Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования
«Иркутский государственный университет»
(ФГБОУ ВО «ИГУ»)

Институт математики и информационных технологий

Кафедра вычислительной математики и оптимизации

КУРСОВАЯ РАБОТА

по направлению «Прикладная математика и информатика»

профиль «Математические методы и информационные технологии»

Решение фракталов на языке Haskell

Студента 3 курса очного отделения
группы 02321-ДБ
Старовойтова Владимира Андреевича

Руководитель:
к. ф.-м. н., доцент
_____Черкашин Е. А.

Иркутск – 2022

Содержание

ВВЕДЕНИЕ

Haskell – функциональный язык программирования, сильно отличающийся от императивных и смешанных языков разработки. Важной особенность языка является поддержка ленивых вычислений, что позволяет ускорить работу программы. В Haskell аргументы функции вычисляются только тогда, когда действительно требуются для вычисления. Причем ленивые вычисления выполняются без вмешательства самого программиста.

На данный момент Haskell является актуальным языком программирования. Он, например, применяется в финансовом секторе – для разработки собственных инструментов в крупных банках и других компаниях. В дополнение к этому Haskell часто используется для обработки текстов, синтаксического анализа, а также веб-разработки.

Целью данной курсовой работы является ознакомление с данным языком посредством решения задач, связанных визуализацией фракталов, укрепление и развитие навыков программирования.

Фракталы нашли применение в физике (моделирование сложных процессов и материалов), биологии (моделирование популяций, описание сложных, ветвящихся структур), технике (фрактальные антенны), экономике. Существуют алгоритмы сжатия с помощью фракталов. В компьютерной графике фракталы используются для построения изображение природных объектов – растений, ландшафтов, поверхности морей и т.д. Открытие фракталов было открытием новой эстетики искусства, науки и математики, а также революцией в человеческом восприятии мира. В данной курсовой работе будут изображены фракталы при помощи языка Haskell. Данный язык является гибким для множества математических задач, благодаря обширным количеством библиотек, а также своей производительностью.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Понятие “Фрактал”

Фрактал — множество, обладающее свойством самоподобия (объект, в точности или приближённо совпадающий с частью себя самого, то есть целое имеет туже форму, что и одна или более частей).

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 само подобных структур, которыми он занимался. Рождение фрактальной геометрии принято связывать свыходом 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.

Задание фрактала в любом языке программирования производится с помощью рекурсивно вызываемых функций (то есть функций, вызывающих сами себя) для некоторых примитивов (отрезков, эллипсов), с повышением степени каждый из полученных в результате выполнения функции примитивов заменяется вызовом той жесамой функции, но уже с другими параметрами.

Для изображения реализованных в программном коде на языке программирования Haskell фракталов будет использоваться библиотека Gloss, котораядает удобный интерфейс для использования спецификации OpenGL для рисования векторной графики на виртуальном полотне.

1.2 Применение фракталов

Естественные науки.

В физике фракталы естественным образом возникают при моделировании нелинейных процессов, таких как турбулентное течение жидкости, сложные процессы диффузии-адсорбции, пламя, облака и тому подобное. Фракталы используются при моделировании пористых материалов, например, в нефтехимии. В биологии они применяются для моделирования популяций и для описания систем внутренних органов (система кровеносных сосудов). После создания кривой Коха было предложено использовать её при вычислении протяжённости береговой линии.

Радиотехника

Использование фрактальной геометрии при проектировании антенных устройств, преимуществом таких антенн является многодиапазонность и сравнительная широкополосность.

Сжатие изображений

Существуют алгоритмы сжатия изображения с помощью фракталов. Они основаны на идее о том, что вместо самого изображения можно хранить сжимающее отображение, для которого это изображение (или некоторое близкое к нему) является неподвижной точкой.

Компьютерная графика

Фракталы широко применяются в компьютерной графике для построения изображений природных объектов, таких как деревья, кусты, горные ландшафты, поверхности морей и так далее. Существует множество программ, служащих для генерации фрактальных изображений.

Децентрализованные сети

Система назначения IP-адресов в сети Netsukuku использует принцип фрактального сжатия информации для компактного сохранения информации об узлах сети. Каждый узел сети Netsukuku хранит всего 4 Кб информации о состоянии соседних узлов, при этом любой новый узел подключается к общей сети без необходимости в центральном регулировании раздачи IP-адресов, что, например, характерно для сети Интернет. Таким образом, принцип фрактального сжатия информации гарантирует полностью децентрализованную, а следовательно, максимально устойчивую работу всей сети.

ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Треугольник Серпинского

Треугольник Серпинского — фрактал, один из двумерных аналогов множества Кантора, математическое описание которого опубликовал польский математик Вацлав Серпинский в 1915 году. Также известен как «салфетка» Серпинского.

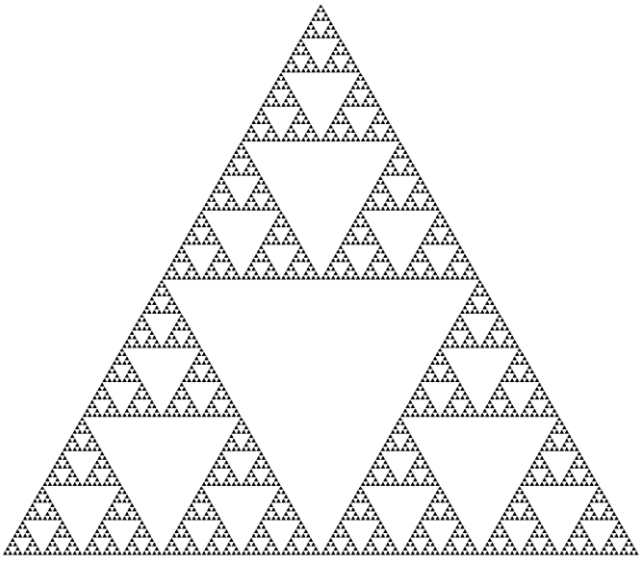

Рис. 1. Треугольник Серпинского

Это один из фракталов, с которыми экспериментировал Мандельброт, когда разрабатывал концепции фрактальных размерностей и итераций. Треугольники, сформированные соединением средних точек большего треугольника, вырезаны из главного треугольника, образовывая треугольник, с большим количеством дырочек. В этом случае инициатор - большой треугольник, а шаблон - операция вырезания треугольников, подобных большему. Так же можно получить и трехмерную версию треугольника, используя обыкновенный тетраэдр и вырезая маленькие тетраэдры.

Свойства:

- Треугольник Серпинского состоит из 3 одинаковых частей, коэффициент подобия 1/2.
- Треугольник Серпинского замкнут.
- Треугольник Серпинского имеет топологическую размерность 1.
- Важным свойством треугольника Серпинского является его самоподобие — ведь он состоит из трёх своих копий, уменьшенных в два раза (это части треугольника

Серпинского, содержащиеся в маленьких треугольниках, примыкающих к углам).

- Треугольник Серпинского имеет промежуточную (то есть нецелую) Хаусдорфову размерность ln3/ln2 = 1.584962501.

2.2 Реализация треугольника Серпинского

Разработанная программа рассматривает алгоритм реализации фрактала, представляющего собой треугольник Серпинского. Результатом работы является вывод данной фигуры на экран в виде 2D рисунка.

Использованные модули:

1. Graphics.UI.GLUT — это программный интерфейс для написания программ OpenGL, независимых от оконной системы.
2. Control.Concurrent — это общий интерфейс для набора полезных абстракций параллелизма.
3. Data.Fixed (mod') — этот модуль определяет тип «Fixed» для арифметики с фиксированной точностью, также содержит обобщения mod для работы с любым экземпляром Real.

Запуск программы производится из файла Main.hs, точка входа – функция main :: IO ().

В основном файле Main.hs реализована визуализация фрактала. В Fractals.hs реализованы функции, на которых основан фрактал.

Реализация Main.hs

Обозначение функций:

1. fWindow – инициализация окна
2. fCallbacks – инициализация обратного вызова
3. refresh – обновление окна
4. display – отображение нашего фрактала

```
module Main where


import Data.IORef
import Fractals
import Graphics.UI.GLUT
import Control.Concurrent
```

```
main :: IO ()

main = do

    fWindow

    fCallbacks

    mainLoop

- - Создание окна
fWindow  ::  IO(Window)

fWindow = do

    initialWindowSize  $=  Size 800  800

    createWindow "Fractal treangle"

- - Обратный вызов
fCallbacks  ::  IO()

fCallbacks = do

    idleCallback $= Just refresh

    displayCallback $= display

-- Обновление окна
refresh :: IO()

refresh = do

    threadDelay 10000 – Задержка потока для обновления окна

-- Отображение
display :: IO()

display = do

    let thirdOfACircle = pi * 2 / 3

        sinThirdOfACircle = sin thirdOfACircle

        cosThirdOfACircle  = cos  thirdOfACircle


    clear [ColorBuffer]

  drawFractal (1, sinThirdOfACircle, cosThirdOfACircle, -sinThirdOfACircle) 6

  flush
```

Листинг 1

Реализация Fractals.hs

Обозначение функций:

1. drawFractal – рисование фрактала
2. drawTriangle – рисование треугольника
3. subdivide – показывает ступени фрактала
4. toVertices – определяет вершины
5. subTriangles – треугольники, входящие в фрактал
6. drawingColor – цвет фрактала по HSV
7. azimuth – азимут точек

```
-- Установка типов данных, которые нужны для работы
type Point = (GLfloat, GLfloat) -- (x, y) расположение начала рисовки

type Area = (GLfloat, GLfloat, GLfloat, GLfloat) -- (top, right, bottom, left) область рисовки определенной ветки

-- GLfloat – это тип данных, используемый в OpenGL, имеющий тип float

-- Рисование фрактала
drawFractal :: Area -> Int -> IO()

drawFractal area 0 = drawTriangle area

drawFractal area iteration = subdivide area iteration

-- Рисование треугольника

drawTriangle :: Area -> IO()

drawTriangle area = do

    let vertices = toVertices area

        setColorAndDraw vertexPosition = do

        color $ drawingColor vertexPosition

        vertex vertexPosition

    renderPrimitive Triangles $ do

         mapM_ setColorAndDraw vertices

-- Ступени фрактала

subdivide :: Area -> Int -> IO()

subdivide area iteration = do

  let iteration' = iteration - 1

  mapM_ (\pos -> drawFractal pos iteration') $ subTriangles area
```

```
-- Определение вершин
toVertices :: Area -> [Vertex2 GLfloat]
toVertices (top, right, bottom, left) = [topVertex, leftVertex, rightVertex]
where middleX = (left + right) / 2
        topVertex = Vertex2 middleX top
        leftVertex = Vertex2 left bottom
        rightVertex = Vertex2 right bottom
-- Треугольники, входящие в фрактал
subTriangles :: Area -> [Area]
subTriangles (top, right, bottom, left) = [topSubTriangle, leftSubTriangle, rightSubTriangle]
    where middleX = (left + right) / 2
          middleY = (top + bottom) / 2
          quarterX = (left + middleX) / 2
          threeQuarterX = (right + middleX) / 2

          topSubTriangle = (top, threeQuarterX, middleY, quarterX)
          leftSubTriangle = (middleY, middleX, bottom, left)
          rightSubTriangle = (middleY, right, bottom, middleX)

-- Цвет фрактала по HSV
drawingColor :: Vertex2 GLfloat -> Color3 GLfloat
drawingColor (Vertex2 x y) = Color3 red green blue
   where distanceFromCenter = sqrt(x*x + y*y)

          hue = azimuth (x, y)
          saturation = distanceFromCenter
          grayness = 1 - saturation

          secondaryColor = let intensity = realToFrac hue / sixthOfACircle
                           in saturation * realToFrac (1 - (abs ((mod' intensity 2) - 1)))

          (r, g, b) | hue < sixthOfACircle * 1 = (saturation, secondaryColor, 0)
                    | hue < sixthOfACircle * 2 = (secondaryColor, saturation, 0)
                    | hue < sixthOfACircle * 3 = (0, saturation, secondaryColor)
                    | hue < sixthOfACircle * 4 = (0, secondaryColor, saturation)
                    | hue < sixthOfACircle * 5 = (secondaryColor, 0, saturation)
                    | hue < sixthOfACircle * 6 = (saturation, 0, secondaryColor)
                    | otherwise = (1, 1, 1)

            red = r + grayness
            green = g + grayness
            blue = b + grayness
```

```
-- Азимут точек
azimuth :: Point -> GLfloat
azimuth (x, y) | pointIsAtTopRightQuarter = quarterOfACircle * 0 + acos((y2 + hypotenuse2
- x2) / (2 * (abs y) * hypotenuse))
               | pointIsAtBottomRightQuarter = quarterOfACircle * 1 + acos((x2 + hypotenuse2
- y2) / (2 * (abs x) * hypotenuse))
               | pointIsAtBottomLeftQuarter = quarterOfACircle * 2 + acos((y2 + hypotenuse2
- x2) / (2 * (abs y) * hypotenuse))
               | pointIsAtTopLeftQuarter = quarterOfACircle * 3 + acos((x2 + hypotenuse2 - y2)
/ (2 * (abs x) * hypotenuse))

        where quarterOfACircle = pi / 2
              hypotenuse = sqrt(x*x + y*y)

              hypotenuse2 = hypotenuse*hypotenuse
              x2 = x*x
              y2 = y*y

              pointIsAtTopRightQuarter = x >= 0 && y > 0
              pointIsAtBottomRightQuarter = x >= 0 && y <= 0
              pointIsAtBottomLeftQuarter = x < 0 && y <= 0
              pointIsAtTopLeftQuarter = x < 0 && y > 0

sixthOfACircle = pi / 3
```

Листинг 2

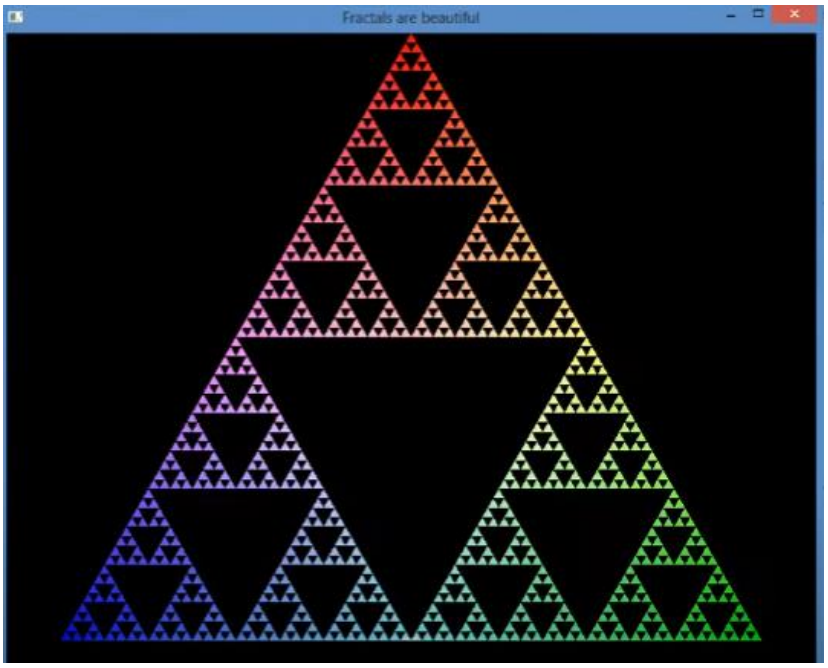

Рис. 2. Результат работы - Треугольник Серпинского

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы был рассмотрен инструментарий языка программирования Haskell для визуализации фракталов. Haskell предоставляет удобные условия для реализации подобных задач.

С помощью функционального языка программирования Haskell была написана программа, реализующая фрактал “Треугольник Серпинского”. Показана работа с визуализацией изображений с помощью сторонней библиотеки GLUT, рекурсивными вызовами функции.

Таким образом были закреплены и показаны навыки работы с данным языком программирования.

СПИСОК ИЗУЧЕННОЙ ЛИТЕРАТУРЫ

1. Миран Липовача Изучай Haskell во имя добра! / Пер. с англ. Леушина Д., Синицына А., Арсанукаева Я.– М.: ДМК Пресс, 2012. – 490 с.: ил. ISBN 978-5-94074-749-9
2. Г.М. Сергиевский, Н.Г. Волченков. Функциональное и логическое программирование. – М.: Академия, 2010. – 320 с.: ил. ISBN: 978-5-7695-6433-8
3. С.В. Микони. Дискретная математика для бакалавра. Множества, отношения, функции, графы. – СПб.: Лань, 2013. – 192 с.: ил. ISBN: 978-5-8114-1386-7
4. Уилл Курт Программируй на Haskell / пер. с англ. Я. О. Касюлевича, А. А. Романовского и С. Д. Степаненко; под ред. В. Н. Брагилевского. – М.: ДМК Пресс, 2019. — 648 с.: ил. ISBN 978-5-97060-694-0
5. Haskell.org: сайт – 1996. – URL : https://www.haskell.org/
6. Hackage.haskell.org: сайт – 1996. – URL: https://hackage.haskell.org/
7. Wikipedia.org: сайт – 2001. – URL: https://ru.wikipedia.org/