

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

Т. В. ГЕРАСИМОВА

**Компьютерная графика:
лабораторный практикум.**

Лабораторная работа 4
"Построение фракталов"

Санкт-Петербург
Издательство СПбГЭТУ «ЛЭТИ»
2021

Задание

На базе предыдущей лабораторной работы разработать программу реализующую фрактал по индивидуальному заданию.

1. Общие сведения

"Среди всех картинок, которые может создавать компьютер, лишь немногие могут поспорить с фрактальными изображениями, когда идет речь о подлинной красоте. У большинства из нас слово "фрактал" вызывает в памяти цветные завитушки, формирующие сложный, тонкий и составной узор."

Из книги Джефа Проузиса "Как работает компьютерная графика"

1.2. Понятие "фрактал"

Понятия фрактал и фрактальная геометрия, появившиеся в конце 70-х, с середины 80-х прочно вошли в обиход математиков и программистов. Слово фрактал образовано от латинского **fractus** и в переводе означает состоящий из фрагментов. Оно было предложено Бенуа Мандельбротом в 1975 году для обозначения нерегулярных, но самоподобных структур, которыми он занимался. Рождение фрактальной геометрии принято связывать с выходом в 1977 году книги Мандельброта 'The Fractal Geometry of Nature'. В его работах использованы научные результаты других ученых, работавших в период 1875-1925 годов в той же области (Пуанкаре, Фату, Жюлиа, Кантор, Хаусдорф). Но только в наше время удалось объединить их работы в единую систему.

Фрактал (лат. fractus — дробленный) — термин, означающий геометрическую фигуру, обладающую свойством самоподобия, то есть составленную из нескольких частей, каждая из которых подобна всей фигуре целиком.

Существует большое число математических объектов называемых фракталами (треугольник Серпинского, снежинка Коха, кривая Пеано, множество Мандельброта). Фракталы с большой точностью описывают многие физические явления и образования реального мира: горы, облака, турбулентные (вихревые) течения, корни, ветви и листья деревьев, кровеносные сосуды, что далеко не соответствует простым геометрическим фигурам.

1.3. Классификация фракталов.

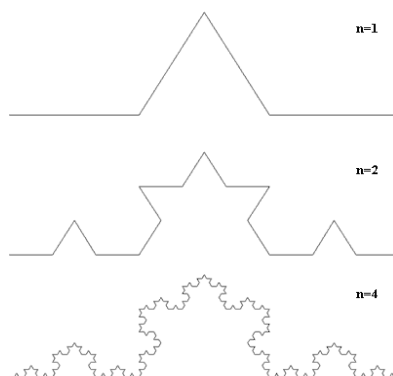
1.3.1. Геометрические фракталы.

Фракталы этого класса самые наглядные. В двумерном случае их получают с помощью ломаной (или поверхности в трехмерном случае), называемой генератором. За один шаг алгоритма каждый из отрезков, составляющих ломаную, заменяется на ломаную-генератор в соответствующем масштабе. В результате бесконечного повторения этой процедуры получается геометрический фрактал.

Рассмотрим на примере один из таких фрактальных объектов – триадную кривую Коха.

Построение триадной кривой Коха.

Возьмем прямолинейный отрезок длины 1. Назовем его затравкой. Разобьем затравку на три равные части длиной в $1/3$, отбросим среднюю часть и заменим ее ломаной из двух звеньев длиной $1/3$.



Построение триадной кривой Коха

Мы получим ломаную, состоящую из 4 звеньев с общей длиной $4/3$, - так называем первое поколение. Для того чтобы перейти к следующему поколению кривой Коха, надо у каждого звена отбросить и заменить среднюю часть. Соответственно длина второго поколения будет $16/9$, третьего – $64/27$. если продолжить этот процесс до бесконечности, то в результате получится

Особенности триадной кривой Коха:

Во-первых, эта кривая не имеет длины – с числом поколений ее длина стремится к бесконечности.
Во-вторых, к этой кривой невозможно построить касательную – каждая ее точка является точкой перегиба, в которой производная не существует, - эта кривая не гладкая.
В-третьих, к триадной кривой Коха традиционные методы геометрического анализа оказались неприменимы.

2.3.2. Алгебраические фракталы

Это самая крупная группа фракталов. Получают их с помощью нелинейных процессов в n-мерных пространствах. Наиболее изучены двумерные процессы. В качестве примера рассмотрим множество Мандельброта.

