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»

Построение фракталов на языке Haskell

Студента 3 курса очного отделения
группы 02321-ДБ
Гредасова Руслана Сергеевича

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.

1. НЕМНОГО О ФРАКТАЛАХ И ГРАФИКЕ В HASKELL

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).

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 же самой функции, но уже с другими параметрами.

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

2. ПОСТРОЕНИЕ НЕКОТОРЫХ ФРАКТАЛОВ

1. Визуализация следующего фрактала:

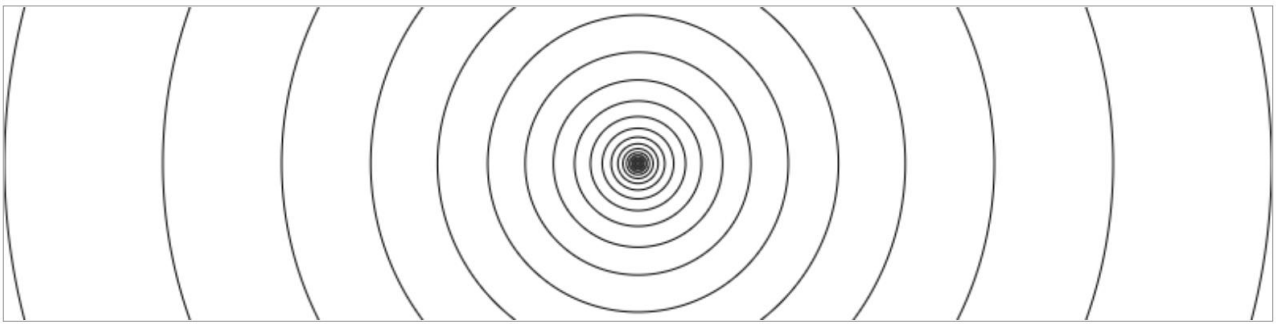

Рис. 1. Рекурсивные круги

На Рисунке 1 изображено простейшее представление самоповторяющейся фигуры – множество кругов, включенных друг в друге. Для изображения этой фигуры будет использована функция simpleCircle, которая имеет единственный параметр n, обозначающий глубину прорисовки фрактала. Данная функция вызывает себя множество раз, при этом уменьшая диаметр отображаемых кругов относительно внешних с масштабом 6:10. В результате получается серия кругов, каждый из которых нарисован внутри предыдущего круга.

Этот пример с кругами довольно тривиален; его можно легко достичь с помощью простой итерации. Однако для сценариев, в которых функция вызывает себя более одного раза, рекурсия становится удивительно элегантной.

```
import Graphics.Gloss

main = do
animate (InWindow "Fractal" (800, 800) (20, 20)) white frame

frame :: Float -> Picture
frame _ =
Color black
$ Scale 100 100
$ simpleCircle 15

simpleCircle :: Int -> Picture
simpleCircle 0 = Blank
simpleCircle n = Pictures [ circle 1, Scale 0.6 0.6 $ simpleCircle
(n-1)]
```

Листинг 1

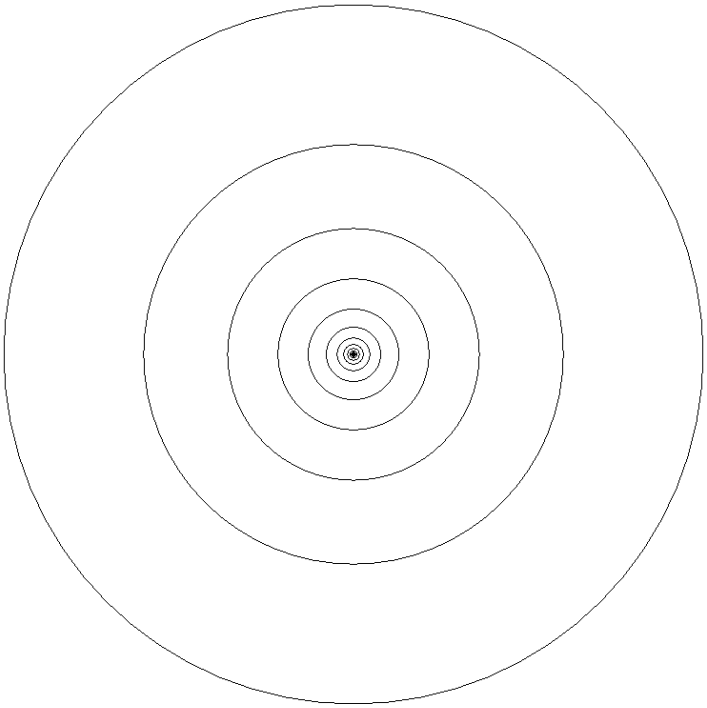

Рис. 2. Результат работы функции simpleCircle для n = 15

2. Визуализация следующего фрактала:

