**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СОГЛАСОВАНО  Доцент департамента  программной инженерии  факультета компьютерных наук  канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.З. Ахметсафина  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. |  | | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **ВИЗУАЛИЗАТОР ПОСТРОЕНИЯ КЛАССИЧЕСКОЙ ДИАГРАММЫ ВОРОНОГО**  **Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.507000-01 81 01-1-ЛУ** | | | | |
|  | |  | | |
| Исполнитель  студент группы 151 ПИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Н.А. Мариносян /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | | |
|  | | | | |
|  | | | |  |

**2017**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729.507000-01 81 01-1-ЛУ |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **ВИЗУАЛИЗАТОР ПОСТРОЕНИЯ КЛАССИЧЕСКОЙ ДИАГРАММЫ ВОРОНОГО**  **Пояснительная записка**  **RU.17701729.507000-01 81 01-1**  **Листов 26** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
|  | | | |  |

**2017**

**АННОТАЦИЯ**

В данном программном документе приведена пояснительная записка к программе «Voronoi-Diagram-Visualizer.jar» («Визуализатор построения классической диаграммы Вороного»), предназначенной для иллюстрации построения классической диаграммы Вороного алгоритмом Форчуна.

В данном программном документе, в разделе «Введение», указано наименование программы, краткое наименование программы и документы, на основании которых ведется разработка.

В разделе «Назначение и область применения» указано функциональное и эксплуатационное назначение программы и краткая характеристика области применения программы.

В данном программном документе, в разделе «Технические характеристики», содержатся следующие подразделы:

* постановка задачи на разработку программы;
* описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи и возможные взаимодействия программы с другими программами;
* описание и обоснование выбора метода организации входных и выходных данных;
* описание и обоснование выбора состава технических и программных средств.

В разделе «Ожидаемые технико-экономические показатели» указана предполагаемая потребность и экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами.

Настоящий документ разработан в соответствии с требованиями:

1. ГОСТ 19.101-77 Виды программ и программных документов [1];
2. ГОСТ 19.102-77 Стадии разработки [2];
3. ГОСТ 19.103-77 Обозначения программ и программных документов [3];
4. ГОСТ 19.104-78 Основные надписи [4];
5. ГОСТ 19.105-78 Общие требования к программным документам [5];
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [7].

Перед прочтением данного документа рекомендуется ознакомиться с терминологией, приведенной в Приложении 1 настоящего технического задания.

СОДЕРЖАНИЕ

[**1.** **ВВЕДЕНИЕ** 4](#_Toc482397479)

[**1.1.** **Наименование программы** 4](#_Toc482397480)

[**1.2.** **Документы, на основании которых ведется разработка** 4](#_Toc482397481)

[**2.** **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ** 5](#_Toc482397482)

[**2.1.** **Назначение программы** 5](#_Toc482397483)

[**2.1.1.** **Функциональное назначение** 5](#_Toc482397484)

[**2.1.2.** **Эксплуатационное назначение** 5](#_Toc482397485)

[**2.2.** **Краткая характеристика области применения** 5](#_Toc482397486)

[**3.** **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ** 7](#_Toc482397487)

[**3.1.** **Постановка задачи на разработку программы** 7](#_Toc482397488)

[**3.2.** **Описание алгоритма и функционирования программы** 7](#_Toc482397489)

[**3.3.** **Описание и обоснование выбора метода организации входных и выходных данных** 14](#_Toc482397490)

[**3.3.1.** **Описание метода организации входных и выходных данных** 14](#_Toc482397491)

[**3.3.2.** **Обоснования выбора метода организации входных и выходных данных** 14](#_Toc482397492)

[**3.4.** **Описание и обоснование выбора состава технических и программных средств** 14](#_Toc482397493)

[**3.4.1.** **Состав технических и программных средств** 14](#_Toc482397494)

[**3.4.2.** **Обоснование выбора технических и программных средств** 14](#_Toc482397495)

[**4.** **ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ** 15](#_Toc482397496)

[**4.1.** **Предполагаемая потребность** 15](#_Toc482397497)

[**4.2.** **Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами** 15](#_Toc482397498)

[**5.** **ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ** 16](#_Toc482397499)

[**ПРИЛОЖЕНИЕ 1** 17](#_Toc482397500)

[**ПРИЛОЖЕНИЕ 2** 18](#_Toc482397501)

[**ПРИЛОЖЕНИЕ 3** 19](#_Toc482397502)

[**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ** 25](#_Toc482397503)

1. **ВВЕДЕНИЕ**
   1. **Наименование программы**

Наименование программы – «Визуализатор построения классической диаграммы Вороного».

Наименование программы на английском – «Visualizer for Classic Voronoi Diagram».

Краткое наименование программы – «Voronoi-Diagram-Visualizer».

* 1. **Документы, на основании которых ведется разработка**

Разработка «Визуализатора построения классической диаграммы Вороного» ведется на основании Приказа № 2.3-02/0812-01 от 08.12.2016 «Об утверждении тем, руководителей курсовых работ студентов образовательной программы Программная инженерия факультета компьютерных наук».

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**
   1. **Назначение программы**
      1. **Функциональное назначение**

Функциональным назначением программы является построение классической диаграммы Вороного алгоритмом Форчуна с возможностью его пошагового выполнения для задаваемого пользователем набора точек на плоскости и с выбором отображаемых элементов. Другими словами, для установленного пользователем набора точек программа производит разбиение плоскости, при котором каждая область этого разбиения образует множество точек, более близких к одному из элементов множества *S*, чем к любому другому элементу множества. При этом на каждом этапе построения диаграммы программа наглядно демонстрирует пользователю текущий результат и другие вспомогательные элементы, необходимые для понимания алгоритма построения.

