بسم الله الرحمن الرحيم

الوثيقة : Vertex Arrays **في** OpenGL .
الكاتب : **عبدالله الشمري**
aboshammar@gmail.com
تم نشرها في موقع : **الفريق العربي للبرمجة** .

www.arabteam2000-forum.com

ملاحظة : **الوثيقة موجهة إلى من يملك معلومات أساسية عن** OpenGL .
وسنستخدم لغة السي في هذه الوثيقة .

الفصل الأول :

القسم الأول :
استخدام **Vertex Arrays** مع الدالة **glArrayElement** ٤
القسم الثاني :
استخدام **Vertex Arrays** مع الدالة **glArrayElement** ١٠
القسم الثالث :
استخدام **Vertex Arrays** مع الدالة **glDrawArrays** ١٦

الفصل الثاني :

القسم الأول : **الألوان** ١٨
القسم الثاني : **الإكساء**..... ٢٣

PDF created with pdfFactory trial version www.pdffactory.com

Vertex Arrays .. "**مصفوفة النقاط**" .

أولا مالهدف ومالفكرة من هذه الطريقة ؟

الهدف هو التسهيل على المبرمج لرسم الاشكال المعقدة والكبيرة .

فمثلا لو أردت أن ترسم مكعب فستحتاج إلى كود طويل عريض كما في هذا المثال :

```
// Without Vertex Arrays
glBegin(GL_QUADS);

        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f( 1.0f, 1.0f,-1.0f);
        glVertex3f(-1.0f, 1.0f,-1.0f);
        glVertex3f(-1.0f, 1.0f, 1.0f);
        glVertex3f( 1.0f, 1.0f, 1.0f);


        glColor3f(1.0f,0.5f,0.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glVertex3f( 1.0f,-1.0f,-1.0f);

        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 1.0f, 1.0f, 1.0f);
        glVertex3f(-1.0f, 1.0f, 1.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);

        glColor3f(1.0f,1.0f,0.0f);
        glVertex3f( 1.0f,-1.0f,-1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glVertex3f(-1.0f, 1.0f,-1.0f);
        glVertex3f( 1.0f, 1.0f,-1.0f);

        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f(-1.0f, 1.0f, 1.0f);
        glVertex3f(-1.0f, 1.0f,-1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);

        glColor3f(1.0f,0.0f,1.0f);
        glVertex3f( 1.0f, 1.0f,-1.0f);
        glVertex3f( 1.0f, 1.0f, 1.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
        glVertex3f( 1.0f,-1.0f,-1.0f);

glEnd();
```

```
// With Vertex Arrays

/*1*/   GLfloat vertex[] ={ 1.0f, 1.0f,-1.0f ,
        -1.0f, 1.0f,-1.0f,-1.0f, 1.0f, 1.0f,
        1.0f, 1.0f, 1.0f ,1.0f,-1.0f, 1.0f ,
        -1.0f,-1.0f, 1.0f,-1.0f,-1.0f,-1.0f,
        1.0f,-1.0f,-1.0f ,1.0f, 1.0f, 1.0f ,
        -1.0f, 1.0f, 1.0f ,-1.0f,-1.0f, 1.0f ,
        1.0f,-1.0f, 1.0f ,1.0f,-1.0f,-1.0f,
        -1.0f,-1.0f,-1.0f,-1.0f, 1.0f,-1.0f,
        1.0f, 1.0f,-1.0f,-1.0f, 1.0f, 1.0f,
        -1.0f, 1.0f,-1.0f,-1.0f,-1.0f,-1.0f,
        -1.0f,-1.0f, 1.0f,1.0f, 1.0f,-1.0f,
        1.0f, 1.0f, 1.0f,1.0f,-1.0f, 1.0f,
        1.0f,-1.0f,-1.0f};

/*2*/   GLfloat color [] ={0.0f,1.0f,0.0f ,
           0.0f,1.0f,0.0f, 0.0f,1.0f,0.0f,
           0.0f,1.0f,0.0f, 1.0f,0.5f,0.0f,
           1.0f,0.5f,0.0f, 1.0f,0.5f,0.0f,
           1.0f,0.5f,0.0f, 1.0f,0.0f,0.0f,
           1.0f,0.0f,0.0f, 1.0f,0.0f,0.0f,
           1.0f,0.0f,0.0f, 1.0f,1.0f,0.0f,
           1.0f,1.0f,0.0f, 1.0f,1.0f,0.0f,
           1.0f,1.0f,0.0f, 0.0f,0.0f,1.0f,
           0.0f,0.0f,1.0f, 0.0f,0.0f,1.0f,
           0.0f,0.0f,1.0f, 1.0f,0.0f,1.0f,
           1.0f,0.0f,1.0f, 1.0f,0.0f,1.0f,
           1.0f,0.0f,1.0f};

void Render()
{
        .
        .
        .

      glEnableClientState (GL_VERTEX_ARRAY);
      glEnableClientState (GL_COLOR_ARRAY);
      glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);
      glColorPointer(3 ,GL_FLOAT,0,&color[0]);
      glDrawArrays(GL_QUADS, 0, 24);

}
```

ستلاحظ من خلال المقارنة بين الكود الايمن والذي يحوي الـVertex Arrays و الكود الايسر الذي يتبع الطريقة التقليدية .

PDF created with pdfFactory trial version www.pdffactory.com

ستلاحظ أننا نختصر كثيرا من الاكواد بالاضافة الى ان الVertex Arrays هي الاسرع.

قد تتساءل أخي القارئ ... !
أين السهولة ... كلا الطريقتين طويلتين ومملتين .. ففي الVertex Arrays نستخدم مصفوفة ونملأها بالاحداثيات وهي ايضا عمليه مملة ’،،،،

الجواب هو كالتالي :
أكثر من يستفيد من Vertex Array هم الذين يقومون بتحميل مجسمات من ملفات خارجية .. مثل ملفات الثري دي ماكس و المايا .. الخ .

فعندما تقوم بتحميل هذه المجسمات الكبيرة .. فأنت تتعامل مع مضلعات يصل حجمها الى مئات المضلعات .. وتقوم ايضا بتحميل احداثيات الاكساء وتقوم بتحميل احداثيات الNormal ... .

