pwr**POLITECHNIKA WROCŁAWSKA**

**Instytut Informatyki, Automatyki i Robotyki**

**Zakład Systemów Komputerowych**

**Grafika komputerowa i komunikacja człowiek – komputer**

**Kurs: INEK00012L**

**Sprawozdanie z ćwiczenia nr 2**

**„OpenGL – podstawy”**

|  |  |
| --- | --- |
| **Wykonał:** | **Adrian Frydmański** |
| **Termin:** | **PN/P 12:00-15:00** |
| **Data wykonania ćwiczenia:** | **13 X 2015** |
| **Data oddania sprawozdania:** | **25 X 2015** |
| **Ocena:** |  |

|  |
| --- |
| **Uwagi prowadzącego:** |

# Wstęp teoretyczny

Dywan Sierpińskiego jest fraktalem otrzymanym z kwadratu przez podzielenie go na dziewięć mniejszych, usunięcia środkowego i ponowne rekurencyjne zastosowania tej procedury do każdego z powstałych ośmiu kwadratów. Nazwa pochodzi od nazwiska polskiego matematyka, Wacława Sierpińskiego.

# Kod źródłowy

#include <windows.h>

#include <gl/gl.h>

#include <gl/glut.h>

#include <iostream>

#include <time.h>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//globalna glebokosc dywanu

int glebokosc;

//losowy float od 0 do 1 – funkcja do szybkiego losowania składowej koloru

float rnd01()

{

return ((float)rand() / (RAND\_MAX));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Funkcja dywanowa, rekurencyjna

void dywanWaclawa(float \_x0, float \_y0, float \_x1, float \_y1, int glebokosc)

{

if (glebokosc > 0)

{

//tablica punktów dla mniejszych dywanów Wacława

float x[4], y[4];

//definiowanie wierzchołków mniejszych dywaników

x[0] = \_x0;

x[1] = (\_x1 - \_x0) / 3.0f;

x[2] = 2.0f \* x[1];

x[3] = 3.0f \* x[1];

for (int i = 1; i < 4; i++)

x[i] += x[0];

y[0] = \_y0;

y[1] = (\_y1 - \_y0) / 3.0f;

y[2] = 2.0f \* y[1];

y[3] = 3.0f \* y[1];

for (int i = 1; i < 4; i++)

y[i] += y[0];

//zmniejszenie głębokości dla rekurencji, która ma się w końcu skończyć

glebokosc--;

//rysowanie mniejszych dywaników

//pętla po wysokości

for (int i = 0; i < 3; i++)

{

//pętla po szerokości

for (int j = 0; j < 3; j++)

{

if (i != 1 || j != 1)

dywanWaclawa(x[i], y[j], x[i+1], y[j+1], glebokosc);

}

}

}

else

{

//ustawienie aktualnego koloru rysowania

glColor3f(rnd01(), rnd01(), rnd01());

//rysowanie kwadratu

glRectf(\_x0, \_y0, \_x1, \_y1);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Funkcaja określająca, co ma być rysowane

// (zawsze wywoływana, gdy trzeba przerysować scenę)

void RenderScene(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

// Czyszczenie okna aktualnym kolorem czyszczącym

dywanWaclawa(-90.0f, 90.0f, 90.0f, -90.0f, glebokosc);

glFlush();

// Przekazanie poleceń rysujących do wykonania

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Funkcja ustalająca stan renderowania

void MyInit(void)

{

glClearColor(0, 0, 0, 1.0f);

// Kolor okna wnętrza okna - ustawiono na szary

//resetowanie randa

srand(time(NULL));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Funkcja służąca do kontroli zachowania proporcji rysowanych obiektów

// niezależnie od rozmiarów okna graficznego

void ChangeSize(GLsizei horizontal, GLsizei vertical)

// Parametry horizontal i vertical (szerokość i wysokość okna) są

// przekazywane do funkcji za każdym razem, gdy zmieni się rozmiar okna

{

GLfloat AspectRatio;

// Deklaracja zmiennej AspectRatio określającej proporcję wymiarów okna

if (vertical == 0)

