**POLITECHNIKA WROCŁAWSKA**
**Instytut Informatyki, Automatyki i Robotyki**
**Zakład Systemów Komputerowych**

**Grafika komputerowa i komunikacja człowiek - komputer**

**Kurs: INEK00012L**

**Sprawozdanie z ćwiczenia nr 2**

# TEMAT ĆWICZENIA: Podstawy OpenGL

| | |
|---|---|
| **Wykonał:** | **Bartosz Szymczak, nr indeksu 252734** |
| **Termin:** | **Poniedziałek TN 7:30-10:30** |
| **Data wykonania ćwiczenia:** | **18.10.2021r.** |
| **Data oddania sprawozdania:** | |
| **Ocena:** | |

| |
|---|
| **Uwagi prowadzącego:** |

# Spis treści



## 1. Cel ćwiczenia

Celem ćwiczenia było poznanie podstawowych możliwości biblioteki graficznej OpenGL wraz z rozszerzeniem GL Utility Toolkit. Ćwiczenie obejmowało inicjalizację i zamykanie trybu OpenGL oraz rysowanie tworów pierwotnych w przestrzeni 2D. Zwieńczeniem powyższego procesu było stworzenie Dywanu Sierpińskiego.

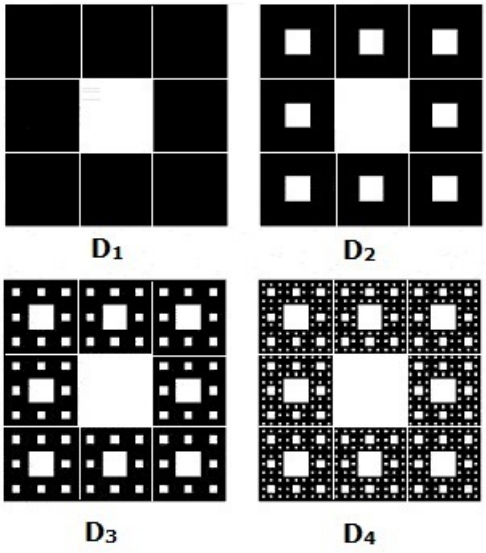
## 2. Podstawy teoretyczne

### 2.1 OpenGL oraz GLUT

OpenGL jest niezależną od platformy sprzętowej i systemowej biblioteką graficzną zoptymalizowaną pod tworzenie grafiki 3D. Składa się z 250 podstawowych wywołań, umożliwiających budowanie złożonych trójwymiarowych scen z podstawowych figur geometrycznych. Wykorzystywany jest często przez gry komputerowe i wygaszacze ekranu. Konkuruje z nim głównie DirectX. Mimo że nie jest przenośny pomiędzy środowiskami i oferuje trochę gorszą jakość grafiki, to jest od niego nieco szybszy. Do wykorzystania OpenGL w aplikacji potrzebne są zwykle również inne biblioteki. Jest tak, ponieważ OpenGL zajmuje się głównie renderem grafiki i nie zawiera funkcji związanych z tworzeniem kontekstu graficznego, obsługą środowiska okienkowego, czy obsługą zdarzeń takich jak naciśnięcie klawisza na klawiaturze lub ruch kursora myszy. Rozszerzenie GLUT wykonuje głównie operacje wejścia/wyjścia na poziomie systemu operacyjnego. Dodatkowo umożliwia m.in. zarządzanie oknami oraz monitorowanie klawiatury i myszy.

### 2.2 Dywan Sierpińskiego

Dywan Sierpińskiego to fraktal, stworzony przez Wacława Sierpińskiego. Polega on na stworzeniu kwadratu, podzieleniu go na 9 mniejszych kwadratów, usunięcia środkowego kwadratu i ponownego rekurencyjnego użycia tej samej metody do każdego powstałego kwadratu. Z założenia, gdy ilość wykonywania rekurencji nad danym kwadratem dąży do nieskończoności, fraktal posiada pole powierzchni równe zero.