* + 1. **Эксплуатационное назначение**

Построение диаграммы Вороного является востребованным на сегодняшний день, так как существует множество задач, при решении которых необходимо для заданного набора точек *S* произвести разбиение плоскости таким образом, что каждая область этого разбиения образует множество точек, более близких к одному из элементов множества *S*, чем к любому другому элементу множества. Построение диаграммы Вороного может быть использовано для решения многих потенциально полезных задач [16], например:

1. В геолокационном рекомендательном программном обеспечении для определения анализа местоположения пользователя и поиска необходимых к нему ближайших объектов, например продуктового магазина;
2. Создания системы навигации в робототехнике;
3. Cоздания игрового движка;
4. Очерчивания границ регионов в картографии и дальнейшего анализа на их основе.

Для ознакомления с полным списком применений построения диаграммы Вороного рекомендуется перейти по ссылке, указанной в [15].

Разрабатываемый программный продукт может быть использован в учебных заведениях для изучения диаграммы Вороного и алгоритма Форчуна. В частности, в рамках дисциплины «Алгоритмы и структуры данных» на факультете компьютерных наук НИУ ВШЭ.

Код программы может быть использован другими разработчиками для решения других задач и создания потенциально полезных приложений.

* 1. **Краткая характеристика области применения**

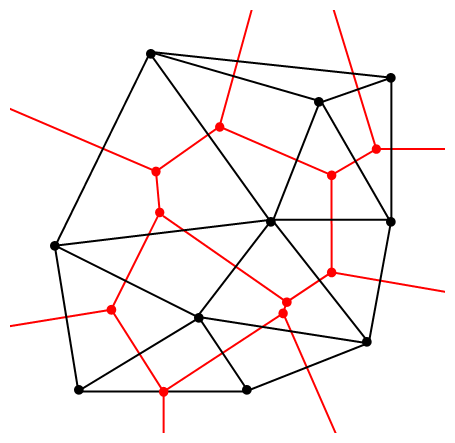
«Визуализатор построения классической диаграммы Вороного» – программа, предназначенная для иллюстрации построения классической диаграммы Вороного алгоритмом Форчуна, с возможностью его пошагового выполнения для задаваемого пользователем набора точек на плоскости и с выбором отображаемых элементов.

«Визуализатор построения классической диаграммы Вороного» может использоваться для изучения построения диаграммы Вороного в учебных заведениях. Построение диаграммы Вороного является востребованным на сегодняшний день, так как существует множество областей ее применения, среди них:

* Моделирование;
* Распознавание образов;
* Разработка игр;
* Картография;
* Геолокация;
* Археология;
* Биология;
* Химия;
* Вычислительная механика;
* Геология;
* Геофизика

и многие другие [15].

Также диаграмма Вороного имеет тесную связь и взаимнооднозначное соответствие с триангуляцией Делоне (рис. 1).



*Рисунок 1 – Триангуляция Делоне с соответствующей ей диаграммой Вороного (выделена красным)*

1. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**
   1. **Постановка задачи на разработку программы**

Разрабатываемая программа должна поддерживать следующие функции:

1. Редактирование области построения диаграммы:

–– создание сайта (см. Приложение 1) путем нажатия пользователем ПКМ по выбранному свободному месту на области построения диаграммы *перед началом выполнения построения диаграммы*;

–– создание сайта (см. Приложение 1) путем нажатия пользователем ПКМ по выбранному свободному месту на области построения диаграммы, правее заметающей прямой, *после начала выполнения построения диаграммы*;

–– очистка области построения диаграммы и установка заметающей прямой в начальное положение;

–– выбор отображаемых элементов: построенной диаграммы Вороного, береговой линии и событий круга.

2) Управление процессом построения диаграммы:

–– запуск/возобновление выполнения алгоритма построения диаграммы Вороного;

–– прерывание выполнения алгоритма построения диаграммы Вороного;

–– перезапуск процесса построения диаграммы Вороного;

–– переход к следующему событию алгоритма: событие круга или событие точки (см. Приложение 1);

–– совершение сдвига заметающей прямой на один пиксель вправо.

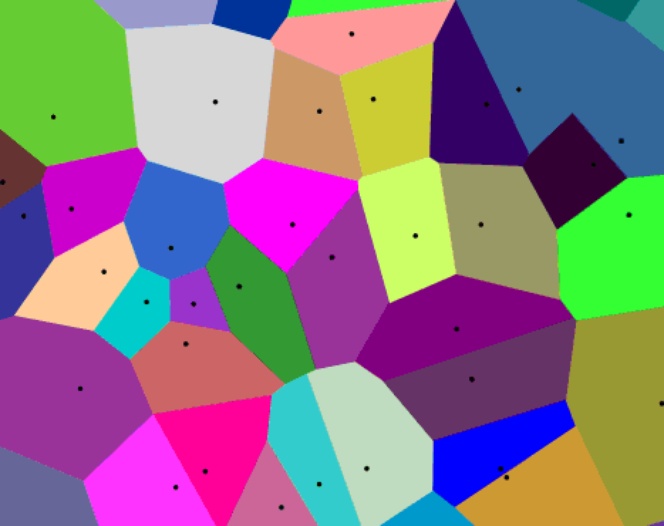
3) Визуализация построения:

–– визуализация необходимых для понимания алгоритма элементов, в том числе выбираемых пользователем, в каждый момент времени во время процесса построения классической диаграммы Вороного;

–– отображение построенной классической диаграммы Вороного в области построения диаграммы главного окна приложения.

