|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл | RU.17701729.04.13-01 81 01-1-ЛУ |

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Доцент департамента больших  данных и информационного поиска  факультета компьютерных наук,  к.ф.-м.н..  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. Л. Чернышев  «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 г. | УТВЕРЖДАЮ  Академический руководитель  образовательной программы  «Программная инженерия»  профессор департамента  программной инженерии, канд.  техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Шилов  «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

**ПРОГРАММА ДЛЯ МОДЕЛИРОВАНИЯ ДВИЖЕНИЯ ТОЧЕК НА**

**ОРИЕНТИРОВАННОМ МЕТРИЧЕСКОМ ГРАФЕ, С УСЛОВИЕМ**

**СИНХРОНИЗАЦИИ В ВЕРШИНАХ**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.13-01 81 01-1-ЛУ**

Исполнитель

студент группы БПИ196

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А. А. Баранова /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл | RU.17701729.04.13-01 81 01-1-ЛУ |

УТВЕРЖДЕНRU.17701729.04.13-01 81 01-1-ЛУ

**ПРОГРАММА ДЛЯ МОДЕЛИРОВАНИЯ ДВИЖЕНИЯ ТОЧЕК НА**

**ОРИЕНТИРОВАННОМ МЕТРИЧЕСКОМ ГРАФЕ, С УСЛОВИЕМ**

**СИНХРОНИЗАЦИИ В ВЕРШИНАХ**

**Пояснительная записка**

**RU.17701729.04.13-01 81 01-1**

**Листов 59**

СОДЕРЖАНИЕ

[1. ВВЕДЕНИЕ 4](#_Toc40706455)

[1.1. Наименование программы 4](#_Toc40706456)

[1.2. Документы, на основании которых ведется разработка 4](#_Toc40706457)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 5](#_Toc40706458)

[2.1. Назначение программы 5](#_Toc40706459)

[2.1.1. Функциональное назначение 5](#_Toc40706460)

[2.1.2. Эксплуатационное назначение 5](#_Toc40706461)

[2.2. Краткая характеристика области применения 5](#_Toc40706462)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 7](#_Toc40706463)

[3.1. Постановка задачи на разработку программы 7](#_Toc40706464)

[3.2. Описание алгоритма и функционирования программы 7](#_Toc40706465)

[3.2.1. Описание используемых алгоритмов 7](#_Toc40706466)

[3.2.2. Обоснование выбора алгоритма решения задачи. 12](#_Toc40706467)

[3.2.3. Возможные взаимодействия программы с другими программами. 13](#_Toc40706468)

[3.3. Описание и обоснование выбора метода организации входных и выходных данных. 13](#_Toc40706469)

[3.3.1. Описание метода организации входных и выходных данных. 13](#_Toc40706470)

[3.3.2. Обоснования выбора метода организации входных и выходных данных. 13](#_Toc40706471)

[3.4. Описание и обоснование выбора состава технических и программных средств. 14](#_Toc40706472)

[3.4.1. Состав технических и программных средств. 14](#_Toc40706473)

[3.4.2. Обоснование выбора технических и программных средств. 14](#_Toc40706474)

[4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 16](#_Toc40706475)

[4.1. Предполагаемая потребность. 16](#_Toc40706476)

[4.2. Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами. 16](#_Toc40706477)

[5. СПИСОК ЛИТЕРАТУРЫ 17](#_Toc40706478)

[ПРИЛОЖЕНИЕ 1 19](#_Toc40706479)

[ПРИЛОЖЕНИЕ 2 20](#_Toc40706480)

[ПРИЛОЖЕНИЕ 3 21](#_Toc40706481)

[ПРИЛОЖЕНИЕ 4 23](#_Toc40706482)

[ПРИЛОЖЕНИЕ 5 25](#_Toc40706483)

[ПРИЛОЖЕНИЕ 6 26](#_Toc40706484)

[ПРИЛОЖЕНИЕ 7 30](#_Toc40706485)

# **ВВЕДЕНИЕ**

* 1. **Наименование программы**

Наименование программы – «Программа для моделирования движения точек на ориентированном метрическом графе, с условием синхронизации в вершинах».

«The Program for Modeling the Movement of Points on Directed Metric Graph, with the Condition of Synchronization at the Vertices».

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

Основанием для разработки является приказ декана факультета компьютерных наук И.В. Аржанцева "Об утверждении тем, руководителей курсовых работ студентов образовательной программы «Программная инженерия» факультета компьютерных наук" № 2.3-02/1112-04 от 11.12.2019.

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

Функциональным назначением разработанного приложения является работа с выбранным пользователем из архива ранее созданных, либо новым, построенным им в специальном редакторе, сильно связным ориентированным метрическим графом с целью моделирования движения на нем точек, с условием синхронизации в вершинах разных типов, а также получения минимального анализа поведения этих точек и визуализации происходящих на графе процессов.

* + 1. **Эксплуатационное назначение**

Программа может быть использована преподавателями, студентами или исследователями, работающими в различных областях математики и информатики при рассмотрении разнообразных задач на графах, или нейробиологии - при моделировании различных реальных процессов в нейронных сетях с использованием данной математической модели. Таким образом, продукт разработки позволит решать задачи, возникающие при составлении теоретических моделей в ходе научных исследований.

* 1. **Краткая характеристика области применения**

«Программа для моделирования движения точек на ориентированном метрическом графе, с условием синхронизации в вершинах» – это программа, позволяющая строить соответствующие графы и моделировать на таких сильно связных графах движение точек двух разных типов: стандартного (с синхронизацией в вершинах по установленным порогам [см. Приложение 3]) и модели песка [см. Приложение 2] (с синхронизацией по степени вершины); получать визуализацию процесса распространения точек на графе, изображение зависимости числа точек от времени в виде графика, изображение графика распределения размеров лавин [см. Приложение 2] при моделировании движения песка; сохранять анимацию процесса в формате GIF-изображения.

Данная программа несет научно-образовательный характер и может использоваться исследователями, работающими в различных областях математики и информатики при рассмотрении разнообразных задач на графах или при изучении абелевой модели песка, области, содержащей много открытых проблем и интенсивно развивающейся. Программа также потенциально может найти применение в нейробиологии при моделировании различных процессов в нейронных сетях с использованием данной математической модели.

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

Разрабатываемая программа должна:

1. иметь отдельное поле для построения ориентированного метрического графа без петель, осуществлять на нем отрисовку графа и его редактирование;
2. выполнять сохранение построенного графа в файл формата .dgmm [см. Приложение 5];
3. выполнять открытие ранее созданного в этой программе графа из файла формата .dgmm;
4. производить моделирование движения точек на графе с условием синхронизации в вершинах по выбранному правилу, демонстрировать анимацию этого движения, предоставлять пользователю дополнительную запрошенную информацию;
5. реализовывать возможность сохранения полученной в ходе моделирования визуализации процесса.
   1. **Описание алгоритма и функционирования программы**
      1. **Описание используемых алгоритмов**
         1. **Алгоритм моделирования движения точек на ориентированном метрическом графе с условием синхронизации в вершинах**

Возможность моделирования движения точек предоставляется членами класса MovementModeling [см. Приложение 7, Таблица 2.20]. В класс передаются сильно связный граф, тип моделирования, список выбранных дополнительных действий, а также ссылка на поверхность рисования и экземпляр GraphDrawing [см. Приложение 7, Таблица 2.18] для отрисовки графа.

После вызова метода, запускающего движение, производится обработка переданных данных и непосредственно запуск моделирования. Далее управление передается обработчикам событий класса.

По тику основного таймера происходят следующие обновления:

1. Обновление состояния и изображения:
   1. Вызов события, оповещающего об очередном обновлении.
   2. Исследование движущихся в данный момент точек на предмет окончания движения.
   3. Исследование вершин на готовность выпустить точки.
   4. Запуск новых точек.
   5. Проверка состояния системы на предмет полного окончания движения и запуск события, оповещающего об этом, если движение окончено (здесь же, если было запрошено построение графика распределения лавин [см. Приложение 2], размер последней лавины добавляется на график). Если моделируется движение песка, в вершину добавляется песчинка (вручную или случайно в зависимости от настроек).
2. Обновление графика зависимости количества точек от времени (в случае его построения), если после действий п.1 общее количество точек на графе было изменено.
3. Добавление текущего изображения в список кадров объекта GifBitmapEncoder для возможности последующего сохранения визуализации процесса (в случае, если данная опция была выбрана).

Процесс можно приостановить нажатием кнопки Stop верхнего меню и возобновить нажатием кнопки Continue, а также полностью сбросить, кликнув по кнопке Reset.

* + - 1. **Алгоритм проверки ориентированного метрического графа на сильную связность**

Проверка графа на сильную связность осуществляется с использованием упрощенного алгоритма Косарайю для поиска областей сильной связности в ориентированном графе. В процессе работы алгоритма выполняются следующие шаги:

1. С первой вершины графа запускается поиск в глубину.
2. По мере выполнения поиска в глубину заполняется массив типа bool[]. После посещения вершины элементу с ее индексом в массиве присваивается значение true.
3. По окончанию первого обхода, если массив содержит хотя бы одно значение false, метод проверки возвращает false.
4. Если при первом обходе все вершины были посещены, граф инвертируется.
5. На инвертированном графе с первой вершины запускается поиск в глубину.
6. Повторяются действия п.2.
7. По окончанию обхода, если хотя бы одна вершина не была посещена, метод проверки возвращает false.
8. Если все вершины при втором обходе были посещены, метод проверки возвращает true.

Данный алгоритм определяет, является ли ориентированный граф 𝐺 = (𝑉, 𝐸) сильно связным, за определенное время 𝑂(|𝑉| + |𝐸|).

Его сложность связана со сложностью алгоритма поиска в глубину, который, в свою очередь, должен быть использован дважды, а также со сложностью нахождения обратного графа. Оба эти алгоритма имеют линейную сложность и именно поэтому Алгоритм Косарайю работает за линейное время.

* + - 1. **Алгоритм вычисления введенных значений длин ребер.**

Вычисление введенных математических выражений производится путем перевода введенной пользователем строки в обратную польскую запись (польскую нотацию) [см. Список литературы 2], особенность которой состоит в том, что такая запись может быть интерпретирована без неоднозначности. В ходе вычисления выражения выполняются следующие действия:

1. На вход подается строка, содержащая введенное выражение.
2. Производится анализ расстановки скобок и перевод букв в нижний регистр.
3. Пока указатель на позицию в строке не выйдет за ее пределы производится:
   1. Лексический анализ подвыражения, начинающегося в текущей позиции: определяется, является ли подвыражение оператором [см. Приложение 4, Поддерживаемые операторы], константой [см. Приложение 4, Поддерживаемые константы], функцией [см. Приложение 4, Поддерживаемые функции] или цифрой. В случае, если подвыражение не было определенно, выбрасывается исключение.
   2. Синтаксический анализ результата лексического анализа. В зависимости от определенного в п. 3.1. типа подвыражения [см. Приложение 4, Маркеры] над этим подвыражением производятся определенные действия:
      1. Если это число, оно добавляется в результирующую строку.
      2. Если это функция или открывающая скобка – помещается в стек.
      3. Если это закрывающая скобка, из стека вынимаются все элементы до открывающей скобки и добавляются в результирующую строку.
      4. Если это оператор, из стека вынимаются операторы, пока их приоритетность меньше либо равна приоритетности данного, вынимаемые элементы добавляются в строку-результат, а сам оператор помещается в стек.
4. Вынимаются все операторы из стека и добавляются в результирующее выражение.
5. Производится подсчет выражения в обратной польской нотации:
   1. Пока указатель на позицию в строке, полученной в п.3.-4., не выйдет за ее пределы производится:
      1. Лексический анализ подвыражения: из строки выбирается следующий элемент выражения.
      2. Синтаксический анализ:
         1. Если лексический анализ вернул операнд, он добавляется в стек.
         2. Если операцию, в зависимости от арности она применяется к одному или двум последним операндам в стеке.
         3. В стек добавляется результат вычислений.
6. В результате в стеке остается единственный элемент – результат вычисления выражения, введенного пользователем.

Удобство обратной польской нотации заключается в том, что выражения, представленные в такой форме, вычисляются легче и быстрее по сравнению с алгебраической нотацией. Приведенный алгоритм считает введенное выражение за линейное время 𝑂(n), где n – длина строки.

* + - 1. **Алгоритмы построения и редактирования графа.**

Выполнение действий при построении и редактировании графа обеспечивается методами классов, реализующих интерфейс ICommand [см. Приложение 7, Таблицы 2.6-2.15], позволяющими добавлять и удалять вершины и дуги, изменять длину дуг, перемещать вершины и граф целиком, изменять размер графа и менять цвета отрисовки элементов графа. Выполнение, возврат и отмена действий осуществляется методами класса CommandsManager [см. Приложение 7, Таблицы 2.16].

По клику пользователя на поле для рисования осуществляется:

1. Проверка возможных факторов, препятствующих редактированию графа (пользователь не сможет изменить граф, если программа в данный момент находится в режиме моделирования движения).
2. Определение текущего инструмента
   1. Если это курсор:
      1. ищется потенциальная вершина для выделения или перетаскивания;
      2. по движению мыши зажатая вершина передвигается, если она была зажата короткое время, вершина просто выделятся.
   2. Если это инструмент построения вершин, по клику на поверхность вызывается команда добавления вершины, в месте клика появляется новая вершина.
   3. Если это инструмент построения дуг, последовательно первым кликом ищется индекс вершины начала, вторым – индекс вершины конца. Если это не петля и такой дуги еще нет в графе, вызывается команда добавления дуги.
   4. Если это ластик, по двойному клику на поверхность ищется объект для удаления и, если он найден, вызывается команда удаления данного объекта.

По клику на кнопки уменьшения и увеличения, перемещения графа вызываются соответствующие команды. Изменение размера графа также возможно по нажатию сочетания клавиш Ctrl++ или Ctrl+–, а таже вращением колесика мыши с зажатой клавишей Ctrl. В последних случаях команда будет вызвана только после окончания движения. Перемещение графа на поле аналогично осуществляется как нажатием сочетания клавиш (Ctrl+Up, Ctrl+Right, Ctrl+Down, Ctrl+Left), так и вращением колесика мыши. Аналогично, команда вызывается только после окончания движения.

Длины дуг меняются в специальном TextBox’е. По нажатию на клетку матрицы смежности имя соответствующей дуги и ее длина выводятся в ComboBox (откуда также можно выбрать дугу для редактирования) и данный TextBox соответственно. По нажатию кнопки Ok веденное значение высчитывается согласно алгоритму, описанному в п. 3.2.1.3, и вызывается команда изменения длины.

Изменение цветов отрисовки происходит путем выбора нового цвета в открытом ColorDialog, после подтверждения действия в котором, вызывается соответствующая команда.

Все приведенные команды можно отменить и вернуть, кликнув на соответствующие кнопки окна, или сочетанием клавиш Ctrl+X для отмены и Ctrl+Y для возврата.

* + - 1. **Алгоритм генерации случайного графа.**

Возможность генерации случайного графа предоставляется методами класса RandomDigraphGeneratorForm [см. Таблица 2.27]. Алгоритм построения графа:

1. Пользователь выбирает желаемое число вершин n (может быть выбрано случайно)
2. Случайно проставляются n вершин, пороги, рефракторные периоды и состояния [см. Приложение 3] выбираются случайным образом.
3. Первая вершина помечается как посещенная и выбирается начальной вершиной.
4. Выбирается случайная вершина, помечается как посещенная и строится ребро случайной длины из начальной вершины в выбранную. Выбранная вершина помечается как начальная.
5. П.4 повторяется, пока все вершины не будут посещены
6. Добавляется ребро случайной длины из последней выбранной вершины в первую.

Получаемый в результате алгоритма граф является сильно связным, так как строится как единый цикл.

* + 1. **Обоснование выбора алгоритма решения задачи.**

Алгоритмы, описанные в п.3.2.1.1., п.3.2.1.4. и п. 3.2.1.5., обусловлены потребностями программы и тем набором необходимых функций, который был определен в Техническом задании на данную разработку.

Выбор алгоритма определения сильной связности графа (п. 3.2.1.2.) обусловлен своей простотой реализации и лучшим временем выполнения по сравнению с другими алгоритмами, решающими данную задачу: алгоритм Флойда-Уоршелла – 𝑂(n3), алгоритм Сильно связанных компонентов (SCC) – 𝑂(|𝑉| + |𝐸|) (сложнее для понимания, выполняет дополнительные действия), выполнение алгоритма DFS V раз – 𝑂 (𝑉 \* (𝑉 + 𝐸)).

Выбор алгоритма вычисления строковых математических выражений (п. 3.2.1.3.) обусловлен скоростью, простотой и точностью производимых с использованием обратной польской нотации вычислений. Альтернативным решением данной задачи могло бы быть отправление введенного выражения в специальную программу, производящую математические вычисления (например, Wolfram), выбранное же решение избавляет от необходимости взаимодействия с другими программами.

* + 1. **Возможные взаимодействия программы с другими программами.**

В целом программа работает самостоятельно. Однако для открытия или редактирования сохраненных данных графа необходим текстовый редактор, например Microsoft Word или Блокнот. Для открытия сохраняемых изображений (орграфа, построенного графика или GIF анимации движения точек) требуется средство открытия изображений соответствующего формата, например Microsoft Photos. Для открытия сохраненных в формате .csv данных построенных графиков может также потребоваться средство просмотра таблиц, например Microsoft Excel.

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

В качестве входных данных программе могут передаваться данные из файла .dgmm [см. Приложение 5]. Также входные данные могут быть внесены пользователем вручную, то есть программа предоставляет возможность построения нового ориентированного метрического графа в редакторе, а также задания всех необходимых дополнительных параметров, таких как длины дуг, пороги, периоды восстановления и начальные состояния вершин [см. Приложение 3].

Программа выводит все полученные в ходе работы результаты в отведенные окна и по желанию пользователя сохраняет их в файл.

Предоставляется возможность сохранения построенного графа в файл формата .dgmm, изображением .jpg или папкой, содержащей как файл с данными графа, так и его изображение. Полученное в процессе моделирования движения точек GIF-изображение таже сохраняется в файл соответствующего формата. Данные графиков сохраняются в файлы формата .csv, а также .jpg изображением или папкой с обоими файлами.

При работе с файлами формата .dgmm используется встроенный механизм XML сериализации и десериализации из XML формата.

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

Механизм сериализации был выбран из-за того, что:

1. позволяет сохранить объект в файле, не теряя данных при сохранении;
2. позволяет воссоздать объект в его первоначальную форму;
3. упрощает процесс сохранения объекта и его открытия.

XML формат был выбран, так как:

1. сериализует данные в читабельный формат;
2. позволят пользователю корректировать данные вне среды.

Формат .csv для сохранения данных графиков был выбран, так как позволяет после сохранения просматривать данные в удобном формате таблиц.

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

Для корректной работы программы необходим следующий состав программных средств:

1. операционная система Windows 7 SP1 (x86 and x64) или более поздняя версия;
2. установленный Microsoft .NET Framework 4.7.

Для корректной работы программы необходим следующий состав технических средств:

1. персональный компьютер, оснащенный 32-разрядным (x86) или 64-разрядным (x64) процессором с тактовой частотой 1 ГГц и выше (рекомендуется процессор Intel Core i7 с частотой 1.8ГГц и выше или аналогичный процессор);
2. 512 МБ оперативной памяти (ОЗУ) или больше;
3. не менее 4.5 ГБ свободного места на жестком диске;
4. видеокарта и монитор с разрешением не менее чем 1258x753 точек;
5. клавиатура и мышь.
   * 1. **Обоснование выбора технических и программных средств.**

При реализации программы для улучшения внешнего вида формы и элементов управления в файл конфигурации был добавлен раздел <System.Windows.Forms.ConfigurationSection>. Данный раздел для поддержки высокого разрешения в Windows Forms впервые появился в .NET Framework 4.7 и поддерживается в данной и более поздних версиях Microsoft .NET Framework. Остальные используемые в программе элементы были представлены и в более ранних версиях Microsoft .NET Framework.

Microsoft .NET Framework 4.7 в свою очередь требует:

1. операционную систему Windows 7 SP1 (x86 and x64), Windows 8.1 (x86 and x64) или Windows 10 Anniversary Update (x86 and x64);
2. процессор с тактовой частотой 1 ГГц и выше;
3. 512 МБ оперативной памяти или больше;
4. 4.5 ГБ свободного места на жестком диске и выше.

Главное окно программы имеет разрешение 1258x753 точек, поэтому разрешение монитора должно быть не менее чем 1258x753 точек.

Мышь и клавиатура необходимы для взаимодействия с приложением.

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

Программа может быть использована преподавателями, студентами и исследователями в области математики, информатики и нейробиологии для исследования движения точек на ориентированных метрических графах, а также различных процессов путем их моделирования посредством таких графов. Помимо этого, программа может использоваться при изучении песочных моделей. Эта область содержит много открытых проблем и интенсивно развивается, что говорит об актуальности разработанного приложения для исследований в этом направлении.

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

Преимуществами программы являются:

* дешевизна, отсутствие встроенных покупок;
* простой интуитивно понятный англоязычный интерфейс;
* свободное распространение.

Несмотря на существование довольно обширного множества различных визуализаторов процессов на графах, быстрый поиск в сети Интернет на момент создания приложения не выявил аналогов данной программы.

Отличие программы от существующих визуализаторов песочных моделей заключается в визуализации процессов именно на ориентированных метрических сильно связных графах, в то время как данные модели, ввиду того что являются клеточным автоматом, чаще реализуются на регулярных сетках ячеек.

1. **СПИСОК ЛИТЕРАТУРЫ**
2. Abelian sandpile model: [Электронный ресурс]: Режим доступа: URL: <https://en.wikipedia.org/wiki/Abelian_sandpile_model>, свободный. (дата обращения: 17.05.20).
3. Reverse Polish notation: [Электронный ресурс]: Режим доступа: URL: <https://en.wikipedia.org/wiki/Reverse_Polish_notation>, свободный. (дата обращения: 17.05.20).
4. Winfried Just. Chapter 6. Neuronal Networks: A Discrete Model // Mathematical Concepts and Methods in Modern Biology: Using Modern Discrete Models / Winfried Just, Sungwoo Ahn и David Terman (авторы), Robeva, R., Hodge, T. (редакторы). – USA: Academic Press, 2013. – 179-211 с.
5. Yerzhan Kalzhani. Проект MathParserTK: Math Parser .NET C# [Электронный ресурс] / GitHub. Режим доступа: URL: https://github.com/kirnbas/MathParserTK, свободный. (дата обращения: 15.05.20).
6. Большакова Е. А. Свойство бисвязности ориентированного графа: Дипломная работа, Санкт-Петербургский государственный университет, Математико-механический факультет, Кафедра исследования операций / Большакова Е. А., научный руководитель: Доктор физ.-мат. наук, профессор Романовский Иосиф Владимирович – Санкт-Петербург, СПбГУ, 2016. – 12-16 с.
7. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с
8. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с.
9. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с.
10. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с.
11. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с.
12. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с.
13. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с.
14. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с.
15. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001. – 126 с.
16. Евстигнеев В. А. Толковый словарь по теории графов в информатике и программировании / Евстигнеев В.А., Касьянов В.Н. – Россия, Наука, Сибирское предприятие РАН, 1999.
17. Калинин, Н. С. Модель пересыпания песка и дивизоры на графах [Электронный ресурс]: курс лекций — Электрон. дан. — Дубна: Летняя школа «Современная математика», 2017. — Режим доступа: URL: <https://www.mccme.ru/dubna/2017/courses/kalinin.html>, свободный. (дата обращения: 14.05.20).

**ПРИЛОЖЕНИЕ 1**

**ТЕРМИНОЛОГИЯ**

Ниже приведен список необходимых терминов для ознакомления.

**Вершина** – базовое понятие. Точка, где могут сходиться/выходить рёбра и/или дуги.

**Ребро** – базовое понятие. Ребро соединяет две вершины графа.

**Дуга** – ребро, имеющее направление. Упорядоченная пара вершин (v, w), где вершину v называют началом, а w - концом дуги. Можно сказать, что дуга v → w ведет от вершины v к вершине w, при этом вершина w смежная с вершиной v.

**Граф** – базовое понятие. Включает множество вершин и множество рёбер, являющееся подмножеством декартова квадрата множества вершин (то есть каждое ребро соединяет ровно две вершины).

**Орграф** – ориентированный граф G = (V, E) есть пара множеств, где V - множество вершин (узлов), E – множество дуг.

**Сильно связный граф** – ориентированный граф, в котором все вершины сильно связаны, т. е. существует путь из первой во вторую и из второй в первую и т. д.

**Метрический граф** – граф, каждое ребро которого имеет заданное вещественное время прохождения.

**Инцидентность** — отношение между ребром (дугой) и его концевыми вершинами, т.е. ребро e = (a,b) инцидентно вершинам a и b и вершины a, b инцидентны ребру e = (a,b).

**ПРИЛОЖЕНИЕ 2**

**ТЕРМИНОЛОГИЯ ПЕСОЧНЫХ МОДЕЛЕЙ**

**Модель песчаной кучи** (англ. sandpile model) — классическая модель теории самоорганизованной критичности, связанная со многими областями математики. Ниже приведен список необходимых терминов для ознакомления [подробнее см. Список литературы 1, 16].

Состояние модели на графе задаётся количествами песчинок в вершинах графа и эволюционирует по следующему правилу: если количество песчинок в вершине не меньше её степени, то вершина отдаёт по одной песчинке каждому из своих соседей. Такая операция называется **обвалом** (toppling).

Процесс выполнения обвалов пока это возможно называется **релаксацией**, а ее размер называется **размером лавины** и равен количеству вершин, в которых произошли при релаксации обвалы.

**Сток** – множество вершин, в которых запрещены обвалы, песок, попадающий в одну из стоковых вершин, попросту исчезает.

Состояние системы называется **стабильным** если ни в одной точке обвал произойти не может.

**ПРИЛОЖЕНИЕ 3**

**ОПИСАНИЕ ИСПОЛЬЗУЕМОЙ МОДЕЛИ**

Используемая в приложении модель, согласно которой реализуется ориентированный метрический граф является модифицированной моделью нейронной сети, описанной американскими математиками Winfried Just, Sungwoo Ahn и David Terman в статье Neuronal Networks: A Discrete Model книги Mathematical Concepts and Methods in Modern Biology: Using Modern Discrete Models [см. Список литературы 3].

Граф в программе задаётся пятью множествами: множеством вершин V, множеством ребер E, множеством th чисел, определяющих пороги каждой из вершин, множеством p чисел, определяющих продолжительность рефракторных периодов каждой из вершин и множеством s чисел, определяющих условное «состояние» каждой вершины. Множества порогов, продолжительностей рефракторных периодов, состояний и множество вершин, очевидно, равномощны. Таким образом, граф здесь: G = (V, E, th, p, s).

**Порогом** вершины здесь называется число точек в ней, необходимое для того, чтобы вершина могла выпустить точки.

**Рефракторный период** вершины – это время (в мс), в течение которого после выпуска точек вершина не может выпустить их снова, даже если количество точек в ней превышает установленный порог.

**Состоянием** вершины здесь называется количество в ней точек.

Таким образом, необходимое и достаточное условие для того, чтобы вершина выпустила точки, – это значение ее состояния, превышающее либо имеющее равное значение с ее порогом, и окончившийся рефракторный период.

Отличием данной модели от приведенной в Networks: A Discrete Model, делающим невозможным использование ее в чистом виде, являются метрические характеристики графа, отсутствующие в приведенной авторами дискретной модели. Winfried Just, Sungwoo Ahn и David Terman используют вектор s, чтобы отслеживать окончание рефракторного периода, определяя факт превышения в вершине порога по значению s для смежных вершин, корректность здесь обуславливается дискретностью времени и отсутствием длин у ребер.

Время при моделировании движения точек в приложении реально, а дуги имеют длину, что вызывает разную скорость прохождения различных дуг и не позволяет судить о готовности вершины выпустить точки по смежным с ней вершинам. Ввиду вышеперечисленного, при реализации программы значение вектора s изменило свою роль на «счетчик» точек, а соблюдение рефракторного периода стало обеспечиваться специальными таймерами.

**ПРИЛОЖЕНИЕ 4**

**ОБРАТНАЯ ПОЛЬСКАЯ НОТАЦИЯ.**

**ПОДДЕРЖИВАЕМЫЕ МАТЕМАТИЧЕСККИЕ ОПЕРАЦИИ, КОНСТАНТЫ И ФУНКЦИИ.**

**Обратная польская запись** (англ. Reverse Polish notation, RPN) — форма записи математических и логических выражений, в которой операнды расположены перед знаками операций [подробнее см. Список литературы 2].

**Маркеры:** При составлении программой выражения в польской записи используется система маркеров, определяющих тип элемента выражения, что позволяет легко управлять членами выражения. В качестве маркеров используются: # – для обозначения чисел; $ – для обозначения операторов; @ – для обозначения функций.

**Поддерживаемые операции:**

1. Сложение (+);
2. Вычитание (–);
3. Умножение (\*);
4. Деление (/);
5. Возведение в степень (^).

**Поддерживаемые функции:**

1. извлечение корня (sqrt);
2. синус (sin);
3. косинус (cos);
4. тангенс (tg);
5. котангенс (ctg);
6. гиперболический синус (sh);
7. гиперболический косинус (ch);
8. гиперболический тангенс (th);
9. натуральный логарифм (log);
10. экспонента (exp);
11. модуль (abs).

**Поддерживаемые константы:**

1. число (pi);
2. число *e* (e).

Помимо перечисленного поддерживаются скобки и унарные плюс и минус.

**ПРИЛОЖЕНИЕ 5**

**ОПИСАНИЕ ФОРМАТА .DGMM**

Файл формата .dgmm содержит в сереализованном с использованием XML сериализации виде информацию об орграфе, созданном в программе.

Программа позволяет открыть файл формата .dgmm, и при этом осуществляет корректное открытие только файлов данного формата, созданных в этой программе.

# **ПРИЛОЖЕНИЕ 6**

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ, СТРУКТУР, ИНТЕРФЕЙСОВ И ПЕРЕЧИСЛЕНИЙ**

Таблица 1

|  |  |
| --- | --- |
| Класс/Структура/  Интерфейс/Перечисление | Назначение |
| MainWindow | Класс наследник Form, представляющий главное окно приложения. Содержит обработчики пользовательских событий, в нем так же содержатся и взаимодействуют объекты других классов, обеспечивающие работу программы. |
| Arc | Структура, представляющая дуги графа. Содержит индексы начальной и конечной вершины в орграфе, а также длину дуги. |
| Vertex | Структура, представляющая вершину орграфа. Содержит координаты вершины на плоскости. |
| Digraph | Класс, представляющий ориентированный метрический граф. Содержит списки вершин и дуг орграфа, списки значений порогов вершин, периодов восстановления и начальных состояний, а также методы для корректного удаления и добавления вершин и дуг в граф. |
| DigraphChangedEventArgs | Класс наследник EventArgs, используется для реализации стандартного шаблона событий класса Digraph. События типа EventHandler <DigraphChangedEventArgs> оповещают об изменениях орграфа, таких как добавление и удаление вершин и ребер. Класс содержит индекс добавленного/удаленного элемента. |
| ICommand | Интерфейс, представляющий команды, совершаемые пользователем. Содержит методы выполнения некоторого действия и выполнения действия к нему обратного. |
| AddArcCommand | Класс, реализующий интерфейс ICommand. Содержит методы для добавления дуги в граф и корректной отмены добавления. |
| AddVertexCommand | Класс, реализующий интерфейс ICommand. Содержит методы для добавления вершины в граф и корректной отмены добавления. |
| ChangeArcLengthCommand | Класс, реализующий интерфейс ICommand. Содержит методы для изменения длины дуги графа и ее возвращения к предыдущему значению. |
| ChangeColorCommand | Класс, реализующий интерфейс ICommand. Содержит методы для изменения цвета отрисовки дуг и вершин графа и возвращения этого цвета к предыдущему значению. |
| EraseArcCommand | Класс, реализующий интерфейс ICommand. Содержит методы для удаления дуги из графа и ее корректного возвращения. |
| EraseVertexCommand | Класс, реализующий интерфейс ICommand. Содержит методы для удаления вершины из графа и ее корректного возвращения. |
| MoveDigraphCommand | Класс, реализующий интерфейс ICommand. Содержит методы для перемещения графа на плоскости и его корректного возвращения в предыдущую позицию. |
| MoveVertexCommand | Класс, реализующий интерфейс ICommand. Содержит методы для перемещения вершины графа и ее корректного возвращения в предыдущую позицию. |
| ResizeDigraphCommand | Класс, реализующий интерфейс ICommand. Содержит методы для изменения размера графа и его корректного возвращения к предыдущему размеру. |
| CommandsManager | Класс, содержащий стек выполненных и отмененных команд. Реализует возможность отмены последних совершенных пользователем изменений при построении графа и возвращения отмененных изменений. |
| ConnectivityCheck | Класс, предоставляющий методы для проверки ориентированного графа на сильную связность. |
| GraphDrawing | Класс, предоставляющий методы для отрисовки графа и его элементов во всех необходимых для работы приложения формах. |
| MathParser | Класс, предоставляющий методы для вычисления строкового представления математического выражения. |
| ChartWindow | Класс наследник Form, представляющий окно, в котором во время моделирования движения отображается запрошенный пользователем график. |
| MovementModeling | Класс, предоставляющий возможность моделирования движения точек на ориентированном метрическом графе с условием синхронизации в вершинах. |
| MovementTickEventArgs | Класс наследник EventArgs, используется для реализации стандартного шаблона событий класса MovementModeling. Содержит число, указывающее прошедшее с начала движения количество миллисекунд. |
| MovementModelingActions | Перечисление, содержащее набор дополнительных действий, запрашиваемых у программы при моделировании движения. |
| MovementModelingType | Перечисление, содержащее набор типов моделирования движения. |
| SandpileChartType | Перечисление, содержащее набор типов графиков, построение которых возможно при моделировании движения песка. |
| DigraphComponentsRemover | Класс, предоставляющий статические методы для удаления вершин и ребер. |
| DigraphInformationDemonstration | Класс, предоставляющий статические методы для отображения различной информации о графе в таблицах DataGridView. |
| RandomDigraphGeneratorForm | Класс наследник Form, представляющий окно для выбора параметров построения случайного графа, реализующий построение этого графа. |
| SquareLatticeForm | Класс наследник Form, представляющий окно для выбора параметров построения квадратной решетки, реализующий построение такого графа. |
| TriangularLatticeForm | Класс наследник Form, представляющий окно для выбора параметров построения треугольной решетки, реализующий построение такого графа. |

**ПРИЛОЖЕНИЕ 7**

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

Таблица 2.1

Описание полей методов и свойств класса MainWindow.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| AboutApp | private | string | Строка с краткой информацией о программе и разработчике. | |
| commandsManager | private | CommandsManager | Предоставляет возможность отмены и возврата действий пользователя при построении графа. | |
| digraph | private | Digraph | Ориентированный метрический граф, с которым ведется работа. | |
| resizeCoefficient | private | double | Коэффициент масштабирования при изменении размера графа. | |
| graphDrawing | private | GraphDrawing | Предоставляет методы для отрисовки графа и его компонент. | |
| isControlPressed | private | bool | Показывает, нажата ли в данный момент клавиша Control. | |
| isOnMovement | private | bool | Показывает, находится ли программа в данный момент в режиме моделирования движения. | |
| isPressed | private | bool | Показывает, зажата ли в данный момент вершина на поле. | |
| movement | private | MovementModeling | Моделирует движение точек на графе. | |
| movedVertex | private | Vertex | Передвигаемая вершина. | |
| movedVertexIndex | private | int | Индекс передвигаемой вершины. | |
| Rnd | private | Random | Генерирует случайные значения. | |
| ticks | private | DateTime | Показывает, как долго была зажата вершина. | |
| vEnd | private | int | Индекс конечной вершины, выбранной для добавления ребра. | |
| vStart | private | int | Индекс начальной вершины, выбранной для добавления ребра. | |
| xCoefficient | private | int | Коэффициент смещения графа по оси Х. | |
| yCoefficient | private | int | Коэффициент смещения графа по оси Y. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| AboutToolStripMenuItem\_Click | private | void | Object, EventArgs | Предоставляет краткую справку о программе и разработчике. |
| AddVertexToGridAdjacencyMatrix | private | void | int | Добавляет вершину в матрицу смежности. |
| AddVertexToGridParameters | private | void | int | Добавляет вершину в матрицу параметров. |
| ArcLength\_TextChanged | private | void | Object, EventArgs | Делает поле для ввода длины недоступным, если выбранного ребра не существует. |
| ArcName\_TextChanged | private | void | Object, EventArgs | Определяет существует ли выбранная дуга и выводит ее длину или сообщение об ошибке. |
| ArcsColorDialogOpen\_Click | private | void | Object, EventArgs | Открывает ColorDialog для выбора цвета отрисовки дуг. |
| ArcsColorPanel\_Click | private | void | Object, EventArgs | Переводит фокус на панель. |
| ArcsColorPanel\_Enter | private | void | Object, EventArgs | Делает кнопку открытия ColorDialog видимой. |
| ArcsColorPanel\_Leave | private | void | Object, EventArgs | Делает кнопку открытия ColorDialog невидимой. |
| BasicTypeCheckBox\_CheckedChanged | private | void | Object, EventArgs | Поддерживает один из типов моделирования выбранным. |
| Build\_Click | private | void | Object, EventArgs | Открывает пустой редактор для создания нового графа. |
| ChangeDrawingElementsState | private | void | bool | Меняет видимость элементов управления, относящихся к редактору графов. |
| ChangeMainMenuState | private | void | bool | Меняет видимость элементов управления, относящихся к главному меню. |
| ChangeWindowStateForMovementModeling | private | void | bool | Подготавливает окно к моделированию движения. |
| ChartCheckBox\_CheckedChanged | private | void | Object, EventArgs | Отображает или скрывает варианты графиков для модели песка. |
| CheckConnectivity | private | bool | – | Проверяет граф на валидность. |
| ClearButton\_Click | private | void | Object, EventArgs | Удаляет созданный граф и возвращает редактор к исходному состоянию. |
| CursorButton\_Click | private | void | Object, EventArgs | Выбирает курсор как текущий инструмент. |
| DataToolStripMenuItem\_Click | private | void | Object, EventArgs | Открывает диалог для сохранения данных графа. |
| DeleteButton\_Click | private | void | Object, EventArgs | Выбирает ластик как текущий инструмент. |
| DigraphOpenFileDialog | private | OpenFileDialog | – | Создает OpenFileDialog для открытия графа из файла. |
| Down\_Click | private | void | Object, EventArgs | Двигает граф вниз. |
| DrawingSurface\_MouseClick | private | void | Object, MouseEventArgs | В зависимости от выбранного инструмента и состояния программы производит определенные действия в построении и редактировании графа. |
| DrawingSurface\_MouseDoubleClick | private | void | Object, MouseEventArgs | Удаляет нажатую вершину или ребро. |
| DrawingSurface\_MouseDown | private | void | Object, MouseEventArgs | Если текущий инструмент – курсор, готовится двигать вершину. |
| DrawingSurface\_MouseMove | private | void | Object, MouseEventArgs | Если выбрана вершина для передвижения, двигает ее. |
| DrawingSurface\_MouseUp | private | void | Object, MouseEventArgs | В зависимости произошедшего ранее производит определенные действия в построении и редактировании графа. |
| EdgeButton\_Click | private | void | Object, EventArgs | Выбирает ластик как текущий инструмент. |
| EnlargeButton\_Click | private | void | Object, EventArgs | Увеличивает изображение графа. |
| ExitToolStripMenuItem\_Click | private | void | Object, EventArgs | Закрывает приложение. |
| FindArcVertices | private | bool | Object, EventArgs | Ищет выбранные для построения дуги вершины. |
| FolderBrowserDialogForGraphSaving | private | FolderBrowserDialog | – | Создает FolderBrowserDialog для полного сохранения графа. |
| GetChartTypes | private | SandpileChartType[] | – | Возвращает массив выбранных типов графиков для моделирования движения песка. |
| GetModelingActions | private | MovementModelingActions[] | – | Возвращает массив выбранных действий для моделирования. |
| GraphBuilder\_KeyDown | private | void | Object, KeyEventArgs | Осуществляет действия, связанные с конкретными клавишами. |
| GraphBuilder\_SizeChanged | private | void | Object, EventArgs | Подстраивает элементы окна под новый размер. |
| GridAdjacencyMatrix\_CellClick | private | void | Object, DataGridViewCellEventArgs | Выводит информацию о выбранной дуге в контролы, отвечающие за их редактирование. |
| GridParameters\_CellValueChanged | private | void | Object, DataGridViewCellEventArgs | Проверяет корректность нового значения и меняет соответствующий параметр. |
| Left\_Click | private | void | Object, EventArgs | Двигает граф влево. |
| MainMenuToolStripMenuItem\_Click | private | void | Object, EventArgs | Осуществляет переход к главному меню. |
| MainWindow | public | – | – | Конструктор класса. |
| MainWindow\_KeyUp | private | void | Object, KeyEventArgs | Вызывает команды движения графа, если клавиша изменила его положение или размер. |
| MainWindow\_MouseWheel | private | void | Object, MouseEventArgs | Меняет положение или размер графа по вращению колесика мыши. |
| Movement\_PreviewKeyDown | private | void | Object, PreviewKeyDownEventArgs | Переводит фокус, чтобы предотвратить некорректные действия по нажатию клавиши. |
| MovementEndedSandpileEventHandler | private | void | Object, EventArgs | Предлагает пользователю добавить песок в вершину после окончания движения. |
| MovementToolStripMenuItem\_Click | private | void | Object, EventArgs | Запускает моделирование движения. |
| NewProjectToolStripMenuItem\_Click | private | void | Object, EventArgs | Открывает пустой редактор для создания нового графа. |
| OkWeight\_Click | private | void | Object, EventArgs | Меняет длину дуги. |
| Open\_Click | private | void | Object, EventArgs | Открывает диалог для открытия графа из файла. |
| OpenProjectToolStripMenuItem\_Click | private | void | Object, EventArgs | Открывает диалог для открытия графа из файла. |
| RadiusTrackBar\_ValueChanged | private | void | Object, EventArgs | Меняет радиус вершин. |
| RandomAddingCheckBox\_CheckedChanged | private | void | Object, EventArgs | Меняет обработчик события окончания движения песка. |
| RandomAddingLabel\_Click | private | void | Object, EventArgs | Добавляет песчинку в случайную вершину. |
| RandomGraph\_Click | private | void | Object, EventArgs | Открывает окно, позволяющее построить случайный сильно связный граф. |
| RedoButton\_Click | private | void | Object, EventArgs | Возвращает последнее отмененное действие. |
| ReduceButton\_Click | private | void | Object, EventArgs | Уменьшает изображение графа. |
| RefreshVariables | private | void | – | Сбрасывает все произведенные изменения. |
| RemoveVertexFromGridAdjacencyMatrix | private | void | int | Убирает вершину из матрицы смежности. |
| RemoveVertexFromGridParameters | private | void | int | Убирает вершину из таблицы параметров. |
| ResetToolStripMenuItem\_Click | private | void | Object, EventArgs | Сбрасывает процесс моделирования и возвращает окно в состояние редактора. |
| Right\_Click | private | void | Object, EventArgs | Двигает граф вправо. |
| SandpileChartType1\_CheckedChanged | private | void | Object, EventArgs | Поддерживает хотя бы один из типов выбранным. |
| SandpileChartType2\_CheckedChanged | private | void | Object, EventArgs | Поддерживает хотя бы один из типов выбранным. |
| SandpilePalette\_SelectionChanged | private | void | Object, EventArgs | Снимает выбор ячейки. |
| SandpileTypeCheckBox\_CheckedChanged | private | void | Object, EventArgs | Поддерживает один из типов моделирования выбранным. |
| SaveAllToolStripMenuItem\_Click | private | void | Object, EventArgs | Открывает диалог для выбора папки для полного сохранения графа. |
| SaveFileDialogForDataSaving | private | SaveFileDialog | Object, EventArgs | Создает SaveFileDialog для сохранения данных графа. |
| SaveFileDialogForGifSaving | private | SaveFileDialog | Object, EventArgs | Создает SaveFileDialog для сохранения GIF-изображения. |
| SaveFileDialogForImageSaving | private | SaveFileDialog | Object, EventArgs | Создает SaveFileDialog для сохранения изображения графа. |
| SaveGif | private | void | Object, EventArgs | Открывает диалог для сохранения Gif-изображения. |
| SaveGraph | private | DialogResult | string, string | Открывает диалог для сохранения графа полностью. |
| SaveImageToolStripMenuItem\_Click | private | void | Object, EventArgs | Открывает диалог для сохранения изображения графа. |
| SelectStock | private | void | int, int | Ищет выбранную вершину и помечает ее как стоковую. |
| SelectVertexToAddSand | private | void | int, int | Ищет выбранную вершину и добавляет в нее песчинку. |
| SquareLattice\_Click | private | void | Object, EventArgs | Открывает окно для построения квадратной решетки. |
| StockLabel\_Click | private | void | Object, EventArgs | Запускает моделирование движения песка. |
| StopToolStripMenuItem\_Click | private | void | Object, EventArgs | Приостанавливает движение. |
| SubscribeToDigraphEvents | private | void | – | Подписывает обработчики на события класса Digraph. |
| TriangleLattice\_Click | private | void | Object, EventArgs | Открывает окно для построения треугольной решетки. |
| UndoButton\_Click | private | void | Object, EventArgs | Отменяет последнее действие в редакторе. |
| Up\_Click | private | void | Object, EventArgs | Двигает граф вверх. |
| UpdateDigraphInfo | private | void | – | Полностью обновляет все отображаемые данные графа. |
| UpdateElapsedTime | private | void | Object, MovementTickEventArgs | Обновляет прошедшее с начала движения время. |
| UpdateImage | private | void | – | Обновляет изображение графа. |
| VertexButton\_Click | private | void | Object, EventArgs | Выбирает инструмент добавления вершин как текущий. |
| VertexColorDialogOpen\_Click | private | void | Object, EventArgs | Открывает ColorDialog для выбора цвета отрисовки вершин. |
| VerticesColorPanel\_Click | private | void | Object, EventArgs | Переносит фокус на панель. |
| VerticesColorPanel\_Enter | private | void | Object, EventArgs | Делает кнопку открытия ColorDialog видимой. |
| VerticesColorPanel\_Leave | private | void | Object, EventArgs | Делает кнопку открытия ColorDialog невидимой. |
| WheelStopped | private | void | Object, EventArgs | Выполняет команду движения графа после остановки движения колесиком. |

Таблица 2.2

Описание полей методов и свойств структуры Arc.cs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поля | | | | | |
| Имя | Модификатор доступа | Тип | Назначение | | |
| startVertex | private | int | Вершина начала дуги. | | |
| endVertex | private | int | Вершина конца дуги. | | |
| length | private | double | Длина дуги. | | |
| Методы | | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | | Назначение |
| Arc | public | – | int, int, double | | Конструктор |
| ToString | public | string | – | | Возвращает строковое представление дуги. |
| Свойства | | | | | |
| Имя | Модификатор доступа | Тип | Доступ | Назначение | |
| StartVertex | public | int | get/set | Вершина начала дуги. | |
| EndVertex | public | int | get/set | Вершина конца дуги. | |
| Length | public | double | get/set | Длина дуги. | |

Таблица 2.3

Описание полей методов и свойств структуры Vertex.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| X | public | int | X координата вершины | |
| Y | public | int | Y координата вершины | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Vertex | public | – | int, int | Конструктор |

Таблица 2.4

Описание полей методов и свойств класса Digraph.cs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Методы | | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | | Назначение |
| AddArc | public | void | Arc, int | | Добавляет ребро в конкретное место в списке. |
| AddVertex | public | void | Vertex, int, int, int, int | | Добавляет вершину в конкретное место в списке. |
| Digraph | public | – | – | | Конструктор |
| RemoveArc | public | void | int | | Удаляет ребро. |
| RemoveVertex | public | void | int | | Удаляет вершину. |
| ResetStock | public | void | – | | Очищает сток. |
| SetTimeTillTheEndOfRefractoryPeriod | public | void | – | | Устанавливает таймеры, отсчитывающие время до конца периода восстановления. |
| Свойства | | | | | |
| Имя | Модификатор доступа | Тип | Доступ | Назначение | |
| AdjacencyMatrix | public | double[,] | get | Матрица смежности. | |
| Arcs | public | List<Arc> | get/set | Список дуг. | |
| RefractoryPeriods | public | List<int> | get/set | Список периодов восстановления вершин. | |
| State | public | List<int> | get/set | Список начальных состояний вершин. | |
| Stock | public | List<int> | get/set | Список индексов стоковых вершин | |
| Thresholds | public | List<int> | get/set | Список порогов вершин. | |
| TimeTillTheEndOfRefractoryPeriod | public | List<Timer> | get/set | Список таймеров, отсчитывающих время до конца периода восстановления. | |
| Vertices | public | List<Vertex> | get/set | Список вершин. | |

Таблица 2.5

Описание полей методов и свойств класса DigraphChangedEventArgs.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| Index | public | int | Индекс удаленного элемента. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| DigraphChangedEventArgs | public | – | int | Конструктор |

Таблица 2.6

Описание полей методов и свойств интерфейса ICommand.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Execute | – | void | – | Выполняет команду. |
| UnExecute | – | void | – | Отменяет команду. |

Таблица 2.7

Описание полей методов и свойств класса AddArcCommand.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| arc | private | Arc | Добавляемая дуга. | |
| digraph | private | Digraph | Орграф, с которым ведется работа. | |
| index | private | int | Индекс добавляемой дуги. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| AddArcCommand | public | – | Digraph, Arc | Конструктор |
| Execute | public | void | – | Добавляет дугу в граф. |
| UnExecute | public | void | – | Удаляет добавленную дугу. |

Таблица 2.8

Описание полей методов и свойств класса AddVertexCommand.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| digraph | private | Digraph | Орграф, с которым ведется работа. | |
| index | private | int | Индекс добавляемой вершины. | |
| refractoryPeriod | private | int | Период восстановления вершины. | |
| state | private | int | Начальное состояние вершины. | |
| threshold | private | int | Порог вершины. | |
| vertex | private | Vertex | Добавляемая вершина. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| AddVertexCommand | public | – | Digraph, Vertex | Конструктор |
| Execute | public | void | – | Добавляет вершину в граф. |
| UnExecute | public | void | – | Удаляет добавленную вершину. |

Таблица 2.9

Описание полей методов и свойств класса ChangeArcLengthCommand.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| digraph | private | Digraph | Орграф, с которым ведется работа. | |
| index | private | int | Индекс изменяемой дуги. | |
| newValue | private | double | Новая длина дуги. | |
| oldValue | private | double | Старая длина дуги. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| ChangeArcLengthCommand | public | – | Digraph, int, double, double | Конструктор |
| Execute | public | void | – | Изменяет длину дуги. |
| UnExecute | public | void | – | Возвращает старую длину дуги. |

Таблица 2.10

Описание полей методов и свойств класса ChangeColorCommand.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| newColor | private | Color | Новый цвет. | |
| oldColor | private | Color | Старый цвет. | |
| target | private | GraphDrawing | Объект, производящий отрисовку графа. | |
| type | private | Type | Тип объекта, цвет которого меняется. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| ChangeColorCommand | public | – | GraphDrawing, Type, Color, Color | Конструктор |
| Execute | public | void | – | Изменяет цвет отрисовки компонента графа. |
| UnExecute | public | void | – | Возвращает старый цвет отрисовки. |

Таблица 2.11

Описание полей методов и свойств класса EraseArcCommand.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| arc | private | Arc | Удаляемая дуга. | |
| digraph | private | Digraph | Орграф, с которым ведется работа. | |
| index | private | int | Индекс удаляемой дуги. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| EraseArcCommand | public | – | Digraph, Arc | Конструктор |
| Execute | public | void | – | Удаляет дугу из графа. |
| UnExecute | public | void | – | Восстанавливает дугу. |

Таблица 2.12

Описание полей методов и свойств класса EraseVertexCommand.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| arcIndices | private | List<int> | Список индексов инцидентных дуг. | |
| digraph | private | Digraph | Орграф, с которым ведется работа. | |
| incidentArcs | private | List<Arc> | Список инцидентных дуг. | |
| index | private | int | Индекс удаляемой вершины. | |
| refractoryPeriod | private | int | Период восстановления вершины. | |
| state | private | int | Начальное состояние вершины. | |
| threshold | private | int | Порог вершины. | |
| vertex | private | Vertex | Удаляемая вершина. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| EraseVertexCommand | public | – | Digraph, Vertex | Конструктор |
| Execute | public | void | – | Удаляет вершину из графа. |
| UnExecute | public | void | – | Восстанавливает вершину. |

Таблица 2.13

Описание полей методов и свойств класса MoveDigraphCommand.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| digraph | private | Digraph | Орграф, с которым ведется работа. | |
| xCoefficient | private | int | Коэффициент смещения по оси X. | |
| yCoefficient | private | int | Коэффициент смещения по оси Y. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Execute | public | void | – | Двигает граф. |
| MoveDigraphCommand | public | – | Digraph, int, int | Конструктор |
| UnExecute | public | void | – | Возвращает граф на прежнее место. |

Таблица 2.14

Описание полей методов и свойств класса MoveVertexCommand.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| digraph | private | Digraph | Орграф, с которым ведется работа. | |
| index | private | int | Индекс двигаемой вершины. | |
| oldPoint | private | Point | Старые координаты вершины. | |
| newPoint | private | Point | Новые координаты вершины. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Execute | public | void | – | Двигает вершину. |
| MoveVertexCommand | public | – | Digraph, int, Point, Point | Конструктор |
| UnExecute | public | void | – | Возвращает вершину на прежнюю точку. |

Таблица 2.15

Описание полей методов и свойств класса ResizeDigraphCommand.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| coefficient | private | double | Коэффициент изменения размера. | |
| digraph | private | Digraph | Орграф, с которым ведется работа. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Execute | public | void | – | Меняет размер графа. |
| ResizeDigraphCommand | public | – | Digraph, double | Конструктор |
| UnExecute | public | void | – | Возвращает графу прежний размер. |

Таблица 2.16

Описание полей методов и свойств структуры CommandsManager.cs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Методы | | | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | | Назначение | |
| CommandsManager | public | – | – | | Конструктор | |
| Execute | public | void | ICommand | | Выполняет команду и помещает ее в стек Undo. | |
| Redo | public | void | – | | Возвращает последнюю отмененную команду. | |
| Undo | public | void | – | | Отменяет последнюю команду. | |
| Свойства | | | | | | |
| Имя | Модификатор доступа | Тип | | Доступ | | Назначение |
| CanRedo | public | bool | | get | | Указывает, есть ли команды в стеке Redo. |
| CanUndo | public | bool | | get | | Указывает, есть ли команды в стеке Undo. |
| RedoStack | private | Stack<ICommand> | | get | | Стек команд для отмены. |
| UndoStack | private | Stack<ICommand> | | get | | Стек команд для возврата. |

Таблица 2.17

Описание полей методов и свойств класса ConnectivityCheck.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля |  |  |  | |
| Имя | Модификатор доступа | Тип | Назначение | |
| adjacencyList | private | List<int> | Список смежности графа. | |
| numberOfVertices | private | int | Количество вершин графа. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| AddArcs | public | void | Arc | Добавляет дугу. |
| ConnectivityCheck | public | – | int | Конструктор. |
| DFS | private | void | Int, bool[] | Поиск в глубину и запоминание посещенных вершин. |
| GetInvertedGraph | private | ConnectivityCheck | – | Инвертирует граф. |
| IsGraphValid | public | bool | Digraph | Определяет валидность графа для моделирования движения точек. |
| IsStronglyConnected | public | bool | – | Проверят граф на сильную связность. |

Таблица 2.18

Описание полей методов и свойств класса GraphDrawing.cs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Поля | |  | |  | |  | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | | |
| \_radius | | private | | int | | Радиус вершин. | | |
| arcsPen | | private | | Pen | | Используется для отрисовки дуг. | | |
| brush | | private | | Brush | | Используется для отрисовки надписей. | | |
| graphics | | private | | Graphics | | Используется для рисования фигур. | | |
| font | | private | | Font | | Шрифт нумерации вершин. | | |
| highlightPen | | private | | Pen | | Используется для выделения вершин. | | |
| highlightSandpilePen | | private | | Pen | | Используется для выделения вершин. | | |
| incidenceList | | private | | List<Arc>[] | | Список смежности графа. | | |
| sandpileFont | | private | | Font | | Шрифт, использующийся при отрисовки графа в процессе моделирования движения песка. | | |
| sandpilePalette | | private | | Color[] | | Палитра цветов для обозначения состояния вершин в процессе моделирования движения песка. | | |
| verticesPen | | private | | Pen | | Используется для отрисовки вершин. | | |
| Методы | | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| ClearTheSurface | | public | | void | | – | | Очищает изображение. |
| Dispose | | public | | void | | – | | Освобождает все используемые классом освобождаемые объекты |
| DrawArc | | public | | void | | Vertex, Vertex, Arc, int, int, double | | Рисует дугу. |
| DrawDot | | public | | void | | PointF | | Рисует точку на графе. |
| DrawGraph | | public | | void | | Digraph, int, int double | | Полностью рисует граф. |
| DrawGraphSandpile | | public | | void | | Digraph, bool, int, int double | | Полностью рисует граф, используя Sandpile палитру. |
| DrawVertex | | public | | void | | int, int, int, Pen | | Рисует вершину. |
| DrawVertices | | public | | void | | Digraph, int, int double | | Рисует все вершины графа. |
| DrawVerticesSandpile | | public | | void | | Digraph, int, int double | | Рисует все вершины графа, используя Sandpile палитру. |
| GetGradientColors | | private | | Color[] | | Color, Color, int | | Возвращает массив градиентных цветов заданного размера с заданным начальным и конечным цветом. |
| GetSandpilePalette | | private | | Color[] | | int | | Возвращает палитру цветов для отрисовки графа в процессе моделирования движения песка. |
| GraphDrawing | | public | | – | | int, int | | Конструктор. |
| HighlightVertex | | public | | void | | Vertex | | Подсвечивает вершину. |
| HighlightVertexToAddSand | | public | | void | | Vertex | | Подсвечивает вершину, выбранную для добавления песка. |
| UnhighlightVertex | | public | | void | | Vertex | | Снимает выделение с вершины. |
| Свойства | | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Доступ | | Назначение | |
| ArcsColor | public | | Color | | get/set | | Цвет отрисовки дуг. | |
| BackColor | public | | Color | | get/set | | Цвет фона. | |
| Image | public | | Bitmap | | get/set | | Созданное изображение. | |
| R | public | | int | | get/set | | Радиус вершины. | |
| SandpilePalette | public | | Color[] | | get/set | | Палитра цветов для обозначения состояния вершин в процессе моделирования движения песка. | |
| Size | public | | Size | | get/set | | Размер изображения | |
| VerticesColor | public | | Color | | get/set | | Цвет отрисовки вершин. | |

Таблица 2.19

Описание полей методов и свойств класса ChartWindow.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| AvalancheSizesDistributionChartPrepare | public | void | – | Меняет настройки соответствующих контролов для отображения графика распределения лавин. |
| ChartWindow | public | – | – | Конструктор. |
| SaveAll\_Click | private | void | Object, EventArgs | Открывает диалог для сохранения графика целиком. |
| SaveData\_Click | private | void | Object, EventArgs | Открывает диалог для сохранения данных графика. |
| SaveImage\_Click | private | void | Object, EventArgs | Открывает диалог для сохранения изображения графика. |

Таблица 2.20

Описание полей методов и свойств класса MovementModeling.cs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Поля | |  | |  | |  | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | | |
| actions | | private | | MovementModelingActions[] | | Массив дополнительных необходимых действий. | | |
| avalanche | | private | | bool[] | | Количество значений true – размер текущей лавины. | | |
| digraph | | private | | Digraph | | Орграф, с которым ведется работа. | | |
| distributionChart | | private | | ChartWindow | | Окно с графиком распределения лавин. | | |
| DrawingSurface | | public | | PictureBox | | Поверхность для отображения анимации. | | |
| GraphDrawing | | public | | GraphDrawing | | Используется для отрисовки графа. | | |
| incidenceList | | private | | List<Arc>[] | | Список инцидентности. | | |
| involvedArcs | | private | | List<Arc> | | Список дуг, по которым движутся точки. | | |
| mainTimer | | private | | Timer | | По тику таймера осуществляется обновление. | | |
| MovementGif | | public | | GifBitmapEncoder | | GIF-изображения процесса. | | |
| numberOfDotsChart | | private | | ChartWIndow | | Окно с графиком изменения числа точек. | | |
| releaseCondition | | private | | Predicate<int> | | Условие выпускания вершинами точек. | | |
| SandpileChartTypes | | public | | SandpileChartType[] | | Массив необходимых для построения типов графика. | | |
| speed | | private | | Double | | Скорость точек. | | |
| stateChange | | private | | Action<int> | | Изменение состояния вершины после выпуска точек. | | |
| stopwatches | | private | | List<Stopwatch> | | Указывают время, прошедшее с момента выпуска конкретной точки. | | |
| stopwatchTime | | private | | Stopwatch | | Отсчитывает время, прошедшее с последнего обновления зафиксированной точки. | | |
| time | | private | | double | | Подсчитанное время процесса в миллисекундах. | | |
| type | | private | | MovementModelingType | | Тип моделирования. | | |
| Методы | | | | | | | | |
| Имя | | Модификатор доступа | | Тип | Аргументы | | Назначение | |
| AddAvalancheSize | | private | | void | – | | Добавляет размер текущей лавины на график. | |
| AddNumberOfDotsChartPoint | | private | | void | long, int | | Добавляет текущее количество точек на график. | |
| ChangeChartInterval | | private | | void | Chart | | Подстраивает интервалы графиков под данные. | |
| CheckDotsNumber | | private | | void | int | | Проверяет, не превосходит ли число точек допустимый предел. | |
| GetIncidenceList | | public | | List<Arc>[] | Digraph | | Строит список инцидентности графа. | |
| GetPoint | | public | | PointF | Vertex, Vertex, double, Stopwatch | | Находит точку на ребре, в которой находится движущаяся точка. | |
| GetTime | | public | | double | double double | | Находит время, необходимое, чтобы пройти данное расстояние с данной скоростью. | |
| Go | | public | | void | – | | Запускает или продолжает движение. | |
| MovementModeling | | public | | void | Digraph, double, MovementModelingType, MovementModelingActions[] | | Конструктор. | |
| PrepareBasicCharts | | private | | void | – | | Подготавливает окно для отображения графика количества точек. | |
| PrepareSandpileCharts | | private | | void | – | | Подготавливает окно для отображения графика распределения лавин. | |
| ProcessDots | | private | | void | – | | Обрабатывает движущиеся точки. | |
| ProcessVertices | | private | | void | – | | Обрабатывает вершины, чтобы выпустить точки. | |
| ReleaseDots | | private | | void | int | | Выпускает новые точки. | |
| StartMovementModeling | | public | | void | – | | Проводит подготовительные процессы и запускает движение. | |
| StartNewTimers | | private | | void | int | | Запускает новые таймеры. | |
| Stop | | public | | void | – | | Останавливает движение. | |
| TickAddFrame | | private | | void | Object, EventArgs | | Добавляет в GIF новый кадр. | |
| TickModeling | | private | | void | Object, EventArgs | | Моделирует движение, производит анимацию. | |
| UpdateChart | | private | | void | int | | Обновляет график при изменении числа точек. | |
| Свойства | | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Доступ | | | Назначение |
| isActive | public | | bool | | get/set | | | Показывает, активно ли в данный момент движение. |
| IsMovementEnded | public | | bool | | get | | | Определяет, окончено ли движение. |
| IsMovementEndedBasic | private | | bool | | get | | | Определяет, окончено ли движение при стандартном моделировании. |
| IsMovementEndedSandpile | private | | bool | | get | | | Определяет, окончено ли движение при моделировании движения песка. |

Таблица 2.21

Описание полей методов и свойств класса MovementTickEventArgs.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля |  |  |  | |
| Имя | Модификатор доступа | Тип | Назначение | |
| ElapsedTime | public | long | Прошедшее время в миллисекундах. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| MovementTickEventArgs | public | – | long | Конструктор. |

Таблица 2.22

Описание членов перечисления MovementModelingActions.cs

|  |  |  |
| --- | --- | --- |
| Константа |  |  |
| Имя | Значение | Назначение |
| Animation | 0 | Анимация процесса. |
| Chart | 1 | Построение графика. |
| Gif | 2 | Сохранение GIF-изображения. |

Таблица 2.23

Описание членов перечисления MovementModelingType.cs

|  |  |  |
| --- | --- | --- |
| Константа |  |  |
| Имя | Значение | Назначение |
| Basic | 0 | Представляет стандартный тип моделирования. |
| Sandpile | 1 | Представляет тип моделирования модель песка. |

Таблица 2.24

Описание членов перечисления SandpileChartType.cs

|  |  |  |
| --- | --- | --- |
| Константа |  |  |
| Имя | Значение | Назначение |
| NumberOfDotsChart | 0 | График изменения количества точек. |
| AvalancheSizesDistributionChart | 1 | График распределения размеров лавин. |

Таблица 2.25

Описание полей методов и свойств класса DigraphComponentsRemover.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| FindSelectedArc | private | int | int, int, Digraph | Ищет дугу в точке. |
| IsArcSelected | private | bool | int, int, int, int, int, int, | Определяет, проходит ли дуга через точку. |
| TryToDeleteArcAt | public | bool | int, int, Digraph, Arc | Ищет дугу в точке и помечает ее для удаления. |
| TryToDeleteVertexAt | public | bool | int, int, Digraph, float, int | Ищет вершину в точке и помечает ее для удаления. |

Таблица 2.26

Описание полей методов и свойств класса DigraphInformationDemonstration.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| DisplayGraphAdjacencyInfo | public | void | double[,], DataGridView | Выводит матрицу смежности в таблицу. |
| DisplayGraphParameters | public | void | Digraph, DataGridView | Выводит параметры графа в таблицу. |
| DisplaySandpileColors | public | void | GraphDrawing, DataGridView | Выводит палитру цветов в таблицу. |

Таблица 2.27

Описание полей методов и свойств класса RandomDigraphGeneratorForm.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поля |  |  |  | |
| Имя | Модификатор доступа | Тип | Назначение | |
| height | private | int | Высота поля отображения графа. | |
| Rnd | private | Random | Генератор случайных значений. | |
| width | private | int | Ширина поля отображения графа. | |
| Методы | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Button2\_Click | private | void | Object, EventArgs | Генерирует случайный сильно связный граф с выбранным количеством вершин. |
| Cancel\_Click | private | void | Object, EventArgs | Закрывает окно. |
| RandomGraphGeneratorForm | public | – | int, int | Конструктор класса |
| VNRandom\_Click | private | void | Object, EventArgs | Присваивает количеству вершин случайное значение. |
| Свойства | | | | |
| Имя | Модификатор доступа | Тип | Доступ | Назначение |
| Digraph | public | Digraph | get/set | Сгенерированный граф. |

Таблица 2.28

Описание полей методов и свойств класса SquareLatticeForm.cs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поля |  |  |  | | |
| Имя | Модификатор доступа | Тип | Назначение | | |
| rnd | private | Random | Генератор случайных значений. | | |
| height | private | int | Высота поля отображения графа. | | |
| width | private | int | Ширина поля отображения графа. | | |
| Методы | | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | | Назначение |
| AddArcs | private | void | – | | Добавляет дуги в формируемую решетку. |
| AddVertices | private | void | – | | Добавляет вершины в формируемую решетку. |
| Cancel\_Click | private | void | Object, EventArgs | | Закрывает окно. |
| OK\_Click | private | void | Object, EventArgs | | Строит квадратную решетку заданных размеров. |
| ParamsCheckBox\_CheckedChanged | private | void | Object, EventArgs | | Определяет необходимость заполнения параметров графа случайными значениями. |
| SquareLatticeForm | public | – | int, int | | Конструктор класса |
| Свойства | | | | | |
| Имя | Модификатор доступа | Тип | Доступ | Назначение | |
| SquareLatticeDigraph | public | Digraph | get/set | Построенная квадратная решётка. | |

Таблица 2.29

Описание полей методов и свойств класса TriangularLatticeForm.cs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поля |  |  |  | | |
| Имя | Модификатор доступа | Тип | Назначение | | |
| rnd | private | Random | Генератор случайных значений. | | |
| height | private | int | Высота поля отображения графа. | | |
| width | private | int | Ширина поля отображения графа. | | |
| Методы | | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | | Назначение |
| AddArcs | private | void | – | | Добавляет дуги в формируемую решетку. |
| AddVertices | private | void | – | | Добавляет вершины в формируемую решетку. |
| Cancel\_Click | private | void | Object, EventArgs | | Закрывает окно. |
| OK\_Click | private | void | Object, EventArgs | | Строит треугольную решетку заданных размеров. |
| ParamsCheckBox\_CheckedChanged | private | void | Object, EventArgs | | Определяет необходимость заполнения параметров графа случайными значениями. |
| TriangularLatticeForm | public | – | int, int | | Конструктор класса |
| Свойства | | | | | |
| Имя | Модификатор доступа | Тип | Доступ | Назначение | |
| TriangularLatticeDigraph | public | Digraph | get/set | Построенная треугольная решётка. | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Изм. | Номера листов | | | | Всего листов (страниц) в документе | № документа | Входя- щий № сопро- водит. докум. и дата | Под- пись | Дата |
| изменен- ных | заменен- ных | новых | аннулиро- ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |