Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Языки программирования (ЯП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

МОЛЕКУЛЯРНЫЙ РЕДАКТОР

БГУИР КР 1-40 01 01 205 ПЗ

Студент: гр. 951002 Будович И.В.

Руководитель: асс. Шостак Е. В.

Минск 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

––––––––––––––––––––––––

(подпись)

––––––––––––––––– 2020 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Будовичу Ивану Витальевичу

1. Тема работы Молекулярный редактор

2. Срок сдачи студентом законченной работы––17.12.2020 г.–––

3. Исходные данные к работе язык программирования C++

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение.

1. Анализ прототипов, литературных источников и моделирование предметной области;

2. Анализ требований к программному средству и разработка функциональных требований;

3. Проектирование программного средства;

4. Создание (конструирование) программного средства;

5. Тестирование, проверка работоспособности и анализ полученных результатов;

6. Руководство по установке и использованию;

Список используемой литературы

Заключение

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. " Молекулярный редактор", А1, схема программы, чертеж.

6. Консультант по курсовой работе

Шостак Е. В.

7. Дата выдачи задания 08.09.2020

8. Календарный график работы над курсовой работой на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1 к 15.09.2020 – 15 % готовности работы;

разделы 2, 3 к 15.10.2020 – 30 % готовности работы;

разделы 4, 5 к 15.11.2020 – 60 % готовности работы;

раздел 6 к 15.12.2020 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 17.12.2020 – 100 % готовности работы.

Защита курсовой работы с 13.12.2020 по 17.12.2020 г.––––––––––––––––––––

РУКОВОДИТЕЛЬ–––––– Шостак Е. В.

(подпись)

Задание принял к исполнению 08.09.2020–––\_\_\_\_\_\_––

(дата и подпись студента)

Содержание

[Содержание 4](#_Toc57896294)

[Введение 6](#_Toc57896295)

[1 Анализ прототипов, литературных источников и моделирование предметной области 7](#_Toc57896296)

[1.1 Молекулярные вещества 7](#_Toc57896297)

[1.2 Способы представления молекулярных веществ 7](#_Toc57896298)

[1.3 Цветовое представление атомов в трехмерных моделях 10](#_Toc57896299)

[1.4 Анализ прототипов 11](#_Toc57896300)

[1.5 Постановка задачи 14](#_Toc57896301)

[2 Анализ требований к программному средству и разработка функциональных требований 15](#_Toc57896302)

[2.1 Описание функциональных требований 15](#_Toc57896303)

[2.2 Описание прочих требований 15](#_Toc57896304)

[3 Проектирование программного средства 17](#_Toc57896305)

[3.1 Разработка алгоритма рисования скелетных формул 17](#_Toc57896306)

[3.2 Разработка алгоритма преобразования двумерного представления в трехмерное 17](#_Toc57896307)

[3.3 Разработка алгоритма рисования полусферической модели 18](#_Toc57896308)

[4 Конструирование программного средства 19](#_Toc57896309)

[4.1 Проектирование интерфейса программы 19](#_Toc57896310)

[4.2 Структура модулей программы 20](#_Toc57896311)

[4.3 Описание класса Application 20](#_Toc57896312)

[4.4 Описание класса MainFrame 20](#_Toc57896313)

[4.5 Описание класса Canvas2D 26](#_Toc57896314)

[4.6 Описание класса Canvas3D 28](#_Toc57896315)

[4.7 Описание класса Converter 30](#_Toc57896316)

[4.8 Описание класса Converter 32](#_Toc57896317)

[4.9 Описание модуля с иконками для приложения 34](#_Toc57896318)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 35](#_Toc57896319)

[5.1 Тестирование алгоритма рисования 35](#_Toc57896320)

[5.2 Тестирование сохранения изображения 36](#_Toc57896321)

[5.3 Тестирование трехмерного моделирования 37](#_Toc57896322)

[6 Руководство по установке и использованию 40](#_Toc57896323)

[Заключение 43](#_Toc57896324)

[Список литературы 44](#_Toc57896325)

[Приложение A 45](#_Toc57896326)

[Приложение Б 46](#_Toc57896327)

[Приложение В 49](#_Toc57896328)

[Листинг файла Application.h 49](#_Toc57896329)

[Листинг файла Application.cpp 49](#_Toc57896330)

[Листинг файла MainFrame.h 49](#_Toc57896331)

[Листинг файла MainFrame.cpp 51](#_Toc57896332)

[Листинг файла Canvas2D.h 55](#_Toc57896333)

[Листинг файла Canvas2D.cpp 56](#_Toc57896334)

[Листинг файла Canvas3D.h 62](#_Toc57896335)

[Листинг файла Canvas3D.cpp 63](#_Toc57896336)

[Листинг файла Converter.h 66](#_Toc57896337)

[Листинг файла Converter.cpp 66](#_Toc57896338)

[Листинг файла structures.h 82](#_Toc57896339)

Введение

Для исследования естественнонаучных дисциплин используются прикладные пакеты программного обеспечения. Они помогают визуализировать информацию, используя общепринятые соглашения, а также научные данные.

Молекулярные вещества – это обширный класс химических веществ. К веществам молекулярного относится большинство веществ, не имеющих в своем составе атомов металлов.

Широким классом веществ молекулярного строения являются органические вещества. Известно 27 миллионов веществ данного класса, что подтверждает их важность и необходимость использования ПО для их изучения и систематизации.

Цель данной курсовой работы – создать молекулярный редактор, способный работать с органическими веществами и другими низкомолекулярными соединениями.

В ходе выполнения курсовой работы я постараюсь найти решение таким техническим вопросам как:

1. Представление химических веществ в виде структур данных;
2. Использование API, работающих с трехмерной графикой.
3. Работа с парадигмой ООП.

В этой пояснительной записке отображены следующие этапы написания курсовой работы:

1. Анализ прототипов, литературных источников и моделирование предметной области;
2. Анализ требований к программному средству и разработка функциональных требований;
3. Проектирование программного средства;
4. Создание (конструирование) программного средства;
5. Тестирование, проверка работоспособности и анализ полученных результатов;
6. Руководство по установке и использованию.
7. Анализ прототипов, литературных источников и моделирование предметной области
   1. Молекулярные вещества

В классической теории химического строения молекула рассматривается как наименьшая стабильная частица вещества, обладающая всеми его химическими свойствами. К молекулам относятся такие известные вещества, как вода, кислород, аммиак, углекислый газ, а также органические вещества, например, метан или этиловый спирт. Органические вещества — соединения углерода с кислородом, водородом, азотом и друг. элементами, встречающиеся в животном и растительном царстве, а также полученные синтетически.

Химические связи в молекулах могут быть одинарными, двойными и тройными, в зависимости от этого определяется их угол.

Некоторые химические вещества, например, органические, сложно представить в виде формулы, так как существует большое количество изомеров, имеющих одинаковый количественный состав. Например, формула может соответствовать n-пентану, 2-метилбутану и 2,2-диметилпропану. В этих целях в химии широкое распространение получили графические способы представления веществ.

* 1. Способы представления молекулярных веществ
     1. Скелетная структурная формула

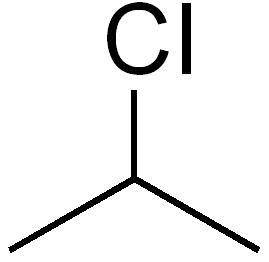


Рисунок 1.1 – представление 2-хлорпропана в виде скелетной структурной формулы

В скелетной структурной формуле длинными чертами представляется связь C-C (основная связь в органической химии), при этом стоящие при атомах углерода атомы водорода опускаются, так как их число легко вычислить исходя из числа связей атома углерода с другими атомами. Связи с атомами других химических элементов обозначаются более короткими чертами. Атомы углерода обозначаются вершинами ломанной, других элементов – соответствующим обозначением в периодической таблице (рис. 1.1).

В соответствии с соглашениями, связи с атомами с и гибридизациями, т.е одинарные и двойные, обозначаются звеньями ломанной, связанными под углом 120 градусов. Связи с гибридизацией (тройные, а также две идущие подряд двойные) обозначаются под углом 180 градусов [1].

Преимущества:

* Удобна для ручного и компьютерного ввода.
* Обладает высокой читабельностью для низкомолекулярных соединений.

Недостатки:

* Не отображает действительную трехмерную структуру.
  + 1. Стержневая модель

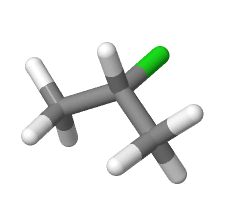


Рисунок 1.2 – представление 2-хлорпропана в виде стержневой модели

Стержневая модель используется редко. Связи в молекуле представлены цилиндрами, со скругленными основаниями, где половина связи обозначена цветом, соответствующим химическому элементу атома (рис. 1.2).

Преимущества:

* Отображает трехмерную структуру.

Недостатки:

* Не реалистична, так как показывает связи, а не атомы.
* Не дает представления о размерах атомов.
  + 1. Шаростержневая модель

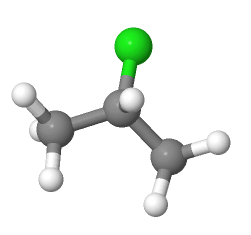


Рисунок 1.3 – представление 2-хлорпропана в виде шаростержневой модели

В шаростержневой модели атомы представлены сферами разных цветов, радиусы которых пропорциональны радиусам Ван-дер-Ваальса для данного химического элемента [2], связи представлены аналогично стержневой модели (рис 1.3).

Преимущества:

* Отображает трехмерную структуру.
* Отображает соотношения радиусов атомов.
* Может быть собрана вручную в реальной жизни.

Недостатки:

* Искажает понятие связи, так как связи являются лишь электронными парами.
  + 1. Полусферическая модель

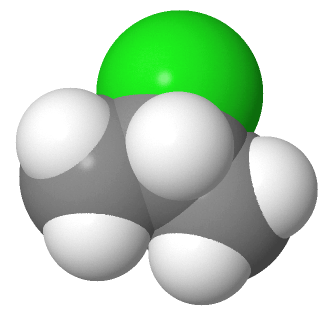


Рисунок 1.4 – представление 2-хлорпропана в виде полусферической модели

Полусферическая модель – трехмерная модель, в которой атомы представлены сферами, радиусы которых пропорциональны радиусам Ван-дер-Ваальса, а расстояния между сферами пропорциональны расстояниям между атомными ядрами (рис. 1.4) [2].

Преимущества:

* Отображает трехмерную структуру.
* Отображает соотношения радиусов атомов и длин связей.
* Наиболее реалистичное представление.

Недостатки:

* Для относительно небольших молекул уже плохо читаема.
* Не дает представления о кратности связи (за исключением меры угла).
  1. Цветовое представление атомов в трехмерных моделях

В химии для обозначения атомов различных химических элементов в молекулярных моделях используется цветовая схема моделей Кори – Полинга – Колтуна (CPK) [3]. Похожая цветовая схема используется Java-приложением Jmol [4].

На таблице 1.1 представлена цветовая схема для наиболее часто встречаемых химических элементов. Исходя из таблицы, видно, что представление Jmol более предпочтительно, так как галогенный ряд представлен более широким спектром цветов.

Таблица 1.1 – цветовая схема отдельных химических элементов

|  |  |  |
| --- | --- | --- |
| Химический элемент | Представление CPK | Представление Jmol |
| Углерод | Черный | Серый |
| Водород | Белый | Белый |
| Кислород | Красный | Красный |
| Азот | Синий | Синий |
| Сера | Желтый | Желтый |
| Фосфор | Фиолетовый | Оранжевый |
| Фтор | Светло-зеленый | Светло-зеленый |
| Хлор | Светло-зеленый | Светло-зеленый |
| Бром | Зеленый | Красно-кирпичный |
| Йод | Темно-зеленый | Фиолетовый |

* 1. Анализ прототипов

Данная курсовая работа ориентирована на создание молекулярного редактора. Для того, чтобы иметь представление о функционале молекулярного редактора, рассмотрим уже существующие в качестве прототипов.

* + 1. Приложение JMol

JMol — программа для просмотра структуры молекул в трёх измерениях. Написана на языке программирования Java [5]. Пример интерфейса представлен на рисунке 1.5.

Преимущества:

* Поддержка макросов и скриптов.
* Возможность сохранения в разные форматы.
* Возможность использования в качестве апплета в веб-странице.

Недостатки:

* Относительно неудобный интерфейс.
* Низкое качество изображения.
* Отсутствие поддержки двухмерных скелетных форм в основной версии.

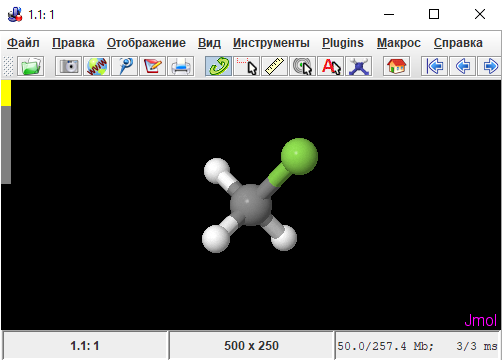


Рисунок 1.5 – интерфейс молекулярного редактора JMol

* + 1. PyMOL

PyMOL – многофункциональная программа для визуализации молекул. Написан на языке Python [6]. Позволяет создавать высококачественные трёхмерные изображения как малых молекул, так и биологических макромолекул, в первую очередь белков. Широко используется в научной литературе. Интерфейс программы представлен на рисунке 1.6.

**Преимущества:**

* Позволяет работать с высокомолекулярными соединениями.
* Позволяет использовать разные модели представления.

**Недостатки:**

* Ориентирован, в основном, на высокомолекулярные органические соединения.
* Отсутствует двумерное представление.

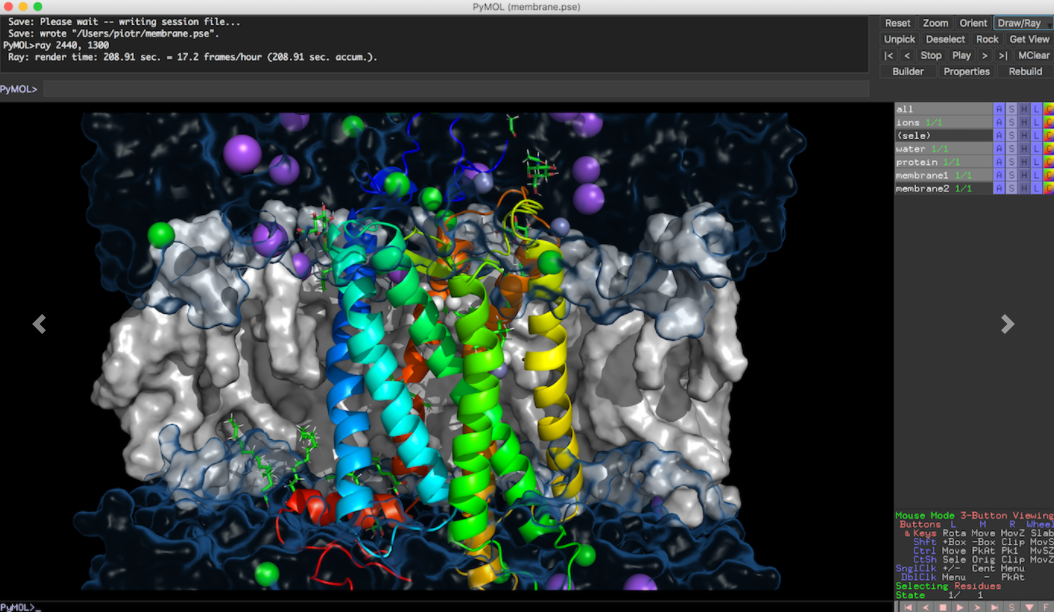


Рисунок 1.6 – интерфейс молекулярного редактора PyMOL

* + 1. Приложение BKChem

BKChem - бесплатный редактор 2D-молекул, написанный на Python 2.5 [7]. Интерфейс программы представлен на рисунке 1.7.

