www.startimes2.com

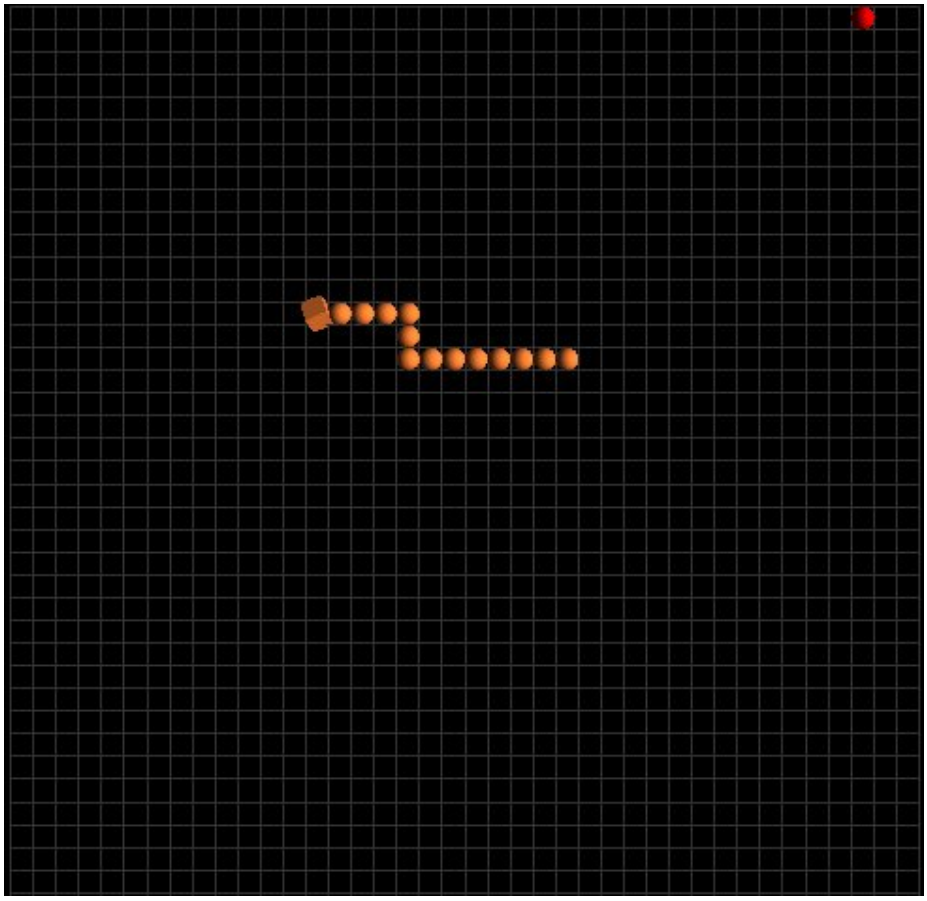
دروس في الـ OpenGL

الدرس 2: برمجة لعبة SNAKE + ترسيخ المفاهيم الأساسية

khatibe_30@hotmail.fr

2010

أهلا و سهلا بكم في هذا الدرس الثاني من دروس الـ OpenGL إن شاء الله, استنادا إلى ما رأيناه في الدرس الأول أصبح يمكننا برمجة اللعبة المشهورة SNAKE , و لترسيخ المفاهيم الأساسية للرسم الثنائي الأبعاد و أساسيات تحريك الأجسام سأشرح في هذا الدرس كيفية برمجة تلك اللعبة, و إن كنت لا تملك فكرة عما سنصل إليه في نهاية الدرس فهذه صورة مأخوذة من اللعبة... أظن أن لا أحد لا يعرف لعبة SNAKE:



أول خطوة... افتح الفيجوال ستديو و أنشأ مشروع C++ جديد و فارغ من نوع Console وسمه Snake, إذا لم تعرف كيف تفعل ذلك فراجع الدرس الأول, بعد ذلك أضف إلى المشروع ملف جديد من نوع cpp. و سمه main.cpp .

في الملف main.cpp نكتب الكود الذي ينشئ نافذة OpenGL فارغة تغلق عند الضغط على مفتاح ESC, يكون الكود كالتالي:

```
#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);
	gluLookAt(0,0,6,0,0,0,0,1,0);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	//طبعا هنا نكتب الكود الخاص بالرسم
	glutSwapBuffers();
}

void Key(unsigned char key, int x, int y )
{
	if(key ==  27 ) //ESCنغلق النافذة عند الظغط على مفتاح
		exit(0);
}

void main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
	glutInitWindowSize(500, 500); //حجم النافذة
	glutInitWindowPosition(0,0); //موقع النافذة على الشاشة
	glutCreateWindow("Snake"); // عنوان النافذة
	glutDisplayFunc(Display); //الدالة المسؤولة عن الرسم
	glutReshapeFunc(Reshape); // الدالة المسؤولة ظبط الإعدادات عند تغير حجم
النافذة
	glutKeyboardFunc(Key); //الدالة التي تستقبل أحداث لوحة المفاتيح
	glutFullScreen(); //لملئ الشاشة
	glutMainLoop();
}
```

إلى الآن لم نرى أي شيء جديد, كل هذا شاهدناه في الدرس الأول, في لعبة SNAKE هناك الأفعى و هي تتكون من عدة خلايا, قد تكون الخلايا دوائر أو مربعات حسب التصميم, و هناك أيضا ذلك الشيء الصغير الذي تأكله الأفعى و سنسميه الفأر.