اذا اردت استعمال الطريقة التقليدية لتخزين ورسم النقاط فانك ستكره البرمجة والكمبيوتر كله . .

مالذي ستقوم به لرسم شكل استوردته من الثري دي ماكس ....

```
glColor(...);
glNormal(....);
glTexCoor2f(...)
glVertex3f(...);
```

السطور السابقة من اجل رسم نقطة واحدة فقط .. تحتوي على لون واكساء ...
فما بالك لو رسمنا مئات النقاط مالذي سيحصل ..
هل تريد ان تقول : نستخدم ال for loop .. صدقني تبقى مشكلة ..
بالتالي نحن نحمل الاحداثيات من ملفات خارجية ونحملها في مصفوفة ...
الملفات الخارجية قد ننشئها من خلال برامج الثريدي .. او ان ننشئها نحن بأنفسنا .
وأيضا.. **يجب أن نذكر أن استعمال الVertex Arrays هي أسرع وأكثر فعاليه**.

الخلاصة .. **نريد أن نطرح عدة أمثلة عن الVertex Array وسنناقشها** ...

وستكون سهلة وواضحة باذن الله . . **ولكن يجب أن تعلم أني قد رتبتها**

بطريقة تجعل كل مثال يعتمد على الذي قبله .. **وحاولت قدر الامكان الابتعاد**

عن كثرة الكلام دون أن يخل ذلك بالفائدة المرجوة من هذه الدروس .

ملاحظات هامة جدا :

١- يجب ان تملك أساسا جيدا لاستخدام المكتبة OpenGL.
٢- جميع الأمثلة الواردة في هذه الوثيقة هي مرفقة معها ومن المفترض ان تكون معها ..تم استخدام لغة السي بلس + GLUT – Win32 API .
بمعنى (اذا اردت الامثلة باستخدام المكتبة GLUT فقم بالذهاب الى المجلد GLUT Examples .. واذا أردت الامثلة باستخدام Win32 API فاذهب الى المجلد WIN32 Examples .. طبعا لايوجد تغيير بين كلا الطريقتين .. لكن البعض يستخدم تلك الطريقة والاخرى الطريقة الاخرى .

٣- قمت باختصار عدد من الاسطر وانا اعرض الامثلة في هذه الوثيقة . . حتى أقلل من حجم الوثيقة .. فمثلا في الغالب ستحتاج لاضافة هذه الاسطر الى الامثلة اذا كنت تريد تطبيقها بشكل صحيح

PDF created with pdfFactory trial version www.pdffactory.com

```
glLoadIdentity();
glTranslatef(0,0,-2);
```

القسم الأول :

استخدام Vertex Arrays مع الدالة **glArrayElement**

أولا خطوات عمل ال Vertex Array :

١- خزن احداثيات النقاط (او الاكساء او قيم الالوان ..الخ) في مصفوفة .
٢- اختر نوع العملية التي تريدها هل هي رسم نقاط او الوان .. .. وقم بتفعيلها.. نحن الان سنختار النقاط vertex فقط
٣- حول تلك القيم الموجودة في المصفوفة الى رسم او لون او إحداثيات للاكساء ..

جميع الامثلة التي سنكتبها هنا .. قم بوضعها في دالة الرسم عندك ..
مثلا أنا وضعت دالة الرسم وسميتها Render()

مثال ١)

```
GLfloat vertex []={ -0.3,0,0  , 0,0.3,0 , 0.3,0,0  };

glEnableClientState (GL_VERTEX_ARRAY);

glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);

glPointSize(3);

glBegin(GL_POINTS);

glArrayElement(0);
glArrayElement(1);
glArrayElement(2);

glEnd();
```

السطر الاول :
كنا في السابق نكتب شيئا شبيها بالتالي :

```
glBegin(GL_POINTS );
glVertex3fv ( -0.3, 0,0  );
glVertex3fv (  0,0.3,0  );
```


PDF created with pdfFactory trial version www.pdffactory.com

```
glVertex3fv ( 0.3,0,0    );
glEnd();
```

الان سنخزن النقاط في مصفوفة .

```
GLfloat vertex  [] ={ -0.3,0,0  , 0,0.3,0  , 0.3,0,0};
```

السطر الثاني :

الان سنفعل vertex arrays يعني نخبر opengl باننا نريد استخدام vertex arrays وذلك عن طريق الدالة
glEnableClientState
هذه الدالة تفعل نوع المصفوفة التي نريد وتأخذ عدة قيم .. نحن الان سنكتفي بقيمة واحدة .
GL_VERTEX_ARRAY

السطر الثالث:

الان نريد ان نخبر opengl بالمصفوفة التي تحمل القيم + نريد ان نخبر opengl بنوع تلك المصفوفة هل هي من نوع float أو int الخ .. + نريد ان نخبر opengl بعدد عناصر المصفوفة ..الخ

كيف نقوم بذلك ..
الجواب عن طريق الدالة

```
glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);
```

- **اول بارمتر تحدد الحجم .. وهو يختلف باختلاف النوع**

الان سنتلكم عن vertex .. بما انك تستخدم الدالة ***glVertexPointer*** فانك تكتب في البارمتر الاول اما ٢ او ٣ او ٤ ..
لكن ايهن تكتب ..
اذا كنت تريد استخدام احداثيين x و y فقط تكتب ٢ اذا اردت ثلاث احداثيات x . y . z تكتب ٣
اذا اردت اربع احداثيات وهذا نادر x.y.z.w تكتب ٤ ..

نحن الان كتبنا في المصفوفة ثلاث مواقع ..

-0.3,0,0
وتمثل الموقع الاول(النقطة الاولى) ويتكون من ثلاث احداثيات

0,0.3,0
وتمثل الموقع الثاني وتتكون من ثلاث احداثيات

0.3,0,0

PDF created with pdfFactory trial version www.pdffactory.com

وتمثل الموقع الثالث وهي ثلاث احداثيات ..
اذا نحن نستخدم الان ثلاث احداثيات اذا نكتب في البارمتر الاول ٣
ومن الممكن ان نكتفي باحداثيين اذا اردنا 2D ..
مثلا
-0.3,0
0,0.3
0.3,0
يعني استغنينا عن z فنكتب في البارمتر الاول ٢

لكن لاتضع بعض عناصر المصفوفة احداثيات ٣ وبعضها احداثيين .. ونحو ذلك .. التزم بطريقة محددة ..