Rysunek 1 – Pierwsze 4 rekurencje dywanu Sierpińskiego

### 2.3 Dywan Sierpińskiego na ekranie monitora

Mając na uwadze ograniczenia sprzętowe w postaci dyskretnej płaszczyzny, będącej monitorem stworzonym z siatki pikseli, fraktal powinien „zniknąć" przy odpowiednich założeniach. Do stworzenia kwadratu potrzebna przynajmniej 9 pikseli. Z tego powodu długość boku będzie wielokrotnością liczby 3 (3 x 3 = 9). Liczba rekurencji, potrzebnej do zaprzestania rysowania dywanu na monitorze, będzie o jeden większa niż potęgą liczby, do której należy podnieść liczbę 3, w celu otrzymania długości pierwotnego kwadratu. Na przykład pierwotny kwadrat o długości 81 przestanie być widoczny z 5 rekurencją.

## 3. Opis programu

Do stworzenia programu wykorzystano język C++, biblioteki OpenGL i GLUT oraz środowisko Visual Studio Community 2019. Szkielet programu bazuje na przykładowych programach opisanych w instrukcji do ćwiczenia.

### 3.1 Biblioteki oraz zmienne globalne

```
7   #include <windows.h>
8   #include <gl/gl.h>
9   #include <gl/glut.h>
10  #include <iostream>
11
12  //Zmienne globalne odpowiedzialne za poziom rekurencji, stopień perturbacji,
13  //zawartość koloru oraz współrzędne
14  int level;
15  float pet;
16  bool col;
17  typedef float point2[2];
```

## 3.2 Poszczególne funkcje

```
19  //Metoda generująca losowy kolor
20  void randomColor()
21  {
22      float f, f1, f2;
23
24      f = (float)rand() / RAND_MAX;
25      f1 = (float)rand() / RAND_MAX;
26      f2 = (float)rand() / RAND_MAX;
27
28      glColor3f(f, f1, f2);
29  }
```

Funkcja randomColor służy do generowania losowego koloru, poprzez generowanie trzech liczb rzeczywistych z zakresu [0;1] oraz przekazania ich jako argument do funkcji glColor3f() ustawiającej kolor.

```
31  //Funkcja rysująca kwadrat
32  void drawSquare(float x, float y, float width)
33  {
34      //Perturbacja współrzędnych kwadratu
35      float p1;
36      //Losowanie perturbacji wspolrzednej z przedzialu [-pet;pet]
37      p1 = -pet + (float)rand() / RAND_MAX * 2 * pet;
38
39      //obliczanie narożników kwadratów z uwzględnieniem perturbacji
40      point2 topL = { x - width / 2 - p1, y + width / 2 + p1 };
41      point2 topR = { x + width / 2 + p1, y + width / 2 + p1 };
42      point2 botL = { x - width / 2 - p1, y - width / 2 - p1 };
43      point2 botR = { x + width / 2 + p1, y - width / 2 - p1 };
44
45      //Rysowanie kolorowych figur
46      if (col == true)
47      {
48          glBegin(GL_POLYGON);
49          //Losowanie nowego koloru
50          randomColor();
51          glVertex2fv(topL);
52          randomColor();
53          glVertex2fv(topR);
54          randomColor();
55          glVertex2fv(botR);
56          randomColor();
57          glVertex2fv(botL);
58          glEnd();
59      }
60      //Rysowanie jednokolorowych figur
61      else
62      {
63          glBegin(GL_POLYGON);
64          glColor3f(1.0f, 1.0f, 1.0f);
65          glVertex2fv(topL);
66          glVertex2fv(topR);
67          glVertex2fv(botR);
68          glVertex2fv(botL);
69          glEnd();
70      }
71  }
```

Funkcja drawSquare służy do rysowania poszczególnych elementów dywanu Sierpińskiego z wykorzystaniem funkcji randomColor. Jej atrybuty to odpowiednio współrzędne środka kwadratu początkowego, długość boku oraz poziom rekurencji. Początkowo następuje losowanie perturbacji z podanego przez użytkownika przedziału. Dalej następuje przypisanie danych wartości do odpowiednich zmiennych wierzchołków z uwzględnieniem stworzonych perturbacji. Następnie w zależności, czy użytkownik zdecydował się na stworzenie obrazu kolorowego czy jednokolorowego, następuje narysowanie pojedynczej figury w odpowiednim miejscu.

```
72  //Funkcja tworzaca dywan Sierpinskiego
73  void drawCarpet(float x, float y, int width, int level)
74  {
75      //Jeżeli jesteśmy na poziomie wyższym niż 0 należy podzielić kwadrat na kolejne mniejsze części
76      if (level > 0)
77      {
78          //Przy rekurencyjnym podziale pomijamy środkowy kwadrat ponieważ jego nie rysujemy
79          drawCarpet(x - width / 3, y + width / 3, width / 3, level - 1);
80          drawCarpet(x, y + width / 3, width / 3, level - 1);
81          drawCarpet(x + width / 3, y + width / 3, width / 3, level - 1);
82          drawCarpet(x - width / 3, y, width / 3, level - 1);
83          drawCarpet(x + width / 3, y, width / 3, level - 1);
84          drawCarpet(x - width / 3, y - width / 3, width / 3, level - 1);
85          drawCarpet(x, y - width / 3, width / 3, level - 1);
86          drawCarpet(x + width / 3, y - width / 3, width / 3, level - 1);
87      }
88
89      //Funkcja rysująca kwadrat
90      else drawSquare(x, y, width);
91  }
```

Funkcja drawCarpet tworzy dywan Sierpińskiego z wykorzystaniem funkcji drawSquare. Argumentami są odpowiednio współrzędne środka początkowego kwadratu, długość boku i poziom rekurencji. Początkowo sprawdzany jest poziom aktualnej rekurencji, w celu kontynuacji musi wynosić więcej niż 0. Jeśli warunek jest spełniony, to funkcja wykonuje się rekurencyjnie dla 8 przeskalowanych współrzędnych (poza środkowym kwadratem), o długości 3 razy mniejszej i poziomie rekurencji o jeden mniejszym. W sytuacji, gdy poziom rekurencji dojdzie do najgłębszego, następuje narysowanie figury za pomocą funkcji drawSquare o przekazanych parametrach.

Funkcja drawCarpet wywoływana jest w funkcji RenderScene, gdzie przekazywane są jej domyślnie: współrzędne początku osi (0,0), długość boku równą 81 oraz poziom rekurencji wprowadzoną przez użytkownika. W funkcji main znajduje się wywołanie funkcji menu, która jest odpowiedzialna za komunikację z użytkownikiem.

### 3.3 Działanie programu

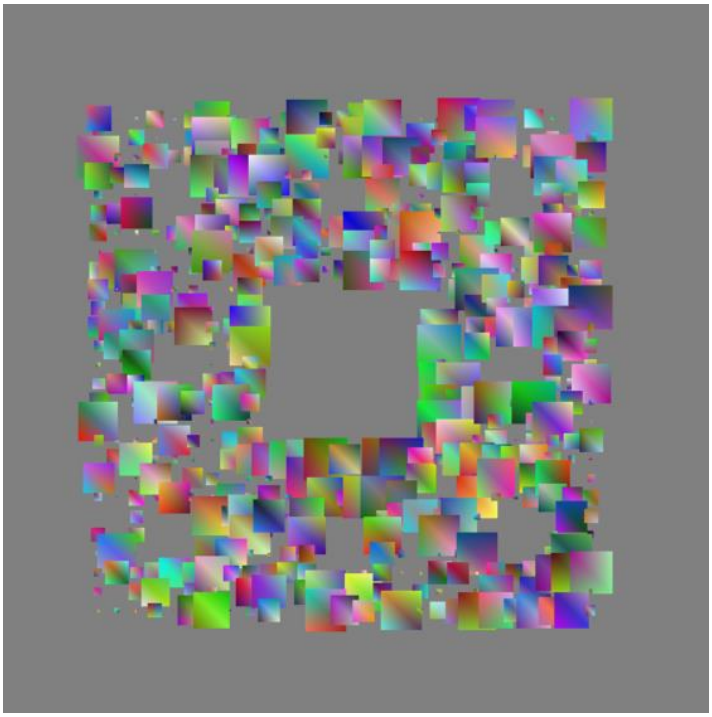
Program po uruchomieniu wyświetla w konsoli prośby o podanie wartości odpowiednich argumentów, następnie uruchamia w osobnym okienku żądaną formę dywanu Sierpińskiego.

```
Podaj poziom dywanu:
3
Podaj perturbacje:
2
Czy obraz ma byc pokolorowany? ( 0 - brak koloru / 1 - kolor )
1
```

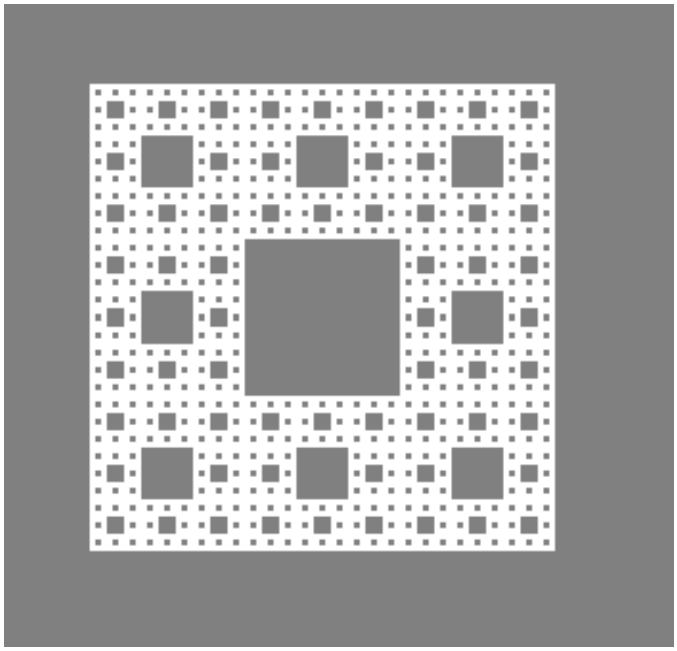
Rysunek 2 – Widok konsoli z przykładowymi wartościami

# 4. Wyniki

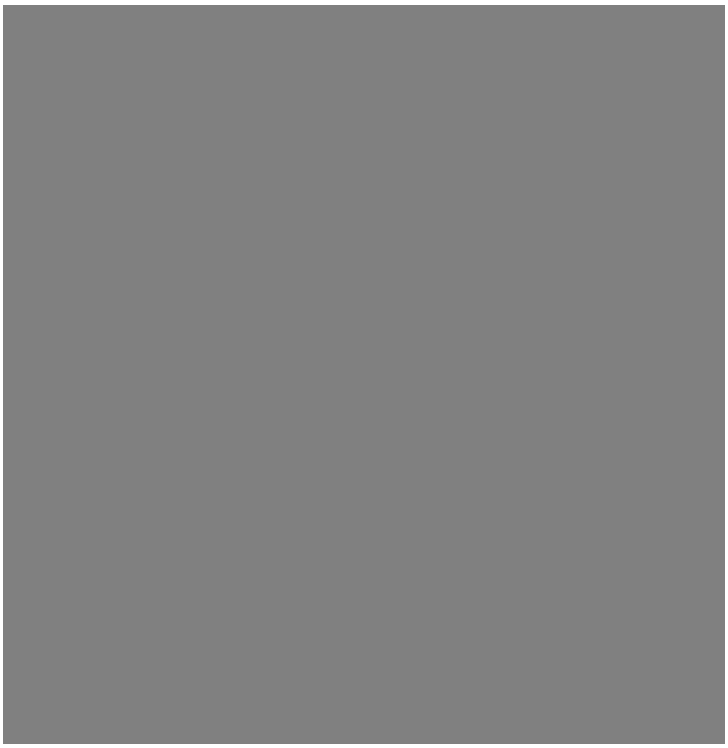
Wyniki działania programu dla poszczególnych wartości argumentów:



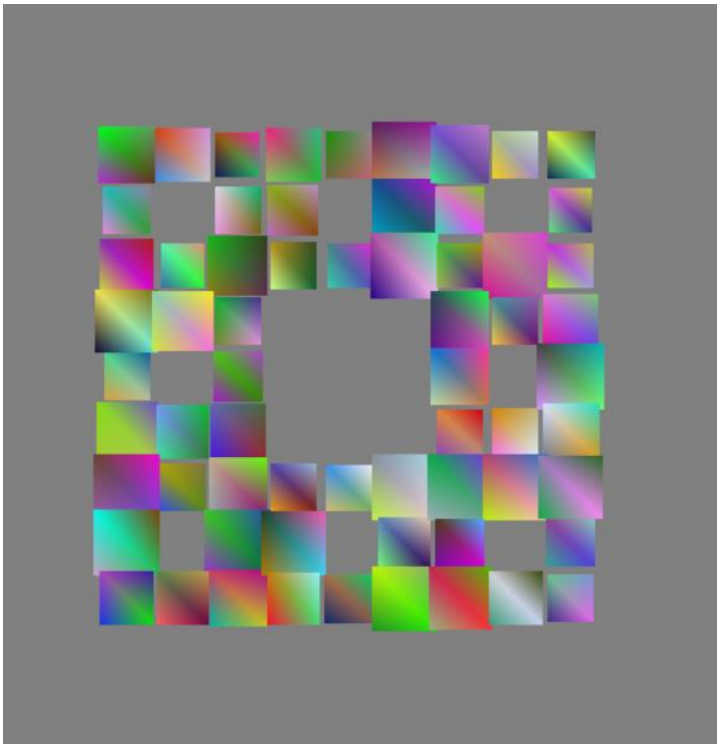
Rysunek 3 – Wynik działania programu dla poziomu rekurencji równego 3, perturbacji równej 2 oraz żądania kolorowego obrazu



Rysunek 4 – Wynik działania programu dla poziomu rekurencji równego 4, perturbacji równej 0 oraz żądania niekolorowego obrazu



Rysunek 5 – Wynik działania programu dla poziomu rekurencji równego 5, perturbacji równej 0 oraz żądania niekolorowego obrazu



Rysunek 6 – Wynik działania programu dla poziomu rekurencji równego 2, perturbacji równej 1 oraz żądania kolorowego obrazu

## 5. Wnioski

Po zapoznaniu się z instrukcją do ćwiczenia byłem w stanie zrozumieć jak wygląda podstawowy schemat programu rysującego proste figury na ekranie. Dzięki temu, wiedzy z zakresu podstaw programowania oraz wiedzy zdobytej na laboratorium i wykładzie z realizowanego kursu, byłem w stanie zrealizować ćwiczenie i uzyskać oczekiwane wyniki.

## 6. Źródła

http://www.zsk.ict.pwr.wroc.pl/zsk/dyd/intinz/gk/lab/cw_2_dz/

https://pl.wikipedia.org/wiki/OpenGL

https://pl.wikipedia.org/wiki/Dywan_Sierpińskiego

https://www.matematyczny-swiat.pl/2013/05/dywan-sierpinskiego.html