1. مقدمـة :

يشهد وقتنا الراهن تطوراً مستمراً في عتاد الحواسيب Hardware، وفي البرمجيات المرافقة له Software ضمن مختلف المجالات (أنظمة التشغيل والبرامج المكتبية وبرامج التصميم والبرامج الرسومية ولغات البرمجة).

سنهتم في هذه الجلسات بتطبيق الخوارزميات الشهيرة المستخدمة في رسوميات الحاسوب باستخدام المكتبة OpenGL (الاصدار الأحدث هو OpenGL 4.5 بتاريخ 2014/8/11 ويتميز بدعم العتاد البرمجي والصلب ذي 64 bit).

2. ما هي OpenGL:

تعني OpenGL (Open Graphic Library) مكتبة الرسوميات المفتوحة وهي واجهة برمجية للعتاد الخاص بالرسوميات. تتألف هذه الواجهة من حوالي 250 أمر مميز (200 ضمن نواة OpenGL و 50 تابعة لمكتبة الخدمات GLU) تسمح لمبرمجي الرسوميات بتحديد العناصر والعمليات المطلوبة لانتاج صور رسومية ملونة عالية الجودة ثلاثية الأبعاد.

3. مزايا OpenGL:

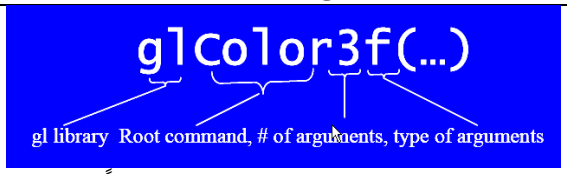
تتمتع OpenGL بالمزايا التالية:

- تقدم تسريعاً ثلاثي الأبعاد على مستوى العتاد Hardware.
- تدعم وبقوة التطبيقات والألعاب التي تعالج كمية ضخمة من البيانات في الزمن الحقيقي باستخدام العناصر الهندسية والإضاءة وعمليات الاقتطاع والتحويلات والتنفيذ rendering.
- تضيف تأثيرات خاصة إلى الصور دون التأثير على الأداء. وكمثال على ذلك إضافة ضباب وصقل وظلال وغباشة متحركة وشفافية وتراكيب ثلاثية الأبعاد في الزمن الحقيقي.
- تُنفَذ OpenGL على أنظمة تشغيل مختلفة.
- تستطيع و بسهولة نقل تطبيقات و ألعاب تدعم OpenGL من نظام تشغيل لآخر. و هذا يجعل تطبيقات OpenGL قابلة للحمل، أي تعمل على أنظمة تشغيل مختلفة مثل Windows و Linux و OS/2.

4. صيغة أوامر OpenGL:

تستخدم OpenGL بادئة تعبر عن المكتبة المأخوذ منها الأمر (مثلاً تعبر البادئة gl عن الأوامر المأخوذة من المكتبة opengl32.lib)، كما تستخدم حروف كبيرة لبداية كل كلمة تؤلف اسم الأمر مثل glClearColor. وتضاف أحياناً حروف إلى أسماء الأوامر مثل 3f في الأمر glColor3f() ، حيث يدل الرقم 3 على وجود ثلاثة وسائط للأمر أما الحرف f فيدل على أن الوسائط من نوع أعداد فاصلة عائمة. يبين الشكل التالي صيغة أوامر OpenGL مع مثال:

<Library prefix><Root command><Argument count><Argument type>



تبدأ الثوابت المعرفة ضمن OpenGL بالسابقة GL وتستخدم حروفاً كبيرة ورمز الشرطة السفلية "_" Underscore لتفصل الكلمات عن بعضها مثل GL _ COLOR_ BUFFER_BIT .

يبين الجدول التالي الحروف المستخدمة لتحديد أنواع بيانات الوسائط:

الحرف	نوع البيانات	النوع المقابل بلغة c	تعريف النوع بـ OpenGL
b	عدد صحيح 8 Bit	Signed char	GLbyte
s	عدد صحيح 16 Bit	short	GLshort
i	عدد صحيح 32 Bit	Long	GLint , GLsizei
f	فاصلة عائمة 32 Bit	Float	GLfloat,GLclampf
d	فاصلة عائمة 64 Bit	double	GLdouble , GLclamped
ub	عدد صحيح غير مؤشر 8 Bit	Unsigned char	GLubyte , GLboolean
us	عدد صحيح غير مؤشر 16 Bit	Unsigned short	GLushort
ui	عدد صحيح غير مؤشر 32 Bit	Unsigned long	GLunit , GLenum , GLbitfield

مثال1:

GLshort A[10];	⟷	**short A[10];**
GLdouble B;	⟷	**double B;**

مثال2:

الأمران glVertex2i(1,3) و glVertex2f(1.0,3.0) متكافآن عدا أن الأول يحدد إحداثيات النقطة كأعداد صحيحة بطول 32bit أما الثاني فيحددها كأرقام فاصلة عائمة أحادية الدقة .

يمكن أن تأخذ بعض أوامر OpenGL حرفاً أخيراً (v) يشير إلى أن الأمر يشير إلى شعاع أو مصفوفة من القيم بدلاً من سلسلة وسائط مستقلة. أغلب الأوامر يمكن استعمالها مع شعاع v أو بدون شعاع . المثال التالي هو أمر لتحديد اللون تمت كتابته مع شعاع وبدون شعاع، كما هو مبين في المثال التالي:

مثال3:

```
glColor3f(1.0,0.0,0.0);                    بدون شعاع

float color_ array[ ] ={1.0,0.0,0.0};      مع شعاع
glColor3fv( color_array);
```

أخيراً تعرف OpenGLالثابت GLvoid بدلاً من void إذا كنت مبرمجاً بلغة c.

5. المكتبات المرتبطة بـ OpenGL:

تحوي OpenGL مجموعة قوية من أوامر الرسم وتكون مخزنة ضمن مكتبات خاصة. نورد فيما يلي أشهر هذه المكتبات:

1. Opengl: تعتبر من أشهر مكتبات OpenGL و أكثرها استخداماً. تبدأ جميع أوامر (توابع) هذه المكتبة بالبادئة gl.
2. glu (OpenGL Utility Library): مكتبة خدمات OpenGL تتضمن مهام مختلفة مثل رسم كرة و أسطوانة و منحني و تغيير حجم صورة. تبدأ جميع أوامر هذه المكتبة بالبادئة glu.
3. glut (OpenGL Utility Toolkit): عبارة عن مجموعة أدوات خدمية تابعة لـ OpenGL مكتوبة بواسطة مارك كيلغارد Mark Kilgard. استخدام هذه المكتبة سهل و مفيد، و سنستخدم هذه المكتبة لتهيئة و إنشاء نافذة OpenGL .تبدأ جميع أوامر هذه المكتبة بالبادئة glut.

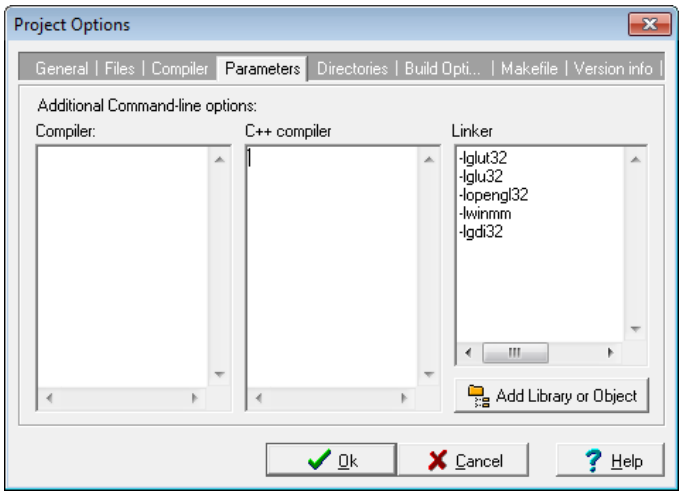
القسم العملي

1. إعداد البرنامج Dev-C++ لتنفيذ أوامر OpenGL:

اتبع الخطوات التالية:

1. حمّل الحزمة الخاصة بالمكتبة glut من الانترنت (glut.3.7.6+.DevPak).
2. شغّل البرنامج **Dev-C++** ثم انتقل إلى القائمة Tools وانتق منها الأمر Package Manager. يظهر عند ذلك مربع الحوار Package Manager. انقر فوق الزر Install وثبت الملف السابق (glut.3.7.6+.DevPak).
3. أنشئ مشروعاً جديداً File->New->Project. واختر نوع المشروع Console Application وحدد اسماً له ضمن حقل Name.
4. انتقل إلى القائمة Project وانتق منها الخيار Project Options، يظهر عند ذلك مربع الحوار المبين في الشكل التالي، انتقل إلى علامة التبويب Parameters، اللوح linker وأضف السطر التالي:

-lglut32 -lglu32 -lopengl32 -lwinmm -lgdi32



2. برنامج إطار OpenGL

حتى تتمكن من تنفيذ أوامر OpenGL وإظهار الرسوميات، أنت بحاجة إلى إطار لإظهار تلك الرسوميات (ناتج تنفيذ الأوامر). تشرح هذه الفقرة بشكل مفصل برنامج يستخدم لتجهيز إطار OpenGL، و سنستفيد من هذا البرنامج لعرض رسوميات OpenGL ضمنه و تأثيراتها الأخرى (كالإضاءة و الإكساء ...). سنستخدم المكتبة Glut لإنشاء برنامج الإطار هذا. و لنبدأ بشرح هذا البرنامج:

1. نُصرح في البداية عن ملفات العناوين headers للمكتبات التي سنستخدمها كما يلي:

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

2. نبدأ بعد ذلك بالتابع الرئيسي في لغة C++ و هو main.

```
int main(int argc, char**argv)
{
 //initialize GLUT
 glutInit(&argc,argv);
 //initialize display mode
 glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB| GLUT_DEPTH);
//set display-window upper-left position
 glutInitWindowPosition(0,0);
 //set display-window width & height
 glutInitWindowSize(500,500);
 //create display-window with a title
 glutCreateWindow("window title");
 //initialize OpenGL view
 glMatrixMode(GL_PROJECTION);
 gluOrtho2D(0.0,30.0,0.0,30.0);
glMatrixMode(GL_MODELVIEW);
 //call graphics to be displayed on the window
 glutDisplayFunc(drawMyLine);
 //display everything and wait
 glutMainLoop();
 return 0;
}
```

1. يحوي المتحول argc السابق عدد الوسائط الممررة لبرنامجنا من سطر الأوامر وبما أن اسم البرنامج هو وسيط، وبالتالي فإن أقل قيمة لـ argc هي 1.
2. يمثل argv مصفوفة من السلاسل النصية ويصرح عنا بأنها مؤشر إلى مؤشرات من نوع char. وأول سلسة نصية [0]argv تمثل اسم البرنامج، وتمثل كل سلسلة تالية وسيط ممرر إلى البرنامج من سطر الأوامر. لا تقلق بشأن هذه المؤشرات فهي تمرر إلى التابع glutInit الذي يستخدم لتهيئة إطار Glut.
3. يستخدم التابع ( )glutInitDisplayMode لإعداد نمط الإظهار. سنستدعي هذا التابع وفق الوسائط التالية:
   - GLUT_RGB: يستخدم لإضافة ذاكرة مؤقتة buffer لألوان RGB ضمن الإطار الخاص بنا.
   - GLUT_DOUBLE: يستخدم لإضافة ذاكرة مؤقتة مزدوجة Double. تمكننا الذاكرة المؤقتة المزدوجة من إنهاء الرسم قبل إرساله إلى الشاشة تجنباً لحدوث الوميض.
   - GLUT_DEPTH: يضيف ذاكرة مؤقتة للعمق إلى إطارنا. تستخدم هذه الذاكرة المؤقتة لتحديد بعد العناصر عن الكاميرا (عين الناظر).
4. يستخدم التابع ()glutInitWindowPosition لتحديد موقع الإطار (احداثيات x,y لزاوية الإطار العليا اليسرى) .
5. أما التابع ()glutInitWindowSize فيستخدم لتعيين أبعاد الإطار (العرض والارتفاع) .
6. ينشئ التابع ( ) glutCreateWindow الإطار الذي سنرسم ضمنه العناصر.

7. تهيئة الرؤية في OpenGL: سنطبق بعد ذلك مصفوفة إسقاط Projection matrix، و تستخدم هذه المصفوفة لتحديد موقع الكاميرا بالنسبة للشاشة. ثم استبدال المصفوفة السابقة بمصفوفة التحويل modelView. وهي عبارة عن مصفوفة 4×4 تحول النقاط المرسومة من إحداثياتها الحقيقية (الفضاء الحقيقي) إلى إحداثيات تتعلق بالكاميرا.
8. يحدد ( ) glutDisplayFunc التابع المستخدم للرسم (التابع drawMyLine).
9. يستخدم التابع ( )glutMainLoop لتكرار إظهار إطار Glut بشكل مستمر.

ملاحظة:
يجب الانتباه لحالة الأحرف عند كتابة الأوامر في لغة ++VC، لأن هذه اللغة حساسة لحالة الأحرف.

3. أوامر OpenGL المستخدمة في هذه الجلسة

1. تعيين لون المسح الحالي (لون خلفية أبيض) لاستخدامه في مسح بفر اللون (يستخدم بفر اللون لتخزين ألوان البكسلات الضوئية):

```
glClearColor(1.0,1.0,1.0,0.0);
```

2. مسح الذواكر المؤقتة buffers الخاصة باللون والعمق حسب قيم المسح الحالية:

```
glClear(GL_COLOR_BUFFER_BIT);
```

3. يستبدل المصفوفة الحالية بالمصفوفة الواحدية وهذا يعيدنا إلى مركز الشاشة (0,0,0)

```
glLoadIdentity();
```

4. يحركنا عبر الشاشة لتحديد مكان الرسم الجديد. الحركة لا تتم من مركز الشاشة وإنما من الموقع الحالي للرسم:

```
glTranslatef(-1.5f,0.0f,-6.0f);
```

5. يعين لون الرسم الحالي للعناصر . جميع العناصر الواردة بعد هذا الأمر سترسم بلونه:

```
glColor3f(1.0, 1.0, 1.0);
```

6. يعين بداية لائحة النقاط المرسومة ،ويحوي بارامتر لتحديد نوع العنصر الهندسي المرسوم. مثلاً GL_TRIANGLES لرسم عنصر هندسي ثلاثي النقاط:

```
glBegin(GL_TRIANGLES);
```

7. يحدد نهاية لائحة النقاط المرسومة.

```
glEnd();
```

8. تحديد إحداثيات النقاط المراد رسمها.

```
glVertex2f(0.5, 0.5);
```

- **تطبيق عملي 1:: برنامج OpenGL لإظهار مستطيل أبيض على خلفية سوداء**

```
#include <GL/glut.h>
#include <stdlib.h>
#include<math.h>
static void redraw(void)
{
      glClearColor(0.0,0.0,1.0,1.0);
      glClear(GL_COLOR_BUFFER_BIT);
      glLoadIdentity();
      glTranslatef(-1.5,0.0,-100.0);
      glColor3f(1.0, 1.0, 1.0);
      glBegin(GL_POLYGON);
                glVertex2f(-30, -30);
                glVertex2f(-30, 10);
                glVertex2f(40, 10);
                glVertex2f(40, -30);
  glEnd();
  glutSwapBuffers();
}
 int main(int argc, char**argv)
{
   glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
        glutInitWindowPosition(100,100);
        glutInitWindowSize(400,400);
        glutCreateWindow("Application11");
        glutDisplayFunc(redraw);
        glMatrixMode(GL_PROJECTION);
        gluPerspective(45,1.0,10.0,200.0);
        glMatrixMode(GL_MODELVIEW);
        glutMainLoop();
        return 0;
}
```

✍ **تطبيق عملي 2:** تنفيذ الخوارزمية **DDA(Digital Differential Analyzer)** باستخدام **OpenGL**:

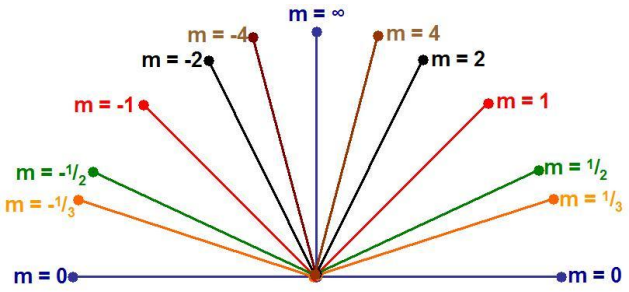
DDA Pseudo-code

```
// assume that slope is gentle
DDA(float x0, float x1, float y0, float y1) {
  float x, y;
  float xinc, yinc;
  int numsteps;

  numsteps = Round(x1) – Round(x0);
  xinc  = (x1 – x0) / numsteps;
  yinc  = (y1 – y0) / numsteps;
  x = x0;
  y = y0;
  ColorPixel(Round(x),Round(y));

  for (int i=0; i<numsteps; i++) {
    x += xinc;
    y += yinc;
    ColorPixel(Round(x),Round(y));
  }

}
```




```
#include <GL/glut.h>
#include <stdlib.h>
#include<math.h>
 inline GLint round (const GLfloat a) {return GLint (a+0.5);}
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 lineDDA(GLint x0, GLint y0, GLint xEnd, GLint yEnd)
{
 GLint dx = xEnd - x0;
 GLint dy = yEnd - y0;
 GLint steps, k;
 GLfloat xIncrement, yIncrement, x=x0, y=y0;
 if(fabs(dx) > fabs(dy))
  steps =(int) fabs(dx);
 else
  steps = (int)fabs(dy);
  xIncrement = GLfloat (dx) / GLfloat (steps);
 yIncrement = GLfloat (dy) / GLfloat (steps);
 setPixel(round (x), round(y));
for(k=0; k<steps; k++)
 {
  x += xIncrement;
  y += yIncrement;
  setPixel(round(x) , round(y));
 }
}
 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;
 lineDDA(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("Digital Differential Analyzer Algorithm: Programmed by
Salha");
 //initialze OpenGL
 init();
 //call graphics to be displayed on the window
 glutDisplayFunc(drawMyLine);
 //display everything and wait
 glutMainLoop();
 return 0;
}
```

تعديل: يطلب رسم المستقيم في أرباع أخرى من الإطار.

تمرين: لدينا مستقيم احداثيات نقطتيه: (10,-20-)و (5,-15-) والمطلوب:

- الحساب اليدوي لبكسلات رسم المستقيم.
- كتابة برنامج لرسم الستقيم السابق باستخدام خوارزمية DDA.

****