_y^2 x_{i+1} + r_y^2$

   otherwise, the next point is $(x_i + 1, y_i - 1)$ and $p1_{i+1} = p1_i + 2r_y^2 x_{i+1} - 2r_x^2 y_{i+1} + r_y^2$

   and continue until $2r_y^2 x \geq 2r_x^2 y$

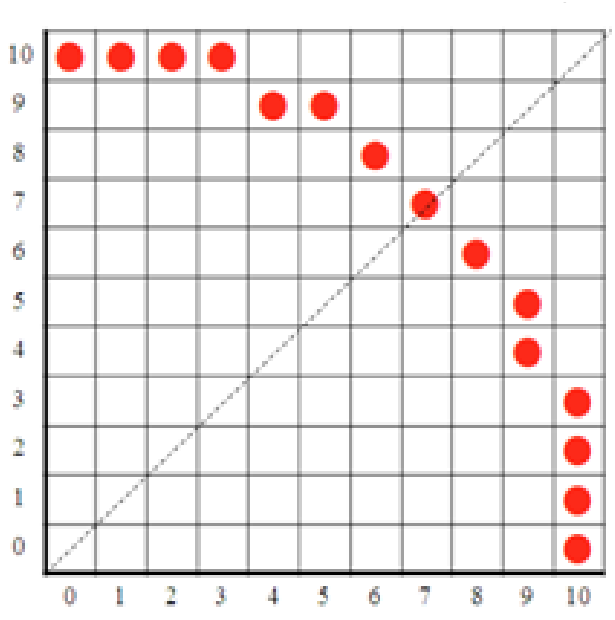
$r = 10$

$p_0 = 1 - r = -9$

Initial point $(x_0, y_0) = (0, 10)$

i	p_i	x_{i+1}, y_{i+1}	$2x_{i+1}$	$2y_{i+1}$
0	-9	(1, 10)	2	20
1	-6	(2, 10)	4	20
2	-1	(3, 10)	6	20
3	6	(4, 9)	8	18
4	-3	(5, 9)	10	18
5	8	(6, 8)	12	16
6	5	(7, 7)		

4. **(x_0, y_0) is the last position calculated in region 1. Calculate the initial parameter in region 2 as** $p2_0 = r_y^2(x_0 + \frac{1}{2})^2 + r_x^2(y_0 - 1)^2 - r_x^2 r_y^2$



5. **At each y_i position, starting at $i = 0$, if $p2_i > 0$, the next point along the ellipse centered on (0, 0) is $(x_i, y_i - 1)$ and** $p2_{i+1} = p2_i - 2r_x^2 y_{i+1} + r_x^2$

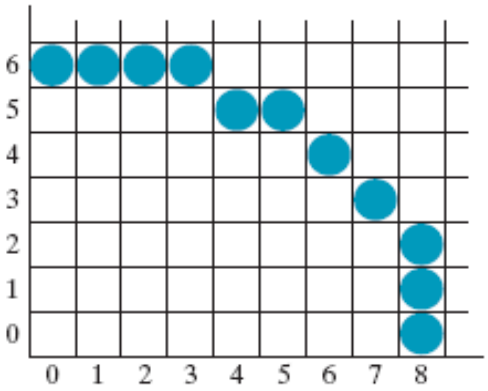
   otherwise, the next point is $(x_i + 1, y_i - 1)$ and $p2_{i+1} = p2_i + 2r_y^2 x_{i+1} - 2r_x^2 y_{i+1} + r_x^2$

   Use the same incremental calculations as in region 1. Continue until $y = 0$.

6. **For both regions determine symmetry points in the other three quadrants.**
7. **Move each calculated pixel position (x, y) onto the elliptical path centered on (x_c, y_c) and plot the coordinate values $x = x + x_c$, $y = y + y_c$**

مثال:

$r_x = 8$
$r_y = 6$
$2r_y^2 x = 0$ with increment $2r_y^2 = 72$
$2r_x^2 y = 2r_x^2 r_y$ with increment $-2r_x^2 = -128$



For region 1:

$(x_0, y_0) = (0, 6)$

$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2 = -332$

k	$p1_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	−332	(1, 6)	72	768
1	−224	(2, 6)	144	768
2	−44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	−108	(5, 5)	360	640
5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

For region 2:

$(x_0, y_0) = (7, 3)$

$p2_0 = \text{fellipse}\left(7 + \frac{1}{2}, 2\right) = -151$

k	$p1_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	−151	(8, 2)	576	256
1	233	(8, 1)	576	128
2	745	(8, 0)	—	—

مثال:

برنامج لرسم مستقيم ودائرة وقطع ناقص باستخدام خوارزمية midpoint

```
#include <windows.h>
#include <GL/glut.h>
#include <stdlib.h>
#include<math.h>


void drawDot (GLint x, GLint y, GLfloat r, GLfloat g, GLfloat b)
{ glColor3f(r,g,b);
  glBegin (GL_POINTS);
          glVertex2i (x,y);
  glEnd();
}

void myInit(void)
 {
   glClearColor(1.0,1.0,1.0,0.0);  // set white background color
   glColor3f (0.0f,0.0f,0.0f);    //default color
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-640.0, 640.0, -480.0, 480.0);
}


void Line (int x0, int y0, int x1, int y1, float r, float g, float b)
{int x,y,dx,dy,d,j;

  x = x0;
  y = y0;
  dx =  (x1 - x0);
  dy =  (y1 - y0);
  d = 2*dy - dx;
 while (x<=x1)
  {
    drawDot (x,y,r,g,b);
    if(d<=0)
          {
          d+=2*dy;
          x++;
          }
    else
          {
            d+=2*(dy-dx);
```

```
            x++;
            y++;
          }
    }

}

//<<<<<<< draw circles
void circlePoints (int x, int y, int xc, int yc, float r, float g, float b)
{
  drawDot (xc+x,yc+y,r,g,b);
  drawDot (xc-x,yc+y,r,g,b);
  drawDot (xc+x,yc-y,r,g,b);
  drawDot (xc-x,yc-y,r,g,b);
  drawDot (xc+y,yc+x,r,g,b);
  drawDot (xc-y,yc+x,r,g,b);
  drawDot (xc+y,yc-x,r,g,b);
  drawDot (xc-y,yc-x,r,g,b);
}

void Circle (int xc, int yc, int rad, float r, float g, float b)
{ int x,y,d;

  x = 0;
  y = rad;
  circlePoints (x,y,xc,yc,r,g,b);
  d = 1 - rad;
  while (x < y)
  { if (d < 0)
     x++;
    else
    { x++;
      y--;
    }
    if (d < 0)
      d += 2*x +1;
    else
     d += 2*(x-y) + 1;
    circlePoints (x,y,xc,yc,r,g,b);
  }
}

//<<<<<<< draw Ellipse
void symmetricPixels (int x, int y, int xc, int yc, float r, float g, float b)
{ drawDot (xc + x, yc + y, r,g,b);
```

```
  drawDot (xc - x, yc + y,r,g,b);
  drawDot (xc + x, yc - y,r,g,b);
  drawDot (xc - x, yc - y,r,g,b);
}

void Ellipse (int a, int b, int xc, int yc, float r, float g, float bl)
{ int aSq,bSq,twoASq,twoBSq,d,dx,dy,x,y;

  aSq = a*a;
  bSq = b*b;
  twoASq = 2*aSq;
  twoBSq = 2*bSq;
  d = bSq - b*aSq + aSq/4;
  dx = 0;
  dy = twoASq*b;
  x = 0;
  y = b;
  symmetricPixels(x,y,xc,yc,r,g,bl);
  while (dx < dy)
  { x++;
    dx += twoBSq;
    if (d >= 0)
    { y--;
      dy -= twoASq;
    }
    if (d < 0)
     d += bSq + dx;
    else
     d += bSq + dx - dy;
    symmetricPixels (x,y,xc,yc,r,g,bl);
  }
  d = (int)(bSq*(x+0.5)*(x+0.5) + aSq*(y-1)*(y-1) -
              aSq*bSq);
  while (y > 0)
  { y--;
    dy -= twoASq;
    if (d <= 0)
    { x++;
      dx += twoBSq;
    }
    if (d > 0)
        d += aSq - dy;
    else
        d += aSq -dy +dx;
    symmetricPixels(x,y,xc,yc,r,g,bl);
```

```
  }
}

void myDisplay(void)
{
  glClear(GL_COLOR_BUFFER_BIT);     // clear the screen
  // Draw a red line
  Line (45,50,100,120,1,0,0);
  // put black center axes on screen for ellipse and circle
  glColor3f(0.0f,0.0f,0.0f);
  glBegin (GL_LINES);    //one long horizontal line (x axis)
    glVertex2i (30,300);
    glVertex2i (600,300);
  glEnd();
  glBegin(GL_LINES);  // vertical line (ellipse y axis)
    glVertex2i(200,200);
    glVertex2i(200,400);
  glEnd();
  glBegin (GL_LINES);  // vertical line (circle y axis)
    glVertex2i(500,200);
    glVertex2i(500,400);
  glEnd();
  // put blue circle on screen
  Circle (500,300,70,0,0,1);
  // draw a green ellipse on the screen
  Ellipse (150,50,200,300,0,1,0);
  glFlush();
}



int main(int argc, char** argv)
{
      glutInit(&argc, argv);          // initialize the toolkit
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); // set display mode
      glutInitWindowSize(640,480);     // set window size
      glutInitWindowPosition(100, 150); // set window position on screen
      glutCreateWindow("Basic drawing"); // open the screen window
     myInit();
     glutDisplayFunc(myDisplay);     // register redraw function
     glutMainLoop();                  // go into a perpetual loop
     return 0;
}
```

ملاحظة:

يمكن تعديل تابع رسم المستقيم باستخدام خوارزمية Midpoint بحيث يمكن رسمه في كافة الأرباع والأثمان فيصبح التابع كما هو أدناه:

```
int sign (int n)
{ if (n > 0) return 1;
  else if (n < 0 ) return (-1);
  return 0;
}

void Line (int x0, int y0, int x1, int y1, float r, float g, float b)
{ int x,y,dx,dy,d,j,s1,s2,flag=0;

  x = x0;
  y = y0;
  dx = abs (x1 - x0);
  dy = abs (y1 - y0);
  s1 = sign (x1-x0);
  s2 = sign (y1-y0);
  if (dy > dx)
  { // swap (dy,dx);
    int temp = dy;
       dy = dx;
       dx = temp;
    flag = 1;
  }
  d = 2*dy - dx;
  for (j = 1; j <= dx; j++)
  { drawDot (x,y,r,g,b);
    while (d > 0)
    { if (flag == 1)
       x += s1;
      else
       y += s2;
      d -=2*dx;
    }
    if (flag == 1)
      y += s2;
    else
       x += s1;
    d += 2*dy;
  }
}
```