POLITECHNIKA WROCŁAWSKA

Grafika komputerowa i komunikacja człowiek-komputer

Kurs: INEK00012L

Sprawozdanie z ćwiczenia nr 2

Dywan Sierpińskiego

Wykonał:	**Janusz Pelc 252799**
Termin:	**Np. PN/TP 7:30-10:30**
Data wykonania ćwiczenia:	**11 Październik 2021**
Data oddania sprawozdania:	**15 Październik 2021**
Ocena:	

Uwagi prowadzącego:

1. **Wstęp**

Ćwiczenie polegało na narysowaniu dywanu Sierpińskiego za pomocą biblioteki OpenGL.

Dywan Sierpińskiego to figura geometryczna fraktalna. Jest to kwadrat, który zostaje podzielony na 9 mniejszych, równych kwadratów i zostaje usunięty środkowy. Tą samą operację wykonujemy na otrzymanych mniejszych kwadratach i każdych kolejnych otrzymanych po operacjach. W rezultacie otrzymuje figurę o polu powierzchni równym 0.

Perturbacja w tym ćwiczeniu to przesunięcie wierzchołków kwadratów o losowy wektor, zależny od współczynnika perturbacji.

2. **Program**

Program jest napisy w języku c++ za pomocą kompilatora Visual Studio 19 i jest bazowany na kodzie z instrukcji ćwiczenia. Główną funkcją programu jest ChangeSize(), tworzy ona i określa parametry środowiska, w którym rysowany będzie dywan. Figura powstaje w funkcji RenderScene(), która wywołuje funkcję Dywan(). Inicjuje ona wektory wierzchołków kwadratu na aktualnym stopniu dywanu i wyznacza ich wartości. Następnie funkcja wywołuje samą siebie dla każdego z ośmiu nowych kwadratów, wyliczając ich nowe współrzędne oraz ustawiając długość boku na 3 razy krótszą. Funkcja kończy działanie kiedy wszystkie wewnętrzne funkcje Dywan() zakończą działanie, lub gdy narysuje kwadrat za pomocą funkcji Kwadrat(), co zachodzi tylko na najniższym stopniu.

```
//-----------------------------------------------------------------
//funkcja rysuje dywan Sierpinskiego
//-----------------------------------------------------------------

void Dywan(int n,double bok, double x, double y) {
	double a[2]; //współrzedne wierzchołka kwadratu
	double b[2];
	double c[2];
	double d[2];

	if (n > 0) {
//wyznaczenie wierzchołków kwadratu ze środkiem w [x,y]
		a[0] = x + bok / 6;
		a[1] = y - bok / 6;
		b[0] = x - bok / 6;
		b[1] = y - bok / 6;
		c[0] = x - bok / 6;
		c[1] = y - bok / 6;
		d[0] = x - bok / 6;
		d[1] = y - bok / 6;

		n--;
		bok /= 3;
//W miejscach kwadratów, z których jest zbudowany aktualny dywan, rysuje 8 nowych
		Dywan(n, bok, x - bok, y - bok);
		Dywan(n, bok, x - bok, y);
		Dywan(n, bok, x - bok, y + bok);
		Dywan(n, bok, x, y + bok);
		Dywan(n, bok, x + bok, y + bok);
		Dywan(n, bok, x + bok, y);
		Dywan(n, bok, x + bok, y - bok);
		Dywan(n, bok, x, y - bok);
	}
	else {
		Kwadrat(bok, x, y);
	}
}
```

Argumentami funkcji jest aktualny stopień, na którym znajduje się kwadrat oraz współrzędne jego środka. Współrzędne środków mniejszych kwadratów są wyznaczane poprzez przesunięcie ze środka większego kwadratu o 1/6 długości jego boku lub 0 wzdłuż obu osi na wszystkie sposoby, bez wektora [0,0].

Funkcja Kwadrat() rysuje kwadrat na pomocą GL_POLYGON, ponieważ po perturbacji figura nie będzie już prostopadła. W zależności od wybranego trybu, dywan może być rysowany jako bały bez perturbacji, lub z perturbacjami i losowymi gradientami. W obu przypadkach wierzchołki są wyznaczane na podstawie środka kwadratu, lecz w wersji z perturbacjami do współrzędnych dodawane są unikatowe wartości wyznaczone na podstawie stopnia w dywanie oraz współczynnika perturbacji. W wersji z kolorowej, dla każdego wierzchołka, każdego kwadratu losowana jest wartość RGB. Funkcja wykonuje się jedynie dla kwadratów na najniższym stopniu.

```
//-----------------------------------------------------------------
//funkcja rysuje kwadrat
//-----------------------------------------------------------------

void Kwadrat(double a, double x, double y) {
	double		v;
	glBegin(GL_POLYGON);
	if (white) {
		glColor3f(1, 1, 1); //ustawienie białego koloru wierzchołka
		glVertex2f(x - a / 2, y - a / 2); //wyznaczenie wierzchołków kwadratu ze
środkiem w [x,y]

		glColor3f(1, 1, 1);
		glVertex2f(x - a / 2, y + a / 2);

		glColor3f(1, 1, 1);
		glVertex2f(x + a / 2, y + a / 2);

		glColor3f(1, 1, 1);
		glVertex2f(x + a / 2, y - a / 2);
	}
	else {
		v = a * (rand() % 10 + 1) / 100 * per;  //losowe przesunięcie kwadratu
		zależne od natężenia i proporcjonalne do wielkości
		glColor3f(float(rand() % 2), float(rand() % 2), float(rand() % 2));
		//ustwienie losowego koloru wierzchołka
		glVertex2f(x - a / 2 + v, y - a / 2 + v);	//wyznaczenie
		wierzchołków kwadratu ze środkiem w [x,y]

		v = a * (rand() % 10 + 1) / 100 * per;  //losowe przesunięcie kwadratu
		zależne od natężenia i proporcjonalne do wielkości
		glColor3f(float(rand() % 2), float(rand() % 2), float(rand() % 2));
		glVertex2f(x - a / 2 + v, y + a / 2 + v);

		v = a * (rand() % 10 + 1) / 100 * per;  //losowe przesunięcie kwadratu
		zależne od natężenia i proporcjonalne do wielkości
		glColor3f(float(rand() % 2), float(rand() % 2), float(rand() % 2));
		glVertex2f(x + a / 2 + v, y + a / 2 + v);

		v = a * (rand() % 10 + 1) / 100 * per;  //losowe przesunięcie kwadratu
		zależne od natężenia i proporcjonalne do wielkości
		glColor3f(float(rand() % 2), float(rand() % 2), float(rand() % 2));
		glVertex2f(x + a / 2 + v, y - a / 2 + v);
	}

	glEnd();
		}
```

3. Wnioski

Największym wyzwaniem w ćwiczeniu było wymyślenie algorytmu wyznaczania pozycji coraz to mniejszych kwadratów.

Na początku próbowałem w miejscach wyciętych wyznaczyć strefę zablokowaną i kazać programowi wszystkie 9 kwadratów, licząc na to, że pominie środkowy. To rozwiązanie okazało się niepotrzebnie skomplikowane.

Kolejnym problemem było pozbycie się deformacji rysowanych kwadratów, co jest szczególnie widoczne przy większej liczbie stopni dywanu. Próbowałem rozwiązać ten problem zamieniając typ zmiennych z float na double aby zniwelować błędy zaokrąglania wartości, lecz nie przyniosło to skutków.

