Grafika komputerowa

Sprawozdanie nr 1

Prowadzący:

Dr inż. Tomasz Kapłon

Autor:

Jan Niewiarowski, 210828

Wydział Elektroniki

III Rok

Pt 8:00 – 11:00 TN

25 października 2018

1. Cel laboratorium

Celem ćwiczenia wykonywanego podczas trwania laboratorium było zapoznanie się elementarnymi możliwościami biblioteki graficznej OpenGL wraz z rozszerzeniem GL Utility Toolkit (GLUT).

Niniejsze ćwiczenie obejmowało inicjalizację i zamykanie trybu OpenGL oraz rysowanie tworów pierwotnych w przestrzeni 2D.

2. Dywan Sierpińskiego

Dywanem Sierpińskiego nazywamy fraktal otrzymany z kwadratu. Kwadrat dzieli się na dziewięć mniejszych o takich samych wymiarach, a następnie usuwa się środkowy kwadrat i ponownie rekurencyjnie stosujemy tę procedurę, na pozostałych ośmiu kwadratach. Nazwa fraktalu pochodzi od nazwiska Wacława Sierpińskiego.

3. Algorytm rekurencyjny

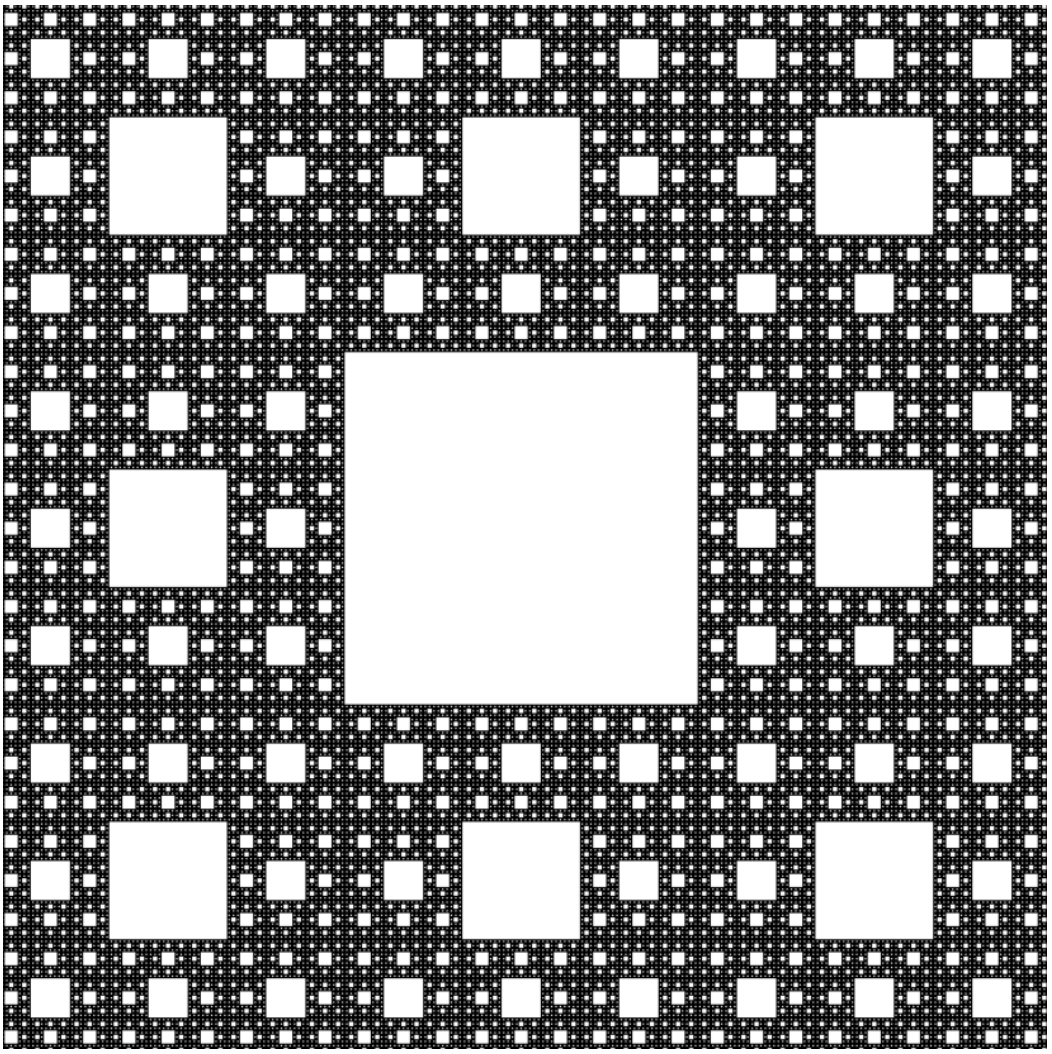
Program rysuje reprezentację dywanu Sierpińskiego o zadanym przez nas rozmiarze, stopniu rekurencji oraz poziomie zniekształcenia. Algorytm przez nas stosowany rekurencyjnie wylicza położenie kolejnych kwadratów, aż do uzyskania zadanego poziomu. Co każdy kolejny stopień fraktalu ilość wycinanych kwadratów rosła trzykrotnie, natomiast pole powierzchni trzykrotnie malało.

4. Listing programu

```
#include <gl/gl.h>
#include <gl/glut.h>

// Funkcaja określająca, co ma być rysowane
// (zawsze wywoływana, gdy trzeba przerysować scenę)

typedef GLfloat point2[2];

// funkcja generujaca losowe kolory naszych obiektow
float randomColor()
{
    return ((float)(rand() % 10) + 1) / 10;
}

void dywan(point2 p1, GLfloat bok, int poziom)
{
    int iloscZaglebien = 3;

    if (poziom == iloscZaglebien)
        return;

    for (int i = 1; i < 4; i++)
    {
        for (int j = 1; j < 4; j++)
        {
            if (i % 2 != 0 || j % 2 != 0)
            {
                if (poziom == iloscZaglebien - 1)
                {
                    glBegin(GL_POLYGON);
                    glColor3f(randomColor(), randomColor(), randomColor());
                    glVertex2f(p1[0] + (bok / 3)*(i - 1), p1[1] + (bok / 3)*(j - 1));
                    glColor3f(randomColor(), randomColor(), randomColor());
                    glVertex2f(p1[0] + (bok / 3)*i, p1[1] + (bok / 3)*(j - 1));
                    glColor3f(randomColor(), randomColor(), randomColor());
                    glVertex2f(p1[0] + (bok / 3)*i, p1[1] + (bok / 3)*j);
                    glColor3f(randomColor(), randomColor(), randomColor());
                    glVertex2f(p1[0] + (bok / 3)*(i - 1), p1[1] + (bok / 3)*j);
                    glEnd();
                }

                point2 punkt;
                int kolejnyPoziom = poziom + 1;



                    int kolejnyBok = (bok / 3);
                    punkt[0] = p1[0] + (bok / 3)*(i - 1);
                    punkt[1] = p1[1] + (bok / 3)*(j - 1);
                    dywan(punkt, kolejnyBok, kolejnyPoziom);
                }

            }
        }

    }


    void RenderScene(void)

    {
        point2 p1;
        p1[0] = -100;
        p1[1] = -100;
        GLfloat bok = 200;
        int poziom = 0;
        dywan(p1, bok, poziom);


        glFlush();
        // Przekazanie poleceń rysujących do wykonania

    }


    // Funkcja ustalająca stan renderowania


    void MyInit(void)

    {
        glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
        // Kolor okna wnętrza okna - ustawiono na szary
    }


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

```
    void ChangeSize(GLsizei horizontal, GLsizei vertical)

    // Parametry horizontal i vertical (szerokość i wysokość okna) są
    // przekazywane do funkcji za każdym razem, gdy zmieni się rozmiar okna

    {

        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)

{

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

    glutCreateWindow("Drugi program w OpenGL");
    // 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

}
```

5. Przykładowy rysunek



Rysunek 1 Dywan Sierpińskiego

6. Wnioski

Laboratorium pozwoliło mi na zapoznanie się z rysowaniem obiektów 2D z wykorzystaniem biblioteki OpenGL. Ponadto zdobyłem wiedzę o algorytmach tworzących podstawowe figury i fraktale.