Projekt Indywidualny

Implementacja (w wybranym języku) generatora linii prostej w przestrzeni 3W wg zadanego algorytmu i przestrzeni barw RGB

Artur Szajdecki

1 Wstęp

Dokumentacja projekt zawiera opis problemu, instrukcje, procedury, przykłady i bibliografie. Do projektu jest dołączony kod programistyczny i program wykonywalny napisany w C++ który generuje linie prostą w przestrzeni 3W na obrazku. Kod zawiera dokumentacje programistyczną.

2 Opis Problemu

Mamy układ 3W. Każdy punkt posiada odpowiednie współrzędne całkowite x, y, z. Na układzie 3W wybieramy dwa punkty (x_1, y_1, z_1) oraz (x_2, y_2, z_2) i chcemy aby odcinek (x_1, y_1, z_1)-(x_2, y_2, z_2) tworzył linie składający się z punktów/pikseli. Problem sprowadza się do pokolorowania kolejnych pikseli, które leżą najbliżej punktu na odcinku o tej samej współrzędnej, co piksel. Algorytm Bresenhama określa, kiedy należy wykonać ruch w kierunku wolnym Y i Z (to która różnica dx, dy, dz jest największa, i przepuśćmy że w tym przykładzie dx jest największą różnicą).
Rozważmy równanie prostej przechodzącej przez dwa punkty (x_1, y_1, z_1) i (x_2, y_2, z_2).

$$dx = (x_2 - x_1)$$

$$dy = (y_2 - y_1)$$

$$dz = (z_2 - z_1)$$

$$y = y_1 + (x_2 - x_1) * \frac{dy}{dx}$$

$$z = z_1 + (x_2 - x_1) * \frac{dz}{dx}$$

W powyższym równaniu x oznacza współrzędną środka kolejnego piksela, Współrzędne y, z natomiast odnoszą się do punktów leżącego na prostej

zawierającej odcinek/piksele x. Środek piksela jest zwykle w pewnej odległości od odpowiadajacego mu punktu na odcinku. Oznaczmy tą odległość przez ddy i ddz, a współrzędną y i z środkami piksela przez y_p i z_p.

$$y = y_p + ddy$$

$$z = z_p + ddz$$

Wystawmy to wyrażenie do równania prostej:

$$y_p + ddy = y_1 + (x_2 - x_1) * \frac{dy}{dx}$$

$$z_p + ddz = z_1 + (x_2 - x_1) * \frac{dz}{dx}$$

Teraz przekształcamy wyrażenie dla y:

$$y_p + ddy = y_1 + (x_2 - x_1) * \frac{dy}{dx}$$

$$ddy = y_1 - y_p + (x_2 - x_1) * \frac{dy}{dx} / * (dx)$$

$$ddy * dx = (y_1 - y_p) * dx + (x_2 - x_1) * dy$$

$$ddy * dx = (y_1 - y_p) * dx + (x_2 - x_1) * dy / * (-1)$$

$$-ddy * dx = (y_p - y_1) * dx - (x_2 - x_1) * dy$$

Oraz przekształcamy wyrażenie dla z:

$$z_p + ddz = z_1 + (x_2 - x_1) * \frac{dz}{dx}$$

$$ddz = z_1 - z_p + (x_2 - x_1) * \frac{dz}{dx} / * (dx)$$

$$ddz * dx = (z_1 - z_p) * dx + (x_2 - x_1) * dz$$

$$ddz * dx = (z_1 - z_p) * dx + (x_2 - x_1) * dz / * (-1)$$

$$-ddz * dx = (z_p - z_1) * dx - (x_2 - x_1) * dz$$

Wyrażenia $-ddy * dx$ i $-ddz * dx$ są tzw. wyrażeniami błędów. Zgodnie ze wzorem krok w kierunku szybkim X powoduje zmiejszenie wyrażenia błędu

o dy i dz. Jeśli wartość wyrażenia będzie ujemna, to wykonujemy krok w kierunku wolny Y lub Z, czyli zwiększamy je o dx. Ponieważ $dx \geq dy$ and $dx \geq dz$, to wyrażenie błędu sprowadzi się znów do wartości dodatniej lub 0. Na tej podstawie algorytm Bresenhama określa kiedy należy wykonać ruch w kierunku Y lub Z - stara się zawsze utrzymać wyrażenie błędu w zakresie liczb dodatnich i zera. Wartości, ddy i ddz wcale nie musimy obliczać. Bardzo ważna jest wartość początkowa wyrażenia błedu. Zwykle przyjmuje się ją równą $\frac{dx}{2}$.

3 Instrukcja Obsługi Programu

Pierw program wymaga żeby użytkownik podał współrzędne początkowe linii 3D, współrzędne końcowe linii 3D oraz podał kod RGB, współrzedne muszą być jako wartość typu integer, a kod RGB jako łańcuch tekstowy w zapisie szesnastkowym, który prezentuje kolor linii, później pogram w terminalu wyświetla po kolej zmienne punktów które są odpowiedzialne za tworzenie linii, na koniec program wygeneruje, obrazek o nazwie 'Graph.png', na której jest układ 3W z linia o danym kolorze RGB.

4 Procedury

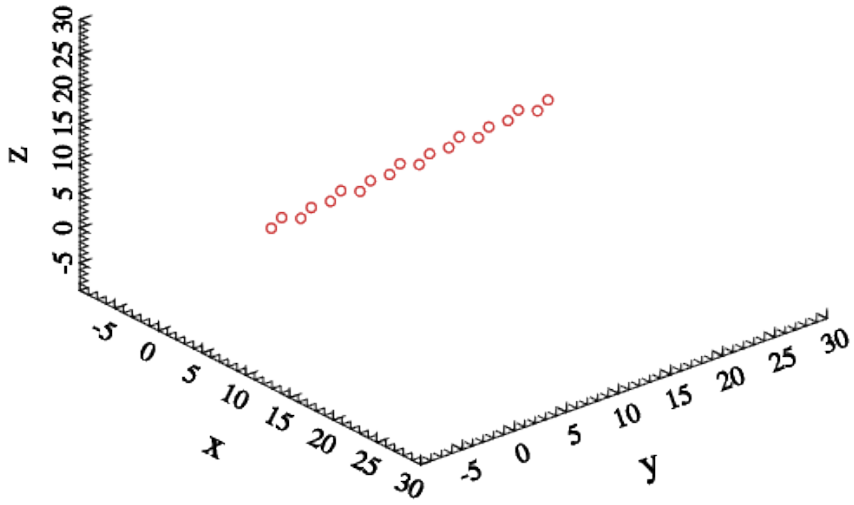
void InitMGL(const int x1, const int y1, const int z1, const int x2, const int y2, const int z2); Jest to procedura inicjalizująca układ 3W, posługująca się metodami z biblioteki MathGL, argumenty ten funkcji prezentują wielkość układu w 3W, czyli 3 pierwsze parametry prezentują początek układu, a 3 pozostałe argumenty prezentują koniec układu.

void Zaznacz Na Wykresie (int point[3], string rgb); Jest to procedura służąca do zaznaczania punktów na układzie 3W, argumentem ten funkcji jest tablica point[], prezentująca punkty osi, oraz argument prezentujący kolor RGB w hex

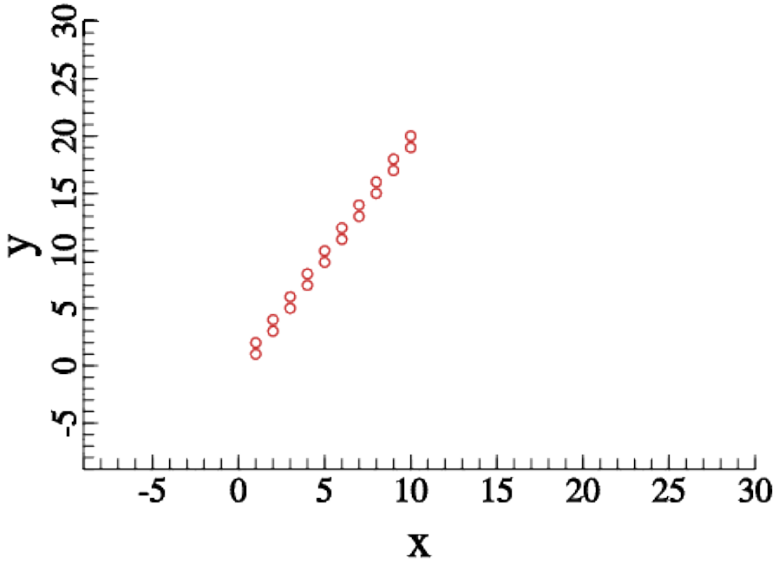
void Linia3D(int x1, int y1, int z1, const int x2, const int y2, const int z2, string rgb)
Procedura wykonująca algorytm Bresenhama dla linii 3D, przyjmuje następujące parametry, pierwsze 3 argumenty które trzeba podać, oznaczają początkową pozycje linii w układzie 3W, następne 3 parametry oznaczają końcową pozycje linii w układzie 3W, za to ostatni parametr prezentuje kolor RGB w zapisie HEX.

5 Przykładowe Wygenerowane Linie

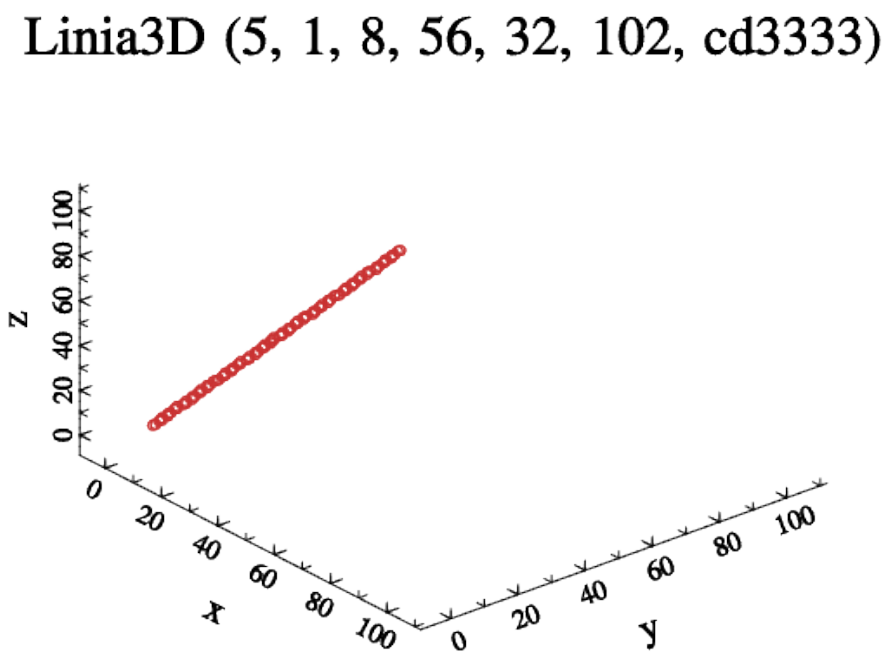
Linia3D (1, 1, 1, 10, 20, 15, cd3333)



Linia3D (1, 1, 1, 10, 20, 15, cd3333)



Powyżej są wygenerowane układy 3W oraz 2W z linią od punktu (1,1,1) do punktu (10,20,15). To są te same układy, tylko że patrzymy na nie z innej perspektywy. A tutaj kolejny przykład:



6 Bibliografie

J. E. Bresenham. Algorithm for computer control of a digital plotter. „IBM Systems Journal”. 4 (1) (ang.). [zarchiwizowane z adresu 2008-05-28].

Michał Jankowski: Elementy grafiki komputerowej. Warszawa: Wydawnictwa Naukowo-Techniczne, 1990, s. 27-34. ISBN 83-204-1326-5.

28 Stycznia 2015