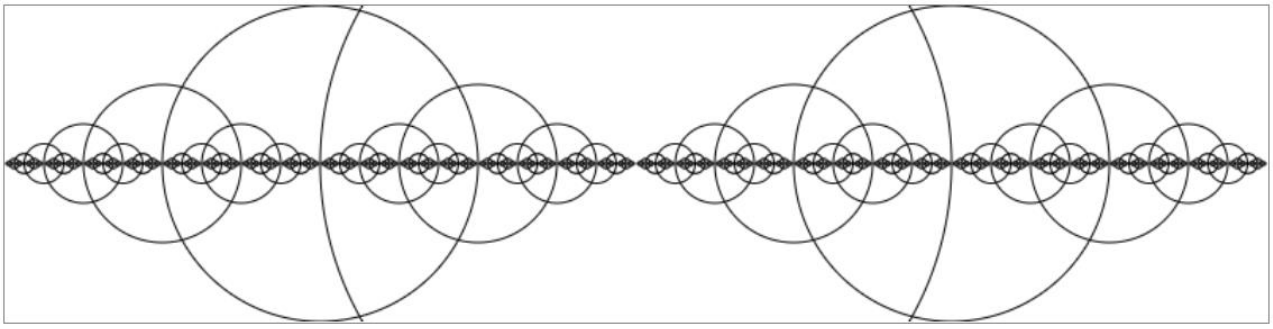

Рис. 3. Усложненный фрактал

Следующий фрактал немного сложнее – здесь для каждого круга рисуется еще два круга, находящихся слева и справа от него, каждый из которых в два раза меньше порождающего. Для отрисовывания малых кругов используется функция Translate, которая перемещает фигуру в пространстве. В данном случае ей передаются параметры 1 (и -1 соответственно) и 0, где первое число – смещение по оси *x*, а второе – по *y*. Для достижения требуемого расположения меньших кругов смещение должно быть равно по модулю радиусу большего круга (в данном случае радиус равен 1).

```
import Graphics.Gloss

main = do
    animate (InWindow "Fractal" (800, 800) (20, 20)) white frame
```

```
frame :: Float -> Picture
frame _ =
    Color black
    $ Scale 100 100
    $ doubleCircle 8

doubleCircle:: Int -> Picture
doubleCircle 0 = Blank
doubleCircle n = Pictures [circle 1, circ1, circ2]
    where
        circ1 = Translate 1 0 $ circNm
        circ2 = Translate -1 0 $ circNm

        circNm =
            Pictures
            [circle 0.5,
            Scale (0.5) (0.5) $ doubleCircle (n-1)]
```