لتحريك الأفعى نستعمل الإتجاهات على لوحة المفاتيح و لذلك نضيف إلى الكود الدالة التي تعالج الأحداث الناجمة عن المفاتيح الخاصة و قد رأينا ذلك في الدرس السابق, نضيف هذه الدالة إلى الكود:

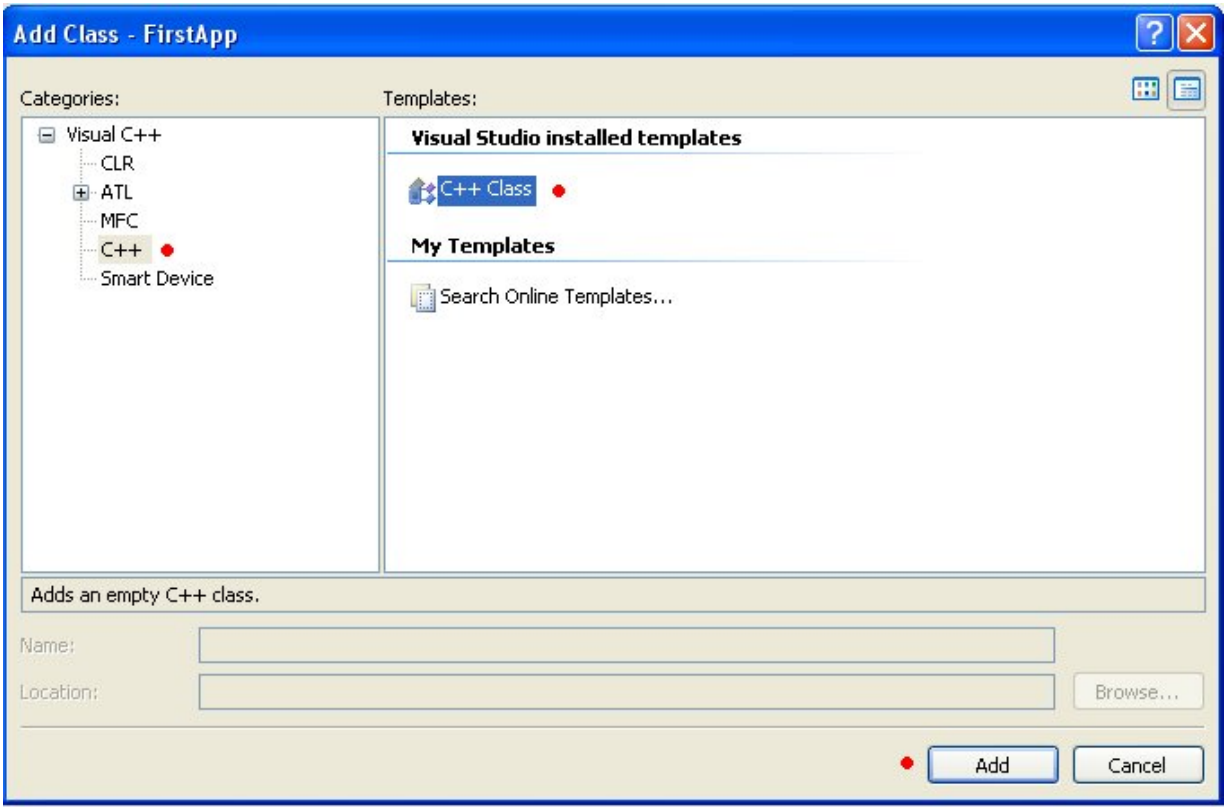
```
void SpecialKeys(int key, int x, int y)
{

}
```

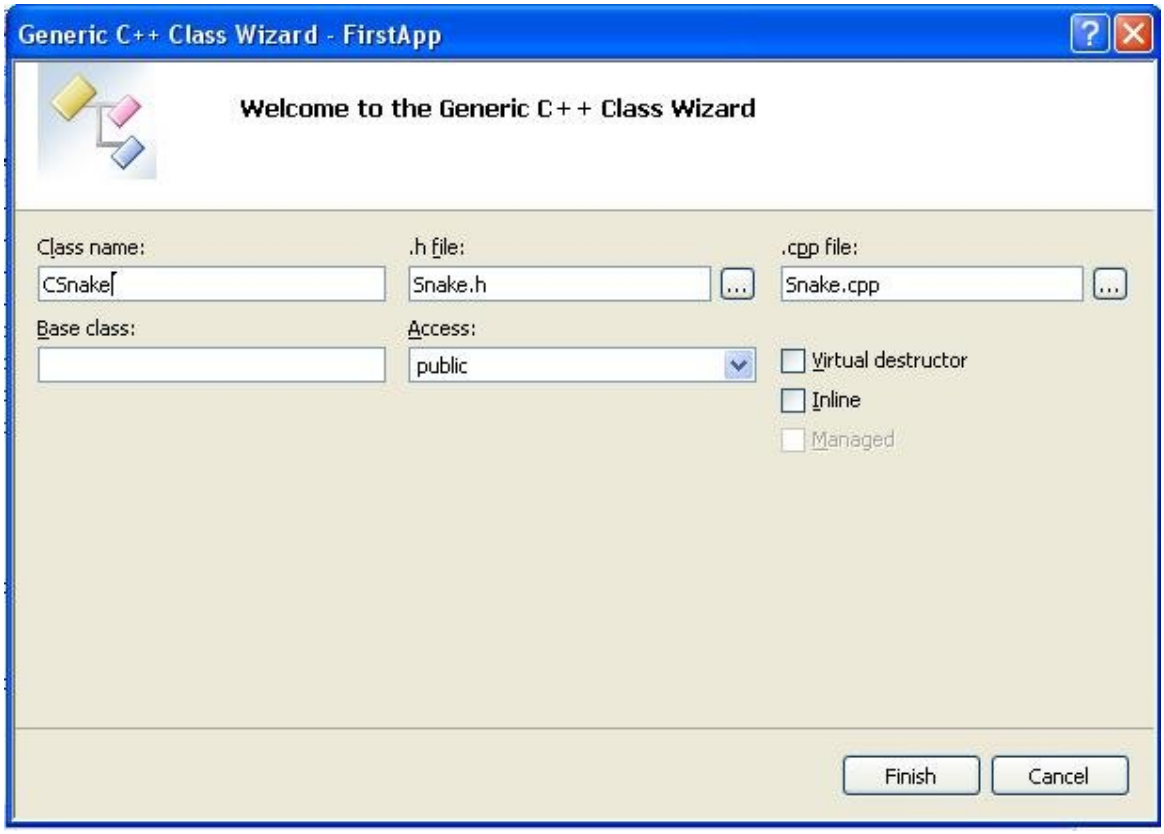
مؤقتا هي لا تفعل شيء, لتصبح الدالة فعالة نضيف هذا السطر إلى الدالة ()main وذلك كي نخبر المكتبة GLUT أن الدالة SpecialKeys هي التي تعالج أحداث المفاتيح الخاصة:

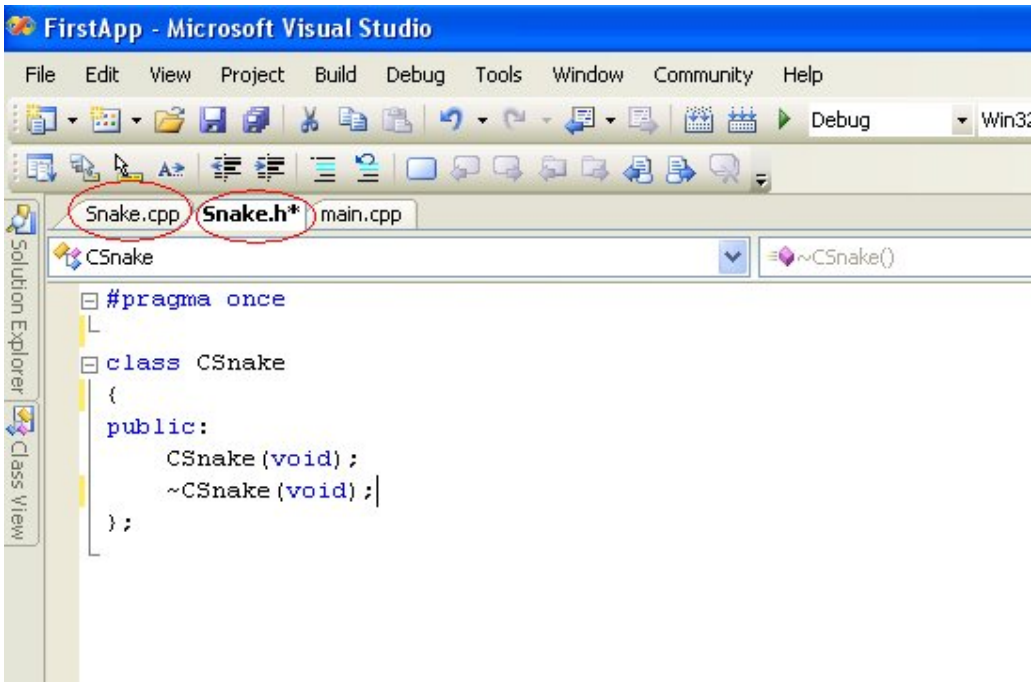
```
void main(int argc, char **argv)
{
	//...
	glutSpecialFunc(SpecialKeys);
	glutFullScreen();
	glutMainLoop();
}
```

الآن, نضيف إلى المشروع class جديدة وهذا من القائمة ...Project -> Add class ثم أكمل كما في الصور:



في خانة النص Class name أكتب CSnake وهو اسم الـ Class الجديدة, الآن افتح الملفين snake.h و snake.cpp لنبدأ كتابة دوال و خواص الفئة الجديدة:





الأفعى تتكون من خلايا, و كل خلية معرفة بالإحداثيات (x, y) ,لذلك نضيف struct جديدة ونسميها Unit في نفس الملف snake.h تمثل خلية الأفعى ليكون الكود الموجود في snake.h كالآتي:

```
#pragma once

typedef struct{
	int posx;
	int posy;
}Unit;

class CSnake
{
public:
	CSnake(void);
	~CSnake(void);
};
```

لقد أنشأنا نوع جديد إسمه Unit وهو عبارة عن struct مكونة من متغيرين posx و posy واللذان يحملان قيم إحداثيات الخلية, إذن الأفعى تتكون من مجموعة من الخلايا أي مصفوفة من النوع Unit و نضيف متغير لإسمه units_nbr والواضح من اسمه أنه يحمل عدد الخلايا الحالية المكونة للأفعى :

```
class CSnake
{
private:
	Unit units[50]; // مصفوفة خلايا الأفعى, هنا الأفعى تكبر لتحتوي على الأكثر 50 خلية
	int  units_nbr; // العدد الحالي لخلايا الأفعى, سيكون 3 في البداية, أي أن الأفعى متكونة من 3خلايا و تكبر كلما أكلت فأرا
```

طبعا المتغيرات تكون private أي خاصة, فقط الـclass من نوع CSnake يمكنها تغيير المتغيرات... طبعا هذه خصائص البرمجة كائنيه التوجه و هي ليست موضوعنا في هذا الدرس.

بعد ذلك نضيف متغير من نوع Unit يمثل الفأر, بما أننا لن نجري عليه حسابات طويلة و لا يحتوي على خصائص فإننا لن ننشئ class خاصة بالفأر بل إننا سندخله مع class الأفعى اختصارا للوقت و الكود, نسمي هذا المتغير mouse :

```
class CSnake
{
private:
	Unit units[50];
	int  units_nbr;
	Unit  mouse;  //!!!الفأر
 //...
};
```

حسنا, كيف سيكون تصميم الأفعى؟ ستكون خلاياها عبارة عن كرات صغيرة, أما رأسها سيكون عبارة عن مكعب و لإضفاء مزيد من المتعة نجعل ذلك المكعب يدور حو نفسه... لذلك ننشئ متغير آخر اسمه SnakeHeadRotateDegree.

