Grafika komputerowa i komunikacja człowiek komputer

Laboratorium 1

Podstawy OpenGL, grafika 2D

Wykonał:

Ivan Hancharyk 264511

Termin zajęć:

WT/TP/8:00

1. Cel ćwiczenia

Celem ćwiczenia było zaprogramowanie fraktala **Dywan Sierpińskiego** z wykorzystaniem biblioteki **OpenGL** z rozszerzeniem **GLUT**. Ćwiczenie miało na celu naukę podstaw programowania grafiki komputerowej, w tym interakcji użytkownika z aplikacją oraz generowania złożonych struktur graficznych na podstawie algorytmu fraktalnego.

2. Wstęp teoretyczny

RGB i inne systemy reprezentacji kolorów

Model **RGB (Red, Green, Blue)** to jedna z podstawowych metod reprezentowania kolorów, używana w grafice komputerowej. Każdy kolor jest definiowany przez kombinację trzech składowych: czerwonej, zielonej i niebieskiej. Intensywność każdej składowej jest wyrażana w przedziale od 0 do 255 lub w formie ułamka (0 do 1), co pozwala na uzyskanie milionów kombinacji kolorów.

Inne modele:

1. **CMYK (Cyan, Magenta, Yellow, Black)**: Jest to model subtraktywny, używany głównie w druku. Zamiast emitować światło jak w RGB, barwy są uzyskiwane poprzez pochłanianie części spektrum widzialnego przez odpowiednie pigmenty.
2. **HSV (Hue, Saturation, Value)**: Model HSV lepiej odzwierciedla sposób, w jaki ludzie postrzegają kolory. Składowe to: odcień (barwa), nasycenie (czystość koloru) i jasność (wartość koloru).

Dywan Sierpińskiego

Dywan Sierpińskiego to fraktal, który można wygenerować poprzez rekurencyjne dzielenie kwadratu na mniejsze części. Z każdego podziału usuwa się centralny kwadrat, a operacja jest powtarzana na pozostałych kwadratach. Proces ten teoretycznie może trwać w nieskończoność, generując coraz bardziej skomplikowany i szczegółowy wzór, przy czym całkowita powierzchnia fraktala zmierza do zera. Nasz program jednak nie działa w nieskończoność, ponieważ wprowadzone jest ograniczenie na liczbę poziomów rekurencji.

Ograniczenie to jest konieczne, aby uniknąć problemów związanych z zasobami obliczeniowymi i pamięcią, ponieważ generowanie fraktala w nieskończoność jest niemożliwe ze względu na skończone zasoby komputerowe.

Właściwości kolorów przypisane do wierzchołków/figur

W grafice komputerowej kolory mogą być przypisane zarówno do całych figur, jak i pojedynczych wierzchołków. W naszym programie kolory są przypisywane losowo do wierzchołków każdej figury, co pozwala na dynamiczne mieszanie barw. Dzięki temu każdy fragment fraktala może mieć różne kolory na krawędziach, co tworzy wizualnie interesujący efekt, tak zwany gradient, (kolory są reprezentowane w modelu RGB).

Rekurencja

Rekurencja to technika programistyczna, w której funkcja wywołuje samą siebie do momentu spełnienia określonego warunku. W przypadku dywanu Sierpińskiego, rekurencja jest używana do podziału kwadratu na mniejsze części. Każdy z mniejszych kwadratów jest dalej dzielony, aż do osiągnięcia maksymalnej głębokości rekurencji, określonej przez użytkownika (np. u mnie to 5 poziomów).

Problem pokazywania nieskończonych obliczeń

Ponieważ fraktale, takie jak dywan Sierpińskiego, teoretycznie mają nieskończoną liczbę poziomów, w praktyce musimy wprowadzić ograniczenia. Program realizuje to poprzez ograniczenie głębokości rekurencji do maksymalnej liczby poziomów, którą definiuje użytkownik. Każdy poziom dodaje coraz więcej szczegółów, ale z powodu ograniczonych zasobów obliczeniowych, możliwe jest generowanie tylko ograniczonej liczby iteracji.

Renderowanie

Renderowanie - proces generowania obrazu na podstawie danych 2D lub 3D. W kontekście OpenGL, renderowanie oznacza wyświetlanie obiektów na ekranie poprzez wywołanie odpowiednich funkcji graficznych, takich jak rysowanie wierzchołków i wielokątów. Wynikowy obraz powstaje poprzez przekształcenie współrzędnych obiektów w scenie na piksele.

