Sprawozdanie z ćwiczenia 2
Open GL - podstawy – Dywan Sierpińskiego
Maksymilian Iwanow
209946
Poniedziałek 10.15

Celem laboratorium było stworzenie programu rysującego losowo zdeformowany dywan Sierpińskiego z wykorzystaniem biblioteki graficznej OpenGL wraz z rozszerzeniem GL Utility Toolkit (GLUT) za pomocą funkcji rekurencyjnej. Algorytm rysowania dywanu polega na stworzeniu jednego dużego kwadratu a następnie wycinaniu mniejszych kwadratów o polu 1/9 pola wyjściowego kwadratu.

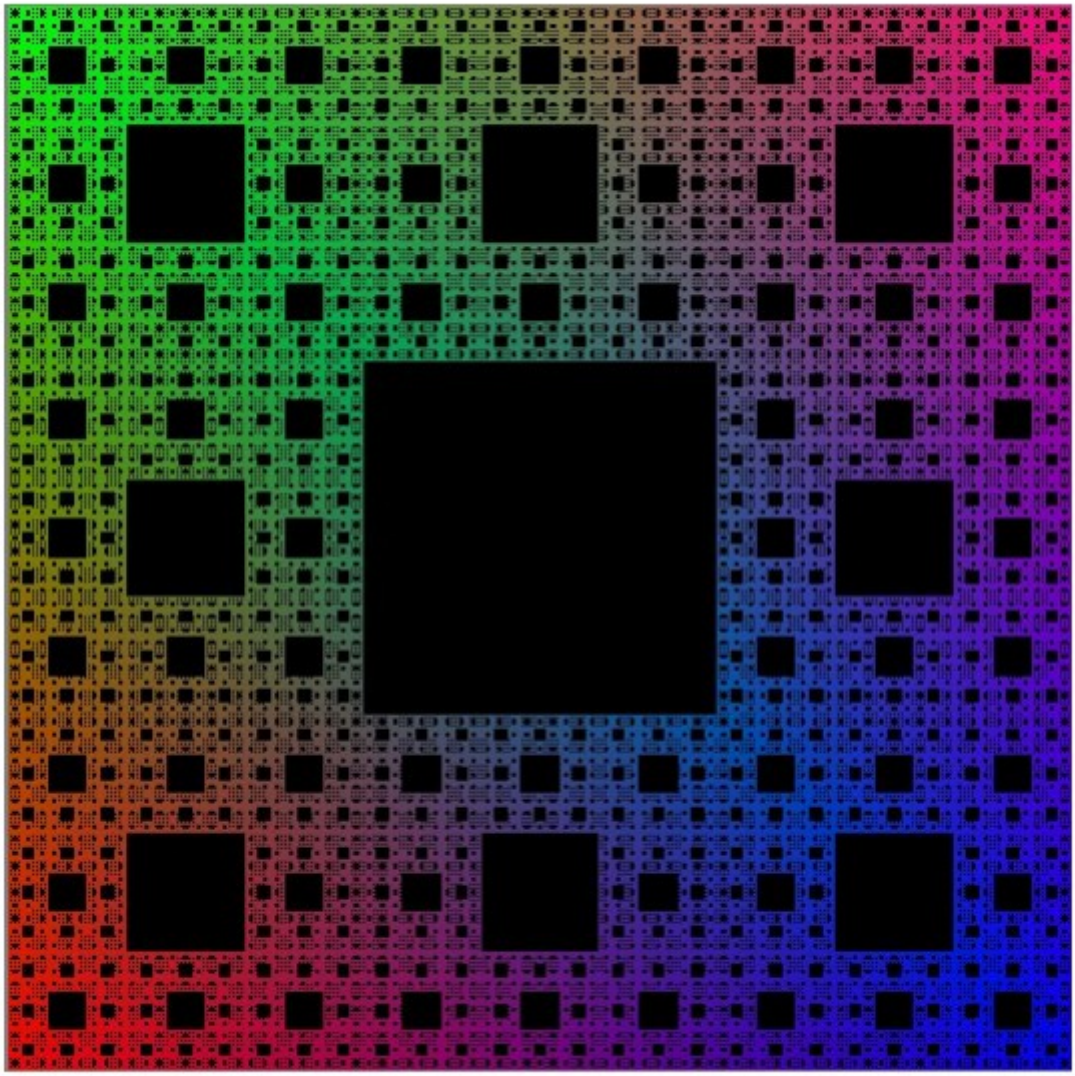
Algorytm:

Na wejściu podawane są 4 współrzędne kwadratu początkowego oraz liczba powtórzeń funkcji rekurencyjnej.

Dzielenie boku na 3 części – wyznaczone zostają współrzędne punktów dzielących – po 2 punkty na każdy bok kwadratu.

Wyznaczony zostaje kwadrat znajdujący się w środku. Zostaje zamalowany na kolor tła.

Dla pozostałych kwadratów wyznaczonych przez, sumarycznie, 16 punktów (4 punkty początkowe kwadratu, 8 punktów na bokach oraz 4 punkty kwadratu środkowego) powtarzana jest rekurencja.



Wynik programu tworzącego dywan Sierpińskiego rekurencyjnie bez deformacji dla liczby powtórzeń równej 10.

Wnioski i przemyślenia:

Największym problemem było zaprojektowanie odpowiedniego algorytmu do rysowania dywanu. Nie był to problem natury programistycznej, lecz czysto algorytmicznej. Drugą trudnością było wytworzenie perturbacji. Udało się to dla kwadratów wewnętrznych, to jest wycinanych. Należy pamiętać, że zajęcia były pierwszymi z wykorzystaniem biblioteki OpenGL, więc niektóre rzeczy, które jeszcze nie są intuicyjne, niedługo takie się zapewne staną.

Kod programu:

```
#include <windows.h>
#include <gl/GL.h>
#include <GL/glut.h>
#include <math.h>
#include <iostream>
#include <time.h>
int powtu;
int peru;

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(-1.0, 1.0, -1.0 / AspectRatio, 1.0 / AspectRatio, 1.0, -1.0);
	else
		glOrtho(-1.0*AspectRatio, 1.0*AspectRatio, -1.0, 1.0, 1.0, -1.0);

	glMatrixMode(GL_MODELVIEW);
	// Określenie układu współrzędnych
	glLoadIdentity();
}
void czworo(float a[],float b[],float c[], float d[], int powt,int per) //funkcja
rekurencyjna - powt - liczba powtórzeń funkcji, a,b,c,d - współrzędne kwadratu wejściowego,
per - perturbacje
{
	//Każdy bok dzielimy na 3 części:
	//temp1,temp11 - współrzędne punktów dzielących na boku AB
	//temp2,temp22 - współrzędne punktów dzielących na boku BC
	//temp3,temp33 - etc...
	//Następnie wyznaczamy kwadrat znajdujący się w środku i malujemy go na kolor tła -
```

```
jego współrzędne to p1,p2,p3,p4
	//Rysujemy na czarno środkowy kwadrat
	//dla pozostalych 8 kwadratów robimy rekurencję
	powt = powt - 1;
	float temp1[2];
	float temp2[2];
	float temp3[2];
	float temp4[2];
	float temp11[2];
	float temp22[2];
	float temp33[2];
	float temp44[2];
	//AB
	temp1[0] = a[0];
	temp1[1] = a[1]+( (b[1]-a[1]) / 3 );
	temp11[0] = a[0];
	temp11[1] = temp1[1]+( (b[1]-a[1]) / 3 );
	//BC
	temp2[0] = b[0]+( ( c[0]-b[0] ) / 3 );
	temp2[1] = b[1];
	temp22[0] = temp2[0]+( ( c[0]-b[0] ) / 3 );
	temp22[1] = b[1];
	//CD
	temp3[0] = c[0];
	temp3[1] = c[1]+( (d[1]-c[1]) / 3);
	temp33[0] = c[0];
	temp33[1] = temp3[1]+( (d[1]-c[1]) / 3);
	//DA
	temp4[0] = d[0]+( (a[0]-d[0]) / 3);
	temp4[1] = d[1];
	temp44[0] = temp4[0]+( (a[0]-d[0]) / 3);
	temp44[1] = d[1];
	float p1[2]; p1[0] = temp2[0]; p1[1] =temp11[1] ;
	float p2[2]; p2[0] = temp22[0]; p2[1] = temp3[1];
	float p3[2]; p3[0] = temp4[0]; p3[1] =temp33[1] ;
	float p4[2]; p4[0] = temp44[0]; p4[1] = temp1[1];

	float tur = powt*((25/(float)RAND_MAX)*per);

	glBegin(GL_POLYGON);
	glColor4f(0.0f, 0.0f, 0.0f, 0.0f);
	glVertex2f((a[0]+(d[0]-a[0])/3)-tur, (a[1]+(b[1]-a[1])/3)+tur);
	glVertex2f((b[0]+(c[0]-b[0])/3)+tur, (b[1]+(a[1]-b[1])/3)-tur);
	glVertex2f((c[0]+(b[0]-c[0])/3)-tur, (c[1]+(d[1]-c[1])/3)+tur);
	glVertex2f((d[0]+(a[0]-d[0])/3)+tur, (d[1]+(c[1]-d[1])/3)-tur);
	glEnd();

	if (powt > 0)
	{
		powt = powt - 1;

		czworo(a,temp1,p4,temp44,powt,per);
		czworo(temp1,temp11,p1,p4,powt,per);
		czworo(temp11,b,temp2,p1,powt,per);
		czworo(p1,temp2,temp22,p2,powt,per);
		czworo(p2, temp22, c, temp3, powt,per);
		czworo(p3, p2, temp3, temp33, powt,per);
		czworo(temp4, p3, temp33, d, powt,per);
		czworo(temp44, p4, p3, temp4, powt,per);

	}

}
void RenderScene(void)
{
```

```
	float tuur = powtu*((25/(float)RAND_MAX)*peru);
	int dziel = 20+80*peru;

	glClear(GL_COLOR_BUFFER_BIT);
	float a[2] = { -0.75, 0.75 };
	float b[2] = { -0.75, -0.75 };
	float c[2] = {  0.75, -0.75 };
	float d[2] = {  0.75, 0.75  };


	glBegin(GL_POLYGON);

		glColor3ub( rand()%255, rand()%255, rand()%255 );
	glVertex2f(a[0],a[1]);
	for (int i=0;i<dziel;i++){glColor3ub( rand()%255, rand()%255, rand()%255 );
glVertex2f(a[0],a[1]-((1.5/dziel)*i));}

		glColor3ub( rand()%255, rand()%255, rand()%255 );
	glVertex2f(b[0],b[1]);
	for (int i=0;i<dziel;i++){glColor3ub( rand()%255, rand()%255, rand()%255 );
glVertex2f(b[0]+((1.5/dziel)*i),b[1]) ;}

		glColor3ub( rand()%255, rand()%255, rand()%255 );
	glVertex2f(c[0],c[1]);
	for (int i=0;i<dziel;i++){glColor3ub( rand()%255, rand()%255, rand()%255 );
glVertex2f(c[0],c[1]+((1.5/dziel)*i)); }

		glColor3ub( rand()%255, rand()%255, rand()%255 );
	glVertex2f(d[0],d[1]);
	for (int i=0;i<dziel;i++){glColor3ub( rand()%255, rand()%255, rand()%255 );
glVertex2f(d[0]-((1.5/dziel)*i),d[1]); }

	glEnd();

	czworo(a,b,c,d,powtu,peru);
	glFlush();
}
void MyInit(void)
{
	glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
}


void main(void)
{
	std::cout<<"Liczba powtorzen: ";
	std::cin>>powtu;
	std::cout<<"Stopien perturbacji: ";
	std::cin>>peru;
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
	glutCreateWindow("Dywan Sierpińskiego");
	glutDisplayFunc(RenderScene);
	glutReshapeFunc(ChangeSize);
	MyInit();
	glutMainLoop();


}
```