www.startimes2.com

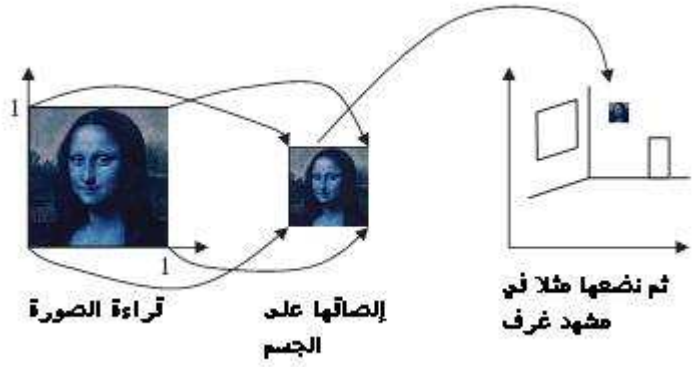
دروس في الـ OpenGL

الدرس 3: الإكساء الإضاءة ومزج الألوان + رائعة

khatibe_30@hotmail.fr

2010

إلى حد الآن, رسمنا بعض الأشكال و قمنا بتضليلها و إضاءتها و تلوينها, باستخدام الإكساء سنضيف واقعية على تصاميمنا, مثلا, إذا أردنا تصميم السماء فإننا نرسم مربعا و نلصق عليه صورة لسماء حقيقية فتظهر وكأنها حقيقية, إذا أول ما سنقوم به هو قراءة صورة نقطية –في البداية سنتعامل مع الصور من النوع *.bmp- ونخزنها على شكل مصفوفة نقطية معروفة الأبعاد ليكون لذلك الإكساء مقبض يعرفه, و باستخدام مقبض الإكساء نضع الصورة في مكان على المشهد و نغلف بها أي جسم نريده.



وكما لاحظت في الشكل السابق, فإن الإكساء يملك إحداثيات نحدد بها المساحة التي نريد استخدامها من الصورة على شكل إكساء, تمتد تلك الإحداثيات من (0,0) إلى (1,1).

بعد هذه المقدمة البسيطة, أنشئ مشروع جديد من نوع Console وأضف إليه ملف فارغ و سمه main.cpp, بعد ذلك أكتب الكود الذي ينشئ نافذة OpenGL فارغة تغلق عند الضغط على المفتاح ESC, رأينا هذا في الدروس السابقة, وإن كنت لا تذكر الكود جيدا فهذا ما يجب أن تكتبه على الملف main.cpp:

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

void Reshape(int w, int h)
{
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(45.0, (float)w/(float)h, 1.0, 100.0);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	glutSwapBuffers();
}

void Key(unsigned char key, int x, int y )
{
	if(key ==  27 )   exit(0);
}
```

```
void action(void)
{
	Display();;
}

bool init(void)
{
	glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
	glClearDepth(1.0f);
	glEnable(GL_DEPTH_TEST); // نفعل الرسم في العمق -إن صح التعبير- وهذا حتى تغطي الأجسام القريبة الأجسام البعيدة
	return TRUE;
}


void main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
	glutInitWindowSize(500, 500);
	glutInitWindowPosition (0,0);
	glutCreateWindow("Texture");
	init();
	glutDisplayFunc(Display);
	glutReshapeFunc(Reshape);
	glutKeyboardFunc(Key);
	glutIdleFunc(action);
	glutFullScreen();
	glutMainLoop();
}
```

بعد ذلك سنرسم مربع أبيض وسط الشاشة لنكسوه بصورة لاحقا, إذا و داخل الدالة Display نكتب:

```
void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	glTranslatef(0.0f,0.0f,-5.0f); //(0,0,-5) نزيح مركز المعلم إلى
	glBegin(GL_QUADS); //سنرسم بين كل أربع نقاط مربع
		glVertex3f(-1.0f, -1.0f,  1.0f); //(-1,-1,1) النقطة
		glVertex3f( 1.0f, -1.0f,  1.0f); //(1,-1,1)النقطة
		glVertex3f( 1.0f,  1.0f,  1.0f); //(1,1,1)النقطة
		glVertex3f(-1.0f,  1.0f,  1.0f); //(-1,1,1)النقطة
	glEnd();
	glutSwapBuffers();
}
```

نفذ البرنامج لتحصل على مربع أبيض كبير وسط الشاشة, نريد الآن إكساء ذلك المربع بصورة نقطية, لذلك سنقوم أولا بقراءة الصورة و سنحتاج إلى استعمال المكتبة glaux.h, سننشئ دالة جديدة نسميها LoadBMP() لتقوم بتحميل الصورة إلى مصفوفة :

```
#include <stdlib.h>
#include <windows.h>
#include <gl\glaux.h> //نحتاج إلى دوال هذه المكتبة
#include <GL\glut.h>
#include <stdio.h>
#include <math.h>
#include <stdio.h>
#pragma comment(lib, "GLAUX.LIB") // glaux.lib نربط البرنامج بالمكتبة

unsigned int      texture[2]; // هذه مصفوفة من عنصرين ستحتوي على معرف الإكساء و
هنا سنقوم بإنشاء إكسائين

AUX_RGBImageRec *LoadBMP(char *Filename) // الدالة التي ستقرأ بيانات الصورة و
سنعطيها كبارامتر مسار الصورة
{
      FILE *File=NULL;
      if (!Filename) //نتأكد من أن إسم الصورة غير خالي
            return NULL;
      File=fopen(Filename,"r"); //نفتح الملف لنتأكد أنه قابل للفتح
      if (File) //إذا استطعنا فتح الملف
      {
            fclose(File); //نغلق الملف
            return auxDIBImageLoad(Filename); // نحمل الصورة المطلوبة بهذه الدالة
و نعيد ما تعيده تلك الدالة
      }
      return NULL;
}
//...
```

إستخدمنا الدالة auxDIBImageLoad والتي تحمل الصورة إلى تركيبة من النوع AUX_RGBImageRec وهي معرفة كالآتي:

```
typedef struct _AUX_RGBImageRec {
   GLint sizeX, sizeY; // طول و عرض الصورة
   unsigned char *data; // المصفوفة التي تحتوي بايتات الصورة
} AUX_RGBImageRec;
```

طبعا لن نعرف تلك التركيبة فهي معرفة في الملف الرأسي glaux.h.

بعد أن حصلنا على بيانات الصورة سننشئ إكساء من تلك الصورة, الدالة التي سنسميها LoadTexture تفعل ذلك, تأخذ هذه الدالة بارمترين, الأول عبارة عن سلسلة حرفية لمسار الملف و الثاني مؤشر لمتغير من نوع unsigned int والذي سنعيد فيه معرف الإكساء لنتمكن من استخدامه فيما بعد:

```
//...
int LoadTextures(char *filename, unsigned int *texture)
{
      int Status=FALSE; //متغير يحمل القيمة التي ستعيدها الدالة, إما نجح أو فشل
      AUX_RGBImageRec *TextureImage[1]; //التركيبة التي ستحمل بيانات الصورة
      memset(TextureImage,0,sizeof(void *)*1); // نحجز ذاكرة للمتغير الذي سيحمل
بيانات الصورة
      if (TextureImage[0]=LoadBMP(filename)) // نحمل الصورة باستعمال الدالة
السابقة
      {
```

```
            Status=TRUE; // إذا وصلنا إلى هذه النقطة فقد استطعنا تحميل الصورة
بنجاح و بالتالي نعيد القيمة -صحيح- إلى الدالة
            glGenTextures(1, texture); //إنشاء معرف واحد صحيح لإنشاء الإكساء
            glBindTexture(GL_TEXTURE_2D, *texture); // نربط المتغير الذي يحمل
معرف الإكساء بإكساء ثنائي الأبعد و هذا يعني أن العمليات القادمة ستؤثر على هذا
الإكساء
            glTexImage2D( //الدالة التي تنشئ إكساء
                  GL_TEXTURE_2D, //إكساء ثنائي الأبعاد
                  0, //0 مستوى التفاصيل في الصورة و نتركها عادة
                  3, //عدد مكونات البانات, لدينا أحمر أخضر و أزرق أي 3 مكونات
                  TextureImage[0]->sizeX, //عرض الصورة
                  TextureImage[0]->sizeY, //إرتفاع الصورة
                  0, //0 خجم حافة الصورة و سنتركها
                  GL_RGB, //مكونات الصورة : أحمر أخضر أزرق
                  GL_UNSIGNED_BYTE, //المكونات التي تشكل الصورة عبارة عن بايتات
                  TextureImage[0]->data); //بيانات الصورة
            //الآن سنحدد بارمترات الإكساء
            glTexParameteri(GL_TEXTURE_2D,
                  GL_TEXTURE_MAG_FILTER, // إذا سنقوم بتطبيق فيلتر علىالصورة
            وضعت على مساحة أكبر من الحجم الحقيقي للصورة
                  GL_LINEAR); // نوع الفيلتر, وباستعمال هذا النوع سيكون الإكساء
            جيد و بدقة عالية و لكن إذا كان جهازك ثقيل نوعا ما فاستعمل النوع
            GL_NEAREST الآخر وهو
            glTexParameteri(GL_TEXTURE_2D,
                  GL_TEXTURE_MIN_FILTER, // إذا سنقوم بتطبيق فيلتر علىالصورة
            وضعت على مساحة أصغر من الحجم الحقيقي للصورة
                  GL_LINEAR);
      }
      if (TextureImage[0])
      { // إذا نجحنا في تحميل الإكساء فإننا سنحرر كل الأماكن المحجوزة في الذاكرة أثناء
تحميل الإكساء
            if (TextureImage[0]->data)
                  free(TextureImage[0]->data);
            free(TextureImage[0]);
      }
      return Status;
}
//...
```

علينا الآن أن نعود إلى الدالة ()init ونحمل فيها الإكساء و نفعله, وبالتالي نغير تلك الدالة إلى:

```
bool init(void)
{
      if (!LoadTextures("wall.bmp", &texture[0])) //نحمل إكساء من الصورة الأولى
            return FALSE;
      if (!LoadTextures("star.bmp", &texture[1])) // نحمل إكساء آخر من صورة
أخرى
            return FALSE;
      glEnable(GL_TEXTURE_2D); // نفعل الإكساء ثنائي الأبعاد
      glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
      glClearDepth(1.0f);
      glEnable(GL_DEPTH_TEST);
      return TRUE;
}
```

إن كنت تتساءل عن الصورتين التين استخدمناهما في الإكساء فهما:

قم بنسخهما و لصقهما ثم أحفظهما على شكل bmp. أو استخدم غيرهما.

ولعرض الإكساء على جسم معين يجب ربط كل إحداثية من إحداثيات الإكساء (s,t) بما يقابلها من إحداثيات الجسم (s,y,z), و لفعل ذلك نستعمل : glTexCoord2f(s,t); glVertex3f(x,y,z) ;

إذا, سنكسو المربع الذي رسمناه سابقا بصورة الجدار, على مستوى الدالة Display :

