Grafika komputerowa i komunikacja człowiek - komputer

Open GL – podstawy

14.10.2022r.

Maciej Radecki
album: 253257

Prowadzący: dr inż. Jan Nikodem

1. Wprowadzenie teoretyczne

Celem ćwiczenia było zapoznanie się z podstawami OpenGL wraz z rozszerzeniem GL Utility Toolkit (GLUT), powtórzenie wiadomości z zakresu rekurencji oraz przygotowanie animacji przedstawiającej kolejne etapy rysowania dywanu Sierpińskiego.

Dywan Sierpińskiego - jest to fraktal otrzymany z kwadratu za pomocą podzielenia go na dziewięć (3x3) mniejszych kwadratów, usunięcia środkowego kwadratu i ponownego rekurencyjnego zastosowania tej samej procedury do każdego z pozostałych ośmiu kwadratów. W nieskończoności charakteryzuje go nieskończony obwód i zerowe pole powierzchni.

Deformacja – w przedstawionym rozwiązaniu deformacja polega na przesunięciu każdego z wierzchołków o losową liczbę pikseli. Stopień deformacji regulowany jest przez użytkownika poprzez podanie liczby z zakresu <0,1> gdzie 0 oznacza brak deformacji, a 1 bardzo mocną deformację (przesunięcie o połowę rysowanego obiektu).

Rekurencja – odwołanie się funkcji do samej siebie. Mówiąc inaczej, podejście rekurencyjne polega na tym, że rozwiązanie problemu wyraża się za pomocą rozwiązania tego samego problemu dla mniejszych danych wejściowych.

2. Realizacja ćwiczenia

Szkielet programu został przygotowany w oparciu o kod z instrukcji laboratoryjnej. Pierwszą z dodanych funkcji jest procedura *init()*. Służy ona do komunikacji z użytkownikiem oraz do pobrania niezbędnych do działania aplikacji parametrów takich jak liczba powtórzeń algorytmu, długość boku dywany czy stopień deformacji - jako liczbę typu float z przedziału <0,1>.

```
void init()
{
    cout<<"Podaj liczbe powtorzen algorytmu: ";
    cin >> stopien;
    cout <<endl << "Podaj dlugosc boku kwadratu: ";
    cin >> width;
    cout << endl << "Podaj stopien deformacji (0-1): ";
    cin >> def_level;
}
```

Rysunek 1 - funkcja init().

Kolejną zmianą wprowadzoną w kodzie z instrukcji jest modyfikacja funkcji *RenderScene()*. Po ustawieniu koloru tła na czarny następuje wyrenderowanie sceny, a następnie w pętli wywołanie funkcji *rysuj(x - width / 2., y - width / 2., width, i)*. Pierwsze dwa parametry wyznaczają punkt początkowy rysunku, a dokładniej lewy dolny wierzchołek kwadratu, kolejny określa ustaloną przez użytkownika długość boku finalnego dywanu, ostatni, będący jednocześnie interatorem z pętli, określa którego stopnia ma być wygenerowany a później wyświetlony dywan. Wykorzystanie pętli już na tym etapie pozwala przedstawiać z opóźnieniem kolejne etapy tworzenia się fraktala.

```
void RenderScene()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
    for (int i = 0; i <= stopien; i++)
    {
        glClear(GL_COLOR_BUFFER_BIT);
        rysuj(x - width / 2., y - width / 2., width, i);
        glFlush();
        sleep_for(500ms);
    }
}
```

Rysunek 2 - funkcja RenderScene().

Funkcja *rysuj(double x, double y, double width, int stopien)* stanowi główny element funkcjonalny programu. Składa się ona z pojedynczej instrukcji warunkowej. Gdy żądany stopień wynosi 0 – wyznaczane są kolejne wierzchołki kwadratu z uwzględnieniem deformacji i losowego koloru.

```
if (stopien == 0)
{
    float deformation_x = 0.f;
    float deformation_y = 0.f;

    glBegin(GL_POLYGON);
    glColor3f(((rand() % 101) * 0.01), ((rand() % 101) * 0.01), ((rand() % 101) * 0.01));
    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);
    glColor3f(((rand() % 101) * 0.01), ((rand() % 101) * 0.01), ((rand() % 101) * 0.01));
    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);
    glColor3f(((rand() % 101) * 0.01), ((rand() % 101) * 0.01), ((rand() % 101) * 0.01));
    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);
    glColor3f(((rand() % 101) * 0.01), ((rand() % 101) * 0.01), ((rand() % 101) * 0.01));
    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();

}
```

Rysunek 3 - funkcja rysuj dla stopnia równego 0.

Gdy stopień jest większy od 0 następuje podział zadanej długości boku na 3 równe części, a następnie dla wszystkich poza środkowym kwadratem następuje wywołanie funkcji *rysuj()* z odpowiednim przesunięciem punktów początkowych oraz stopniem pomniejszonym o 1. Występuje tu zjawisko rekurencyjnego wywoływania funkcji.

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

Rysunek 4 - funkcja rysuj dla stopnia różnego od 0.

3. Podsumowanie i wnioski

Zadanie stanowiło dobre przypomnienie języka C++ oraz zastosowania rekurencji. Instrukcja laboratoryjna w przyjazny sposób przedstawiała podstawowe funkcje biblioteki GLUT. Żądany efekt można osiągnąć na wiele sposobów, nie są one jednak implementacją algorytmu Sierpińskiego.

Podczas instalacji biblioteki występowały liczne problemy. Można za to obwiniać jej wiek – najnowsza wersja została wydana ponad 20 lat temu. W internecie ciężko znaleźć aktualne instrukcje dopasowane do współczesnych środowisk programistycznych i systemów operacyjnych. Alternatywą może być zastosowanie biblioteki FreeGlut – otwarto-źródłowego projektu tworzonego przez internautów mającego stanowić kontynuację i rozwój oryginalnej biblioteki.