POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Wprowadzenie do grafiki komputerowej

Kurs: INE4234L

Sprawozdanie z ćwiczenia nr 2

Rysowanie obiektów fraktalnych na płaszczyźnie

Wykonał:	**Kamil Kamyszek**
Termin:	**PT/NP 11.00-14.00**
Data wykonania ćwiczenia:	**12.10.2018r**
Data oddania sprawozdania:	**26.10.2018r.**
Ocena:	

Uwagi prowadzącego:

1 Wstęp

Na drugie laboratorium z grafiki mieliśmy za zadanie skonfigurować środowisko do pracy (zainstalować odpowiednie biblioteki do Visual Studio) i wykonać kilka prostych programów.

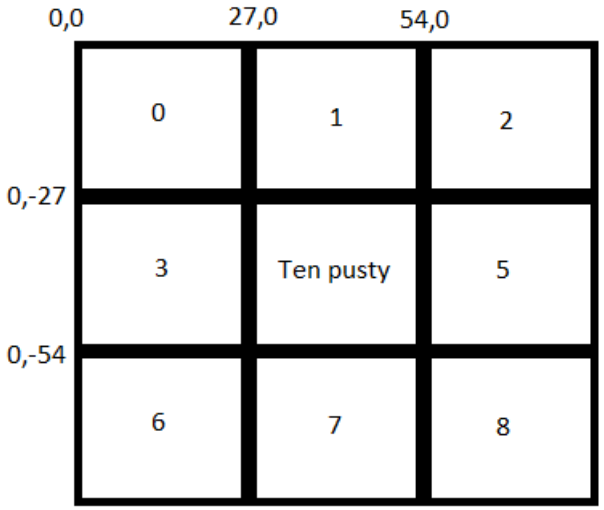
2 Przebieg Laboratorium

Pracę rozpoczęliśmy od wygenerowania okienka, a na nim narysowanego kwadratu. Następnie do wykonania był dwukolorowy trójkąt. Ostatnim zadaniem przed samodzielnym poleceniem do wykonania było pokolorowanie trójkąta trzema kolorami tak aby powstał gradient wychodzący od rogów figury. Po wprowadzeniu dostaliśmy do wykonania Dywan Sierpińskiego.

3 Zadanie do samodzielnego wykonania

Do wykonania otrzymaliśmy dywan Sierpińskiego. Jest to fraktal zbudowany z kwadratu 3x3 w którym usuwamy środkowy kwadrat i rekurencyjnie wykonujemy to samo dla pozostałych 8 kwadratów. Poniżej kod trzech funkcji odpowiedzialnych za wygenerowanie struktury:

```
void bigestSquare(float x, float y) // Funkcja odpowiedzialna za „główny” duży kwadrat 3x3
{
	int a = 0;
	for (int j = 0; j < 3; j++) //pętla do ustawienia współrzędnych kwadratów pionowo
	{

		for (int i = 0; i < 3; i++) //ustawienie współrzędnych kwadratów poziomo
		{
			if (a != 4)//ominięcie środkowego kwadratu żeby był pusty
			{
				middleSquare(x, y);//Funkcja do ustalenia koordynatów dla
						//średnich kwadratów
				a++;
			}
			else a++;

			x = x + 9 * squareSize;//Przesuniecie x do miejsca kolejnego kwadratu
		}
		x = x - 27 * squareSize;  //Powrot X do początku
		y = y - 9 * squareSize;  //Zwiekszenie Y w celu rysowania kolejnych kwadratów
	}
}
```

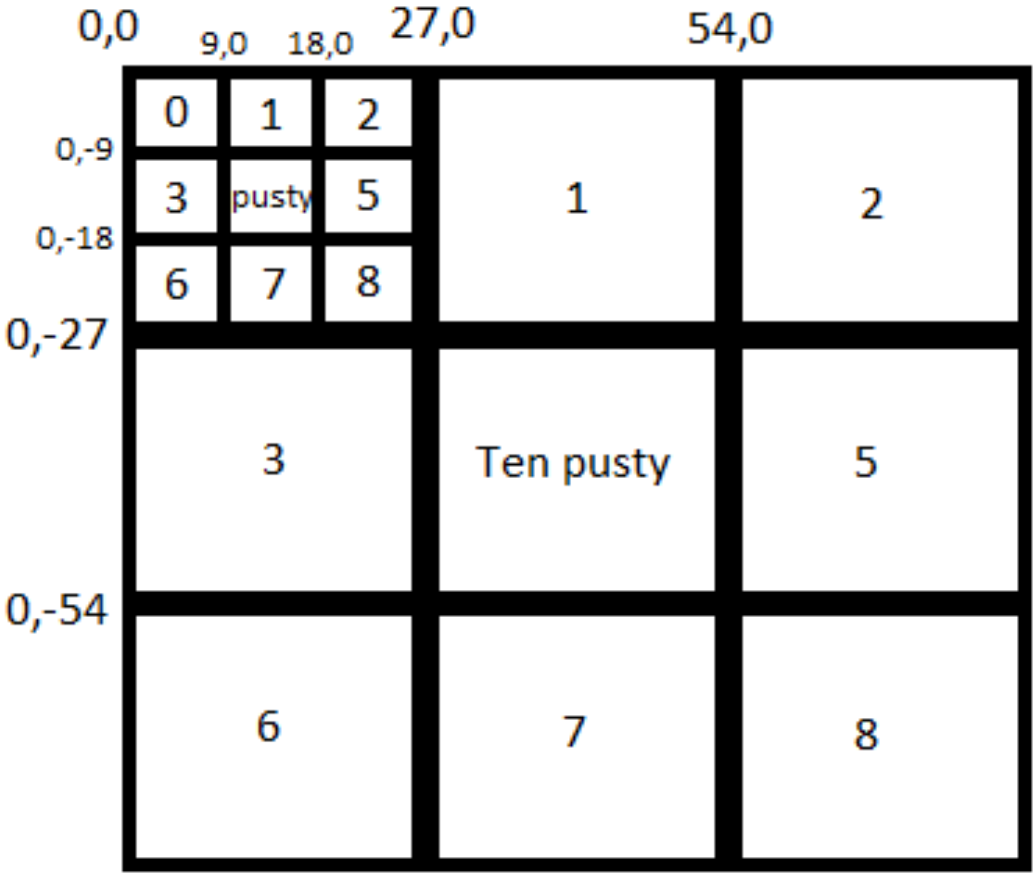


Zobrazowanie działania funkcji biggestSquare o rozmiarze kwadratu = 3

```
void middleSquare(float x, float y) //Funkcja ustawiająca koordynaty średniego
					//kwadratu
```

```
{
	int a = 0;
	for (int j = 0; j < 3; j++)
	{

		for (int i = 0; i < 3; i++)
		{
			if (a != 4)
			{
				smallestSquare(x, y);
				a++;
			}
			else a++;

			x = x + 3 * squareSize;
		}
		x = x - 9 * squareSize;
		y = y - 3 * squareSize;
	}
}
```

Powyższa funkcja działa analogicznie do funkcji *biggestSquare*

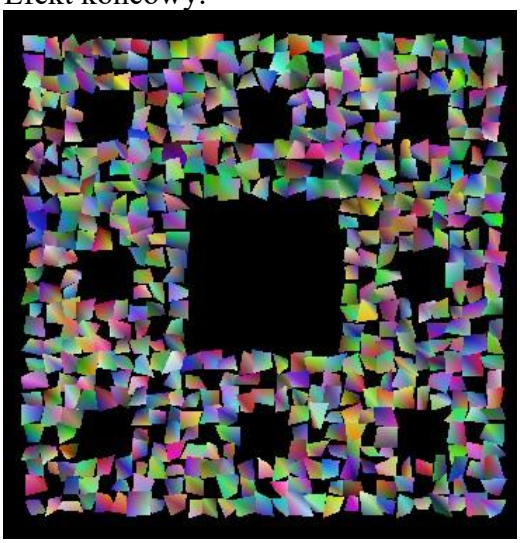


Zobrazowanie działania funkcji middleSquare dla kwadratu o rozmiarze = 3

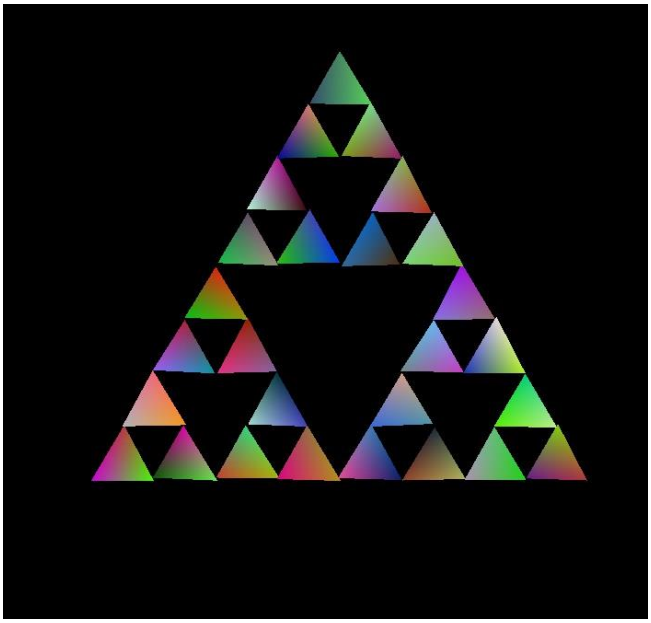
```
void smallestSquare(float x, float y) //Funkcja rysująca kwadraty
{
```

```
        float _x = x - 3 * squareSize;
        float _y = y + 3 * squareSize;
        int a = 0;


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

                for (int i = 0; i < 3; i++)
                {
                        if (a != 4)
                        {

glBegin(GL_POLYGON);
glColor3f(squareColor(), squareColor(), squareColor()); //Losuje kolor pierwszego rogu
glVertex2f(_x+ perturbation(), _y+ perturbation());
glColor3f(squareColor(), squareColor(), squareColor()); // losuje kolor drugiego rogu
glVertex2f(_x+ perturbation() +  squareSize, _y+ perturbation());
glColor3f(squareColor(), squareColor(), squareColor()); // losuje kolor trzeciego rogu
glVertex2f(_x+ perturbation() +  squareSize, _y+ perturbation() -  squareSize);
glColor3f(squareColor(), squareColor(), squareColor()); // losuje kolor czwartek rogu
glVertex2f(_x+ perturbation(), _y+ perturbation() - squareSize);

                                glEnd();
                                glFlush();
                                a++;
                        }
                        else a++;

                        _x = _x + 1 * squareSize;
                }
                _x = _x - 3 * squareSize;
                _y = _y - 1 * squareSize;
        }
}
```

Funkcja *smallestSquare* działa analogicznie do pozostałych dwóch z tą różnicą, że ona rysuje czterokolorowy kwadrat i dokonuje perturbacji na nim (przesuwa o mały x i y względem poprawnych koordynatów i sprawia że kwadrat jest postrzępiony).
Efekt końcowy:



Podczas zajęć podjąłem się również wykonania trójkąta Sierpińskiego. Z powodu braku czasu dokończyłem zadanie w domu. Podczas wykonywania tego ćwiczenia zmodyfikowałem kod z poprzedniego zadania do uzyskania pożądanych kształtów. Musiałem pamiętać, aby zwracać uwagę na wysokość trójkąta równobocznego i nie pomylić się podczas zapisywania jej w kodzie. Oto efekt:



Trójkąt tak samo jak kwadrat są pokolorowanie gradientem. Kolory są generowane losowo za pomocą funkcji rand():

```
float triangleColor(void)
{
	float x = rand() % 100;
	return x/100;
}
```

Dokonana na nim została również perturbacja analogiczna do tej przy kwadracie:

```
float perturbation(void)
{
	float x = rand() % 100;
	return x / 100;
}
```

4 Wnioski

Dzięki zamieszczonej instrukcji na stronie www, wykonanie zadań na zajęcia nie sprawiło większych trudności i pozwoliło oswoić się ze środowiskiem pracy jak również biblioteką OpenGL i Glut. Niestety wykonane powyżej figury Sierpińskiego nie mają większego zastosowania poza ładnie wyglądającymi wygaszaczami ekranów lub ćwiczeniami z wykorzystaniem koordynatów na siatce współrzędnych. Jednak dzięki ich wykonaniu zapoznałem się z programowaniem prostych figur geometrycznych.