МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования
«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Математики, Информатики и Технологий

КУРСОВАЯ РАБОТА
По Функциональному программированию
По теме «ФРАКТАЛЫ»
5 семестр

Студент:	Савченко И.В.
Группа:	2321-ДБ
Преподаватель:	Черкашин Е.А.
Подпись:	
Оценка:	
Дата:	26 декабря 2022 г.

Иркутск
2022

Содержание

Введение

Фрактал — множество, обладающее свойством самоподобия (объект, в точности или приближённо совпадающий с частью себя самого, то есть целое имеет ту же форму, что и одна или более частей). В математике под фракталами понимают множества точек в евклидовом пространстве, имеющие дробную метрическую размерность (в смысле Минковского или Хаусдорфа), либо метрическую размерность, отличную от топологической, поэтому их следует отличать от прочих геометрических фигур, ограниченных конечным числом звеньев. Самоподобные фигуры, повторяющиеся конечное число раз, называются предфракталами.

Фракталы известны уже век, хорошо изучены и имеют многочисленные приложения в жизни. Однако в основе этого явления лежит очень простая идея: бесконечное по красоте и разнообразию множество фигур можно получить из относительно простых конструкций при помощи всего двух операций – копирования и масштабирования.

1 Теоретическая часть

Основная идея фркаталов как геометрических фигур заключается в рекурсивном самоподобии. Из-за этой рекурсии подобные фигуры не сложно рисовать при помощи функциональных языков программирования, которые специализируются на рекурсивных функциях. В данной работе используется язык Haskell.

Идея создания проста — имеется функция, возвращающая фигуру, и которая принимает число — "уровень"фрактала, его глубину. Внутри этой фигуры создаются такие-же фигуры, уровнем ниже, рекурсивно. Если уровень фигуры ноль, то на этом детализацию можно остановить.

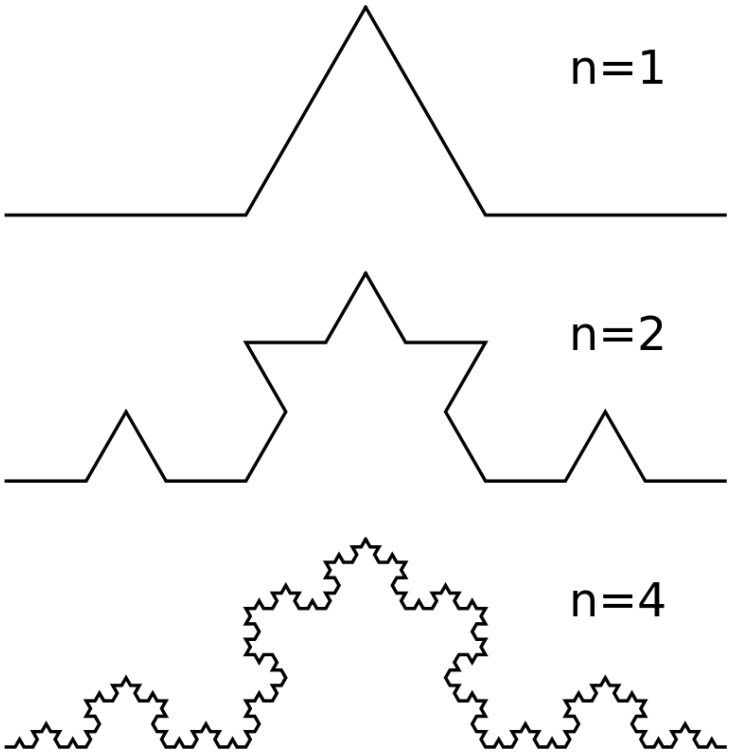

Рис. 1: Пример процедурного создания кривой Коха

2 Графическая библиотека "gloss"

Для создания процедурно созданных рисунков мы будем использовать библиотеку gloss, которая предоставляет нам простой абстрактный интерфейс для работы с библиотекой OpenGL, который очень сильно облегчит нам создание наших непростых фигур.

Данная библиотека предоставляет нам функции, которые возвращают фигуры, позволяют нам их перемещать, изменять размер и совмещать, а также создавать программируемые анимации.

После создания объекта фигуры, мы передаём её в функцию создания окна, где мы увидим её на виртуальном полотне.

3 Реализация

Мы ознакомились с теоретической частью, поэтому можно сразу приступить к реализации. Следующая программа создаст множество окружностей, по обе стороны которых окружности будут повторяться.

```
import Graphics.Gloss

mywindow = InWindow "Circles" (600, 600) (20, 20)

mycirc :: Int -> Picture
mycirc 0 = Blank
mycirc n = Pictures [circ_c, circ_l, circ_r]
    where x = 200
          circ_c = color black $ circle x
          circ_small
              = color black
              $ scale 0.45 0.45
              $ mycirc (n-1)
          circ_r = translate   x  0 $ circ_small
          circ_l = translate (-x) 0 $ circ_small

myfigure = Pictures [line [(-1000,0),(1000,0)], mycirc 6]

main = display mywindow white myfigure
```

Как видно из кода, за создание фрактала отвечает функция "mycirc которая рисует центральную окружность, а за тем слева и справа рекурсивно вызывает саму себя, опускаясь в "уровне".