**Преимущества:**

* Удобен в создании скелетных формул.

**Недостатки:**

* Отсутствует трехмерное представление.
* Поддержка приложения прекращена.

Вывод: в ходе работы следует создать программное средство, способное одновременно работать с двух- и трехмерным представлениями молекул.

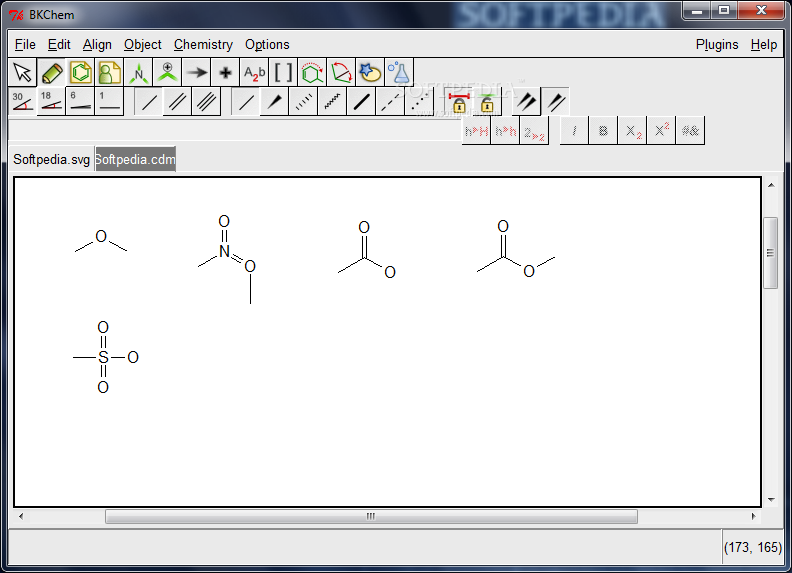


Рисунок 1.7 – интерфейс молекулярного редактора BKChem

* 1. Постановка задачи

Программное средство должно представлять собой редактор низкомолекулярных веществ. Программа должна обеспечивать выполнение перечисленных ниже функций:

* + Работа с двухмерным (скелетная формула) и трехмерным (полусферическая модель) представлениями молекул;
  + возможность выбора основных химических элементов;
  + возможность сохранения полученных изображений в графические форматы.

Пользователь может выбирать химических элемент и необходимую кратность связи и рисовать с помощью компьютерной мыши скелетные формулы. Выходными данными являются полученные изображения.

Пользователь может преобразовать полученную формулу в полусферическую модель, поддерживающую масштабирование.

Для разработки я выбрал среду Соde::Blocks 17.12. Данная среда является бесплатной, она позволяет разрабатывать графический пользовательский интерфейс с помощью открытой библиотеки wxWidgets 3.1.4 [9], а также обеспечивать взаимодействие с другими библиотеками, такими как OpenGL. В качестве языка программирования выбран C++ в связи с поддержкой парадигмы ООП, наличием стандартных шаблонов STL для упрощенной работы с динамическими массивами и строковым типом данных.

1. Анализ требований к программному средству и разработка функциональных требований
   1. Описание функциональных требований

Программное средство для функциональности должно работать с двухмерной и трехмерной графикой. Должен существовать алгоритм преобразования скелетной формулы в полусферическую модель.

Основными функциями ПС являются создание двухмерного изображения и преобразование в трехмерное, сохранение двухмерного и трехмерного изображений.

Для моего программного средства я определил следующие функциональные требования:

* 1. Наличие в верхней части окна программы панели управления, на которой можно выбрать химический элемент, кратность связи, сохранить изображение;
  2. Скелетные формулы рисуются нажатием левой кнопки мыши по одной из вершин формулы или по произвольной части предназначенного для этого экрана при отсутствии изображения;
  3. При нажатии на сохранение изображения отображается диалоговое окно с выбором каталога, имени и расширения файла. К доступным расширениям относятся BMP, PNG, JPEG;
  4. На панели инструментов присутствует кнопка, преобразующая скелетную формулу в полусферическую;
  5. Полусферическая модель может быть масштабирована посредством прокрутки колеса мыши.
  6. **Описание прочих требований**

При выполнении работы определяются дополнительные требования:

* К основным химическим элементам относятся: углерод, кислород, азот, фосфор, сера, фтор, хлор, бром, йод. Водород нельзя выбрать на панели инструментов, свободные связи других атомов заполняются атомами водорода.
* Углерод принимает валентность IV, кислород и сера– II, азот и фосфор – III. Остальные принимают валентность – I.
* Вещество может иметь связь, кратность которого не превышает валентность или число 3.
* Интерфейс программного средства должен быть простым и удобным.
* Программное средство должно работать на устройстве под управлением операционной системы Windows.

1. Проектирование программного средства
   1. Разработка алгоритма рисования скелетных формул
      1. Алгоритм добавления двумерных координат

Область рисования скелетных формул представлена холстом. В начальный момент при нажатии на произвольную область добавляется первая координата. Добавление осуществляется в динамический массив.

Последующие координаты рассчитываются относительно первой. В зависимости от кратности связи определяется, куда будет направлено звено ломанной линии: если связь тройная, или две рядом стоящие связи двойные, то угол между тремя координатами будет равен 180 градусам. В противном случае – 120 градусам. Расчет координат осуществляется вращением точки на окружности с радиусом, равным длине связи, на указанный выше угол.

Помимо координаты вершины, добавляется также предыдущая вершина, сведения об элементе (число свободных связей, кратность связи, атом).

Существует так же «безопасная зона» – 30 пикселей с каждой стороны, в пределах которых нельзя рисовать. Это создано для того, чтобы изображение не выходило за пределы холста.

Схема алгоритма представлена на рисунке Б.1.

* + 1. Алгоритм рисования скелетных формул

После того как координаты скелетной формулы получены, поступает команда перерисовать содержимое холста. Если вершина представлена атомом углерода, то рисуется просто ломанная линия, иначе рисуется укороченная с одного или двух концов линия, а в место расположения вершин вставляется строка, с обозначением химического элемента. Если элемент имеет свободные связи, то к этой строке также приписывается строка с обозначением водорода и числом свободных связей, если оно больше единицы. Схема алгоритма представлена на рисунке Б.2.

* 1. Разработка алгоритма преобразования двумерного представления в трехмерное

Для того, чтобы преобразовать скелетную формулу в полусферическую модель, нужно рассчитать координаты. В данной части важное место занимает именно связь соседних атомов. На начальном этапе проверяется, состоит ли соединения из одного элемента в массиве координат. Если это выполняется, то в массив трехмерных координат подаются уже ранее готовые координаты для метана, аммиака, фосфина, сероводорода, воды и галогеноводородов.

В противном случае строится дерево, где каждый элемент имеет четыре потомка и указатель на предка. Элементы добавляются в дерево на основании соответствия текущей двумерной координаты предыдущей двумерной координате в массиве. Указываются кратности их связей. Если у узла дерева есть свободные для данного элемента связи, то они заполняются водородами.

После построения дерева происходит сортировка потомков для каждого узла так, чтобы впереди была связь с наибольшей кратностью.

Координатой корня объявляется начало координат. Далее рассчитываются координаты: если связь узла или связь потомка равны трем или если связи и узла, и потомка равны двум, то они расположены на прямой линии, в противном случае – рассчитываются под углом 120 градусов с помощью угла , определяемого как угол нормали, и угла между осями абсцисс и аппликат.

После определения всех координат, производится обход дерева, добавления необходимых структур (координат и имя химического элемента) в динамический массив и удаление узлов дерева.

Затем выполняется нормировка координат так, чтобы момент всех координат был в начале координат. Для этого находится среднее арифметическое и вычитается из каждой координаты.

После этого массив передается области моделирования и поступает команда перерисовать содержимое. Схема алгоритма представлена на рисунке Б.3.

* 1. Разработка алгоритма рисования полусферической модели

После того, как координаты получены, определяются цвета и радиусы для каждой сферы. Радиусы пропорциональны реальным радиусам Ван-дер-Ваальса, используемым в благородных газах. Цвета определяются также по химическому элементу аналогично тому, как это используется в приложении JMol. Для того, чтобы большая часть изображения была охвачена, вводится коэффициент масштабирования, в начальный момент равный -7. Он представляет собой дополнительное слагаемое в оси аппликат. Он может быть изменен с помощью прокрутки мыши в пропорции: одно деление к 0.2 единицам оси.

1. Конструирование программного средства
   1. Проектирование интерфейса программы

Используемая мною библиотека wxWidgets позволяет создавать нативные для каждой платформы интерфейсы [9][10].

Рассмотрим компоненты интерфейса, которые я использовал.

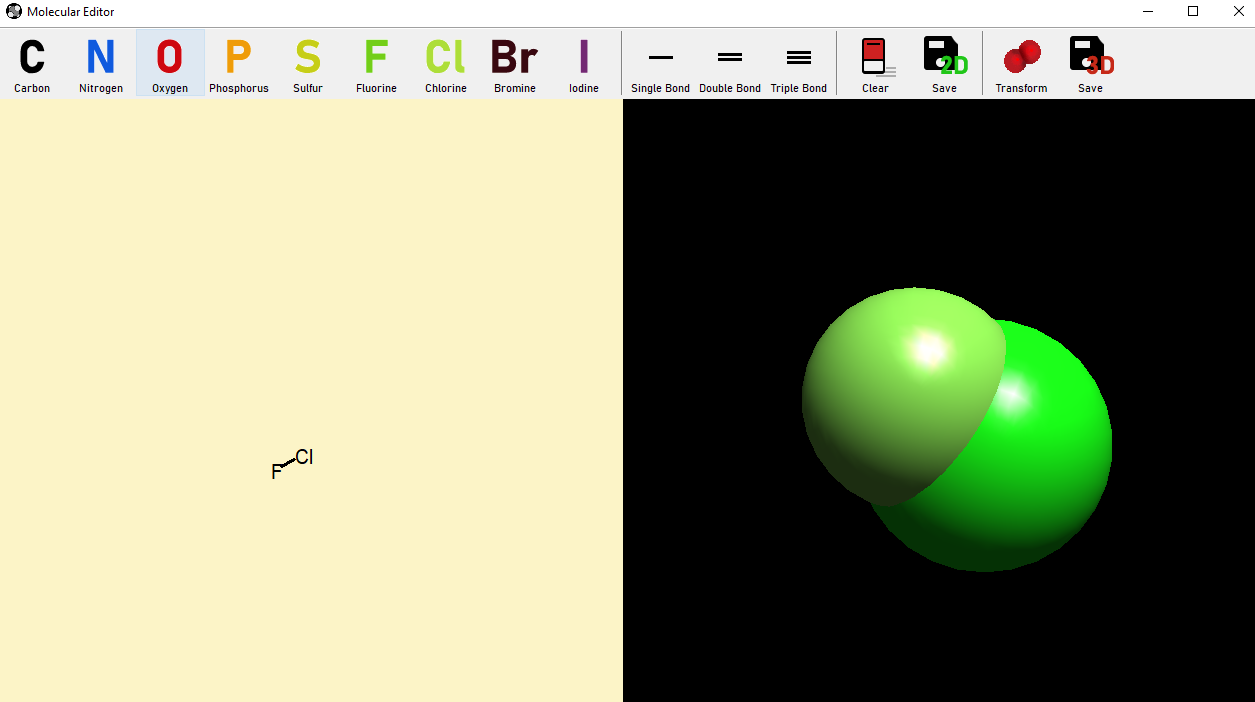


Рисунок 4.1 – интерфейс приложения

Приложение является однооконным. Стандартный размер – 1280x720. Окном является объект класса MainFrame, который унаследован от библиотечного класса wxFrame. В верхней части расположена панель инструментов класса wxToolBar и 16 кнопок класса wxToolBarToolBase.

Вся остальная часть представлена областями: областью рисования и областью моделирования. Область рисования является объектом класса Canvas2D, унаследованным от класса wxPanel с добавлением возможностей рисования. Область моделирования класса Canvas3D является классом-наследником класса wxGLCanvas, предназначенным для работы с OpenGL [8].

* 1. Структура модулей программы

Программа состоит из следующих файлов:

1. Файлы модуля приложения Application.h и Application.cpp;
2. Файлы модуля окна MainFrame.h и MainFrame.cpp;
3. Файлы модуля Canvas2D.h и Canvas2D.cpp;
4. Файлы модуля Canvas3D.h и Canvas3D.cpp;
5. Файлы модуля Converter.h и Converter.cpp;
6. Заголовочный файл со структурами данных structures.h;
7. Заголовочный файл с xpm-иконками xpm\_icons.h
   1. Описание класса Application

Модуль Application является модулем класса Application – приложения, реализующего шаблон «Одиночка». Класс Application является классом-реализатором класса wxApp. Класс не имеет полей данных. В таблице 4.1 представлены открытые методы класса Application.

Таблица 4.1 Открытые методы класса Application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| OnInit | Метод-конструктор приложения | virtual bool OnInit() | - | - |

* 1. Описание класса MainFrame

Модуль MainFrame является модулем класса MainFrame – главного окна приложения. Класс MainFrame наследуется от класса wxFrame. Открытые методы представлены в таблице 4.2, закрытые – в таблице 4.3. Закрытые поля данных описаны в таблице 4.4.

Таблица 4.2 Открытые методы класса MainFrame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| MainFrame | Метод-конструктор | MainFrame(wx Window\* parent, wxWindowID id = -1) | parent | Родительское окно |
| id | Идентифика-тор компонента интерфейса |
| ~MainFrame | Метод-деструктор | virtual ~MainFrame() | - | - |

Таблица 4.3 Закрытые методы класса MainFrame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| CarbonClick | Метод-события при нажатии на кнопку «Углерод» на панели инструментов | void CarbonClick(wxCommandEvent& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| NitrogenClick | Метод-события при нажатии на кнопку «Азот» на панели инструментов | void NitrogenClick( wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| OxygenClick | Метод-события при нажатии на кнопку «Кислород» на панели инструментов | void OxygenClick( wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| SulfurClick | Метод-события при нажатии на кнопку «Сера» на панели инструментов | void SulfurClick(wxCommandEvent& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |

Продолжение таблицы 4.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| Phosphorus Click | Метод-события при нажатии на кнопку «Фосфор» на панели инструментов | void Phosphorus Click(wx Command Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| FluorineClick | Метод-события при нажатии на кнопку «Фтор» на панели инструментов | void FluorineClick(wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| ChlorineClick | Метод-события при нажатии на кнопку «Хлор» на панели инструментов | void ChlorineClick(wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| BromineClick | Метод-события при нажатии на кнопку «Бром» на панели инструментов | void BromineClick(wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| IodineClick | Метод-события при нажатии на кнопку «Йод» на панели инструментов | void IodineClick( wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| SingleClick | Метод-события при нажатии на кнопку «Одинарная связь» на панели инструментов | void SingleClick( wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |

Продолжение таблицы 4.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| DoubleClick | Метод-события при нажатии на кнопку «Двойная связь» на панели инструментов | void DoubleClick( wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| TripleClick | Метод-события при нажатии на кнопку «Тройная связь» на панели инструментов | void TripleClick(wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| EraseClick | Метод-события при нажатии на кнопку «Очистить» на панели инструментов | void EraseClick(wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| Save2DClick | Метод-события при нажатии на кнопку «Сохранить» на панели инструментов | void Save2DClick(wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| Save3DClick | Метод-события при нажатии на кнопку «Сохранить» на панели инструментов | void Save3DClick( wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |
| CalotteClick | Метод-события при нажатии на кнопку «Преобразо-ние» | void CalotteClick( wxCommand Event& event) | event | Инкапсуляция события нажатия по кнопке панели инструментов |

Таблица 4.4 Закрытые поля данных класса MainFrame

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение параметра** |
| MainCanvas | Canvas2D\* | Холст для двумерного рисования |
| OtherCanvas | Canvas3D\* | Холст для трёхмерного моделирования |
| MainToolBar | wxToolBar\* | Панель инструментов |
| ToolBarItem\_ Carbon | wxToolBarToolBase\* | Кнопка «Углерод» на панели инструментов |
| ToolBarItem\_ Nitrogen; | wxToolBarToolBase\* | Кнопка «Азот» на панели инструментов |
| ToolBarItem\_ Oxygen; | wxToolBarToolBase\* | Кнопка «Кислород» на панели инструментов |
| ToolBarItem\_ Phosphorus; | wxToolBarToolBase\* | Кнопка «Фосфор» на панели инструментов |
| ToolBarItem\_ Sulfur; | wxToolBarToolBase\* | Кнопка «Сера» на панели инструментов |
| ToolBarItem\_Fluorine | wxToolBarToolBase\* | Кнопка «Фтор» на панели инструментов |
| ToolBarItem\_Chlorine | wxToolBarToolBase\* | Кнопка «Хлор» на панели инструментов |
| ToolBarItem\_Bromine | wxToolBarToolBase\* | Кнопка «Бром» на панели инструментов |
| ToolBarItem\_Iodine | wxToolBarToolBase\* | Кнопка «Йод» на панели инструментов |
| ToolBarItem\_Single | wxToolBarToolBase\* | Кнопка «Одинарная связь» на панели инструментов |
| ToolBarItem\_Double | wxToolBarToolBase\* | Кнопка «Двойная связь» на панели инструментов |
| ToolBarItem\_Triple | wxToolBarToolBase\* | Кнопка «Тройная связь» на панели инструментов |
| ToolBarItem\_Eraser | wxToolBarToolBase\* | Кнопка «Очистить» на панели инструментов |
| ToolBarItem\_Calotte | wxToolBarToolBase\* | Кнопка «Преобразовать» на панели инструментов |
| ToolBarItem\_Save2D | wxToolBarToolBase\* | Кнопка «Сохранить» на панели инструментов для двумерного изображения |
| ToolBarItem\_Save3D | wxToolBarToolBase\* | Кнопка «Сохранить» на панели инструментов для трехмерной модели |

Продолжение таблицы 4.4

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение параметра** |
| ID\_Main Canvas | static const long | Идентификатор компонента интерфейса |
| ID\_Other Canvas | static const long | Идентификатор компонента интерфейса |
| ID\_MainToolBar | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBar Item\_ Carbon | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBar Item\_ Nitrogen; | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBar Item\_ Oxygen; | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBar Item\_ Phosphorus; | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBar Item\_ Sulfur; | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBar Item\_Fluorine | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBar Item\_Chlorine | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBarItem\_Bromine | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBarItem\_Iodine | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBarItem\_Single | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBarItem\_Double | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBarItem\_Triple | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBarItem\_Eraser | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBarItem\_Calotte | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBarItem\_Save2D | static const long | Идентификатор компонента интерфейса |
| ID\_ToolBarItem\_Save3D | static const long | Идентификатор компонента интерфейса |

* 1. Описание класса Canvas2D

В данном модуле объявлен класс Canvas2D, подкласс класса wxPanel, имеющий контекст устройства (Device Context, DC) для графических возможностей.

Реализованные методы являются методами-событиями (нажатие ЛКМ, рисование), служебными методами (расчет координат, определение строки для вершины) или связывают главное окно с холстом (сеттеры, метод сохранения изображения). Открытые методы класса Canvas2D представлены в таблице 4.5, закрытые методы – в таблице 4.6, открытые поля данных – 4.7, закрытые поля – 4.8.

Таблица 4.5 Открытые методы класса Canvas2D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| Canvas2D | Метод-конструктор | Canvas2D (wxFrame\* parent, wxWindowID id=wxID\_ANY, const wxPoint &pos= wxDefault Position, const wxSize &size = wxSize (625,648), long style=wxTAB\_ TRAVERSAL, const wxString &name= wxPanelName Str) | parent | Родительское окно |
| id | Идентифика-тор компонента интерфейса |
| pos | Положение на окне |
| size | Размер компонента |
| style | Стиль компонента |
| name | Имя панели |
| ~Canvas2D | Метод-деструктор | virtual ~ Canvas2D() | - | - |
| paint | Событие рисования | void paint(wxPaintEvent &event) | event | Инкапсуляция события |
| leftClick | Событие нажатия ЛКМ | void leftClick(wxMouseEvent& event) | event | Инкапсуляция события |
| clearVertex | Очистка | void clearVertex() | - | - |

Продолжение таблицы 4.5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| setBond | Сеттер кратности связи | void setBond (char bondNumber) | bondNumber | Устанавливае-мая кратность связи |
| setActive Element | Сеттер элемента | void setActiveElement (elem activeElement) | activeElement | Устанавливае-мый элемент |
| getBitmap | Получение растрового изображения | wxBitmap getBitmap() | - | - |

Таблица 4.6 Закрытые методы класса Canvas 2D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| render | Метод рисования, вызываемый из сохранения или события рисования | void render(wxDC& dc) | dc | Контекст устройства (холст для рисования или растровое изображение) |
| getVertex Location | Расчет координаты путем ее вращения вокруг окружности | void getVertexLocation (short int x0,short int y0,short int x,short int y,short int value,short int &x1,short int &y1) | x0 | Точка на окружности |
| y0 |
| x | Центр окружности |
| y |
| value | Опция вращения |
| x1 | Результат |
| y1 |
| getStringForVertex | Получение строки для вершины | std::string getStringForVertex (atom active) | active | Атом вершины |
| getPossibleBonds | Получение валентности элемента | char getPossibleBonds (elem activeElement) | activeElement | Элемент |

Продолжение таблицы 4.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| isWithin SafeZone | Проверка на нахождение в «безопасной зоне» | bool isWithinSafeZone (short int x0, short int y0) | x0 | Координата |
| y0 |

Таблица 4.7 Открытые поля данных класса Canvas 2D

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение параметра** |
| vertices | std::vector <vertex2D> | Массив двумерных вершин |

Таблица 4.8 Закрытые поля данных класса Canvas 2D

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение параметра** |
| counter | short int | Счетчик координат |
| length | const char | Длина связи |
| bond | char | Текущая кратность связи |
| active | elem | Текущий элемент |

* 1. Описание класса Canvas3D

Класс Canvas3D является классом-реализацией класса wxGLCanvas. Данный класс поддерживает работу с OpenGL, в связи с чем поддерживает рисование продвинутой двумерной и трехмерной графики. В своей работе я использовал функции OpenGL и функции библиотеки утилит GLU [8]. Библиотека GLU позволяет создавать готовые сферы по радиусам и координатам.

Открытые методы класса Canvas3D представлены в таблице 4.9, закрытые методы – в таблице 4.10, открытые поля данных – 4.11, закрытые поля – 4.12.

Таблица 4.9 Открытые методы класса Canvas3D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| Canvas3D | Метод-конструктор | Canvas3D(wxFrame\* parent, wxWindowID id,int\* attributes, const wxPoint&pos, const wxSize& size) | parent | Родительское окно |
| id | Идентифика-тор компонента интерфейса |
| attributes | Массив свойств |
| pos | Положение на окне |
| size | Размер компонента |
| ~Canvas3D | Метод-деструктор | virtual ~ Canvas3D () | - | - |
| Get Transformed | Метод активации моделирования | void getTransformed() | - | - |
| save | Метод вызова сохранения | void save() | - | - |

Таблица 4.10 Закрытые методы класса Canvas 3D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| render | Событие рисования | void render(wxPaint Event& evt) | evt | Инкапсуляция события рисования |
| zoom | События прокрутки колесика (масштабирование) | void zoom(wxMouse Event &evt) | evt | Инкапсуляция состояния мыши |
| sphere | Рисование сферы | void sphere(vertex3D &vertex, int pos) | vertex | Координата |
| pos | Индекс вершины |
| prepare | Подготовка рендеринга | void prepare() | - | - |

Таблица 4.11 Открытые поля данных класса Canvas 3D

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение параметра** |
| vertices3D | std::vector <vertex3D> | Массив трехмерных вершин |
| filePath | wxString | Выбранный путь файла для сохранения |

Таблица 4.12 Закрытые поля данных класса Canvas 3D

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение параметра** |
| reusableQuadrics | std::vector <GLUquadricObj\*> | Массив тел библиотеки GLU |
| m\_context | wxGLContext\* | Графический контекст OpenGL |
| light\_ambient | const GLfloat[4] | Цветовая матрица рассеянного света |
| light\_diffuse | const GLfloat[4] | Цветовая матрица рассеянного света для одной плоскости |
| light\_specular | const GLfloat[4] | Цветовая матрица точечного света |
| light\_position | const GLfloat[4] | Цветовая матрица света |
| mat\_ambient | const GLfloat[4] | Матрица рассеянного света |
| mat\_diffuse | const GLfloat[4] | Матрица рассеянного света для одной плоскости |
| mat\_specular | const GLfloat[4] | Матрица точечного света |
| high\_shininess | const GLfloat[1] | Матрица яркости |
| shouldPaint | bool | Флаг начала моделирования |
| shouldSave | bool | Флаг сохранения изображения |
| dist | float | Изменение оси аппликат |

* 1. Описание класса Converter

Класс Converter реализует шаблон «Адаптер». Его задача – организовать преобразование двумерных вершин в трехмерные. Детальное описание алгоритма приведено в пункте 3.2. Открытые методы класса представлены в таблице 4.13 закрытые методы – в таблице 4.14, закрытые поля данных – 4.15.

Таблица 4.13 Открытые методы класса Canvas3D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| Converter | Метод-конструктор | Converter() | - | - |
| ~Converter | Метод-деструктор | ~Converter() | - | - |
| convertTo Callote | Рисование сферы | std::vector <vertex3D> convertToCallote (std::vector <vertex2D> &vertices2D) | vertices2D | Ссылка на массив двумерных координат |

Таблица 4.14 Закрытые методы класса Canvas3D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя метода** | **Описание** | **Заголовок метода** | **Имя параметра** | **Назначение параметра** |
| tethrahedron | Тетраэдр для простого объекта | void tethrahedron (elem vertex) | vertex | Атом вершины |
| triangle | Рисование треугольной молекулы для простого объекта | void triangle(elem vertex) | vertex | Атом вершины |
| linear | Рисование линейной молекулы для простого объекта | void linear (elem vertex) | vertex | Атом вершины |
| isSimple | Проверка на «простоту» | bool isSimple (std::vector <vertex2D> &vertices2D) | vertices2D | Ссылка на массив двумерных координат |
| spin | Расчет координат простого тетраэдра | Point spin(Point &root, float phi, float theta) | root | Координата центра сферы |
| phi | Угол нормали |
| theta | Угол между осями абсцисс и аппликат. |

Продолжение таблицы 4.14

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Get possible bonds | Определение валентности | char getPossibleBonds(elem activeElement) | activeElement | Химический элемент |
| makeTree | Построение дерева с четырьмя потомками | void makeTree (node\*root, std::vector <vertex2D> &vertices2D) | root | Узел дерева |
| vertices2D | Массив вершин |
| sortTree | Сортировка потомков по возрастанию | void sortTree(node\* root) | root | Узел дерева |
| init Coordinates | Начало расчета координат | void initCoordinates (node\* root) | root | Узел дерева |
| implement Coordinates | Рекурсивный расчет координат | void implement Coordinates (node\* root) | root | Узел дерева |
| clearTree | Очистка дерева | void clearTree (node\*root) | root | Узел дерева |
| centrify Coordinates | Центрирование координат относительно начала координат | void centrify Coordinates() | - | - |
| Advanced Spin | Непосредственный координат для нелинейных связей | void advancedSpin (node\* root) | root | Узел дерева |

Таблица 4.15 Закрытые поля данных класса Canvas 3D

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение параметра** |
| vertices3D | std::vector <vertex3D> | Массив тел библиотеки GLU |

* 1. Описание класса Converter

Основные структуры данных объявляются в заголовочном файле structures.h. Они перечислены в таблице 4.16. Здесь также объявляется значение числа.

Таблица 4.16 Основные структуры данных

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя идентификатора структуры** | **Назначение структуры** | **Тип структуры** | **Поля структуры** |
| elem | Химический элемент | Перечисление | carbon |
| nitrogen |
| phosphorus |
| oxygen |
| sulfur |
| fluorine |
| chlorine |
| bromine |
| iodine |
| hydrogen |
| atom | Атом со связями и именем элемента | Структура | elem atomName |
| char freebonds |
| char bond |
| colour | Цвет в формате RGB  (от 0 до 1.0) | Структура | float red |
| float green |
| float blue |
| vertex2D | Вершина (связь) в скелетной формуле | Структура | short int x |
| short int y |
| short int prevx |
| short int prevy |
| bool shortRear |
| bool shortFront |
| atom element |
| Point | Трехмерная координата | Структура | float x |
| float y |
| float z |
| vertex3D | Трехмерная вершина с указанием элемента | Структура | Point point |
| elem atomName |

Продолжение таблицы 4.16

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя идентификатора структуры** | **Назначение структуры** | **Тип структуры** | **Поля структуры** |
| node | Узел дерева с четырьмя потомками | Структура | struct quaternaryTree \*first |
| struct quaternaryTree \*second |
| struct quaternaryTree \*third |
| struct quaternaryTree \*fourth |
| struct quaternaryTree \*parent |
| vertex3D vertex |
| short int index |
| char lastBond |
| char bond |

Примечание: node определено как struct quaternaryTree с помощью класса хранения typedef.

4.9 Описание модуля с иконками для приложения

Для того, чтобы в приложении были иконки, они добавляются из кода с помощью формата XPM. Расширение .xpm кодируется попиксельно с помощью массива на языке С. Таким образом, иконки можно собрать в один заголовочный файл xpm\_icons.h.