4. Kod źródłowy

```
//*************************************************************************
//
//  PLIK ŹRÓDŁOWY:          GK_LAB1_PELC.cpp
//
//  OPIS:                           Program służy do tworzenia dywanu sierpinskiego z
perturbacjami
//
//  AUTOR:                          Janusz Pelc
//
//  DATA                            14.10.2021
//     MODYFIKACJI:
//
//  PLATFORMA:                      Kompilator:         Microsoft Visual studio 19.
//
//  MATERIAŁY               Instrukcja ćwiczenie 2 - OpenGL podstawy
//     ŹRÓDŁOWE:
//
//  UŻYTE BIBLIOTEKI Nie używano.
//  NIESTANDARDOWE
//
//*************************************************************************

#include <windows.h>
#include <iostream>
#include <gl/gl.h>
#include <gl/glut.h>
#include <cstdlib>
#include <time.h>

using namespace std;

double per; //natezenie perturbacji
double def_bok = 200; //wielkosc dywanu
int stopien; //stopien dywanu
bool white; //1 = biały bez perturbacji

//-------------------------------------------------------------------
//funkcja rysuje kwadrat
//-------------------------------------------------------------------

void Kwadrat(double a, double x, double y) {
	double		v;
	glBegin(GL_POLYGON);

	if (white) {
		glColor3f(1, 1, 1); //ustawienie białego koloru wierzchołka
		glVertex2f(x - a / 2, y - a / 2); //wyznaczenie wierzchołków kwadratu ze
środkiem w [x,y]

		glColor3f(1, 1, 1);
		glVertex2f(x - a / 2, y + a / 2);

		glColor3f(1, 1, 1);
		glVertex2f(x + a / 2, y + a / 2);

		glColor3f(1, 1, 1);
		glVertex2f(x + a / 2, y - a / 2);
	}
	else {
		v = a * (rand() % 10 + 1) / 100 * per;  //losowe przesunięcie kwadratu
zależne od natężenia i proporcjonalne do wielkości
```

```
			glColor3f(float(rand() % 2), float(rand() % 2), float(rand() % 2));
		//ustwienie losowego koloru wierzchołka
			glVertex2f(x - a / 2 + v, y - a / 2 + v);	//wyznaczenie
wierzchołków kwadratu ze środkiem w [x,y]

			v = a * (rand() % 10 + 1) / 100 * per;  //losowe przesunięcie kwadratu
zależne od natężenia i proporcjonalne do wielkości
			glColor3f(float(rand() % 2), float(rand() % 2), float(rand() % 2));
			glVertex2f(x - a / 2 + v, y + a / 2 + v);

			v = a * (rand() % 10 + 1) / 100 * per;  //losowe przesunięcie kwadratu
zależne od natężenia i proporcjonalne do wielkości
			glColor3f(float(rand() % 2), float(rand() % 2), float(rand() % 2));
			glVertex2f(x + a / 2 + v, y + a / 2 + v);

			v = a * (rand() % 10 + 1) / 100 * per;  //losowe przesunięcie kwadratu
zależne od natężenia i proporcjonalne do wielkości
			glColor3f(float(rand() % 2), float(rand() % 2), float(rand() % 2));
			glVertex2f(x + a / 2 + v, y - a / 2 + v);
		}

		glEnd();
}

//-----------------------------------------------------------------
//funkcja rysuje dywan Sierpinskiego
//-----------------------------------------------------------------

void Dywan(int n,double bok, double x, double y) {
		double a[2]; //współrzedne wierzchołka kwadratu
		double b[2];
		double c[2];
		double d[2];

		if (n > 0) {

			//wyznaczenie wierzchołków kwadratu ze środkiem w [x,y]
			a[0] = x + bok / 6;
			a[1] = y - bok / 6;

			b[0] = x - bok / 6;
			b[1] = y - bok / 6;

			c[0] = x - bok / 6;
			c[1] = y - bok / 6;

			d[0] = x - bok / 6;
			d[1] = y - bok / 6;

			n--;
			bok /= 3;
			//W miejscach kwadratów, z których jest zbudowany aktualny dywan, rysuje
8 nowych
			Dywan(n, bok, x - bok, y - bok);
			Dywan(n, bok, x - bok, y);
			Dywan(n, bok, x - bok, y + bok);
			Dywan(n, bok, x, y + bok);
			Dywan(n, bok, x + bok, y + bok);
			Dywan(n, bok, x + bok, y);
			Dywan(n, bok, x + bok, y - bok);
			Dywan(n, bok, x, y - bok);
		}
```

```
	else {
		Kwadrat(bok, x, y);
	}
}

void RenderScene(void){
	glClear(GL_COLOR_BUFFER_BIT);

	Dywan(stopien,def_bok,0,0);

	glFlush();
}

//-----------------------------------------------------------------
// Funkcja ustalająca stan renderowania
//-----------------------------------------------------------------

void MyInit(void){

	glClearColor(0.2f, 0.2f, 0.2f, 1.0f);
	// Kolor okna wnętrza okna
}

//-----------------------------------------------------------------
// Funkcja służąca do kontroli zachowania proporcji rysowanych obiektów
// niezależnie od rozmiarów okna graficznego
//-----------------------------------------------------------------

void ChangeSize(GLsizei horizontal, GLsizei vertical){

	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 stopien dywanu: ";
	cin >> stopien;

	cout << "Czy wersja biala bez perturbacji?: (t/n)";

	char x;
	do {
		cin >> x;
		if (x == 't') white = true;
		else if (x == 'n') {
			white = false;
			cout << "Podaj natezenie perturbacji (zalecane 0-10): ";
			cin >> per;
		}
	} while (x != 't' && x != 'n');


	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
	// Ustawienie trybu wyświetlania
	// GLUT_SINGLE - pojedynczy bufor wyświetlania
	// GLUT_RGBA - model kolorów RGB

	glutCreateWindow("Dywan Sierpińskiego");
	// 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
		}
```