Математическое описание модели следующее: на комплексной плоскости в некоем интервале для каждой точки c вычисляется рекурсивная функция $Z=Z^2+c$. В модели Мандельброта изменяющимся фактором является начальная точка c , а параметр z , является зависимым.

Графическая реализация: начальная точка модели равна нулю. Графически она соответствует центру тела “груши”. Через N шагов заполнятся все тело груши и в том месте, где закончилась последняя итерация, начинает образовываться “голова” фрактала. “Голова” фрактала будет ровно в четыре раза меньше тела, так как математическая формула фрактала представляет из себя квадратный полином. Затем опять через N итераций у “тела” начинает образовываться “почка” (справа и слева от “тела”). И так далее. Чем больше задано число итераций N, тем более детальным получится изображение фрактала, тем больше будет у него различных отростков. Схематическое изображение стадий роста фрактала Мандельброта представлено на рис.2:

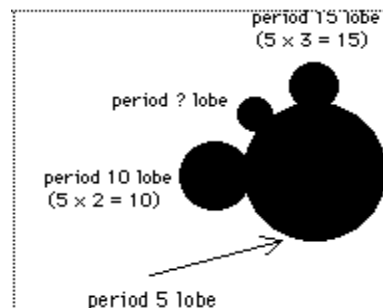
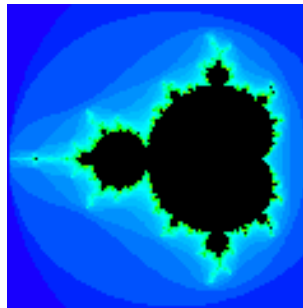


Схема образования фрактала Мандельброта



компьютерное изображение фрактала Мандельброта

1.2.3. Стохастические (случайные) фракталы

Еще одним известным классом фракталов являются стохастические фракталы, которые получаются в том случае, если в итерационном процессе хаотически менять какие-либо его параметры. При этом получаются объекты очень похожие на природные - несимметричные деревья, изрезанные береговые линии и т.д. Двумерные стохастические фракталы используются при моделировании рельефа местности и поверхности моря. Примерами стохастических фракталов являются фрактальные кривые, возникающие в критических двумерных моделях статистической механики, траектория броуновского движения на плоскости и в пространстве, плазма.

2. Способы построения фракталов

2.1. L-система

L-система (от имени Lindenmayer) - это грамматика некоторого языка (достаточно простого), которая описывает инициатор и преобразование, выполняемое над ним, при помощи средств, аналогичных средствам языка Лого (аксиоматическое описание простейших геометрических фигур и допустимых преобразований на плоскости и в пространстве).

L-системы часто называются ещё и системами черепашьей графики. Черепашья графика - это такой способ рисования линий на экране компьютера. Он состоит в том, что программист как бы управляет движением как бы черепашки. Черепашка, ползая по экрану, оставляет за собой след. При этом цель программиста – управлять черепашкой так, чтобы черепашка нарисовала нужную линию. Команды управления черепашкой просты: сделать шаг вперёд (обозначается F), повернуть направо (обозначается +), повернуть налево (обозначается -), сделать шаг вперёд без перерисовки (прыжок, обозначается V). Вот из этих команд и составляется сценарий построения линии – строка команд. Величина одного шага и угол одного поворота при движении черепашки всегда остаются постоянными и задаются предварительно.

Например, F++F++F это равносторонний треугольник, если угол поворота равен $\pi/3$, а F+F+F+F – это квадрат, если угол поворота равен $\pi/2$.

Построение L-системы происходит в три этапа.

Сначала создаётся сценарий поведения черепашки.

Потом подсчитывается размер линии, которая получится, если запустить этот сценарий на исполнение. Линия как бы рисуется в уме, а потом смотрится её размер. На основе этого размера подправляется масштаб, чтобы вся линия уместилась на экране.

На экран запускается черепашка, которая рисует линию.

Фракталы – самоподобные фигуры, значит, и сценарии у них должны быть самоподобные. Вот как это делается.

- Берётся начальная фигура (называется - аксиома), например, F++F++F с углом $\pi/3$.
- Задаётся правило замены F (называется правило newF), например, new F = F-F++F-F;
- в имеющейся фигуре все F заменяются на newF.

