НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №1

з дісципліни **«**Комп’ютерна графіка**»**

**«Графічні примітиви»**

Виконав:

бригада №5

ФІОТ гр. ІО-31

Долинний Олександр

Горох Олександр

Перевірив:

Саверченко В. Г.

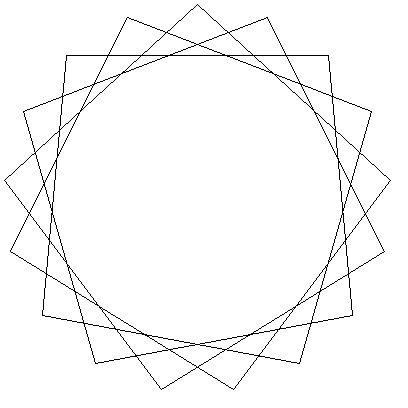
Київ 2015 р.

**Лабораторна робота №1**

**Тема: графічні примітиви.**

**Мета:** Навчитися будувати складні геометричні орнаменти з простих графічних примітивів.

**Завдання:** Відповідно до варіанту завдання графічно отримати зображення (орнамент) із однієї геометричної фигури, що повторюється. Використовуючи у якості параметрів відображення геометричної фігури розміри, кількість, крок тиражування, взаємне розташування, колір і інше, відтворити орнамент інтерактивної графіки.



**Варіант завдання № 5:**

Код програми:

**Клас Vector**

**public** **class** Vector {

**private** **int** n;

**private** **double**[][] vector;

**public** Vector(**int** n) {

vector = **new** **double**[n][2];

**this**.n = n;

}

**public** **double** getIX(**int** N) {

**return** vector[N][0];

}

**public** **double** getIY(**int** N) {

**return** vector[N][1];

}

**public** **void** setIX(**int** N, **double** X) {

vector[N][0] = X;

}

**public** **void** setIY(**int** N, **double** Y) {

vector[N][1] = Y;

}

}

**Клас Element**

**import** java.awt.\*;

**public** **class** Element {

**private** **double** x;

**private** **double** y;

**private** **double** r;

**private** **int** n;

**private** **int** d;

**private** **double** a;

**private** Vector vector;

**public** Element(**double** x, **double** y, **double** r, **int** n, **int** d) {

**this**(x, y, r, n, d, 0);

}

**public** Element(**double** x, **double** y, **double** r, **int** n, **int** d, **double** a) {

**this**.x = x;

**this**.y = y;

**this**.r = r;

**this**.n = n;

**this**.d = d;

**this**.a = a;

**this**.vector = **new** Vector(**this**.n);

rotate();

}

**public** **void** rotate() {

rotate(0);

}

**public** **void** rotate(**double** a) {

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

vector.setIX(i, **this**.x + **this**.r \* Math.*sin*(**this**.a + a + 2 \* Math.***PI*** \* i / n));

vector.setIY(i, **this**.y + **this**.r \* Math.*cos*(**this**.a + a + 2 \* Math.***PI*** \* i / n));

}

}

**public** **void** draw(Graphics g) {

**if** (**this**.getN() % **this**.getD() == 0) {

**for** (**int** i = 0; i < **this**.getN(); i = i + **this**.getD()) {

g.drawLine((**int**) **this**.getVector().getIX(i), (**int**) **this**.getVector().getIY(i), (**int**) **this**.getVector().getIX((i + **this**.getD()) % **this**.getN()), (**int**) **this**.getVector().getIY((i + **this**.getD()) % **this**.getN()));

}

} **else** {

**for** (**int** i = 0; i < **this**.getN(); i++) {

g.drawLine((**int**) **this**.getVector().getIX(i), (**int**) **this**.getVector().getIY(i), (**int**) **this**.getVector().getIX((i + **this**.getD()) % **this**.getN()), (**int**) **this**.getVector().getIY((i + **this**.getD()) % **this**.getN()));

}

}

}

**public** **double** getX() { **return** **this**.x; }

**public** **double** getY() { **return** **this**.y; }

**public** **double** getR() { **return** **this**.r; }

**public** **int** getN() { **return** **this**.n; }

**public** **int** getD() { **return** **this**.d; }

**public** Vector getVector() { **return** **this**.vector; }

}

**Клас Drawing**

**import** javax.swing.\*;

**import** java.awt.\*;

**public** **class** Drawing **extends** JPanel {

**public** Element elem;

**public** **static** **final** **int** ***FRAME\_WIDTH*** = 1200;

**public** **static** **final** **int** ***FRAME\_HEIGHT*** = 700;

**public** **void** paintComponent(Graphics g) {

**super**.paintComponent(g);

**int** sum = 1000;

**int** num = 20;

**int** step = 7;

**double** rot = 0;

Element[] test = **new** Element[sum];

**for** (**int** i = 1; i <= sum; i++) {

test[i - 1] = **new** Element(***FRAME\_WIDTH*** / 2, ***FRAME\_HEIGHT*** / 2, 350.0 / sum \* i, num, step, rot);

}

**for** (**int** k = 0; k < sum; k++) {

g.setColor(**new** Color(255 - (**int**) (100.0 / sum \* k), 255 - (**int**) (100.0 / sum \* k), 255 - (**int**) (100.0 / sum \* k)));

**if** (test[k].getN() % test[k].getD() == 0) {

**for** (**int** i = 0; i < test[k].getN(); i = i + test[k].getD()) {

// g.setColor(new Color(255-(int)(200.0/test[k].getN()\*i),255-(int)(200.0/test[k].getN()\*i),255-(int)(200.0/test[k].getN()\*i)));

g.drawLine((**int**) test[k].getVector().getIX(i),

(**int**) test[k].getVector().getIY(i),

(**int**) test[k].getVector().getIX((i + test[k].getD()) % test[k].getN()),

(**int**) test[k].getVector().getIY((i + test[k].getD()) % test[k].getN()));

}

} **else** {

**for** (**int** i = 0; i < test[k].getN(); i++) {

//g.setColor(new Color(255-(int)(200.0/test[k].getN()\*i),255-(int)(200.0/test[k].getN()\*i),255-(int)(200.0/test[k].getN()\*i)));

g.drawLine((**int**) test[k].getVector().getIX(i), (**int**) test[k].getVector().getIY(i), (**int**) test[k].getVector().getIX((i + test[k].getD()) % test[k].getN()), (**int**) test[k].getVector().getIY((i + test[k].getD()) % test[k].getN()));

}

}

}

}

**public** **void** addElements(Element el) { elem = el; }

**public** **static** **void** main(String[] args) {

JFrame frame = **new** JFrame();

Drawing test = **new** Drawing();

test.setOpaque(**true**);

test.setBackground(**new** Color(255, 255, 255));

frame.add(test);

frame.setVisible(**true**);

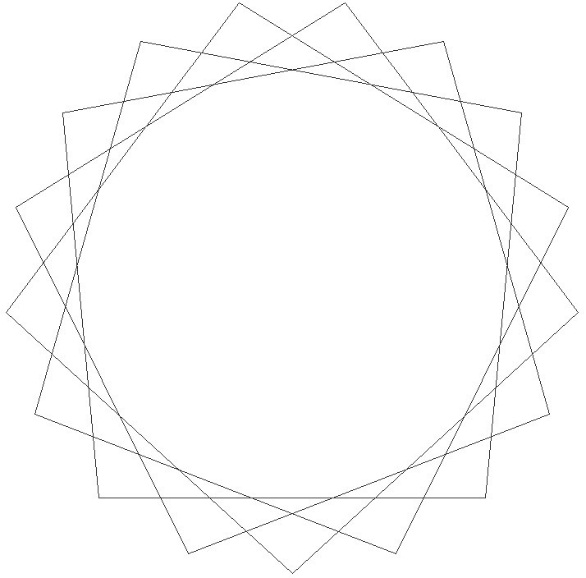
}

}

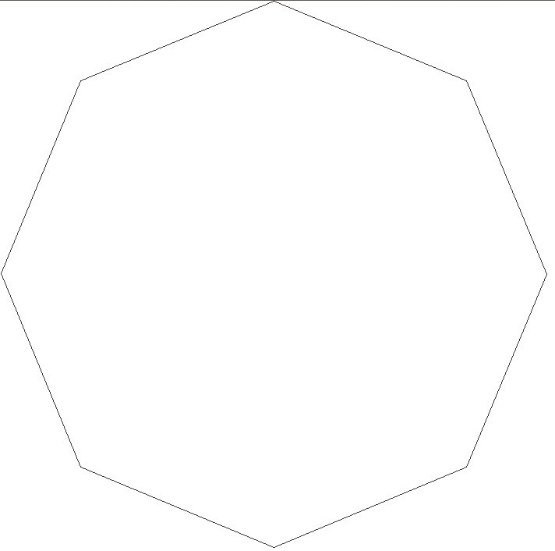
**Висновки:**

Для створення вікна для малювання використали бібліотеки AWT та Swing.

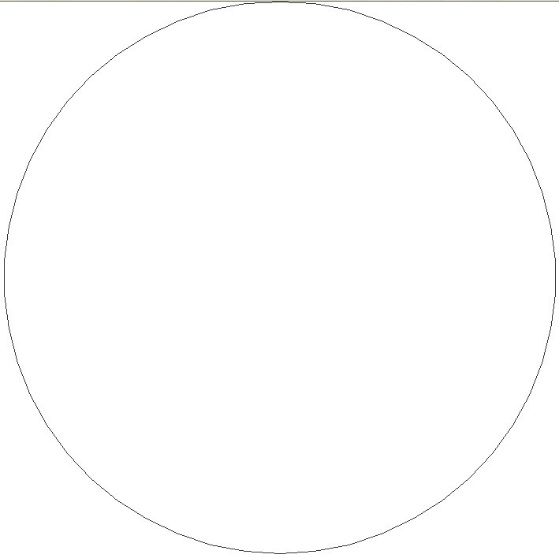
Кожен графічний примітив задається параметрами ***x,y*** координат центра кола, радіусом ***r*** кола, по якому розташовані точки, загальною кількістю ***n*** точок, крок ***d***, через який з’єднувались точки, та масивом ***vector*** координат вершин, що обчислюється. Починаючи з початкової вершини методом drawLine() з’єднувалися вершини згідно з початковими даними.



Звичайна фігура, що задана у завданні. n=17, d=4

Якщо фігура, що задана кількістю n\*p точок та кроком d\*p, та d є дільником k, то така фігура вироджується да правильного n/d –кутника

Наприклад, n=16, d=2



При досить великих значеннях відношення n/d (наприклад, для r=300 k/d=50) правильний k/d –кутник стає схожим на коло.

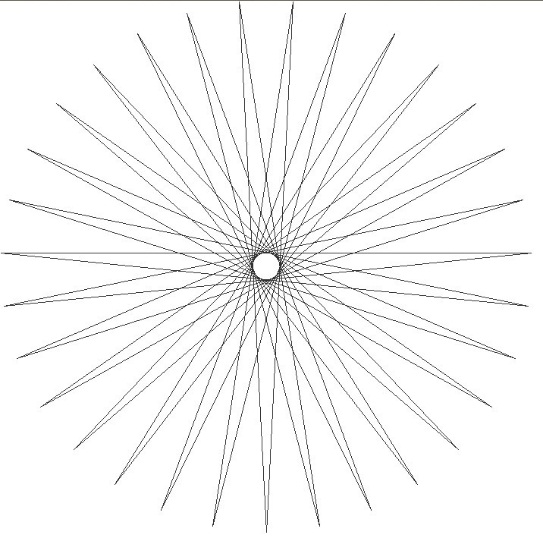
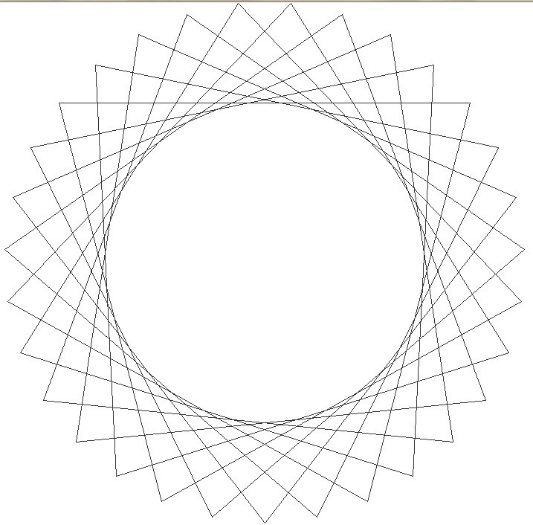
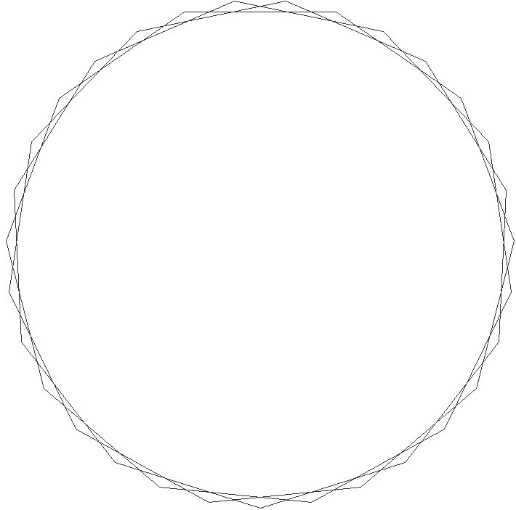
Наприклад, n=100, d=2

Фігури із кількістю точок n та параметрами кроку d і (n-d) э еквівалентними.

При невеликій кількості точок n(декілька десятків) і НСД(n,d)=1 отримуємо при малих d порівняно з n приблизно d рядів ліній(зосереджені біля границі початкового кола), при d->n/2 отримуємо “зірку”(зосереджені рівномірно в крузі). При проміжних значеннях d отримуємо комбінацію цих критичних випадків.

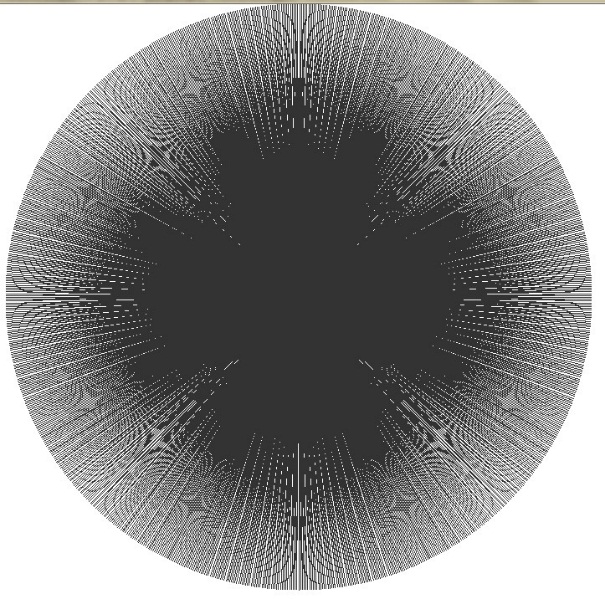
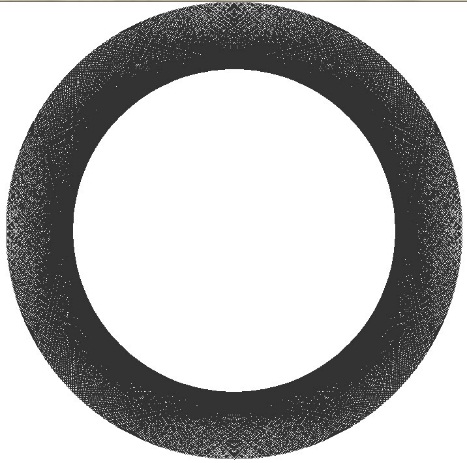
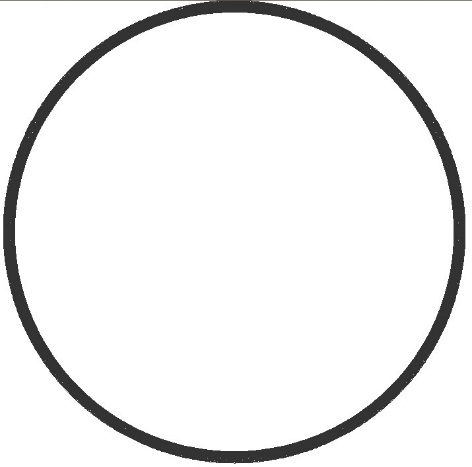
Наприклад,

n=31, d=3; n=31, d=9; n=31, d=15



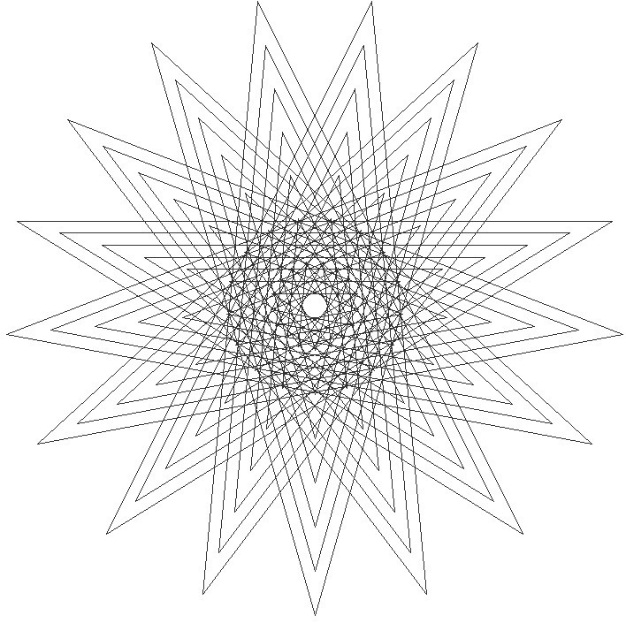
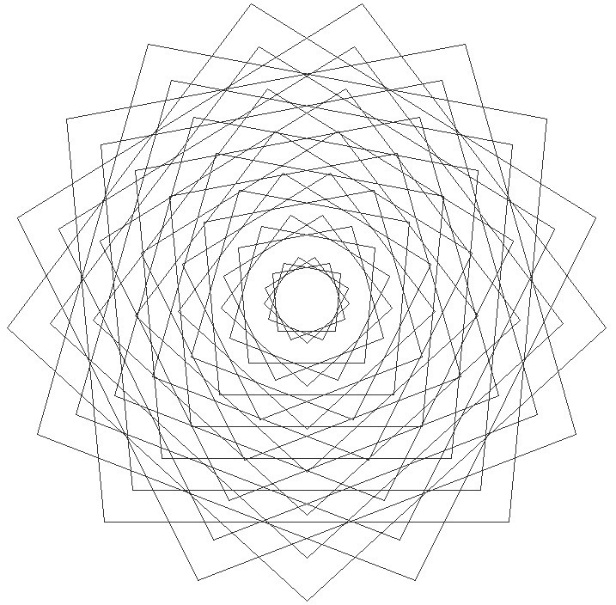
При кількості точок n(біля тисячі) отримуємо при малих d порівняно з n кільце, при d->n/2 отримуємо інтерференційну картинку. При проміжних значеннях d отримуємо комбінацію цих критичних випадків.

n=1001, d=100; n=1001, d=250; n=1001, d=500



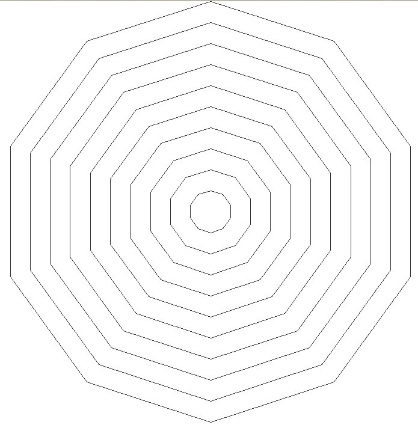
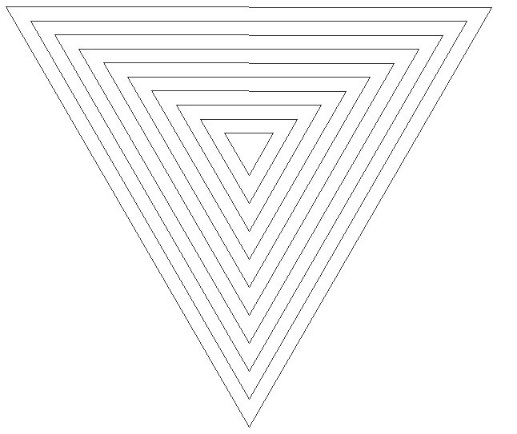
Комбінуючи декілька примітивів, отримуємо орнамент. (***sum***=кількість примітивів)

sum=7; n=17; d=4 – ефект рози; sum=7; n=17; d=7 – ефект зірки



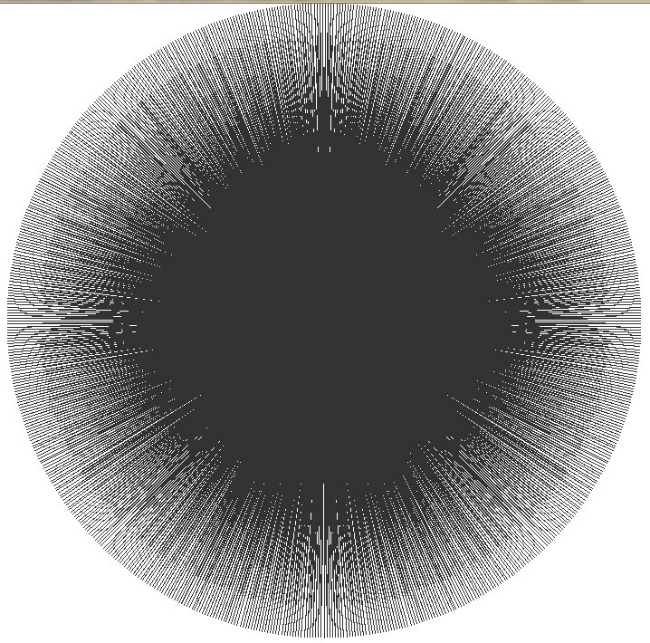
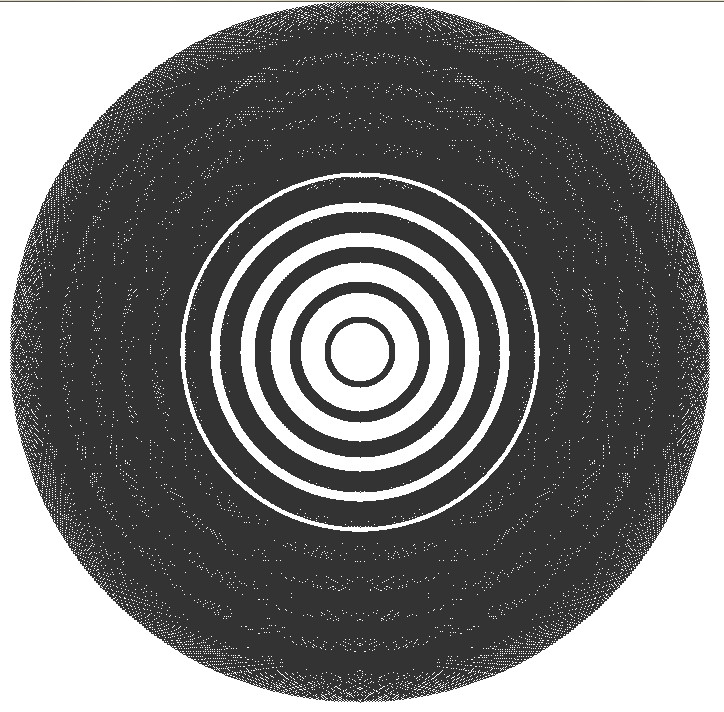
Комбінуючи вироджені фігури:

sum=10; n=40; d=4; sum=10; n=3; d=1

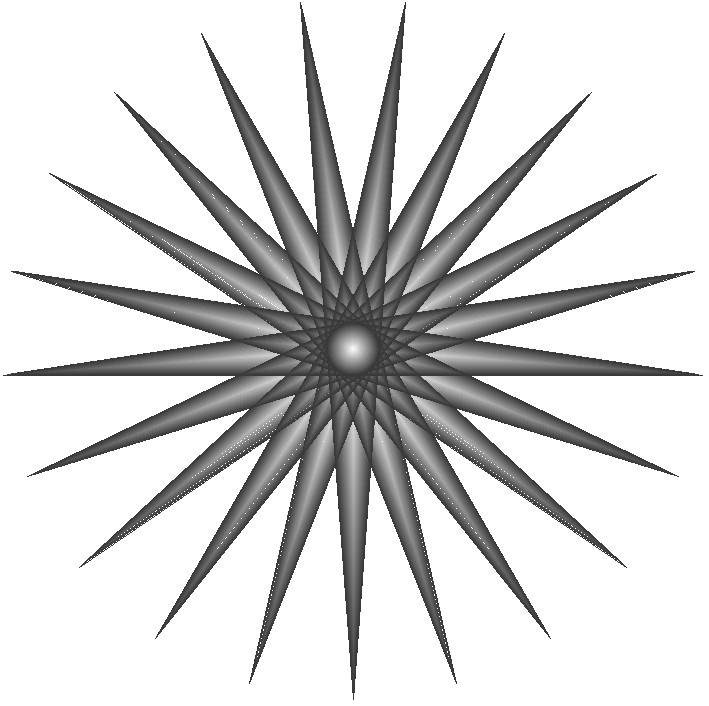
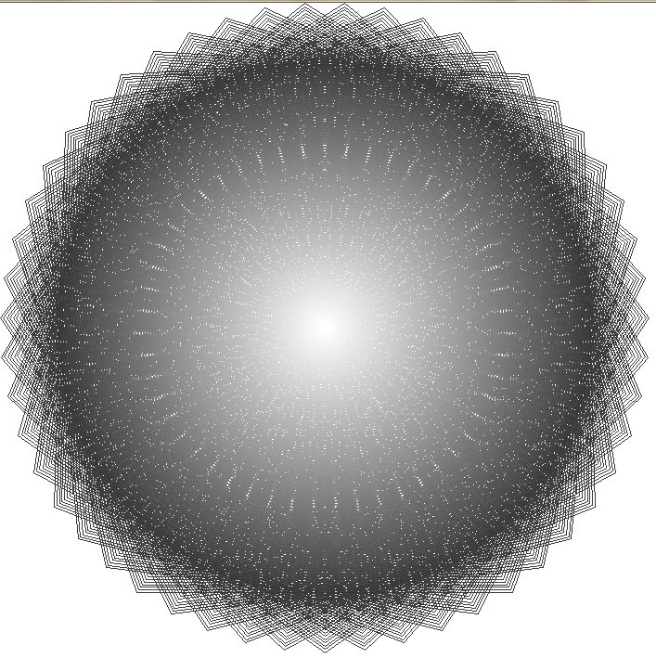
Комбінуємо примітиви при великих кількостях точок.

sum=10; n=701; d=120; sum=10; n=701; d=350



Проте для великої кількості точок та великої кількості операцій краще використати ефект градієнту – новіший елемент малювати світлішим кольором.

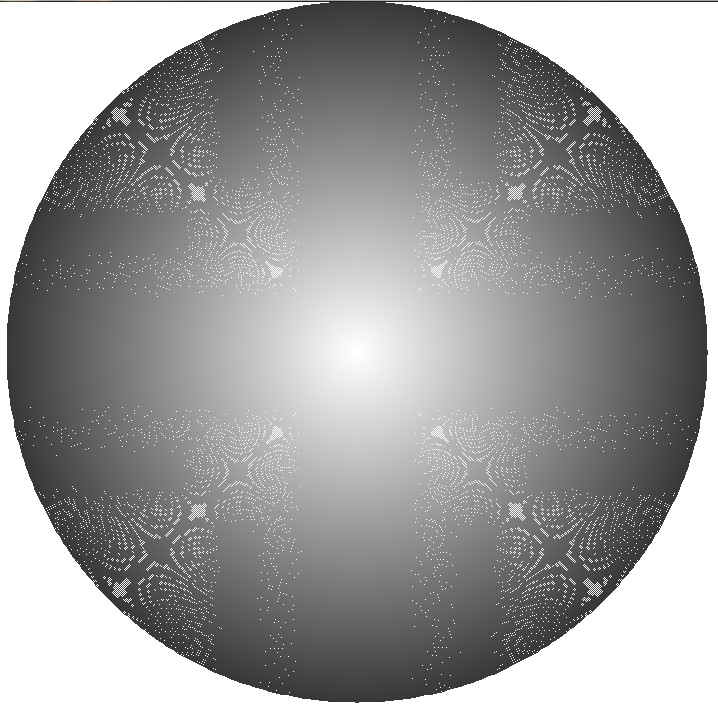
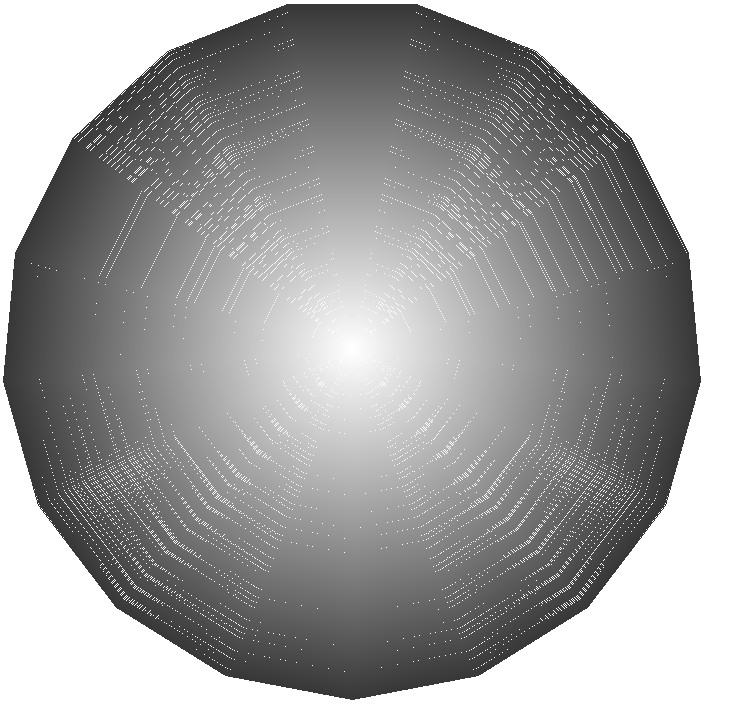
sum=100; n=51; d=10; sum=100; n=21; d=10



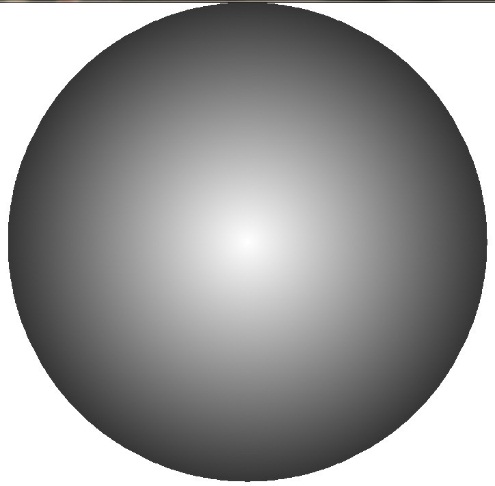
Збільшуючи кількість ітерацій, ми збільшуємо деталізацію

sum=1000; n=51; d=10 sum=1000; n=500; d=1

ефект «вентилятора» ефект «вишиванка»



sum=1000; n=500; d=37 – ефект «шара»

****