* 1. **Описание алгоритма и функционирования программы**

Диаграмма Вороного конечного множества точек точек на плоскости представляет такое разбиение плоскости, при котором каждая область этого разбиения образует множество точек, более близких к одному из элементов множества *S*, чем к любому другому элементу множества. На рисунке ниже (рис. 2) изображен пример диаграммы Вороного.



*Рисунок 2 – Диаграмма Вороного*

Линии, образующие границы локусов (см. Приложение 1), называются ребрами диаграммы Вороного. А точки пересечения ребер – вершинами диаграммы Вороного.

Для простоты понимания можно сформулировать неформальное определение диаграммы Вороного. Пусть есть карта города, на которой точками обозначены почтовые отделения. Человек хочет отправить письмо, и он пойдет на ближайшую почту. Ему интересно знать, какое отделение ближе, для любой точки города — необходимость отправить письмо может наступить неожиданно. Для этого он может взять карту и расчертить ее на ячейки так, чтобы внутри каждой ячейки находилось только одно отделение, а для всех остальных точек ячейки именно эта почта была ближайшей. Полученная картинка и будет диаграммой Вороного для точек-почт.

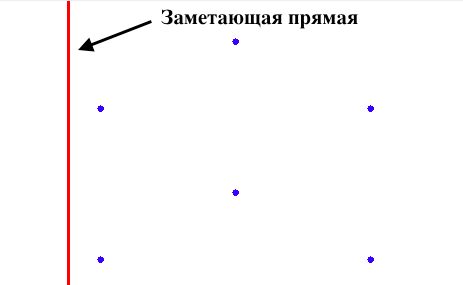
Существует несколько алгоритмов построения диаграммы Вороного:

* Алгоритм построения диаграммы Вороного «в лоб»;
* Алгоритм построения диаграммы Вороного путем пересечения полуплоскостей.;
* Алгоритм Форчуна построения диаграммы Вороного на плоскости;
* Рекурсивный алгоритм построения диаграммы Вороного.

Для построения классической диаграммы Вороного в программе использован алгоритм Стива Форчуна (англ. Steve Fortune), предложенного им в 1987 году. Такой выбор обусловлен тем, что данный алгоритм обладает лучшим показателем временной сложности, но при этом не очень сложен в понимании и реализации.

* + 1. **Описание алгоритма Форчуна**

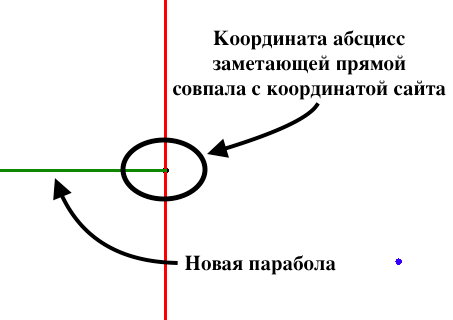
Основная идея алгоритма — это так называемая заметающая прямая, которая позволяет удобно моделировать движение прямой по некоторому множеству сайтов (см. Приложение 1). При этом влияние на построение диаграммы производят только те сайты (см. Приложение 1), которые лежат правее или на заметающей прямой (рис. 3).



*Рисунок 3 – Заметающая прямая, моделирующая движение по множеству сайтов*

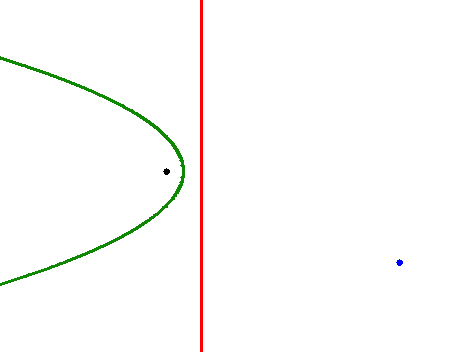
В данной программной реализации заметающая прямая плавно движется по области построения диаграммы слева направо, поэтому создается впечатления непрерывности вычислений. Однако на самом деле при построении диаграммы Вороного алгоритмом Форчуна все вычисления происходят дискретно, поскольку в алгоритме построения диаграммы присутствуют всего два типа событий: точки и круга.

Событие точки происходит, когда координаты абсцисс заметающей прямой и очередного сайта совпадают. В этот момент времени создается новая парабола - геометрическое место точек, равноудаленных от заметающей линии и данного сайта (изначально это прямая линия) (рис. 4).



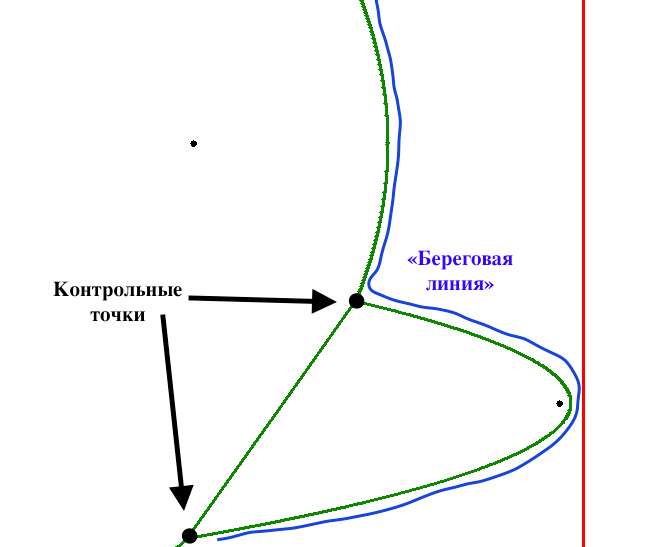
*Рисунок 4 – Событие точки*

Эта парабола в каждый момент времени своего существования разделяет область построения на две части — внутренняя область параболы содержит в себе точки, которые в данный момент времени лежат ближе к соответствующему сайту, а внешняя — точки, которые ближе к заметающей прямой (рис. 5).



*Рисунок 5 – Парабола, разделяющая область построения на две части*

Вместе с продвижением заметающей прямой вправо по области построения диаграммы, парабола расширяется и в какой-то момент образует пересечения с другими параболами (рис. 6), называемые контрольными точками. Вместе все дуги парабол от одной точки пересечения их друг с другом до другой образуют «береговую линию» (рис. 6).



*Рисунок 6 – Контрольные точки и береговая линия*

Алгоритм Форчуна моделирует движение этой «береговой линии», т.к. любые две контрольные точки равноудалены от обоих сайтов которым соответствуют эти параболы, и от заметающей прямой (рис. 7, для простоты восприятия показана равноудаленность только одного из сайтов). Поэтому эти контрольные точки двигаются по ребрам диаграммы Вороного (прямым, образующем локусы [см. Приложение 1] диаграммы Вороного) и фактически «рисуют» их.

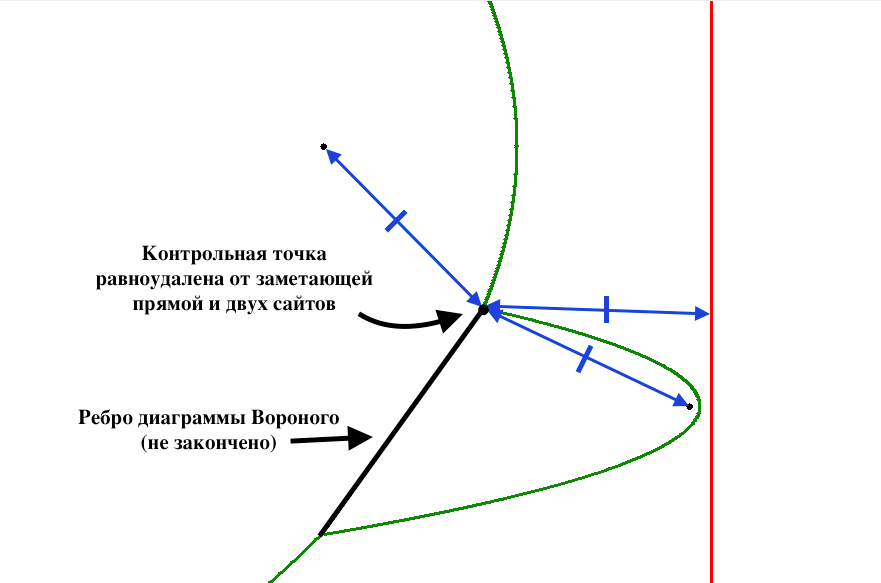
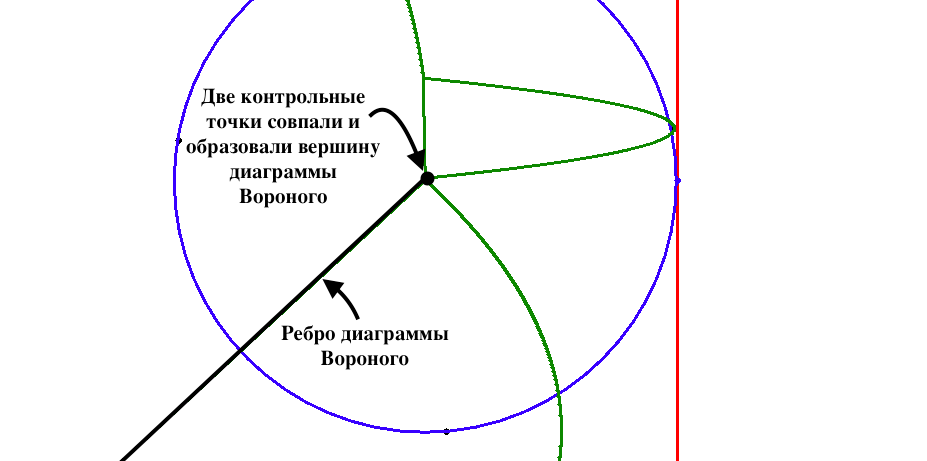


Рисунок 7 – Равноудаленность сайтов и ребро диаграммы Вороного.

Событие круга происходит, когда координаты двух контрольных точек полностью совпадают, а дуга, которая находилась между ними, пропадает (рис. 8). В этот момент точка, в которую «слились» эти две контрольные точки, становится вершиной диаграммы Вороного. После этого, соединив прямой линией образованную точку с предыдущей, соответствующей ей, мы получаем ребро диаграммы Вороного (если такой точки еще нет, то мы должны продолжить ребро до границы области построения диаграммы) (рис. 8).



*Рисунок 8 – Событие круга и законченное ребро диаграммы Вороного*

* + 1. **Особенности программной реализации алгоритма**

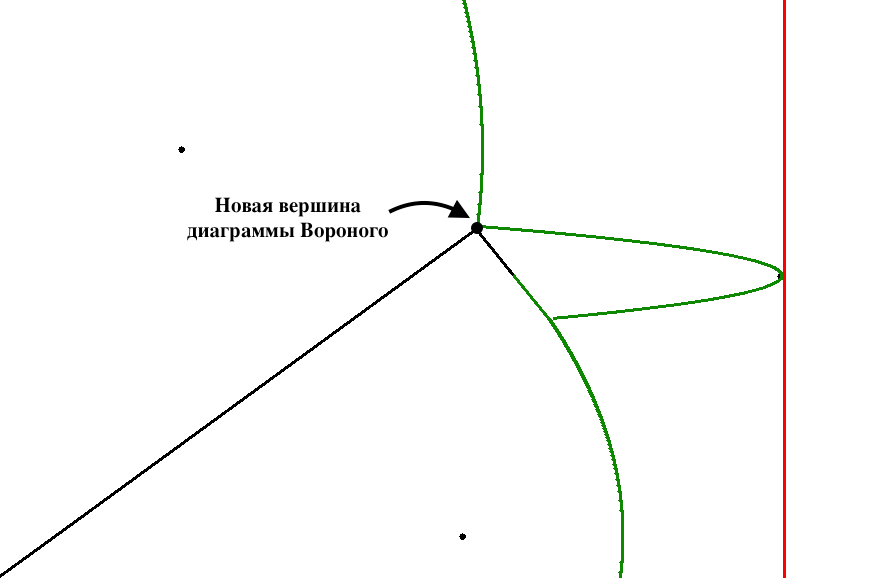
Программа моделирует движение заметающей прямой. При ее движении слева направо происходит два типа событий:

* Событие точки — попадание заметающей прямой на один из сайтов (рис. 4). Создается новая парабола для данного сайта и возникают две контрольные точки (сначала — одна, т.к. изначально это прямая, а при расширении параболы – две). При этом арка – часть параболы, принадлежащая «береговой линии» — вставляется в «береговую линию» только во время события точки;
* Событие круга — это возникновение новой вершины диаграммы Вороного вместе с удалением одной арки (рис. 8). Когда происходит это событие, существует три некоторых арки: верхняя, средняя и нижняя. Средняя исчезает и удаляется из «береговой линии» вследствие сближения верхней и нижней контрольных точек арок (рис 9).



*Рисунок 9 – Сближение контрольных точек и «исчезновение» средней арки*

Создается новая вершина диаграммы Вороного (рис. 10). Арка удаляется из «береговой линии» только в случае события круга.

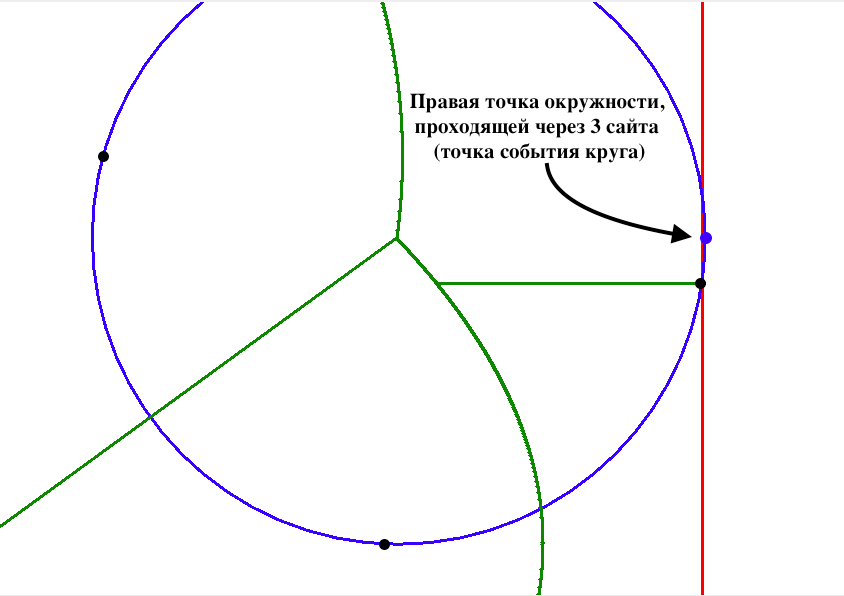


*Рисунок 10 – Новая вершина диаграммы Вороного*

Все события точки и круга (см. Приложение 1) добавляются в общую очередь с приоритетом относительно значения координаты по оси абсцисс. Таким образом, двигаясь дискретно от события к событию с помощью заметающей прямой, выполняется процесс построения диаграммы.

В программе для события круга используется теорема, согласно которой вершина диаграммы Вороного всегда лежит на пересечении ровно трех ребер диаграммы, и следствие из этой теоремы, которое говорит о том, что вершина диаграммы является центром окружности, проходящей через три сайта и расстояние от этой точки до заметающей прямой тоже равно радиусу этой окружности.

Таким образом, когда самая правая точка окружности, проходящей через три сайта (см. Приложение 1), лежит правее или на заметающей прямой (рис. 11), в очередь событий добавляется событие круга с этой правой точкой окружности.



*Рисунок 11 – Правая точка окружности, проходящей через 3 сайта*

И когда заметающая прямая на нее попадет, произойдет событие круга и создастся новая вершина диаграммы Вороного.

В программе существует взаимооднозначное соответствие между событиями (круга или точки) и арок, что помогает легко обрабатывать события.

Общий алгоритм работы программы следующий:

1. Создается очередь с приоритетом для событий:

VEventQueue() **Events = new** VEventQueue();

(см описание и код класса VEventQueue в Приложении 2 и Приложении 3).

Приоритет в очереди отдается событию, точка наступления которого имеет наименьшее значение координаты по оси абсцисс.

1. В очередь для событий добавляются события точки, создаваемыми на основе заданного пользователем множества сайтов:

**for**(**int** i = 0; i < **Voronoi**.size(); i++)  
 **Events**.insert(**new** VEventPoint((VPoint)**Voronoi**.elementAt(i)));

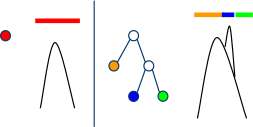
Здесь Voronoi – экземпляр класса VoronoiClass, который содержит информацию о сайтах (см. Приложение 1), ребрах и вершинах диаграммы Вороного.

1. Пока очередь не пуста:
2. Из очереди берется очередное событие и вызывается функция его обработки (метод performAction() определен в обоих классах события точки и круга):

VEventPoint vEventPoint = **Events**.pop();

vEventPoint.performAction(**this**);

1. Если это событие точки, то обрабатываем событие точки: ищем место в двоичном дереве «береговой линии» и добавляем в него новую арку (рис. 12) и контрольные точки (см. коды классов ArcNode, ArcTree и ParabolaPoint).



*Рисунок 12 – Бинарное дерево «Береговой линии»*

1. Если это событие круга, то обрабатываем событие круга. Определяем центр и самую правую точку окружности, проходящую через три сайта (см. метод calculateCenter() класса ParabolaPoint). События круга проверяются и добавляются в очередь для событий автоматически с помощью методов классов ParabolaPoint и ArcNode: calculateCenter(), checkCircle() и других вспомогательных (см. коды классов ParabolaPoint и ArcNode в Приложении 3).
2. Визуализируем произошедшие изменения с помощью метода класса VCanvas repaint():
3. Закончить все оставшиеся ребра. После обработки последнего события остается одно или несколько незаконченных ребер, которые программа достраивает до границ области построения.

При реализации были использованы следующие структуры данных:

* Очередь с приоритетом, содержащая события. (см. класс VEventQueue в Приложении 1);
* Двоичное дерево для хранения «береговой линии» (текущего положения парабол и контрольных точек). При этом у любого узла либо ровно два «ребенка», либо ноль (рис. 12). Данная структура данных реализована в классе ArcTree.

Временная сложность алгоритма Форчуна составляет O(n \* log(n)), но из-за необходимости визуализации время построения диаграммы возрастает.

* + 1. **Краткое описание работы пользователя с программой**

Пользователь задает набор сайтов (см. Приложение 1) путем нажатия правой кнопки указывающего устройства по области построения диаграммы. Затем, нажимая кнопки управления созданием диаграммы Вороного, пользователь может наблюдать за процессом выполнения построения и контролировать его. Подробнее с функционалом приложения и инструкцией по работе с ним можно ознакомиться в Руководстве оператора к данному программного продукту.

* 1. **Описание и обоснование выбора метода организации входных и выходных данных**
     1. **Описание метода организации входных и выходных данных**

Входными данными для программы можно считать сайты (см. Приложение 1), устанавливаемые пользователем в области построения диаграммы.

Выходными данными программы можно считать визуализацию построения диаграммы в каждый момент времени и уже готовую, нарисованную диаграмму Вороного после выполнения алгоритма построения.

* + 1. **Обоснования выбора метода организации входных и выходных данных**

Диаграмма Вороного по определению строится по сайтам (см. Приложение 1). Такой метод входных и выходных данных обусловлен удобством работы пользователя. Для понимания алгоритма построения диаграммы пользователю удобнее работать только с самой программой, не выходя за ее пределы. Поэтому программа не создает никаких выходных файлов и не требует никаких файлов на вход.

* 1. **Описание и обоснование выбора состава технических и программных средств**
     1. **Состав технических и программных средств**

Для надежной и бесперебойной работы программы требуется следующий состав технических средств [12]:

1) персональный компьютер, оснащенный 32-разрядным (x86) или 64-разрядным (x64) процессором с тактовой частотой 1 ГГц или выше;

2) 1 ГБ (для 32-разрядной системы) или 2 ГБ (для 64-разрядной системы) оперативной памяти или больше;

3) не менее 2 ГБ свободного места на жестком диске;

4) видеокарта и монитор, поддерживающие режим SVGA с разрешением не менее чем 1080x720 точек;

5) совместимое указывающее устройство;

6) клавиатура;

7) CD-ROM привод;

8) установленная программная платформа JRE 1.4 или более поздняя.

* + 1. **Обоснование выбора технических и программных средств**

При реализации программы использовался язык Java 1.8. Однако программа поддерживает и более старые версии, но не ниже 1.4. Поэтому для надежной и бесперебойной работы указываются характеристики, соответствующие этим версиям программных средств, использованных при реализации. Разрешение монитора не должно быть меньше размера окна приложения, чтобы оно смогло поместить на экране компьютера пользователя. Необходим CD-ROM привод для считывания компакт-диска, на котором поставляется данный программный продукт. Остальной набор технических средств обуславливается минимальными требованиями для работы с операционной системой.

1. **ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**
   1. **Предполагаемая потребность**

Построение классической диаграммы Вороного довольно востребовано во множестве областей науки и производства. Программу могут использовать все, кто нуждается в построении диаграммы Вороного для заданного набора точек на плоскости. Также программа может быть использована преподавателями и обучающимися в учебных заведениях для изучения процесса построения диаграммы Вороного с помощью алгоритма Форчуна.

* 1. **Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами**

Данное приложение:

* является бесплатным;
* работает на всех платформах с JRE 1.4 и выше;
* может свободно распространяться;
* не требует вложения денежных средств во время использования;
* имеет неограниченный срок службы;
* может быть модифицировано другими лицами в том случае, если они согласуют детали с разработчиком, который, в свою очередь, предоставит им доступ к исходному коду данного программного продукта.

1. **ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ**
2. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
11. ГОСТ 15150-69 Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды. – М.: Изд-во стандартов, 1997.
12. Устинов В. Надежность оптических дисков: как их правильно хранить и использовать. //Журнал «625» №7. М.: Издательство «625», 2005.
13. Системные требования ОС Windows 7. [Электронный ресурс]// URL: <http://windows.microsoft.com/systemrequirements?4bcfd458> (Дата обращения: 11.03.2016, режим доступа: свободный).
14. ГОСТ Р 7.02-2006 Консервация документов на компакт-дисках. Общие требования. – М.: ИПК Издательство стандартов, 2006.
15. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
16. Применения диаграммы Вороного. [Электронный ресурс]// URL: [https://www.voronoi.com/wiki/index.php?title=Voronoi\_Applications - General\_applications](https://www.voronoi.com/wiki/index.php?title=Voronoi_Applications#General_applications) (Дата обращения: 03.05.2017, режим доступа: свободный).
17. Диаграмма Вороного и ее применения. [Электронный ресурс]// URL: <https://habrahabr.ru/post/309252> (Дата обращения: 04.05.2017, режим доступа: свободный).
18. Статья Стива Форчуна «A sweepline algorithm for Voronoi diagrams». [Электронный ресурс]// URL: <http://link.springer.com/article/10.1007%2FBF01840357> (Дата обращения: 04.05.2017, режим доступа: свободный).

# ПРИЛОЖЕНИЕ 1

**ТЕРМИНОЛОГИЯ**

Ниже приведен список необходимых терминов для ознакомления:

* Арка – часть параболы, принадлежащая «береговой линии» — вставляется только во время события точки;
* Береговая линия – кривая, образованная одной или пересечением нескольких парабол, которые, в свою очередь, были созданы во время события точки;
* Контрольные точки – точки пересечения парабол на береговой линии [19].
* Локус – область, в которой присутствуют все точки, которые находятся ближе к данной точке, чем ко всем остальным [19].
* Сайт (англ. site) – точка, для которой строится локус [19].
* Событие круга – момент, когда две контрольные точки – по одной из разных парабол совмещаются (превращаются в одну). Эта точка и становится вершиной ячейки Вороного, причем в это время та дуга, которая находилась между этими двумя точками — «схлопывается» и удаляется из береговой линии [19].
* Событие точки – событие, которое происходит, когда заметающая прямая попадает на очередной сайт и создается новая парабола, фокусом которой является данный сайт, а директрисой – заметающая прямая [19].

**ПРИЛОЖЕНИЕ 2**

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ**

Таблица 3.1 – Описание и функциональное назначение классов

|  |  |
| --- | --- |
| Класс | Назначение |
| ArcNode | Класс вершины бинарного дерева, хранящего в себе информацию об арках. |
| ArcTree | Класс бинарного дерева, хранящего в себе информацию о «береговой линии». |
| CirclePoint | Класс, представляющий собой событие круга. |
| ParabolaPoint | Класс, представляющий собой контрольную точку. |
| VCanvas | Класс, представляющий собой область построения диаграммы Вороного со всеми необходимыми утилитарными функциями. |
| VEventPoint | Класс, представляющий собой событие точки. |
| VEventQueue | Класс, представляющий собой очередь с приоритетом для событий точки и круга. |
| VLine | Класс, представляющий собой прямую линию, разработанный специально для удобства работы с диаграммой Вороного. |
| VoronoiClass | Класс, представляющий собой структуру диаграммы Вороного. |
| VPaintable | Утилитарный интерфейс для визуализации элементов. |
| VPoint | Класс, представляющий собой точку (сайт), разработанный специально для удобства работы с диаграммой Вороного. |
| Utilities | Утилитарный класс, содержащий несколько вспомогательных функций. |
| Controls | Класс для создания панели кнопок управления процессом построения диаграммы Вороного. |
| Settings | Класс для создания панели выбора отображаемых элементов. |
| VDFrame | Класс главного окна программы, с которым производит работу пользователь. |

**ПРИЛОЖЕНИЕ 3**

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ, МЕТОДОВ И СВОЙСТВ**

Таблица 4.1 – Описание полей, методов и свойств класса ArcNode.java

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| Next | | package private | | ArcNode | | Ссылка на следующую вершину дерева. | |
| Prev | | package private | | ArcNode | | Ссылка на предыдущую вершину дерева. | |
| circlePoint | | private | | CirclePoint | | Ссылка на точку события круга. | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| checkCircle | package private | | void | | VEventQueue vEventQueue | | Метод для проверки события круга и добавления его в очередь событий. |
| removeCircle | private | | void | | VEventQueue vEventQueue | | Метод для удаления события круга из очереди событий. |
| completeTrace | package private | | void | | VCanvas vCanvas, VPoint mypoint | | Метод для добавления последовательности точек в область построения диаграммы. |
| checkBounds | package private | | void | | VCanvas vCanvas, double d | | Метод проверки правильности координат сайта относительно области построения диаграммы. |
| paint | package private | | void | | Graphics g, double d, double d1, boolean flag, boolean drawBeachline | | Метод для отрисовки элементов построения диаграммы. |

Таблица 4.2 – Описание полей, методов и свойств класса ArcTree.java

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | |
| Имя | Модификатор доступа | | Тип | | Назначение | |
| Arcs | private | | Arcnode | | Дерево арок «береговой линии». | |
| **Методы** | | | | | | |
| Имя | Модификатор доступа | Тип | | Аргументы | | Назначение |
| insert | package private | void | | VPoint mypoint, double d, VEventQueue eventqueue | | Метод вставки контрольной точки «береговой линии». |
| checkBounds | package private | void | | VCanvas vCanvas, double d | | Метод проверки границ области построения диаграммы и «береговой линии». |
| paint | package private | void | | Graphics g, double d, boolean flag, boolean drawBeach | | Метод рисования береговой линии. |

Таблица 4.3 – Описание полей, методов и свойств класса CirclePoint.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| radius | public | double | Радиус окружности события круга. | |
| arc | public | ArcNode | Ссылка на дугу «береговой линии». | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| paint | public | void | Метод рисования события круга. |
| performAction | public | void | Метод для обработки события круга. |

Таблица 4.4 – Описание полей, методов и свойств класса ParabolaPoint.java

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| a | | package private | | double | | Координата точки | |
| b | | package private | | double | | Координата точки | |
| c | | package private | | double | | Координата точки | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| RealX | package private | | double | | – | | Возвращает координату точки по оси абсцисс. |
| RealY | package private | | double | | double d | | Возвращает координату точки по оси ординат. |
| calculateCenter | package private | | CirclePoint | | int index | | Метод подсчета координат новой вершины диаграммы. |

Таблица 4.5 – Описание полей, методов и свойств класса VCanvas.java

|  |  |  |  |
| --- | --- | --- | --- |
| **Поля** | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| offScreenGraphics | private | Graphics | Список аккаунтов. |
| offScreenImage | private | BufferedImage | Значение максимального допустимого отличия в почерках. |
| Voronoi | package private | VoronoiClass | Переменная для хранения построенной диаграммы Вороного. |
| Events | package private | VEventQueue | Очередь событий алгоритма Форчуна. |
| Arcs | package private | ArcTree | Дерево дуг «береговой линии». |
| **Методы** | | | |
| paint | public | Graphics g | Метод рисования элементов, необходимых для понимания процесса построения диаграммы. |
| update | public | Graphics g | Метод для обновления изменений в области построения диаграммы. |
| singleStep | public | – | Метод сдвига заметающей прямой на 1 пиксель вправо. |
| step | public | – | Метод перехода к следующему событию построения диаграммы. |
| clear | public | – | Метод очистки области построения диаграммы. |
| restart | public | – | Метод перезапуска процесса построения диаграммы. |
| mousePressed | public | – | Метод добавления сайта путем нажатия правой кнопки указывающего устройства. |

Таблица 4.6 – Описание полей, методов и свойств класса VEventPoint.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| Prev | package private | VEventPoint | Ссылка на предыдущую точку | |
| Next | package private | VEventPoint | Ссылка на следующую точку | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| insert | package private | void | VEventPoint vEventPoint | Метод вставки события. |
| performAction | public | void | VCanvas mycanvas | Метод для обработки события. |

Таблица 4.7 – Описание полей, методов и свойств класса VEventQueue.java

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | |
| Имя | Модификатор доступа | | Тип | | Назначение | |
| Events | package private | | VEventPoint | | Список событий алгоритма Форчуна. | |
| **Методы** | | | | | | |
| Имя | Модификатор доступа | Тип | | Аргументы | | Назначение |
| insert | package private | void | | VEventPoint vEventPoint | | Метод вставки события. |
| remove | package private | void | | VEventPoint vEventPoint | | Метод удаления события. |
| pop | package private | void | | Keys k | | Стандартный метод pop для списка событий. |
| paint | package private | void | | Graphics g, boolean flag | | Метод отрисовки элементов. |

Таблица 4.8 – Описание полей, методов и свойств класса VLine.java

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| P1 | | private | | VPoint | | Начальная точки отрезка. | |
| P2 | | private | | VPoint | | Конечная точка отрезка. | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| paint | public | | void | | Graphics g | | Метод рисования отрезка между двумя точками. |

Таблица 4.9 – Описание полей, методов и свойств класса VoronoiClass.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| checkDegenerate | package private | void | – | Метод проверки наложения одного сайта на другой. |
| paint | package private | void | Graphics g | Метод рисования ребер диаграммы Вороного. |
| clear | public | void | – | Метод удаления диаграммы Вороного. |

Таблица 4.10 – Описание полей, методов и свойств интерфейса VPaintable.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| paint | package private | void | Graphics g | Абстрактный метод для рисования. |

Таблица 4.11 – Описание полей, методов и свойств класса VPoint.java

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| x | | volatile double | | double | | Координата точки по оси абсцисс. | |
| y | | volatile double | | double | | Координата точки по оси ординат. | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| paint | public | | void | | Graphics g | | Метод для рисования точки. |
| getDistance | package private | | double | | VPoint other | | Метод подсчета расстояния между данной точкой, и переданной через параметры. |

Таблица 4.12 – Описание полей, методов и свойств класса Utilities.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| openWebpage | public | void | URI uri, JComponent component | Метод открытия веб-страницы. |
| readAllTextFromResource | public | void | String resourceName | Метод считывания текста из ресурса для руководства пользователя. |

Таблица 4.13 – Описание полей, методов и свойств класса Controls.java

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| canvas | | private | | VCanvas | | Область построения диаграммы. | |
| running | | private | | boolean | | Переменная, показывающая статус исполнения треда. | |
| buttons | | package private | | JButton[] | | Массив кнопок управления процессом построения диаграммы. | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| actionPerformed | public | | void | | ActionEvent actionevent | | Метод обработки события нажатия на кнопки управления процессом построения диаграммы. |
| threadRunning | package private | | void | | boolean flag | | Метод для контролирования активированных кнопок управления процессом построения диаграммы. |

Таблица 4.14 – Описание полей, методов и свойств класса Settings.java

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| canvas | | private | | VCanvas | | Область построения диаграммы. | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| itemStateChanged | public | | void | | ItemEvent itemEvent | | Метод обработки события изменения отображения вспомогательных элементов. |

Таблица 4.15 – Описание полей, методов и свойств класса VDFrame.java

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| canvas | | private | | VCanvas | | Область построения диаграммы. | |
| controls | | private | | Controls | | Панель управления процессом построения диаграммы. | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| init | private | | void | | – | | Метод инициализации. |
| start | private | | void | | – | | Метод запуска потока. |
| run | public | | void | | – | | Метод исполнения потока. |
| onPaint | private | | void | | – | | Добавление интерфейса при запуске. |
| createMenu | private | | void | | – | | Метод создания меню. |
| openAbout | public | | void | | – | | Метод открытия окна с краткой информацией о программе. |

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированх |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |