**ЛАБОРАТОРНА РОБОТА №1**

З дисципліни «Комп’ютерна графіка»

|  |  |
| --- | --- |
| **Виконав**  Студент групи КВ-21  Комарницький О.Б.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Перевірив**  Ст. викл. кафедри СПіСКС  Клятченко Я.М.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Код програми

Алгоритм псевдопікселя:

public void putPixel(int x, int y)

{

Graphics g = Graphics.FromHwnd(pictureBox1.Handle);

SolidBrush myPixel = new SolidBrush(Color.Black);

g.FillRectangle(myPixel, x, y, N, N);

}

Алгоритм простого ЦДА:

public void SimpleDDA(int n, int x\_st, int y\_st, int x\_fin, int y\_fin)

{

putPixel(x\_st, y\_st);

int x\_prev = x\_st, y\_prev = y\_st, d\_x = x\_fin - x\_st, d\_y = y\_fin - y\_st;

for (int i = 0; i < n; i++)

{

int x = x\_prev + d\_x / n, y = y\_prev + d\_y / n;

putPixel(x, y);

x\_prev = x;

y\_prev = y;

}

}

Алгоритм несиметричного ЦДА:

public void AsymmetricalDDA(int x\_st, int y\_st, int x\_fin, int y\_fin)

{

int dx, dy, s;

dx = x\_fin - x\_st;

dy = y\_fin - y\_st;

if (dx >= 0 && dy >= 0)

{

putPixel(x\_st, y\_st);

dx += N;

dy += N;

if (dy == dx)

{

while (x\_st < x\_fin)

{

x\_st += N;

putPixel(x\_st, x\_st);

}

}

else if (dx > dy)

{

s = 0;

while (x\_st < x\_fin)

{

x\_st += N;

s += dy;

if (s >= dx)

{

s -= dx;

y\_st += N;

}

putPixel(x\_st, y\_st);

}

}

else // нахил > 45 градусів

{

s = 0;

while (y\_st < y\_fin)

{

y\_st += N;

s += dx;

if (s >= dy)

{

s -= dy;

x\_st += N;

}

putPixel(x\_st, y\_st);

}

}

}

else if (dx <= 0 && dy >= 0) // малюємо в другому октанті

{

putPixel(x\_st, y\_st);

dx -= N;

dy += N;

if (dy == Math.Abs(dx))

{

while (x\_st > x\_fin)

{

x\_st -= N;

putPixel(x\_st, x\_st);

}

}

else if (Math.Abs(dx) > dy) // нахил < 45 градусів

{

s = 0;

while (x\_st > x\_fin)

{

x\_st -= N;

s += dy;

if (s >= Math.Abs(dx))

{

s -= Math.Abs(dx);

y\_st += N;

}

putPixel(x\_st, y\_st);

}

}

else // нахил > 45 градусів

{

s = 0;

while (y\_st < y\_fin)

{

y\_st += N;

s += Math.Abs(dx);

if (s >= dy)

{

s -= dy;

x\_st -= N;

}

putPixel(x\_st, y\_st);

}

}

}

else if (dx <= 0 && dy <= 0) // малюємо в третьому октанті

{

putPixel(x\_st, y\_st); // малюємо початкову точку вектора

dx -= N; // вираховуємо кількість позицій по X та Y

dy -= N;

if (dy == dx) // генерація вектора (нахил 45 градусів)

{

while (x\_st > x\_fin)

{

x\_st -= N;

putPixel(x\_st, x\_st);

}

}

else if (dx < dy) // нахил < 45 градусів

{

s = 0;

while (x\_st > x\_fin)

{

x\_st -= N;

s += Math.Abs(dy);

if (s >= Math.Abs(dx))

{

s -= Math.Abs(dx);

y\_st -= N;

}

putPixel(x\_st, y\_st);

}

}

else // нахил > 45 градусів

{

s = 0;

while (y\_st > y\_fin)

{

y\_st -= N;

s += Math.Abs(dx);

if (s >= Math.Abs(dy))

{

s -= Math.Abs(dy);

x\_st -= N;

}

putPixel(x\_st, y\_st);

}

}

}

else // малюємо в четвертому октанті

{

putPixel(x\_st, y\_st);

dx += N;

dy -= N;

if (Math.Abs(dy) == dx) // генерація вектора (нахил 45 градусів)

{

while (x\_st < x\_fin)

{

x\_st += N;

putPixel(x\_st, x\_st);

}

}

else if (dx < Math.Abs(dy)) // нахил < 45 градусів

{

s = 0;

while (x\_st < x\_fin)

{

x\_st += N;

s += Math.Abs(dy);

if (s >= dx)

{

s -= dx;

y\_st -= N;

}

putPixel(x\_st, y\_st);

}

}

else // нахил > 45 градусів

{

s = 0;

while (y\_st > y\_fin)

{

y\_st -= N;

s += dx;

if (s >= Math.Abs(dy))

{

s -= Math.Abs(dy);

x\_st += N;

}

putPixel(x\_st, y\_st);

}

}

}

}

Алгоритм Брезенхема для побудови вектора:

public void AlgorithmBrezenhema(int x\_st, int y\_st, int x\_fin, int y\_fin) // вектор за алгоритмом Брезенхема

{

int dx, dy, s, sx, sy, kl, swap, incr1, incr2;

// обчислюємо прирощення і кроки

sx = 0;

if ((dx = x\_fin - x\_st) < 0)

{

dx = -dx;

--sx;

}

else if (dx > 0)

++sx;

sy = 0;

if ((dy = y\_fin - y\_st) < 0)

{

dy = -dy;

--sy;

}

else if (dy > 0)

++sy;

// враховуємо нахил

swap = 0;

if ((kl = dx) < (s = dy))

{

dx = s;

dy = kl;

kl = s;

swap++;

}

s = (incr1 = 2 \* dy) - dx; // incr1 - константа переобчислення різниці якщо поточне s < 0 і s - початкове значення різниці

incr2 = 2 \* dx; // константа для переобчислення різниці якщо поточне s >= 0

putPixel(x\_st, y\_st); // перший псевдопіксель вектора

while ((kl -= N) >= 0)

{

if (s >= 0)

{

if (swap != 0)

x\_st += (sx \* N);

else

y\_st += (sy \* N);

s -= incr2;

}

if (swap != 0)

y\_st += (sy \* N);

else

x\_st += (sx \* N);

s += incr1;

putPixel(x\_st, y\_st); // поточна точка вектора

}

}

Алгоритм ВУ:

public void AlgorithmWu(int x\_st, int y\_st, int x\_fin, int y\_fin)

{

var steep = Math.Abs(y\_fin - y\_st) > Math.Abs(x\_fin - x\_st);

if (steep)

{

Swap(ref x\_st, ref y\_st);

Swap(ref x\_fin, ref y\_fin);

}

if (x\_st > x\_fin)

{

Swap(ref x\_st, ref x\_fin);

Swap(ref y\_st, ref y\_fin);

}

WuPseudopixel(x\_st, y\_st, 1);

WuPseudopixel(x\_fin, y\_fin, 1);

double dx = x\_fin - x\_st;

double dy = y\_fin - y\_st;

double gradient = dy / dx;

double y = y\_st + gradient;

for (var x = x\_st + N; x <= x\_fin - N; x++)

{

WuPseudopixel(x, (int)y, 1 - (y - (int)y));

WuPseudopixel(x, (int)y + N, y - (int)y);

y += gradient;

}

}

Швидкодія**:**

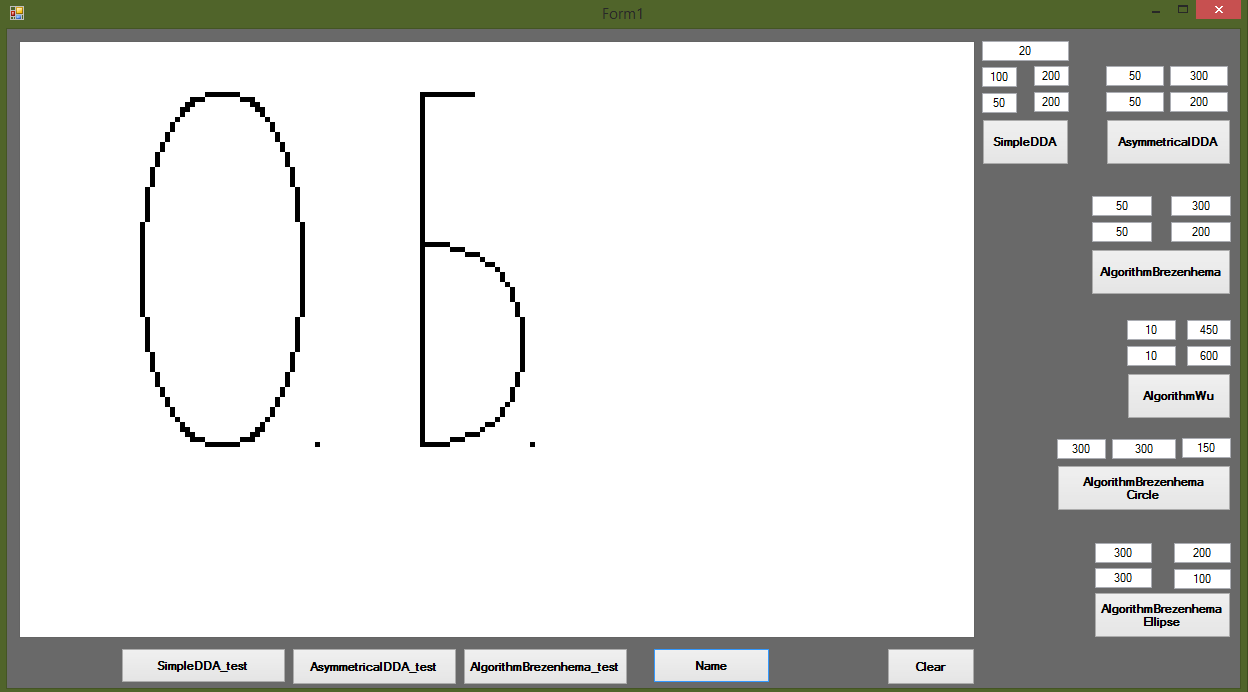
Згідно із оцінкою швидкодії найшвидшим виявився алгоритм просто ЦДА, адже це найпростіший метод , де вказується кількість вузлів для апроксимації відрізку та здійснюється мінімальна кількість математичних перевірок та обчислень.

Асиметричний алгоритм ЦДА показав найгірші результати, бо це дуже складний алгоритм, в ньому використовується багато перевірок (наприклад, для побудови вектора в різних октантах слід розглядати окремі випадки).

Алгоритм Брезенхема виконується швидше, ніж алгоритм Ву, що пояснюється тим, що в алгоритмі Ву будуємо попарно два псевдопікселя з різною яскравістю.

Побудова еліпса та кола не можна порівнювати у швидкодії ні між собою, ні з алгоритмами для побудови векторів. Побудова еліпса буде повільнішою ніж коло, адже коло є видом еліпса (2 радіуси однакові), тобто побудова часткового випадку простіша за цілісний повний підхід до реалізації даної задачі.

Приклади побудови програмою власних ініціалів:

****