www.startimes2.com

دروس في الـ OpenGL

الدرس 4: التجول داخل عالم ثلاثي الأبعاد

khatibe_30@hotmail.fr

2010

السلام عليكم ورحمة الله وبركاته, في هذا الدرس سننشئ عالم ثلاثي الأبعاد بسيط و نوضح كيف تتم عملية حساب الإحداثيات أثناء الحركة, أول خطوة هي إنشاء مشروع Console جديد, أضف إليه ملف جديد باسم main.cpp, كنا قد تطرقنا في الدرس السابق إلى كيفية إكساء الأسطح و لذلك أضف إلى المشروع ملف جديد باسم textures.h وسنكتب فيه الدوال المسؤولة عن الإكساء حتى لا تختلط الأكواد, لذلك سنكتب فيه:

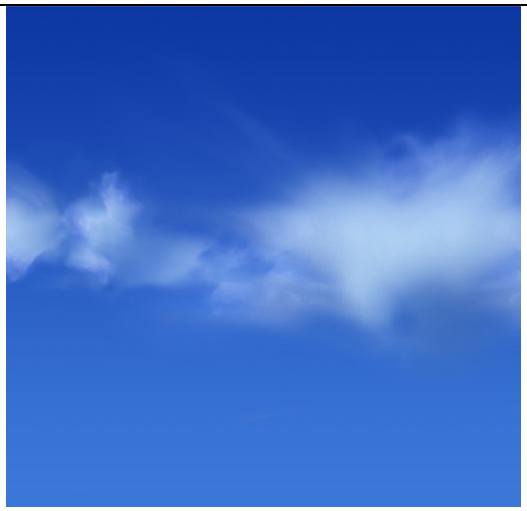
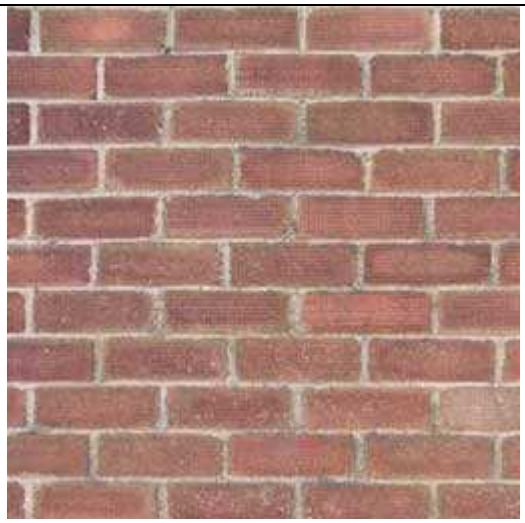
```
#pragma once
#include <stdlib.h>
#include <windows.h>
#include <gl\glaux.h>
#include <gl\glu.h>
#include <GL\glut.h>
#include <stdio.h>
#include <math.h>
#include <stdio.h>
#pragma comment(lib, "GLAUX.LIB")


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 LoadTextures(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;
}
```

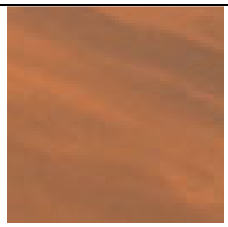
الكود السابق شرحناه في الدرس السابق, الآن نفتح الملف main.cpp ونكتب الكود المعهود الذي ينشئ نافذة فارغة, وهذا هو الكود:

```
#include <stdlib.h>
#include <windows.h>
#include <gl\glaux.h>
#include <gl\glu.h>      //سنستخدم بعض الدوال من هنا
#include <GL\glut.h>
#include <stdio.h>
#include <math.h>
#include <stdio.h>
#include "textures.h"  //الملف الذي يحتوي على دوال الاكسا الذي أنشأناه للتو
#pragma comment(lib, "GLAUX.LIB")

void Reshape(int w, int h)
{
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(45.0, (float)w/(float)h, 1.0, 400.0);
	gluLookAt(0,3,0.1,0,3,0,0,1,0);
	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);
}

bool init(void)
{
	glEnable(GL_TEXTURE_2D);
	glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
	glClearDepth(1.0f);
	glEnable(GL_DEPTH_TEST);
	glShadeModel(GL_SMOOTH);
	glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
	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("Lesson4");
```

```
	if(!init())
	{
		MessageBox(NULL, "Initializing Error", "Error", 0);
		exit(1);
	}
	glutDisplayFunc(Display);
	glutReshapeFunc(Reshape);
	glutKeyboardFunc(Key);
	ShowCursor(0);
	glutFullScreen();
	glutMainLoop();
}
```

سنستعمل لإنشاء عالمنا ثلاثي الأبعاد هذه الإكساءات:

 sky.bmp	 wall.bmp
 montada.bmp	 grass.bmp

 cylinder.bmp	

طبعا يمكنك اختيار إكساءات أخرى,قم بنسخ هذه الإكساءات و احفظها في مجلد جديد باسم data وضعه في نفس مجلد المشروع, أول ما نقوم به بعد ذلك هو كتابة الكود الذي يحمل الإكساءات :

```
#include <stdlib.h>
#include <windows.h>
#include <gl\glaux.h>
#include <gl\glu.h>
#include <GL\glut.h>
#include <stdio.h>
#include <math.h>
#include <stdio.h>
#include "textures.h"
unsigned int      texture[5]; //لدينا 5 اكساءات
//...
bool init(void)
{
      if (!LoadTextures(".\\data\\wall.bmp", &texture[0]))
            return FALSE;
      if (!LoadTextures(".\\data\\grass.bmp", &texture[1]))
            return FALSE;
      if (!LoadTextures(".\\data\\sky.bmp", &texture[2]))
            return FALSE;
      if (!LoadTextures(".\\data\\cylinder.bmp", &texture[3]))
            return FALSE;
      if (!LoadTextures(".\\data\\montada.bmp", &texture[4]))
            return FALSE;
      //...
      return TRUE;
}
```

سنستخدم الـ DisplayList لرسم الأشياء الساكنة في المشهد حتى نسرع عملية سم كل مشهد, ولكن ما هي؟

هي عبارة عن مجموعة من أوامر الرسم المعالجة و المخزنة في الذاكرة حيث أن استدعاءها يعطي نتيجة أسرع, هذا هو المفهوم المبدئي الذي سنتعامل معه الآن, مع أن لها حدود و نقائص إلا أننا سنتطرق لكل شيء في وقته.

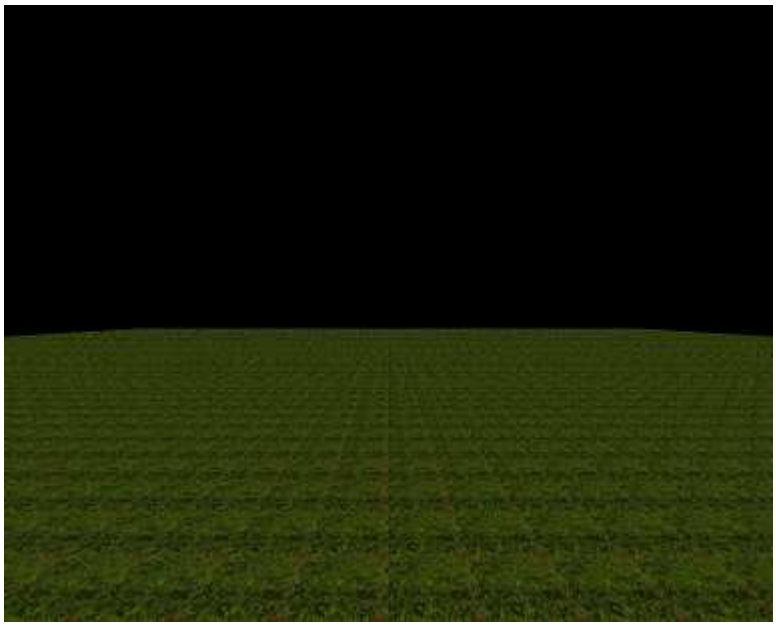
لذلك, نضيف دالة جديدة تقوم بإنشاء DisplayList ترسم لنا المشاهد الثابتة و نسمي هذه الدالة SetupWorldList:

```
//...
void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	glCallList(1);  //DisplayList إستدعاء الـ
	glutSwapBuffers();
}
//...
void SetupWorldList() //DisplayList الدالة التي تنشئ الـ
{
 glNewList(1, GL_COMPILE); //1 وسيكون إسمها DisplayList إنشاء الـ
	//هنا نكتب الأوامر التي ترسم بعض الأشياء عند إستدعاء الليست
 glEndList(); //النهاية
}
//...
bool init(void)
{
	//...
	SetupWorldList(); //DisplayList إنشاء الـ
	return TRUE;
}
//...
```

أول ما ننشئه هو الأرضية, عبارة عن مربع مكسو باكساء الحشيش و نرسمه داخل الـDisplayList:

```
//...
void SetupWorldList()
{
 glNewList(1, GL_COMPILE);
 glBindTexture(GL_TEXTURE_2D, texture[1]); //نستخدم إكساء الحشيش
	glBegin(GL_QUADS);
		glNormal3f( 0.0f, 1.0f, 0.0f);
		glTexCoord2f(0.0f, 0.0f); glVertex3f(-50.0f, 0.0f,  -100.0f);
		glTexCoord2f(100.0f, 0.0f); glVertex3f( 50.0f, 0.0f,  -100.0f);
		glTexCoord2f(100.0f, 100.0f); glVertex3f( 50.0f,  0.0f,  100.0f);
		glTexCoord2f(0.0f, 100.0f); glVertex3f(-50.0f,  0.0f,  100.0f);
	glEnd();
 glEndList();
}
//...
```

النتيجة يجب أن تكون كالآتي:



قبل رسم أي شيء آخر يجب علينا أن نكتب الكود الذي يسمح لنا بالحركة بحرية, لذلك نضيف بضعة متغيرات و هي:

```
//...
#define PI_OVER_180  0.0174532925f // هذه القيمة هي بي(3.14) مقسومة على 180
وسنستخدمها للتحويل من الدرجة إلى الراديان
float Xrotate=0, // متغير يحمل زاوية الدوران إذا ضغطنا على المفتاح الأيمن أو الأيسر في
لوحة المفاتيح
      Zmove=-50, //مقدار إزاحة المعلم على المحور زد
      Xmove=0, //مقدار إزاحة المعلم على المحور اكس
      Ymove=0, //مقدار إزاحة المعلم على المحور واي
      YAngel=0; //نشرح دوره لاحقا
//...
```

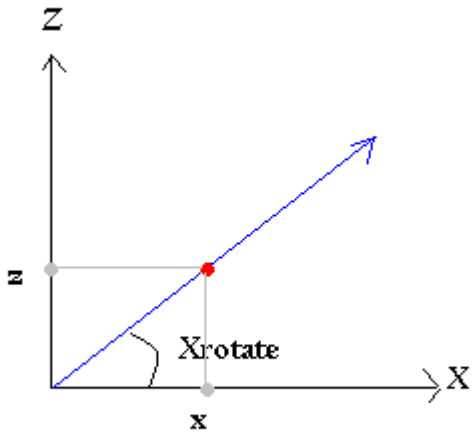
كيف تتم الحركة, بالنسبة للدوران فهذا سهل, إذا ضغطنا على الإتجاه الأيمن من لوحة المفاتيح فإننا نزيد من قيمة المتغير Xrotate الذي يحمل قيمة زاوية الدوران على المحور Y, وإذا ضغطنا على على الإتجاه الأيسر من لوحة المفاتيح يحدث العكس, لنكتب هذا الكود أولا, طبعا يجب إضافة دالة تعالج حدث الضغط على المفاتيح و دالة أخرى تعالج حدث رفع الضغط عن المفاتيح و قد رأينا هذا في الدروس السابقة, وأيضا نضيف الدالة التي تقوم بتحديث المشهد بشكل متواصل, وهذا الكود هو ما يجب إضافته:

```
//...
bool    skey[255]; //مصفوفة تمثل كل المفاتيح الـ255
//...
void SpecialKeys(int key, int x, int y)
{
  skey[key] = true; //نعطي القيمة صحيح للمفتاح المضغوط
}
```

```
void action(void)
{
 if(skey[GLUT_KEY_LEFT]) //إذا كان مفتاح الإتجاه الأيسر مضغوط
       Xrotate--; //ندور عكس إتجاه عقارب الساعة
 if(skey[GLUT_KEY_RIGHT]) //إذا كان مفتاح الإتجاه الأيمن مضغوط
       Xrotate++; //ندور في إتجاه عقارب الساعة
 Display(); //نرسم المشهد في كل مرة
}

void SpecialKeysUP(int key, int x, int y)
{
      skey[key] = false; //نعطي القيمة خطأ للمفتاح الذي رفع الضغط عنه
}
//...
void main(int argc, char **argv)
{
      //...
      glutSpecialFunc(SpecialKeys);
      glutSpecialUpFunc(SpecialKeysUP);
      glutIdleFunc(action);
      //...
}
```

شغل البرنامج و ستجد أننا نستطيع الدوران حول المحور Y, ماذا عن الحركة إلى الأمام و إلى الوراء؟ ...

عندما نتقدم إلى الأمام فإن قيمة كل من Xmove و Zmove ستتغير و هذا حسب الزاوية التي نميل بها حول المحور Y, كيف هذا؟ لنرى هذا الشكل:



كما نرى فإن موقعنا الحالي هو تلك النقطة الحمراء لذلك إذا ضغطنا على الإتجاه الأعلى أو الأسفل سنسير في اتجاه الشعاع الأزرق أو عكس اتجاهه, لذلك في كل مرة نضغط على أحد المفتاحين نزيد إلى قيمة Xmove القيمة x على الصورة, و نزيد إلى قيمة Zmove القيمة z, حيث أن :

$$x = \sin(Xrotate) \text{ و } z = \cos(Xrotate)$$

و بما أن Xrotate وحدتها الدرجة يجب تحويلها إلى الراديان و بذلك تكون قيم Xmove و Zmove كالتالي:

Xmove = Xmove + sin (-Xrotate * PI / 180);

Zmove = Zmove + cos (-Xrotate * PI / 180);

