Министерство образования и науки Российской Федерации
Национальный исследовательский технологический университет «МИСиС»
Институт информационных технологий и
автоматизированных систем управления
Кафедра Инженерной Кибернетики

Курсовая работа

по дисциплине «Технологии программирования»
на тему : «Редактор «скелета»»

Выполнила:
Студентка гр. БПМ-18-2
Самсонова Е.О.

Проверил:
доцент кафедры ИК, к.т.н.
Полевой Д.В.

Москва, 2020

Оглавление:

1. Задача

Редактор «скелета».

Скелет задается как :

1. Граф на плоскости («суставы»): Qt компонента при помощи QGraphicScene/QraphicView для визуального редактирования.
   Пользователь может:
   - Загрузить скелет из файла
   - Сохранить скелет в файл
   - Мышкой «тянуть» за ключевые точки (модифицировать скелет, не нарушая ограничения)
2. Набор ограничений скелета :
   - минимальная/максимальная длина ребер
   - диапазон допустимых углов

2. Описание программы

При разработке приложения были создан графический интерфейс при помощи Qt (см. Рис. 1)

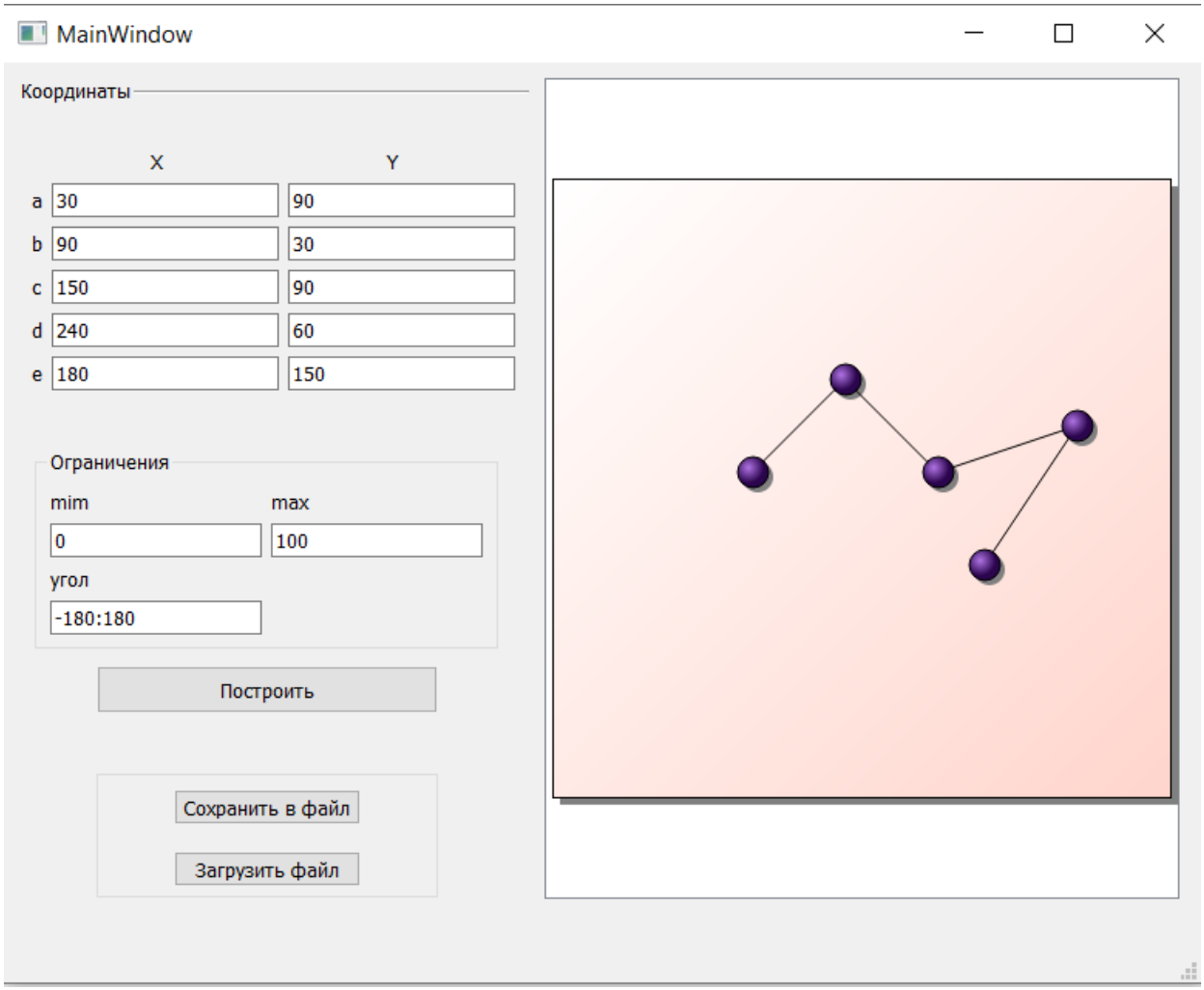

Рис.1 (Основной вид приложения при запуске.)