При запуске программы, мы наблюдаем следующую картину (Рис. 2).

Как мы увидели, создание несложных фракталов и узоров не является сложной задачей. Мы можем нарисовать другой фрактал, например дерево. Библиотека gloss также позволяет нам его разукрасить в различные цвета.

```
import Graphics.Gloss

mywindow = InWindow "Tree" (600, 600) (20, 20)

brown = makeColorI 119 67 21 255
```

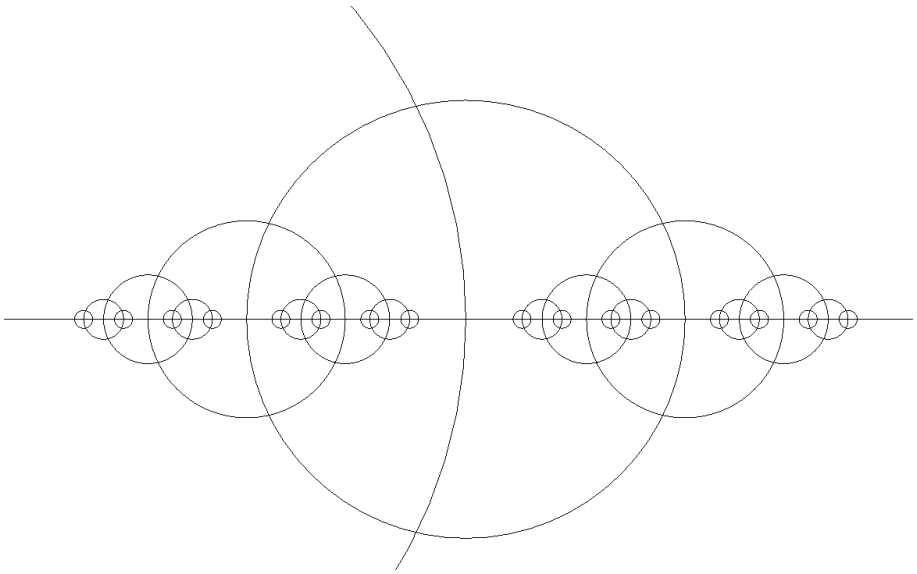

Рис. 2: Простой фрактал, созданный из окружностей

```
mytree :: Int -> Picture
mytree 0 = Blank
mytree n = Pictures [branch_m, branch_l, branch_r]
    where angle = 45
          colprop = fromIntegral (n-1) / 12.0 :: Float
          mycolor = mixColors colprop (1.0 - colprop) brown green
          myline = line [(0,   0),
                         (0, 100)]
          branch_m
              = color mycolor
              $ myline
          smallerbranch
              = scale 0.6 0.6
              $ mytree (n-1)
          branch_l
              = translate 0 100
              $ rotate (-angle)
              $ smallerbranch
          branch_r
              = translate 0 100
              $ rotate angle
              $ smallerbranch


myfigure = mytree 8

main = display mywindow white myfigure
```

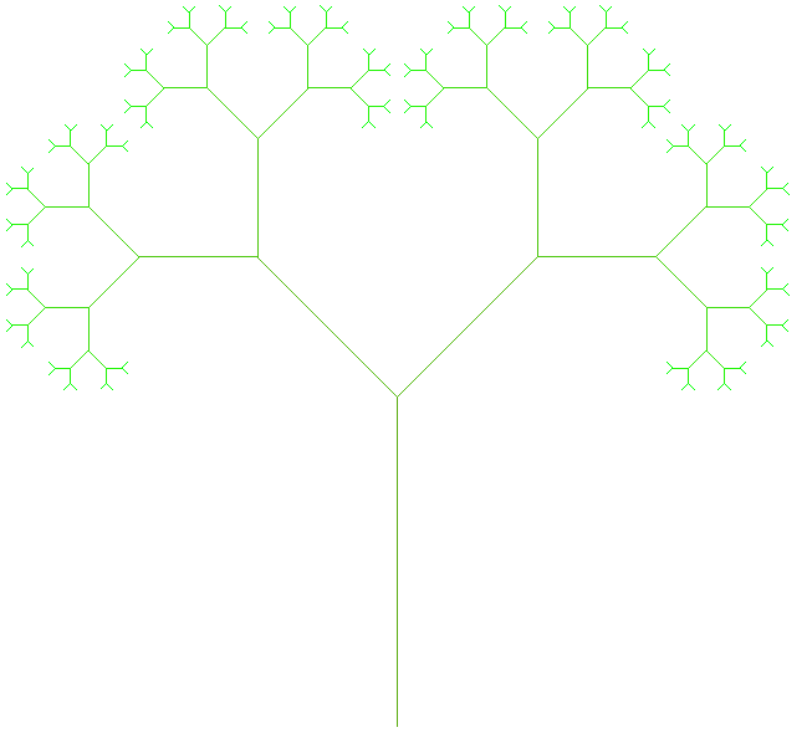

Рис. 3: Фрактал в виде симметричного дерева

Причём ничего не останавливает нас также нарушить симметрию, создав при этом более интересные фигуры. Если левые и правые ветви будут "расти"под разными углами, и если их размеры не будут совпадать, что мы делаем в следующем коде, то мы получим результат, как на рис. 4.

```
import Graphics.Gloss

mywindow = InWindow "Nicer Tree" (600, 600) (20, 20)

brown = makeColorI 119 67 21 255

mytree' :: Int -> Picture
mytree' 0 = Blank
mytree' n = Pictures [branch_m, branch_l, branch_r]
    where angle = 40
          colprop = fromIntegral (n-1) / 7.0 :: Float
          mycolor = mixColors colprop (1.0 - colprop) brown green
          myline = line [(0,   0),
                         (0, 100)]
          branch_m
              = color mycolor
              $ myline
```

```
            smallerbranch = mytree' (n-1)
            branch_l
                = translate 0 100
                $ scale 0.65 0.65
                $ rotate (20 - angle)
                $ smallerbranch
            branch_r
                = translate 0 100
                $ scale 0.5 0.5
                $ rotate angle
                $ smallerbranch

myfigure = mytree' 8

main = display mywindow white myfigure
```

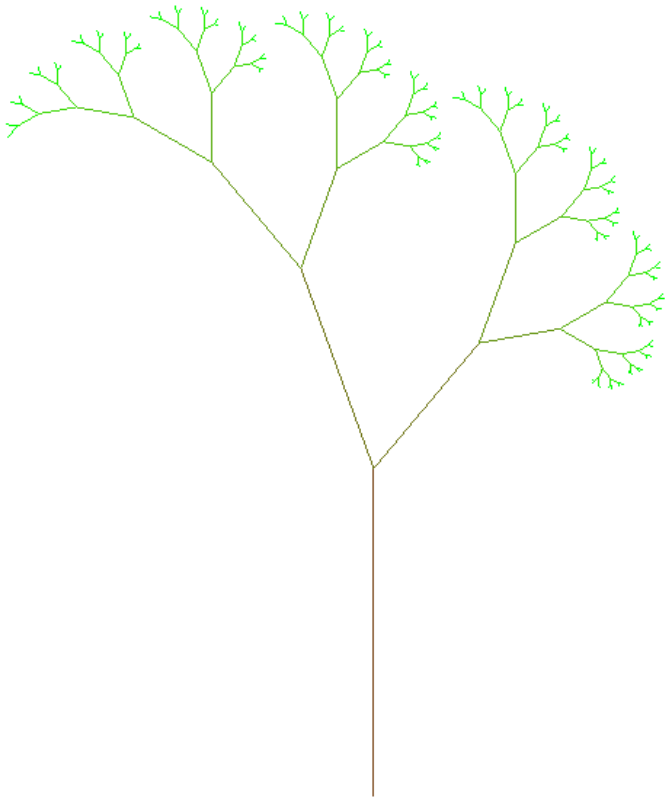

Рис. 4: Фрактал – дерево, с нарушенной симметрией

Как мы видим, функциональные языки программирования позволяют нам легко и быстро создавать самоподобные фигуры, узоры и фракталы, при этом имея очень лаконичные и простые в понимании программы.

Заключение

Нами были изучены инструменты функционального языка программирования Haskell, которые позволяют нам строить различные изображения и фигуры, в частности фракталы и рекурсивные самоподобные узоры. Для этого нам понадобилось изучить работу с менеджером пакетов Cabal, для установки библиотеки для работы с графикой gloss, что так-же даёт важные навки по работе с подобными пакетами. Также подобное упражнение отлично и наглядно показывает способности функциональных языков к реализации рекурсивных структур и просто решении задач через создание рекурсивных функций.

Список литературы

[1] Graphics.Gloss — hackage.haskell.org [Электронный ресурс]
URL: https://hackage.haskell.org/package/gloss-1.13.2.2/docs/Graphics-Gloss.html
(дата обращения: 26.12.2022)

[2] Your First Haskell Application (with Gloss) [Электронный ресурс]
URL: https://andrew.gibiansky.com/blog/haskell/haskell-gloss/
(дата обращения: 26.12.2022)

[3] Gloss [Электронный ресурс]
URL: http://gloss.ouroborus.net
(дата обращения: 26.12.2022)

[4] Spirals, Snowflakes & Trees: Recursion in Pictures [Электронный ресурс]
URL: http://learn.hfm.io/fractals.html
(дата обращения: 26.12.2022)

[5] Как устроены фракталы: бесконечность и красота математики [Электронный ресурс]
URL: https://www.techinsider.ru/science/8906-krasota-povtora-fraktaly/
(дата обращения: 26.12.2022)