ليكون الكود النهائي كالتالي:

```
//...
void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	glRotatef(Xrotate,0,1,0); //ندور أولا
	glTranslatef(Xmove, Ymove, Zmove); //ثم نتحرك
	glCallList(1);
	glutSwapBuffers();
}
//...
void action(void)
{
 if(skey[GLUT_KEY_UP])
 {
	Zmove+=(float)cos(-Xrotate*PI_OVER_180)*0.6; // نضرب في 0.6 لنتحكم في سرعة
الحركة
	Xmove+=(float)sin(-Xrotate*PI_OVER_180)*0.6;
 }
 if(skey[GLUT_KEY_DOWN])
 {
	Zmove-=(float)cos(-Xrotate*PI_OVER_180)*0.6;
	Xmove-=(float)sin(-Xrotate*PI_OVER_180)*0.6;
 }
 if(skey[GLUT_KEY_LEFT])
	Xrotate--;
 if(skey[GLUT_KEY_RIGHT])
	Xrotate++;
 Display();;
}
//...
```

شغل البرنامج و اشعر بحرية الحركة إلى أي مكان تريد, ولكن إذا لم تلاحظ فإننا لم نستعمل ولم نغير من قيمة المتغير Ymoveو الذي يمثل قيمة التغير على المحور Y أي إلى الأعلى أو الأسفل, و بما أن قيمته 0 دائما فالحركة تكون على مستوى ثابت من ناحية الارتفاع, ماذا إذا غيرنا من تلك القيمة بشكل متواصل حتى نتحرك بشكل واقعي أكثر, أي أننا أثناء الحركة مرة نرتفع و مرة ننخفض و هذا باستعمال متغير و معادلة جيبية بسيطة:

```
//...
void action(void)
{
 if(skey[GLUT_KEY_UP])
 {
	Zmove+=(float)cos(-Xrotate*PI_OVER_180)*0.6;
	Xmove+=(float)sin(-Xrotate*PI_OVER_180)*0.6;
	YAngel+=10;
	Ymove=(float)sin(YAngel * PI_OVER_180)/5;
 }
```

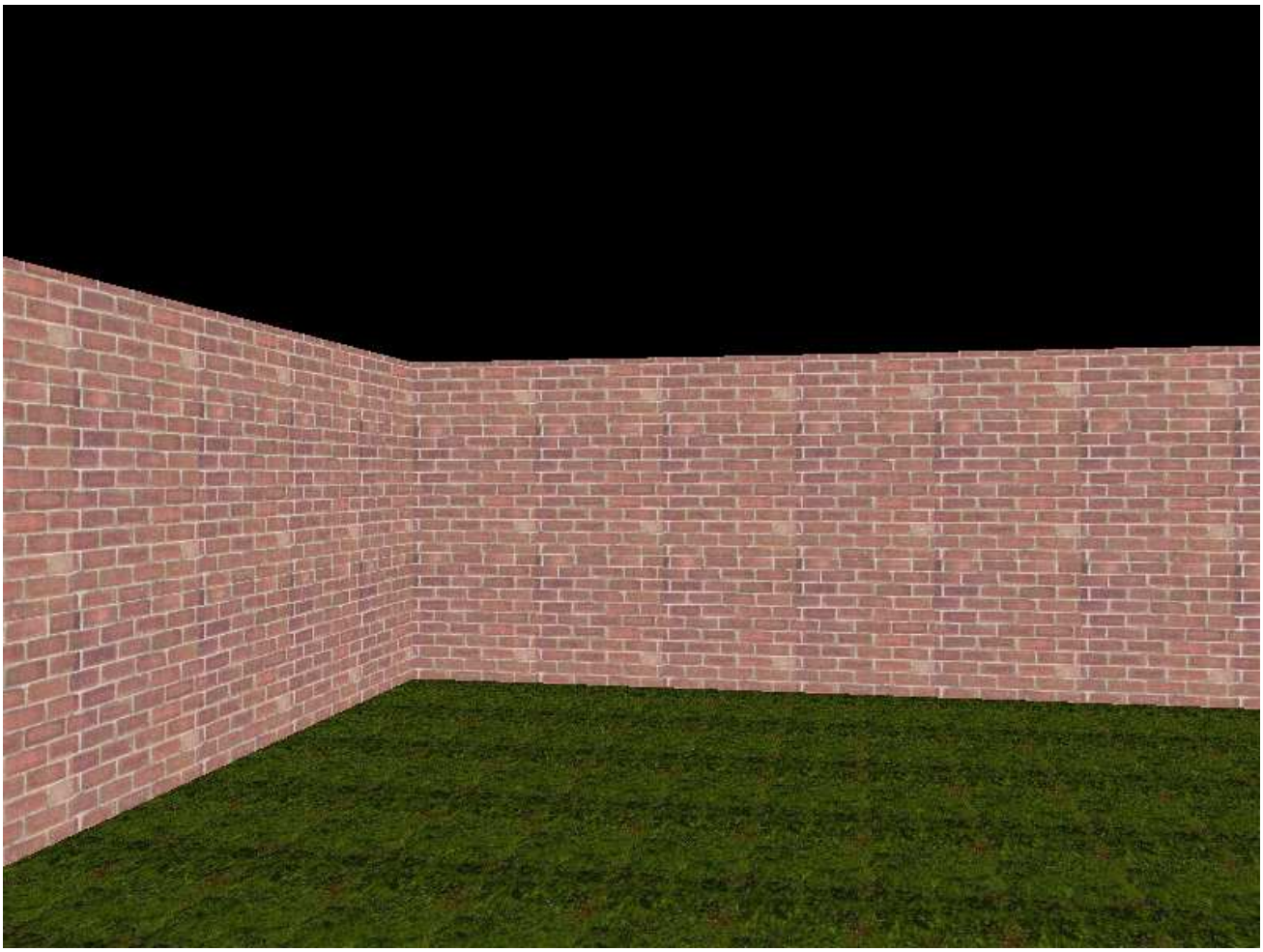
```
if(skey[GLUT_KEY_DOWN])
 {
       Zmove-=(float)cos(-Xrotate*PI_OVER_180)*0.6;
       Xmove-=(float)sin(-Xrotate*PI_OVER_180)*0.6;
       YAngel+=10;
       Ymove=(float)sin(YAngel * PI_OVER_180)/5;
 }
 //...
}
//...
```

الآن أصبحت الحركة تهتز و أكثر واقعية و كأنها حركة شخص يركض, بعد هذا نكمل تصميم عالمنا ثلاثي الأبعد, بعد أن وضعنا أرضية سنضع جدران تمتد من النقطة(x, z) = (-50, -100) إلى النقطة (x, z) = (50, 100) , طبعا سنشكل مربعا أقصى حدوده النقطتين السابقتين و نكسوه بإكساء الجدار, لا نحتاج إلى شرح الكود فكل ذلك مشروح في الدرس السابق لذلك سأكتب الكود فقط مع بعض التعليقات:

```
//...
void SetupWorldList()
{
 glNewList(1, GL_COMPILE);
 glBindTexture(GL_TEXTURE_2D, texture[0]); //نختار إكساء الجدار
      //الجدار الخلفي
      glBegin(GL_QUADS);
            glNormal3f( 0.0f, 0.0f, -1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-50.0f, 0.0f, 100.0f);
            glTexCoord2f(50.0f, 0.0f); glVertex3f( 50.0f, 0.0f, 100.0f);
            glTexCoord2f(50.0f, 2.5f); glVertex3f( 50.0f,  5.0f, 100.0f);
            glTexCoord2f(0.0f, 2.5f); glVertex3f(-50.0f,  5.0f, 100.0f);
      glEnd();
      //الجدار الأمامي
      glBegin(GL_QUADS);
            glNormal3f( 0.0f, 0.0f, 1.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-50.0f, 0.0f, -100.0f);
            glTexCoord2f(50.0f, 0.0f); glVertex3f( 50.0f, 0.0f, -100.0f);
            glTexCoord2f(50.0f, 2.5f); glVertex3f( 50.0f,  5.0f, -100.0f);
            glTexCoord2f(0.0f, 2.5f); glVertex3f(-50.0f,  5.0f, -100.0f);
      glEnd();
      //الجدار الأيسر
      glBegin(GL_QUADS);
            glNormal3f( -1.0f, 0.0f, 0.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-50.0f, 0.0f, 100.0f);
            glTexCoord2f(100.0f, 0.0f); glVertex3f(-50.0f, 0.0f, -100.0f);
            glTexCoord2f(100.0f, 2.5f); glVertex3f(-50.0f,  5.0f, -100.0f);
            glTexCoord2f(0.0f, 2.5f); glVertex3f(-50.0f,  5.0f, 100.0f);
      glEnd();
      //الجدار الأيمن
      glBegin(GL_QUADS);
            glNormal3f( 1.0f, 0.0f, 0.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(50.0f, 0.0f, 100.0f);
            glTexCoord2f(100.0f, 0.0f); glVertex3f(50.0f, 0.0f, -100.0f);
            glTexCoord2f(100.0f, 2.5f); glVertex3f(50.0f,  5.0f, -100.0f);
            glTexCoord2f(0.0f, 2.5f); glVertex3f(50.0f,  5.0f, 100.0f);
      glEnd();
      //الأرضية
      glBindTexture(GL_TEXTURE_2D, texture[1]);
      glBegin(GL_QUADS);
            glNormal3f( 0.0f, 1.0f, 0.0f);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-50.0f, 0.0f,  -100.0f);
```

```
            glTexCoord2f(100.0f, 0.0f); glVertex3f( 50.0f, 0.0f,  -100.0f);
            glTexCoord2f(100.0f, 100.0f); glVertex3f( 50.0f,  0.0f,  100.0f);
            glTexCoord2f(0.0f, 100.0f); glVertex3f(-50.0f,  0.0f,  100.0f);
      glEnd();
glEndList();
}
//...
```

طبعا رسمنا الجدار باستعمال الـ DisplayList وهذا لأنه ثابت و لاتوجد به إحداثيات متغيرة, و النتيجة:



والآن نضيف شروط بسيطة كي لا نتجاوز حدود الجدار أثناء الحركة ز هذا في الدالة Display:

```
//...
void Display(void)
{
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glLoadIdentity();
      glRotatef(Xrotate,0,1,0);
      if(Xmove < -48)   Xmove = -48;
      if(Xmove > 48)    Xmove = 48;
      if(Zmove < -98)   Zmove = -98;
      if(Zmove > 98)    Zmove = 98;
      glTranslatef(Xmove, Ymove, Zmove);
      glCallList(1);
//...
```

الآن سنضيف سماء للمشهد, ستكون في البداية عبارة عن كرة كبيرة نصف قطرها 200 مكسوة بإكساء السماء,وسنستعمل كان من المكتبة glu.h لإنشاء الكرة بدل أن نحسب إحداثياتها بأنفسنا, ولكن إكساء الكرة ليس كإكساء مربع, إذ أنه يحتاج إلى معادلات رياضية للحصول على إحداثيات إكساء دقيقة و لكننا في البداية سنستعمل خاصية توليد الإحداثيات أوتوماتكيا بإضافة هذا الكود إلى الدالة Init():

```
//...
GLUquadricObj *quadratic; //الكائن الذي سنرسم به الكرة
//...
bool init(void)
{
      //...
      glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP); // نولد إحداسيات
إكساءخاصة بالكرةعلى المحور اس و هو المحور العمودي بطريقة آلية
      glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP); // نولد إحداسيات
إكساءخاصة بالكرةعلى المحور تي و هو المحور الأفقي بطريقة آلية
      quadratic = gluNewQuadric(); // نهيئ المتغير الذي سنرسم به كرة
      gluQuadricTexture(quadratic, GL_TRUE); //تفعيل الإكساء بالنسبة للكرة
      SetupWorldList();
      return TRUE;
}
//...
```

جيد, الآن نرسم السماء على مستوى الدالة SetupWorldList:

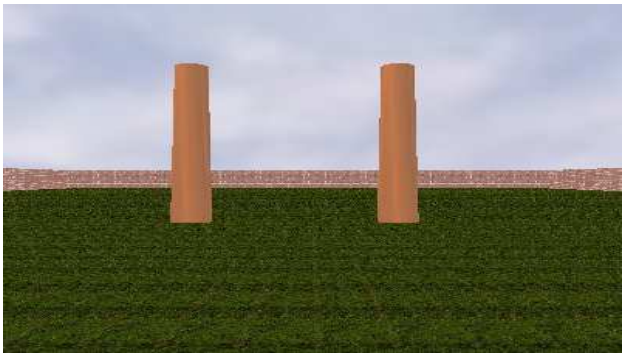
```
//...
void SetupWorldList()
{
 glNewList(1, GL_COMPILE);
      //...
      //نرسم السماء
      glBindTexture(GL_TEXTURE_2D, texture[2]); //نختار إكساء السماء
      glEnable(GL_TEXTURE_GEN_S); // تفعيل التوليد الآلي لإحداثيات الإكساء على المحور
الأفقي
      glEnable(GL_TEXTURE_GEN_T); // تفعيل التوليد الآلي لإحداثيات الإكساء على المحور
العمودي
      gluSphere(quadratic,200.0f,32,32); //نرسم الكرة بنصف قطر 200
      glDisable(GL_TEXTURE_GEN_S); //تعطيل التوليد الآلي لإحداثيات الإكساء على المحور
الأفقي

      glDisable(GL_TEXTURE_GEN_T); //تعطيل التوليد الآلي لإحداثيات الإكساء على المحور
العمودي

 glEndList();
}
//...
```

بذلك نحصل على هذه النتيجة:

الآن نضيف عمودين إلى المشهد بنفس طريقة إضافة كرة السماء:

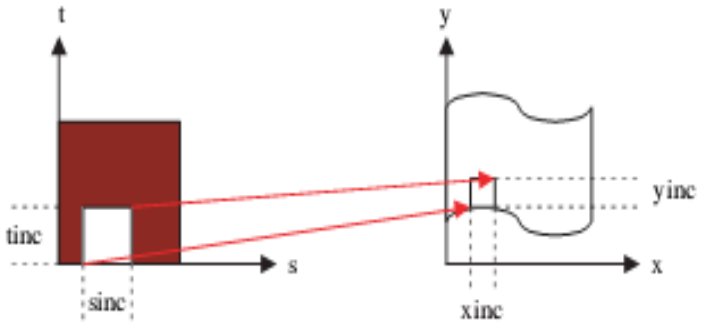
```
//...
void SetupWorldList()
{
 glNewList(1, GL_COMPILE);
      //...
      gluSphere(quadratic,200.0f,32,32);
      glPushMatrix(); //تخزين المصفوفة
      glTranslatef(5, 10, -10); //نغير موقع الاسطوانة
      glRotatef(90,1,0,0); //ندور الأسطوانة لأنها ترصم أفقيا
      gluCylinder(quadratic,0.8f,1.0f,10.0f,32,32);
      glPopMatrix(); //إستعادة المصفوفة
      glPushMatrix();//تخزين المصفوفة
      glTranslatef(-5, 10, -10);
      glRotatef(90,1,0,0);
      gluCylinder(quadratic,0.8f,1.0f,10.0f,32,32);
      glPopMatrix();//إستعادة المصفوفة
      glDisable(GL_TEXTURE_GEN_S);
      glDisable(GL_TEXTURE_GEN_T);
 glEndList();
}//...
```

والنتيجة:



آخر جسم سنرسمه في هذا الدرس هو علم ستارتايمز يتموج, كيف ذلك ؟

طبعا لنتمكن من تحريك العلم في كل الأحوال سنحتاج إلى تقسيمه إلى قطع و نحرك إحداثيات تلك القطع, و طبعا نحسب إحداثيات الإكساء الموافقة لكل قطعة هكذا :



أول ما سنقوم به هو إنشاء مصفوفة تحمل إحداثيات نقط العلم :

```
//...
float points[ 45 ][ 45 ][3];  // المصفوفة التي تحمل إحداثيات النقاط التي تحدد مربعات العلم, و قد قمنا بتقسيم العلم إلى 44 قطعة على الطول و 44 قطعة على العرض أي بمجموع 44 * 44 قطعة
int wiggle = 0;     // متغير نحدد بواسطته سرعة تموج العلم
//...
```

كيف يتموج العلم ؟ طبعا بتغيير إحداثياته في كل مرة, بالنسبة للإحداثيات على المحور X و Y فهي لا تتغير, أما إحداثيات العمق والتي هي على المحور Z فهي تتغير حسب معادلة جيبية, وبما أن المعادلة الجيبية مسارها على شكل أمواج فستعطيننا حركة متموجة للعلم, و لكن أولا علينا تهيئة إحداثيات العلم بقيم ابتدائية على مستوى الدالة Init():

```
//...
bool init(void)
{
	//...
	SetupWorldList();
	for (int x = 0; x < 45; x++) {
		for(int y = 0; y < 45; y++) {
			points[x][y][0]=float((x/5.0f)-4.5f);
			points[x][y][1]=float((y/5.0f)-4.5f);
			points[x][y][2]=float(sin(((x * 8.0f)/360.0f)*6.0f));
		}
	}
	return TRUE;
}
//...
```

الآن لنرسم العلم و نغير من إحداثيات العمق في كل مرة, و بما أن إحداثيات العلم ستتغير في كل مرة لن نرسمه باستخدام الـ DisplayListو انما مباشرة في الدالة Display():

```
//...
void Display(void)
{
	//...
	glBindTexture(GL_TEXTURE_2D, texture[4]);
	glTranslatef(0, 5, -10);
	glBegin(GL_QUADS);
	int x, y;
	float float_x, float_y, float_xb, float_yb, tmp;
	for (x = 0; x < 44; x++) {
		for (y = 0; y < 44; y++) {
			float_x = float(x)/44.0f;
			float_y = float(y)/44.0f;
			float_xb = float(x+1)/44.0f;
			float_yb = float(y+1)/44.0f;

			glTexCoord2f(float_x, float_y);
			glVertex3f(points[x][y][0], points[x][y][1],
points[x][y][2]);

			glTexCoord2f(float_x, float_yb);
			glVertex3f(points[x][y+1][0], points[x][y+1][1],
points[x][y+1][2]);

			glTexCoord2f(float_xb, float_yb);
			glVertex3f(points[x+1][y+1][0], points[x+1][y+1][1],
points[x+1][y+1][2]);

			glTexCoord2f(float_xb, float_y);
			glVertex3f(points[x+1][y][0], points[x+1][y][1],
points[x+1][y][2]);
		}
	}
	glEnd();
	if (wiggle == 2) {
		for (y = 0; y < 45; y++) {
			tmp = points[0][y][2];
			for (x = 0; x < 44; x++) points[x][y][2] = points[x+1][y][2];
			points[44][y][2] = tmp;
		}
		wiggle = 0;
	}
	wiggle++;
	glutSwapBuffers();
}
//...
```

لن أعلق على الكود السابق لسهولته و لأني تعبت من الكتابة, سأختصر الجهد فقط على المفاهيم الجديدة, نفذ الكود و يجب أن تحصل على نتيجة مشابهة لهذه الصورة:

قبل أن نختم الدرس سنضيف مصدرا للإضاءة تماما كما فعلنا سابقا و بهذا الكود:

```
//...
float LightAmbient[]= { 1.0f, 1.0f, 1.0f, 1.0f };
float LightDiffuse[]= { 1.0f, 1.0f, 1.0f, 1.0f };
float LightPosition[]= { 0.0f, 10.0f, 10.0f, 0.0f };
//...
bool init(void)
{
	//...
	glLightfv(GL_LIGHT1, GL_AMBIENT, LightAmbient);
	glLightfv(GL_LIGHT1, GL_DIFFUSE, LightDiffuse);
	glLightfv(GL_LIGHT1, GL_POSITION,LightPosition);
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT1);
	//...
	return TRUE;
}
//...
```

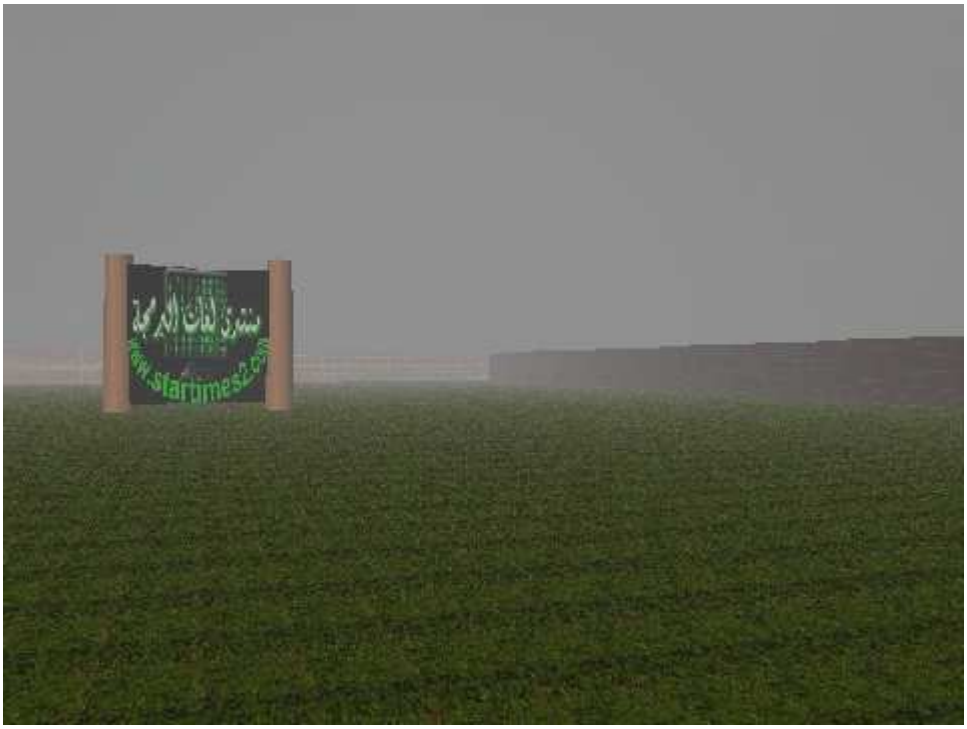
طبعا لا تقل إنك حصلت على نتيجة لا تشبه هذه:

طبعا النتيجة تبدو أكثر واقعية, ما رأيك ؟ هل نستطيع برمجة لعبة ثلاثية الأبعاد الآن ؟ و لما لا؟ بحلول الدرس العاشر سنفعل ذلك إن شاء الله, أما الآن سنضيف طبقة من الضباب لعالمنا الصغير ثلاثي الأبعاد.

على مستوى الدالة Init() و ببضع دوال نفعل ذلك:

```
//...
bool init(void)
{
	//...
	float  fog_colour[] = {0.6,0.6,0.6,0}; //لون الضباب و هنا رمادي
	glFogf(GL_FOG_START,  0); //بداية الضباب
	glFogf(GL_FOG_END,    50); //نهايته
	glFogfv(GL_FOG_COLOR,  fog_colour);  // نحدد لون الضباب
	glFogi(GL_FOG_MODE,   GL_EXP); //نوع الضباب, لا تهتم, دائما سنتعامل مع هذا النوع
	glFogf(GL_FOG_DENSITY,0.01f); //كثافة الضباب
	glEnable(GL_FOG); //نفعل الضباب
	//...
	return TRUE;
}
//...
```

النتيجة النهائية لهذا الدرس كالتالي:



لتحميل المشروع المنجز خلال هذا الدرس استخدم هذا الرابط:

http://www.mediafire.com/?mjymzmx1kmm

بهذا نختتم هذا الدرس وإلى درس قادم بإذن الله أترككم في رعاية الله.