3. Realizacja zadania

Fraktal jest rysowany na podstawie parametrów wprowadzonych przez użytkownika, takich jak liczba iteracji (powtórzeń algorytmu), stopień deformacji oraz wybór trybu kolorystycznego. Poniżej omówiono główne elementy programu odpowiedzialne za tworzenie dywanu.

Wprowadzenie danych

Na początku program pobiera dane od użytkownika, które kontrolują proces generowania fraktala. Dane te obejmują liczbę powtórzeń (stopni rekurencji), stopień deformacji wierzchołków oraz wybór trybu kolorystycznego (kolorowy lub czarno-biały).

Rekurencyjne rysowanie Dywanu Sierpińskiego

Najważniejszą częścią programu jest rekurencyjna funkcja **rysuj(),** która dzieli każdy kwadrat na mniejsze części, tworząc strukturę fraktalną. Dywan Sierpińskiego powstaje poprzez powtarzanie schematu dzielenia kwadratu na 9 części i pomijanie środkowego kwadratu.

```
void rysuj(double x, double y, double width, int stopien)
{
    if (stopien == 0)
    {
        float deformation_x = 0.f;
        float deformation_y = 0.f;

        glBegin(GL_POLYGON);

        // W trybie kolorowym przypisujemy różne kolory do wierzchołków
        if (kolorowy) {
            for (int i = 0; i < 4; ++i) {
                int indeks = rand() % 15;  // Losowy indeks z rozszerzonej
palety kolorów
                glColor3f(paleta_kolorow[indeks][0],
paleta_kolorow[indeks][1], paleta_kolorow[indeks][2]);

                // Losowo generowane wierzchołki z deformacją
                if (i == 0) glVertex2f(x + ((float)rand() / RAND_MAX - 0.5f)
* width * def_level, y + width + ((float)rand() / RAND_MAX - 0.5f) * width *
def_level);
                if (i == 1) glVertex2f(x + width + ((float)rand() / RAND_MAX
- 0.5f) * width * def_level, y + width + ((float)rand() / RAND_MAX - 0.5f) *
width * def_level);
                if (i == 2) glVertex2f(x + width + ((float)rand() / RAND_MAX
- 0.5f) * width * def_level, y + ((float)rand() / RAND_MAX - 0.5f) * width *
def_level);
                if (i == 3) glVertex2f(x + ((float)rand() / RAND_MAX - 0.5f)
* width * def_level, y + ((float)rand() / RAND_MAX - 0.5f) * width *
def_level);
            }
        } else {
            // Tryb czarno-biały, losowy wybór między czarnym a białym dla
całej figury
            int indeks = rand() % 2;
            glColor3f(paleta_czarno_biala[indeks][0],
paleta_czarno_biala[indeks][1], paleta_czarno_biala[indeks][2]);

            // Rysowanie figury z losowymi deformacjami
            deformation_x = ((float)rand() / RAND_MAX - 0.5f) * width *
def_level;
            deformation_y = ((float)rand() / RAND_MAX - 0.5f) * width *
def_level;
            glVertex2f(x + deformation_x, y + width + deformation_y);
```

```
            deformation_x = ((float)rand() / RAND_MAX - 0.5f) * width *
def_level;
            deformation_y = ((float)rand() / RAND_MAX - 0.5f) * width *
def_level;
            glVertex2f(x + width + deformation_x, y + width + deformation_y);

            deformation_x = ((float)rand() / RAND_MAX - 0.5f) * width *
def_level;
            deformation_y = ((float)rand() / RAND_MAX - 0.5f) * width *
def_level;
            glVertex2f(x + width + deformation_x, y + deformation_y);

            deformation_x = ((float)rand() / RAND_MAX - 0.5f) * width *
def_level;
            deformation_y = ((float)rand() / RAND_MAX - 0.5f) * width *
def_level;
            glVertex2f(x + deformation_x, y + deformation_y);
        }

        glEnd();
    }
    else
    {
        width = width / 3.0f;
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++)
                if (i != 1 || j != 1)
                {
                    rysuj(x + i * width, y + j * width, width, stopien - 1);
                }
        }
    }
}
```

- **Rekurencja**: Funkcja wywołuje samą siebie dla każdej części kwadratu, aż do osiągnięcia zerowego stopnia rekurencji, co oznacza narysowanie najmniejszego kwadratu.
- **Deformacja**: Parametr deformacji pozwala na losowe zniekształcenie wierzchołków kwadratów, co nadaje nieregularny wygląd fraktalowi.
- **Kolorowanie**: Program obsługuje dwa tryby kolorystyczne – kolorowy oraz czarno-biały.

Sterowanie rysowaniem

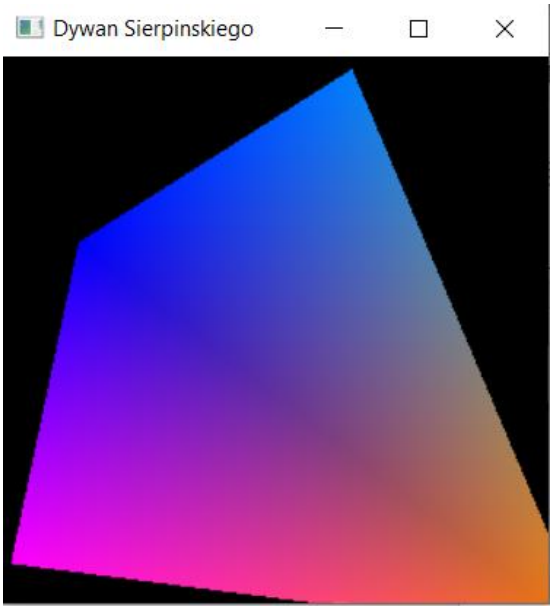
Użytkownik steruje rysowaniem fraktala za pomocą klawiszy. Funkcja **`RenderScene()`** odpowiada za stopniowe rysowanie kolejnych poziomów fraktala. Użytkownik może naciskać klawisz `F`, aby przechodzić do kolejnych kroków, a także klawisz `R`, aby zresetować rysowanie.

```
void RenderScene()
{
    if (current_step <= stopien)
    {
        glClear(GL_COLOR_BUFFER_BIT);
        rysuj(x - width / 2., y - width / 2., width, current_step);
        glFlush();
        current_step++; // Zwiększamy krok po każdym rysowaniu
```

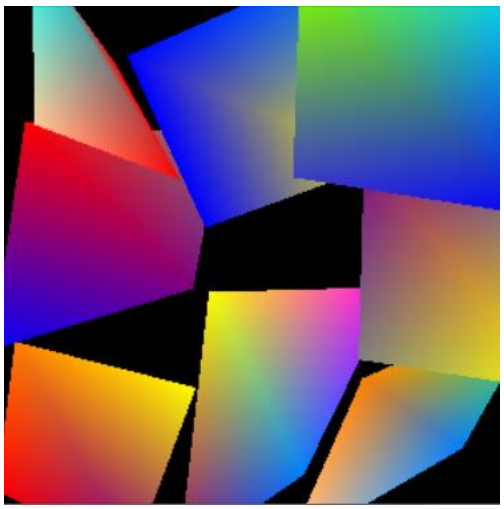
```
    }
    else
    {
        glBegin(GL_QUADS);
        glColor3f(0.0f, 0.0f, 0.0f);  // Kolor czarny

        glVertex2f(-150.0f, 150.0f);    // Lewy górny róg
        glVertex2f(150.0f, 150.0f);     // Prawy górny róg
        glVertex2f(150.0f, -150.0f);    // Prawy dolny róg
        glVertex2f(-150.0f, -150.0f);   // Lewy dolny róg

        glEnd();
        glFlush();
    }
}
```

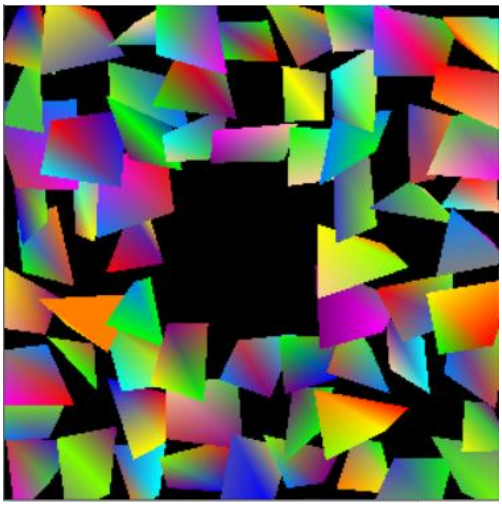
Początkowy wygląd fraktalu przy kolejnych danych:

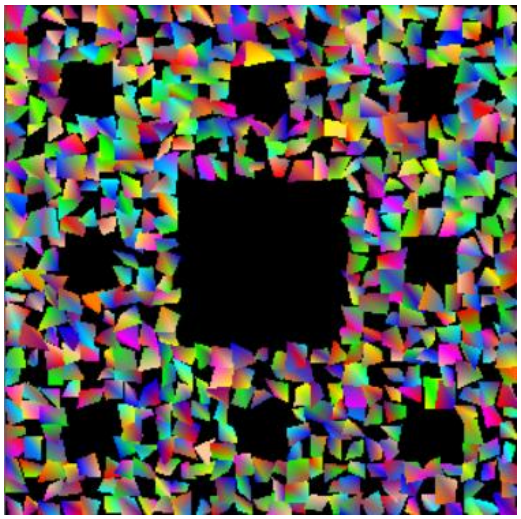
Podaj liczbe powtorzen algorytmu (1-5):5
Podaj stopien deformacji (0-1):0.75
Wybierz tryb: 1 - Kolorowy, 2 - Czarno-bialy:1

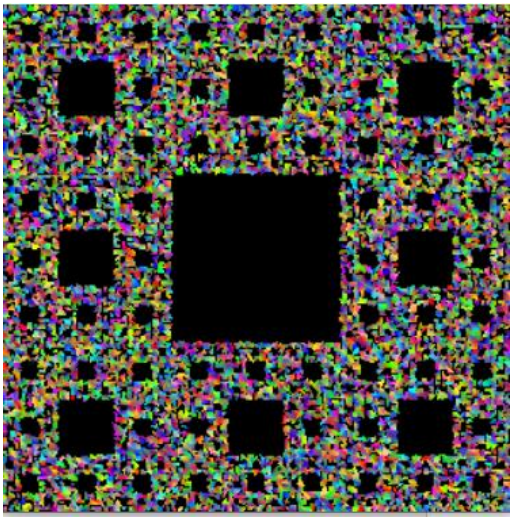


Kolejne poziomy rekurencji:

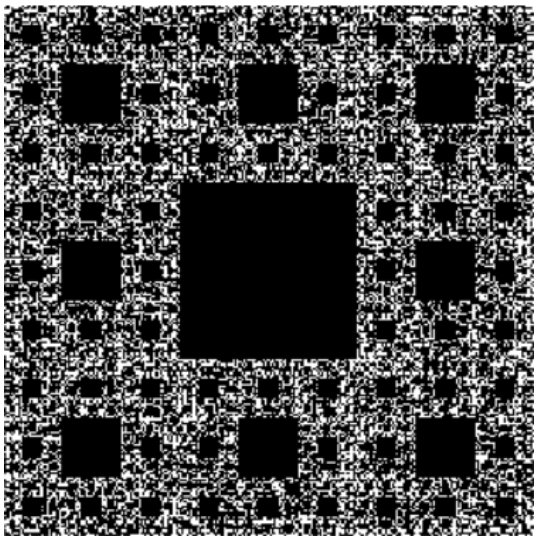








Fraktal w trybie czarno-białym przy kolejnych danych:
Podaj liczbe powtorzen algorytmu (1-5):5
Podaj stopien deformacji (0-1):0
Wybierz tryb: 1 - Kolorowy, 2 - Czarno-bialy:2



Wnioski

W ramach tego zadania udało się zrealizować generowanie Dywanu Sierpińskiego, wykorzystując rekurencję oraz interaktywne sterowanie procesem rysowania przez użytkownika. Program umożliwia dynamiczny wybór parametrów, takich jak liczba iteracji, deformacja oraz tryb kolorystyczny, co pozwala na uzyskanie różnych wariantów fraktala. Dzięki temu zadaniu nauczyłem się, jak efektywnie stosować rekurencję w grafice komputerowej oraz jak sprawnie obsługiwać dane wejściowe użytkownika, zapewniając poprawność i elastyczność działania programu. Dodatkowo zdobyłem praktyczne doświadczenie w pracy z OpenGL i FreeGLUT, co rozszerzyło moje umiejętności w zakresie renderowania i cieniowania w aplikacjach graficznych.