```
void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	glTranslatef(0.0f,0.0f,-5.0f);
	glBindTexture(GL_TEXTURE_2D, texture[0]); // نحدد الإكساء الذي سنستخدمه الآن و هو الموجود في المتغير texture[0]
	glBegin(GL_QUADS);
		glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
		glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
		glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
		glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
	glEnd();
	glutSwapBuffers();
}
```

إذا وبعد تنفيذ البرنامج يصبح المربع مكسو بصورة الجدار:

وإذا أردنا إمالة الإكساء فإننا نختار مصفوفة الإكساء ثم نحركها و ندورها كيفما نشاء ثم نعود إلى مصفوفة الجسم, مثلا هنا سندور الإكساء :

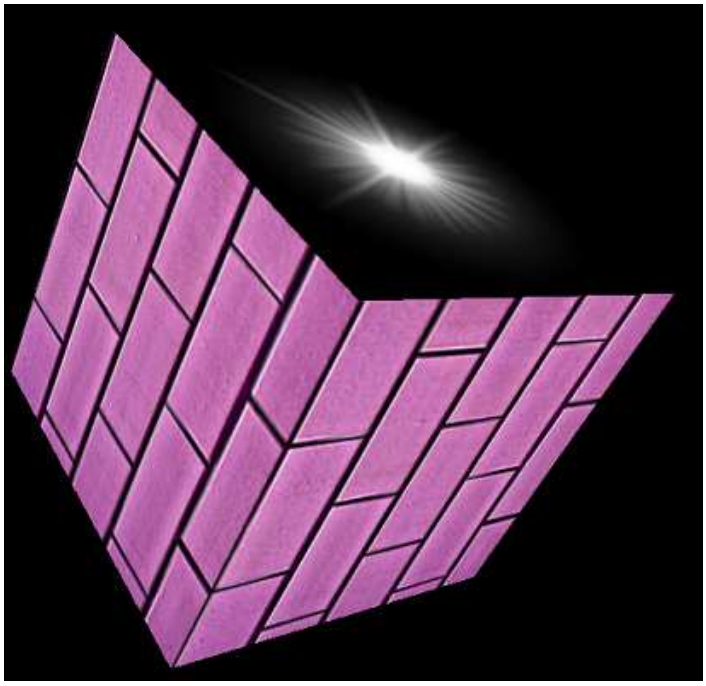
```
void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glMatrixMode(GL_TEXTURE); //نختار مصفوفة الإكساء
	glLoadIdentity(); //نرجعها إلى حالتها الإبتدائية في كل مرة
	glRotatef(45,0,0,1); //zندورها بزاوية 45 على المحور
	glMatrixMode(GL_MODELVIEW); // نعود إلى مصفوفة الجسم
	glLoadIdentity();
	glTranslatef(0.0f,0.0f,-5.0f);
	glBindTexture(GL_TEXTURE_2D, texture[0]);
	glBegin(GL_QUADS);
		glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
		glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
		glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
		glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
	glEnd();
	glutSwapBuffers();
}
//...
```

تنفذ البرنامج بعد ذلك التغيير يعطينا هذه النتيجة:

الآن سننشئ مكعبا يدور حول نفسه بحيث تكون 3 أوجه من أوجهه مكسوة بالجدار و الأخرى مكسوة بصورة النجمة,إذا نغير الدالة Display كالعادة و نضيف إليها الكود الذي يرسم بقية الأوجه الخمسة و يكسوها, ولكن قبل ذلك نضيف في أعلى الملف 3 متغيرات تحمل قيم الزوايا التي سيدور بها المكعب حول نفسه بالنسبة للمحاور الثلاث :

```
//...
float       xrot; //x زاوية الدوران حول
float       yrot; //yزاوية الدوران حول
float       zrot; //zزاوية الدوران حول
unsigned int      texture[2];
//...
void Display(void)
{
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glLoadIdentity();
      glTranslatef(0.0f,0.0f,-5.0f);
      glRotatef(xrot,1.0f,0.0f,0.0f); //x تدوير المكعب بالنسبة لـ
      glRotatef(yrot,0.0f,1.0f,0.0f); //y تدوير المكعب بالنسبة لـ
      glRotatef(zrot,0.0f,0.0f,1.0f); //z تدوير المكعب بالنسبة لـ
      glBindTexture(GL_TEXTURE_2D, texture[0]); //إختيار الإكساء الأول
      glBegin(GL_QUADS); //رسم الثلاث أوجه الأولى
            // الوجه الأمامي
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
            glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
            // الوجه الخلفي
            glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
            // الوجه الأعلى
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
            glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
      glEnd();
      glBindTexture(GL_TEXTURE_2D, texture[1]); //إختيار الإكساء الثاني
      glBegin(GL_QUADS); //رسم الثلاث أوجه الأولى
            // الوجه الأسفل
            glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
            // الوجه الأيمن
            glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
            // الوجه الأيسر
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
      glEnd();
      xrot+=0.3f; //زيادة زاوية الدوران
      yrot+=0.2f; //زيادة زاوية الدوران
      zrot+=0.4f; //زيادة زاوية الدوران
      glutSwapBuffers();
}
```

بعد التنفيذ نحصل على المكعب يدور و هذه مشهد له:



سنستعمل الإضاءة لإعطاء مزيد من الواقعية للمكعب, سنستعمل فقط إكساء الجدار حتى لأن اللون الأسود لا تظهر عليه الإضاءة.

رأينا في الدرس السابق كيف نضيف مصدر إضاءة و لكن باختصار شديد, سنتعرف هنا إلى مزيد من خصائص الإضاءة.

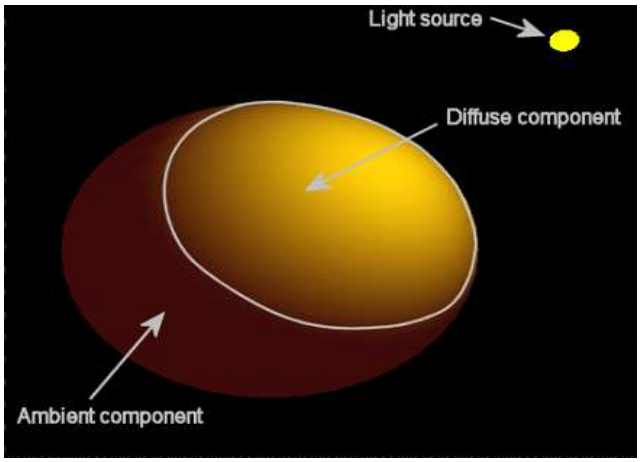
عند تفعيل الإضاءة يعتمد لون الجسم على لونه و أيضا على أربع مكونات للإضاءة :

Diffuse (GL_DIFFISE) : شعاع الضوء يأتي من مصدر الإضاءة و يضرب مباشرة السطح, لذلك هذا اللون يعتمد على موقع الجسم واتجاهه نحو مصدر الإضاءة.

Ambient (GL_AMBIENT) : عندما يصطدم شعاع الضوء بالسطح فإنه ينتشر نحو كل الإتجهات و يبدو الضوء قادم من كل مكان لذلك يتأثر كل المشهد بهذا اللون.

Specular (GL_SPECULAR) : وهو ذلك اللون الذي يكون على المرآة و المعادن, ولذلك يجب أن يستعمل فقط على المواد.

Emission (GL_EMMISION) : بعض الأجسام مثل المصابيح تستطيع إرسال الأضواء, لذلك نستعمل هذا المكون لإنشاء تلك الأجسام.



نعود إلى برنامجنا, وفي كل مرة نحتاج إلى الإضاءة سأشرح المزيد عنها و هذا حتى لا نشعر بالملل في البقاء في موضوع واحد, في البداية نحدد مصدر الإضاءة بثلاث أشياء: الموقع, اللون Diffuse و اللون Ambient , ونحدد كل لون بمصفوفة من أربعة ألوان: أحمر أخضر أزرق و ألفا:

```
//...
unsigned int      texture[2];
float LightAmbient[]= { 1.0f, 1.0f, 1.0f, 1.0f }; //Ambient لون المكون
float LightDiffuse[]= { 1.0f, 1.0f, 1.0f, 1.0f }; //Diffuse لون المكون
float LightPosition[]= { 0.0f, 0.0f, 2.0f, 1.0f }; //موقع الإضاءة
//...
```

نضيف مصدر إضاءة و نحدد خصائصه و نفعله على مستوى الدالة Init():

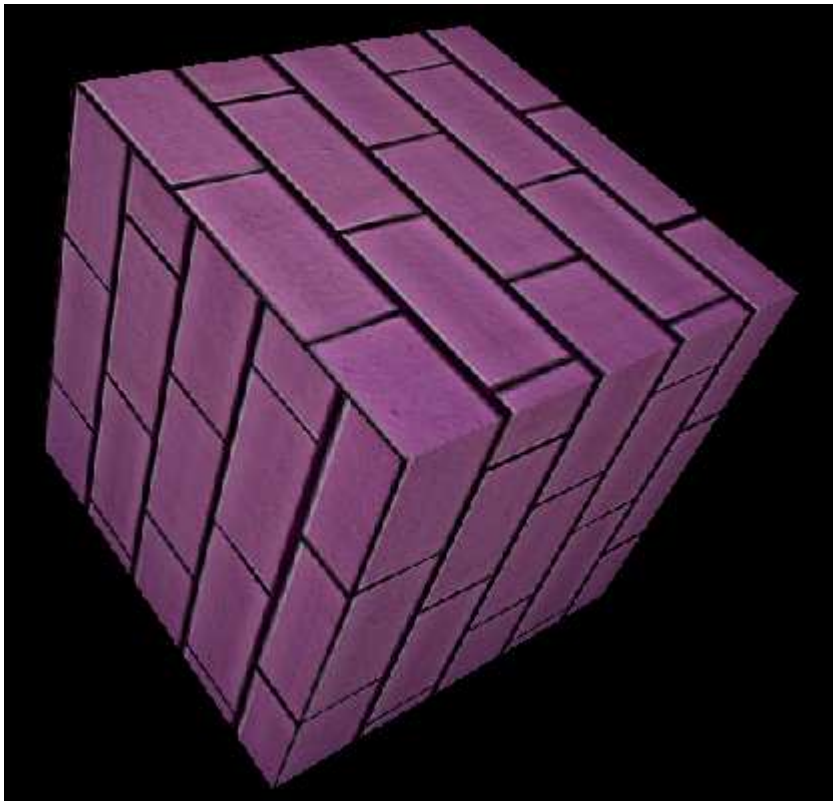
```
//...
bool init(void)
{
      //...
      // نحدد خصائص مصدر الضوء رقم 1 بالمتغيرات التي أنشأناها من قبل والتي تحدد
الموقع و لون مكونين للضوء
      glLightfv(GL_LIGHT1, GL_AMBIENT, LightAmbient);
      glLightfv(GL_LIGHT1, GL_DIFFUSE, LightDiffuse);
      glLightfv(GL_LIGHT1, GL_POSITION,LightPosition);
      glEnable(GL_LIGHTING); //نفعل الإضاءة
      glEnable(GL_LIGHT1); //1 نفعل مصدر الإضاءة رقم
      return TRUE;
}
//...
```

الآن شغل البرنامج و اترك المكعب يدور –طبعا باستعمال إكساء الجدار فقط, أي قم بالغاء السطر رقم27 الذي نختار فيه اكساء النجمة- وسترى أن المكعب تارة نرى عليه تأثير الإضاءة و تارة نراه مظلم, وهذا لأننا لم نحدد إلى أين يتجه كل وجه من أوجه المكعب, نستعمل الدالة glNormal3f(x, y, z) ; والتي نعطيها النقطة التي يتجه إليها السطح, أي سنحدد الوجه الأمامي للسطح, نضيف بعض الأسطر إلى الدالة Display():