هناك احتمالين, أن يفوز اللاعب أو يحدث تحطم للأفعى نتيجة اصطدامها بالجدار أو ذيلها, لذلك ننشئ متغيرين اثنين من نوع bool يحددان حالة الأفعى :

```
class CSnake
{
private:
	Unit units[50];
	int  units_nbr;
	Unit  mouse;
	float SnakeHeadRotateDegree; // درجة دوران رأس الأفعى
public:
	bool  IsGameOver; //؟ هل انتهت اللعبة
	bool  IsWinner; // هل فاز اللاعب؟
//...
} ;
```

مبدئيا, هذا كل ما نحتاجه من متغيرات لتقييد حركة الأفعى و التحكم بها, ننتقل إلى إضافة الدوال التي ستقوم بالتحكم في حركة الأفعى, نغير من snake.h ليصبح كما يلي:

```
class CSnake
{
//...
public: //الدوال العامة, أي أننا نستطيع استعمالهم من خارج الكائن
	CSnake(void); // دالة الباني
	~CSnake(void); // دالة الهادم
	void ResetSnake(); // تقوم بإعادة الأفعى إلى حالتها الإبتدائية
	void DrawSnake(); //تقوم برسم الأفعى
	void DrawMouse(); //تقوم برسم الفأر
	void StepForward(int direction); //تقوم بتحريك الأفعى خطوة إلى الأمام
private: //دوال خاصة, فقط الكائن من يستعملهم
	void DrawUnit(int index); // index ترسم خلية واحدة من الأفعى, الخلية
	void DrawSpecialUnit(int index); // ترسم رأس الأفعى, وهو عبارة عن خلية
خاصة
	bool IsThereCrash(int direction); //تتحقق إذا كان هنا اصطدام أم لا
};
```

كل ما قمنا به هو تعريف الدوال... نحن لم نكتب كود بنية كل دالة بعد, التعليقات الموجودة في الكود السابق تشرح وضيفة كل دالة, حسنا... نذهب إلى الملف snake.cpp لنكتب بنية كل دالة ليصبح الملف كما يلي :

```
#include "Snake.h"
#include <GL/glut.h>
#include <stdlib.h>
#include <time.h>

CSnake::CSnake(void)
{
}

CSnake::~CSnake(void)
{
}

void CSnake::ResetSnake()
{
}

void CSnake::DrawUnit(int index)
{
}

void CSnake::DrawSpecialUnit(int index)
{
}

void CSnake::DrawSnake()
{
}

void CSnake::DrawMouse()
{
}

bool CSnake::IsThereCrash(int direction)
{
	return false;
}

void CSnake::StepForward(int direction)
{
}
```

هذه هي كل الدوال التي سنحتاجها و لكننا لم نكتب أكواد الدوال بعد, قبل أن نرسم خلايا الأفعى و البقية يجب إيجاد سلم جيد للرسم إن صح التعبير, لذلك نعود إلى main.cpp .

نسينا شيئا, طبعا نحن الآن على مستوى الملف main.cpp, نضيف الدالة action() والتي تقوم بتحديث الرسم بشكل متواصل, هذا موجود في الدرس الأول :

```
void action(void)
{
	Display();;
}
//...
void main(int argc, char **argv)
{
	//...
	glutIdleFunc(action);
	glutFullScreen();
	glutMainLoop();
}
```

في الدرس السابق لم نستعمل الإضاءة, لكن هنا نضيف مصدرا للإضاءة, كيف ذلك؟ نضيف دالة جديدة init() تقوم بإنشاء مصدر إضاءة و تحديد إحداثياته ثم نستدعيها داخل الدالة main():

```
void init(void)
{
 glEnable(GL_DEPTH_TEST); //لاستعمال ذاكرة مؤقتة للرسم في عمق,
 static GLfloat lightpos[4]={1.0,0.0,1.0,0.0}; // مصفوفة تحدد إحداثيات مصدر الإضاءة و هنا الموقع هو (1, 0, 1)
 glLightfv(GL_LIGHT0,GL_POSITION,lightpos); // نعطي الموقع المحدد بالمصفوفة السابقة إلى المصدر رقم 0 للإضاءة
 glEnable(GL_LIGHTING); // تمكين الإضاءة
 glEnable(GL_LIGHT0); // تشغيل المصدر الأول رقم 0 للإضاءة
}
```

الدالة glEnable() تقوم بتمكين الخاصية المعطات لها على شكل بارمتر, مثلا ; glEnable(GL_LIGHT0) تقوم بتفعيل مصدر الإضاءة الأول و هو رقم 0, الآن نستدعي الدالة init() في الدالة الرئيسية main() :

```
void main(int argc, char **argv)
{
	//...
	init();
	glutDisplayFunc(Display);
	//...
}
```

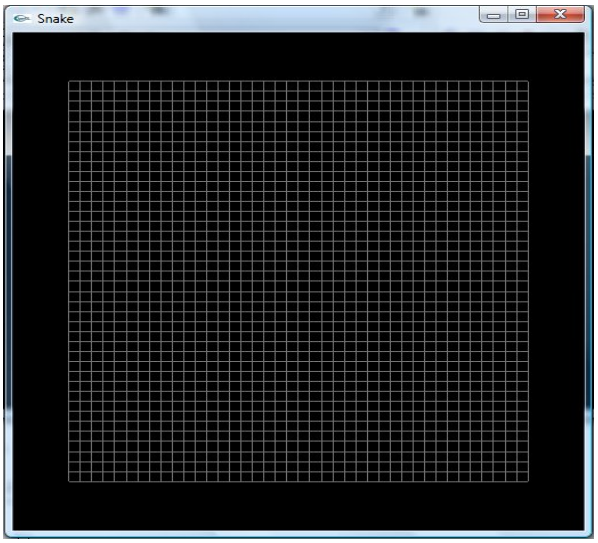
إحداثيات المعلم الذي نعمل عليه –الإحداثيات تتغير حسب بعد الكاميرا عن المعلم و استعمالات بعض الدوال- تمتد تقريبا من -2.5 إلى +2.5, لذلك سنستخدم المساحة الممتدة من -2 إلى +2 أي [-2, +2], لذلك سنقسم هذه المساحة إلى مربعات بحجم 0.1 , أي أن الفرق بين كل خط و خط سيكون 0.1 .

الآن و لتحديد مسرح الأفعى نرسم خطوط طول و خطوط عرض على امتداد المساحة المخصصة لذلك و هي من -2 إلى +2, إذن و باستعمال حلقتين واحدة للطوا و أخرى للعرض داخل الدالة Display نطبق ذلك:

```
void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	float color[4] = {0.5, 0.5, 0.5, 1};
	glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, color);
	float i;
	for(i=-2; i<=2.1; i+=0.1)
	{
		glBegin(GL_LINES); //GL_LINES تعني أننا سنرسم بين كل نقطتين خط
مستقيم
			glVertex3f(i, 2.0, 0.0); // إحداثيات النقطة الأولى والتي تتغير
فقط عند اكس
			glVertex3f(i, -2.0, 0.0); //احداثيات النقطة الثانية
		glEnd();
	}
	for(i=-2; i<=2.1; i+=0.1)
	{
		glBegin(GL_LINES);
			glVertex3f(-2, i, 0.0); // فقطyالإحداثيات هنا تتغير بالنسبة ل
			glVertex3f(2, i, 0.0);
		glEnd();
	}
	glutSwapBuffers();
}
```

الدالة glMaterialfv تقوم بإعطاء لون محدد للمادة التي سنرسمها, وهنا لإعطاء اللون color وهو مصفوفة تحمل أربعة قيم – أحمر أخضر أزرق و الإضاءة – للمادة التي سنرسها و هي خطوط الطول و العرض, بعد يأتي دور الحلقتين لرسم الخطوط و بكود واضح جدا.

نفذ البرنامج و ستحصل على هذه النتيجة :



ماذا بعد؟... نضيف إلى الملف main.cpp هذه المتغيرات:

```
//...
#include "Snake.h"

CSnake snake; // CSnake المتغير الذي يمثل الأفعى و هو من نوع
int    CurrentDirection = GLUT_KEY_RIGHT; // الإتجاه الحالي للأفعى
bool   IsGameStarted = false; //؟ هل بدأت اللعبة
char  *GameOverMessage="Game Over"; //رسالة نعرضها إذا خسر اللاعب
char  *WinnerMessage="Winner!"; //رسالة نعرضها إذا فاز اللاعب
//...
```

ما هذه المتغيرات؟ دورها واضح من التعليقات الموجودة في الكود, بالنسبة للمتغير CurrentDirection والذي سيحمل ثابت يمثل الاتجاه الحالي للأفعى, سنضع فيه قيمت الثابت الذي يمثل الزر المضغوط و هو في البداية GLUT_KEY_RIGHT الذي يمثل الاتجاه الأيمن على لوحة المفاتيح –هذا الثبت معرف في الملف الرأسي glut.h - , إذن في البداية الأفعى تتجه إلى اليمين.

متى تبدأ اللعبة؟ تبدأ إذا ضغطنا على أحد الاتجاهات, لذلك نضيف الدالة المسؤولة عن معالجة أحداث المفاتيح الخاصة و هي SpecialKeys :

```
void SpecialKeys(int key, int x, int y)
{
 CurrentDirection = key; //نعطي قيمة المفتاح الحلي إلى متغير الإتجاه الحالي
 if(!IsGameStarted)
 {
        IsGameStarted = true; //اللعبة تبدأ هنا
        snake.IsGameOver = false; // -هل انتهت اللعبة- نغير قيمة ال متغير الذي يمثل
 بالنسبة إلى الأفعى إلىfalse
        snake.IsWinner = false; // هل: كذلك نفس الشيء بالنسبة للمتغير الذي يسأل
فاز اللاعب؟
 }
}
//...
//...
void main(int argc, char **argv)
{
      //...
      glutKeyboardFunc(Key);
      glutSpecialFunc(SpecialKeys);
      glutIdleFunc(action);
      //...
}
```

الآن نحتاج إلى Timer –ومن منا لايعرفه- لنحرك به الأفعى وكذلك نتحكم بسرعة الأفعى به:

```
void SnakeForwardingTimer(int value)
{
	if(IsGameStarted) // ؟ هل بدأ المستعمل يلعب
	{
		switch(CurrentDirection){
			case GLUT_KEY_UP: //إذا كان الإتجاه الحالي: الأعلى
				snake.StepForward(GLUT_KEY_UP); // : نحرك الأفعى إلى
الأعلى
			break;
			case GLUT_KEY_DOWN: //إذا كان الإتجاه الحالي: الأسفل
				snake.StepForward(GLUT_KEY_DOWN); // : نحرك الأفعى إلى
الأسفل
			break;
			case GLUT_KEY_LEFT: //إذا كان الإتجاه الحالي: اليسار
				snake.StepForward(GLUT_KEY_LEFT); // : نحرك الأفعى إلى
اليسار
			break;
			case GLUT_KEY_RIGHT: //إذا كان الإتجاه الحالي: اليمن
				snake.StepForward(GLUT_KEY_RIGHT); // : نحرك الأفعى إلى
اليمين
			break;
			default:break;
		}
	}
	glutTimerFunc(60, SnakeForwardingTimer, value); // بهذه الدالة التايمر
ينفذ دالتنا كل 60 ملي ثانية, ومن هنا نتحكم في سرعة الأفعى
}
```

ثم نستدعي هذه الدالة بشكل عادي داخل main() :

```
void main(int argc, char **argv)
{
	//...
	glutIdleFunc(action);
	SnakeForwardingTimer(0);
	glutFullScreen();
	glutMainLoop();
}
```

الآن و بعد أن حددنا كيف تتغير الإتجاهات؟ متى يبدأ اللعب ؟ سرعة الأفعى؟ بقي لنا أن نرسم الفأر و الأفعى بشكل متواصل فإذا تغيرت إحداثيات الأفعى عند استدعاء الدالة CSnake ::StepForward() سيتغير طبعا موقع الأفعى على الشاشة.