Листинг 2

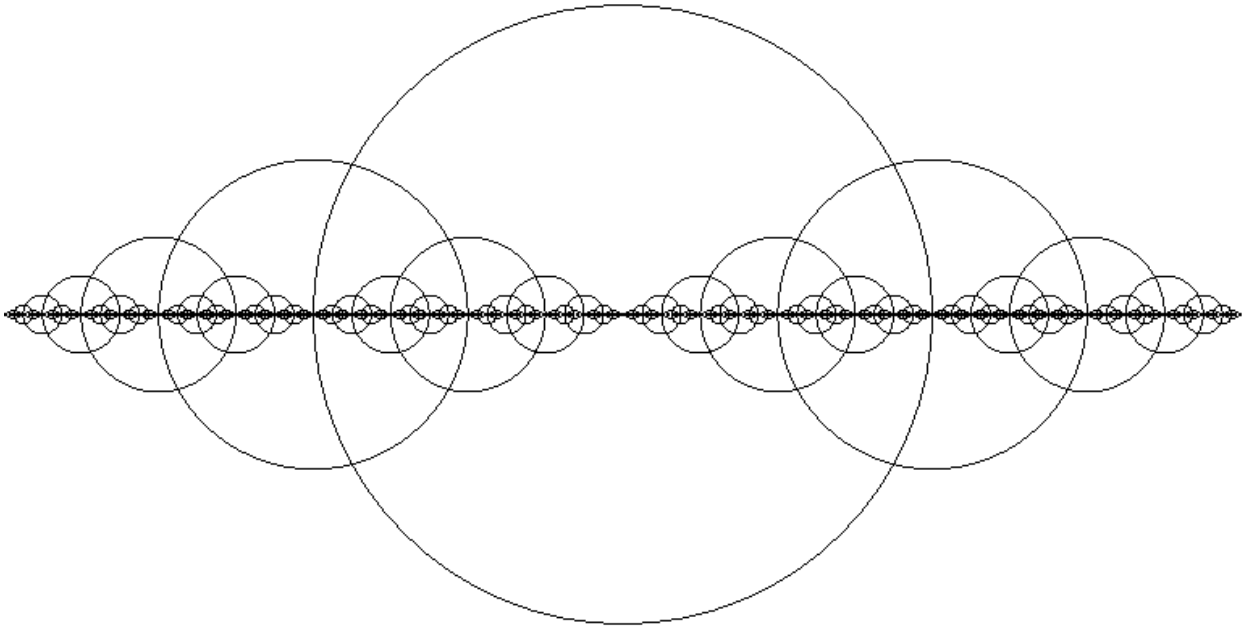

Рис. 4. Результат работы функции doubleCircle для n = 8

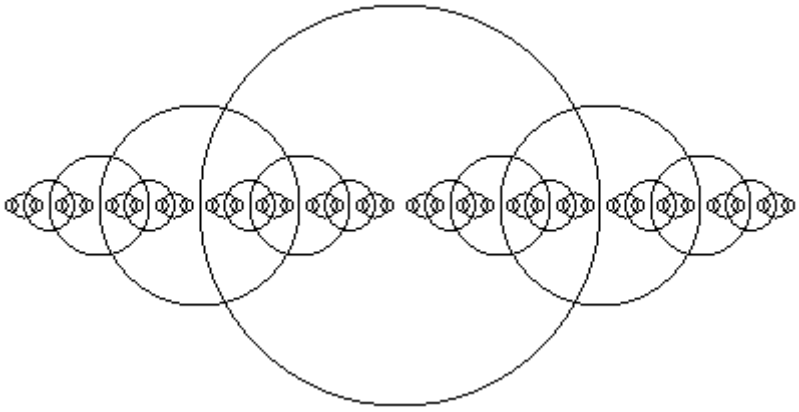

Рис. 5. Результат работы функции doubleCircle для n = 5

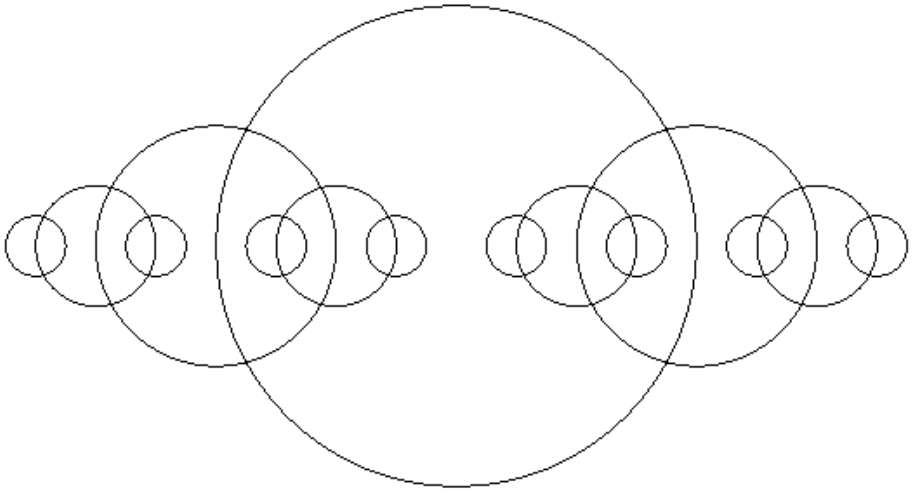

Рис. 6. Результат работы функции doubleCircle для n = 3

Для наглядности работы были приведены результаты работы алгоритма для разных n.