- **البارمتر الثاني نوع المصفوفة** ..

نحن كتبنا

```
GLfloat vertex [] ={ -0.3,0,0 , 0,0.3,0 , 0.3,0,0 };
```

اذا النوع GLfloat يقابله GL_FLOAT

وهذه الانواع وما يقابلها ..

النوع	مايقابله
GLfloat	GL_FLOAT
GLdouble	GL_DOUBLE
GLint	GL_INT
GLshort	GL_SHORT
GLbyte	GL_BYTE
GLubyte	GL_UNSIGNED_BYTE
GLushort	GL_UNSIGNED_SHORT
GLuint	GL_UNSIGNED_INT

- **البارمتر الثالث** ضعه صفر .. في الغالب لن تحتاج اليه .

- **البارمتر الاخير** عنوان المصفوفة التي تتعامل معها

انت الان ضعها
&vertex[0]

وفي تطبيق قادم سنفصل اكثر ،

اذا الصورة النهائية

PDF created with pdfFactory trial version www.pdffactory.com

```
glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);
```

السطور الاخيرة من الكود السابق رسمنا فيه النقطة

الان سنرسم ..
ونحدد نوع الرسم الذي نريده .. اخترنا النقاط ،،

```
glBegin(GL_POINTS);
 glArrayElement(0);
glArrayElement(1);
 glArrayElement(2);
glEnd();
```

الان لماذا كتبنا صفر .. او واحد الخ ..

الدالة glArrayElement تقوم باختيار المجموعة التي تريد من المصفوفة

لاحظ الصورة

طبعا كل مجموعة مكونة من ثلاث ارقام لاننا نحن من امرنا بذلك .. في الدالة

glVertexPointer .. كتبنا في البارمتر الاول ٣ ...

الان لو قلنا

```
glArrayElement(٢);
```

هذا امر منا الى OPENGL نقول فيه ..

قم برسم نقطة احداثياتها هي احداثيات المجموعة رقم ٢ وهي اخر مجموعة
ثم نذهب الى المجموعة الرقم ٢ لنرى احداثياتها ..
0.3,0,0
اذا سيتم الرسم في الجهة اليمنى ..

وهكذا ،،

PDF created with pdfFactory trial version www.pdffactory.com

اتمنى ان تكون الامور توضحت الان

انتهى المثال الأول .

مثال ٢)

```
GLdouble vertex [] ={ -0.3 , 0   ,  0.3 , 0   };

glEnableClientState (GL_VERTEX_ARRAY);

glVertexPointer(2 ,GL_DOUBLE,0,&vertex[0]);

glBegin(GL_LINES);

glArrayElement(0);
glArrayElement(1);

glEnd();
```

الان اجرينا بعض التغييرات حتى تتنوع الامثلة ويحصل فهم اكثر ..

في السطر الاول: النوع GLdouble لذا وضعنا في الدالة glVertexPointer - GL_DOUBLE

ثانيا نحن الان استغنينا عن Z واكتفينا باحداثيين في المصفوفة :
-0.3 , 0
0.3 , 0

وكتبنا في البارمتر الاول في الدالة glVertexPointer - الرقم ٢
لاننا نتعامل مع احدثيين هما السيني والصادي

ثالثا : رسمنا الخط وهو يحتاج الى نقطتين

```
glBegin(GL_LINES);
 glArrayElement(0);
 glArrayElement(1);
glEnd();
```

النتيجة رسم خط .. جميل :)

الان اعتقد اصبح لديك المرونة الكافية ..

PDF created with pdfFactory trial version www.pdffactory.com

سأحاول ان اعقدك قليلا ..

نرجع لمثالنا السابق

```
GLfloat   vertex  [] ={ -0.3,0,0  , 0,0.3,0  , 0.3,0,0  };

glEnableClientState (GL_VERTEX_ARRAY);

glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);

glPointSize(3);

   glBegin(GL_POINTS);
      glArrayElement(0);
      glArrayElement(1);
      glArrayElement(2);
   glEnd();
```

الان ماذا يعني &vertex[0] في الدالة glVertexPointer

يعني اننا نريد ان نبدأ من اول المصفوفة .. وهو -0.3,0,0

لكن لنغير الى

glVertexPointer(3 ,GL_FLOAT,0,&vertex[١]);

هنا بدأنا من المجموعة الثانية في المصفوفة وهي 0,0.3,0
وبالتالي تعتبر المجموعة 0,0.3,0 هي اول المجموعة ...
يعني لو كتبنا

```
glBegin(GL_POINTS);

glArrayElement(0);

glEnd();
```

فسيتم رسم نقطة احدثياتها هي 0,0.3,0 وليس -0.3,0,0
اتمنى ان تكون الامور قد توضحت الان ..

هنا ينتهي القسم الأول من الموضوع ..

PDF created with pdfFactory trial version www.pdffactory.com

القسم الثاني :

استخدام Vertex Arrays مع الدالة **glArrayElement**

تعرفنا على الدالة glArrayElement لكن يوجد عدد من الدوال الاخرى التي تستخدم مع Vertex Arrays ..
كل دالة لها مايميزها .. لذا انت من يقرر أي الدوال تستخدم ..

من هذه الدوال أيضاً : **glDrawElements** .

هذه الدالة تعتمد في عملها على مصفوفتين .

- المصفوفة الاولى التي ستخزن فيها النقاط .
- المصفوفة الثانية التي ستخزن فيها أعداد وترتيب النقاط.

هذه الدالة تطلب منك عدة بارمترات :

```
void glDrawElements(GLenum mode, GLsizei count, GLenum type,
void *indices);
```

البارمتر الأول :

نوع الرسم مثلاً : **GL_LINES - GL_POINTS ...الخ**

البارمتر الثاني:

عدد النقاط التي سترسمها .

