Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Отчёт по технологической (проектно-технологической) практике на тему:

РОСТ ДЕНДРИТА

Выполнил студент 5.205-1 группы:

_____ М. Е. Пивоваров

«___» _____ 2024 г.

Проверил: преп.

_____ И. А. Шмаков

«___» _____ 2024 г.

Барнаул 2024

РЕФЕРАТ

Объем работы листов 21

Количество рисунков 3

Количество используемых источников 8

Количество таблиц 0

ДЕНДРИТ, ЧАСТИЦА, ФУНКЦИЯ, МОДЕЛИРОВАНИЕ.

Данная работа посвещена разработке программы на языке программирования Python с использованием библиотеки FLTK, моделирующую рост дендрита.

Целью работы является создание программы на языке программирования Python с использованием библиотеки FLTK, моделирующую рост дендрита.

В работе рассмотрены теоретические сведения о дендрите и его росте, о библиотеке FLTK, языке программирования Python, среде разработки Geany. Практические сведения о ходе разработки программы ”Рост дендрита”.

Отчёт оформлена с помощью системы компьютерной вёрстки TEX и его расширения XETEX из дистрибутива *TeX Live*.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность Актуальность обусловлена непрерывном совершенствованием информационных технологий и обновлением информационных систем. Программные обеспечения используются во многих сферах человеческой деятельности. Написание своей собственной программы является важной частью обучения навыкам программирования. А обучение навыкам программирования в свою очередь является важной частью моего направления подготовки.

Цель

Создать программу на языке Python с использованием библиотеки FLTK, моделирующую рост дендрита с возможностью настроек различных параметров.

Задачи:

1. Установить библиотеку FLTK.
2. Выбрать редактор для разработки программы.
3. Изучить библиотеку FLTK.
4. Написать программу.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Дендрит

Дендри́ты [1] (от греч. δένδρον — дерево) — сложнокристаллические образования древовидной ветвящейся структуры.

Формирование

Дендрит представляет собой ветвящееся и расходящееся в стороны образование, возникающее при ускоренной или стеснённой кристаллизации в неравновесных условиях, когда кристалл расщепляется по определённым законам. В результате он утрачивает свою первоначальную целостность, появляются кристаллографически разупорядоченные блоки. Они ветвятся и разрастаются в разные стороны подобно дереву, тянущемуся к солнечному свету, кристаллографическая закономерность изначального кристалла в процессе его дендритного развития утрачивается по мере его роста. Дендриты могут быть трёхмерными объёмными (в открытых пустотах) или плоскими двумерными (если растут в тонких трещинах горных пород).

Процесс образования дендрита принято называть дендритный рост.

Самый крупный дендрит был обнаружен в конце XIX века в 100-тонном слитке стали и был назван в честь русского учёного Дмитрия Константиновича Чернова, детально исследовавшего процесс зарождения и роста кристаллов (в частности, дендритных стальных кристаллов). Вес «кристалла Д. К. Чернова» составил 3,45 кг, длина — 39 см, химический состав — 0,78 % углерода, 0,255 % кремния, 1,055 % марганца, 97,863 % железа.

Разновидности

В качестве примера дендритов можно привести снежинки, ледяные узоры на оконном стекле, живописные окислы марганца, имеющие вид деревьев в пейзажных халцедонах («моховой агат») и в тонких трещинах розо-

вого родонита. Другие примеры — веточки самородной меди в зонах окисления рудных месторождений, дендриты самородных серебра и золота, решётчатые дендриты самородного висмута и ряда сульфидов. Почковидные или кораллообразные дендриты известны для малахита, барита и многих других минералов, к ним относятся и так называемые «пещерные цветы» кальцита и арагонита в карстовых пещерах.

1.2. FLTK

Fast, Light Toolkit [2] — кросс-платформенная библиотека инструментов с открытым исходным кодом (лицензия LGPL) для построения графического интерфейса пользователя (GUI). FLTK произносится как «фултик».

Изначально разрабатывалась Биллом Спицтаком (Bill Spiztak). FLTK создавалась для поддержки 3D графики и поэтому имеет встроенный интерфейс к OpenGL, но хорошо подходит и для программирования обычных интерфейсов пользователя.

Библиотека использует свои собственные независимые системы виджетов, графики и событий, что позволяет писать программы одинаково выглядящие и работающие на разных операционных системах. В отличие от других подобных библиотек (Qt, GTK, wxWidgets) FLTK ограничивается только графической функциональностью. Поэтому она имеет малый размер и обычно компонуется статически (это исключение из лицензии GNU Lesser General Public License, разрешенное разработчиками). FLTK не использует сложных макросов, препроцессоров и продвинутых возможностей языка C++ (шаблоны, исключения, пространства имен). Вкупе с малым размером кода, это облегчает использование библиотеки не очень искушенными пользователями.

Однако эти достоинства порождают недостатки библиотеки, такие как меньшее число виджетов, несколько упрощенная графика и невозможность сборки приложения, выглядящего естественно под конкретной операционной системой.

Название

Изначально назывался FL (Forms Library). При переходе в open source выяснилось, что поиск по названию FL практически невозможен — аббревиатура FL также означает штат Флорида. Поэтому пакет был переименован в FLTK (FL Toolkit), позднее ему был придуман бэкроним Fast, Light Toolkit.

История

FLTK начал разрабатываться как замена библиотеке XForms, а позднее был портирован на Mac OS и Windows. FLTK появился раньше, чем другие популярные библиотеки для создания GUI, но был практически неизвестен до 1998 года.

Особенности

FLTK представляет собой библиотеку виджетов и работает на ОС UNIX/Linux X11, Microsoft Windows и MacOS X. Малый объём библиотеки делает её подходящей для использования во встраиваемых системах.

Для встраиваемых систем на основе embedded Linux возможны следующие варианты:

FLTK + nxlib + nano-X (довольно стабильно работает, но есть проблемы с кириллицей)

FLNX — порт FLTK 1.0.7 на nano-X (работает только с версией 0.92)

DirectFB FLTK — порт FLTK на DirectFB + собственно сам DirectFB (данная сборка нестабильная, шрифты необходимо инсталлировать как для X11 и указать путь в конфиге)

1.3. Visual Studio Code

Visual Studio Code (VS Code) [3] — текстовый редактор, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий»

редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

Visual Studio Code основан на Electron и реализуется через веб-редактор Monaco, разработанный для Visual Studio Online.

История

Visual Studio Code был анонсирован 29 апреля 2015 года компанией Microsoft на конференции Build, и вскоре была выпущена бета-версия.

18 ноября 2015 года Visual Studio Code был выпущен под лицензией MIT, а исходный код был опубликован на GitHub. Анонсирована поддержка расширений.

14 апреля 2016 года Visual Studio Code вышел из стадии бета-тестирования.

Возможности

Visual Studio Code — это редактор исходного кода. Он имеет многоязычный интерфейс пользователя и поддерживает ряд языков программирования, подсветку синтаксиса, IntelliSense, рефакторинг, отладку, навигацию по коду, поддержку Git и другие возможности. Многие возможности Visual Studio Code недоступны через графический интерфейс, зачастую они используются через палитру команд или JSON-файлы (например, пользовательские настройки). Палитра команд представляет собой подобие командной строки, которая вызывается сочетанием клавиш.

VS Code также позволяет заменять кодовую страницу при сохранении документа, символы перевода строки и язык программирования текущего документа.

С 2018 года появилось расширение Python для Visual Studio Code с открытым исходным кодом. Оно предоставляет разработчикам широкие возможности для редактирования, отладки и тестирования кода.

Также VS Code поддерживает редактирование и выполнение файлов типа «Блокнот Jupyter» (Jupyter Notebook) напрямую «из коробки» без установки внешнего модуля в режиме визуального редактирования и в режиме редактирования исходного кода.

На март 2019 года посредством встроенного в продукт пользовательского интерфейса можно загрузить и установить несколько тысяч расширений только в категории «programming languages» (языки программирования).

Также расширения позволяют получить более удобный доступ к программам, таким как Docker, Git и другие. В расширениях можно найти линтеры кода, темы для редактора и поддержку синтаксиса отдельных языков.

Visual Studio Code имеет поддержку плагинов, доступных через Visual Studio Marketplace. Они могут включать в себя дополнения к редактору, поддержку дополнительных языков программирования, статические анализаторы кода.

С мая 2019 года доступен закрытый тест редактора Visual Studio Online на основе VS Code. Он поддерживает все расширения и IntelliCode.

1.4. Python

Python [4] (в русском языке встречаются названия пито́н или па́йтон) — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является

выделение блоков кода отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

Python является мультипарадигменным языком программирования, поддерживающим императивное, процедурное, структурное, объектно-ориентированное программирование, метапрограммирование, функциональное программирование и асинхронное программирование. Задачи обобщённого программирования решаются за счёт динамической типизации. Аспектно-ориентированное программирование частично поддерживается через декораторы, более полноценная поддержка обеспечивается дополнительными фреймворками. Такие методики как контрактное и логическое программирование можно реализовать с помощью библиотек или расширений. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений с глобальной блокировкой интерпретатора (GIL), высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор CPython, который поддерживает большинство активно используемых платформ, являющийся стандартом де-факто языка. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. CPython компилирует исходные тексты в высокоуровневый байт-код, который исполняется в стековой виртуальной машине. К другим трём основным реализациям языка относятся Jython (для JVM), IronPython (для CLR/.NET) и PyPy.

PyPy написан на подмножестве языка Python (RPython) и разрабатывался как альтернатива CPython с целью повышения скорости исполнения программ, в том числе за счёт использования JIT-компиляции. Поддержка версии Python 2 закончилась в 2020 году. На текущий момент активно развивается версия языка Python 3. Разработка языка ведётся через предложения по расширению языка PEP (англ. Python Enhancement Proposal), в которых описываются нововведения, делаются корректировки согласно обратной связи от сообщества и документируются итоговые решения.

Стандартная библиотека включает большой набор полезных переносимых функций, начиная с возможностей для работы с текстом и заканчивая средствами для написания сетевых приложений. Дополнительные возможности, такие как математическое моделирование, работа с оборудованием, написание веб-приложений или разработка игр, могут реализовываться посредством обширного количества сторонних библиотек, а также интеграцией библиотек, написанных на Си или C++, при этом и сам интерпретатор Python может интегрироваться в проекты, написанные на этих языках[14]. Существует и специализированный репозиторий программного обеспечения, написанного на Python, — PyPI. Данный репозиторий предоставляет средства для простой установки пакетов в операционную систему и стал стандартом де-факто для Python. По состоянию на 2019 год в нём содержалось более 175 тысяч пакетов.

Python стал одним из самых популярных языков, он используется в анализе данных, машинном обучении, DevOps и веб-разработке, а также в других сферах, включая разработку игр. За счёт читабельности, простого синтаксиса и отсутствия необходимости в компиляции язык хорошо подходит для обучения программированию, позволяя концентрироваться на изучении алгоритмов, концептов и парадигм. Отладка же и экспериментирование в значительной степени облегчаются тем фактом, что язык является интерпретируемым. Применяется язык многими крупными компаниями, такими как Google или Facebook.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. Установка FLTK и выбор редактора кода

Установка FLTK

1. Чтобы установить FLTK для Python нужно в командной строке ввести команду:
   pip install pyFltk
2. Чтобы импортировать FLTK в код нужно в коде написать строку:

```
1 import fltk as fl
```

Выбор редактора кода

Для выполнения работы я выбрал редактор кода Visual Studio Code. Это довольно хороший редактор. Удобен в использовании, имеет подсветку синтаксиса, технологию автодополнения.

2.2. Создание главного окна и его наполнение

- Для создания главного окна создаем объект main_window класса Fl_Double_Window
- Создаем кнопку для запуска/остановки процесса роста дендрита, кнопку быстрой постройки дендрита, кнопку вкл/выкл сетки на поле. Это будут соответственно объекты btn_stop, btn_bid, btn_net класса Fl_Button
- Создаем объект menu_bar класса Fl_Menu_Bar - интерфейс меню
- Создаем группу объектов menuitems [5], которая определяет пункты меню: Меню, Настройки, Помощь, О программе, Выход

2.3. Функции кнопок

Функции, вызываемые при нажатии на кнопки будут иметь подпись call_. Для установки функции кнопки используется метод Fl_Button.callback([название функции]).

2.4. Создание дополнительных окон

Каждое дополнительное окно будет являться объектом определенного класса. Нужно создать следующие классы: Window_Setting, Window_Help, Window_About и соответственно объекты данных классов: window_s, window_h, window_a. Атрибутами классов дополнительных окон являются кнопки, поля ввода, поля вывода. Данные атрибуты являются объектами классов Fl_Button, Fl_Value_Input, Fl_Multiline_Output [6].

2.5. Создание констант, глобальных объектов и переменных

Создаем константы:

- MAIN_WIN_W = 800 и MAIN_WIN_H = 600 - ширина и высота главного окна;
- S_MIN = 50, W_MAX = MAIN_WIN_W - 10, H_MAX - мин. и макс. размеры сторон поля;
- G_MAX_PAR - макс. возможное кол-во частиц;
- G_MAX_PS - макс. размер частицы
- INDENT_WH - отступы поля
- SETTING_DFL - настройки по умолчанию

Создаем следующие глобальные объекты и переменные:

- particles - список, хранящий координаты частиц;
- part - текущая частица (которая движется);
- c_p - кол-во обычных частиц;
- c_pi - кол-во инверсных частиц;
- c_tol - общее число частиц;
- p_s - размер частицы;

- live - логическая. Определяет продолжать или закончить процесс роста дендрита;
- count - текущее кол-во частиц на поле;
- stop - логическая. Останавливает или продолжает процесс роста дендрита;
- exn - логическая. Определяет, активна ли функция расширение поля;
- bid_prs - логическая. Управляет процессом быстрой постройки дендрита;
- y_spn - координата *y* изначального положения частицы;
- p - список, хранящий вероятности движения частицы;
- arr_init - список, хранящий координаты *x*, которые может иметь частица в изначальном положении;
- fid_wh - список для хранения текущей ширины и высоты поля
- arr_xy - двумерный список, хранящий нули и единицы. Нужен для проверки наличия частиц около текущей частицы;
- fid_xy - правая и нижняя граница поля;

2.6. Создание класса рисования

Класс рисования будет называться Drawing [7]. У данного класса есть функция draw, которая будет выполнять рисование.

Работа данной функции: сначала вызывается функция fl_rectf(self.x(), self.y(), self.w(), self.h(), FL_GRAY), которая создает прямоугольную область рисования: первые два аргумента - это координаты левого верхнего угла области, следующие два аргумента - это ширина и высота области, и последний аргумент - это цвет области. Далее на этой области происходит рисование текущей частицы, которая движется. Затем отрисовываются частицы, которые уже приклеены на поле. Если включено отображение сетки, то она тоже будет отрисована. Для перерисовки вызывается функция redraw().

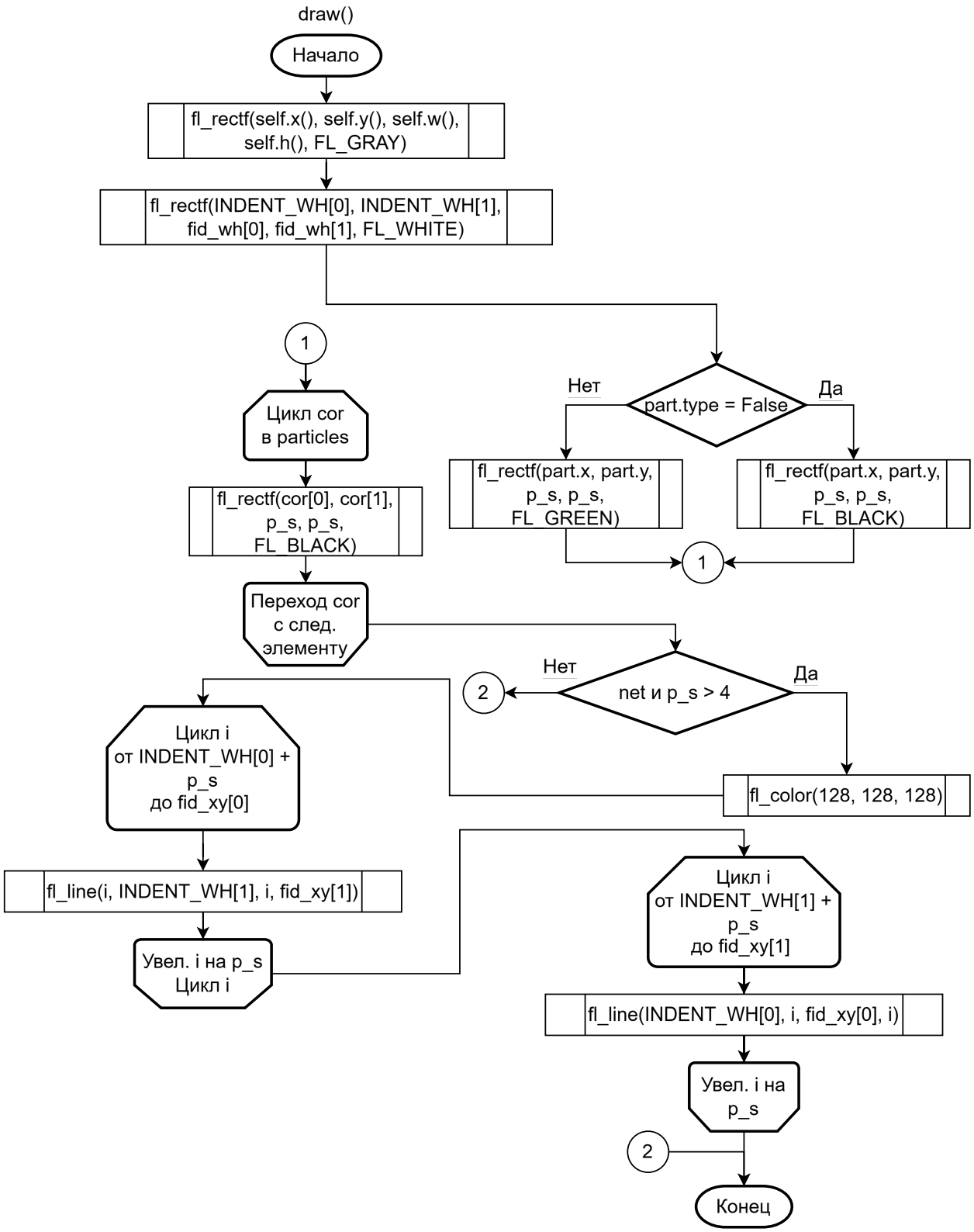

Рис. 2.1 Блок-схема функции draw.

2.7. Создание класса частицы

Класс частицы будет называться Particle. Атрибуты данного класса: x - координата x; y - координата y; tch - определяет коснулась ли частица чего-

либо; type - тип частицы. Данный класс имеет функцию move(), отвечающую за движение частицы и функцию inv(), отвечающую за инверсию (для инверсной частицы). Работа функции inv: функция работает только для инверсной частицы. Если произошло касание с какой-либо другой частицей, то создается список mass, который содержит координаты соседних клеток поля. Далее происходит проход по этому списку циклом for: проверяется значение ячейки списка arr_xy с индексами соответствующими координатам из списка mass. Если значение 0, то оно становится равным 1 и в список particles добавляются координаты из списка mass, иначе - равным 0 и из particles удаляются координаты из mass.

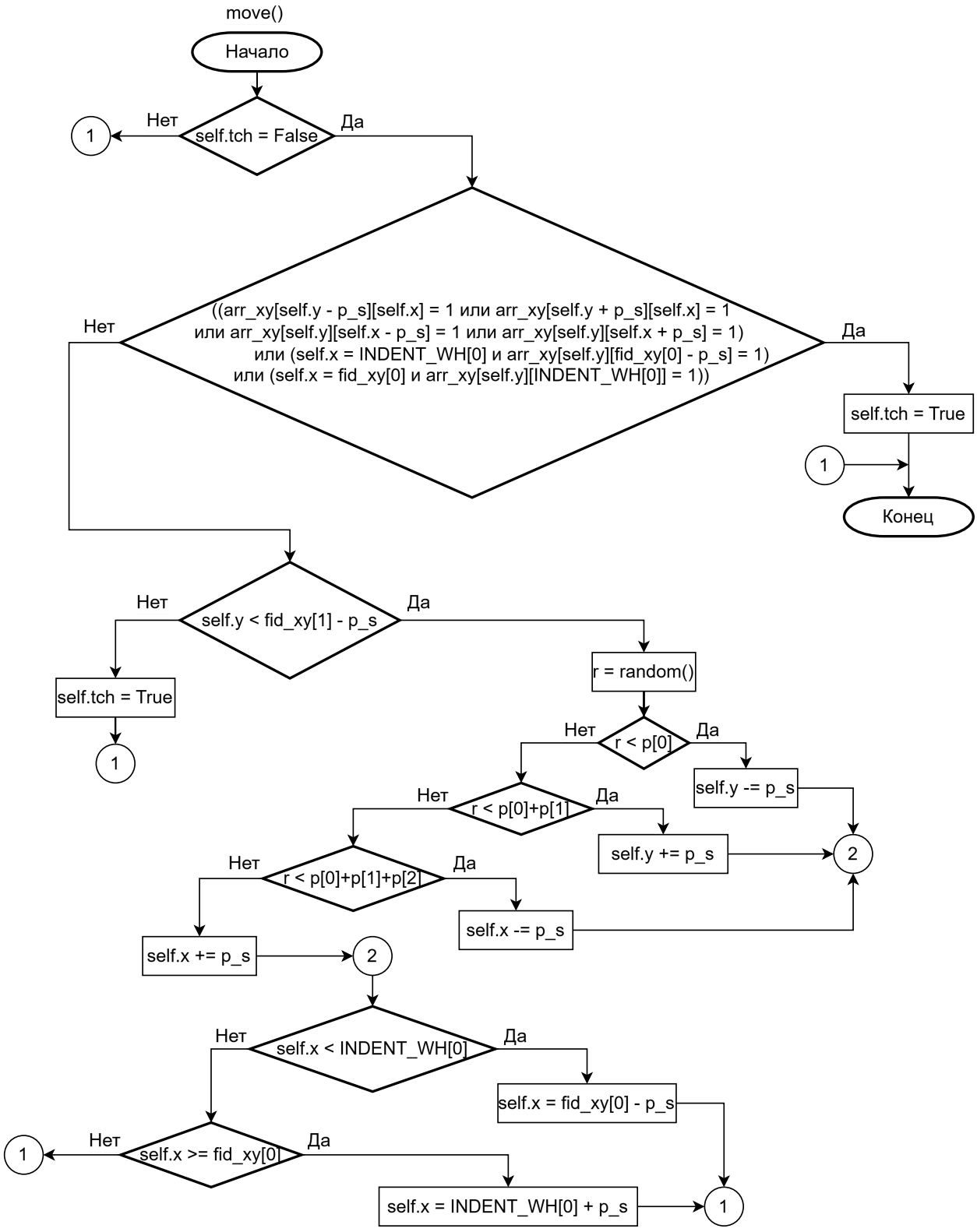

Рис. 2.2 Блок-схема функции move.

2.8. Создание основных функций

Функция core

Она управляет частицами: создает их, приводит их в движение. Данная функция работает следующим образом: если у текущей частицы произошло касание, то вычисляются вероятности выпадения обычной и инверсной частицы, далее создается частица тип которой выбирается случайно с учетом вычисленных ранее вероятностей. Далее запускается функция движения частицы move. Если частица коснулась чего-либо, то счетчик частиц (count) увеличивается на 1, в список particles добавляются координаты частицы, которая коснулась. В списке arr_xy элемент с индексами соответствующими координатам текущей частицы принимает значение 1. Вызывается функция инверсии inv(). Далее проверяется является ли частица, у которой произошло касание, последней частицей. Если да, то процесс роста дендрита прекращается (live становится равным False).

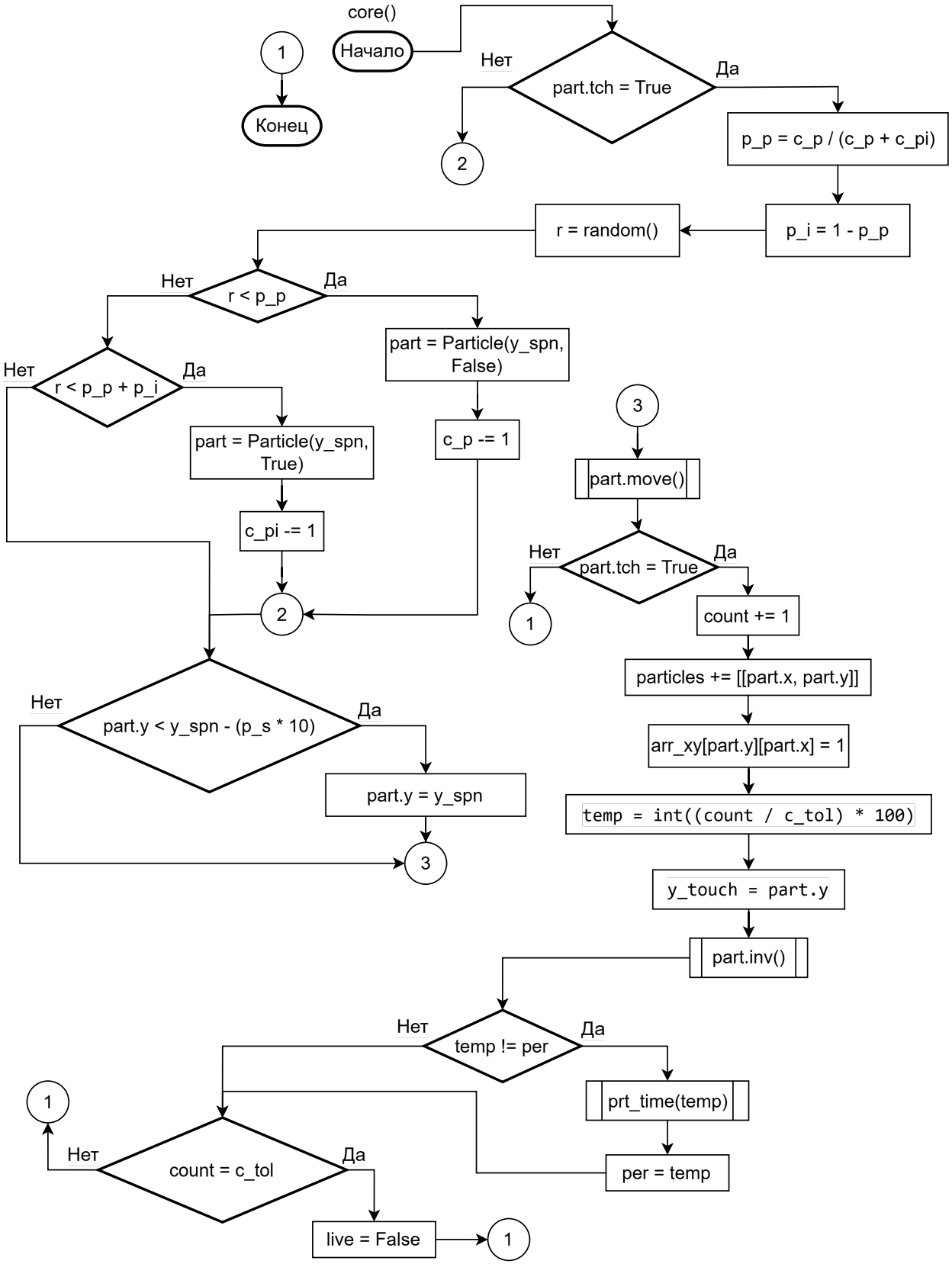

Рис. 2.3 Блок-схема функции core.

Функция расширения

Данная функция используется в режиме моделирования с расширением поля. Работа данной функции: проверяется, находится ли координата *y* текущей частицы слишком близко к линии создания частиц (y_spn). Если находится, то поле увеличивается по высоте к низу окна. Для этого происходит сдвиг координат *y* всех частиц на поле на одну клетку вниз, также изменяются и расположения единиц в списке arr_xy.

Функция повтора

Сначала проверяется, запущен ли процесс быстрой постройки и если нет, то вызывается функция redraw(), а затем - функция repeat_timeout(), которая сделает шаг моделирования по истечении небольшого промежутка времени [8]

ЗАКЛЮЧЕНИЕ

В результате выполнения работы ее цель была достигнута. Была изучена библиотека FLTK и получен опыт работы с ней. Получено больше опыта написания кода на языке программирования Python, написания отчета с помощью Latex, написания программы с графическим интерфейсом. В итоге была разработана программа, моделирующая рост дендрита с возможностью настройки различных параметров и также было успешно проведено ее тестирование.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Дендрит (кристалл) [Электронный ресурс] Википедия — свободная энциклопедия. Режим доступа: https://ru.wikipedia.org/wiki/Дендрит_(кристалл). — (Дата обр. 18.06.2024).
2. FLTK [Электронный ресурс] Википедия — свободная энцциклопедия. Режим доступа: https://ru.wikipedia.org/wiki/FLTK. — (Дата обр. 18.06.2024).
3. Visual Studio Code [Электронный ресурс] Википедия — свободная энциклопедия. Режим доступа: https://ru.wikipedia.org/wiki/Visual Studio Code. — (Дата обр. 17.12.2024).
4. Python [Электронный ресурс] Википедия — свободная энцциклопедия. Режим доступа: https://ru.wikipedia.org/wiki/Python. — (Дата обр. 18.06.2024).
5. pyFLTK Documentation [Электронный ресурс] pyfltk.sourceforge.io. Режим доступа: https://pyfltk.sourceforge.io/docs/CH0 Preface.html. — (Дата обр. 18.06.2024).
6. FLTK Documentation [Электронный ресурс] fltk.org. Режим доступа: https://www.fltk.org/doc-1.3/index.html. — (Дата обр. 18.06.2024).
7. pyFltk scripts [Электронный ресурс] fhoerni.free.fr. Режим доступа: http://fhoerni.free.fr/comp/pyFltk.html. — (Дата обр. 18.06.2024).
8. FLTK PLATFORMER [Электронный ресурс] GitHub. Режим доступа: https://github.com/SubwayMan/FLTK-Platformer. — (Дата обр. 18.06.2024).

ПРИЛОЖЕНИЕ

Текст программы

```
import fltk as fl
import random as rd
import os
import re
import time

FILENAME = "setting.txt" # Файл настройки

# Размеры главного окна
MAIN_WIN_W = 800
MAIN_WIN_H = 600

# Мин. и макс. размеры сторон поля
S_MIN = 50
W_MAX = MAIN_WIN_W - 10
H_MAX = MAIN_WIN_H - 40

G_MAX_PAR = 100000 # Макс. возможное кол-во частиц
G_MAX_PS = 50 # Макс. размер частицы
INDENT_WH = [5, 35] # Отступы поля

SETTING_NAMES = [
                    "number_particles", "number_particles_inv", "w", "h", "particle_size", "intervals",
                    "p_u", "p_d", "p_l", "p_r", "extension"
                    ]

SETTING_DFL = [500, 0, 300, 300, 5, "", 0.15, 0.35, 0.25, 0.25, 0] # Настройки по умолчанию
setting = [0, 0, 0, 0, 0, "", 0.0, 0.0, 0.0, 0.0, 0] # Текущие настройки

p = [0.15, 0.35, 0.25, 0.25] # Вероятности движения частицы (вверх, вниз, влево, вправо)

# Кол-во частиц (обычных и инверсных)
c_p = 0
c_pi = 0

fid_wh = [0, 0] # Размер поля
p_s = 0 # Размер частицы
inter = "" # Интервалы
fid_xy = [0, 0] # Граница поля справа и снизу
arr_xy = [] # Массив 0 и 1 для обнаружения частиц на поле
arr_init = [0] # Координаты x иниц-ции частицы
particles = [] # Частицы на поле
count = 0 # Кол-во частиц на поле
c_tol = 0 # Общее кол-во частиц
live = True # Процесс роста
net = False # Сетка
```

```
47 stop = True # Пауза
48 bid_prs = False # Процесс быстрой постройки
49 y_spn = 0 # Кор. y - на которой создается частица
50 exn = False # Расширение поля
51 per = 0 # Процент частиц на поле
52 speed = 0.015 # Скорость движения частиц
53 v_time = time.time() # Точка отсчета
54 ref = 0 # Ссылка на нужный режим моделирования
55 y_touch = 1000 # Кор. y касания частицы
56
57 # Класс отрисовки
58 class Drawing(fl.Fl_Widget):
59     def draw(self):
60         fl.fl_rectf(self.x(), self.y(), self.w(), self.h(), fl.FL_GRAY)
61         fl.fl_rectf(INDENT_WH[0], INDENT_WH[1], fid_wh[0], fid_wh[1], fl.FL_WHITE)
62         if part.type == False:
63             fl.fl_rectf(part.x, part.y, p_s, p_s, fl.FL_BLACK)
64         else:
65             fl.fl_rectf(part.x, part.y, p_s, p_s, fl.FL_GREEN)
66         for cor in particles:
67             fl.fl_rectf(cor[0], cor[1], p_s, p_s, fl.FL_BLACK)
68         if net and p_s > 4:
69             fl.fl_color(128, 128, 128)
70             for i in range(INDENT_WH[0] + p_s, fid_xy[0], p_s):
71                 fl.fl_line(i, INDENT_WH[1], i, fid_xy[1])
72             for i in range(INDENT_WH[1] + p_s, fid_xy[1], p_s):
73                 fl.fl_line(INDENT_WH[0], i, fid_xy[0], i)
74
75 # Класс частицы
76 class Particle():
77     def __init__(self, y, t):
78         self.x = rd.choice(arr_init)
79         self.y = y
80         self.tch = False
81         self.type = t
82
83     # Движение
84     def move(self):
85         if self.tch == False:
86             if ((arr_xy[self.y - p_s][self.x] == 1 or arr_xy[self.y + p_s][self.x] == 1 or arr_xy[self.y][self.x - p_s] == 1 or
            ↪    arr_xy[self.y][self.x + p_s] == 1)
87             or (self.x == INDENT_WH[0] and arr_xy[self.y][fid_xy[0] - p_s] == 1) or (self.x == fid_xy[0] and
            ↪    arr_xy[self.y][INDENT_WH[0]] == 1)):
88                 self.tch = True
89
90             elif self.y < fid_xy[1] - p_s:
91                 r = rd.random()
92                 if r < p[0]:
93                     self.y -= p_s
94
95                 elif r < p[0] + p[1]:
96                     self.y += p_s
97
98                 elif r < p[0] + p[1] + p[2]:
```

```
99                          self.x -= p_s
100
101                     else:
102                         self.x += p_s
103
104                     if self.x < INDENT_WH[0]:
105                         self.x = fid_xy[0] - p_s
106
107                     elif self.x >= fid_xy[0]:
108                         self.x = INDENT_WH[0] + p_s
109
110                 else:
111                     self.tch = True
112
113     # Инверсия соседних частиц
114     def inv(self):
115         global particles, arr_xy
116         if self.type == True:
117             if self.tch == True and self.y != fid_xy[1] - p_s:
118                 mass = [[self.x, self.y - p_s], [self.x, self.y + p_s]]
119                 if self.x == INDENT_WH[0]:
120                     mass += [[fid_xy[0] - p_s, self.y], [fid_xy[0] - p_s, self.y - p_s], [fid_xy[0] - p_s, self.y + p_s]]
121                 else:
122                     mass += [[self.x - p_s, self.y], [self.x - p_s, self.y - p_s], [self.x - p_s, self.y + p_s]]
123                 if self.x == fid_xy[0] - p_s:
124                     mass += [[INDENT_WH[0], self.y], [INDENT_WH[0], self.y - p_s], [INDENT_WH[0], self.y + p_s]]
125                 else:
126                     mass += [[self.x + p_s, self.y], [self.x + p_s, self.y - p_s], [self.x + p_s, self.y + p_s]]
127                 for i in mass:
128                     if arr_xy[i[1]][i[0]] == 0 and i[1] != fid_xy[1]:
129                         arr_xy[i[1]][i[0]] = 1
130                         particles += [i]
131                     elif i[1] != fid_xy[1]:
132                         arr_xy[i[1]][i[0]] = 0
133                         particles.remove(i)
134
135 # Окно настроек
136 class Window_Setting():
137     def __init__(self):
138         self.x = 180
139         self.w = 450
140         self.h = 300
141         self.y = 30
142         self.yb = 30
143         self.wi = 10
144         self.hi = 20
145         self.window = fl.Fl_Window(self.w, self.h, "Настройки")
146         self.button = fl.Fl_Button(self.w - 110, self.h - 30, 100, 25, "Прменить")
147         self.button.callback(call_set)
148         self.btn_dfl = fl.Fl_Button(self.w - 110, self.h - 65, 100, 25, "По умолчанию")
149         self.btn_dfl.callback(call_dfl)
150         self.ipt_cp = fl.Fl_Value_Input(self.x, self.yb, self.wi * 10, self.hi, "Кол-во обычных частиц")
151         self.ipt_cp.type(fl.FL_INT_INPUT)
152         self.ipt_cpi = fl.Fl_Value_Input(self.x, self.yb + self.y, self.wi * 10, self.hi, "Кол-во инверсных частиц")
```

```
153             self.ipt_cpi.type(fl.FL_INT_INPUT)
154             self.ipt_w = fl.Fl_Value_Input(self.x, self.yb + self.y * 2, self.wi * 10, self.hi, "Ширина поля")
155             self.ipt_w.type(fl.FL_INT_INPUT)
156             self.ipt_h = fl.Fl_Value_Input(self.x, self.yb + self.y * 3, self.wi * 10, self.hi, "Высота поля")
157             self.ipt_h.type(fl.FL_INT_INPUT)
158             self.ipt_ps = fl.Fl_Value_Input(self.x, self.yb + self.y * 4, self.wi * 10, self.hi, "Размер частицы")
159             self.ipt_ps.type(fl.FL_INT_INPUT)
160             self.ipt_inter = fl.Fl_Input(self.x, self.yb + self.y * 5, self.wi * 23, self.hi, "Интервал")
161             self.box = fl.Fl_Box(self.x * 2, self.yb, self.wi, self.hi, "Вероятности:")
162             self.ipt_p_u = fl.Fl_Value_Input(self.x * 2, self.yb + self.y, self.wi * 5, self.hi, "Вверх")
163             self.ipt_p_d = fl.Fl_Value_Input(self.x * 2, self.yb + self.y * 2, self.wi * 5, self.hi, "Вниз")
164             self.ipt_p_l = fl.Fl_Value_Input(self.x * 2, self.yb + self.y * 3, self.wi * 5, self.hi, "Влево")
165             self.ipt_p_r = fl.Fl_Value_Input(self.x * 2, self.yb + self.y * 4, self.wi * 5, self.hi, "Вправо")
166             self.btn_exn = fl.Fl_Check_Button(self.x, self.yb + self.y * 6, self.wi * 11, self.hi, "Расширение")
167
168 # Окно помощи
169 class Window_Help():
170     def __init__(self):
171             self.w = 800
172             self.h = 600
173             self.window = fl.Fl_Window(self.w, self.h, "Помощь")
174             self.out = fl.Fl_Multiline_Output(5, 5, 790, 590)
175             self.out.insert("Рост дендрита прекращается, если\n"
176                             "все частицы находятся на поле или высота дендрита достигла верхней границы поля;\n\n"
177                             "О кнопках:\n"
178                             "    Пуск/Пауза - запуск/остановка процесса роста дендрита;\n"
179                             "    Построить - запуск быстрой постройки дендрита;\n"
180                             "    Сетка - вкл/выкл сетки на поле (при размере частицы > 4);\n"
181                             "    Очистить - очистить консоль;\n\n"
182                             "О настройках:\n"
183                             "    Макс. значения для параметров:\n"
184                             "        Кол-во обычных и инверсных частиц - 100000;\n"
185                             "        Ширина / Высота - 790 / 560;\n"
186                             "        Размер частицы - 50;\n\n"
187                             "    Сумма выбранных вероятностей должна быть равна 1;\n"
188                             "    Расширение - если этот параметр активен:\n"
189                             "        Каждый раз, когда рост дендрита достигает верхней границы поля,\n"
190                             "        высота поля увеличивается.\n"
191                             "        Расширение перестанет работать, если высота поля достигла макс. значения (560).\n\n"
192                             "    Можно устанавливать определенные интервалы (по координате x),\n"
193                             "    в которых создаются частицы.\n\n"
194                             "        Интервалы вводятся по шаблону:\n"
195                             "            (от 1 до 3 цифр)..(от 1 до 3 цифр), (от 1 до 3 цифр)..(от 1 до 3 цифр), и т.д\n"
196                             "        Примеры интервалов:\n"
197                             "        1) 100..200\n"
198                             "        2) 10..20, 90..120\n\n"
199                             "Инверсная частица при касании модифицирует клетки вокруг себя:\n"
200                              "клетки с частицами очищаются, без - заполняются частицой")
201
202 # Окно "О программе"
203 class Window_About():
204     def __init__(self):
205             self.w = 500
206             self.h = 300
```

```
207             self.window = fl.Fl_Window(self.w, self.h, "О программе")
208             self.out = fl.Fl_Multiline_Output(5, 5, 490, 290)
209             self.out.insert("Название программы: Рост дендрита\n"
210                             "Программа моделирует рост дендрита с различными параметрами\n"
211                             "Автор: Пивоваров М.Е.\nГруппа: 5.205-1\n"
212                             "Используемая графическая библиотека: FLTK\n\n"
213                             "2024")
214
215 # Переключение сетки
216 def call_net(widget):
217     global net
218     if net:
219         net = False
220     else:
221         net = True
222     drawing.redraw()
223
224 # Установка значений по умолчанию
225 def call_dfl(widget):
226     window_s.ipt_cp.value(SETTING_DFL[0])
227     window_s.ipt_cpi.value(SETTING_DFL[1])
228     window_s.ipt_w.value(SETTING_DFL[2])
229     window_s.ipt_h.value(SETTING_DFL[3])
230     window_s.ipt_ps.value(SETTING_DFL[4])
231     window_s.ipt_inter.value(SETTING_DFL[5])
232     window_s.ipt_p_u.value(SETTING_DFL[6])
233     window_s.ipt_p_d.value(SETTING_DFL[7])
234     window_s.ipt_p_l.value(SETTING_DFL[8])
235     window_s.ipt_p_r.value(SETTING_DFL[9])
236     window_s.btn_exn.value(SETTING_DFL[10])
237
238 # Переключение Пуск/Пауза
239 def call_stop(widget):
240     global stop
241     if stop == False:
242         stop = True
243         btn_stop.label("Пуск")
244
245     else:
246         stop = False
247         btn_stop.label("Пауза")
248
249 # Вызов окна настроек
250 def call_setting(widget):
251     global live
252     live = False
253     window_s.window.end()
254     window_s.window.show()
255
256 # Запуск быстрой постройки
257 def call_bid(widget):
258     global stop, bid_prs, live, v_time
259     per_l = int(ipt_per.value())
260     if per_l < 0 or per_l > 100:
```

```
261             per_l = 100
262             ipt_per.value(100)
263         if count < c_tol and per < per_l:
264             live = True
265             bid_prs = True
266             stop = False
267             v_time = time.time()
268             btn_stop.label("Пауза")
269             window_s.window.hide()
270             window_a.window.hide()
271             window_h.window.hide()
272             main_window.hide()
273             bild(per_l)
274             bid_prs = False
275             main_window.show()
276             main_window.end()
277             drawing.redraw()
278
279 # Действие кнопки "Применить"
280 def call_set(widget):
281     apply()
282
283 # Вызов окна помощи
284 def call_help(widget):
285     window_h.window.end()
286     window_h.window.show()
287
288 # Вызов окна "О программе"
289 def call_about(widget):
290     window_a.window.end()
291     window_a.window.show()
292
293 # Действия при закрытии окна
294 def call_hid(widget):
295     global live
296     if widget == window_s.window:
297         if count < c_tol:
298             live = True
299     widget.hide()
300
301 # Завершение программы
302 def call_end(widget):
303     global setting
304     with open(FILENAME, "w") as file:
305         for i in range(len(SETTING_DFL)):
306             file.write(f"{SETTING_NAMES[i]} = {setting[i]}\n")
307     window_s.window.hide()
308     window_a.window.hide()
309     window_h.window.hide()
310     main_window.hide()
311
312 # Очистка консоли
313 def call_clr(widget):
314     os.system('cls')
```

```
315
316 # Изменение скорости
317 def call_speed(widget):
318         global speed
319         speed = 1 / widget.value()
320
321 # Сброс скорости на значение по умолчанию
322 def call_sp_res(widget):
323         global speed
324         speed = 0.015
325         slider.value(60)
326
327 # Создание доп. окон
328 window_s = Window_Setting()
329 window_h = Window_Help()
330 window_a = Window_About()
331
332 # Главное окно
333 main_window = fl.Fl_Double_Window(MAIN_WIN_W, MAIN_WIN_H, "Рост дендрита")
334 main_window.callback(call_end)
335
336 # Иниц-ция частицы
337 part = Particle(y_spn, False)
338 part.tch = True
339
340 # Поле рисования
341 drawing = Drawing(INDENT_WH[0], INDENT_WH[1], MAIN_WIN_W, MAIN_WIN_H)
342
343 # Установка функций при закрытии окон
344 window_s.window.callback(call_hid)
345 window_a.window.callback(call_hid)
346 window_h.window.callback(call_hid)
347
348 # Элементы меню
349 menuitems = (( "&Меню", 0, 0, 0, fl.FL_SUBMENU ),
350         ( "&Настройки", 0, call_setting ),
351         ( "&Помощь", 0, call_help ),
352         ( "&О программе", 0, call_about ),
353         ( "&Выход", 0, call_end ),
354         ( None, 0 )
355 )
356
357 # Панель меню
358 menu_bar = fl.Fl_Menu_Bar(0, 0, 100, 30)
359 menu_bar.copy(menuitems)
360
361 # Кнопки
362 btn_stop = fl.Fl_Button(55, 0, 80, 30, "Пуск")
363 btn_stop.callback(call_stop)
364 btn_bid = fl.Fl_Button(215, 0, 80, 30, "Построить") #295
365 btn_bid.callback(call_bid)
366 btn_net = fl.Fl_Button(135, 0, 80, 30, "Сетка")
367 btn_net.callback(call_net)
368 btn_clr = fl.Fl_Button(718, 0, 80, 30, "Очистить") #135
```

```
369 btn_clr.callback(call_clr)
370 btn_sp_res = fl.Fl_Button(658, 0, 60, 30, "Сброс")
371 btn_sp_res.callback(call_sp_res)
372
373 # Для ввода процента быстрой постройки
374 ipt_per = fl.Fl_Value_Input(297, 0, 35, 30, "")
375 ipt_per.align(fl.FL_ALIGN_RIGHT)
376 ipt_per.value(100)
377
378 # Для изменения скорости частицы
379 slider = fl.Fl_Slider(400, 0, 257, 30, "Скорость")
380 slider.type(fl.FL_HORIZONTAL)
381 slider.color2(fl.FL_RED)
382 slider.align(fl.FL_ALIGN_LEFT)
383 slider.callback(call_speed)
384 slider.minimum(1)
385 slider.maximum(100)
386 slider.step(1)
387 slider.value(60)
388
389 # Быстрая постройка
390 def bild(per_l):
391     while per < per_l and count < c_tol:
392         ref()
393
394 # Конец моделирования
395 def live_off():
396     global live, count, y_touch
397     if y_touch <= y_spn + p_s:
398         live = False
399         count = c_tol
400     y_touch = 1000
401
402 # Для режима с расширением поля
403 def f_exn():
404     global y_touch, live, count
405     if fid_wh[1] + p_s <= H_MAX and y_touch <= y_spn + p_s:
406         fid_wh[1] += p_s
407         fid_xy[1] = INDENT_WH[1] + fid_wh[1]
408         for i in particles:
409             arr_xy[i[1]][i[0]] = 0
410             i[1] += p_s
411             arr_xy[i[1]][i[0]] = 1
412     elif fid_wh[1] + p_s > H_MAX and y_touch <= y_spn + p_s:
413         live = False
414         count = c_tol
415     y_touch = 1000
416
417 # Повтор шага моделирования
418 def repeat():
419     if bid_prs == False:
420         drawing.redraw()
421         fl.Fl.repeat_timeout(speed, ref)
422
```

```
423 # Стандартный режим
424 def way_0():
425     if stop == False and live == True:
426         core()
427         live_off()
428     repeat()
429 
430 # Режим с расширением поля
431 def way_1():
432     if stop == False and live == True:
433         core()
434         f_exn()
435     repeat()
436 
437 # Применение настроек
438 def apply():
439     global particles, part, c_p, c_pi, count, c_tol, arr_xy, fid_wh, p_s, arr_init, y_spn, live, v_time, inter, setting, exn, p, per, ref
440     arr_xy = []
441     for i in range(MAIN_WIN_H):
442         l = []
443         for j in range(MAIN_WIN_W):
444             l.append(0)
445         arr_xy.append(l)
446     cp = int(window_s.ipt_cp.value())
447     cpi = int(window_s.ipt_cpi.value())
448     w = int(window_s.ipt_w.value())
449     h = int(window_s.ipt_h.value())
450     ps = int(window_s.ipt_ps.value())
451     p_u = abs(window_s.ipt_p_u.value())
452     p_d = abs(window_s.ipt_p_d.value())
453     p_l = abs(window_s.ipt_p_l.value())
454     p_r = abs(window_s.ipt_p_r.value())
455     if p_u + p_d + p_l + p_r == 1:
456         p = [p_u, p_d, p_l, p_r]
457     else:
458         p = [SETTING_DFL[6], SETTING_DFL[7], SETTING_DFL[8], SETTING_DFL[9]]
459         window_s.ipt_p_u.value(SETTING_DFL[6])
460         window_s.ipt_p_d.value(SETTING_DFL[7])
461         window_s.ipt_p_l.value(SETTING_DFL[8])
462         window_s.ipt_p_r.value(SETTING_DFL[9])
463     if cp >= 0 and cp <= G_MAX_PAR:
464         c_p = cp
465     if cpi >= 0 and cpi <= G_MAX_PAR:
466         c_pi = cpi
467     if ps > 0 and ps <= G_MAX_PS:
468         p_s = ps
469 
470     if w > W_MAX:
471         window_s.ipt_w.value(W_MAX)
472         fid_wh[0] = W_MAX - W_MAX % p_s
473     elif w < S_MIN:
474         window_s.ipt_w.value(S_MIN)
475         fid_wh[0] = S_MIN - S_MIN % p_s
476     else:
```

```
            fid_wh[0] = w - w % p_s

    if h > H_MAX:
            window_s.ipt_h.value(H_MAX)
            fid_wh[1] = H_MAX - H_MAX % p_s
    elif h < S_MIN:
            window_s.ipt_h.value(S_MIN)
            fid_wh[1] = S_MIN - S_MIN % p_s
    else:
            fid_wh[1] = h - h % p_s

    c_tol = c_p + c_pi
    if c_tol > G_MAX_PAR or c_tol < 1:
            c_p = SETTING_DFL[0]
            c_pi = SETTING_DFL[1]
            c_tol = c_p + c_pi
    exn = bool(window_s.btn_exn.value())
    setting[10] = window_s.btn_exn.value()
    if exn:
            ref = way_1
    else:
            ref = way_0
    window_s.ipt_cp.value(c_p)
    window_s.ipt_cpi.value(c_pi)
    window_s.ipt_ps.value(p_s)
    fid_xy[0] = INDENT_WH[0] + fid_wh[0]
    fid_xy[1] = INDENT_WH[1] + fid_wh[1]
    y_spn = INDENT_WH[1] + p_s

    arr_init = []
    s = window_s.ipt_inter.value()

    b = INDENT_WH[0]
    e = fid_xy[0]
    reg = "(\d{1,3}\.\.\d{1,3})(,\s\d{1,3}\.\.\d{1,3})*"

    if re.fullmatch(reg, s):
            arr = []
            s = s.split(', ')
            q = 0
            for i in s:
                    arr += [i.split('..')]
                    q += 1
            lgh = len(arr)
            for i in range(lgh):
                    for j in range(len(arr[0])):
                            arr[i][j] = int(arr[i][j])
            for i in range(lgh):
                    if arr[i][0] < arr[i][1] and arr[i][1] < e:
                            a = arr[i][0] + INDENT_WH[0]
                            c = arr[i][1] + INDENT_WH[0]
                            if a % p_s != 0:
                                    a -= a % p_s
                                    c -= c % p_s
```

```
531                             a += INDENT_WH[0]
532                             c += INDENT_WH[0]
533                         if c > fid_xy[0]:
534                             c = fid_xy[0] - fid_xy[0] % p_s
535
536                         for j in range(a, c, p_s):
537                             arr_init += [j]
538
539             arr_init = set(arr_init)
540             arr_init = list(arr_init)
541             arr_init = sorted(arr_init)
542
543         if len(arr_init) == 0:
544             window_s.ipt_inter.value("")
545             for i in range(b, e, p_s):
546                 arr_init += [i]
547
548         inter = window_s.ipt_inter.value()
549
550         setting[0] = c_p
551         setting[1] = c_pi
552         setting[2] = int(window_s.ipt_w.value())
553         setting[3] = int(window_s.ipt_h.value())
554         setting[4] = p_s
555         setting[5] = inter
556         setting[6] = p[0]
557         setting[7] = p[1]
558         setting[8] = p[2]
559         setting[9] = p[3]
560
561         particles = []
562         count = 0
563         per = 0
564         part = Particle(y_spn, False)
565         live = True
566         v_time = time.time()
567         window_s.window.hide()
568         drawing.redraw()
569
570 # Вывод процента и времени моделирования
571 def prt_time(temp):
572         t_sec = round(time.time() - v_time, 2)
573         arr_time = [0, 0, 0]
574         arr_time[0] = int(t_sec) // 60
575         arr_time[1] = int(t_sec - (arr_time[0] * 60))
576         arr_time[2] = int((t_sec - int(t_sec)) * 100)
577         for i in range(len(arr_time)):
578             arr_time[i] = str(arr_time[i])
579             if len(arr_time[i]) < 2:
580                 arr_time[i] = '0' + arr_time[i]
581         print(f"{int(temp)}% - {arr_time[0]}:{arr_time[1]}.{arr_time[2]}")
582
583 # Основная функция
584 def core():
```

```
585     global count, fid_wh, particles, part, c_p, c_pi, y_spn, live, fid_xy, per, y_touch
586
587     if part.tch == True:
588         p_p = c_p / (c_p + c_pi)
589         p_i = 1 - p_p
590         r = rd.random()
591         if r < p_p:
592             part = Particle(y_spn, False)
593             c_p -= 1
594         elif r < p_p + p_i:
595             part = Particle(y_spn, True)
596             c_pi -= 1
597
598
599     if part.y < y_spn - (p_s * 10):
600         part.y = y_spn
601
602     part.move()
603
604     if part.tch == True:
605         count += 1
606         particles += [[part.x, part.y]]
607         arr_xy[part.y][part.x] = 1
608         temp = int((count / c_tol) * 100)
609         y_touch = part.y
610         part.inv()
611         if temp != per:
612             prt_time(temp)
613             per = temp
614         if count == c_tol:
615             live = False
616
617 def main():
618     global c_p, c_pi, fid_wh, fid_xy, p_s, inter, c_tol, y_spn, arr_init, exn, p, setting, ref
619
620     reg_inter = "(\w+\s=\s).*\s"
621     reg_0 = "(\w+\s=\s)\d+\s"
622     reg_1 = "(\w+\s=\s)((\d+\.\d+)|\d+)\s"
623     reg_cor = f"{reg_inter}|{reg_0}|{reg_1}"
624
625     if os.path.exists("setting.txt"):
626         with open(FILENAME, "r") as file:
627             j = 0
628             for line in file:
629                 if re.fullmatch(reg_cor, line):
630                     s = line.split("\n")
631                     s = s[0].split(" = ")
632                     if (re.fullmatch(reg_0, line) and j in (0, 1, 2, 3, 4, 10)) \
633                         or (re.fullmatch(reg_1, line) and j in (6, 7, 8, 9)) \
634                         or j == 5:
635                         setting[j] = type(SETTING_DFL[j])(s[1])
636                         j += 1
637                     else:
638                         print(f"Ошибка в формате данных в настройках: строка {j + 1}")
```

```
639                                 return 0
640                         else:
641                                 print(f"Ошибка в структуре файла настроек: строка {j + 1}")
642                                 return 0
643         else:
644                 with open(FILENAME, "w") as file:
645                         for i in range(len(SETTING_DFL)):
646                                 file.write(f"{SETTING_NAMES[i]} = {SETTING_DFL[i]}\n")
647                 setting = SETTING_DFL
648
649         window_s.ipt_cp.value(setting[0])
650         window_s.ipt_cpi.value(setting[1])
651         window_s.ipt_w.value(setting[2])
652         window_s.ipt_h.value(setting[3])
653         window_s.ipt_ps.value(setting[4])
654         window_s.ipt_inter.value(setting[5])
655         window_s.ipt_p_u.value(setting[6])
656         window_s.ipt_p_d.value(setting[7])
657         window_s.ipt_p_l.value(setting[8])
658         window_s.ipt_p_r.value(setting[9])
659         window_s.btn_exn.value(setting[10])
660
661         apply()
662         ref()
663
664         main_window.show()
665         main_window.end()
666
667         fl.Fl.run()
668 main()
669
670
```