مقارنة بين Open GL و Direct 3D

- يوفر Microsoft Direct3D API (Direct3D) واجهة لوظائف العرض ثلاثي الأبعاد المضمنة في معظم محولات الفيديو الجديدة.
- يتضمن Direct3D دعم مجموعات تعليمات CPU الخاصة، وبذلك يؤمن تسريعاً إضافياً على الحواسيب الأحدث.
- نبين فيما يلي مقارنة بين Open GL و Direct 3D

1. قابلية الحمل:

Open GL	Direct 3D
قابلة للحمل تنفذ على أنظمة مختلفة مثل: Microsoft Windows, Linux, UNIX-based systems, Mac OS X and game consoles by Nintendo and Sony such as the PlayStation 3	غير قابلة للحمل تنفذ على : Microsoft's Windows family of operating systems

2. مجال الاستخدام:

Open GL	Direct 3D
مجال الرسوميات الاحترافية .	مجال ألعاب الكمبيوتر

مثال:

Open GL	Direct 3D
ArchiCAD Autodesk (AutoCAD 2000) Quake (GL-Quake, Quake II & III) Microstation 95 MathGL3D (Mathematica) Maya (Character animation, Modeling)	Simply 3D Unreal Tournament Earth 2150 Warcraft III Microsoft Flight Simulator Battle Isle 4

3. الحقوق والسماحيات:

Open GL	Direct 3D
مفتوحة المصادر Open Source وهناك بطئ في صدور الاصدارات .	تابعة لـ Microsoft وهناك اصدار جديد كل سنة وهذا يؤدي إلى اتعاب المبرمجين في تعلم المزايا الجديدة.

4. الكود البرمجي:

Open GL	Direct 3D
يحتاج إلى القليل من الأسطر البرمجية لعمل شيء ما.	يحتاج إلى الكثير من الأسطر البرمجية. مثلا عملية التهيئة في الاصدار 5 تحتاج 800 سطر برمجي. وفي الاصدار 7 نقصت على 200 سطر

مثال: انشاء مثلث من ثلاثة رؤوس :

Direct 3D	Open GL

(psuedo code, and incomplete)
v = &buffer.vertexes[0];
v->x = 0; v->y = 0; v->z = 0;
v++;
v->x = 1; v->y = 1; v->z = 0;
v++;
v->x = 2; v->y = 0; v->z = 0;
c = &buffer.commands;
c->operation = DRAW_TRIANGLE;
c->vertexes[0] = 0;
c->vertexes[1] = 1;
c->vertexes[2] = 2;
IssueExecuteBuffer (buffer);
	glBegin (GL_TRIANGLES);
	glVertex (0,0,0);
	glVertex (1,1,0);
	glVertex (2,0,0);
	glEnd ();

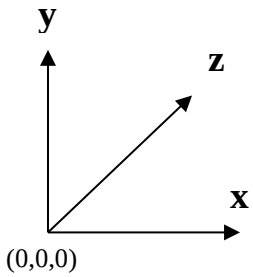
OpenGL كآلة حالة:

- توضع OpenGL في حالات مختلفة تبقى متأثرة بها حتى نقوم بتغييرها.
- مثلاً اللون الحالي متحول حالة يحتفظ بالقيمة التي نسندها إليه (مثلاً أبيض أو أحمر ...) بحيث ترسم كل الكائنات بعد هذه التعليمة باستخدام هذا اللون إلى أن نعطي هذا المتحول لوناً آخر .
- تستخدم متحولات الحالة للتحكم بالمظهر الحالي والنماذج المنقطة والخطوط والمضلعات وأنماط رسمها وخصائص الإضاءة .
- تشير معظم متحولات الحالة إلى أنماط يمكن تفعيلها أو تعطيلها من خلال الأمرين ( )glEnable و glDisable ().

مثال:

```
glEnable(GL_LIGHTING);
glShadeModel(GL_SMOOTH);
glBegin(GL_LINE);
        glColor3f(1.0, 1.0, 1.0);
        glVertex2f(5.0, 5.0);
        glColor3f(0.0, 1.0, 0.0);
        glVertex2f(25.0, 5.0);
glEnd();
glDisable(GL_LIGHTING);
```

المحاور والمستويات الإحداثية :



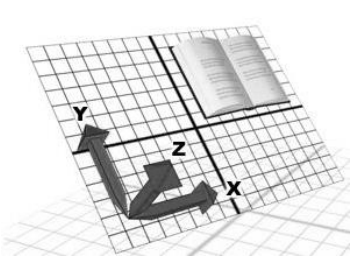
- لدينا ثلاثة محاور هي x,y,z

X : تكون الحركة يمين مركز الاحداثيات بقيم موجبة و يساراً بقيم سالبة .

Y : تكون الحركة أعلى مركز الاحداثيات بقيم موجبة و اسفل مركز الاحداثيات بقيم سالبة .

Z : تكون الحركة نحو الداخل (للشاشة) بقيم سالبة ونحو الخارج (للمستخدم) بقيم موجبة.

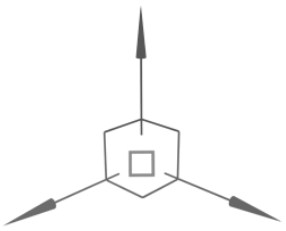
- ولدينا ثلاثة مستويات :



xy : الحركة وفق المحورين xy

yz : الحركة وفق المحورين yz

zx : الحركة وفق المحورين zx



النقاط (الرؤوس) والخطوط والمضلعات

1. النقاط (الرؤوس) :

- تمثل النقطة بمجموعة أرقام فاصلة عائمة وتسمى vertex.
- تنفذ جميع الحسابات الداخلية كما لو كانت الرؤوس ثلاثية الأبعاد. فإذا حدد المستخدم بعدين للرأس فقط (x,y) فعند ذلك تسند OpenGL القيمة 0 للبعد الثالث z.
- تضيف OpenGL قيمة إحداثية رابعة w وذلك عند عمل OpenGL بالإحداثيات المتجانسة حيث يتم من خلال تقسيم x,y,z على w الحصول على إحداثيات البعد الاقليدي (اقليدس) للنقطة (x/w , y/w, z/w). w نادراً ما تستخدم في OpenGL، وبشكل افتراضي يسند OpenGL القيمة 1 للاحداثي w.

تحديد النقاط :

- في OpenGL توصف جميع العناصر الهندسية كمجموعة مرتبة من الرؤوس. الصيغة العامة لأمر رسم رأس:

Void glVertex {234} {sifd} [v] (TYPE coords);

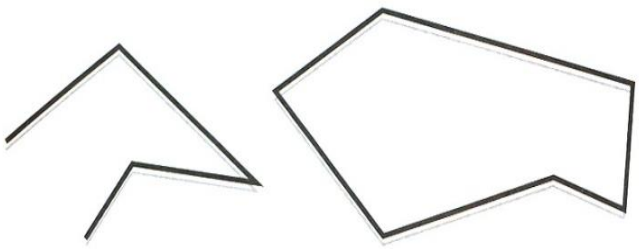
- يمكن تزويد حتى 4 إحداثيات (x,y,z,w). القيمة الافتراضية لـ z=0 و w=1 عند عدم تحديدها.
- استدعاء هذا الأمر يجب أن يكون بين ()glBegin و ()glEnd.

أمثلة :

```
glVertex2s(2, 3);
glVertex3d(0.0, 0.0, 3.1415926535898);
glVertex4f(2.3, 1.0, -2.2, 2.0);
GLdouble dvect[3] = {5.0, 9.0, 1992.0};
glVertex3dv(dvect);
```

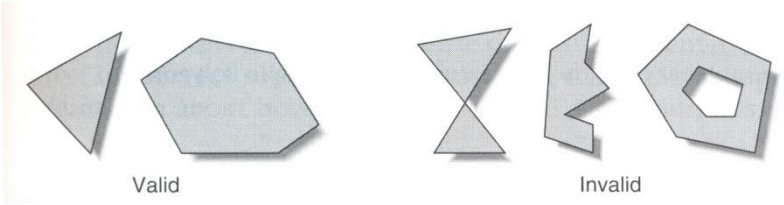
2. الخطوط lines :

- الخط في OpenGL عبارة عن قطعة مستقيمة، وليس كتعريف الرياضيون بأنه عبارة عن خط يمتد إلى اللانهاية من الطرفين.
- يمكن رسم عدة خطوط متصلة وكذلك يمكن بواسطة الخطوط المتصلة رسم شكل مغلق، يتم رسم الخطوط بتحديد رؤوس vertices النهايات لتلك الخطوط.

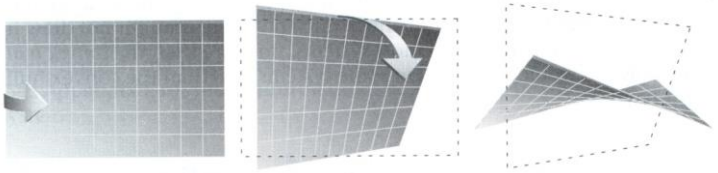


3. المضلعات Polygons :

- المضلع عبارة عن مساحة متضمنة بواسطة حلقة مغلقة وحيدة من مقاطع الخطوط.
- كل حلقة مغلقة تمثل ضلعاً. يمكن أن تكون المضلعات ممتلئة (ملئ البكسلات داخليا) كما يمكن أن ترسم بخطوط خارجية.
- هناك بعض القيود التي تضعها OpenGL لبناء مضلع قياسي (بسيط).

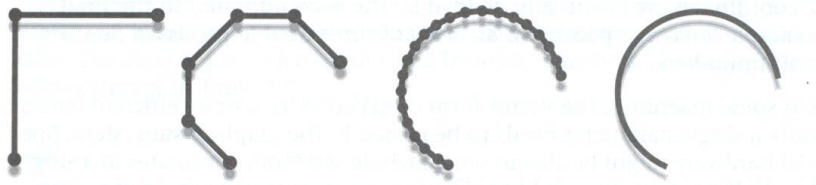


- يؤثر إسناد قيم مختلفة لرؤوس المضلع في المستوي على شكل المضلع بحيث يصبح مضلع معقد أحيانا.



المنحنيات Curves:

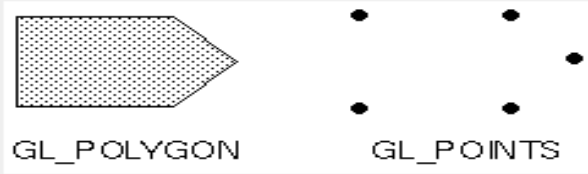
- ينتج الخط المنحني بتجميع مقاطع خطوط صغيرة .كلما ازدادت عدد المقاطع الخطية كلما ازدادت دقة الانحناء.



أساسيات الرسم الهندسي في Open GL :

- باستخدام أمر رسم الرؤوس تستطيع رسم مجموعة من النقاط، أو خط، أو مضلع، وذلك بوضع كل مجموعة من النقاط بترتيب معين ضمن ( )glBegin , ( ) glEnd .

```
glBegin(GL_POLYGON);
   glVertex2f(0.0, 0.0);
   glVertex2f(0.0, 3.0);
   glVertex2f(3.0, 3.0);
   glVertex2f(4.0, 1.5);
   glVertex2f(3.0, 0.0);
glEnd();
```



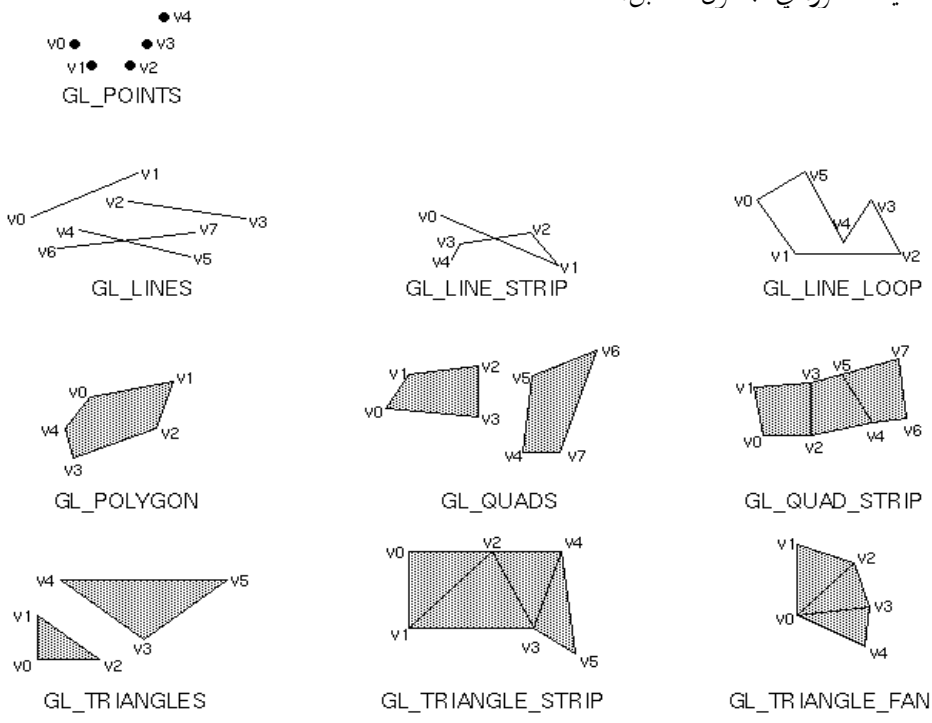
الشكل العام للأمر ( )glBegin :

Void glBegin(GLenum mode);

- يبين الجدول التالي معاملات الأمر ( )glBegin المحتملة ونوع العنصر الهندسي المقابل:

Value	Meaning
GL_POINTS	individual points
GL_LINES	pairs of vertices interpreted as individual line segments
GL_POLYGON	boundary of a simple, convex polygon
GL_TRIANGLES	triples of vertices interpreted as triangles
GL_QUADS	quadruples of vertices interpreted as four-sided polygons
GL_LINE_STRIP	series of connected line segments
GL_LINE_LOOP	same as above, with a segment added between last and first vertices
GL_TRIANGLE_STRIP	linked strip of triangles
GL_TRIANGLE_FAN	linked fan of triangles
GL_QUAD_STRIP	linked strip of quadrilaterals

يبين الشكل التالي العناصر الهندسية المذكورة في الجدول السابق:



الصيغة العامة للأمر glEnd

- يحدد نهاية لائحة النقاط:

```
Void glEnd(void);
```

قيود استخدام تعليمتي glEnd() و glBegin() :

- يحوي الجدول التالي الأوامر الممكن وقوعها يبين glBegin( ) و glEnd( )

Command	Purpose of Command
glVertex*()	set vertex coordinates
glColor*()	set current color
glIndex*()	set current color index
glNormal*()	set normal vector coordinates
glEvalCoord*()	generate coordinates
glCallList(), glCallLists()	execute display list(s)
glTexCoord*()	set texture coordinates
glEdgeFlag*()	control drawing of edges
glMaterial*()	set material properties

استدعاء أي أوامر أخرى بين الأمرين السابقين غير صحيح وقد يسبب حدوث أخطاء .

المستطيل Rectangle :

- تؤمن OpenGL أمر رسم مستطيل أساسي باستخدام الأمر glRect*().
- يمكن أيضاً رسم المستطيل كمضلع (باستخدام النقاط).

الصيغة العامة لأمر رسم المستطيل:

```
Void glRect {sifd} (TYPE x1, TYPE y1, TYPE x2, TYPE y2);
Void glRect {sifd} v (TYPE * v1, TYPE * v2);          أو
```

مثال:

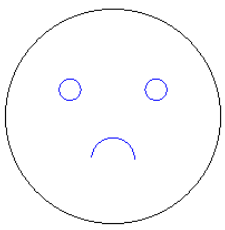
```
glRecti (-1,-1,2,2);
int r1 [] = {-1,-1};
int r2 [] = {2,2};
glRectiv (r1,r2);
```

مثال: رسم دائرة باستخدام الثابت :GL_POLYGON

```
drawCircle()
{
  int i;
  int edges=100;
  int factor=1;
  float cosine, sine;
  const float PI=3.142857;
    glBegin(GL_POLYGON);
      for(i=0;i<edges;i++)
    {
        cosine=factor*cos(i*2*PI/ edges);
        sine= factor*sin(i*2*PI/ edges);
        glVertex2f(cosine,sine);
     }
   glEnd();
}
```

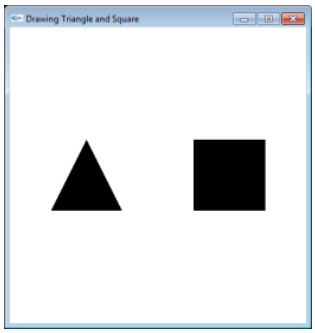
تعديلات:

- اجعل factor=20. ماذا تلاحظ؟
- اجعل edges=10. ماذا تلاحظ؟
- استبدل الثابت GL_POLYGON بالثابت GL_LINE_LOOP. ماذا تلاحظ؟
- عدل التعليمات بحيث يتم رسم نصف الدائرة العلوي فقط، ثم ربع الدائرة الأول فقط.
- ارسم الشكل المبين جانباً بالاعتماد على التابع drawCircle(factor,edges) لرسم دائرة كاملة فارغة و التابع drawHalhCircle(factor,edges) لرسم نصف دائرة فارغة (نصف علوي).
- عدل التعليمات لرسم قطع ناقص.

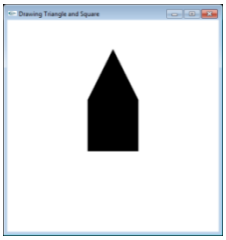


القسم العملي

تطبيق1: برنامج OpenGL لرسم مثلث ومربع




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

static void redraw(void);
int main(int argc, char **argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
        glutInitWindowPosition(100,100);
        glutInitWindowSize(400,400);
        glutCreateWindow("Drawing Triangle and Square");
        glutDisplayFunc(redraw);
        glMatrixMode(GL_PROJECTION);
        gluPerspective(45,1.0,10.0,200.0);
        glMatrixMode(GL_MODELVIEW);
        glutMainLoop();
        return 0;
}

static void redraw(void)
{
        glLoadIdentity();
        glTranslatef(-20.0,0.0,-100.0);
        glBegin(GL_TRIANGLES);                  // رسم المثلث
                glVertex3f(0.0,10.0, 0.0);
                glVertex3f(-10.0,-10.0, 0.0);   // النقطة السفلية اليسارية
                glVertex3f( 10.0,-10.0, 0.0);   // النقطة السفلية اليمينية
        glEnd();
        glTranslatef(40.0,0.0,0.0);
        glBegin(GL_QUADS);                      // رسم المربع
                glVertex3f(-10.0, 10.0, 0.0);   // الزاوية العلوية اليسارية
                glVertex3f( 10.0, 10.0, 0.0);   // الزاوية العلوية اليمينية
                glVertex3f( 10.0,-10.0, 0.0);   // الزاوية السفلية اليمينية
                glVertex3f(-10.0,-10.0, 0.0);   // الزاوية السفلية اليسارية
        glEnd();
        glutSwapBuffers();
}
```



- تعديل1: يطلب تلوين الخلفية أصفر والمثلث أخضر والمربع أزرق؟
- تعديل2: يطلب وضع المثلث فوق المربع في مركز الشاشة؟

تطبيق4: خوارزمية Bresenham لرسم المستقيم والدائرة:

تعد خوارزمية Bresenham طريقةً دقيقةً وفعالة لرسم مستقيم، وتكمن أهميتها في أنها تستخدم فقط حسابات تزايدية صحيحة، وبالتالي فهي طريقة أسرع من خوارزمية DDA، ويمكن تعديلها بحيث تستخدم لرسم دائرة ومنحنيات أخرى.

سندرس في هذه الجلسة طريقة رسم كل من المستقيم والدائرة باستخدام هذه الخوارزمية.

خوارزمية Bresenham لرسم دائرة:

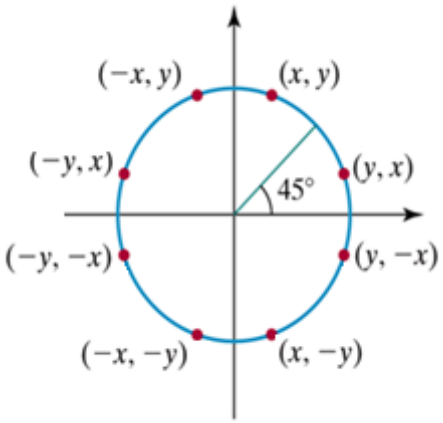
Bresenham Circle (X_c, Y_c, R):

Description: Here X_c and Y_c denote the x – coordinate and y – coordinate of the center of the circle. **R** is the radius.

```
1. Set X = 0 and Y = R
2. Set D = 3 - 2R
3. Repeat While (X < Y)
4. Call Draw Circle(Xc, Yc, X, Y)
5. Set X = X + 1
6. If (D < 0) Then
7. D = D + 4X + 6
8. Else
9. Set Y = Y - 1
10. D = D + 4(X - Y) + 10
[End of If]
11. Call Draw Circle(Xc, Yc, X, Y)
[End of While]
12. Exit
```

Draw Circle (X_c, Y_c, X, Y):

```
1. Call PutPixel(Xc + X, Yc + Y)
2. Call PutPixel(Xc - X, Yc + Y)
3. Call PutPixel(Xc + X, Yc - Y)
4. Call PutPixel(Xc - X, Yc - Y)
5. Call PutPixel(Xc + Y, Yc + X)
6. Call PutPixel(Xc - Y, Yc + X)
7. Call PutPixel(Xc + Y, Yc - X)
8. Call PutPixel(Xc - Y, Yc - X)
9. Exit
```

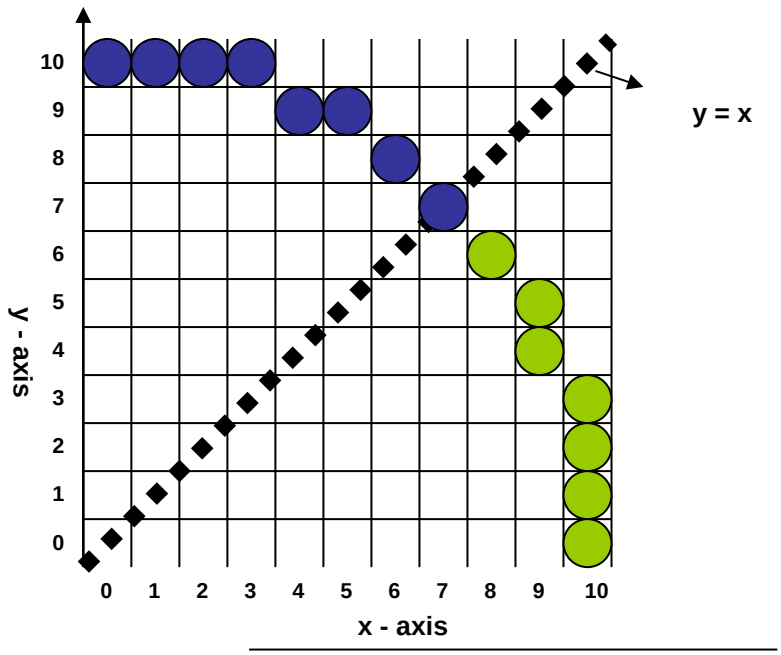


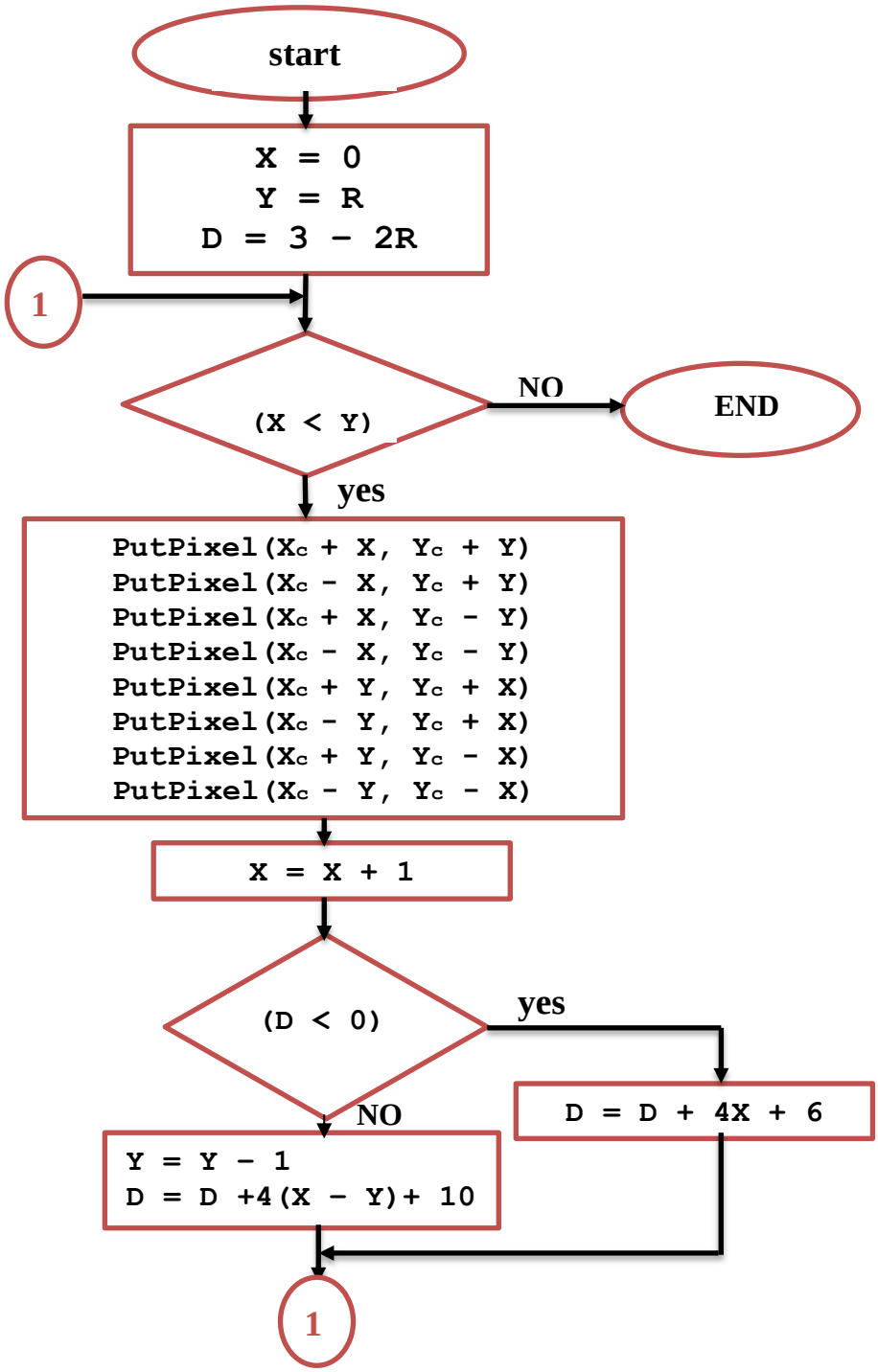
مثال عددي:

استخدم خوارزمية Bresenham Circle لرسم دائرة نصف قطرها (r =10)، وذلك في الربع الأول والثمن الثاني، معتبراً أن نقطة البداية هي (0,r).

على طول ثمن الدائرة الذي تتغير قيمة x فيه من x=0 إلى x=y في الربع الأول والثمن الثاني ، لذا سنستخدم خطوة تزايدية موجبة واحدية على المحور x لرسم ثمن الدائرة، ثم سنستخدم مبدأ التماثل بين أثمان الدائرة الثمانية لرسم الأثمان السبعة الأخرى.

$(x_0,y_0) = (0 , r)$	**(0,10)**
$p0=3-2r=3-2(10)=-17<0$	**(1,10)**
$p1=p0+4x0+6=-17+4(0)+6=-11<0$	**(2,10)**
$p2=p1+4x1+6=-11+4(1)+6=-1<0$	**(3,10)**
$p3=p2+4x2+6=-1+4(2)+6=13>0$	**(4,9)**
$p4=p3+4(x3-y3)+10=13+4(3-10)+10=-5<0$	**(5,9)**
$p5=p4+4x4+6=-5+4(4)+6=17>0$	**(6,8)**
$p6=p5+4(x5-y5)+10=17+4(5-9)+10=11>0$	**(7,7)**




start
X = 0
Y = R
D = 3 – 2R
1
(X < Y)
NO
END
yes
PutPixel(Xc + X, Yc + Y)
PutPixel(Xc - X, Yc + Y)
PutPixel(Xc + X, Yc - Y)
PutPixel(Xc - X, Yc - Y)
PutPixel(Xc + Y, Yc + X)
PutPixel(Xc - Y, Yc + X)
PutPixel(Xc + Y, Yc - X)
PutPixel(Xc - Y, Yc - X)
X = X + 1
(D < 0)
yes
NO
D = D + 4X + 6
Y = Y – 1
D = D +4(X – Y)+ 10
1

برنامج رسم دائرة بواسطة خوارزمية Bresenham:

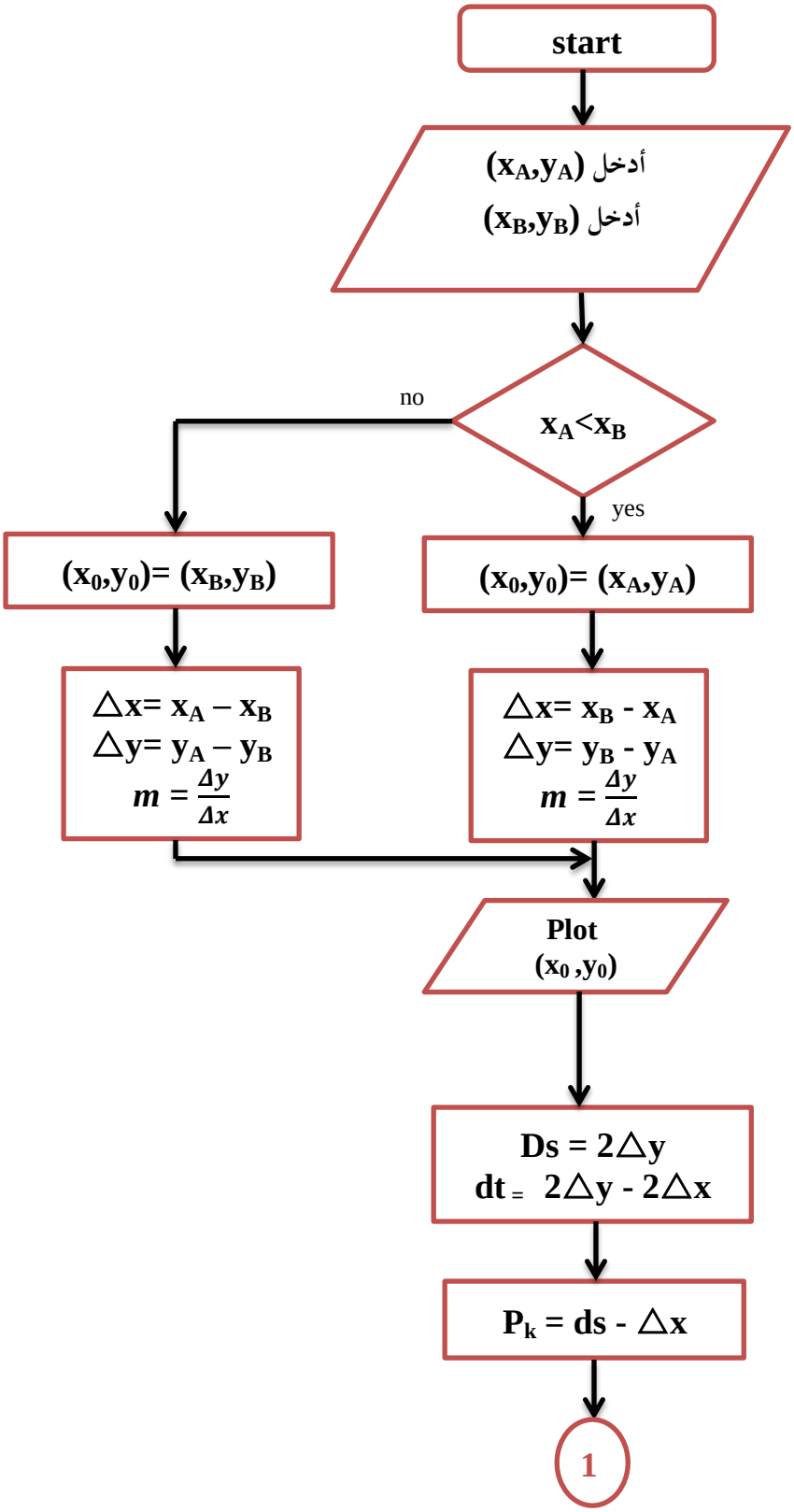
```
#include <windows.h>
#include <GL/glut.h>
#include <stdio.h>
#include <math.h>

void init(void){
    glClearColor(1.0,1.0,0.0,0.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0,200.0,0.0,150.0);
}
void setPixel(GLint x,GLint y)
{
    glBegin(GL_POINTS);
      glVertex2i(x,y);
    glEnd();
}
void Circle(){
            int xCenter=100,yCenter=100,r=50;
        int x=0,y=r;
        int d=3-(2*r);
        glClear(GL_COLOR_BUFFER_BIT);
     glColor3f(1,0,0.5);
   while(x<=y){
            setPixel(xCenter+x,yCenter+y);
            setPixel(xCenter+y,yCenter+x);
            setPixel(xCenter-x,yCenter+y);
            setPixel(xCenter+y,yCenter-x);
            setPixel(xCenter-x,yCenter-y);
            setPixel(xCenter-y,yCenter-x);
            setPixel(xCenter+x,yCenter-y);
            setPixel(xCenter-y,yCenter+x);
            if (d<0)
                d += (4*x)+6;
            else
            {
                d += (4*(x-y))+10;
```

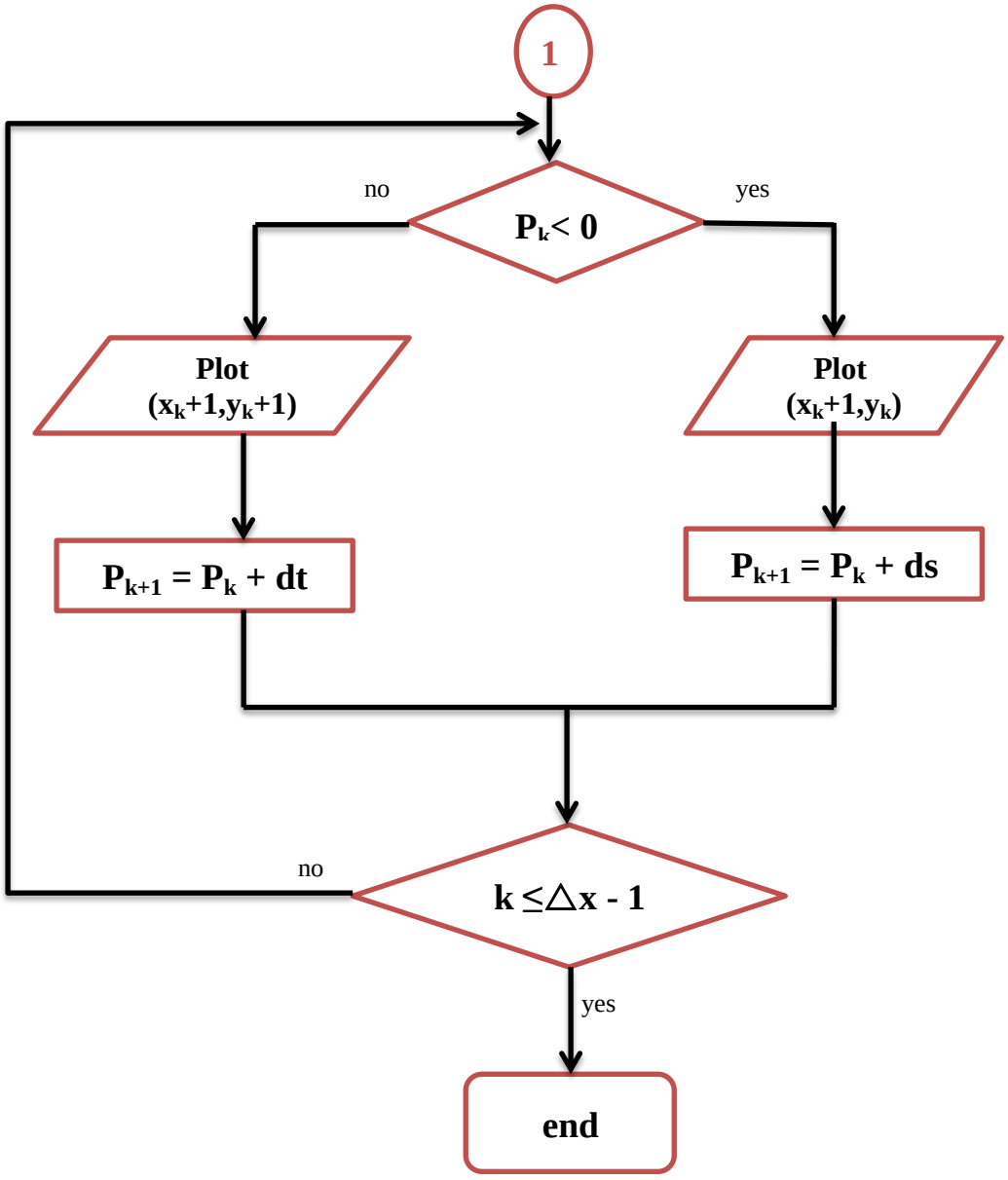
```
                y -= 1;
            }
            x++;
        }
        glFlush();
}

int main(int argc,char **argv){
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
   glutInitWindowPosition(0,0);
   glutInitWindowSize(500,500);
   glutCreateWindow("Bresenham Circle");
   init();
   glutDisplayFunc(Circle);
   glutMainLoop();
   return 0;
}
```

خوارزمية Bresenham لرسم مستقيم:

Bresenham's Line-Drawing Algorithm for $|m| < 1.0$

1. Input the two line endpoints and store the left endpoint in (x_0, y_0).
2. Set the color for frame-buffer position (x_0, y_0); i.e., plot the first point.
3. Calculate the constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as
$$p_0 = 2\Delta y - \Delta x$$
4. At each x_k along the line, starting at $k = 0$, perform the following test. If $p_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and
$$p_{k+1} = p_k + 2\Delta y$$
Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and
$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$
5. Perform step 4 $\Delta x - 1$ times.


start
أدخل (x_A,y_A)
أدخل (x_B,y_B)
$x_A<x_B$
no
yes
$(x_0,y_0)= (x_B,y_B)$
$(x_0,y_0)= (x_A,y_A)$
$\triangle x= x_A - x_B$
$\triangle y= y_A - y_B$
$m = \frac{\Delta y}{\Delta x}$
$\triangle x= x_B - x_A$
$\triangle y= y_B - y_A$
$m = \frac{\Delta y}{\Delta x}$
Plot
(x_0 ,y_0)
$Ds = 2\triangle y$
$dt = 2\triangle y - 2\triangle x$
$P_k = ds - \triangle x$
1


1
no
yes
$P_k < 0$
Plot
(x_k+1, y_k+1)
Plot
(x_k+1, y_k)
$P_{k+1} = P_k + dt$
$P_{k+1} = P_k + ds$
no
$k \leq \triangle x - 1$
yes
end

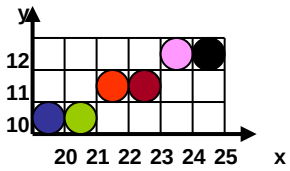
مثال عددي:

استخدم خوارزمية Bresenham لتحديد أماكن البيكسلات على مسار الخط الواصل بين النقطتين (x_A,y_A) = (20, 10) و (x_B,y_B) = (25,12) بحيث يبدأ الرسم بـ A.

$$\triangle x = x_B - x_A = 25 - 20 = 5$$
$$\triangle y = y_B - y_A = 12 - 10 = 2$$

$$m = \frac{\Delta y}{\Delta x} = \frac{2}{5} \qquad (0 < m < 1)$$

بما أن قيمة الميل m هي بين الصفر والواحد، وبما أننا نرسم المستقيم من اليسار إلى اليمين (إذاً الرسم سيكون في الربع الأول والثمن الأول)، لذا سوف نتحرك بخطوة موجبة واحدية إلى كل عمود (موقع x) ونرسم البيكسل الذي تكون قيمة y عنده أقرب إلى مسار المستقيم الذي نقوم برسمه.



$(x_0,y_0) = (x_A,y_A)$ **(20,10)**
$P_0 = 2\triangle y - \triangle x = 2(2) - 5 = -1 < 0,$ **(21,10)**
$P_1 = P_0 + 2\triangle y = -1 + 2(2) = 3 > 0,$ **(22,11)**
$P_2 = P_1 + 2\triangle y - 2\triangle x = 3 + 2(2) - 2(5) = -3 < 0,$ **(23,11)**
$P_3 = P_2 + 2\triangle y = -3 + 2(2) = 1 > 0,$ **(24,12)**
$P_4 = P_3 + 2\triangle y - 2\triangle x = 1 + 2(2) - 2(5) = -5 < 0,$ **(25,12)**

برنامج رسم مستقيم بواسطة خوارزمية Bresenham:

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

void init(void)
{
//set display-window background color to white
 glClearColor(1.0,1.0,1.0,0.0);
//set projection paramaters
 glMatrixMode(GL_PROJECTION);
 gluOrtho2D(0.0,300.0,0.0,300.0);
```

```
 }

void setPixel(GLint xCoordinate, GLint yCoordinate)
{
 glBegin(GL_POINTS);
  glVertex2i(xCoordinate,yCoordinate);
glEnd();
  glFlush(); //executes all OpenGL functions as quickly as possible
}

void lineBres(int x1, int y1, int xn, int yn)
{
 int dx = xn - x1, dy = yn - y1;
 int p0 = 2 * dy - dx;
 int ds = 2 * dy, dt = 2 * (dy - dx);
 setPixel(x1, y1);
 while (x1 < xn)
 {
        x1++;
        if (p0 < 0)
        p0 = p0 + ds;
        else
        {
                y1++;
                p0 = p0 + dt;
        }
        setPixel(x1, y1);
}
}

void drawMyLine(void)
{
 glClear(GL_COLOR_BUFFER_BIT);
 glColor3f(1.0,0.0,0.0);
 glPointSize(4.0);
 GLint x0 = 100;
 GLint y0 = 100;
 GLint xEnd = 200;
```

```
GLint yEnd = 200;
lineBres(x0,y0,xEnd,yEnd);
}

int main(int argc, char**argv)
{
//initialize GLUT
glutInit(&argc,argv);
//initialize display mode
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
//set display-window width & height
glutInitWindowSize(500,500);
//set display-window upper-left position
glutInitWindowPosition(0,0);
//create display-window with a title
glutCreateWindow("Bresenham  Algorithm for line ");
//initialze OpenGL
init();
//call graphics to be displayed on the window
glutDisplayFunc(drawMyLine);
//display everything and wait
glutMainLoop();
return 0;
}
```

<u>تعديل</u> : يطلب رسم المستقيم في أرباع أخرى من الإطار.