Пользователь может построить граф с помощью координат, как показано на Рис.1. Также есть возможность с помощью мышки «перетягивать» точки, в таком случае координаты, отображенные слева меняются автоматически (см. Рис.2). В случае нарушения ограничений появляется предупреждение (см. Рис. 3). Еще пользователь может загружать раннее созданный файл (в формате JSON File) (см. Рис. 4)

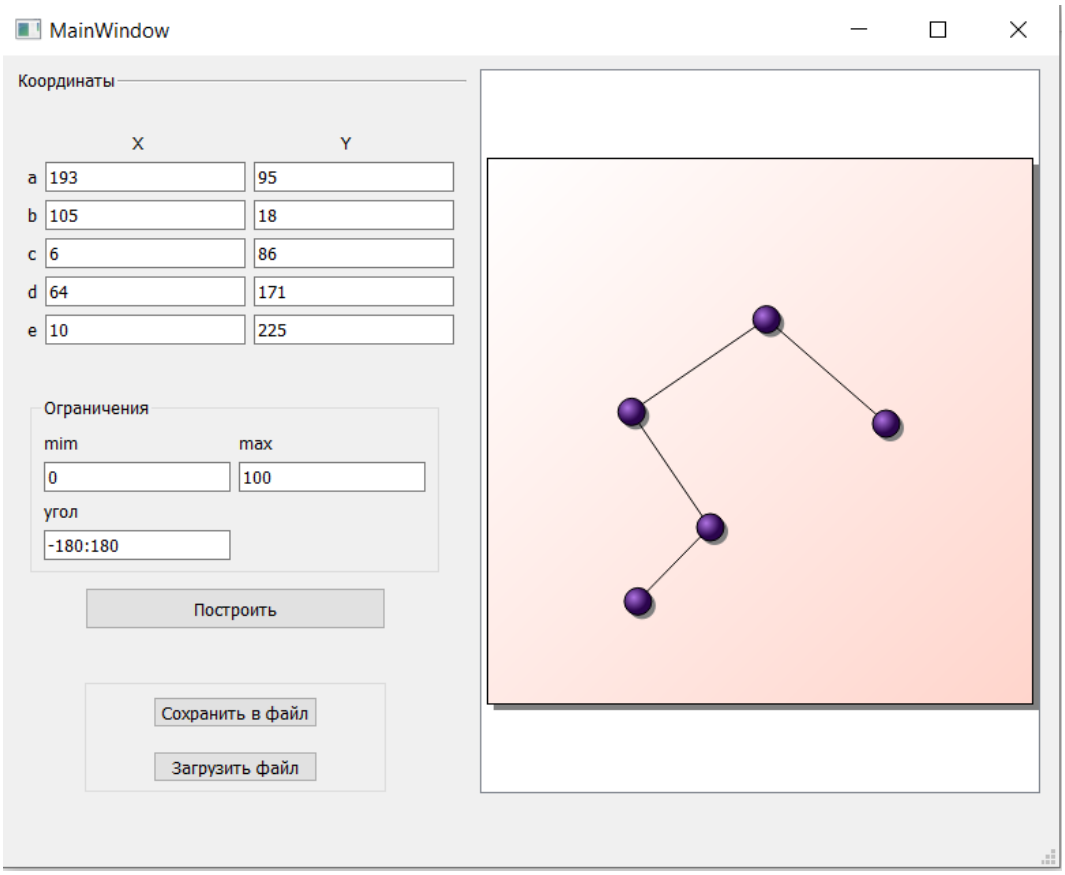

Рисунок 2 (Автоматическая смена координат при передвижении точек «вручную»)

