Kurs:
Grafika komputerowa i komunikacja człowiek-komputer - Laboratorium

Open GL - podstawy

Autor:
Aleksandra Chrustek, 263900

Prowadzący:
dr inż. Jan Nikodem

18.10.2023

1 Wstęp

Celem drugiego laboratorium było zapoznanie się z elementarnymi możliwościami biblioteki graficznej OpenGL wraz z rozszerzeniem GL Utility Toolkit (GLUT). Ćwiczenie obejmowało inicjalizację i zamykanie trybu OpenGL oraz rysowanie tworów pierwotnych (prymitywów) w przestrzeni 2D.

2 Przebieg laboratorium

Pierwszym uruchomionym programem było okienko, stworzone na podstawie kodu przedstawionego w instrukcji. Następnie w ten sam sposób utworzone zostały kolejne programy z figurami: zielonym kwadratem, dwoma trójkątami, trojkątem wypełniony kolorem interpolowanym. Kolejną częścią laboratorium było zadanie do samodzielnego zrealizowania - dywan Sierpińskiego z perturbacjami.

3 Informacje teoretyczne

Dywan Sierpińskiego jest obiektem geometrycznym zdefiniowanym przez Wacława Sierpińskiego. Obiekt ten to fraktal, który otrzymuje się według następującego algorytmu:

1. Obiektem wyjściowym jest kwadrat o boku a.
2. Kwadrat dzielony jest n 9 mniejszych równych kwadratów o boku a/3 i usuwany jest środkowy kwadrat.
3. Każdy z mniejszych kwadratów ponownie dzielony jest na 9 części i z każdej części usuwana jest część środkowa.
4. Powtarzanie dalszych kroków według wyżej opisanej zasady.

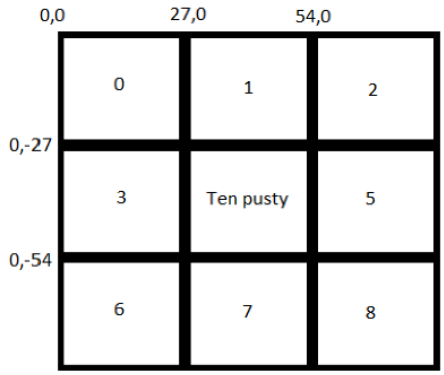
W programie dla uzyskania perturbacji wykorzystano sugerowaną w instrukcji funkcję rand() i obiekty typu GL_POLYGON. Ważne w realizacji zadania jest dobranie odpowiedniej długości boku kwadratu - powinna być to potęga liczby 3, aby proces dzielenia na kolejne kwadraty przebiegał pomyślnie. Cechą charakterystyczną dywanu Sierpińskiego jest to, że jego pole powierzchni wynosi 0. Przez perturbacje w tym zadaniu, rozumie się przesunięcie narożników kwadratów o losową wartość oraz losowanie ich kolorów.

4 Realizacja zadania

Szkielet programu i funkcja ChangeSize bazują na kodzie z instrukcji laboratoryjnej. Poniżej przedstawiono kod trzech funkcji odpowiedzialnych za wygenerowanie struktury.

```
void BigSquare(float x, float y)
{
int a = 0;

for (int i = 0; i < 3; i++)
{
for (int j = 0; j < 3; j++)
{
if (a != 4)
{
MiddleSquare(x- 10* squaresize,y+ 10* squaresize);
a++;
}
else a++;
x = x + 9 * squaresize;
}
x = x - 27 * squaresize;
y = y - 9 * squaresize;}
}
```

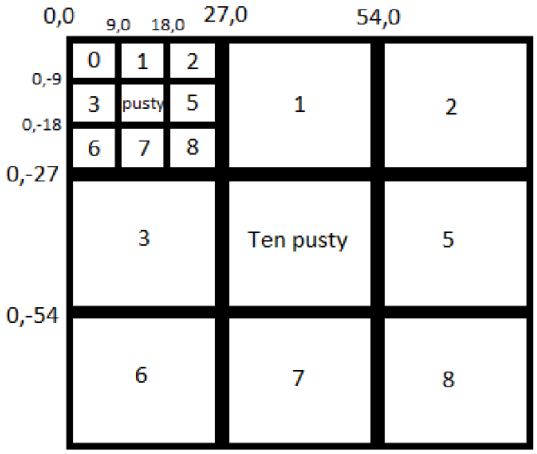


Rysunek 1: Działanie funkcji BigSquare dla kwadratu o wielkości 3

```
void MiddleSquare(float x, float y)
{
int a = 0;

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

```
if (a != 4)
{
SmallSquare(x, y);
a++;
}
else a++;
x = x + 3 * squaresize;
}
x = x - 9 * squaresize;
y = y - 3 * squaresize;}
}
```



Rysunek 2: Działanie funkcji MiddleSquare - analogiczne do funkcji BigSquare

```
void SmallSquare(float x, float y)
{
float _x = x - 3 * squaresize;
float _y = y + 3 * squaresize;
int a = 0;

for (int i = 0; i < 3; i++)
{
for (int j = 0; j < 3; j++)
{
if (a != 4)
{
float p = Perturbation();
glBegin(GL_POLYGON);
```

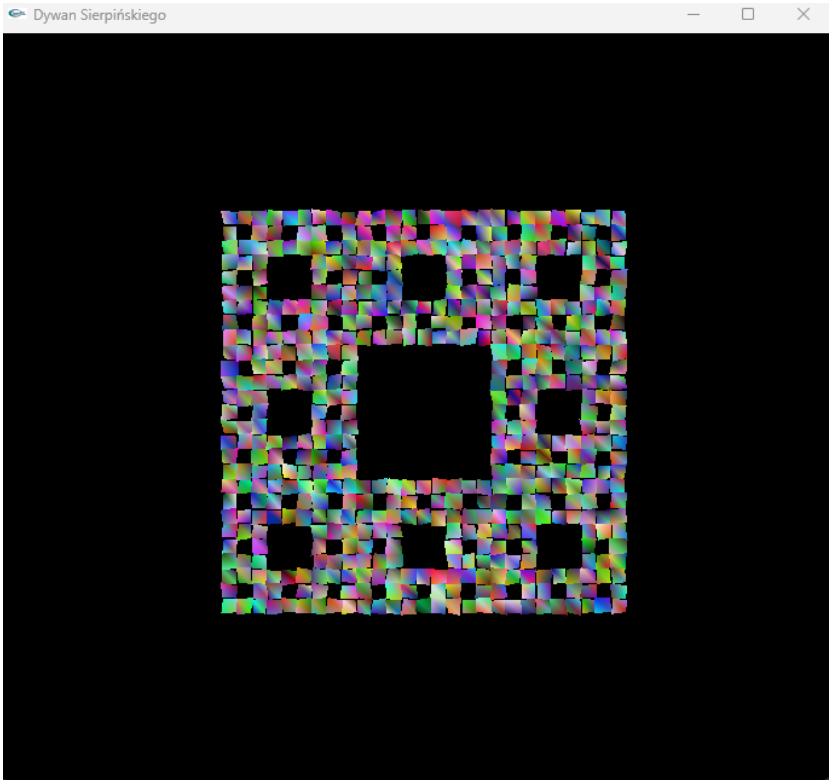
```
glColor3f(SquareColour(), SquareColour(), SquareColour());
    //Losowanie koloru dla pierwszego rogu
glVertex2f(_x + Perturbation(), _y + Perturbation());
glColor3f(SquareColour(), SquareColour(), SquareColour());
    //Losowanie koloru dla drugiego rogu
glVertex2f(_x + Perturbation() + squaresize, _y + Perturbation());
glColor3f(SquareColour(), SquareColour(), SquareColour());
    //Losowanie koloru dla trzeciego rogu
glVertex2f(_x + Perturbation() + squaresize, _y + Perturbation() - squaresize);
glColor3f(SquareColour(), SquareColour(), SquareColour());
    //Losowanie koloru dla czwartego 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 SmallSquare działa analogicznie do poprzednich funkcji, z tą różnicą, że rysuje ona kwadrat, który ma cztery wylosowane kolory. Następnie dokonuje na nim perturbacji dzięki przesunięciu o mały x i y względem odpowiednich współrzędnych. W efekcie kwadrat jest postrzępiony. Kolory oraz perturbacje generowane były losowo, a ich implementacja widoczna jest poniżej.

```
float SquareColour()
{
float x = rand() % 100;
return x / 99;
}

float Perturbation()
{
float x = rand() % 500;
return x / 499;
}
```

Uzyskany efekt końcowy prezentował się w następujący sposób:



Rysunek 3: Dywan Sierpińskiego

5 Wnioski

Instrukcja laboratoryjna pozwoliła na zapoznanie się ze środowiskiem oraz bibliotekami OpenGL i GLUT. Rozwiązanie napotkanych po drodze trudności wpłynęło pozytywnie na umiejętność pracy z figurami na płaszczyźnie i rozwinęło wyobraźnię, konieczną do przedstawienia figur za pomocą kodu programu. Dodatkowo zadanie pozwoliło na poznanie obiektu, jakim jest dywan Sierpińskiego, dzięki - niezbędnym do wykonania zadania - przygotowaniu do zajęć.