3. Визуализация следующего фрактала:

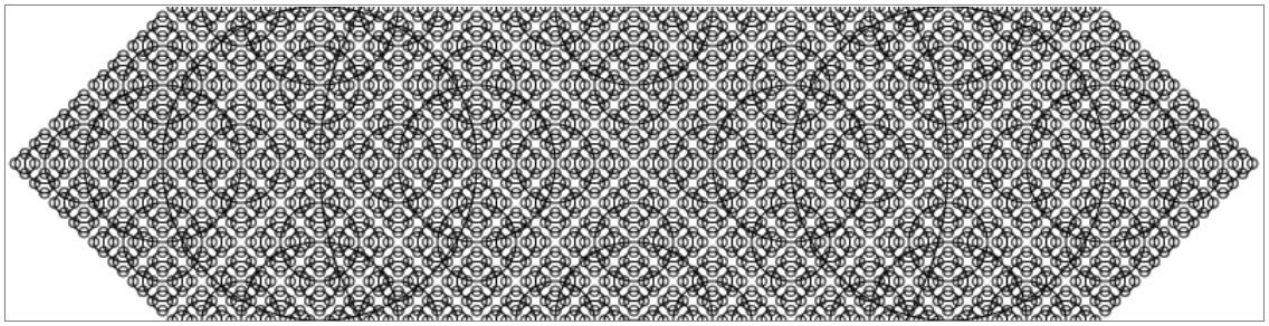

Рис. 7. Фрактал №2 в случае расширение по четырем направлениям

Данный фрактал можно получить незначительным изменением предыдущего алгоритма – здесь с помощью функции Translate меньшие круги отрисовываются не только по оси *y*, но и по оси *y*

```
import Graphics.Gloss

main = do
    animate (InWindow "Fractal" (800, 800) (20, 20)) white frame

frame :: Float -> Picture
frame _ =
    Color black
    $ Scale 150 150
    $ quadroCircles 6

quadroCircles :: Int -> Picture
quadroCircles 0 = Blank
quadroCircles n = Pictures [circle 1, circ1, circ2, circ3, circ4
]
    where
        circ1 = Translate 1 0 circNm
        circ2 = Translate (-1) 0 circNm
        circ3 = Translate 0 1 circNm
        circ4 = Translate 0 (-1) circNm

        circNm =
            Pictures
            [circle 0.5,
            Scale 0.5 0.5 $ quadroCircles (n-1)]
```