لنرسم الأفعى و الفأر وهذا طبعا داخل الدالة Display() :

```
void Display(void)
{
	//...
	snake.DrawSnake(); //إستدعاء الدالة المسؤولة عن رسم الأفعى
	snake.DrawMouse(); //إستدعاء الدالة المسؤولة عن رسم الفأر
	glutSwapBuffers();
}
```

طبعا... هكذا و بكل بساطة, لكننا لم نكتب بعد الكود الموجود جسم كل دوال إدارة الأفعى, لذلك تنفيذ البرامج في هذه المرحلة لن يعطي أي نتيجة.

قبل أن نتجه إلى كتابة دوال الأفعى و الفأر نضيف كود بسيط لعرض رسالة تقول "انتهت اللعبة" إذا خسر المستعمل, وأخرى تقول" فائز" إذا فاز المستعمل, إذا على مستوى الدالة Display() نضيف هذا الكود :

```
void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	if(snake.IsGameOver) //هل خسر اللاعب؟
	{
		IsGameStarted = false; //...إذن اللعبة توقفت
		ShowMessage(0, 0, 0, GameOverMessage);
		snake.ResetSnake(); //...نعيد الأفعى إلى حالتها الإبتدائية
		Sleep(500);
	}
	if(snake.IsWinner) //هل فاز اللاعب؟
	{
		ShowMessage(0, 0, 0, WinnerMessage);
		snake.ResetSnake();
	}
	//...
}
```

الدالة الجديدة في الكود السابق هي ShowMessage() والتي تعرض لنا الرسالة GameOverMessage مثلا في مركز المعلم (0,0,0) , يجب أن نعرفها طبعا كما يجب التصريح عن المتغيرين GameOverMessage و WinnerMessage وذلك قبل الدالة Display() :

```
//...
CSnake snake;
int      CurrentDirection = GLUT_KEY_RIGHT;
bool   IsGameStarted = false;
char  *GameOverMessage="Game Over";
char  *WinnerMessage="Winner!";

void ShowMessage(float x, float y, float z, char *message)
{
  glDisable(GL_LIGHTING); // إطفاء الإضاءة حتى تظهر الكتابة
  glRasterPos2f(x, y);
  while (*message) {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *message);
    message++;
  }
  glEnable(GL_LIGHTING); //إعادة إشعال الكتابة
}
//...
```

حسنا, في الدالة السابقة استخدمنا الدالة glutBitmapCharacter() والتي تقوم برسم حرف معين على الشاشة و كأنه صورة نقطية, أي أن الحرف لن تكتبه على الشاشة بل سترسمه وهذا بعا حسب نوع الخريطة النقطية التي نقدمها على شكل بارامتر أول, وهذه أنواع الخطوط الموجودة :

```
GLUT_BITMAP_8_BY_13
GLUT_BITMAP_9_BY_15
GLUT_BITMAP_TIMES_ROMAN_10
GLUT_BITMAP_TIMES_ROMAN_24
GLUT_BITMAP_HELVETICA_10
GLUT_BITMAP_HELVETICA_12
GLUT_BITMAP_HELVETICA_18
```

أما البارمتر الثاني فهو الحرف الذي نريد رسمه, وبما أننا سنرسم الحرف تلو الآخر نستخدم حلقة لرسم كل الرسالة, و لتحديد الموقع الذي نبدأ منه رسم الحروف نستخدم الدالة glRasterPos2f(float x, float y) ; الواضحة جدا.

الآن أنهينا الكود الخاص بالملف main.cpp , لذلك ننتقل الآن إلى تحرير الكود المكون لدوال الكائن CSnake وهذا يتم على مستوى الملف snake.cpp

إذن نحن الآن على مستوى الملف snake.cpp , لكن قبل كتابة أي شيء, مسرح الأفعى على الشاشة يمتد من -2 إلى +2 بفارق 0.1 بين كل خانة و خانة, وحتى نتجنب التعامل مع أعداد حقيقة نفترض أن المسرح يمتد من -200 إلى +200 بفارق 10 بين كل خانة و خانة.

خلايا الأفعى هي عبارة عن مصفوفة من 50 خلية كعدد أقصى من النوع Unit لذي يتكون من متغيرين posx و posy واللذان يحملان إحداثيات هذه الخلية, ولتسهيل الحسابات ستمتد هذه الإحداثيات من -19 إلى +20 بما في ذلك لصفر و بفارق 1, لذلك فإن الإحداثيات الحقيقية realx و realy تحسب كالتالي:

```
Realx = ((posx * 10 )+ 5)  / 100 ;
Realy = ((posy * 10 )+ 5)  / 100 ;
```

طبعا قمنا بزيادة 5 للإحداثيات الافتراضية – أي بزيادة 0.05 للإحداثيات الحقيقية – حتى نحصل على مركز الخانة.