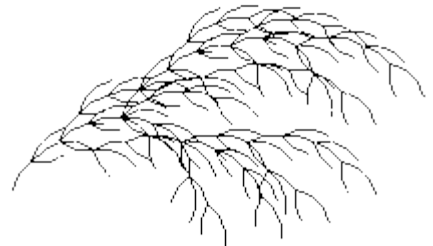
У деревьев есть ветки. Тут имеющимся набором команд не обойдётся. Приходится вводить ещё пару команд: начало ветви (обозначается [), конец ветви (обозначается]). Что бедная черепашка должна делать по этим командам? В начале ветви она должна запомнить своё состояние (положение и направление взгляда), а вот когда ей встретится соответствующий конец ветви, она должна вернуться в то положение, которое запомнила.

Вот, например, данные для построения куста.

axiom = F

newF = -F+F+[+F-F-]-[-F+F+F]

turn = $\pi / 8$



Если вы хотите строить объёмные деревья и другие фракталы, то вам придётся ввести ещё пару команд – поворотов из плоскости (предлагаю символы >, <). Короче, + - это повороты в плоскости XY, а >, < это повороты в плоскости XZ.

Вот, например, данные для построения 3d-кустика.

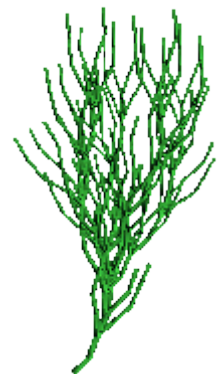
axiom = F

newF = -F+F[>F+F<F-F][<F-F>F+F]

[+F<F-F>F][<F>F+F<F]

turn = $\pi / 8$

Для двух шагов получите вот такое деревце ---> Для трёх шагов получим такую картинку ->



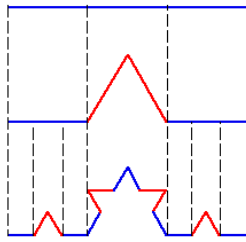
2.2. Система итерирующих функций IFC

Система итерирующих функций IFC. Применение таких преобразований, которые дают ту фигуру которую необходимо. Система итерирующих функций - это совокупность сжимающих аффинных преобразований. Как известно, аффинные преобразования включают в себя масштабирование, поворот и параллельный перенос. Аффинное преобразование считается сжимающим, если коэффициент масштабирования меньше единицы.

Рассмотрим подробнее построение кривой Коха с использованием аффинных преобразований. Каждый новый элемент кривой содержит четыре звена, полученных из образующего элемента использованием масштабирования, поворота и переноса.

1. Для получения первого звена достаточно сжать исходный отрезок в три раза. Следует отметить, что тоже масштабирование применяется для всех звеньев.
2. Следующее звено строится с использованием всех возможных преобразований, а именно: сжатие в три раза, поворот на -60° и параллельный перенос на $1/3$ по оси X.
3. Третье звено строится аналогично второму: сжатие в три раза, поворот на 60° , параллельный перенос на $2/3$ по оси X.
4. Последнее звено: сжатие в три раза, параллельный перенос на $2/3$ по оси X.

В дальнейшем правила построения кривой Кох будем называть IFS для кривой Кох.



На первой итерации кривая состоит из 4 фрагментов с коэффициентом сжатия $r = 1/3$, два сегмента повернуты на 60 град. по час. и против час. ст.

$f_1(x) \rightarrow$ масшт. на r

$f_2(x) \rightarrow$ масшт. на r , поворот на 60°

$f_3(x) \rightarrow$ масшт. на r , поворот на -60°

$f_4(x) \rightarrow$ масшт. на r

$$f_1(\mathbf{x}) = \begin{bmatrix} 0.333 & 0 \\ 0 & 0.333 \end{bmatrix} \mathbf{x}$$

Scale by r

$$f_2(\mathbf{x}) = \begin{bmatrix} 0.167 & -0.289 \\ 0.289 & 0.167 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0.333 \\ 0 \end{bmatrix}$$

Scale by r , rotation by 60°

$$f_3(\mathbf{x}) = \begin{bmatrix} 0.167 & 0.289 \\ -0.289 & 0.167 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0.500 \\ 0.289 \end{bmatrix}$$

Scale by r , rotation by -60°

$$f_4(\mathbf{x}) = \begin{bmatrix} 0.333 & 0 \\ 0 & 0.333 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0.667 \\ 0 \end{bmatrix}$$

Scale by r

Вывод

В результате выполнения лабораторной работы должна быть разработана программа, реализующая представление заданного фрактала

2.3. Цель, требования и рекомендации к выполнению задания

Цель выполнения задания: ознакомление с основными примитивами OpenGL.