Листинг 3

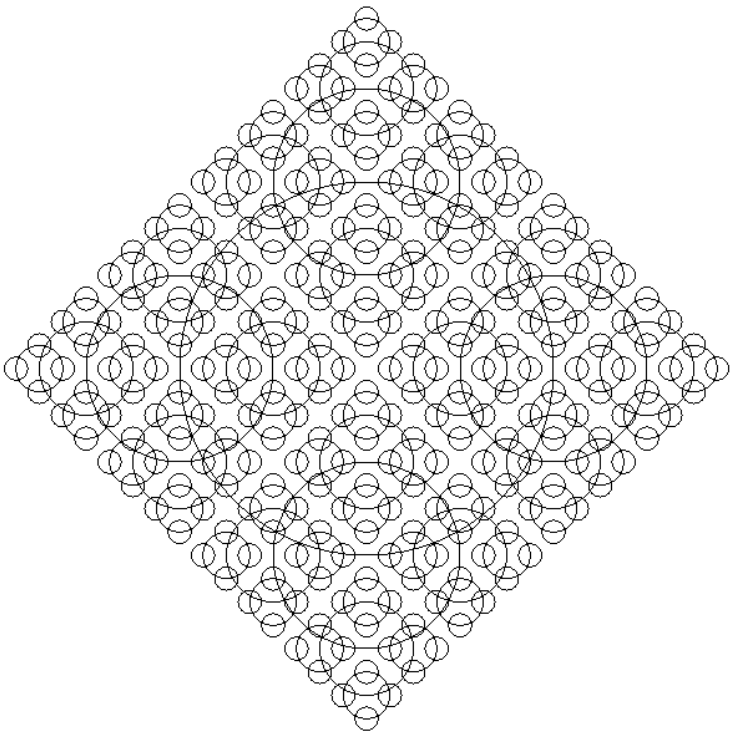

Рис. 8. Результат работы функции quadroCircle для n = 4

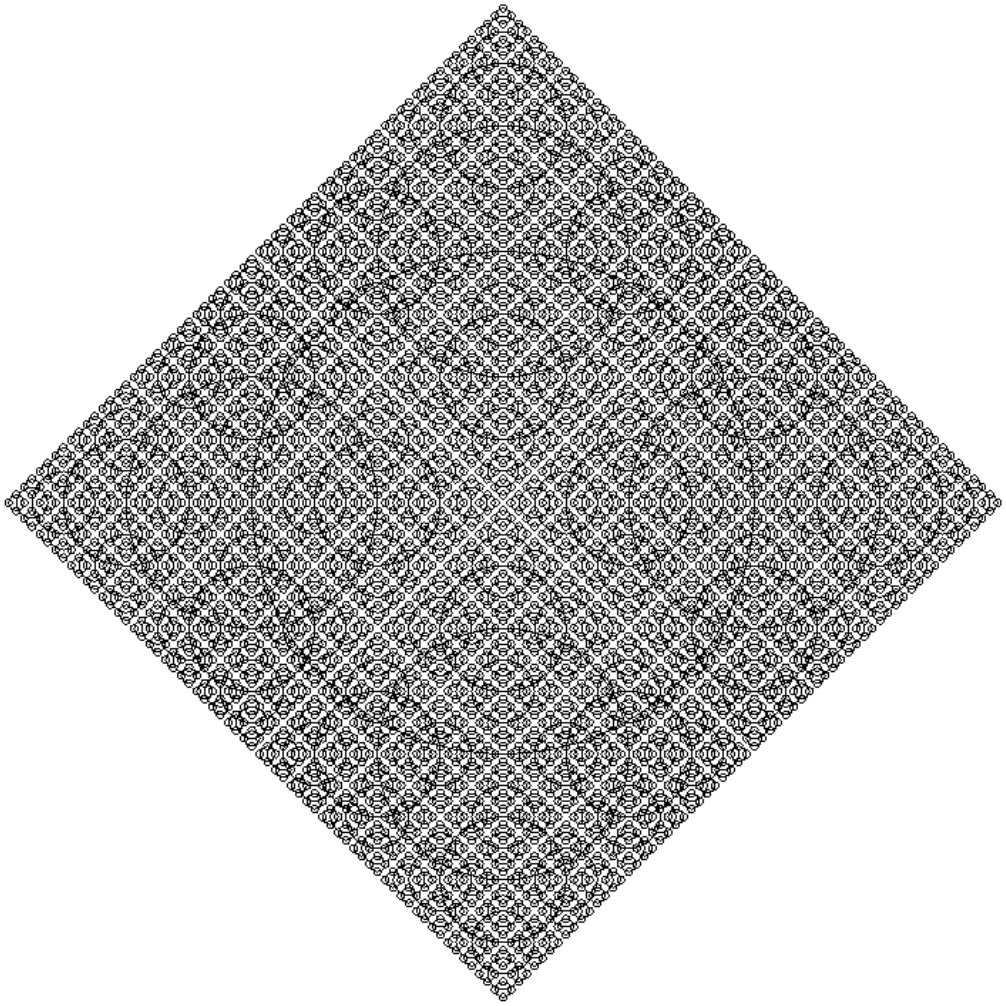

Рис. 9. Результат работы функции quadroCircle для n = 6

Для наглядности работы так же были приведены результаты работы алгоритма для разных n.

4. Визуализация следующего фрактала:

Рис. 10. Канторово множество

Данный фрактал представляет собой канторово множество: подмножество единичного отрезка вещественной прямой, которое является классическим примером дисконтинуума в математическом анализе.

Данный фрактал строится следующим образом:

Имеется отрезок длины n^3. Мы удаляем среднюю часть, то есть интервал $\left(\frac{n^2}{3};\frac{2n^2}{3}\right)$ и оставшиеся два отрезка проецируем отдельно. Повторяем удаление и отображение пока $n > 0$.

```
import Graphics.Gloss

main = do
    animate (InWindow "Fractal" (800, 800) (20, 20)) white frame

frame :: Float -> Picture
frame time =
    Color black
    $ Scale 100 100
    $ cantorSet 6 0 0

cantorSet :: Float -> Float -> Float-> Picture
cantorSet n start y
    | n > 0 = Pictures [circNm1]
    | otherwise = Blank
 where
    circNm1 =
        Pictures
        [ Line[(start, y), (start + 3**n, y)],
          cantorSet (n-1) start (y-5),
          cantorSet (n-1) (start + 2*3**(n-1)) (y-5)]
```

Листинг 4

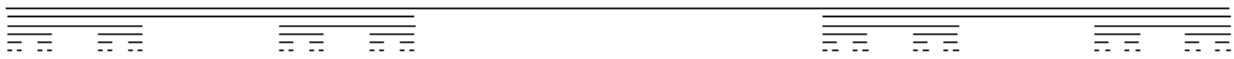

Рис. 11. Результат работы функции cantorSet для n = 6

Рис. 12. Результат работы функции cantorSet для n = 4

5. Визуализация заключительного фрактала

Последний фрактал является вариацией треугольника Серпинского.

Треугольник Серпинского — фрактал, один из двумерных аналогов множества Кантора, математическое описание которого опубликовал польский математик Вацлав Серпинский.

Однако представленный фрактал будет отличаться от него. Вместо закрашивания треугольников по вершинам на этих вершинах будут отрисовываться круги, соединенные линиями с центром большего круга.

В данном алгоритме производится перемещение рекурсивно отрисовываемых кругов с помощью коэффициентов *a* и *b*, где

$$a = \frac{1}{\sin\left(\frac{\pi}{3}\right)}, \quad b = a * \cos\left(\frac{\pi}{3}\right)$$

За счет этих коэффициентов точки вызова функции clock представляют собой вершины равностороннего треугольника.

```
import Graphics.Gloss

main = do
    animate (InWindow "Fractal" (800, 800) (20, 20)) white frame

frame :: Float -> Picture
frame time =
    Color black
    $ Scale 165 165
    $ clock 6

clock :: Int -> Picture
clock 0 = Blank
clock n = Pictures [circ1, circ2, circ3, lines]
    where
        circ1 = Translate 0 a circNm1
        circ2 = Translate 1 (-b) circNm1
        circ3 = Translate (-1) (-b) circNm1

        a = 1 / sin (pi / 3)
        b = a * cos (pi / 3)

        circNm1 =
            Pictures
            [ circle 1,
             Scale (a/2.5) (a/2.5) $ clock (n-1)]

        lines =
            Pictures
            [ Line [(0, 0), (0, a)],
              Line [(0, 0), (1, -b)],
              Line [(0, 0), (-1, -b)]]
```

