МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет математики и компьютерных наук**

**Кафедра вычислительной математики и информатики**

**КУРСОВАЯ РАБОТА**

**ПОСТРОЕНИЕ ФРАКТАЛЬНЫХ КРИВЫХ**

Работу выполнил\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. С. Остришко

(подпись)

Направление подготовки 02.03.01 Математика и компьютерные науки курс 2

Научный руководитель

ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И. В. Сухан

(подпись, дата)

Нормоконтролер

преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.Ю. Шимохина

(подпись, дата)

Краснодар

2020

Оглавление

[Введение 3](#_Toc39590292)

[Анализ поставленной задачи и разработка алгоритма 4](#_Toc39590293)

[Выполнение программы 8](#_Toc39590294)

[Заключение 10](#_Toc39590295)

[Список источников 11](#_Toc39590296)

Введение

Была поставлена задача: программа моделирует «Кривую дракона» ⎯ фрактальную кривую. Строятся они на основе рекурсивного метода: каждой кривой ставится в соответствие последовательность нулей и единиц, где единицы соответствуют повороту кривой налево, а нуль ⎯ направо.

Для реализации был выбран язык Python из-за его модульности и простоты.

Цель: разработать программу, демонстрирующую построение кривой дракона на экране

Из поставленной цели были определенны данные задачи:

1. Проанализировать поставленную задачу
2. Разработать алгоритм решения задачи
3. Реализовать алгоритм на языке Python
4. Разработать документацию

Требования к программе: вывод результатов работы

Анализ поставленной задачи и разработка алгоритма

По условию, компьютер должен сгенерировать последовательность нулей и единиц, которой соответствует кривая, а затем смоделировать её на экране.

Для начала опишем алгоритм: кривая n-го порядка строится рекурсивно из кривой прошлого порядка. Для начала в конце изначальной кривой ставится единица, затем слева добавляется последовательность прошлого порядка, с изменением центральной единицы на нуль.

Для данного алгоритма была реализована функция:

def code(code\_):

if int(code\_) == 0:

return [1]

else:

c = code(int(code\_) - 1)

c += [1] + c

c[ len(c) - len(c) // 4 - 1] = 0

return c

В данной функции мы используем тип данных list для удобства работы. На вход функция принимает порядок кривой, а возвращает list с двоичным кодом кривой заданного порядка. Этот алгоритм мы используем далее для задания кривой.

Для моделирования кривой на экране используем пакет pygame, который подходит для простой графики. Для этого мы подключаем данный пакет:

import pygame

Инициализируем полотно pygame

pygame.init()

ui = pygame.display.set\_mode( (1000, 1000))

Далее опишем класс кривой с его методами построения, рисования и настройки задержки рисования.

class curve(object):

## Массив точек и задержка рисования ##

points = []

delay = 10

########################################

## Инициализация кривой ##

def \_\_init\_\_(self, points\_):

self.points = points\_

##########################

## Функция меняющая задержку ##

def change\_delay(self, d):

if self.delay + d > 0:

self.delay += d

else:

self.delay = 0

################################

## Функция рисования до номера поворота ##

def draw\_n(self, a, color):

pygame.time.delay(self.delay)

pygame.draw.line(ui,color,self.points[a],self.points[a+1])

##########################################

## Функция кодировки координат из кода поворотов ##

def encode(self, n):

i = 1

for c in code(n - 1):

dx = self.points[i][0] - self.points[i - 1][0]

dy = self.points[i][1] - self.points[i - 1][1]

if c == 1:

self.points.append((self.points[i][0] + 3 \* sign(dy), self.points[i][1] + 3 \* sign(dx)))

else:

self.points.append((self.points[i][0] - 3 \* sign(dy), self.points[i][1] - 3 \* sign(dx)))

i += 1

####################################################

В основной части программы инициализируем основные переменные:

dSpeed = 0

running = True

stop = False

i = 1

cu = curve([(500,500), (500,495)])

cu.encode(18)

ui.fill((255,255,255))

font=pygame.font.SysFont("arial",16)

Далее цикл while чтобы отслеживать работает ли программа:

while running:

И внутри цикла будем рисовать кривую и отслеживать нажатие клавиш пользователем. По нажатии Esc программа заканчивает работу, при удержании пробела рисование приостанавливается, для клавиши r назначено рисование заново, а по нажатии стрелок меняется скорость отрисовки. А так же выводится задержка рисования в виде текста.

for event in pygame.event.get(): ##Обработка событий

if event.type == pygame.QUIT: ##Если закрывается окно - выкл программа

running = False

## Функции рестарта и ускорения ##

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_r:

i = 1

ui.fill((255,255,255))

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

dSpeed = +2

elif event.key == pygame.K\_RIGHT:

dSpeed = -2

elif event.key == pygame.K\_SPACE:

stop = True

elif event.type == pygame.KEYUP:

if (event.key == pygame.K\_LEFT) or (event.key == pygame.K\_RIGHT):

dSpeed = 0

if event.key == pygame.K\_SPACE:

stop = False

####################################

if not stop:

text = font.render("Delay = %d"%cu.delay,1,(10,10,10),(255,255,255)) ## Инициализация текста

ui.blit(text,(0,0)) ## Наложение текста

cu.change\_delay(dSpeed) ## Смена задержки

cu.draw\_n(i,(i % 250, 0, 0) ) ## рисование отрезка

pygame.display.update() ## обновление экрана

i += 1

Выполнение программы

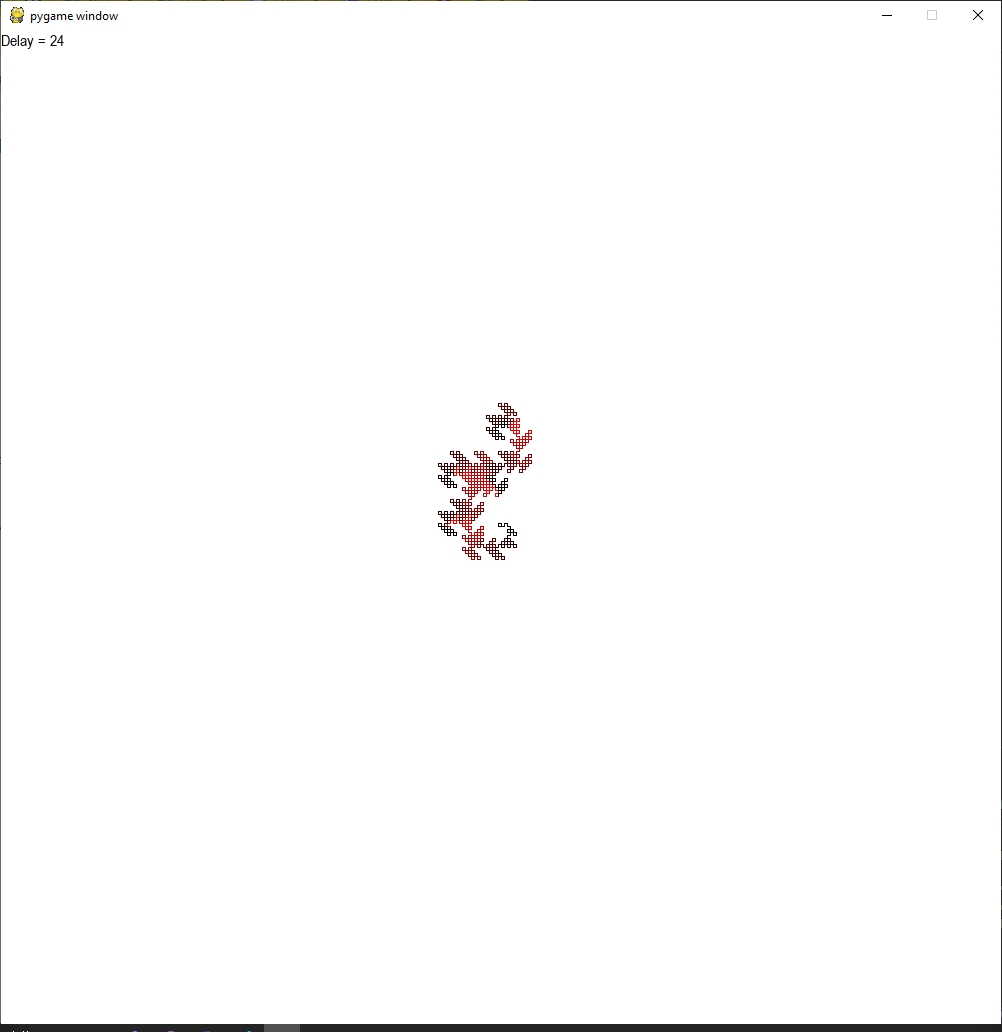


Рисунок 1

На рисунке выше продемонстрирована работа программы

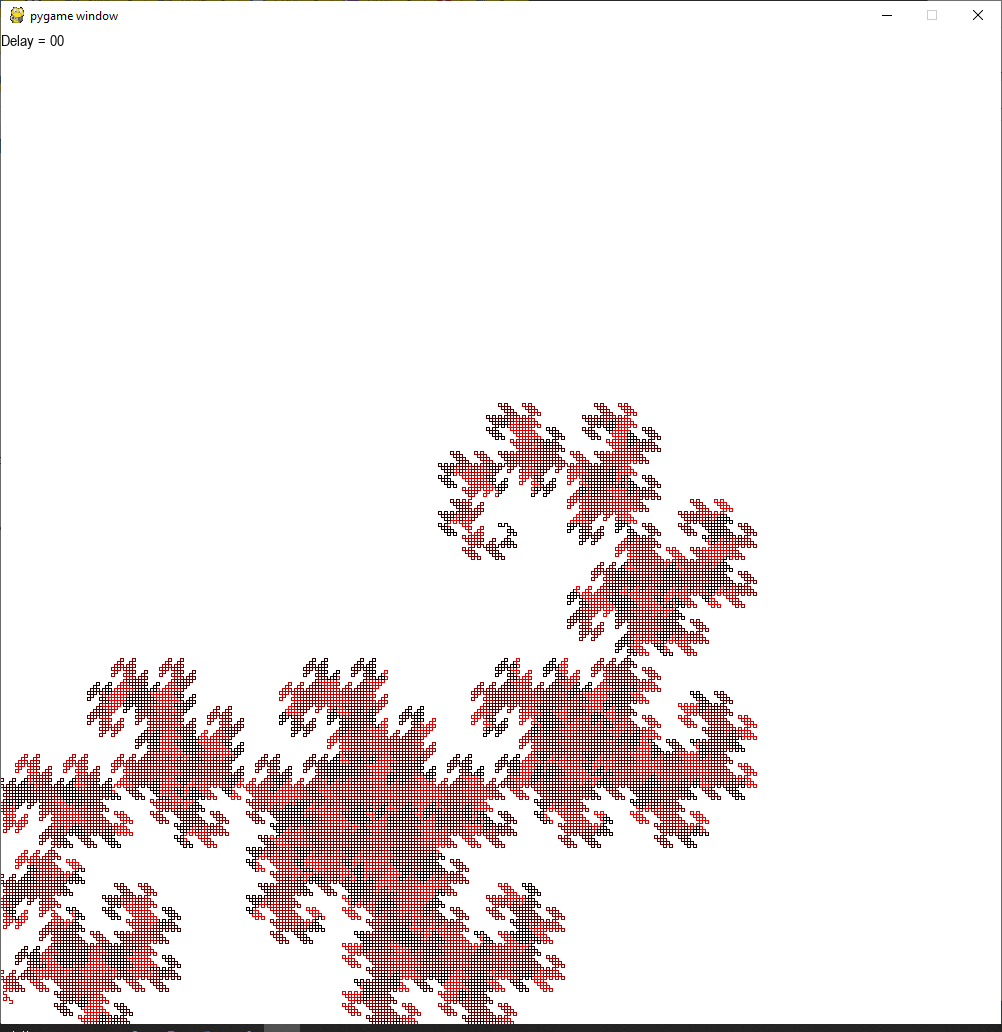


Рисунок 2

На *Рисунок* *2* представлена кривая, большего порядка. Программа работает корректно и кривая моделируется верно

Заключение

По итогу было сделано:

1. Анализ задачи и составлена математическая модель
2. Описан алгоритм решения задачи
3. Реализован алгоритм на языке Python
4. Составлена документация

Кривая была успешно просчитана компьютером с помощью рекурсивного отношения и выведена на экран с помощью пакета pygame.

Программа может быть доработана:

1. Добавление возможности приближения кривой
2. Закругление углов кривой для имитации отсутствия самопересечений

Список источников

1. https://www.pygame.org/