В результате этапа конструирования создано программное средство. Схема программы представлена в приложении А. Листинг программы представлен в приложении В.

1. Тестирование, проверка работоспособности и анализ полученных результатов

В данном разделе я остановлюсь на тестировании программного средства. В таблицах 5.1, 5.2 и 5.3 представлены результаты тестирования программного средства.

* 1. Тестирование алгоритма рисования

Таблица 5.1 – тестирование алгоритма рисования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Тестируемая функциональ-ность** | **Последовательность действий** | **Ожидаемый результат** | **Полученный результат** |
| 1 | Рисование разветвленных углеводород-ных молекул | Рисование одинарными связями. Из одной вершины рисуется несколько других. | Связи рисуются под углом 120 градусов к другим, кроме четвертой (60 градусов) |  |
| 2 | Рисование многоэлемент-ных молекул | Рисование молекулы со сменой активного элемента | Связи у неуглеродных атомов укорочены, имеются обозначения буквенные атомов |  |
| 3 | Рисование тройных связей | Рисование молекулы, смена активной связи на тройную | Две соседние связи стройной расположены на одной прямой с тройной |  |
| 4 | Рисование нескольких двойных связей | Выбор двойной связи для углерода и рисование молекулы | Двойные связи расположены на одной прямой |  |
| 5 | Тестирование молекулы с одной вершиной | Выбор элемента, нажатие ЛКМ по области рисования | Водородное соединение |  |
| 6 | Очистка экрана | Рисование молекулы, нажатие на кнопку «Очистить» | Очистка экрана | Тест пройден |

* 1. Тестирование сохранения изображения

Таблица 5.2 – тестирование сохранения изображения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Тестируемая функциональ-ность** | **Последовательность действий** | **Ожидаемый результат** | **Полученный результат** |
| 1 | Сохранение двумерного изображения в формат PNG | Рисование молекулы, нажатие на кнопку «сохранить» для двумерного изображения. Формат файла –  PNG. | Получение изображения  “Molecule.png”в выбранной папке | Тест пройден |
| 2 | Сохранение двумерного изображения в формат BMP | Рисование молекулы, нажатие на кнопку «сохранить» для двумерного изображения. Формат файла –  BMP. | Получение изображения  “Molecule.bmp”в выбранной папке | Тест пройден |
| 3 | Сохранение двумерного изображения в формат JPEG | Рисование молекулы, нажатие на кнопку «сохранить» для двумерного изображения. Формат файла –  JPEG. | Получение изображения  “Molecule.jpg”в выбранной папке | Тест пройден |
| 4 | Сохранение трехмерного изображения в формат PNG | Рисование молекулы, моделирование, нажатие на кнопку «сохранить» для трехмерного изображения. Формат файла –  PNG. | Получение изображения  “Molecule3D.png”в выбранной папке | Тест пройден |
| 5 | Сохранение трехмерного изображения в формат BMP | Рисование молекулы, моделирование, нажатие на кнопку «сохранить» для трехмерного изображения. Формат файла –  BMP. | Получение изображения  “Molecule3D.bmp”в выбранной папке | Тест пройден |
| 6 | Сохранение трехмерного изображения в формат JPEG | Рисование молекулы, моделирование, нажатие на кнопку «сохранить» для трехмерного изображения. Формат файла –  JPEG. | Получение изображения  “Molecule3D.jpg”в выбранной папке | Тест пройден |

* 1. Тестирование трехмерного моделирования

Таблица 5.3 – тестирование трехмерного моделирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Тестируемая функциональ-ность** | **Последователь-ность действий** | **Ожидаемый результат** | **Полученный результат** |
| 1 | Моделирование молекулы воды | Выбор элемента «кислород», нажатие ЛКМ на область рисования, нажатие на кнопку «Преобразовать». | Треугольная бело-красно-белая молекула |  |
| 2 | Моделирование молекулы фтора | Выбор элемента «фтор», нажатие ЛКМ на область рисования, нажатие на нарисованную вершину, нажатие на кнопку «Преобразовать». | Двойная светло-зеленая молекула |  |
| 3 | Моделирование линейной молекулы на примере сероуглерода | Выбор элемента «сера», нажатие ЛКМ на область рисования, выбор элемента «углерод», кратности связи «двойная», нажатие ЛКМ на область рисования, выбор элемента «сера», нажатие ЛКМ на область рисования, нажатие на кнопку «Преобразовать». | Желтая сфера, серая сфера, желтая сфера расположенные на одной прямой |  |
| 4 | Моделирование линейной молекулы на примере ацетилена | Выбор элемента «углерод», нажатие ЛКМ на область рисования, выбор кратности связи «тройная», нажатие ЛКМ на область рисования, нажатие на кнопку «Преобразовать». | Белая сфера, две серые сферы, белая сфера, расположенные на одной прямой |  |

Продолжение таблицы 5.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Тестируемая функциональ-ность** | **Последователь-ность действий** | **Ожидаемый результат** | **Полученный результат** |
| 5 | Моделирование молекулы с дисульфидным мостиком | Выбор элемента «углерод», нажатие ЛКМ на область рисования, выбор элемента «сера», два нажатия ЛКМ на область рисования, выбор элемента «углерод», нажатие ЛКМ на область рисования, нажатие на кнопку «Преобразовать». | Две серые сферы с тремя белыми маленькими сферами рядом и две желтые сферы между серыми |  |
| 6 | Моделирование молекулы перекиси водорода | Выбор элемента «фосфор», два клика ЛКМ, выбор элемента, нажатие на кнопку «Преобразовать». | Две красные сферы и две белые по краям.  Молекула изогнута |  |
| 7 | Моделирование молекулы хлорэтана | Выбор элемента «углерод», два нажатия ЛКМ на область рисования, выбор элемента «хлор», нажатие ЛКМ по второй вершине, нажатие на кнопку «Преобразовать». | Две серые сферы: у первой три белых сферы рядом, у другой две белых и одна зеленая |  |

Продолжение таблицы 5.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Тестируемая функциональ-ность** | **Последователь-ность действий** | **Ожидаемый результат** | **Полученный результат** |
| 8 | Моделирование молекулы тетрафторида дифосфора | Выбор элемента «фосфор», два клика ЛКМ, выбор элемента «фтор», по два клика ЛКМ по вершинам, нажатие на кнопку «Преобразовать». | Две оранжевые сферы и четыре зеленые |  |
| 9 | Масштабирование изображения | Получение модели, прокрутка колесиком в одном из направлений | Удаление или приближение молекулы | Тест пройден |

Подводя итог, отмечу, что программа отвечает заданным функциональным требованиям, наблюдается стабильность в работе. Вопросов к эстетической части не имеется.

1. Руководство по установке и использованию

Молекулярный редактор обеспечивает рисование скелетных формул молекулярных веществ, а также преобразовывает создание пользователем формулы в трехмерные полусферические модели.

Данное программное средство разработано для использования на операционной системе Windows (рис. 6.1).

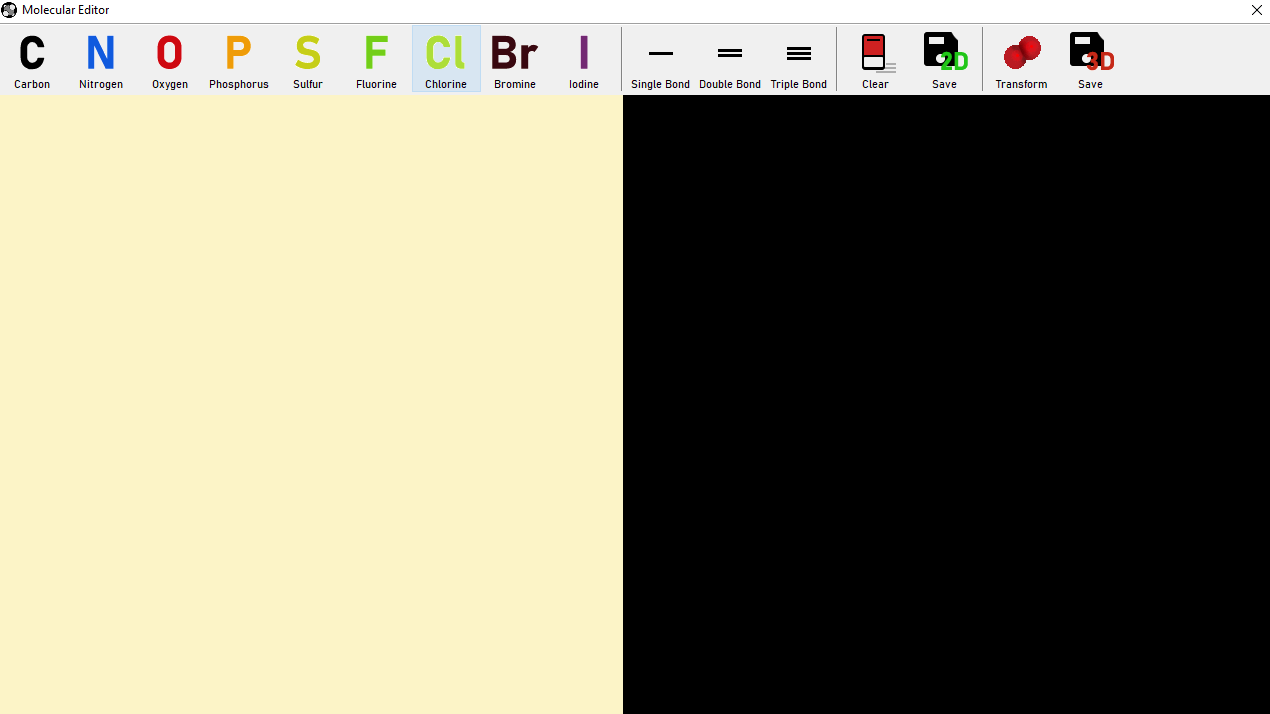


Рисунок 6.1 – Программное средство на устройстве под управлением Windows 10

Для нормальной работы программы папка программного средства должна содержать исполняемый файл Project.exe.

При запуске программы отображается основное окно (рис 6.1). В верхней части окна приложения расположена панель инструментов (рис 6.2).

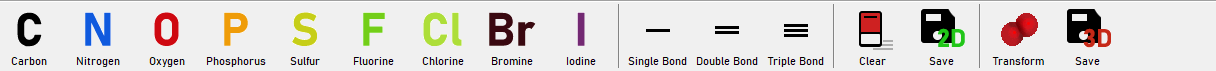


Рисунок 6.2 – панель инструментов

Панель инструментов можно условно разделить на следующие области:

1. Область выбора химического элемента;
2. Область выбора химической связи;
3. Дополнительная область для работы с двумерными изображениями;
4. Область для работы с трехмерным моделированием;

При запуске программы по умолчанию текущим элементом устанавливается «углерод», а текущей кратностью связи – «одинарная связь».

Рисование осуществляется нажатием левой кнопки мыши по области рисования светло-желтого цвета (рис 6.3).

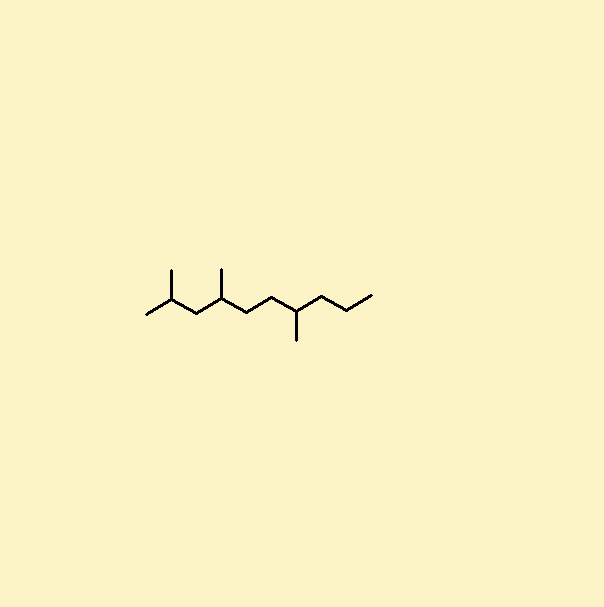


Рисунок 6.3 – область рисования

В начале пользователь нажимает на произвольную область, после чего все остальные вершины строятся относительно первой. Добавление связи к вершине осуществляется нажатием на нее или ее строковое обозначение, если свободные связи имеются.

К предлагаемым пользователю элементам относятся: углерод, азот, кислород, фосфор, сера, фтор, хлор, бром, йод.

Очистка области рисования осуществляется нажатием на кнопку «очистить».

Нарисованную структурную формулу можно преобразовать в трехмерную полусферическую модель, которая отобразится в правой (черной) части окна (рис 6.4). Преобразование осуществляется посредством нажатия на кнопку «Преобразовать».

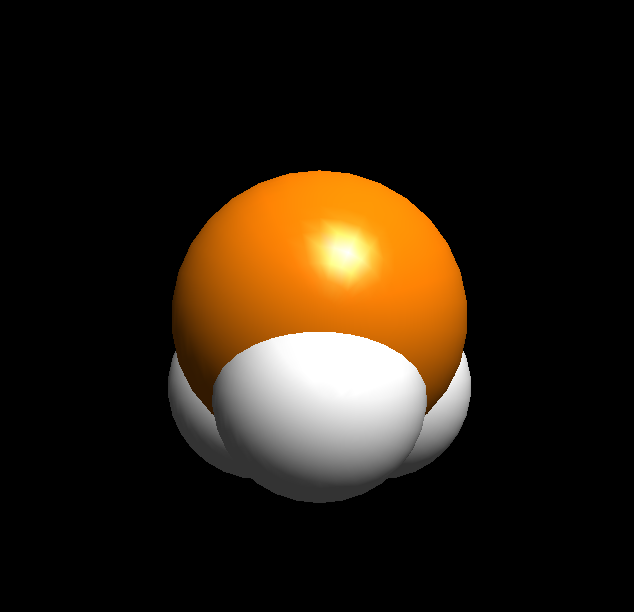


Рисунок 6.4 – область моделирования

Трехмерные модели можно масштабировать с помощью прокрутки колесиком мыши.

Полученные структурные формулы и трехмерные модели можно сохранить в один из представленных графических форматов: PNG, BMP, JPEG. Для этого необходимо нажать на кнопку «Сохранить» для нужной области.

Заключение

В ходе данной работы создано программное средство «Молекулярный редактор», которое предоставляет пользователю возможность рисовать и моделировать молекулы.

При этом в ходе работы мною получен опыт работы с парадигмой ООП, а, точнее, с наследованием библиотечных и созданием собственных классов. Я использовал разделение методов и полей данных на открытые и закрытые, ознакомился с работой конструкторов и деструкторов.

При работе с трехмерными изображениями я ознакомился с работой библиотеки OpenGL и библиотекой утилит GLU.

