Graficacion por Computadora
Tarea 1 OpenGL Fractales

José Ricardo Rodríguez Abreu

03/15/16

1. Algoritmo de la Alfombra de Sierpinski

Para este fractal usé dos clases:

La clase punto lo único que tiene es dos vértices que son X y Y. Me sirve para localizar los puntos de un cuadrado.

La clase Alfombra lo que contiene es un constructor que incluye 4 vértices. Los vértices que recibe marca el primer cuadro, que es el caso base de nuestra recursión. Dependiendo de que tantos niveles de nuestro fractal deseemos será las cantidades de veces que recursamos.

Lo interesante de este clase es el método draw que recibe 7 parámetros:

Next es nivel de recursión que se hará en el fractal.

4 puntos que son donde se encuentra nuestro cuadrado actual.

Un número flotante dif que nos ayuda para saber el tamaño de los siguientes cuadrados que se dibujan en la siguiente iteración.

Quitar es un booleano que no dice si una vez que pasemos por un nivel, hay que dibujar de blanco el último nivel de la recursión.

Básicamente nuestro método dibuja el cuadrado con nuestro 4 puntos, encuentra el centro de los siguiente 8 cuadrados y para cada uno de ellos, con el centro y nuestro parámetro dif, encuentra los 32 puntos de la siguiente iteración para poder dibujarlos.

Para este fractal se usaron 2 archivos (3 incluyendo el main) y 2 clases.

Alfombra.h (Declaración de funciones y clases)

```
/* ----------------------------------------------------------------------
* Alfombra.h
* versión 1.0
* Copyright (C) 2016
* José Ricardo Rodríguez Abreu,
* Facultad de Ciencias,
* Universidad Nacional Autónoma de México, Mexico.
*
```

```
* Este programa es software libre; se puede redistribuir
* y/o modificar en los términos establecidos por la
* Licencia Pública General de GNU tal como fue publicada
* por la Free Software Foundation en la versión 2 o
* superior.
*
* Este programa es distribuido con la esperanza de que
* resulte de utilidad, pero SIN GARANTÍA ALGUNA; de hecho
* sin la garantía implícita de COMERCIALIZACIÓN o
* ADECUACIÓN PARA PROPÓSITOS PARTICULARES. Véase la
* Licencia Pública General de GNU para mayores detalles.
*
* Con este programa se debe haber recibido una copia de la
* Licencia Pública General de GNU, de no ser así, visite el
* siguiente URL:  * http://www.gnu.org/licenses/gpl.html
* o escriba a la Free Software Foundation Inc.,
* 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
* ------------------------------------------------------------------------
*/
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <GL/glew.h>
#ifdef __APPLE__
# define __gl_h_
# define GL_DO_NOT_WARN_IF_MULTI_GL_VERSION_HEADERS_INCLUDED
#endif
//#include "Dependencies\freeglut\freeglut.h"
// cambienlo de acuerdo a como ustedes lo tengan
#include <OpenGL/gl3.h>
#define __gl_h_
#include <GL/freeglut.h>
#include <GL/glut.h>
#include <vector>
//#include "Dependencies\glew\glew.h"
//cambienlo de acuerdo a como ustedes lo tengan
//#include "Dependencies\freeglut\freeglut.h"
// cambienlo de acuerdo a como ustedes lo tengan
using namespace std;

/**
* Definimos una clase punto.
*/
class Punto {
/**
```

```
* Atributos x, y del punto.
*/
private:   float x,y;

/**
* Constructor y métodos.
*/
public:
Punto(float x, float y);
float getX();
float getY();
void setX(float x);
void setY(float y);
};
/**
* Definimos nuestra clase alfombra que será el fractal.
*/
class Alfombra {

/*
*
* Aquí van los atributos y los métodos privados.
*/
private:
unsigned int level;
Punto *p1,*p2,*p3,*p4;
void draw(int next,Punto *p1, Punto *p2, Punto *p3, Punto *p4,float dif,bool qui

/**
* Métodos contructores y de la clase.
*/

public:

Alfombra(int level,Punto
*p1, Punto *p2, Punto *p3, Punto *p4);
int getLevel();
void setLevel(int level);
void draw(int next);
void quita_ultimo_nivel(int next);
}; //Fin de Alfombra.h
```

Alfombra.cpp (Implementación de draw)

```
/**
* Método privado para dibujar los niveles del fractal.
```

```
*/
void Alfombra::draw(int next,
Punto *p1,
Punto *p2,
Punto *p3,
Punto *p4,
float dif,
bool quitar)
{
if(next == 0)
    return;
 if(quitar && next == 1)
  {
glColor3f(255,255,255);
 }
else
 {
glColor3f(0,0,0);
}
glBegin(GL_POLYGON);
glVertex2f(p1->getX(),p1->getY());
glVertex2f(p2->getX(),p2->getY());
glVertex2f(p3->getX(),p3->getY());
glVertex2f(p4->getX(),p4->getY());
glEnd();
glFlush();
if(next > 1)
{
float sum = sqrt(pow(p1->getY()-p2->getY(),2)+
                        pow(p1->getX()-p2->getX(),2));
dif = sum/3;
vector<Punto*> lista;
Punto *c1 = new Punto(p1->getX()-(2*dif),(p1->getY()-(sum/2)));
Punto *c2 = new Punto(p1->getX()-(2*dif),p1->getY()+(2*dif));
Punto *c3 = new Punto(p1->getX()+(sum/2),p1->getY()+(2*dif));
Punto *c4 = new Punto(p2->getX()+(2*dif),p2->getY()+(2*dif));
Punto *c5 = new Punto(p2->getX()+(2*dif),(p2->getY()-(sum/2)));
Punto *c6 = new Punto(p3->getX()+(2*dif),p3->getY()-(2*dif));
Punto *c7 = new Punto(p4->getX()+(sum/2),p4->getY()-(2*dif));
Punto *c8 = new Punto(p4->getX()-(2*dif),p3->getY()-(2*dif));
lista.push_back(c1);
lista.push_back(c2);
lista.push_back(c3);
lista.push_back(c4);
lista.push_back(c5);
lista.push_back(c6);
```

```
lista.push_back(c7);
lista.push_back(c8);
for(int i = 0; i < lista.size(); i++)
{
Punto *c = lista.at(i);
        draw(next-1,
new Punto((c->getX())-(dif/2),(c->getY())+(dif/2)),
              new Punto((c->getX())+(dif/2),(c->getY())+(dif/2)),
              new Punto((c->getX())+(dif/2),(c->getY())-(dif/2)),
              new Punto((c->getX())-(dif/2),(c->getY())-(dif/2)),dif,quitar);
     }
}

}
```

2. Arbol fractal

Para este fractal usé también dos clases:

La clase punto que definí e implementé previamente para el fractal de la Alfombra.

La clase Arbol lo que contiene es un constructor que incluye 2 vértices. Los vértices que recibe marca la primer línea, que es el caso base de nuestra recursión. Dependiendo de que tantos niveles de nuestro fractal deseemos será las cantidades de veces que recursamos.

En este caso, en nuestra clase y en su método draw sólo recibe 4 parámetros:

Next es nivel de recursión que se hará en el fractal.

2 puntos que son donde se encuentra nuestra línea actual.

Y "Quitar" que es un booleano que, como en el fractal pasado, no dice si una vez que pasemos por un nivel, hay que dibujar de blanco el último nivel de la recursión.

La única diferencia en este fractal con el otro es la rotación de las líneas. Usamos dos ángulos distintos para dale la forma a nuestro árbol (si se cambian estos ángulos, también cambia el tipo de árbol), ya que creamos nuestros ángulos en radiales, procedemos a hacer cálculos con funciones trigonométricas para obtener los nuevos dos puntos que serán las nuevas dos líneas que salen de mi punto más alto del árbol y recursamos con esos dos puntos.

Para este fractal se usaron 2 archivos (3 incluyendo el main) y 2 clases, una previamente definida.

Arbol.h: Declaración de funciones y clases

```
/* ------------------------------------------------------------------------
* Arbol.h
* versión 1.0
* Copyright (C) 2016
```

```
* José Ricardo Rodríguez Abreu,
* Facultad de Ciencias,
* Universidad Nacional Autónoma de México, Mexico.
*
* Este programa es software libre; se puede redistribuir
* y/o modificar en los términos establecidos por la
* Licencia Pública General de GNU tal como fue publicada
* por la Free Software Foundation en la versión 2 o
* superior.
*
* Este programa es distribuido con la esperanza de que
* resulte de utilidad, pero SIN GARANTÍA ALGUNA; de hecho
* sin la garantía implícita de COMERCIALIZACIÓN o
* ADECUACIÓN PARA PROPÓSITOS PARTICULARES. Véase la
* Licencia Pública General de GNU para mayores detalles.
*
* Con este programa se debe haber recibido una copia de la
* Licencia Pública General de GNU, de no ser así, visite el
* siguiente URL:  * http://www.gnu.org/licenses/gpl.html
* o escriba a la Free Software Foundation Inc.,
* 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
* ------------------------------------------------------------------------
*/
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <GL/glew.h>
#include "Alfombra.cpp"
#ifdef __APPLE__
# define __gl_h_
# define GL_DO_NOT_WARN_IF_MULTI_GL_VERSION_HEADERS_INCLUDED
#endif
//#include "Dependencies\freeglut\freeglut.h"
// cambienlo de acuerdo a como ustedes lo tengan
#include <OpenGL/gl3.h>
#define __gl_h_
#include <GL/freeglut.h>
#include <GL/glut.h>
#include <vector>
//#include "Dependencies\glew\glew.h"
//cambienlo de acuerdo a como ustedes lo tengan
//#include "Dependencies\freeglut\freeglut.h"
// cambienlo de acuerdo a como ustedes lo tengan
using namespace std;
/**
```

```
* Definimos nuestra clase alfombra que será el fractal.
*/
class Arbol {
  /**
* Aquí van los atributos y los métodos privados.
*/
private:
unsigned int level;
Punto *p1,*p2;
void draw_arbol(int next,Punto *p1, Punto *p2,bool quitar);

  /**
   * Métodos contructores y de la clase.
*/
public:
Arbol(int level,Punto *p1, Punto *p2);
int getLevel_arbol();
void setLevel_arbol(int level);
void draw_arbol(int next);
void quita_ultimo_nivel_arbol(int next);
}; //Fin de Arbol.h
```

Arbol.cpp (Implementación de draw)

```
/**
* Método privado para dibujar los niveles del fractal.
*/
void Arbol::draw_arbol(int next,Punto *p1, Punto *p2,bool quitar)
{
if(next == 0)
return;
if(quitar && next == 1)
{
glColor3f(255,255,255);
}
else
{
glColor3f(0,0,0);
}
glBegin(GL_LINES);
glVertex2f(p1->getX(),p1->getY());
glVertex2f(p2->getX(),p2->getY());
glEnd();
glFlush();
if(next > 1){
vector<Punto*> lista;
```

```
float pi = 3.14159265;
float ang1 = (60) * pi / 180;
float ang2 = (300) * pi / 180;
float factor = 0.6;
float x = (p2->getX()-p1->getX());
float y = (p2->getY()-p1->getY());
float a1 = (cos(ang1) * x) - (sin(ang1) * y);
float b1 = (sin(ang1) * x) + (cos(ang1) * y);
float a2 = (cos(ang2) * x) - (sin(ang2) * y);
float b2 = (sin(ang2) * x) + (cos(ang2) * y);

Punto *np1 = new Punto(p2->getX()+(factor*a1),p2->getY()+((factor*b1)));
Punto *np2 = new Punto(p2->getX()+(factor*a2),p2->getY()+((factor*b2)));

   lista.push_back(np1);
lista.push_back(np2);

for(int i = 0; i < lista.size(); i++)
     {
Punto *l = lista.at(i);
draw_arbol(next-1,p2,l,quitar);
}
}
} //Fin de Arbol.cpp
```