// Zabezpieczenie pzred dzieleniem przez 0

vertical = 1;

glViewport(0, 0, horizontal, vertical);

// Ustawienie wielkościokna okna urządzenia (Viewport)

// W tym przypadku od (0,0) do (horizontal, vertical)

glMatrixMode(GL\_PROJECTION);

// Określenie układu współrzędnych obserwatora

glLoadIdentity();

// Określenie przestrzeni ograniczającej

AspectRatio = (GLfloat)horizontal / (GLfloat)vertical;

// Wyznaczenie współczynnika proporcji okna

// Gdy okno na ekranie nie jest kwadratem wymagane jest

// określenie okna obserwatora.

// Pozwala to zachować właściwe proporcje rysowanego obiektu

// Do określenia okna obserwatora służy funkcja glOrtho(...)

if (horizontal <= vertical)

glOrtho(-100.0, 100.0, -100.0 / AspectRatio, 100.0 / AspectRatio, 1.0, -1.0);

else

glOrtho(-100.0\*AspectRatio, 100.0\*AspectRatio, -100.0, 100.0, 1.0, -1.0);

glMatrixMode(GL\_MODELVIEW);

// Określenie układu współrzędnych

glLoadIdentity();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Główny punkt wejścia programu. Program działa w trybie konsoli

void main(void)

{

cout << "Podaj glebokosc\n";

cin >> glebokosc;

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGBA);

// Ustawienie trybu wyświetlania

// GLUT\_SINGLE - pojedynczy bufor wyświetlania

// GLUT\_RGBA - model kolorów RGB

glutCreateWindow("Lab2 - 209865");

// Utworzenie okna i określenie treści napisu w nagłówku okna

glutDisplayFunc(RenderScene);

// Określenie, że funkcja RenderScene będzie funkcją zwrotną (callback)

// Biblioteka GLUT będzie wywoływała tą funkcję za każdym razem, gdy

// trzeba będzie przerysować okno

glutReshapeFunc(ChangeSize);

// Dla aktualnego okna ustala funkcję zwrotną odpowiedzialną za

// zmiany rozmiaru okna

MyInit();

// Funkcja MyInit (zdefiniowana powyżej) wykonuje wszelkie

// inicjalizacje konieczneprzed przystąpieniem do renderowania

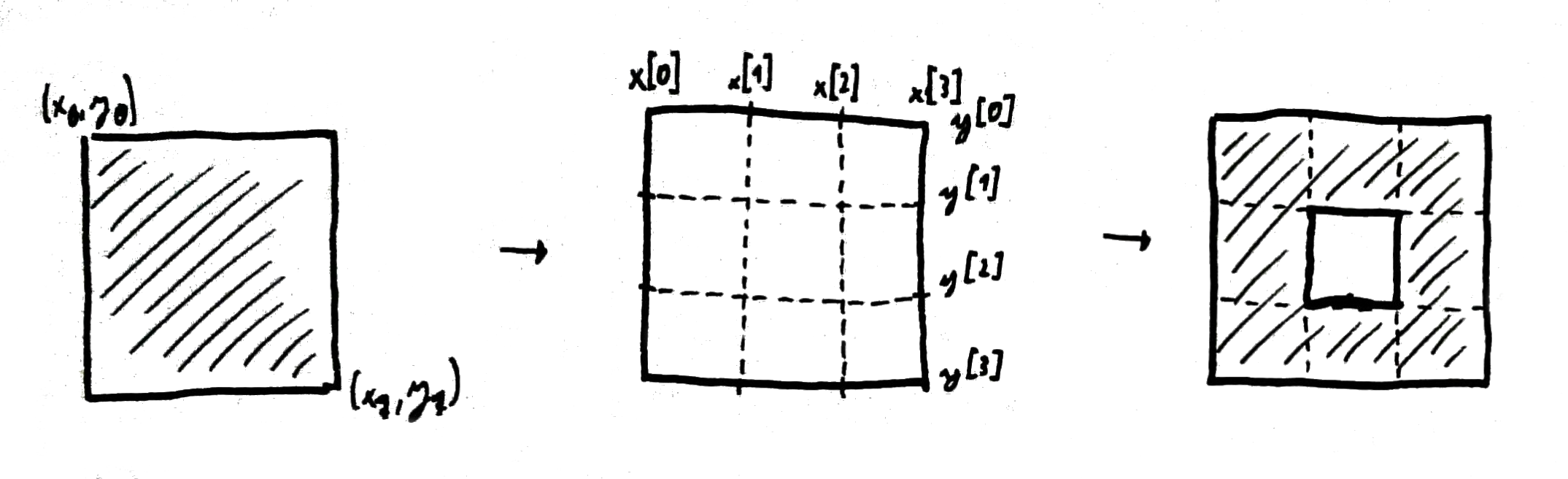
glutMainLoop();

// Funkcja uruchamia szkielet biblioteki GLUT

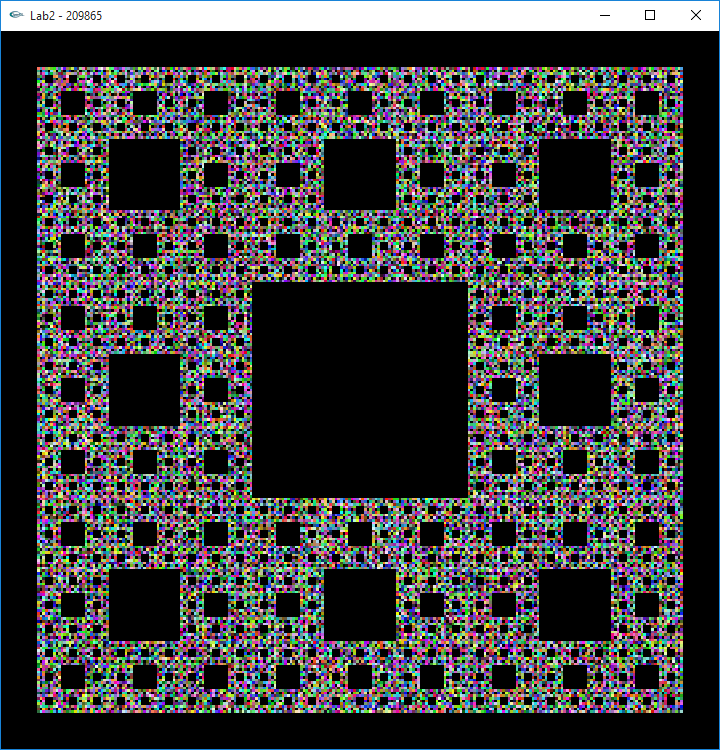
}

# Opis działania „funkcji dywanowej”

Funkcja dywanowa (zaznaczona w kodzie na szaro) jest rekurencyjną funkcją rysującą Dywan Sierpińskiego o określonej „głębokości”, która mówi o liczbie kolejnych „wycięć” kwadratu w większym kwadracie.

  
Rys. 1 schematyczny opis dzielenia kwadratu na mniejsze.

Głównym zadaniem jej jest podział istniejącego kwadratu na 9 mniejszych (bez rysowania środkowego). Wyliczane są nowe współrzędne wierzchołków i dla nich wywoływane są kolejne funkcje. Normalnie rekurencja trwa w nieskończoność, dlatego ograniczeniem jest zmniejszająca się z każdym poziomem głębokość, która to osiągając poziom 0 zmienia działanie funkcji dywanowej na rysowanie prostokąta, bez wywoływania podfunkcji.

  
Rys. 2 wynik działania programu dla ustalonej głębokości równej 5.

# Podsumowanie

Tak skonstruowany mechanizm rysowania Dywanu Sierpińskiego jest dość prostym rozwiązaniem, ale nie idealnym. Aby przy rysowaniu przemieszczać kwadraty nie wystarczy pozmieniać liczb w tablicach x i y, a należy to robić przy rysowaniu każdego kwadratu, przez co liczba wywołań funkcji losowej wzrasta dwukrotnie – prościej byłoby wywoływać tylko dla dwóch współrzędnych – współrzędnych środka kwadratu. Zaletą tego rozwiązania jest intuicyjność.