III International Scientific Practical Conference of graduate and postgraduate students, lecturers «APPLIED ISSUES OF EXACT SCIENCES»
01-02 November 2019, Armavir

ИССЛЕДОВАНИЕ МЕТОДОВ ПРОГРАММИРОВАНИЯ ФРАКТАЛЬНЫХ ИЗОБРАЖЕНИЙ

И.А. Груднов[1)], Л.А. Горовенко[2)]

1) студент Армавирского механико–технологического института (филиала) ФГБОУ ВО «Кубанский государственный технологический университет», г. Армавир, Россия, fidjerald63@mail.ru
2) к.т.н., доцент Армавирского механико–технологического института (филиала) ФГБОУ ВО «Кубанский государственный технологический университет», г. Армавир, Россия, lgorovenko@mail.ru

Аннотация. В данной статье рассматривались методы программирования фрактальных изображений. Приведены результаты моделирования фрактального изображения без использования рекурсивного вызова.
Ключевые слова: фракталы, программирование, программирование повторяющихся изображений.

RESEARCH OF METHODS OF FRACTAL IMAGES PROGRAMMING

Ilya A. Grudnov[1)], Lyubov A. Gorovenko[2)]

1) the student Armavir Institute of Mechanics and Technology (branch) of Federal State Budgetary Institution of Higher Education “Kuban State Technological University”, city of Armavir, Russia, fidjerald63@mail.ru
2) Ph. D., associate Professor, Armavir Institute of Mechanics and Technology (branch) of Federal State Budgetary Institution of Higher Education “Kuban State Technological University”, city of Armavir, Russia, lgorovenko@mail.ru

Abstract. In this article the methods of programming fractal images were considered. The results of fractal image modeling without recursive call are presented.
Key words: fractals, programming, programming of repetitive images.

Актуальность: достаточно сложно просчитать в уме многочисленные вызовы рекурсивной функции, так чтобы можно было представить график этой функции с целью проанализировать все зависимости между параметрами этой функции. Без применения ЭВМ построить такой график представляется весьма затруднительным, поэтому

исследование методов программирования фрактальных изображений является актуальным.

Практическое применение: фрактальны изображения используют в архитектуре, а также при создании оригинальных и сложных дизайнерских объектов, в науке построение фрактальных изображений используется для исследования функций. Графики функций, порождающих фрактальные изображения также полезны при некоторых физических расчетах и в конкретных технических решениях.

Рассмотрим некоторые принципы построения функций, порождающих фрактальные изображения.

Один из способов – это рекурсия. Для построения рекурсивного изображения программа должна обладать как рекурсивными ветвями, так и терминальными, содержащими условия завершения процедур. Рекурсивные ветви выполняют заданные процессы, приостанавливаясь, вызывают следующий рекурсивный процесс, выполняют его, запоминая предыдущий, и так последовательно производя рекурсивный вызов, пока не будет достигнута терминальная (завершающая) ветвь. И только после этого происходит возврат на уровень выполнения предыдущей функции.

Именно поэтому при создании рекурсивной программы необходимо заранее продумать расположение в ней терминальных ветвей, т.е. тех ветвей, при выполнении которых функция возвращает значение, но не вызывает рекурсивный процесс.

Второй способ построения фрактальных изображений – организация цикла с пересчётом координат. Именно этот способ был использован нами при построении описанных ниже изображений

Для реализации задачи построения фрактального изображения нами была выбрана среда программирования ABC Pascal, которая реализует поддержку графического режима посредством включённого в неё модуля GraphABC. В качестве реализуемой задачи мы выбрали построение изображения «Дерава Пифагора».

Мы установили условия выполнения для цикла, чтобы определить количество шагов цикла, т. е. глубину повторения рисунка, от которой будет зависеть степень «разрастания» рекурсивного изображения.

Это можно осуществить ограничением по длине линий либо введением новой переменной или константы.

Итак, опишем алгоритм, в соответствии с которым нами была написана программа построения дерева Пифагора:

Инициализация начальных значений параметров осуществляется в конце программы.

Перед телом цикла:

1) Объявим константу Const max = …

Будем использовать её в дальнейшем в качестве ограничения шагов рекурсии, так что вводим вместо «…» нужное нам значение.

2) Инициализируем процедуру построения линий **procedure** LineTo1(x,y:Integer;l,u:Real), которая реализует перерасчёт координат и прорисовку линий по заданным координатам процедурой

```
Line(x,y,Round(x + l * cos(u)),Round(y - l * sin(u))).
```

Координаты конечной точки при этом вычисляются по формуле:

```
x = x + l * cos(u); y = y – l * sin(u),
```

Теперь приступим к циклу

Условием выполнения цикла будет служить условие того, что длина линии на достигла ограничения по константе max, т.е. выполняется условие l>max. Это значит, что цикл будет выполняться, пока длина линий (ветвей дерева) будет больше константы.

В теле цикла присваиваем длине выражение, уменьшающее её значение, данная строка будет задавать плотность фрактального изображения (l := l*0.7;). Обозначаем цвет линий, например, голубой: SetPenColor(clBlue).

Применяем:

- процедуру построения линий (LineTo1(x,y,l,u);) ;
- изменение координат по указанным ранее принципам ;
- изображение линий с новой длиной и началом отрезка в изменяемых координатах, а также по заданному углу, изменение которого указываем здесь же:

Draw(x,y,l,u + pi/7);
Draw(x,y,l,u - pi/5);

При этом u – это наш начальный угол.

После тела цикла вводим начальные данные для наших процедур:

Draw(550, 640,180,pi/2);

Приведём код программы, которая у нас получилась

```
uses GraphABC;
Const
max = 4;
procedure LineTo1(x,y:Integer;l,u:Real);
Begin
Line(x,y,Round(x + l * cos(u)),Round(y - l * sin(u)));
End;
procedure Draw(x,y:Integer;l,u:Real);
begin
if l>max then
begin
l := l*0.7;
SetPenColor(clBlue);
```

```
LineTo1(x,y,l,u);
x := Round(x + l*cos(u));
y := Round(y - l*sin(u));
Draw(x,y,l,u + pi/7);
Draw(x,y,l,u - pi/5);
end;
end;
begin
SetWindowCaption('Derevo Piphagora');
SetWindowSize(900,600);
Draw(550, 640,180,pi/2);
end.
```

Результаты тестирования программы представлены на рисунке 1.

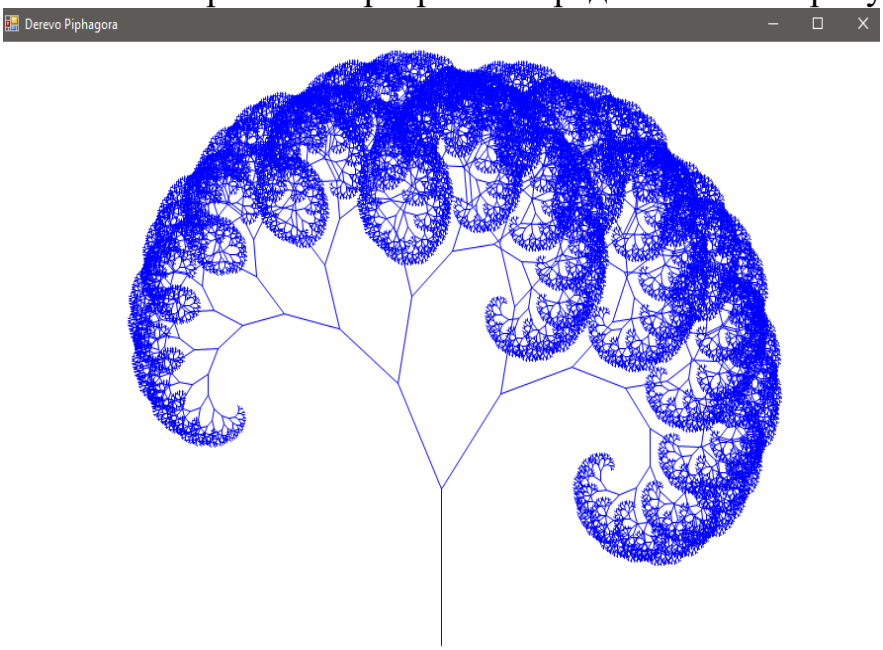


Рис.1 – Фрактальное изображение «Дерево Пифагора»