كما تعلم GL_POINTS تحتاج الى نقطة واحدة اونقطتين او ثلاث نقاط الخ ..
GL_LINES تحتاج الى نقطتين .. ومضاعفاتها
GL_TRIANGLES تحتاج ثلاث نقاط ومضاعفاتها
وهكذا .. اكيد انك تعلم ذلك من خلال التطبيقات السابقة .

البارمتر الثالث:

تضع نوع المصفوفة التي **تمثل عدد وترتيب النقاط**.

وأمامك ثلاثة أنواع:

- **GL_UNSIGNED_BYTE** اذا كان نوع مصفوفتك GLubyte
- **GL_UNSIGNED_SHORT** اذا كان نوع مصفوفتك GLushort
- **GL_UNSIGNED_INT** اذا كان نوع مصفوفتك GLuint

البارمتر الرابع:

عنوان المصفوفة التي تم فيها **تخزين ترتيب وعدد النقاط**.

PDF created with pdfFactory trial version www.pdffactory.com

مثال ١ :

```
/*1*/   static GLfloat vertex  [] ={ 1,0   ,   0,0  };
/*2*/   static GLubyte d[]={0 , 1 };

/*3*/   void Render()
/*4*/   {
/*5*/   glClear(GL_COLOR_BUFFER_BIT |...);
/*6*/   glLoadIdentity();
/*7*/   glTranslatef(0,0,-6);

/*8*/   glEnableClientState (GL_VERTEX_ARRAY);
/*9*/   glVertexPointer(2 ,GL_FLOAT,0,&vertex[0]);

/*10*/ glDrawElements(GL_POINTS, 2, GL_UNSIGNED_BYTE, &d[0]);


/*11*/  }
```

الشرح :

سأشرح فقط الزيادات عن الامثلة السابقة :

السطر الأول : مصفوفة قمنا بتخزين احداثيات نقطتين فيها .. النقطة الاولى **1,0** والثانية **0,0** .

السطر الثاني : مصفوفة قمنا بتخزين أعداد وترتيب النقاط .
فمثلا العدد " صفر " : يشير الى مجموعة الاحداثيات الاولى وهي **1,0**
والعدد رقم " واحد " : يشير الى مجموعة الاحداثيات الثانية وهي **0,0**

السطر الثامن والتاسع: سبق أن شرحته في الامثلة السابقة ..

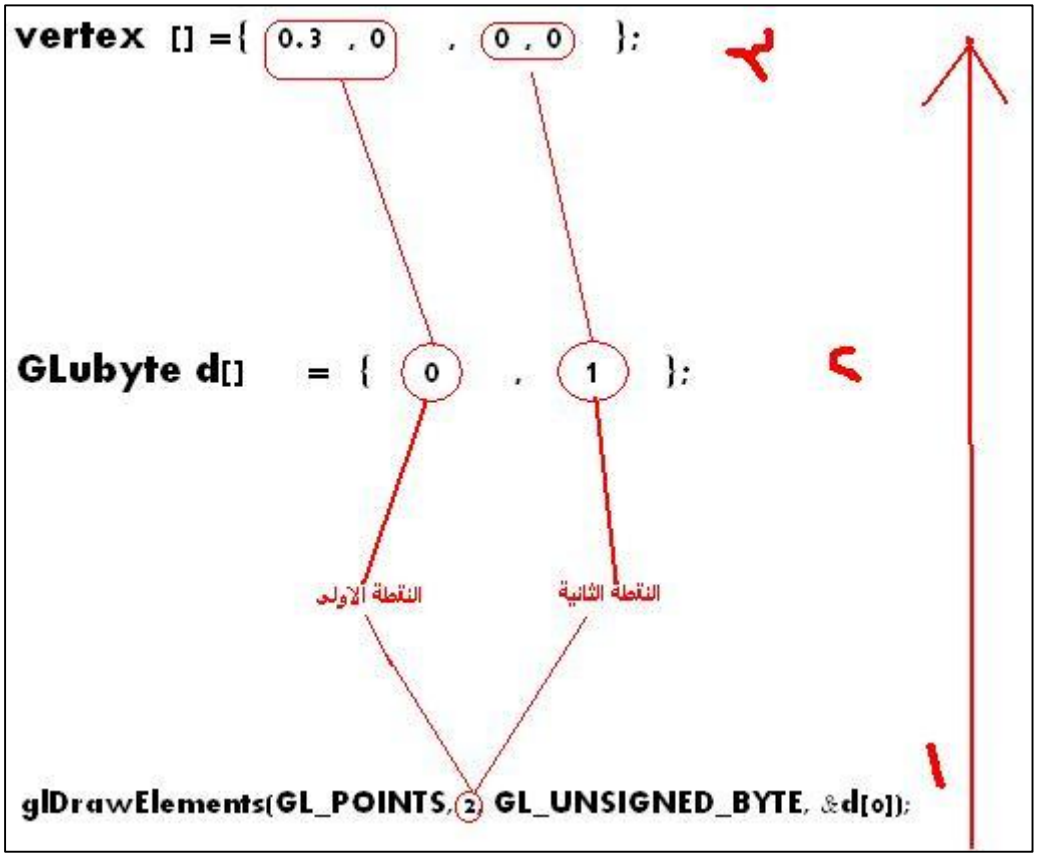
السطر العاشر: هذه الدالة تقوم برسم النقاط .. الموجودة في المصفوفة وقد شرحت بارمتراتها قبل قليل ،
حيث :

- حددنا نوع الرسم وهو **GL_POINTS** .
- حددنا عدد النقاط التي سنرسمها وهي نقطتين فقط .
- حددنا نوع المصفوفة التي تمثل ترتيب النقاط (المصفوفة التي اسمها d)
- حددنا عنوان المصفوفة التي تحمل ترتيب وعدد النقاط .

PDF created with pdfFactory trial version www.pdffactory.com

لاحظ أننا من خلال **glDrawElements** لا نقوم باالتصال مباشرة بالمصفوفة التي تحتوي على احداثيات النقاط وهي هنا **vertex []** ولكن نقوم بالاتصال بالمصفوفة التي اسمها d والتي تقوم بالاتصال بمصفوفة النقاط ..

لاحظ هذه الصورة التوضيحية :



ان شاء الله تكون الامور توضحت الى حد ما ..

انتهى المثال الأول .

PDF created with pdfFactory trial version www.pdffactory.com

مثال ٢ :

```
/*1*/    GLfloat vertex[]={ 1,1,0 , 0,0,0 , -1,-1,0 , 1,-1,0};
/*2*/    GLuint d[]={0 ,1 ,2 ,3  };

/*3*/    void Render()
/*4*/    {
/*5*/    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
/*6*/    glLoadIdentity();
/*7*/    glTranslatef(0,0,-6);

/*8*/    glEnableClientState (GL_VERTEX_ARRAY);
/*9*/    glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);

/*10*/   glDrawElements(GL_LINES, 4, GL_UNSIGNED_INT, &d[0]);

/*11*/   }
```

ماالذي تغير ؟

- اولا غيرنا المصفوفة من احداثيين الى ثلاث احداثيات وذلك حتى نتعود على ذلك .

GLfloat vertex[]={ 1,1,0 , 0,0,0 , -1,-1,0 , 1,-1,0};

كل مجموعة تكون احداثيات نقطة .

- واخبرنا opengl بذلك وذلك عن طريق الدالة
glVertexPointer(**3** ,**GL_FLOAT**,0,&vertex[0]);
قلنا نريد استخدام ثلاث احداثيات يعني كل ثلاث ارقام تكون مجموعة (نقطة).

- وغيرنا نوع المصفوفة الثانية الى GLuint بدلا من GLubyte وذلك حتى تتعود على هذه التغييرات فقط ..

GLuint d[]={0 ,1 ,2 ,3 };

وبالتالي غيرنا النوع في الدالة الى GL_UNSIGNED_INT هكذا
glDrawElements(GL_LINES, **4**, **GL_UNSIGNED_INT**, &d[0]);

PDF created with pdfFactory trial version www.pdffactory.com

- ايضا لاحظ اننا نريد رسم خط ..
كل نقطتين تمثلان خط ..
اذا اربع نقاط تمثل خطين .. وهكذا .

لاحظ المصفوفة ..

static GLuint d[]={0 , 1 ,2 ,3 };

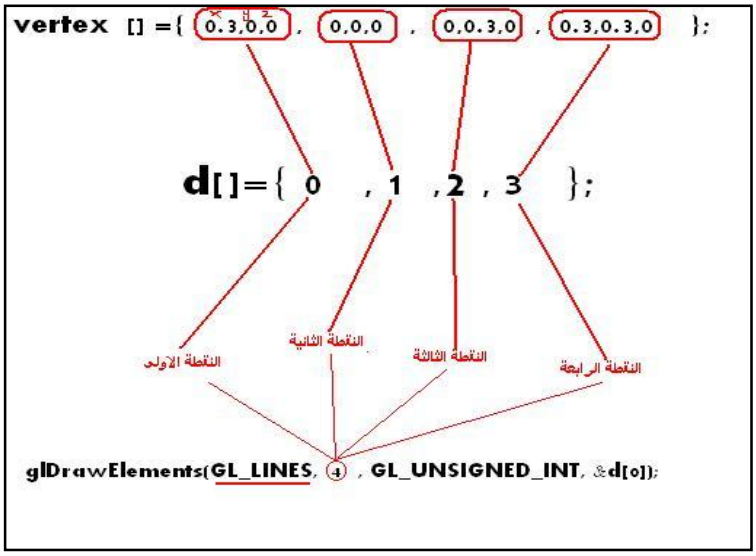
الرقم صفر يعني .. النقطة الاولى .. وهي العناصر الثلاثة الاولى في المصفوفة vertex وهي: 1,1,0

الرقم ١ يعني .. النقطة الثانية .. وهي العناصر الثلاثة الثانية 0,0,0

الرقم ٢ .. يعني ..النقطة الثالثة ..وهي العناصر الثلاثة الثالثة -1,-1,0

الرقم ٣ ..يعني .. النقطة الرابعة .. وهي العناصر الثلاثة الاخيرة 1,-1,0
سيتم الوصل بين النقطة صفر و واحد وبين النقطة ٢ و ٣ ..

لاحظ الصورة:



إضافة:

في نفس هذا المثال .. لو غيرنا السطرين :

```
GLfloat vertex[]={ 1,1,0 , 0,0,0 , -1,-1,0 , 1,-1,0};

glDrawElements(GL_LINES, 4, GL_UNSIGNED_INT, &d[0]);
```

الى - > :

```
GLfloat vertex  [] ={  1,1,0 , 0,1,0 , 0,0,0 , 1,0,0};
glDrawElements(GL_QUADS, 4, GL_UNSIGNED_INT, &d[0]);
```

ستجد اننا رسمنا مربع .. حيث غيرنا في احداثيات النقاط وفي طريقة الرسم

PDF created with pdfFactory trial version www.pdffactory.com

انتهى المثال الثاني .

مثال ٣ :

هذا مثال لرسم مثلثين .. الهدف منه زيادة في الفهم :

```
/*1*/  GLfloat vertex  [] ={  1,0,0  ,  0,0,0   ,  0,1,0 ,

                              -1,-1,0 ,   0,-1,0 ,   0,-0.1,0 };

/*2*/   GLuint d[]={0 , 1 ,2 ,3, 4 ,5  };

/*3*/  void Render()
/*4*/  {
/*5*/      glClear( GL_COLOR_BUFFER_BIT ..... );
/*6*/      glLoadIdentity();
/*7*/      glTranslatef(0,0,-6);
/*8*/
/*9*/        glEnableClientState (GL_VERTEX_ARRAY);
/*10*/        glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);
/*11*/ glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, &d[0]);
/*12*/
/*13*/ }
```

حددنا النقاط ..

العناصر الثلاثة الاولى من المصفوفة تمثل المثلث الاول .
والعناصر الثلاثة الاخيرة من المصفوفة تمثل المثلث الاخير .

طبعا أخبرنا OpenGL بكامل التفاصيل .. حيث أخبرناه بأننا نريد طريقة رسم GL_TRIANGLES وأننا نريد رسم ٦ نقاط .

والنتيجة رسم مثلثين .

ملاحظة أخيرة :

لو غيرت السطر رقم ١١ من المثال السابق الى :

glDrawElements(GL_TRIANGLES, **3**, GL_UNSIGNED_INT, &d[**3**]);

فهذا يعني أنك تريد رسم ثلاث نقاط اعتبارا من العنصر الثالث في المصفوفة .
(تذكر ان عد عناصر المصفوفة يبدأ من صفر) .

PDF created with pdfFactory trial version www.pdffactory.com

انتهى المثال الثالث .

القسم الثالث :

استخدام Vertex Arrays مع الدالة **glDrawArrays**

هذه الدالة بسيطة عملية أيضا .

خطوات العمل مع هذه الدالة :

١ – تعلن عن مصفوفة واحدة تحتوي على النقاط .

٢ – تفعّل الVertex Arrays

٣ – تستخدم الدالة **glDrawArrays**.

صيغة الدالة glDrawArrays :

void glDrawArrays(GLenum mode, GLint first, GLsizei count);

البارمتر الأول : نوع الرسم ..هل هو GL_POINTS او GL_LINES الخ.

البارمتر الثاني : ماهو ترتيب **اول مجموعة** احداثيات في المصفوفة.

البارمتر الثالث : : ماهو **عدد النقاط** التي تريد رسمها.

مثال ١ :

```
/*1*/  GLfloat vertex [] ={  1,0,0 , 0,0,0 ,  0,1,0 ,
                            -1,-1,0 , 0,-1,0 , 0,-0.1,0 };



/*2*/  void Render()
/*3*/  {
/*4*/  glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
/*5*/  glLoadIdentity();
/*6*/  glTranslatef(0,0,-6);

/*7*/  glEnableClientState (GL_VERTEX_ARRAY);
/*8*/  glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);
/*9*/  glDrawArrays(GL_TRIANGLES, 0, 6);
/*10*/ }
```


PDF created with pdfFactory trial version www.pdffactory.com

لايوجد جديد إلا في السطر التاسع ..

السطر التاسع : استخدمنا الدالة **glDrawArrays** لرسم النقاط الموجودة في المصفوفة ..

قلنا نريد أن نرسم ابتداءً من أول مجموعة في المصفوفة وهي **1,0,0**
وفي البارمتر الاخير حددنا عدد النقاط التي نريد رسمها وهي ست نقاط .

والنتيجة رسم مثلثين .. كل مثلث له ثلاث نقاط .

انتهى المثال الأول .

إضافة :
تستطيع مثلا ان تقول اريد الرسم من ثالث مجموعة وذلك في ثلاث نقاط يعني هكذا :

glDrawArrays(GL_TRIANGLES, 3, 3);

والنتيجة رسم ثلاث نقاط لتشكل لنا مثلث .

لو غيرت السطر التاسع إلى :

```
glDrawArrays(GL_LINES, 2, 2);
```

سيت حينها رسم خط .. إعتبارا من ثالث مجموعة في المصفوفة (تذكر ان عد عناصر الصفوفة يبدأ من الصفر) وهي هنا **0,1,0**
وسنرسم نقطتين .. فقط ..

بالتالي OpenGL ستقوم بشيء شبيه بالتالي :

```
glBegin(GL_LINES);
glVertex3f(0,1,0);
glVertex3f(-1,-1,0);
glEnd();
```

أتمنى تكون الصورة قد توضحت ،

تنبيه !

- ***GL_POINTS ترسم نقطة نقطة ... يعني تقدر تكتب في البارمتر الاخير في الدالة glDrawArrays: ١ او ٢ او ٣ او ٤ الخ ..***
- ***GL_LINES ترسم في كل نقطتين خط يعني في البارمتر الاخير تكتب ٢ او ٤ او ٦ الخ***
- ***GL_TRIANGLES كل ثلاث نقاط شكل اذا تكتب ٣ او ٦ او ٩ الخ .. وذلك في البارمترالاخير .***
- ***GL_QUADS فتكتب ٤ او ٨ او ١٢ الخ .***

PDF created with pdfFactory trial version www.pdffactory.com

- ***GL_POLYGON تكتب من ٣ فما فوق ..***

هذا من اساسيات opengl ولكن المبرمج كثير النسيان على ما اعتقد !!!

الفصل الثاني .. الـVertex Arrays والاكساء والالوان ..

تعلمنا في الاول كيف نتعامل مع الـVertex Array اذا كنا نريد رسم النقاط ،،
وعرفنا أن الهدف الرئيسي من الـVertex Array هو السهولة في الرسم .. بحيث أننا نخزن نقاط الرسم في مصفوفة ثم نقوم بالرسم .

أيضا بالمثل ... يمكننا تخزين قيم الالوان أو احداثيات الاكساء في مصفوفة ثم الرسم .. بنفس الطريقة التي كنا نستخدمها في الفصل الاول .

بالتالي هذا الفصل هو يعتمد بشكل رئيسي على فهمك للفصل الاول .

أولا : الألوان :

كنت أنت في السابق ترسم شكل مربع وتقوم بتلوينه بهذا الشكل ... مثلا :

```
void Render()
{
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
glLoadIdentity();
glTranslatef(0,0,-6);

glColor3f(1,0,1);
glBegin(GL_QUADS);
glVertex3f (1,1,0);
glVertex3f (-1,1,0);
glVertex3f (-1,-1,0);
glVertex3f (1,-1,0);
glEnd();
}
```

الان لو أردنا تحويل هذا المثال إلى Vertex Arrays .. ماذا سنفعل ؟

جواب :

نحن بحاجة لأمرين ... بما أننا نريد استخدام الـVertex Arrays مع النقاط والالوان فيجب أن نفعل كلا الامرين . . هكذا :

glEnableClientState (**GL_VERTEX_ARRAY**);
glEnableClientState (**GL_COLOR_ARRAY**);

ثم نخزن قيم الالوان في مصفوفة

```
glColor3f(1,0,1); ------> GLfloat color_array[]={ 1, 0, 1};
```

لاحظ أخي الكريم ... المصفوفة .. نوعها .. والقيم التي تأخذها أنت من يحددها ..
فلو أردت استخدام الدالة **glColor3f** فإن المصفوفة التي سنستخدمها بدلا من الدالة هي كالتالي :

PDF created with pdfFactory trial version www.pdffactory.com

- نوع المصفوفة GLfloat .
- القيم التي تأخذها المصفوفة (المدى) من صفر إلى واحد .
- يوجد ثلاث قنوات للالوان ..RGB .