Требования и рекомендации к выполнению задания:

- проанализировать полученное задание, выделить информационные объекты и действия;
- разработать программу с использованием требуемых примитивов и атрибутов.

2.3.1. Задания

Индивидуальное задание для выполнения лабораторной работы выдается преподавателем.

2.3.2. Пример выполнения задания

2.3.2.1. Теоретические сведения:

Пеано нарисовал особый вид линии (рис 4.6.). Для ее рисования Пеано использовал следующий алгоритм.

На первом шаге он брал прямую линию и заменял ее на 9 отрезков длиной в 3 раза меньшей, чем длина исходной линии. Далее он делал то же самое с каждым отрезком получившейся линии.



Рис. 4.6

И так до бесконечности. Ее уникальность в том, что она заполняет всю плоскость. Доказано, что для каждой точки на плоскости можно найти точку, принадлежащую линии Пеано.

2.3.2.2. Текст процедур построения фрактала

1. В этом фрагменте показана процедура рисования линии.

```
void CLab_4View::DrawLines(float x, float y, float x1, float y1, float width,
float red, float green, float blue)
{
    glLineWidth(width); glBegin(GL_LINES);
    glColor3f(red,green,blue); glVertex3f(x,y,0.0f); glVertex3f(x1,y1,0.0f);
    glEnd(); }
```

2. В этом фрагменте показана процедура рисования трех линий.

```
void CLab_4View::Draw3Lines(float length, float a, float& x, float& y, int b,
int type, int i11)
{
    float dx, dy, x1, y1;      int i011;      i011 = i11;
    if (type == 0)
    { //горизонтальная линия
        for (int j = b; j <= b+2; j++)
        {
            if ((j == 2) || (j == 4) || (j == 6) || (j == 8))
            { dx = 0;      dy = length * a; }
            else
            {
                if ((j == 1) || (j == 3) || (j == 5) || (j == 7) || (j ==
9))
                { dx = length * a; dy = 0; }
            }
            x1 = x + dx; y1 = y + dy;
            if (dx != 0)
            { type = 0; /*горизонтальная линия*/ }
            else
            {
                if (dx == 0)
                { type = 1; /*вертикальная линия*/ }
            }
            DrawFractal(length, a, x, y, x1, y1, type, i11);
            i11 = i011; x = x1; y = y1; type = 0; }
        }
    }
    else
    { //вертикальная линия
        for (int j = b; j <= b+2; j++)
        {
            if ((j == 2) || (j == 4) || (j == 6) || (j == 8))
            {
                if ((j == 1) || (j == 3) || (j == 5) || (j == 7) || (j ==
9))
                { dx = length * a; dy = 0; }
            }
            x1 = x + dx; y1 = y + dy;
            if (dx != 0)
            { type = 0; /*горизонтальная линия*/ }
            else
            {
                if (dx == 0)
                { type = 1; /*вертикальная линия*/ }
            }
            DrawFractal(length, a, x, y, x1, y1, type, i11);
            i11 = i011; x = x1; y = y1; type = 0; }
        }
    }
}
```

```

dx = length * a; dy = 0; }
else
{
    if((j == 1) || (j == 3) || (j == 5) || (j == 7) || (j ==
9))
        { dx = 0; dy = length * a; }
}
x1 = x + dx; y1 = y + dy;
if (dy == 0) { type = 0; /*горизонтальная линия*/}
else
{ if(dy != 0) { type = 1; /*вертикальная линия*/}
}
DrawFractal(length,a,x,y,x1,y1,type,i11);
i11 = i011; x = x1; y = y1; type = 1; }
}
}

```

3. В этом фрагмента показана процедура рисования фрактала. Пеано

```

void CLab_4View::DrawFractal(float length,float a0, float x,float y,float
x1,float y1,int typeLine, int& i1)
{ int i0;
  float a; //знак приращения
  if (i1 == 1)
  { rawLines(x,y,x1,y1,width_Line,red_Line,green_Line,blue_Line); }
  else
  { length = length/3; i1--; i0 = i1; a = a0;
    Draw3Lines(length,a, x,y,1,typeLine,i1);
    i1 = i0; a = (-1)*a0;
    Draw3Lines(length,a, x,y,4,typeLine,i1);
    i1 = i0; a = a0;
    Draw3Lines(length,a, x,y,7,typeLine,i1);
  }
}

```

Задание 55

IFS-фракталы "Ветка"