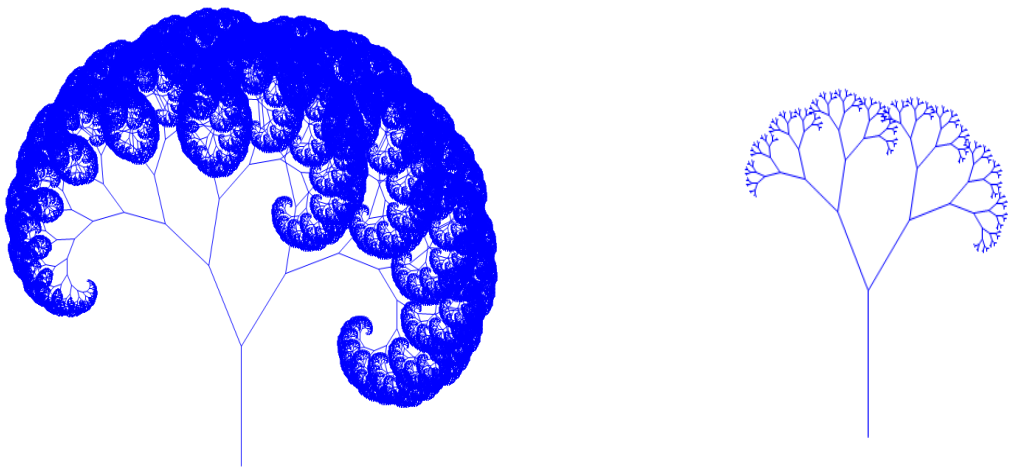
Мы провели исследование зависимости изображения от варьируемых параметров. При уменьшении константы увеличилось число шагов программы, а значит увеличилась плотность изображения в области крайних ветвей. Уменьшение коэффициента изменения длины линий привело к уменьшению плотности изображения и его размера. Изменяя угол в начальных данных, мы поворачиваем наше изображение. При ином значении изменения угла в строке Draw получим дерево, ветви которого будут расти под другим углом (рис. 2).

Аналогичным образом построен программный код, реализующий построение фрактального изображения «Папоротник».

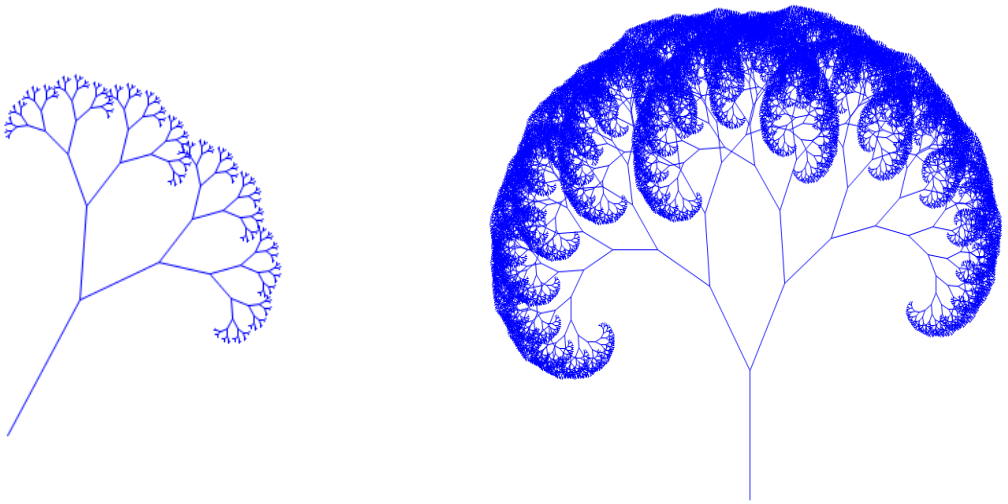
```
uses GraphABC;
```

```
const
    min = 0.01;
    procedure lineto1(x, y : Integer; l, u : real);
begin
    Line(x, y, Round(x + l * cos(u)), Round(y - l * sin(u)));
end;
 procedure Draw(x, y : Integer; l, u : real);
 begin
    if l > min then
    begin
        lineto1(x, y, l, u);
        x := Round(x + l * cos(u));
        y := Round(y - l * sin(u));
        Draw(x, y, l*0.4, u - 14*pi/30);
        Draw(x, y, l*0.4, u + 14*pi/30);
        Draw(x, y, l*0.7, u + pi/30);
    end;
end;
begin
     SetWindowCaption('Фракталы: Папоротник');
   SetWindowSize(800,600);
   ClearWindow;
    Draw(400, 560, 170, pi/2)
end.
```

Объявляем константу, которая будет условием окончания программы, инициализируем все необходимые процедуры построений и изменения координат по тем же принципам, что и в первой программе. Результаты тестирования программы с варьированием параметров представлены на рисунке 3.



а) б)



в) г)

Рис. 2 – Моделирование параметров изображения:
а) при уменьшении константы; б) при уменьшении коэффициента изменения длины; в) при изменяи угла в начальных данных; г) при ином значении изменения угла в строке Draw

В ходе проведённого исследования были получены следующие результаты:

1) изучена терминология программирования фрактальных и рекурсивных изображений.

2) изучены принципы построения рекурсивных изображений и фрактальных изображений без использования рекурсивного вызова.

3) проведено моделирование получаемого фрактального изображения с различными значениями параметров исходных данных.

Также было выяснено, что с базовыми знаниями графической среды и принципов построения рекурсивных изображений можно создать программы построения рекурсивного изображения. Поэтому стоит продолжать изучение фрактальных изображений и зависимостей их элементов в программе. Это позволит совершенствовать методы создания программ, реализующих фрактальную графику.

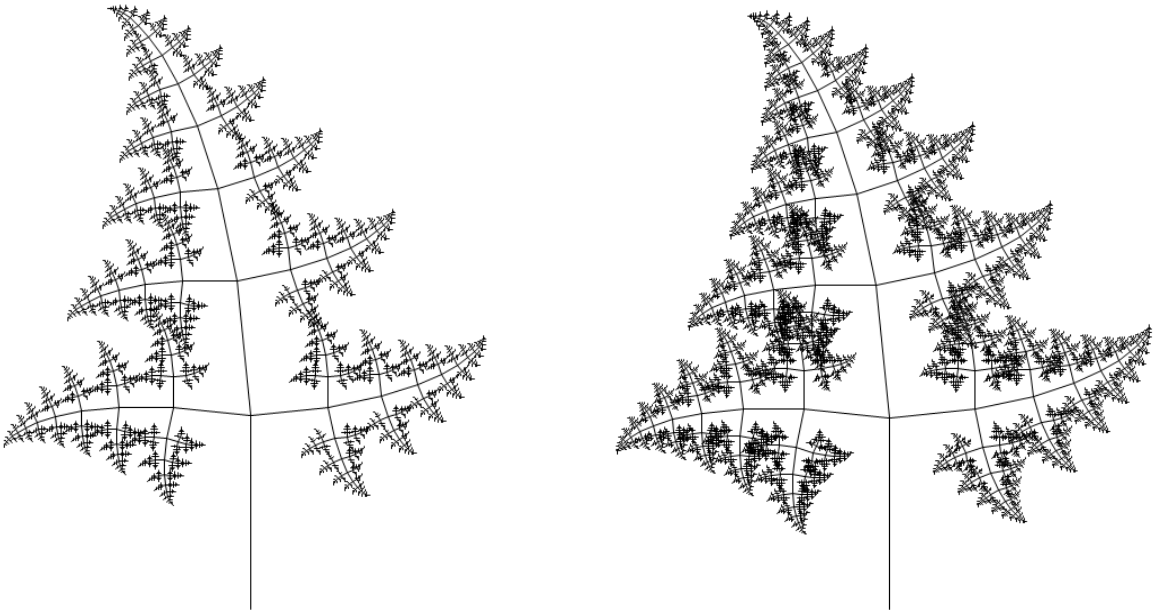


Рис.3 – Моделирование построения фрактального изображения «папоротник»

Список использованных источников:

1. Горовенко Л.А. Математические методы компьютерного моделирования физических процессов// Международный журнал экспериментального образования. Пенза: ИД «Академия естествознания», 2017. – №2. – с. 92–93.

2. Вахрушев А.И., Алексанян Г.А. Аппроксимация функций // Прикладные вопросы точных наук: Материалы II Международной научно-практической конференции студентов, аспирантов, преподавателей.- Армавир: РИО АГПУ, 2018. – С. 35-37.