أما لو أردت الاستعاضة عن الدالة **glColor4f** بمصفوفة فلاحظ أنه :
- نوع المصفوفة GLfloat .
- القيم التي تأخذها المصفوفة (المدى) من صفر إلى واحد .
- يوجد أربع قنوات للالوان ..RGBA .

أما لو أردت الاستعاضة عن الدالة **glColor4ub** بمصفوفة فلاحظ أنه :
- نوع المصفوفة GLubyte.
- القيم التي تأخذها المصفوفة (المدى) من صفر إلى ٢٥٥ .
- يوجد أربع قنوات للالوان ..RGBA .

وهكذا .. يوجد عدة صيغ لدالة الالوان في opengl .. كل صيغة تستطيع أنت ان تحولها بعد الممارسة والاتقان ،،

----------==========----------

لاحظ أخي أننا فعلنا الـVertex Arrays للالوان والنقاط .
وأيضا خزنا النقاط والالوان في مصفوفة ...

بقي شيء أخير .. وهو أن نخبر Opengl بنوع المصفوفة .. عدد القنوات التي نستخدم ... وذلك عن طريق الدالة glColorPointer .

مثال ١ :

لو أردنا تحويل الكود

```
void Render()
{
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
glLoadIdentity();
glTranslatef(0,0,-6);

glBegin(GL_QUADS);
glColor3f(1,0,1);
glVertex3f (1,1,0);
glColor3f(1,0,0;
glVertex3f (-1,1,0);
glColor3f(0,0,1);
glVertex3f (-1,-1,0);
glColor3f(1,1,0);
glVertex3f (1,-1,0);
glEnd();
}
```


PDF created with pdfFactory trial version www.pdffactory.com

الى Vertex Array نكتب التالي

```
/*1*/  GLfloat vertex[] ={ 1,1,0  , -1,1,0 ,  -1,-1,0 ,1,-1,0 };
/*2*/  GLfloat color [] ={  1,0,1 , 1,0,0   ,  0,0,1   ,1,1,0 };
/*3*/  void Render()
/*4*/  {
/*5*/  glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
/*6*/  glLoadIdentity();
/*7*/  glTranslatef(0,0,-12);

/*8*/  glEnableClientState (GL_VERTEX_ARRAY);
/*9*/  glEnableClientState (GL_COLOR_ARRAY);

/*10*/ glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);
/*11*/ glColorPointer(3 ,GL_FLOAT,0,&color[0]);

/*12*/ glDrawArrays(GL_QUADS, 0, 4);

/*13*/ }
```

السطر الثامن والتاسع :
فعلنا مصفوفة النقاط والالوان .

السطر الحادى عشر:
أخبرنا OpenGL بخصائص مصفوفة الالوان حيث المتطلبات هي :

```
glColorPointer(3 ,GL_FLOAT,0,&color[0]);
```

البارمتر الاول : عدد قنوات اللون .. هل هي RGBفنكتب ٣ . . أو RGBA فنكتب ٤ .

البارمتر الثاني : نوع المصفوفة .

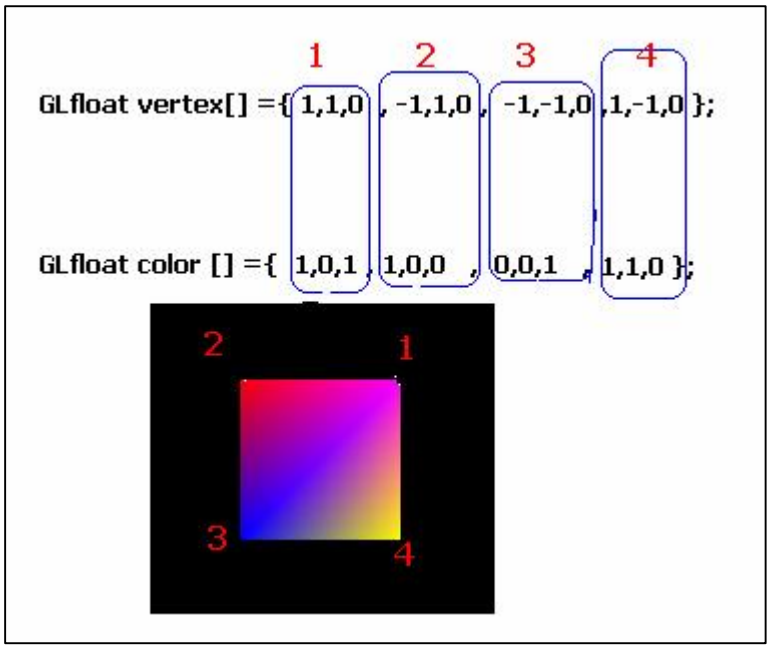
البارمتر الاخير عنوان أول عنصر في المصفوفة .

السطر الثانى عشر :

نرسم النقاط الموجودة في المصفوفة vertex و color باستخدام الدالة
glDrawArrays والتي سبق شرحها في الفصل الأول .
ولكن نستطيع الرسم بطريقة أخرى .. مثلا باستخدام الدالة **glDrawElements** أو
glArrayElement .. وكلها تم شرحها .

PDF created with pdfFactory trial version www.pdffactory.com

شاهد هذه الصورة التوضيحية التي تشرح كيف يعمل OpenGL مع الكود السابق .



مثال ٢ :

```
/*1*/ GLfloat vertex [] ={ 1,1,0 , -1,1,0 , -1,-1,0 ,1,-1,0 };
/*2*/ GLubyte color [] ={  0,0,255,255 , 255,0,0,255   ,
                            0,0,255,255   ,255,255,0,255 };
/*3*/ void Render()
/*4*/ {
/*5*/    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
/*6*/    glLoadIdentity();
/*7*/    glTranslatef(0,0,-12);

/*8*/    glEnableClientState (GL_VERTEX_ARRAY);
/*9*/    glEnableClientState (GL_COLOR_ARRAY);

/*10*/    glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);
/*11*/   glColorPointer(4 ,GL_UNSIGNED_BYTE,0,&color[0]);

/*12*/    glDrawArrays(GL_QUADS, 0, 4);

/*13*/    }
```


PDF created with pdfFactory trial version www.pdffactory.com

الاختلاف هنا أننا استخدمنا ما يقابل الدالة glColor4ub .
حيث رلدينا أربع قنوات RGBA . . كل قناة مداها 255 – 0 .
ونوع المصفوفة هنا أيضا تغير .

PDF created with pdfFactory trial version www.pdffactory.com

القسم الثاني: الاكساء و الVertex Arrays

بنفس الطريقة التي تعاملنا فيها مع الألوان .. نتعامل مع الاكساء .
سأفرض أنك تعرف طريقة االكساء .. كيفية تعيين الاحداثيات .. وغيرها من الامور الاساسية.

فمثلا لو أردنا رسم مربع واكشائه سنكتب شيئا شبيها بالتالي :

```
void Render()
{
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
  glLoadIdentity();
  glTranslatef(0,0,-12);

            glEnable(GL_TEXTURE_2D) ;
            glBindTexture(GL_TEXTURE_2D,ID);

            glBegin(GL_QUADS);
            glTexCoord2f(0,0);
            glVertex3f(-1,-1,0);
            glTexCoord2f(1,0 );
            glVertex3f(1,-1,0);
            glTexCoord2f(1,1);
            glVertex3f( 1, 1,0);
            glTexCoord2f(0,1);
            glVertex3f(-1,1,0);
            glEnd();

}
```

لو أردنا تحويل الكود السابق إلى Vertex Arrays .. فهو كالتالي :

```
  /*1*/ unsigned int ID ;
  /*2*/ GLfloat vertex [] ={ -1,-1,0 , 1,-1,0 ,  1,1,0 ,-1,1,0 };
  /*3*/ GLfloat texture [] ={ 0,0  , 1,0  ,1,1  , 0,1 };

/*4*/  void Render()
  /*5*/ {
  /*6*/ glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
  /*7*/ glLoadIdentity();
  /*8*/ glTranslatef(0,0,-12);

  /*9*/  glEnable(GL_TEXTURE_2D) ;
  /*10*/ glBindTexture(GL_TEXTURE_2D,ID);

  /*11*/    glEnableClientState (GL_VERTEX_ARRAY);
  /*12*/    glEnableClientState (GL_TEXTURE_COORD_ARRAY);

  /*13*/    glVertexPointer(3 ,GL_FLOAT,0,&vertex[0]);
  /*14*/    glTexCoordPointer(2 ,GL_FLOAT,0,&texture[0]);

  /*15*/    glDrawArrays(GL_QUADS, 0, 4);
  /*16*/    }
```


PDF created with pdfFactory trial version www.pdffactory.com

الشرح:

السطر الثاني و الثالث:
قمنا بتخزين قيم النقاط والاكساء (الاحداثيات).

السطر ١١ – ١٢ :
فعلنا الVertexArrays للاكساء و النقاط .

السطر ١٤ :
حددنا خصائص مصفوفة الاكساء ..

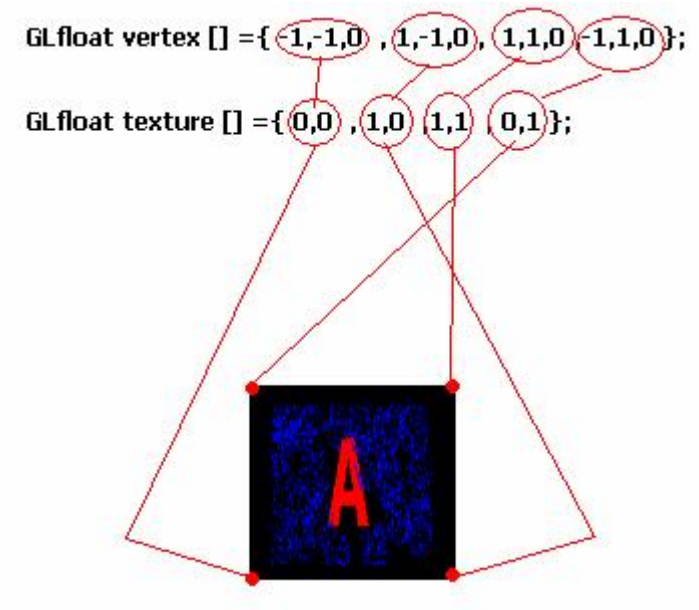
```
glTexCoordPointer(2 ,GL_FLOAT,0,&texture[0]);
```

حيث البارمتر الاول هو نوع احداثيات الاكساء .. احداثي واحد أو اثنين أو ثلاثة .. في الغالب نحن نحتاج إلى احداثيين فقط .

البارمترات الباقية أظن واضحة .

النتيجة رسم مربع + اكسائه .

وهذه الصورة توضح كيف قام OpenGL بذلك .



هل الأمر يحتاج مثال آخر ... لا أظن .. إلال أنه من البديهي أنك أيضا تستطيع استخدام طرق الرسم الاخرى بدلا من `glDrawArrays` ..

الامر سهل جدا فقط طبق عدة أمثلة وستفهم بكل تأكيد .

PDF created with pdfFactory trial version www.pdffactory.com

الان تعلمنا كيف نستخدم الVertex Arrays من أجل التعامل مع النقاط .. والالوان .. واحداثيات الاكساء .. ولكن أيضا بقي الكثير .. **ولكنها كلها بنفس الاسلوب** .. **وهذا السبب الذي يجعل الكتب لاتركز كثيرا على الVertex Arrays** ..

فمثلا تستطيع استخدام الNormal مع الVertex Arrays ..GL_NORMAL_ARRAY

وكلها فقط تعتمد على الفصل الاول من هذه الوثيقة .. متى ما أتقنت الفصل الاول فإنك بإذن الله قادر على تطبيق الفكرة على اكثر من صعيد ...

تم بحمد الله .

PDF created with pdfFactory trial version www.pdffactory.com

PDF created with pdfFactory trial version www.pdffactory.com