الآن, أول ما نفعله هو تحديد الحالة الابتدائية للأفعى عند بناء الكائن :

```
#include "Snake.h"
#include <GL/glut.h>
#include <stdlib.h>
#include <time.h>

CSnake::CSnake(void)
{
	srand(time(NULL)); // تهيئة جدول الأعداد العشوائية بعدد عشوائي حسب الوقت الحالي للنضام
	mouse.posx = (rand() % 40) - 19; //إعطاء إحداثيات عشوائية للفأر
	mouse.posy = (rand() % 40) - 19; //إعطاء إحداثيات عشوائية للفأر
	IsGameOver = false; //هل خسر اللاعب ؟ طبعا لا
	IsWinner = false; //هل فاز اللاعب ؟ طبعا لا
	SnakeHeadRotateDegree = 0; //درجة دوران رأس الأفعى في البداية تكون 0
	ResetSnake(); // إستدعاء الدالة التي تقوم بإعادة الأفعى إلى حالتها الإبتدائية
}
//...
```

أثناء بناء الكائن قمن باستدعاء الدالة CSnake ::ResetSnake() , ماذا تفعل ؟... هذا هو الكود الخص بها :

```
void CSnake::ResetSnake()
{
	units_nbr = 3; //الأفعى في البداية تتكون من 3 خلايا
	units[0].posx = 2;units[0].posy = 0; //(2,0) موقع الرأس هو
	units[1].posx = 1;units[1].posy = 0; //(1,0) الخلية الأولى في
	units[2].posx = 0;units[2].posy = 0; //(10) الخلية الثانية في
}
//...
```

الآن نذهب إلى الدالة DrawUnit() والتي ترسم خلية واحدة من جسم الأفعى:

```
//...
void CSnake::DrawUnit(int index) //index سنرسم الخلية ذات الرقم
{
	glPushMatrix(); //نخزن مصفوفة إحداثيات مركز المعلم و الإسقاط
	glTranslatef( //نحرك المعلم بأكمله كي نرسم الخلة في المكان المحدد
		(float)((units[index].posx * 10) - 5) / 100, // نحسب الإحداثيات
	الحقيقية باستعمالالإحداثيات الإفتراضية
		(float)((units[index].posy * 10) - 5) / 100, // نحسب الإحداثيات
	الحقيقية باستعمالالإحداثيات الإفتراضية
		0.0); //
	glutSolidSphere(0.05, 50, 50); //0.05 نرسم كرة ذات نصف قطر يساوي
	glPopMatrix(); //نستعيد مصفوفة إحداثيات مركز المعلم و الإسقاط
}
//...
```

بنفس الطريقة, نكتب الكود الخاص بالدالة DrawSpecialUnit التي تقوم برسم رأس الأفعى و هو عبارة عن مكعب يدور حول نفسه, وكذلك الدالة DrawMouse() المسؤولة عن رسم الفأر:

```
//...
void CSnake::DrawSpecialUnit(int index) //مسؤولة عن رسم رأس الأفعى
{
	glPushMatrix();
	glTranslatef(
		(float)((units[index].posx * 10) - 5) / 100,
		(float)((units[index].posy * 10) - 5) / 100,
		0.0);
	SnakeHeadRotateDegree += 2; //2 في كل مرة نزيد زاوية دوران رأس الأفعى ب
	glRotatef(SnakeHeadRotateDegree, 1, 1, 1); // ندير المعلم حسب تلك
الزاوية و على المحاور الثلاث
	glutSolidCube(0.1);
	glPopMatrix();
}

void CSnake::DrawMouse() //مسؤلة عن رسم الفأر
{
	float mouse_color[4] = {1, 0, 0, 1}; // مصفوفة تحمل لون الفأر على شكل
 قيم ل RGBA
	glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mouse_color); // نحدد
لون الفأر و هو أحمر هنا
	glPushMatrix();
	glTranslatef(
		(float)((mouse.posx * 10) - 5) / 100,
		(float)((mouse.posy * 10) - 5) / 100,
		0.0);
	glColor4f(0.3, 0.7, 0.1, 1);
	glutSolidSphere(0.05, 50, 50);
	glPopMatrix();
}
//...
```

الآن نحتاج لإضافة دالة تقوم برسم كل الأفعى, أي أنها سترسم لرأس لوحده ثم و باستعمال حلقة تقوم برسم خلايا الأفعى, نسمي هذه الالة DrawSnake() :

```
//...
void CSnake::DrawSnake()
{
	float unit_color[4] = {1, 0.5, 0.2, 1}; //مصفوفة تحمل قيم لون الفأر
	glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, unit_color);
	this->DrawSpecialUnit(0); //نرسم رأس الأفعى و هو الخلية رقم 0
	for(int i=1; i<units_nbr; i++) //باستعمال هذه الحلقة نرم كل الخلايا
		this->DrawUnit(i);
}
//...
```

بقيت الدالة الأهم, الدالة CSnake ::StepForward(int direction) ; التي تحرك الأفعى خطوة إلى الأمام حسب الإتجاه direction المعطى لها, كيف تتحرك الأفعى ؟ كيف نحسب الإحداثيات الجديدة لكل خلية عندما تتقدم الأفعى خطوة إلى الأمام؟

في الحقيقة ليت هناك حسابات معقدة, عندما تتقدم الأفعى بمقدار خلية فإن الخلية الأخيرة تأخذ إحداثيات الخلية التي قبلها و هكذا إلى أن نصل إلى الخلية رقم 0 وهي رأس الأفعى التي سنقوم بزيادة إحداثياتها بمقدار 1, مثلا إذا كانت الأفعى تتحرك إلى اليمين فإن إحداثيات x للرأس تزداد في كل مرة بمقدار 1 أما الإحداثية y بقى ثابتة و هكذا.

ولكن و قبل أن نقوم بتغيير إحداثيات رأس الأفعى علينا أن نتأكد أنه لا يوجد هناك اصطدام سواء مع الجدار الخارجي أو مع ذيل الأفعى فيحد ذاتها, وكذلك يجب أن لا ترجع الأفعى إلى الخلف, فإذا كانت تتجه إلى الأسفل لا يمكن أن نتجه مباشرة إلى الأعلى ,وأيضا أثناء تقدم الأفعى, إذا صادفت فأرا في طريقها فإن طولها سيزداد – أي أننا سنزيد من قيمة المتغير CSnake ::units_nbr بـ 1 – طولها و سيغير الفأر من موقعه, كل ذلك يكون في هذا الكود :

```
//...
void CSnake::StepForward(int direction)
{
	int temp_x = units[0].posx; // نخزن إحداثيا الرأس في متغير مؤقت قبل أن
نغيرها و ها لنعطيها للخلية رقم 1
	int temp_y = units[0].posy;
	switch(direction){
		case GLUT_KEY_UP: //إذا كان إتجاه الأفعى إلى الأعلى
			if(IsThereCrash(GLUT_KEY_UP)) // إذا كان هناك اصطدام تنتهي
اللعبة
			{
				IsGameOver = true;
				return;
			}
			else if(units[1].posy != (units[0].posy + 1)) // قبل أن نحرك
الأفعى إلى الإتجاه الجديد يجب أن نتأكد أن هذا الإتجاه ليس إلى الخلف
				units[0].posy = units[0].posy + 1;
			else // إذا كان الإتجاه الجديد عكس الإتجاه القديم فإن الأفعى تكمل
طريقها على الإتجاه القديم
				units[0].posy = units[0].posy - 1;
			break;
```

```
                case GLUT_KEY_DOWN:  //إذا كان إتجاه الأفعى إلى الأسفل
                if(IsThereCrash(GLUT_KEY_DOWN))
                {
                      IsGameOver = true;
                      return;
                }
                else if(units[1].posy != (units[0].posy - 1))
                      units[0].posy = units[0].posy - 1;
                else
                      units[0].posy = units[0].posy + 1;
                break;
          case GLUT_KEY_LEFT: //إذا كان إتجاه الأفعى إلى اليسار
                if(IsThereCrash(GLUT_KEY_LEFT))
                {
                      IsGameOver = true;
                      return;
                }
                else if(units[1].posx != (units[0].posx - 1))
                      units[0].posx = units[0].posx - 1;
                else
                      units[0].posx = units[0].posx + 1;
                break;
          case GLUT_KEY_RIGHT: //إذا كان إتجاه الأفعى إلى اليمين
                if(IsThereCrash(GLUT_KEY_RIGHT))
                {
                      IsGameOver = true;
                      return;
                }
                else if(units[1].posx != (units[0].posx + 1))
                      units[0].posx = units[0].posx + 1;
                else
                      units[0].posx = units[0].posx - 1;
                break;
          default:break;
      }
      if(units[0].posx == mouse.posx)
            if(units[0].posy == mouse.posy)
            {
                  // إذا كانت إحداثيات رأس الأفعى مساوية لإحداثيات الفأر فإننا
نزيد طول الأفعى بـ 1 و نعطي الفأر إحداثيات قديمة
                  units_nbr++;
                  mouse.posx = (rand() % 21);
                  mouse.posy = (rand() % 21);
            }
      for(int i = units_nbr-1; i>1; i--)
      {
            // نعطي لكل خلية من الأفعى إحداثيات الخلية التي قبلها باستثناء الخلية
الأولى
            units[i].posx = units[i-1].posx;
            units[i].posy = units[i-1].posy;
      }
      units[1].posx = temp_x; //نعطي الخلية الأوى الإحداثيات القديمة للرأس
      units[1].posy = temp_y;
      if(units_nbr == 49) //إذا كان عدد خلايا الأفعى يساوي 49 فإن اللاعب يفوز
            IsWinner = true;
}
//...
```

في الدالة StepForward() استعملنا الدالة IsThereCrash() والتي تعيد القيمة true إذا كان هناك تصادم, يكون كود هذه الدالة كالآتي:

```
//...
bool CSnake::IsThereCrash(int direction)
{
	int newx = units[0].posx; //متغير مؤقت نضع فيه الإحداثيات الجديدة
	int newy = units[0].posy; //متغير مؤقت نضع فيه الإحداثيات الجديدة
	switch(direction){
		case GLUT_KEY_UP: // إذا كان الإتجاه الجديد إلى الأعلى
			if(units[0].posy == 20)  // يحدث اصطدام إذا كانت الإحداثية العمودية للأفعى مساوية لـ 20, فلا يمكن أن نزيد عن 20
				return true;
			else // إذا كانت الإحداثية أقل من 20 نحسب الإحداثية الجديدة حتى نتأكد فيما بعد أنها لاتساوي إحداثية أي خلية من جسم الأفعى كي نتجنب اصطدام الأفعى بذيلها
				newy++;
			break;
		case GLUT_KEY_DOWN:
			if(units[0].posy == -19)
				return true;
			else
				newy--;
			break;
		case GLUT_KEY_LEFT:
			if(units[0].posx == -19)
				return true;
			else
				newx--;
			break;
		case GLUT_KEY_RIGHT:
			if(units[0].posx == 20)
				return true;
			else
				newx++;
			break;
		default:break;
	}
	// إذا وصلنا هنا فإن ذلك يعني أن الأفعى لم تصطدم بالجدار الخارجي و علينا أن نتأكد أنها لن تصطدم بذيلها, إذا اصطدمت نعيد القيمة true
	for(int i=2; i< units_nbr-1; i++)
		if((units[i].posx == newx) && (units[i].posy == newy))
			return true;
	return false;
}
//...
```

إلى هنا نكون أنهينا هذا الدرس, ولتحميل لعبة SNAKE التي صممناها استخدم هذا الرابط:

http://www.mediafire.com/?fto3nwv2oyv