При создании приложения все компоненты интерфейса прописывались вручную, для чего потребовалось хорошо разобраться в англоязычной документации.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

Список литературы

[1] Wikipedia [Электронный ресурс] – Skeletal Formula – Режим доступа: <https://en.wikipedia.org/wiki/Skeletal_formula>

[2] University of California – Van der Waals radii. – Режим доступа: <https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/midas/vdwrad.html>

[3] Wikipedia [Электронный ресурс] – CPK Coloring – Режим доступа: <https://en.wikipedia.org/wiki/CPK_coloring>

[4] Цветовая палитра JMOL [Электронный ресурс] – Coloring. – Режим доступа: <http://jmol.sourceforge.net/jscolors/>

[5] Редактор JMOL [Электронный ресурс] – JMOL – Режим доступа: <https://sourceforge.net/projects/jmol/>

[6] Редактор PyMol [Электронный ресурс] – PyMol – Режим доступа: <https://pymol.org/2/>

[7] Редактор BKChem [Электронный ресурс] – BKChem – Режим [доступа: http://bkchem.zirael.org/](доступа:%20%20http:/bkchem.zirael.org/)

[8] Официальная документация OpenGL [Электронный ресурс] – OpenGL® 2.1, GLX, and GLU Reference Pages. – Режим доступа: <https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/>

[9] Официальная документация wxWidgets [Электронный ресурс] – Documentation. – Режим доступа: <https://docs.wxwidgets.org/3.0/>

[10] Smart, Julian. Cross-Platform GUI Programming with wxWidgets / Julian Smart, Kevin Hock, Stefan Csomor. – Upper Saddle River, NJ, 2006.

Приложение A

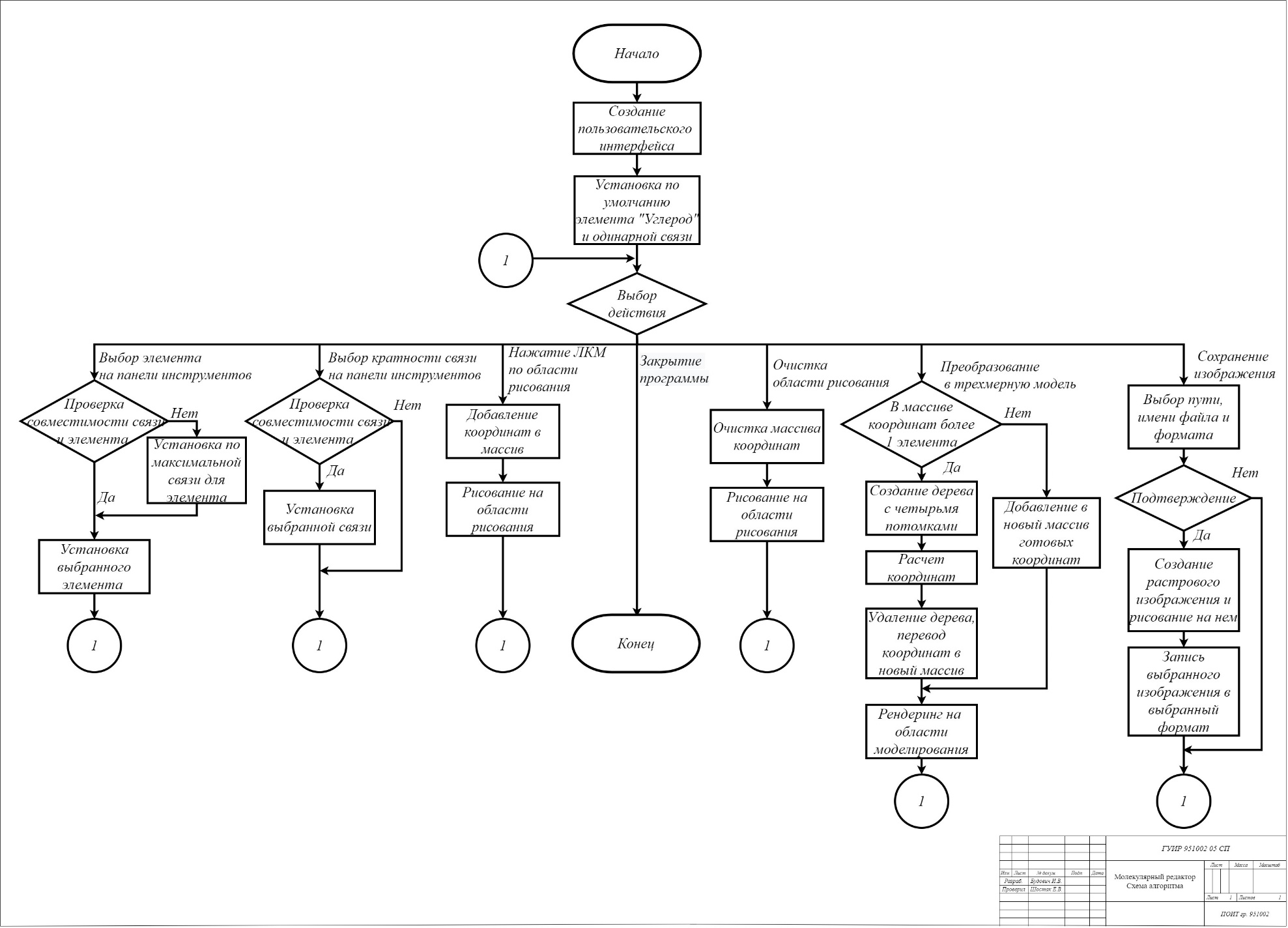


Рисунок А.1 – Схема алгоритма программы

Приложение Б

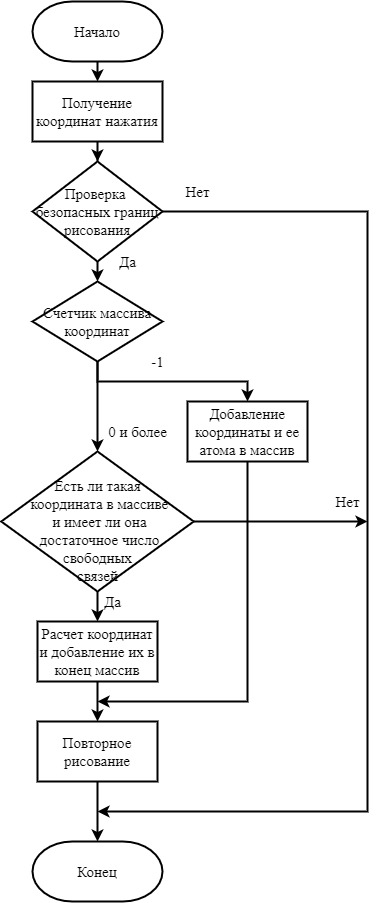


Рисунок Б.1 – Схема алгоритма добавления координат

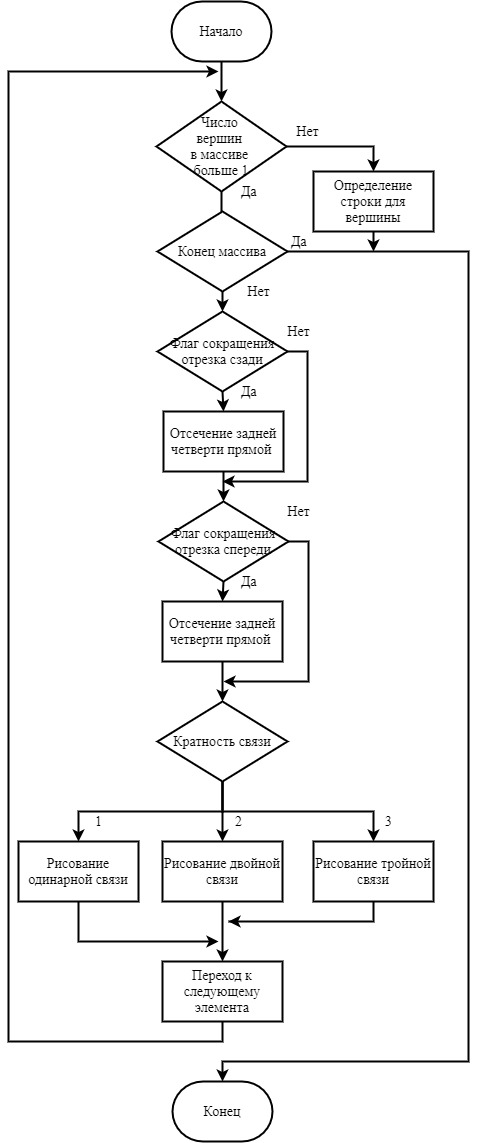


Рисунок Б.2 – Схема алгоритма рисования

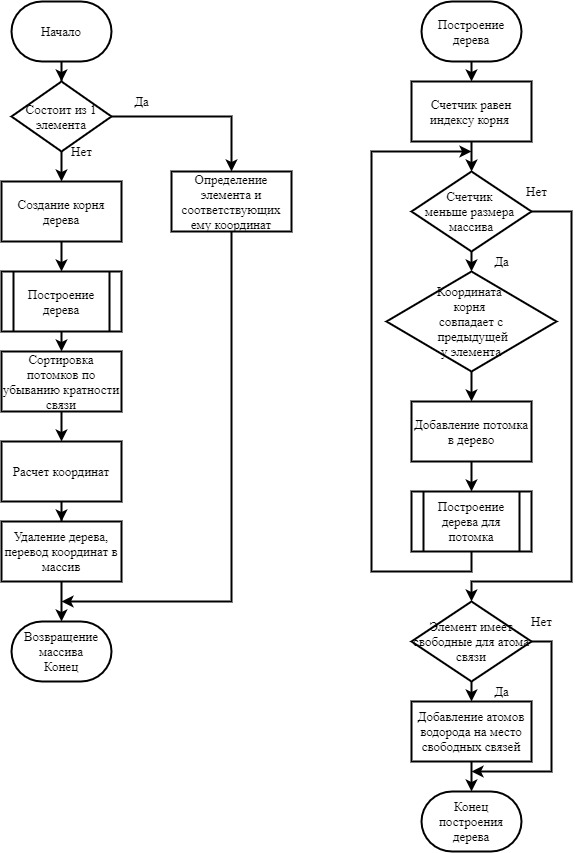


Рисунок Б.3 – Схема алгоритма преобразования в трехмерное представление

Приложение В

Листинг файла Application.h

#ifndef APPLICATION\_H

#define APPLICATION\_H

#include <wx/app.h>

class Application : public wxApp

{

public:

virtual bool OnInit();

};

#endif // APPLICATION\_H

Листинг файла Application.cpp

#include "Application.h"

#include "MainFrame.h"

IMPLEMENT\_APP(Application);

bool Application::OnInit()

{

bool wxsOK = true;

if ( wxsOK )

{

MainFrame\* Frame = new MainFrame(0);

Frame->Show(true);

}

return wxsOK;

}

Листинг файла MainFrame.h

#ifndef MAINFRAME\_H

#define MAINFRAME\_H

#include <wx/file.h>

#include <wx/toolbar.h>

#include <wx/frame.h>

#include <wx/filedlg.h>

#include "Canvas2D.h"

#include "Canvas3D.h"

#include "Converter.h"

class MainFrame : public wxFrame

{

public:

MainFrame(wxWindow\* parent,wxWindowID id = -1);

virtual ~MainFrame();

private:

Canvas2D\* MainCanvas;

Canvas3D\* OtherCanvas;

wxToolBar\* MainToolBar;

wxToolBarToolBase\* ToolBarItem\_Carbon;

wxToolBarToolBase\* ToolBarItem\_Nitrogen;

wxToolBarToolBase\* ToolBarItem\_Oxygen;

wxToolBarToolBase\* ToolBarItem\_Phosphorus;

wxToolBarToolBase\* ToolBarItem\_Sulfur;

wxToolBarToolBase\* ToolBarItem\_Fluorine;

wxToolBarToolBase\* ToolBarItem\_Chlorine;

wxToolBarToolBase\* ToolBarItem\_Bromine;

wxToolBarToolBase\* ToolBarItem\_Iodine;

wxToolBarToolBase\* ToolBarItem\_Single;

wxToolBarToolBase\* ToolBarItem\_Double;

wxToolBarToolBase\* ToolBarItem\_Triple;

wxToolBarToolBase\* ToolBarItem\_Eraser;

wxToolBarToolBase\* ToolBarItem\_Calotte;

wxToolBarToolBase\* ToolBarItem\_Save2D;

wxToolBarToolBase\* ToolBarItem\_Save3D;

static const long ID\_Canvas2D;

static const long ID\_ToolBar;

static const long ID\_ToolBarItem\_Carbon;

static const long ID\_ToolBarItem\_Nitrogen;

static const long ID\_ToolBarItem\_Oxygen;

static const long ID\_ToolBarItem\_Phosphorus;

static const long ID\_ToolBarItem\_Sulfur;

static const long ID\_ToolBarItem\_Fluorine;

static const long ID\_ToolBarItem\_Chlorine;

static const long ID\_ToolBarItem\_Bromine;

static const long ID\_ToolBarItem\_Iodine;

static const long ID\_ToolBarItem\_Single;

static const long ID\_ToolBarItem\_Double;

static const long ID\_ToolBarItem\_Triple;

static const long ID\_ToolBarItem\_Eraser;

static const long ID\_ToolBarItem\_Calotte;

static const long ID\_ToolBarItem\_Save2D;

static const long ID\_ToolBarItem\_Save3D;

static const long ID\_Canvas3D;

void CarbonClick(wxCommandEvent& event);

void NitrogenClick(wxCommandEvent& event);

void OxygenClick(wxCommandEvent& event);

void PhosphorusClick(wxCommandEvent& event);

void SulfurClick(wxCommandEvent& event);

void FluorineClick(wxCommandEvent& event);

void ChlorineClick(wxCommandEvent& event);

void BromineClick(wxCommandEvent& event);

void IodineClick(wxCommandEvent& event);

void SingleClick(wxCommandEvent& event);

void DoubleClick(wxCommandEvent& event);

void TripleClick(wxCommandEvent& event);

void EraseClick(wxCommandEvent& event);

void CalotteClick(wxCommandEvent& event);

void Save2DClick(wxCommandEvent&event);

void Save3DClick(wxCommandEvent&event);

DECLARE\_EVENT\_TABLE()

};

#endif // MAINFRAME\_H

Листинг файла MainFrame.cpp

#include "MainFrame.h"

#include "Canvas2D.h"

#include "xpm\_icons.h"

const long MainFrame::ID\_Canvas2D= wxNewId();

const long MainFrame::ID\_ToolBar= wxNewId();

const long MainFrame::ID\_ToolBarItem\_Carbon = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Nitrogen = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Oxygen = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Phosphorus = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Sulfur = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Fluorine = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Chlorine = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Bromine = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Iodine = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Single = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Double = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Triple = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Eraser = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Calotte = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Save2D = wxNewId();

const long MainFrame::ID\_ToolBarItem\_Save3D = wxNewId();

const long MainFrame::ID\_Canvas3D = wxNewId();

MainFrame::MainFrame(wxWindow\* parent,wxWindowID id)

{

Create(parent, id, \_("Molecular Editor"), wxDefaultPosition, wxSize(1280,720), (wxSYSTEM\_MENU |wxCLOSE\_BOX |wxCAPTION |wxCLIP\_CHILDREN), \_T("id"));

SetForegroundColour(wxColour(0,0,0,1));

SetBackgroundColour(wxColour(128, 128, 128, 1));

SetIcon(wxIcon(ChemEditor));

wxFont thisFont(8,wxFONTFAMILY\_SWISS,wxFONTSTYLE\_NORMAL,wxFONTWEIGHT\_NORMAL,false,\_T("Bahnschrift"),wxFONTENCODING\_DEFAULT);

SetFont(thisFont);

MainCanvas = new Canvas2D(this, ID\_Canvas2D,wxPoint(0, 72), wxSize(625,648),wxTAB\_TRAVERSAL,\_T("ID\_Canvas2D"));

MainCanvas->SetForegroundColour(wxSystemSettings::GetColour(wxSYS\_COLOUR\_HIGHLIGHT));

MainCanvas->SetBackgroundColour(wxColor(252,244,199,1));

int attributes[] = {WX\_GL\_RGBA, WX\_GL\_DOUBLEBUFFER, WX\_GL\_DEPTH\_SIZE, 16, 0};

OtherCanvas = new Canvas3D((wxFrame\*)this, ID\_Canvas3D, attributes, wxPoint(625, 72), wxSize(655, 648));

MainToolBar = new wxToolBar(this, ID\_ToolBar, wxDefaultPosition, wxDefaultSize, wxTB\_HORIZONTAL|wxTB\_TEXT|wxTB\_FLAT, \_T("ID\_ToolBar"));

ToolBarItem\_Carbon = MainToolBar ->AddTool(ID\_ToolBarItem\_Carbon, \_("Carbon"), wxBitmap(carbon\_xpm), wxBitmap(carbon\_xpm), wxITEM\_NORMAL, \_("Carbon"), \_("Carbon"));

ToolBarItem\_Nitrogen = MainToolBar ->AddTool(ID\_ToolBarItem\_Nitrogen, \_("Nitrogen"), wxBitmap(nitrogen\_xpm), wxBitmap(nitrogen\_xpm), wxITEM\_NORMAL,\_("Nitrogen"), \_("Nitrogen"));

ToolBarItem\_Oxygen = MainToolBar ->AddTool(ID\_ToolBarItem\_Oxygen, \_("Oxygen"), wxBitmap(oxygen\_xpm), wxBitmap(oxygen\_xpm), wxITEM\_NORMAL, \_("Oxygen"), \_("Oxygen"));

ToolBarItem\_Phosphorus = MainToolBar ->AddTool(ID\_ToolBarItem\_Phosphorus, \_("Phosphorus"),wxBitmap(phosphorus\_xpm), wxBitmap(phosphorus\_xpm), wxITEM\_NORMAL, \_("Phosphorus"), \_("Phosphorus"));

ToolBarItem\_Sulfur = MainToolBar ->AddTool(ID\_ToolBarItem\_Sulfur, \_("Sulfur"), wxBitmap(sulfur\_xpm), wxBitmap(sulfur\_xpm), wxITEM\_NORMAL, \_("Sulfur"), \_("Sulfur"));

ToolBarItem\_Fluorine = MainToolBar ->AddTool(ID\_ToolBarItem\_Fluorine, \_("Fluorine"), wxBitmap(fluorine\_xpm), wxBitmap(fluorine\_xpm), wxITEM\_NORMAL, \_("Fluorine"), \_("Fluorine"));

ToolBarItem\_Chlorine = MainToolBar ->AddTool(ID\_ToolBarItem\_Chlorine, \_("Chlorine"), wxBitmap(chlorine\_xpm), wxBitmap(chlorine\_xpm), wxITEM\_NORMAL,\_("Chlorine"), \_("Chlorine"));

ToolBarItem\_Bromine = MainToolBar ->AddTool(ID\_ToolBarItem\_Bromine, \_("Bromine"), wxBitmap(bromine\_xpm), wxBitmap(bromine\_xpm), wxITEM\_NORMAL,\_("Bromine"), \_("Bromine"));

ToolBarItem\_Iodine = MainToolBar ->AddTool(ID\_ToolBarItem\_Iodine, \_("Iodine"), wxBitmap(iodine\_xpm), wxBitmap(iodine\_xpm), wxITEM\_NORMAL, \_("Iodine"), \_("Iodine"));

MainToolBar->AddSeparator();

ToolBarItem\_Single = MainToolBar ->AddTool(ID\_ToolBarItem\_Single, \_("Single Bond"), wxBitmap(single\_xpm), wxBitmap(single\_xpm), wxITEM\_NORMAL, \_("Single Bond"), \_("Single Bond"));

ToolBarItem\_Double = MainToolBar ->AddTool(ID\_ToolBarItem\_Double, \_("Double Bond"), wxBitmap(double\_xpm), wxBitmap(double\_xpm), wxITEM\_NORMAL, \_("Double Bond"), \_("Double Bond"));

ToolBarItem\_Triple = MainToolBar ->AddTool(ID\_ToolBarItem\_Triple, \_("Triple Bond"), wxBitmap(triple\_xpm), wxBitmap(triple\_xpm), wxITEM\_NORMAL, \_("Triple Bond"), \_("Triple Bond"));

MainToolBar->AddSeparator();

ToolBarItem\_Eraser = MainToolBar ->AddTool(ID\_ToolBarItem\_Eraser, \_("Clear"), wxBitmap(eraser\_xpm), wxBitmap(eraser\_xpm), wxITEM\_NORMAL, \_("Clear"), \_("Clear"));

ToolBarItem\_Save2D = MainToolBar ->AddTool(ID\_ToolBarItem\_Save2D, \_("Save"), wxBitmap(save2d\_xpm), wxBitmap(save2d\_xpm),wxITEM\_NORMAL, \_("Save"), \_("Save"));

MainToolBar->AddSeparator();

ToolBarItem\_Calotte = MainToolBar ->AddTool(ID\_ToolBarItem\_Calotte, \_("Transform"), wxBitmap(calotte\_xpm),wxBitmap(calotte\_xpm), wxITEM\_NORMAL, \_("Transform"), \_("Transform"));

ToolBarItem\_Save3D = MainToolBar ->AddTool(ID\_ToolBarItem\_Save3D, \_("Save"), wxBitmap(save3d\_xpm), wxBitmap(save3d\_xpm),wxITEM\_NORMAL, \_("Save"), \_("Save"));

MainToolBar->Realize();

SetToolBar(MainToolBar);

MainCanvas->Bind(wxEVT\_LEFT\_DOWN, &Canvas2D::leftClick, MainCanvas);

MainCanvas->Bind(wxEVT\_PAINT, &Canvas2D::paint, MainCanvas);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::CarbonClick, this, ID\_ToolBarItem\_Carbon);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::NitrogenClick, this, ID\_ToolBarItem\_Nitrogen);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::OxygenClick, this, ID\_ToolBarItem\_Oxygen);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::PhosphorusClick, this, ID\_ToolBarItem\_Phosphorus);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::SulfurClick, this, ID\_ToolBarItem\_Sulfur);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::FluorineClick, this, ID\_ToolBarItem\_Fluorine);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::ChlorineClick, this, ID\_ToolBarItem\_Chlorine);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::BromineClick, this, ID\_ToolBarItem\_Bromine);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::IodineClick, this, ID\_ToolBarItem\_Iodine);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::SingleClick, this, ID\_ToolBarItem\_Single);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::DoubleClick, this, ID\_ToolBarItem\_Double);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::TripleClick, this, ID\_ToolBarItem\_Triple);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::EraseClick, this, ID\_ToolBarItem\_Eraser);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::CalotteClick, this, ID\_ToolBarItem\_Calotte);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::Save2DClick, this, ID\_ToolBarItem\_Save2D);

Bind(wxEVT\_COMMAND\_TOOL\_CLICKED, &MainFrame::Save3DClick, this, ID\_ToolBarItem\_Save3D);

}

BEGIN\_EVENT\_TABLE(Canvas3D, wxGLCanvas)

EVT\_PAINT(Canvas3D::render)

EVT\_MOUSEWHEEL(Canvas3D::zoom)

END\_EVENT\_TABLE()

void MainFrame::CarbonClick(wxCommandEvent&event)

{

MainCanvas->setActiveElement(carbon);

}

void MainFrame::NitrogenClick(wxCommandEvent&event)

{

MainCanvas->setActiveElement(nitrogen);

}

void MainFrame::OxygenClick(wxCommandEvent&event)

{

MainCanvas->setActiveElement(oxygen);

}

void MainFrame::PhosphorusClick(wxCommandEvent&event)

{

MainCanvas->setActiveElement(phosphorus);

}

void MainFrame::SulfurClick(wxCommandEvent&event)

{

MainCanvas->setActiveElement(sulfur);

}

void MainFrame::FluorineClick(wxCommandEvent&event)

{

MainCanvas->setActiveElement(fluorine);

}

void MainFrame::ChlorineClick(wxCommandEvent&event)

{

MainCanvas->setActiveElement(chlorine);

}

void MainFrame::BromineClick(wxCommandEvent&event)

{

MainCanvas->setActiveElement(bromine);

}

void MainFrame::IodineClick(wxCommandEvent&event)

{

MainCanvas->setActiveElement(iodine);

}

void MainFrame::SingleClick(wxCommandEvent&event)

{

MainCanvas->setBond(1);

}

void MainFrame::DoubleClick(wxCommandEvent&event)

{

MainCanvas->setBond(2);

}

void MainFrame::TripleClick(wxCommandEvent&event)

{

MainCanvas->setBond(3);

}

void MainFrame::EraseClick(wxCommandEvent&event)

{

MainCanvas->clearVertex();

}

void MainFrame::CalotteClick(wxCommandEvent&event)

{

Converter\* converter = new Converter();

OtherCanvas->vertices3D = converter->convertToCalotte(MainCanvas->vertices);

delete converter;

OtherCanvas->getTransformed();

}

void MainFrame::Save2DClick(wxCommandEvent&event)

{

wxString caption = \_T("Choose a directory and filename");

wxString wildcard =\_T("BMP files (\*.bmp)|\*.bmp| PNG files (\*.png)|\*.png| JPG files (\*.jpg)|\*.jpg");

wxString defaultDir = wxEmptyString;

wxString defaultFilename = \_T("Molecule.bmp");

wxFileDialog dialog(this,caption, defaultDir, defaultFilename, wildcard, wxFD\_SAVE);

if (dialog.ShowModal() == wxID\_OK)

{

wxString path = dialog.GetPath();

wxString ext=path.AfterLast('.');

wxImage image = MainCanvas->getBitmap().ConvertToImage();

wxInitAllImageHandlers();

if (ext == "bmp"||ext == "png"||ext == "jpg")

image.SaveFile(path);

}

}

void MainFrame::Save3DClick(wxCommandEvent&event)

{

wxString caption = \_T("Choose a directory and filename");

wxString wildcard =\_T("BMP files (\*.bmp)|\*.bmp| PNG files (\*.png)|\*.png| JPG files (\*.jpg)|\*.jpg");

wxString defaultDir = wxEmptyString;

wxString defaultFilename = \_T("Molecule 3D.bmp");

wxFileDialog \*dialog = new wxFileDialog(this,caption, defaultDir, defaultFilename, wildcard, wxFD\_SAVE);

if (dialog->ShowModal() == wxID\_OK)

{

OtherCanvas->filePath = dialog->GetPath();

delete dialog;

OtherCanvas->save();

}

else

{

delete dialog;

}

}

BEGIN\_EVENT\_TABLE(MainFrame,wxFrame)

END\_EVENT\_TABLE()

MainFrame::~MainFrame()

{

}

Листинг файла Canvas2D.h

#ifndef CANVAS2D\_H

#define CANVAS2D\_H

#include <vector>

#include <string>

#include "wx/dc.h"

#include "wx/dcclient.h"

#include "wx/dcmemory.h"

#include "wx/panel.h"

#include "wx/frame.h"

#include "structures.h"

class Canvas2D : public wxPanel

{

public:

Canvas2D(wxFrame\* parent, wxWindowID id=wxID\_ANY, const wxPoint &pos=wxDefaultPosition,

const wxSize &size = wxSize(625,648), long style=wxTAB\_TRAVERSAL, const wxString &name=wxPanelNameStr);

virtual ~Canvas2D();

void paint(wxPaintEvent &event);

void leftClick(wxMouseEvent& event);

void clearVertex();

void setBond(char bondNumber);

void setActiveElement(elem activeElement);

wxBitmap getBitmap();

std::vector<vertex2D> vertices;

private:

void render(wxDC& dc);

void getVertexLocation(short int x0,short int y0,short int x,short int y,short int value,short int &x1,short int &y1);

std::string getStringForVertex(atom active);

bool isWithinSafeZone(short int x0, short int y0);

char getPossibleBonds(elem activeElement);

short int counter;

const char length = 30;

char bond;

elem active;

};

#endif // CANVAS2D\_H

Листинг файла Canvas2D.cpp

#include "Canvas2D.h"

Canvas2D::Canvas2D(wxFrame\* parent, wxWindowID id, const wxPoint &pos,

const wxSize &size, long style, const wxString &name):

wxPanel(parent, id, pos, size, style, name)

{

counter = -1;

bond = 1;

active = carbon;

}

Canvas2D::~Canvas2D()

{

}

bool Canvas2D::isWithinSafeZone(short int x0, short int y0)

{

wxSize size = wxWindow::GetClientSize();

int x = size.GetWidth();

int y = size.GetHeight();

return ((x0>50) && (x0<x -50) && (y0>50) && (y0<y-50));

}

void Canvas2D::paint(wxPaintEvent &event)

{

wxPaintDC dc(this);

dc.Clear();

if (counter != -1)

render(dc);

}

wxBitmap Canvas2D::getBitmap()

{

if (counter ==1)

return wxNullBitmap;

wxMemoryDC memDC;

wxBitmap bitmap(600, 600);

memDC.SelectObject(bitmap);

memDC.SetBrush(wxBrush(wxColor(255, 255, 255, 1),wxBRUSHSTYLE\_SOLID));

memDC.DrawRectangle(0,0,600, 600);

render(memDC);

memDC.SelectObject(wxNullBitmap);

return bitmap;

}

void Canvas2D::leftClick(wxMouseEvent& event)

{

vertex2D current;

short int x = event.GetX();

short int y = event.GetY();

if (isWithinSafeZone(x,y))

{

switch (counter)

{

case -1:

counter++;

current.x = x;

current.y = y;

current.prevx = x+length\*cos(pi/6);

current.prevy = y-length\*sin(pi/6);

current.shortFront = (active != carbon);

current.shortRear = false;

current.element.atomName = active;

current.element.freebonds = getPossibleBonds(current.element.atomName);

current.element.bond = 0;

vertices.push\_back(current);

break;

case 0:

if ((vertices[0].x - 10 < x) &&(vertices[0].x + 10 > x) && (vertices[0].y - 10 < y) &&(vertices[0].y + 10 > y)&& (vertices[0].element.freebonds - bond >=0))

{

counter++;

current.x = vertices[0].prevx;

current.y = vertices[0].prevy;

current.prevx = vertices[0].x;

current.prevy = vertices[0].y;

current.element.atomName = active;

vertices[0].element.freebonds-=bond;

vertices[0].element.bond = bond;

current.element.freebonds = getPossibleBonds(active)-bond;

current.element.bond = bond;

current.shortFront = (active != carbon);

current.shortRear = vertices[0].shortFront;

vertices.push\_back(current);