```
//...
void Display(void)
{
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glLoadIdentity();
      glTranslatef(0.0f,0.0f,-5.0f);
      glRotatef(xrot,1.0f,0.0f,0.0f);
      glRotatef(yrot,0.0f,1.0f,0.0f);
      glRotatef(zrot,0.0f,0.0f,1.0f);
      glBindTexture(GL_TEXTURE_2D, texture[0]);
      glBegin(GL_QUADS);
            glNormal3f( 0.0f, 0.0f, 1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
            glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
            glNormal3f( 0.0f, 0.0f,-1.0f);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
```

```
            glNormal3f( 0.0f, 1.0f, 0.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
            glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
      glEnd();
      //glBindTexture(GL_TEXTURE_2D, texture[1]); //تعطيل الإكساء الثاني
      glBegin(GL_QUADS);
            glNormal3f( 0.0f,-1.0f, 0.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
            glNormal3f( 1.0f, 0.0f, 0.0f);
            glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
            glNormal3f(-1.0f, 0.0f, 0.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
            glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
            glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
      glEnd();
      xrot+=0.3f;
      yrot+=0.2f;
      zrot+=0.4f;
      glutSwapBuffers();
}
//...
```

لتكون النتيجة أكثر واقعية:



بعد هذه الإضافة السريعة لخصائص الإضاءة سنرى تأثير مزج الألوان على المشهد, إذ ستبدو الأسطح شفافة, لنحاول إذا جعل المكعب شفاف.

عند استعمال المزج فإننا نمزج لون النقطة المرسومة على الشاشة مع لون النقطة الجديدة التي سنرسمها و بذلك يبدو السطح شفافا, قبل ذلك لنلقي نظرة على كيفية دمج الألون.

عندما نريد دمج نقطتين سيكون لدينا أربعة عوامل:

لون المصدر (النقطة التي سنرسمها): (srcR, srcG, srcB, srcA)

عامل المصدر: (sFR, sFG, sFB, sFA)

لون الوجهة(النقطة المرسومة على الشاشة): (destR, destG, destB, destA)

عامل الوجهة: (dFR, dFG, dFB, dFA)

إذا سيكون اللون الناتج عن الدمج كالتالي:

لون المصدر * عامل المصدر + لون الوجهة * عامل الوجهة

وهو مكافئ لـ:

(destA*dFA+ srcR*sFR + destR*dFR, srcG*sFG + destG*dFG, srcB*sFB + destB*dFB, srcA*sFA)

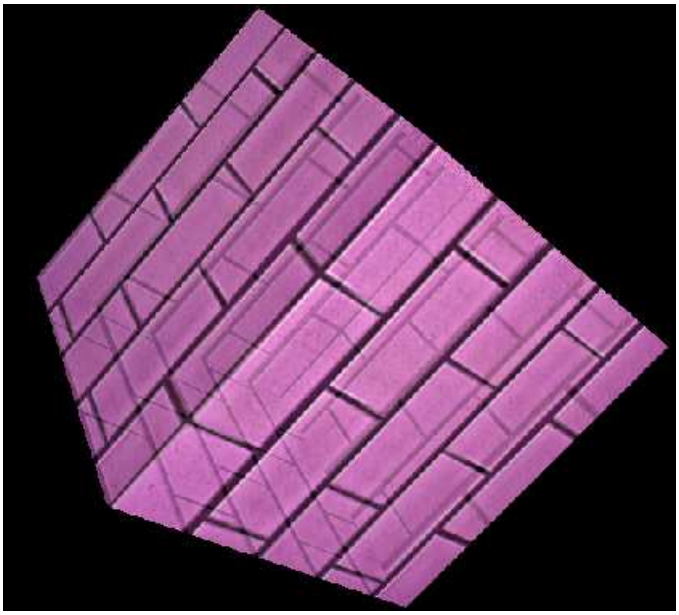
إذا كنت قد لاحظت أننا نعرف لون المصدر و لون الوجهة لكننا لا نعرف عامل المصدر و لا عامل الوجهة؟, هي عبارة عن ثوابت و هي:

GI constants	parameter	value
GL_ZERO	src & dest	(0 ,0 ,0 ,0)
GL_ONE	src & dest	(1 ,1 ,1 ,1)
GL_DST_COLOR	src	(destR, destG, destB, destA)
GL_ONE_MINUS_DST_COLOR	src	(destR, 1-destG, 1-destB, 1-destA-1)
GL_SRC_COLOR	dest	(srcR, srcG, srcB, srcA)
GL_ONE_MINUS_SRC_COLOR	dest	(srcR, 1-srcG, 1-srcB, 1-srcA-1)
GL_SRC_ALPHA	src & dest	(srcA, srcA, srcA, srcA)
GL_ONE_MINUS_SRC_ALPHA	src & dest	(srcA, 1-srcA, 1-srcA, 1-srcA-1)
GL_DST_ALPHA	src & dest	GL_DST_ALPHA src & dest
GL_ONE_MINUS_DST_ALPHA	src & dest	(destA, 1-destA, 1-destA, 1-destA-1)
GL_SRC_ALPHA_SATURATE	src	(... ,min(srcA, 1-destA), idem)

الآن نعود إلى كود الدالة Init() ونفعل دمج الألوان:

```
//...
bool init(void)
{
	//...
	glBlendFunc(GL_SRC_ALPHA,GL_ONE); //تحديد معامل دمج المصدر و الوجهة
	glEnable(GL_BLEND); //تفعيل دمج الألوان
	glDisable(GL_DEPTH_TEST); // يجب إلغاء التأكد من العمق حتى يكون دمج الألوان
صحيحا
	return TRUE;
}
//...
```

نفذ البرنامج و ستحصل على هذه النتيجة:



وكتطبيق أخير لهذا الدرس سنقوم بإنشاء لوحة لونية بتطبيق مبادئ الإكساء و دمج الألوان –لن نحتاج إلى الإضاءة-, إذن, أغلق المشروح الحالي و أنشئ مشروع جديد و سمه أي اسم ثم أضف إليه ملف main.cpp و اكتب فيه الكود الذي ينشئ نافذة فارغة, وبعد ذلك أضف الدوال المسؤولة عن تحميل الإكساء و قم بتفعيل دمج الألوان, باختصار, هذا هو الكود الذي يجب أن يكون في البداية:

```
#include <stdlib.h>
#include <windows.h>
#include <gl\glaux.h>
#include <GL\glut.h>
#include <stdio.h>
#include <math.h>
#include <stdio.h>
#pragma comment(lib, "GLAUX.LIB")

unsigned int      texture[1]; //سنحتاج إلى إكساء واحد

AUX_RGBImageRec *LoadBMP(char *Filename)
{
      FILE *File=NULL;
      if (!Filename)
            return NULL;
      File=fopen(Filename,"r");
      if (File)
      {
            fclose(File);
            return auxDIBImageLoad(Filename);
      }
      return NULL;
}

int LoadGLTextures(char *filename, unsigned int *texture)
{
      int Status=FALSE;
      AUX_RGBImageRec *TextureImage[1];
      memset(TextureImage,0,sizeof(void *)*1);
```

```
	if (TextureImage[0]=LoadBMP(filename))
	{
		Status=TRUE;
		glGenTextures(1, texture);
		glBindTexture(GL_TEXTURE_2D, texture[0]);
		glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
		glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
		glTexImage2D(GL_TEXTURE_2D, 0, 3, TextureImage[0]->sizeX,
TextureImage[0]->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, TextureImage[0]->data);
	}
	if (TextureImage[0])
	{
		if (TextureImage[0]->data)
		{
			free(TextureImage[0]->data);
		}

		free(TextureImage[0]);
	}
	return Status;
}


void Reshape(int w, int h)
{
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(45.0, (float)w/(float)h, 1.0, 100.0);
	gluLookAt(0,0,18,0,0,0,0,1,0);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glBindTexture(GL_TEXTURE_2D, texture[0]);
	glutSwapBuffers();
}

void Key(unsigned char key, int x, int y )
{
	if(key ==  27 )   exit(0);
}


void action(void)
{
	Display();;
}

bool init(void)
{
	if (!LoadGLTextures("star.bmp", &texture[0])) //تحمل صورة النجمة للإكساء
	{
		return FALSE;
	}
	glEnable(GL_TEXTURE_2D);
	glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
```

```
	glBlendFunc(GL_SRC_ALPHA,GL_ONE);
	glEnable(GL_BLEND);
	return TRUE;
}


void main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
	glutInitWindowSize(500, 500);
	glutInitWindowPosition (0,0);
	glutCreateWindow("Colors");
	if(!init())
	{
		MessageBox(NULL, "Initializing Error", "Error", 0);
		exit(1);
	}
	glutDisplayFunc(Display);
	glutReshapeFunc(Reshape);
	glutKeyboardFunc(Key);
	glutIdleFunc(action);
	glutFullScreen();
	glutMainLoop();
}
```

سنستعمل للإكساء صورة النجمة السابقة و باستعمال دمج الألوان سنتمكن من إخفاء المناطق السوداء في الصورة و تلوين المنطقة البيضاء بألوان عشوائية, إذن نحن سنرسم مجموعة من المربعات مكسوة بصورة النجمة فتبدو و كأنها نجوم مضيئة, لكل نجمة موقع على الشاشة (x,y) و زاوية دوران angle حول المحور z و كذلك متغير dist ليحمل المسافة بين موقع النجمة و مركز المعلم.

ننشئ التركيبة star لتحمل معلومات كل نجمة , في بداية الملف:

```
#include <stdlib.h>
#include <windows.h>
#include <gl\glaux.h>
#include <GL\glut.h>
#include <stdio.h>
#include <math.h>
#include <stdio.h>
#pragma comment(lib, "GLAUX.LIB")

typedef struct
{
	int r, g, b; //ثلاث متغيرات تحمل قيم ألوان النجمة
	float dist; //المسافة بين النجمة و المركز
	float angle; //z زاوية دوران النجمة حول المحور
} star;

star circle_stars[40]; //سنعرف مصفوفة من 40 نجمة
//...
```

إذن, أنشأنا 40 نجمة ستدور حول المركز, سنرسم كل نجمتين في موقع واحد لإضفاء مزيد من الإثارة, كيف؟ سنرسم نجمة أولى في الموقع (x,y) و نرسم فوقها في نفس الموقع نجمة أخرى تدور خول نفسها بزاوية يجددها المتغير spin.

على مستوى الدالة Init() نعطي قيم ابتدائية لمواقع و ألوان النجوم:

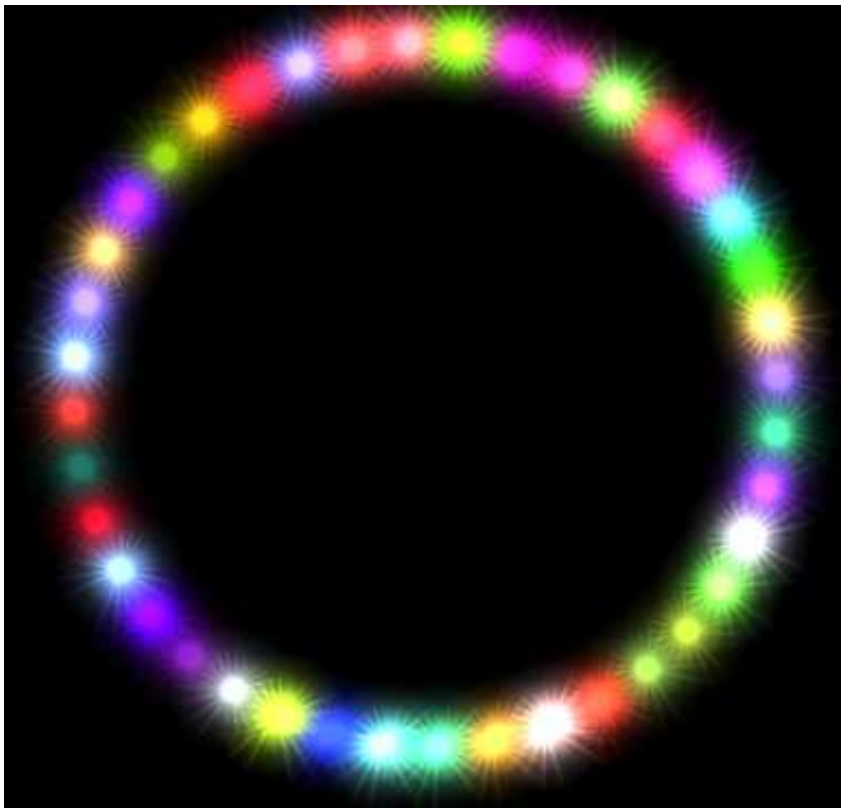
```
//...
float spin; //درجة دوران النجمة حول نفسها
unsigned int      texture[1];
//...
bool init(void)
{
      //...
      int i;
      for (i=0; i<40; i++)
      {
            circle_stars[i].angle=i * 9.0f; // زاوية دوران النجمة حول المحور زد, كل نجمة تميل عن أختها بـ 9 درجات, أي أن 9*40 تساوي 360 درجة, النجوم موزعة على طول الدائرة
            circle_stars[i].dist=5.0f; //النجوم تبعد عن المركز بـ 5وحدات
            circle_stars[i].r=rand()%256; //لون عشوائي
            circle_stars[i].g=rand()%256; //لون عشوائي
            circle_stars[i].b=rand()%256; //لون عشوائي
      }
      return TRUE;
}
//...
```

نتجه إلى الدالة Display ونكتب الكود الذي يرسم و النجوم:

```
//...
void Display(void)
{
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glBindTexture(GL_TEXTURE_2D, texture[0]);
      int i;
      for (i=0; i<40; i++)
      {
            glLoadIdentity(); //إعادة الإعدادات كما كانت
            glRotatef(circle_stars[i].angle,0.0f,0.0f,1.0f); // ندور النجمة حول المركز حسب الزاوة المعطاة
            glTranslatef(circle_stars[i].dist,0.0f,0.0f); // نبعد النجمة عن المركز حسب المسافة الموجودة في التركيبة الخاصة بكل نجمة
            glColor4ub(circle_stars[(40-i)-1].r, // نلون الإكساء عشوائيا حسب قيم الألوان الخاصة بكل نجمة
                  circle_stars[(40-i)-1].g,
                  circle_stars[(40-i)-1].b,255);
            glBegin(GL_QUADS); //نرسم مربع و نكسوه بإكساء النجمة
                  glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f,-1.0f, 0.0f);
                  glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f,-1.0f, 0.0f);
                  glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f, 1.0f, 0.0f);
                  glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 0.0f);
            glEnd();
            glRotatef(spin,0.0f,0.0f,1.0f); // الآن ندير المعلم لنرسم فوق النجمة التي رسمناها نجمة أخرى وهي النجمة التي تقابلها في مصفوفة النجوم وبالطبع ستكون بلون مختلف و باستعمل دمج الألوان ستظهر النجمتان كأنهما نجمة واحدة مشعة
            glColor4ub(circle_stars[(40-i)-1].r, // نلون النجمة الجديدة بلون النجمة المقابلة للنجمة الحالية
                  circle_stars[(40-i)-1].g,
                  circle_stars[(40-i)-1].b,255);
```