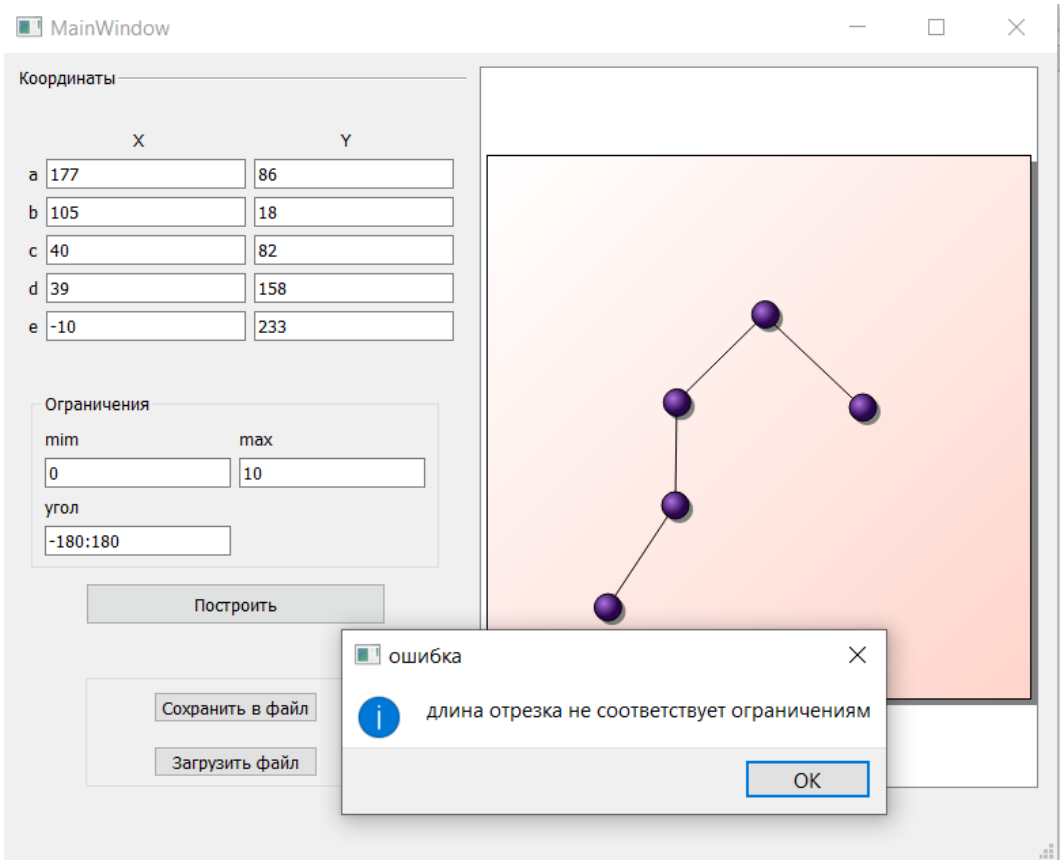

Рисунок 3 (Ошибка, возникающая при нарушении заданных ограничений.)

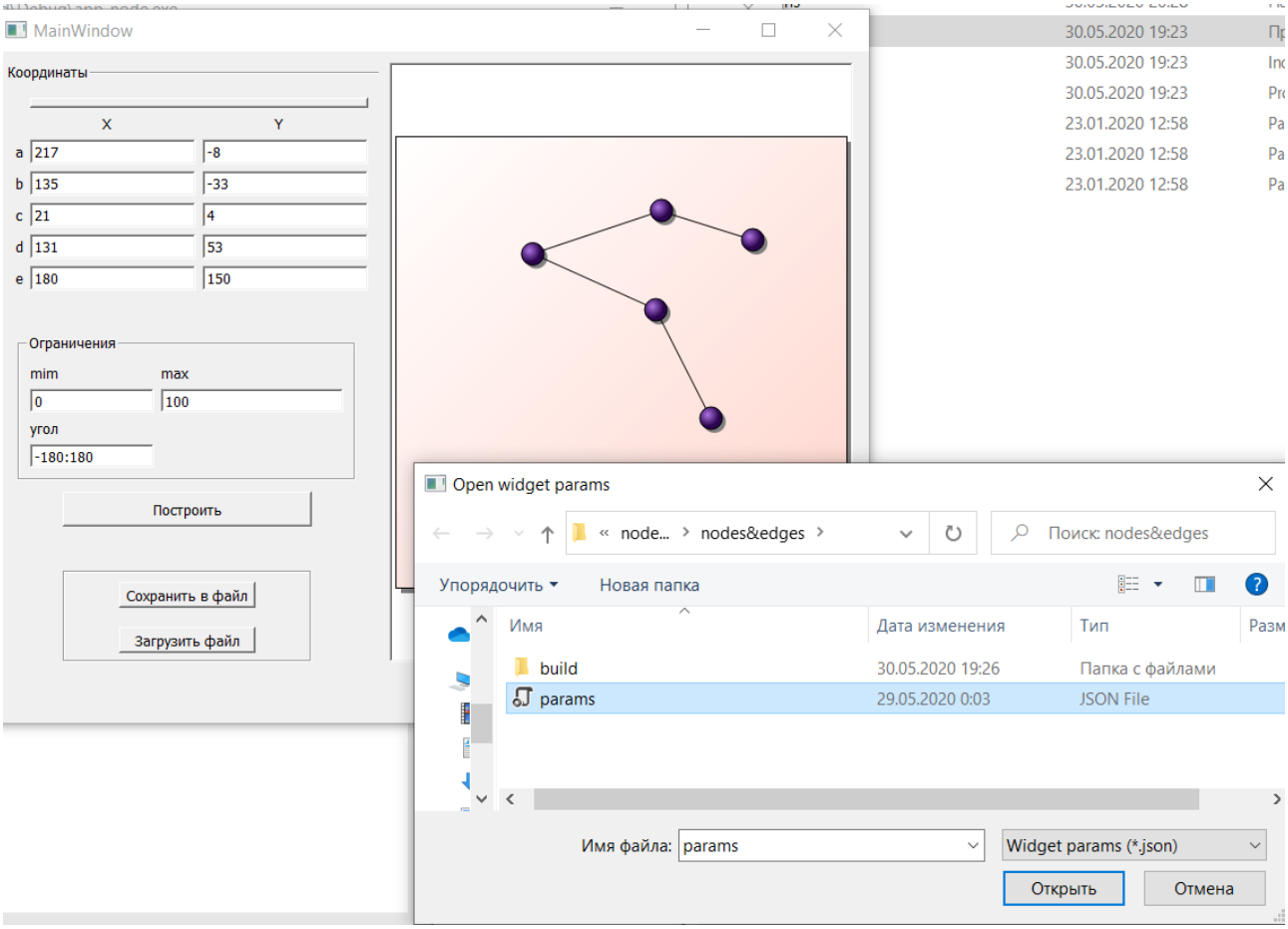

Рисунок 4 (Загрузка созданного раннее файла в формате JSON file.)

3. Инструкция по сборке

Ссылка на репозиторий: https://github.com/kate-coder/Kursovaya.git

Для сборки требуется наличие: Qt5.12.6+ (проверено на версии 5.14.1), Visual Studio (проверено на Visual Studio 2019, должно работать на версии Visual Studio 2017), CMake 3.17+ (проверено на 3.17+).

1. Установить Qt, Visual Studio, CMake при необходимости
2. Скачать репозиторий с проектом и распаковать в удобном месте
3. Открыть директорию с проектом в CMake GUI
4. Нужно исправить путь к Qt в CMakeLists.txt (см. Рисунок 5)
5. Установить опции QT_PROJECT_PATH в проекте (это - путь к папке, содержащей Qt5Config.cmake) (см. Рисунок 6)
6. Сконфигурировать проект и открыть его в Visual Studio (configure – generate – open project) (см. Рисунок 6)
7. Когда откроется Visual Studio нужно нажать Ctrl+Shift+B, а после этого необходимо «собрать только INSTALL» (см. Рисунок 7)
8. После выполнения всех пунктов заходим в нашу папку с приложением node&edge\nodes&edges\build\Debug и видим приложение, которое нужно запустить, однако после запуска возможно появление ошибки (см. Рисунок 8)
9. Для устранения нашей ошибки нужно создать папку platforms в той же самой папке, где и находится наше приложение (см. Рисунок 9)
10. В папку platforms копируем два файла : qwindows.dll, qwindowsd.dll, которые я взяла по адресу (в зависимости от версии qt на разных компьютерах он может отличаться) : C:/Qt/5.14.1/msvc2017_64/plugins/platforms. После чего снова запускаем наше приложение и видим, что ошибка устранена, все работает.

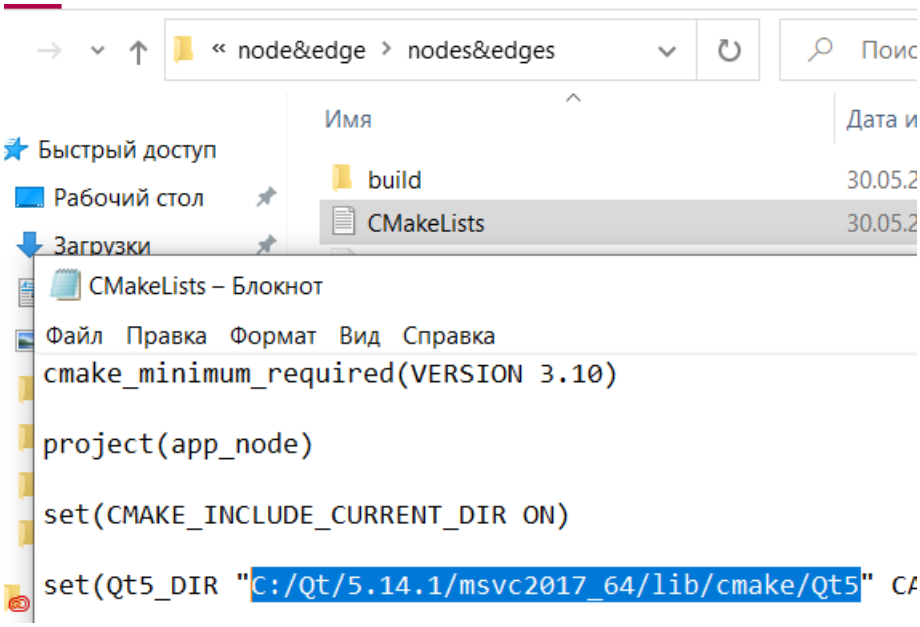

Рисунок 5 (Выделенную строчку необходимо заменить.)

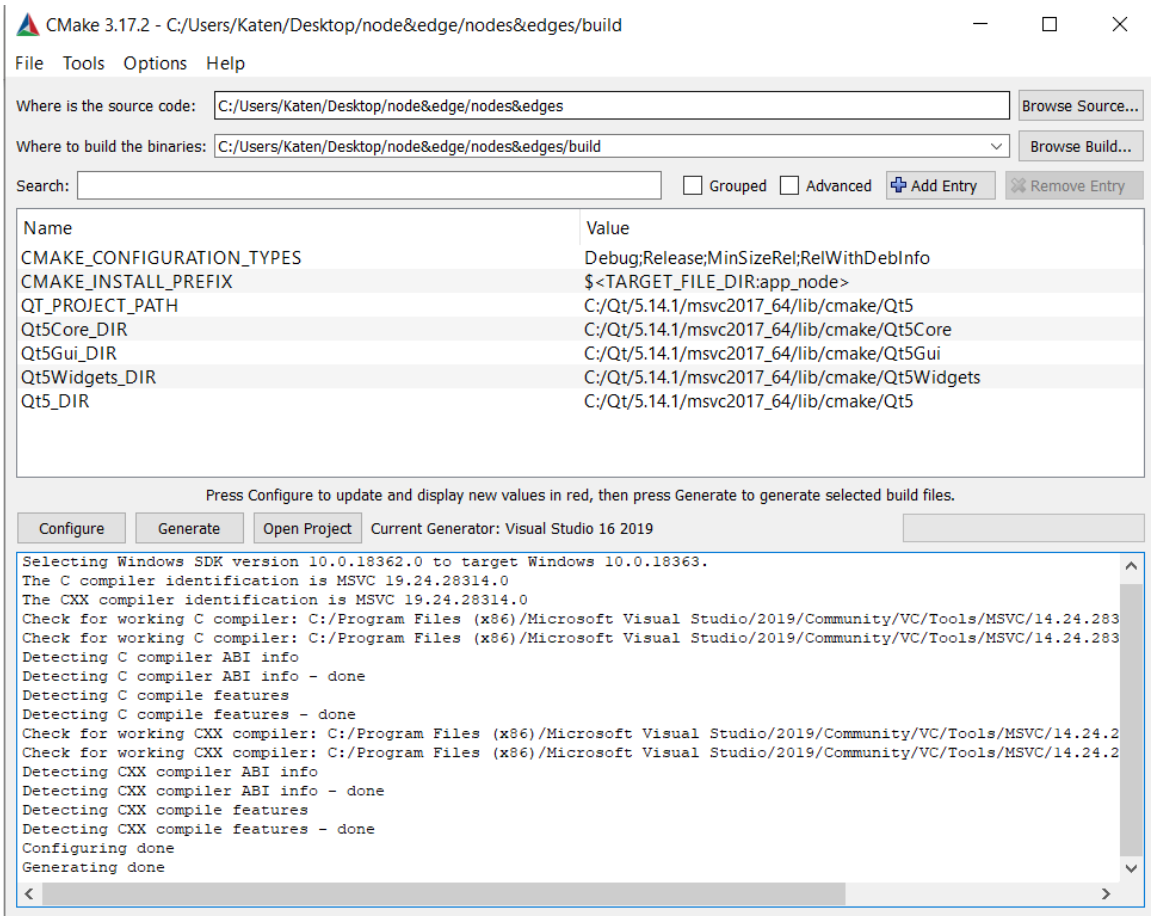

Рисунок 6 (Результат успешной работы CMake.)

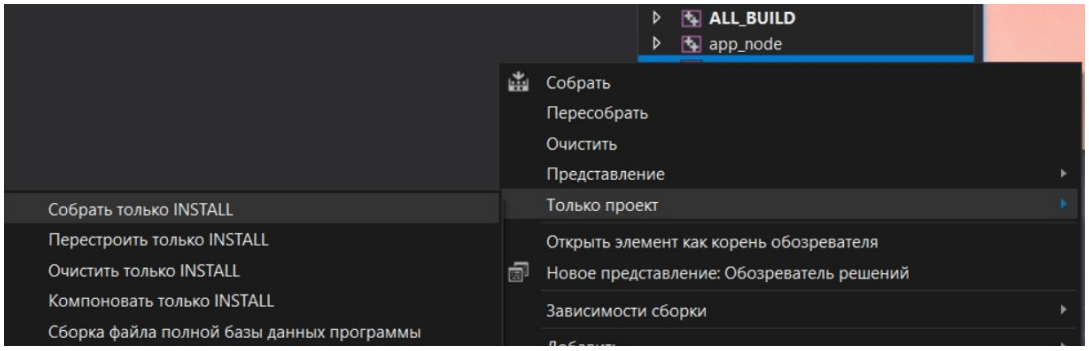

Рисунок 7

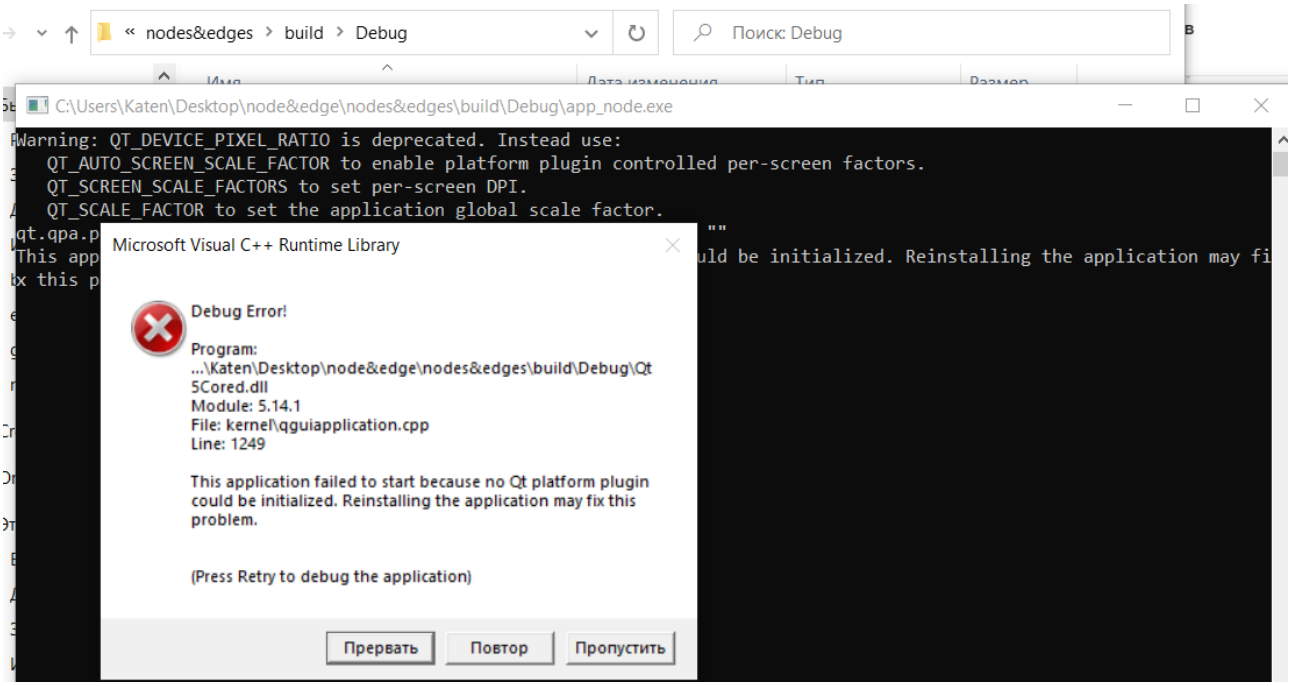

Рисунок 8 (Ошибка при запуске приложения.)

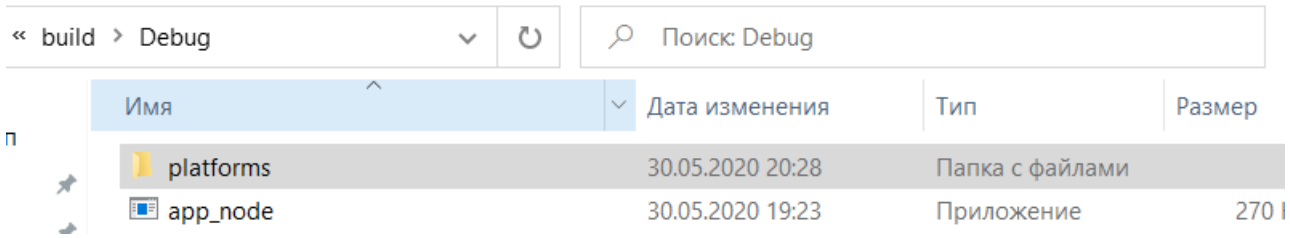

Рисунок 9 (Создание папки platforms в той же директории, что и наше приложение.)

4. Реализация классов

4.1 Класс MainWindow

Основной класс для графического интерфейса

#include <mainwindow.h>

#include "ui_mainwindow.h"

#include <QMessageBox>

#include <graphwidget.h>

#include <QtMath>

#include <QFileDialog>

#include "node.h"

Открытые члены

MainWindow (QWidget *parent= nullptr)

Конструктор Qt.

~MainWindow () override Де*структор Qt.*

Закрытые слоты

void on_pushButton_clicked()

Событие при нажатии кнопки Построить граф

void on_pushButton_2_clicked()

Событие при нажатии кнопки Сохранение файла

void on_pushButton_3_clicked()

Событие при нажатии кнопки Загрузить из файла

Закрытые члены

void setPos();

Метод для построения точек по заданным координатам

void setRestrictions();

Метод для установки ограничений макс/мин длины ребер, а также угла между ребрами

double angle_point(QPoint *a, QPoint *b, QPoint *c);

Метод для определения угла между двумя ребрами (тремя точками)

Закрытые данные

Ui::MainWindow * **ui_** {}

*Интерфейс объекта класса **MainWindow**.*

GraphWidget ***test**

Объект класса QtWidget предназначенный для взаимодействия с графом и

его построением на плоскости

Методы

void setPos() - *Строит точки по заданным координатам*

nodes	Точки

void setRestrictions();

Метод для установки ограничений макс/мин длины ребер, а также угла между ребрами

angleFrom	Минимальный угол
angelTo	Максимальный угол
minLength	Минимальная длина ребра
maxLength	Максимальная длина ребра

double angle_point(QPoint *a, QPoint *b, QPoint *c);

Метод для определения угла между двумя ребрами (тремя точками)

4.2 Класс Node

Класс для обработки узлов (точек)

```
#include "edge.h"
#include "node.h"
#include "graphwidget.h"
#include <QGraphicsScene>
#include <QGraphicsSceneMouseEvent>
#include <QPainter>
#include <QStyleOption>
```

Классы

Class **Edge**

Класс Edge представляет в данном приложении линии, соединяющие узлы (точки)

Class **GraphWidget**

Подкласс QGraphicsView, который представляет главное окно с графом

Class **Node**

Класс для работы с узлами (точками)

Открытые члены

Node(GraphWidget *graphWidget);

Конструктор для точек

void **addEdge**(Edge *edge);

Функция добавления рёбер.

QVector<Edge *> **edges**() const;

Функция возвращает список присоединённых рёбер

enum { Type = UserType + 1 };

Переопределение пользовательского типа

int **type**() const override { return Type; }

Функция возвращает тип элемента в виде целого числа.

void **calculateForces**();

Функция для перемещения точек с помощью мыши

bool **advancePosition**();

Функция обновляет текущую позицию элемента.

QRectF **boundingRect**() const override;

Эта чисто виртуальная функция определяет внешние границы элемента в виде прямоугольника

QPainterPath **shape**() const override;

Функция возвращает фигуру этого элемента в виде QPainterPath в локальных координатах.

void **paint**(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override;

Функция реализует отрисовку узла

QLineEdit *textX;

Абсцисса узла

QLineEdit *textY;

Ордината узла

Закрытые члены

QVariant **itemChange**(GraphicsItemChange change, const QVariant &value) override;

Я переопределила функцию itemChange(), чтобы настроить позицию всех соединяющих рёбер, и для уведомления сцены о том, что элемент передвинулся (т.е., "что-то произошло"). Это запустит новый перерасчёт координат.

void **mousePressEvent**(QGraphicsSceneMouseEvent *event) override;

Метод переопределения обработчики события нажатия кнопки мыши для обновления визуального внешнего вида узлов

void **mouseReleaseEvent**(QGraphicsSceneMouseEvent *event) override;

Метод переопределения обработки события отпускания кнопки мыши для обновления визуального внешнего вида узлов

QVector<Edge *> edgeList;

Список определенных ребер графа

QPointF newPos;

Позиция элемента, равная pos(), которая задана в координатах родителя.

4.3 Класс Edge

#include "edge.h"

#include "node.h"

 # include <QGraphicsItem>

Классы

Class Node

Класс для обработки узлов

Class **Edge**

Класс для обработки рёбер графа

Открытые члены

Edge(Node *sourceNode, Node *destNode);

Конструктор

Node ***sourceNode()** const;

Указатель на источник

Node ***destNode**() const;

Указатель на приемник

void **adjust**();

Функция для обновления начальной и конечной позиции данного ребра. Определяет две точки: sourcePoint и destPoint, указывающие на исходные точки узлов источника и приёмника, соответственно.

enum { Type = UserType + 2 };

int **type**() const override { return Type; }

int maxLength = -1;

Максимальная длина

int minLength = -1;

Минимальная длина

int angelFrom = -1;

Мин угол

int angelTo = -1;

Макс угол

Закрытые члены

QRectF **boundingRect**() const override;

void **paint**(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override;

Функция реализует отрисовку ребра

Node *source, *dest;

QPointF sourcePoint;

Источник (начальная точка ребра)

QPointF destPoint;

Приемник (конечная точка ребра)

4.4 Класс Graphwidget

#include "graphwidget.h"

#include "edge.h"

#include "node.h"

#include <math.h>

#include <QKeyEvent>

#include <QJsonParseError>

#include <QJsonObject>

#include <QJsonArray>

Подкласс QGraphicsView, который предоставляет главное окно

Открытые члены

GraphWidget(QWidget *parent = nullptr);

Конструктор

void **nodeSetPos(**int index, qreal x, qreal y);

void **setRestrictions**(qreal maxLength, qreal minLength, qreal angelFrom, qreal angelTo);

void **itemMoved()**;

Функция для уведомления об изменениях в графе узлов сцены

QVector<Node *> **getNodes**();

void **saveToJson**();

Метод для сохранения графа в файл Json

QList<NodeParams> **readFromJson**(QString fileName);

Метод для загрузки графа из файла Json

Открытые слоты

void **shuffle**();

Перетаскивание графа

void **zoomIn**();

Приближение сцены с графом

void **zoomOut**();

Отдаление сцены

Закрытые данные

void **keyPressEvent**(QKeyEvent *event) override;

Это обработчик GraphWidget события клавиши. Клавиши стрелок перемещают центральный узел, клавиши '+' и '-' увеличивают и уменьшают масштаб.

void **timerEvent**(QTimerEvent *event) override;

Ф-ия, обработывающая события таймера. Суть заключается в запуске алгоритмов вычисления всех сил в виде плавной анимации

void **wheelEvent**(QWheelEvent *event) override;

Ф-ия, обработывающая события колёсика мыши

void **drawBackground**(QPainter *painter, const QRectF &rect) override;

Метод для визуализации фона (цвет,тени)

void **scaleView**(qreal scaleFactor);

Метод для увеличения и уменьшения масштаба фона с графом

int timerId = 0;

Таймер спроектирован для остановки, когда граф стабилизируется, и запускается когда граф снова становится нестабильным.