Листинг 5

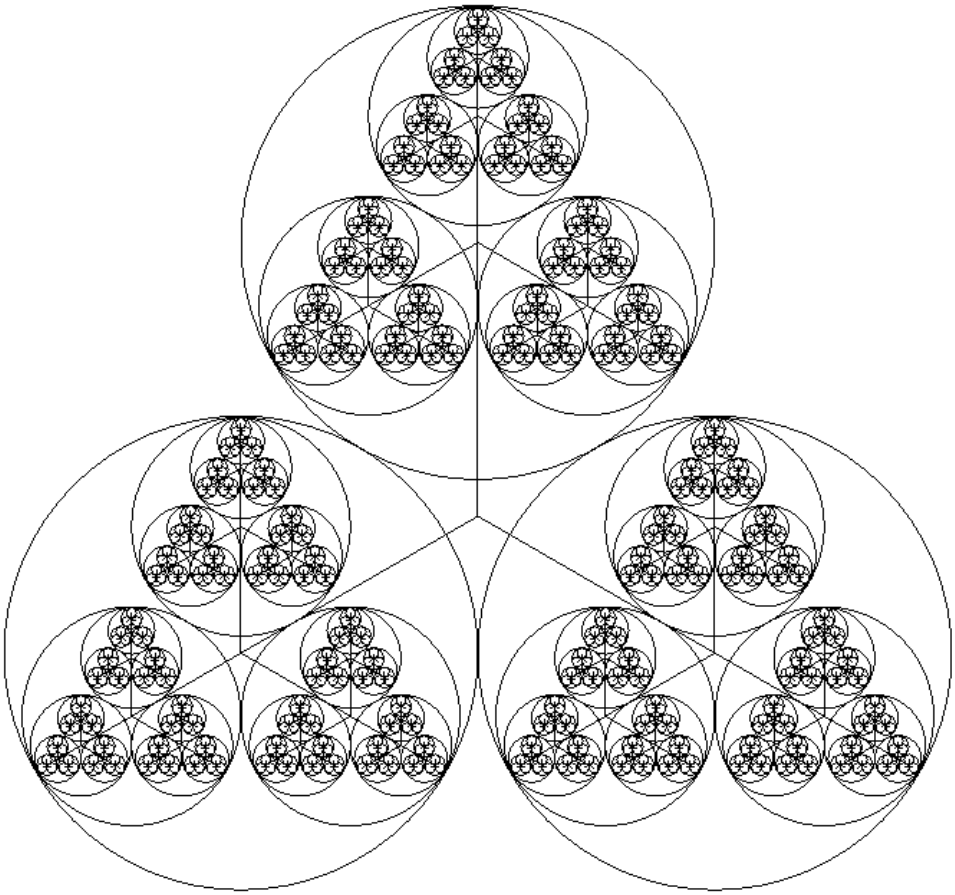

Рис. 13. Результат работы функции clock для n = 6

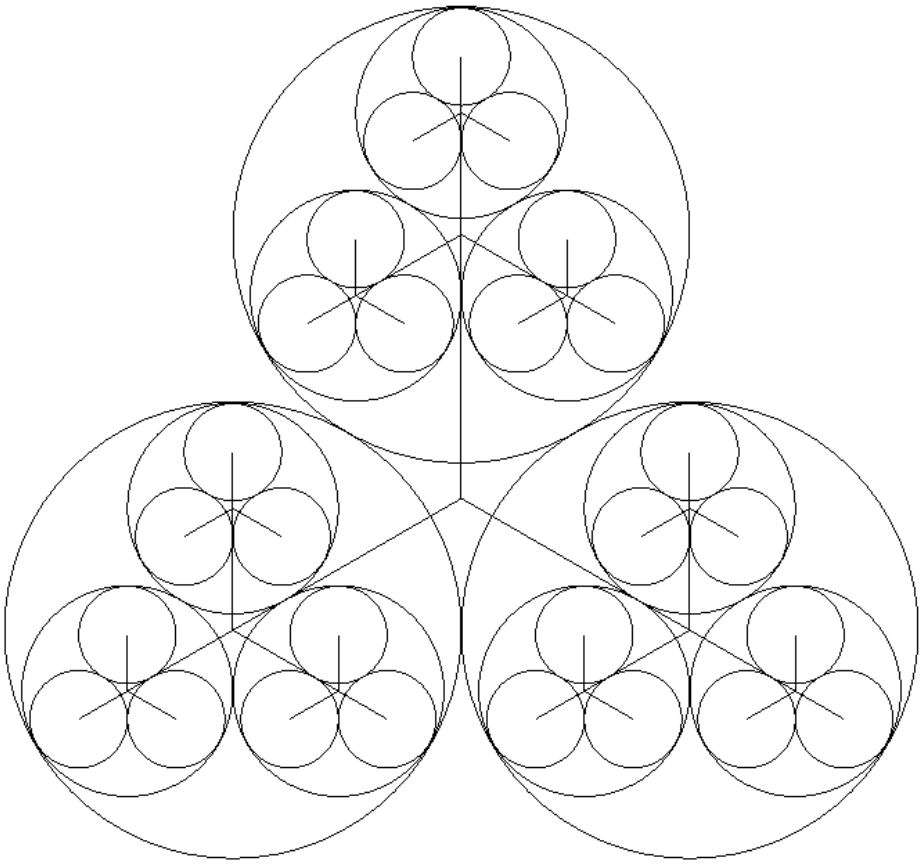

Рис. 14. Результат работы функции clock для n = 3

ЗАКЛЮЧЕНИЕ

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

Бали продемонстрированы возможности Haskell в визуализации изображений с помощью сторонней библиотеки Gloss, а также рекурсивные вызовы функций, выступающие основным инструментом работы. Для наглядного представления работы алгоритмов построения фракталов были приведены изображения фракталов разной глубины. Таким образом, были закреплены навыки работы с рассматриваемым языком.

СПИСОК 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. 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

4. Daniel Shiffman. The Nature of Code; 1st edition: The Nature of Code, 2012. – 520p.