```
            glBegin(GL_QUADS); //نرسم النجمة الثانية
                  glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f,-1.0f, 0.0f);
                  glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f,-1.0f, 0.0f);
                  glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f, 1.0f, 0.0f);
                  glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 0.0f);
            glEnd();
            spin+=0.01f; // نضيف 0.01 للدرجة التي ندور بها النجمة خول نفسها قبل أن نرسمها
            circle_stars[i].angle -= 1.0f; // ننقص الزاوية التي تميل بها النجمة عن محور السينات بمقدار 1 و بهذا تظهر انجوم وكأنها تدور في إتجاه عقارب الساعة
      }
      glutSwapBuffers();
}
//...
```

يجب أن يعطيك تنفيذ البرنامج هذه النتيجة:



سنضيف مزيدا من الإثارة للمشهد و هذا بإضافة 60 نجمة أخرى تدور عكس إتجاه دوران النجوم الحالية كما أن مقدار بعدها عن المركز ينقص باستمرار فتظهر و كأنها تتجه إلى المركز وعند وصولها إلى المركز نعيد قيمة بعدها عن المركز كما كانت, إذن نضيف مصفوفة أخرى من النجوم و نعطيها قيم إبتدائية لألوانها و بعدها و زاوية ميلان عن محور السينات بقيمة 0 درجة, كل هذا على مستوى الدالة Init():

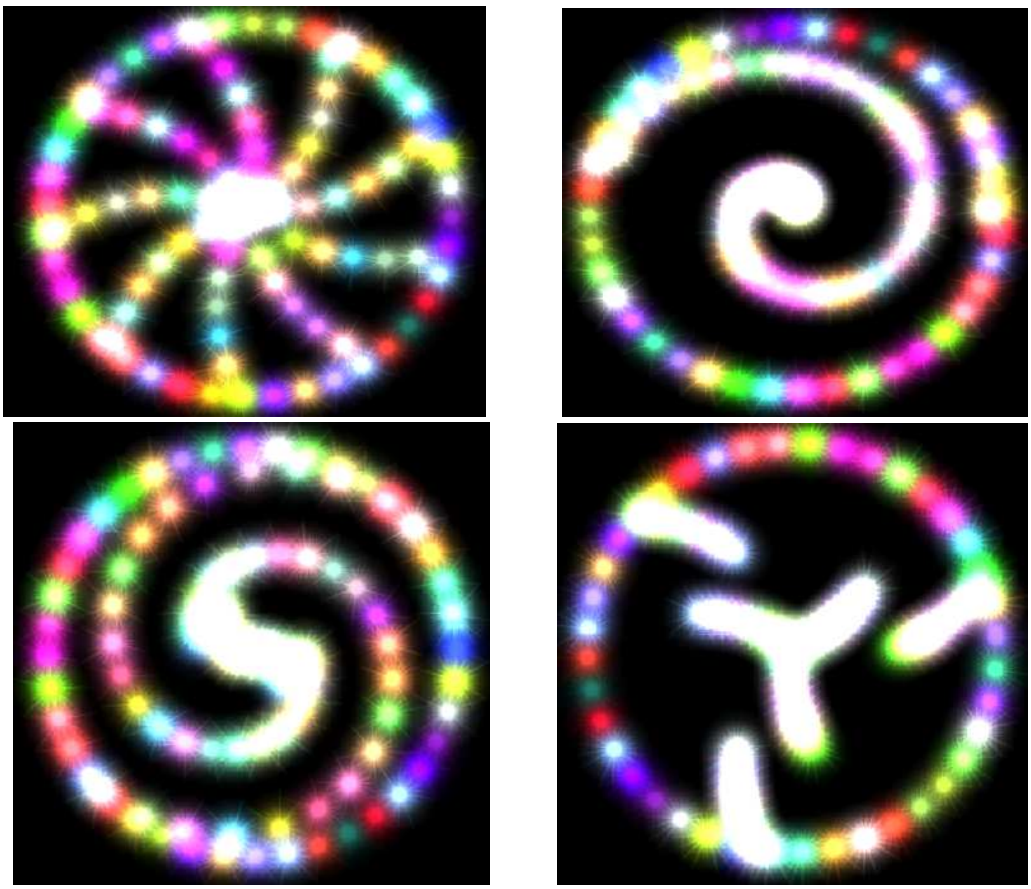
```
//...
star circle_stars[40];
star stars[60]; //مصفوفة الستون نجمة الجدد
//...
bool init(void)
{
	//...
	for (i=0; i<60; i++)
	{
		stars[i].angle=0.0f; // 0 زاوية دورانها الإبتدائية تساوي
		stars[i].dist=(float(i)/60)*5.0f; // النجمة الأولى في المركز, النجمة
الثانية تبعد بـ 5 وحدات عن المركز, باقي انجوم تتغير المسافة بينها و بين المركز حسب
موقعها في المصفوفة
		stars[i].r=rand()%256; //لون عشوائي
		stars[i].g=rand()%256; //لون عشوائي
		stars[i].b=rand()%256; //لون عشوائي
	}
	return TRUE;
}
//...
```

ثم نرسم النجوم الجديدة بإضافة كود إلى الدالة Display():

```
//...
void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glBindTexture(GL_TEXTURE_2D, texture[0]);
	int i;
	for (i=0; i<60; i++) //بنفس الطريقة نرسم الستون نجمة الجدد
	{
		glLoadIdentity();
		glRotatef(stars[i].angle,0.0f,0.0f,1.0f);
		glTranslatef(stars[i].dist,0.0f,0.0f);
		glColor4ub(stars[(60-i)-1].r,stars[(60-i)-1].g,stars[(60-i)-
1].b,255);
		glBegin(GL_QUADS);
			glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f,-1.0f, 0.0f);
			glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f,-1.0f, 0.0f);
			glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f, 1.0f, 0.0f);
			glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 0.0f);
		glEnd();
		glRotatef(spin,0.0f,0.0f,1.0f);
		glColor4ub(stars[i].r,stars[i].g,stars[i].b,255);
		glBegin(GL_QUADS);
			glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f,-1.0f, 0.0f);
			glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f,-1.0f, 0.0f);
			glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f, 1.0f, 0.0f);
			glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 0.0f);
		glEnd();
```

```
            stars[i].angle+=float(i)/60; // نزيد من قيمة دوران النجوم حول المركز وهنا نحدد سرعة الدوران
            stars[i].dist-=0.01f; // ننقص في كل مرة قيمة البعد عن المركز حتى تبدو النجمة تتجه نحو المركز
            if (stars[i].dist<0.0f) //هل وصلت النجمة إلى المركز
            {
                  stars[i].dist = 5.0f; //نعيد النجمة إلى النقطة 5 من المركز
                  stars[i].r=rand()%256; //و نعطيها لون آخر عشوائيا
                  stars[i].g=rand()%256;
                  stars[i].b=rand()%256;
            }
      }
      //...
      glutSwapBuffers();
}
//...
```

نفذ الآن و ستحصل على نتيجة رائعة, وهذه بعض الصور:



وصلنا الآن إلى نهاية هذا الدرس, وهذا رابط تحميل برنامج المكعب و النجوم:

http://www.mediafire.com/?imz5yrnq2id

وإلى اللقاء في درس آخر.