}

break;

default:

bool flag = false;

int i;

for (i = 0; i<=counter; i++)

{

if ((vertices[i].x - 10 < x) &&(vertices[i].x + 10 > x) && (vertices[i].y - 10 < y) &&(vertices[i].y + 10 > y) && (vertices[i].element.freebonds - bond >=0))

{

flag = true;

break;

}

}

if (flag)

{

short int x1;

short int y1;

short int exodus;

if (bond == 3 || vertices[i].element.bond == 3|| (bond==2 && vertices[i].element.bond == 2))

exodus = 3;

else

{

if (vertices[i].element.bond == 2)

exodus = 2-vertices[i].element.freebonds;

else

exodus = (getPossibleBonds(vertices[i].element.atomName)-vertices[i].element.freebonds-1);

}

getVertexLocation(vertices[i].prevx,vertices[i].prevy, vertices[i].x, vertices[i].y, exodus, x1, y1);

counter++;

current.x = x1;

current.y = y1;

current.prevx = vertices[i].x;

current.prevy = vertices[i].y;

current.element.atomName = active;

vertices[i].element.freebonds -=bond;

current.element.bond = bond;

current.element.freebonds = getPossibleBonds(active)-bond;

current.shortFront = (active != carbon);

current.shortRear = vertices[i].shortFront;

vertices.push\_back(current);

}

}

Refresh();

}

}

void Canvas2D::render(wxDC& dc)

{

dc.SetPen( wxPen( wxColor(0,0,0,1), 3));

dc.SetTextForeground(wxColor(0,0,0,1));

dc.SetFont(wxFont(15,wxFONTFAMILY\_SWISS,wxFONTSTYLE\_NORMAL,wxFONTWEIGHT\_NORMAL,false,\_T("Arial"),wxFONTENCODING\_DEFAULT));

std::string str;

if (counter == 0)

{

str = getStringForVertex(vertices[0].element);

dc.DrawText(wxString(str), vertices[0].x, vertices[0].y-10);

}

else

{

if (vertices[0].element.atomName != carbon)

{

str = getStringForVertex(vertices[0].element);

dc.DrawText(wxString(str), vertices.at(0).x-10, vertices.at(0).y-10);

}

int x0, x1, y0, y1;

for (int i = 1; i<=counter; i++)

{

if (vertices[i].element.atomName != carbon)

{

str = getStringForVertex(vertices[i].element);

dc.DrawText(wxString(str), vertices[i].x-5, vertices[i].y-10);

}

if (vertices[i].shortRear)

{

x0 = round(vertices[i].x/4.0 + vertices[i].prevx\*3/4.0);

y0 = round(vertices[i].y/4.0 + vertices[i].prevy\*3/4.0);

}

else

{

x0 = vertices[i].prevx;

y0 = vertices[i].prevy;

}

if (vertices[i].shortFront)

{

x1 = round(vertices[i].x/4.0\*3 + vertices[i].prevx/4.0);

y1 = round(vertices[i].y/4.0\*3 + vertices[i].prevy/4.0);

}

else

{

x1 = vertices[i].x;

y1 = vertices[i].y;

}

switch (vertices[i].element.bond)

{

case 2:

if(vertices[i].prevx == vertices[i].x)

{

dc.DrawLine(x0 - 2, y0, x1 -2, y1);

dc.DrawLine(x0 + 2, y0, x1 +2, y1);

}

else

{

dc.DrawLine(x0, y0-3, x1, y1-3);

dc.DrawLine(x0, y0+3, x1, y1+3);

}

break;

case 3:

if(vertices[i].prevx == vertices[i].x)

{

dc.DrawLine(x0 - 5, y0, x1 -5, y1);

dc.DrawLine(x0 + 5, y0, x1 +5, y1);

}

else

{

dc.DrawLine(x0, y0-5, x1, y1-5);

dc.DrawLine(x0, y0+5, x1, y1+5);

}

case 1:

dc.DrawLine(x0, y0, x1, y1);

break;

}

}

}

}

void Canvas2D::clearVertex()

{

counter = -1;

vertices.clear();

Refresh();

}

char Canvas2D::getPossibleBonds(elem activeElement)

{

switch (activeElement)

{

case carbon:

return 4;

case nitrogen:

case phosphorus:

return 3;

case oxygen:

case sulfur:

return 2;

default:

return 1;

}

}

void Canvas2D::getVertexLocation(short int x0,short int y0,short int x,short int y,short int value,short int &x1,short int &y1)

{

float alpha, c, s;

int rx, ry;

switch (value)

{

case 0:

alpha = 2\*pi/3;

if (y0>=y)

alpha = -alpha;

break;

case 1:

alpha = 2\*pi/3;

if (y0<y)

alpha = -alpha;

break;

case 2:

alpha = pi/3;

if (y0>=y)

alpha = -alpha;

break;

case 3:

alpha = pi;

break;

default:

alpha = 2\*pi/3;

}

rx = x0 - x;

ry = y0 - y;

c = cos(alpha);

s = sin(alpha);

x1 = round( x + rx \* c - ry \* s);

y1 = round(y + rx \* s + ry \* c);

}

std::string Canvas2D::getStringForVertex(atom activeAtom)

{

std::string vertex = "";

if (activeAtom.freebonds >0)

{

vertex = "H";

if (activeAtom.freebonds >1)

vertex += std::to\_string(activeAtom.freebonds);

}

switch (activeAtom.atomName)

{

case chlorine:

vertex += "Cl";

break;

case bromine:

vertex += "Br";

break;

case fluorine:

vertex += " F";

break;

case iodine:

vertex += " I";

break;

case sulfur:

vertex += "S";

break;

case oxygen:

if (activeAtom.freebonds != 1)

vertex += "O";

else

vertex = "O" + vertex;

break;

case phosphorus:

vertex = "P" + vertex;

break;

case nitrogen:

vertex = "N" + vertex;

break;

case carbon:

vertex = "C" + vertex;

break;

}

return vertex;

}

void Canvas2D::setBond(char bondNumber)

{

if (getPossibleBonds(active)>=bondNumber)

bond = bondNumber;

}

void Canvas2D::setActiveElement(elem activeElement)

{

active = activeElement;

if (bond > getPossibleBonds(activeElement))

bond = 1;

}

Листинг файла Canvas3D.h

#ifndef CANVAS3D\_H

#define CANVAS3D\_H

#include <wx/glcanvas.h>

#include <wx/dcclient.h>

#include <wx/dcmemory.h>

#include <GL/gl.h>

#include <GL/glu.h>

#include <vector>

#include "structures.h"

class Canvas3D : public wxGLCanvas

{

public:

Canvas3D(wxFrame\* parent, wxWindowID id,int\* attributes, const wxPoint&pos, const wxSize& size);

virtual ~Canvas3D();

void getTransformed();

void save();

wxString filePath;

std::vector<vertex3D> vertices3D;

private:

void render(wxPaintEvent& evt);

void zoom(wxMouseEvent &evt);

void sphere(vertex3D &vertex, int pos);

void prepare();

std::vector<GLUquadricObj\*> reusableQuadrics;

wxGLContext\* m\_context;

const GLfloat light\_ambient[4] = { 0.0f, 0.0f, 0.0f, 1.0f };

const GLfloat light\_diffuse[4] = { 1.0f, 1.0f, 1.0f, 1.0f };

const GLfloat light\_specular[4] = { 1.0f, 1.0f, 1.0f, 1.0f };

const GLfloat light\_position[4] = { 2.0f, 5.0f, 5.0f, 0.0f };

const GLfloat mat\_ambient[4] = { 0.7f, 0.7f, 0.7f, 1.0f };

const GLfloat mat\_diffuse[4] = { 0.8f, 0.8f, 0.8f, 1.0f };

const GLfloat mat\_specular[4] = { 1.0f, 1.0f, 1.0f, 1.0f };

const GLfloat high\_shininess[1] = { 100.0f };

bool shouldPaint;

bool shouldSave;

float dist;

DECLARE\_EVENT\_TABLE()

};

#endif // CANVAS3D\_H

Листинг файла Canvas3D.cpp

#include "Canvas3D.h"

Canvas3D::Canvas3D(wxFrame\* parent, wxWindowID id,int\* attributes, const wxPoint&pos, const wxSize& size):

wxGLCanvas((wxWindow\*)parent, id, attributes, pos, size, 0, \_T("GLCanvas"), wxNullPalette)

{

m\_context = new wxGLContext(this);

SetBackgroundStyle(wxBG\_STYLE\_COLOUR);

SetBackgroundColour(wxColor(0,0, 0, 1));

shouldPaint = false;

shouldSave = false;

dist = 7.0;

}

Canvas3D::~Canvas3D()

{

delete m\_context;

int j = reusableQuadrics.size();

for ( int i = 0; i < j; i++ )

{

gluDeleteQuadric(reusableQuadrics[i]);

}

}

void Canvas3D::zoom(wxMouseEvent &evt)

{

int lines = evt.GetColumnsPerAction();

int direction = ((evt.GetWheelRotation() > 0)? -1: 1) ;

dist +=direction\*lines\*0.2;

Refresh();

}

void Canvas3D::sphere(vertex3D &vertex, int pos)

{

colour currentColour;

float radius;

switch (vertex.atomName)

{

case carbon:

currentColour = {0.57, 0.57, 0.57};

radius = 1.12;

break;

case oxygen:

currentColour = {0.96, 0.07, 0.01};

radius = 1;

break;

case nitrogen:

currentColour = {0.2, 0.33, 0.9255};

radius = 1.02;

break;

case phosphorus:

currentColour = {0.98, 0.5, 0.02};

radius = 1.18;

break;

case sulfur:

currentColour = {1, 0.98, 0.267};

radius = 1.18;

break;

case chlorine:

currentColour = {0.094, 0.964, 0.09};

radius = 1.15;

break;

case fluorine:

currentColour = {0.54, 0.89, 0.325};

radius = 0.97;

break;

case bromine:

currentColour = {0.65, 0.16, 0.15};

radius = 1.22;

break;

case iodine:

currentColour = {0.58, 0, 0.58};

radius = 1.3;

break;

default:

currentColour = {1,1,1};

radius = 0.79;

break;

}

glPushMatrix();

glColor3f(currentColour.red, currentColour.green, currentColour.blue);

glTranslatef(vertex.point.x, vertex.point.y, vertex.point.z-dist);

gluSphere(reusableQuadrics[pos], radius, 30, 30);

glPopMatrix();

}

void Canvas3D::getTransformed()

{

shouldPaint = true;

Refresh();

}

void Canvas3D::prepare()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black Background

glClearDepth(1.0f); // Depth Buffer Setup

glEnable(GL\_DEPTH\_TEST); // Enables Depth Testing

glDepthFunc(GL\_LEQUAL); // The Type Of Depth Testing To Do

glHint(GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_NICEST);

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_CULL\_FACE);

glCullFace(GL\_BACK);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

glEnable(GL\_LIGHT0);

glEnable(GL\_NORMALIZE);

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, high\_shininess);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

float ratio\_w\_h = 1;

gluPerspective(45 /\*view angle\*/, ratio\_w\_h, 0.1 /\*clip close\*/, 200 /\*clip far\*/);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void Canvas3D::render(wxPaintEvent& evt)

{

if(!IsShown())

return;

if (!shouldPaint)

return;

wxGLCanvas::SetCurrent(\*m\_context);

wxPaintDC dc(this);

prepare();

while (reusableQuadrics.size()<vertices3D.size())

{

GLUquadricObj\* quad = gluNewQuadric();

reusableQuadrics.push\_back(quad);

}

for (unsigned int i =0; i<vertices3D.size(); i++)

sphere(vertices3D.at(i), i);

glFlush();

SwapBuffers();

if (shouldSave)

{

wxBitmap bitmap(625, 620);

wxMemoryDC dcDest;

dcDest.SelectObject(bitmap);

dcDest.Blit(0, 0,625, 620,&dc,0, 0,wxCOPY);

dcDest.SelectObject(wxNullBitmap);

wxImage image = bitmap.ConvertToImage();

wxInitAllImageHandlers();

wxString ext=filePath.AfterLast('.');

if (ext == "bmp"||ext == "png"||ext == "jpg")

image.SaveFile(filePath);

shouldSave = false;

}

}

void Canvas3D::save()

{

shouldSave = true;

Refresh();

}

Листинг файла Converter.h

#ifndef CONVERTER\_H

#define CONVERTER\_H

#include "structures.h"

#include <vector>

#include <math.h>

#include <stdlib.h>

class Converter

{

public:

Converter();

~Converter();

std::vector<vertex3D> convertToCalotte(std::vector<vertex2D> &vertices2D);

private:

void tethrahedron(elem vertex);

void triangle(elem vertex);

void linear(elem vertex);

bool isSimple(std::vector<vertex2D> &vertices2D);

Point spin(Point &root, float phi, float theta);

char getPossibleBonds(elem activeElement);

void makeTree(node\*root, std::vector<vertex2D> &vertices2D);

void sortTree(node\*root);

void initCoordinates(node\* root);

void implementCoordinates(node\* root);

void clearTree(node\*root);

void centrifyCoordinates();

void advancedSpin(node\* root);

std::vector<vertex3D> vertices3D;

};

#endif // CONVERTER\_H

Листинг файла Converter.cpp

#include "Converter.h"

Converter::Converter()

{

}

Converter::~Converter()

{

}

std::vector<vertex3D> Converter::convertToCalotte(std::vector<vertex2D> &vertices2D)

{

vertices3D.clear();

if (isSimple(vertices2D))

return vertices3D;

node \*root = (node\*) malloc(sizeof(node));

root->vertex.atomName = vertices2D.at(0).element.atomName;

root->first = nullptr;

root->second = nullptr;

root->third = nullptr;

root->fourth = nullptr;

root->parent = nullptr;

root->bond = 0;

root->index = 0;

root->lastBond = 0;

makeTree(root, vertices2D);

sortTree(root);

initCoordinates(root);

clearTree(root);

centrifyCoordinates();

return vertices3D;

}

void Converter::makeTree(node\*root, std::vector<vertex2D> &vertices2D)

{

for (unsigned k=root->index+1; k<vertices2D.size(); k++)

{

if ((vertices2D.at(root->index).x == vertices2D.at(k).prevx) && (vertices2D.at(root->index).y == vertices2D.at(k).prevy))

{

node \*daughter = (node\*) malloc(sizeof(node));

daughter->parent = root;

daughter->first = nullptr;

daughter->second = nullptr;

daughter->third = nullptr;

daughter->fourth = nullptr;

daughter->index = k;

daughter->vertex.atomName = vertices2D.at(k).element.atomName;

daughter->bond = vertices2D.at(k).element.bond;

daughter->lastBond = 0;

switch (root->lastBond)

{

case 0:

root->first = daughter;

break;

case 1:

root->second = daughter;

break;

case 2:

root->third = daughter;

break;

default:

root->fourth = daughter;

break;

}

root->lastBond++;

if (getPossibleBonds(daughter->vertex.atomName)-daughter->bond > 0)

makeTree(daughter, vertices2D);

}

}

if (vertices2D.at(root->index).element.freebonds)

for (int j = 0; j<vertices2D.at(root->index).element.freebonds; j++)

{

node \*daughter = (node\*) malloc(sizeof(node));

daughter->parent = root;

daughter->first = nullptr;

daughter->second = nullptr;

daughter->third = nullptr;

daughter->fourth = nullptr;

daughter->index = -1;

daughter->vertex.atomName = hydrogen;

daughter->bond = 1;

daughter->lastBond = 0;

switch (root->lastBond)

{

case 0:

root->first = daughter;

break;

case 1:

root->second = daughter;

break;

case 2:

root->third = daughter;

break;

default:

root->fourth = daughter;

break;

}

root->lastBond++;

}

}

void Converter::initCoordinates(node\*root)

{

root->vertex.point = {0, 0, 0};

if (root->first->bond == 3)

{

root->first->vertex.point = {0, 0.8, 0};

if (root->second != nullptr)

root->second->vertex.point = {0, -0.8, 0};

}

else

advancedSpin(root);

if (root->first != nullptr)

implementCoordinates(root->first);

if (root->second != nullptr)

implementCoordinates(root->second);

if (root->third != nullptr)

implementCoordinates(root->third);

if (root->fourth != nullptr)

implementCoordinates(root->fourth);

}

void Converter::implementCoordinates(node \*root)

{

if (root->lastBond == 0)

return;

if ((root->first->bond == 3) || ((root->first->bond == 2)&&(root->bond == 2)) || (root->bond == 3))

{

root->first->vertex.point.x = 2 \* root->vertex.point.x - root->parent->vertex.point.x;

root->first->vertex.point.y = 2 \* root->vertex.point.y - root->parent->vertex.point.y;

root->first->vertex.point.z = 2 \* root->vertex.point.z - root->parent->vertex.point.z;

}

else

advancedSpin(root);

if (root->first != nullptr)

implementCoordinates(root->first);

if (root->second != nullptr)

implementCoordinates(root->second);

if (root->third != nullptr)

implementCoordinates(root->third);

}

void Converter::sortTree(node\*root)

{

switch(root->lastBond)

{

case 2:

if (root->first->bond < root->second->bond)

{

node\* temp = root->second;

root->second = root->first;

root->first = temp;

}

break;

case 3:

if (root->second->bond > root->first->bond)

{

node\* temp = root->second;

root->second = root->first;

root->first = temp;

}

else if(root->third->bond > root->second->bond)

{

node\* temp = root->third;

root->third = root->second;

root->second = temp;

}

break;

}

if (root->first != nullptr)

sortTree(root->first);

if (root->second != nullptr)

sortTree(root->second);

if (root->third != nullptr)

sortTree(root->third);

if (root->fourth != nullptr)

sortTree(root->fourth);

}

void Converter::clearTree(node\* root)

{

if (root->first != nullptr)

clearTree(root->first);

if (root->second != nullptr)

clearTree(root->second);

if (root->third != nullptr)

clearTree(root->third);

if (root->fourth != nullptr)

clearTree(root->fourth);

vertices3D.push\_back(root->vertex);

free(root);

}

Point Converter::spin(Point &root, float phi, float theta)

{

Point result;

float r = 0.8;

result.z = r \* cos(theta) \* sin(phi);

result.x = r \* sin(theta) \* sin(phi);

result.y = r \* cos(phi);

return result;

}

void Converter::advancedSpin(node\* root)

{

const float r = 0.8;

if (root->parent == nullptr)

{

switch (root->lastBond)

{

case 4:

root->fourth->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(0);

root->fourth->vertex.point.y = root->vertex.point.y + r \* cos(0);

root->fourth->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(0);

case 3:

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(2.0944);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(2.0944);

case 2:

root->second->vertex.point.x = root->vertex.point.x + r \* sin(4.18879) \* sin(2.0944);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(4.18879) \* sin(2.0944);

case 1:

root->first->vertex.point.x = root->vertex.point.x + r \* sin(2.0944) \* sin(2.0944);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(2.0944) \* sin(2.0944);

break;

}

}

else

{

if ((round(root->vertex.point.x \*100) == round(root->parent->vertex.point.x \* 100))&& (round(root->vertex.point.z \*100) == round(root->parent->vertex.point.z \* 100)))

{

if (root->vertex.point.y>root->parent->vertex.point.y)

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(3.14159) \* sin(1.0472);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(3.14159) \* sin(1.0472);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(-1.0472) \* sin(1.0472);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(-1.0472) \* sin(1.0472);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(1.0472) \* sin(1.0472);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(1.0472) \* sin(1.0472);

}

return;

}

else

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(2.0944);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(2.0944);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(4.18879) \* sin(2.0944);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(4.18879) \* sin(2.0944);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(2.0944) \* sin(2.0944);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(2.0944) \* sin(2.0944);

}

return;

}

}

else

{

if ((root->vertex.point.x < root->parent->vertex.point.x) && (root->vertex.point.y < root->parent->vertex.point.y) && (root->vertex.point.z < root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(3.14159);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(3.14159);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(3.14159);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(3.14159) \* sin(1.0472);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(3.14159) \* sin(1.0472);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(-1.0472) \* sin(1.0472);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(-1.0472) \* sin(1.0472);

}

return;

}

if ((root->vertex.point.x > root->parent->vertex.point.x) && (root->vertex.point.y < root->parent->vertex.point.y) && (root->vertex.point.z < root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(3.14159);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(3.14159);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(3.14159);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(3.14159) \* sin(1.0472);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(3.14159) \* sin(1.0472);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(1.0472) \* sin(1.0472);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(1.0472) \* sin(1.0472);

}

return;

}

if ((root->vertex.point.x < root->parent->vertex.point.x) && (root->vertex.point.y > root->parent->vertex.point.y) && (root->vertex.point.z > root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(0);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(0);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(0);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(2.0944);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(2.0944);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(4.18879) \* sin(2.0944);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(4.18879) \* sin(2.0944);

}

return;

}

if ((root->vertex.point.x > root->parent->vertex.point.x) && (root->vertex.point.y > root->parent->vertex.point.y) && (root->vertex.point.z > root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(0);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(0);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(0);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(2.0944);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(2.0944);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(2.0944) \* sin(2.0944);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(2.0944) \* sin(2.0944);

}

return;

}

if ((round(root->vertex.point.x \*100) == round(root->parent->vertex.point.x \* 100)) && (root->vertex.point.y < root->parent->vertex.point.y) && (root->vertex.point.z > root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(3.14159);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(3.14159);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(3.14159);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(1.0472) \* sin(1.0472);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(1.0472) \* sin(1.0472);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(-1.0472) \* sin(1.0472);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(-1.0472) \* sin(1.0472);

}

return;

}

if ((round(root->vertex.point.x \*100) == round(root->parent->vertex.point.x \* 100)) && (root->vertex.point.y > root->parent->vertex.point.y) && (root->vertex.point.z < root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(0);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(0);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(0);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(4.18879) \* sin(2.0944);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(4.18879) \* sin(2.0944);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(2.0944) \* sin(2.0944);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(2.0944) \* sin(2.0944);

}

return;

}

if ((round(root->vertex.point.x \*100) == round(root->parent->vertex.point.x \* 100)) && (root->vertex.point.y < root->parent->vertex.point.y) && (root->vertex.point.z < root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(3.14159);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(3.14159);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(3.14159);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(2.0944) \* sin(1.0472);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(2.0944) \* sin(1.0472);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(4.18879) \* sin(1.0472);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(4.18879) \* sin(1.0472);

}

return;

}

if ((round(root->vertex.point.x \*100) == round(root->parent->vertex.point.x \* 100)) && (root->vertex.point.y > root->parent->vertex.point.y) && (root->vertex.point.z > root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(0);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(0);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(0);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(1.0472) \* sin(1.0472);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(1.0472) \* sin(1.0472);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(-1.0472) \* sin(1.0472);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(-1.0472) \* sin(1.0472);

}

return;

}

if ((root->vertex.point.x < root->parent->vertex.point.x) && (root->vertex.point.y > root->parent->vertex.point.y) && (root->vertex.point.z < root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(0);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(0);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(0);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(3.14159) \* sin(2.0944);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(3.14159) \* sin(2.0944);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(-1.0472) \* sin(2.0944);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(-1.0472) \* sin(2.0944);

}

return;

}

if ((root->vertex.point.x > root->parent->vertex.point.x) && (root->vertex.point.y > root->parent->vertex.point.y) && (root->vertex.point.z < root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(0);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(0);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(0);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(3.14159) \* sin(2.0944);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(3.14159) \* sin(2.0944);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(1.0472) \* sin(2.0944);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(2.0944);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(1.0472) \* sin(2.0944);

}

return;

}

if ((root->vertex.point.x < root->parent->vertex.point.x) && (root->vertex.point.y < root->parent->vertex.point.y) && (root->vertex.point.z > root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(3.14159);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(3.14159);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(3.14159);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(4.18879) \* sin(1.0472);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(4.18879) \* sin(1.0472);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(1.0472);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(1.0472);

}

return;

}

if ((root->vertex.point.x > root->parent->vertex.point.x) && (root->vertex.point.y < root->parent->vertex.point.y) && (root->vertex.point.z > root->parent->vertex.point.z))

{

if (root->third != nullptr)

{

root->third->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(3.14159);

root->third->vertex.point.y = root->vertex.point.y + r \* cos(3.14159);

root->third->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(3.14159);

}

if (root->second != nullptr)

{

root->second->vertex.point.x = root->vertex.point.x + r \* sin(2.0944) \* sin(1.0472);

root->second->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->second->vertex.point.z = root->vertex.point.z + r \* cos(2.0944) \* sin(1.0472);

}

if (root->first != nullptr)

{

root->first->vertex.point.x = root->vertex.point.x + r \* sin(0) \* sin(1.0472);

root->first->vertex.point.y = root->vertex.point.y + r \* cos(1.0472);

root->first->vertex.point.z = root->vertex.point.z + r \* cos(0) \* sin(1.0472);

}

return;

}

}

}

}

void Converter::tethrahedron(elem vertex)

{

bool top = false;

float degree;

switch(vertex)

{

case carbon:

top = true;

degree = 1.9111355; //109.5

break;

case phosphorus:

degree = 1.623156; //93.5

break;

case nitrogen:

default:

degree = 1.8675023; //107.8

break;

}

Point root, \_1, \_2, \_3, \_4;

if (top)

{

root = {0, 0, 0};

\_1 = spin(root, 0, 0);

\_2 = spin(root, degree, 2.0944);

\_3 = spin(root, degree, 4.18879);

\_4 = spin(root, degree, 0);

}

else

{

root = {0, 0, 0};

degree = cos(degree/2);

degree = pi - acos(sqrt(1-4/3\*degree\*degree));

\_1 = spin(root, degree, 2.0944);

\_2 = spin(root, degree, 4.18879);

\_3 = spin(root, degree, 0);

}

vertex3D \_point = {root, vertex};

vertices3D.push\_back(\_point);

\_point.point =\_1;

\_point.atomName = hydrogen;

vertices3D.push\_back(\_point);

\_point.point = \_2;

vertices3D.push\_back(\_point);

\_point.point = \_3;

vertices3D.push\_back(\_point);

if (top)

{

\_point.point = \_4;

vertices3D.push\_back(\_point);

}

}

void Converter::centrifyCoordinates()

{

float x, y, z;

x = y = z = 0;

for (unsigned int i = 0; i<vertices3D.size(); i++)

{

x+=vertices3D[i].point.x;

y+=vertices3D[i].point.y;

z+=vertices3D[i].point.z;

}

x /= vertices3D.size();

y /= vertices3D.size();

z /= vertices3D.size();

for (unsigned int i = 0; i<vertices3D.size(); i++)

{

vertices3D[i].point.x -= x;

vertices3D[i].point.y -= y;

vertices3D[i].point.z -= z;

}

}

void Converter::triangle(elem vertex)

{

float degree;

switch(vertex)

{

case sulfur:

degree = 1.605703;

break;

default:

degree = 1.81514242;

}

vertex3D root = {{0, 0, 0}, vertex};

vertices3D.push\_back(root);

float r = 0.8;

Point \_1;

\_1.y = - r \* cos(degree/2);

\_1.z = 0;

\_1.x = r \* sin(degree/2);

root.point = \_1;

root.atomName = hydrogen;

vertices3D.push\_back(root);

Point \_2;

\_2 = \_1;

\_2.x = - \_2.x;

root.point = \_2;

vertices3D.push\_back(root);

}

void Converter::linear(elem vertex)

{

vertex3D \_1 = {{-0.4, 0, 0}, vertex};

vertices3D.push\_back(\_1);

vertex3D \_2 = {{0.4, 0, 0}, hydrogen};

vertices3D.push\_back(\_2);

}

bool Converter::isSimple(std::vector<vertex2D> &vertices2D)

{

if (vertices2D.size() ==1)

{

switch(vertices2D[0].element.atomName)

{

case carbon:

case phosphorus:

case nitrogen:

tethrahedron(vertices2D[0].element.atomName);

break;

case sulfur:

case oxygen:

triangle(vertices2D[0].element.atomName);

break;

default:

linear(vertices2D[0].element.atomName);

}

return true;

}

return false;

}

char Converter::getPossibleBonds(elem activeElement)

{

switch (activeElement)

{

case carbon:

return 4;

case nitrogen:

case phosphorus:

return 3;

case oxygen:

case sulfur:

return 2;

default:

return 1;

}

}

Листинг файла structures.h

#ifndef STRUCTURES\_H\_INCLUDED

#define STRUCTURES\_H\_INCLUDED

#ifndef pi

#define pi 3.14159265358979323846

#endif // pi

enum elem {carbon, nitrogen, phosphorus, oxygen, sulfur, fluorine, chlorine, bromine, iodine, hydrogen};

typedef struct

{

elem atomName;

char freebonds;

char bond;

} atom;

typedef struct

{

float red;

float green;

float blue;

} colour;

typedef struct

{

short int x;

short int y;

short int prevx;

short int prevy;

bool shortRear;

bool shortFront;

atom element;

} vertex2D;

typedef struct

{

float x;

float y;

float z;

} Point;

typedef struct

{

Point point;

elem atomName;

} vertex3D;

typedef struct quaternaryTree

{

struct quaternaryTree \*first;

struct quaternaryTree \*second;

struct quaternaryTree \*third;

struct quaternaryTree \*fourth;

struct quaternaryTree \*parent;

vertex3D vertex;

short int index;

char lastBond;

char bond;

} node;

#endif // STRUCTURES\_H\_INCLUDED

ВЕДОМОСТЬ

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КР 1–40 01 01 205 ПЗ | | | | Пояснительная записка | | | | 83 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 951002 205 СП | | | | Схема программы «Молекулярный редактор» | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КР 1-40 01 01 205 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Молекулярный редактор  Ведомость курсовой  работы |  | | | | Лист | Листов |
| Разраб. | | Будович И.В. |  | 07.12.20 | Т |  | |  | 83 | 83 |
| Пров. | | Шостак Е.В. |  | 07.12.20 | Кафедра ПОИТ  гр. 951002 | | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |