МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

# АКТУАЛИЗАЦИЯ МОДУЛЯ АНАЛИЗА И ФОРМИРОВАНИЯ ДАННЫХ О ГРУЗЕ. РАЗРАБОТКА МОДУЛЯ АНАЛИЗА ПРОХОДИМОСТИ ГРУЗА ПО УЧАСТКУ ПУТИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Русяева Сергея Алексеевича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2021

# ВВЕДЕНИЕ

Данная дипломная работа является частью проекта, направленного на оптимизацию перевозки негабаритных грузов. Данный проект разрабатывается уже второй год студентами факультета КНиИТ совместно с сотрудниками географического факультета Саратовского государственного университета и представителями компании «Российские железные дороги» в Поволжье.

Для упрощения работы операторов и повышения скорости и эффективности расчетов был разработан проект по частичной автоматизации расчетов, связанных с определением негабаритности груза.

Целью данной дипломной работы доработка полуавтоматического способа ввода критических точек негабаритного груза модуля *«Анализа и формирования данных о грузе»* (далее первый модуль), внести некоторые доработки в данный модуль, а также разработать модуль *«Анализа проходимости груза на участке пути»* (далее второй модуль) в рамках комплекса приложений для пропуска негабаритных грузов по железной дороге.

Из цели работы вытекают следующие задачи:

- изучить литературу, связанную с предметной областью;
- исследовать средства языка Python;
- исследовать фреймворк PyQt5;
- внести доработки в первый модуль приложения в соответствии с требованиями заказчика;
- реализовать считывание схем груза в новом формате;
- изучить средства ГИС;
- разработать способ автоматизации пропуска груза по участку пути;
- исследовать необходимый математический аппарат для расчётов, необходимых во втором модуле приложения;
- реализовать второй модуль приложения, предназначенный для пропуска негабаритных грузов по участку пути (станции).

Объектом исследования являются все теоретические аспекты, связанные с пропуском негабаритного груза по участку пути, а предметом — язык Python в совокупности с фреймворком PyQt5, необходимым для реализации GUI.

Данная дипломная работа состоит из:

1. Теоретическая часть.
2. Практическая часть

# 1 Теоретическая часть

## 1.1 Постановка задачи

В первом модуле комплекса приложений для перевозки негабаритных грузов реализовано три способа ввода критических точек: ручной, полуавтоматический и автоматический. В рамках данной дипломной работы требовалось доработать полуавтоматический способ ввода.

В рамках данной дипломной работы необходимо модифицировать первый модуль приложения следующим образом:

1. Перестроить структуру классов проекта, провести рефакторинг кода для улучшения читаемости, удаления дублирующих элементов, а также для оптимизации. Кроме того, для повышения понятности кода необходимо добавить сигнатуры методов.
2. Добавить возможность вывода данных о грузе на форму для наглядности и удобства пользователя.
3. Модифицировать полуавтоматический способ ввода следующим образом:
   - добавить возможность считывать данные формата DXF, не нарушая при этом существующий способ ввода чертежей груза;
   - добавить возможность выбора направления движения груза на схеме.

Кроме того, в рамках данной работы необходимо реализовать прототип второго модуля комплекса приложений для автоматизации перевозок негабаритных грузов по железной дороге.

Основная цель второго модуля приложений — проверить, пройдёт ли негабаритный груз по участку пути или нет. Для решения данной задачи необходимы следующие данные:

- данные о негабаритном грузе, полученные на выходе первого модуля: критические точки, данные о примитиве, вагоне и т. п.;
- данные об участке пути (каждый участок пути состоит из конечного числа слоев, каждый из которых представляет один тип объектов).

Оператор при работе с данным модулем должен иметь возможность импортировать данные о грузе и станции, произвести необходимые настройки и получить графическое представление о возможности провоза груза по данному участку.

Для реализации прототипа второго модуля необходимо:

1. Реализовать интерфейс второго модуля, состоящий из:
   — сцены для отрисовки возможности провоза негабаритного груза через участок пути;
   — списка элементов, представляющих собой слои участка железнодорожного пути, с возможностью выбора необходимых для анализа слоев;
   — кнопки для импорта данных о грузе;
   — кнопки для импорта данных об участке пути;
   — кнопки для выбора слоя с дорогами;
   — полей для вывода данных о грузе (для наглядного представления);
   — кнопки для проведения анализа провоза груза по участку пути.
2. Реализовать считывание данных об участке пути в нужном формате.
3. Реализовать считывание данных о грузе в формате JSON.
4. Реализовать расчёты, необходимые для проверки возможности перевозки груза по данному участку пути.

### 1.2 Форматы MIF, MID

Поскольку множество карт местности, объектов, областей сегодня создаётся в географической информационной системе `MapInfo` пришлось изучить форматы `MIF/MID`.

`MIF/MID` форматы позволяют сопоставлять различным геометрическим объектам настраиваемые данные. Файл `MIF` представляет собой текстовый файл, который содержит в себе информацию об объектах с геометрической точки зрения. Файл `MID` содержит текстовую информацию об объекте [1].

Рассмотрим объекты, которые хранит в себе `MIF` файл:

— `Point` — объект типа точка, который имеет координаты $x$ и $y$;
— `Line` — объект типа линя, который имеет координаты $x$ и $y$ для двух концов отрезка;
— `Polyline` — объект типа полилиния, задаётся множеством координат, соединённых между собой точек. Может состоять из нескольких секций, в этом случае для полилинии задаётся параметр `Multiple`;
— `Region` — объект типа область, который состоит из одного или нескольких полигонов. Полигоны задаются аналогично полилинии;

— `Ellipse` — объект типа эллипс который содержит координаты противоположных углов описанного прямоугольника;
— `Arc` — объект типа дуга, который содержит координаты противоположных углов описанного прямоугольника, а также значение начальных и конечных углов дуги;
— `Rect` — объект типа прямоугольник, который содержит координаты своих противоположных углов.

### 1.3 Библиотека ezdxf для работы с DXF-файлами

DXF — это открытый формат хранения чертежей. Он представляет собою текстовый файл. В нём каждый объект описывается отдельными тегами и значениями. Поскольку данный формат текстовый, то он имеет существенный недостаток: чертёж с большим количеством объектов может занимать много места.

В файле DXF есть несколько типов примитивов:

1. `LINE` — прямая линия. Задаётся координатами начальной и конечной точек;
2. `POINT` — точка. Задаётся своими координатами;
3. `CIRCLE` — круг. Задаётся координатами центра и величиной радиуса;
4. `ARC` — дуга. Задаётся аналогично кругу, но также добавляются значения начального и конечного угла;
5. `TEXT` — текст. Задаётся координатами места вставки, высотой и фразой, которую необходимо вывести на чертёж;
6. `INSERT` — вставка. Представляет собой набор элементов с одинаковыми параметрами;
7. `POLYLINE` — ломаная линия. Задаётся координатами каждой точки [2].

`ezdxf` — это интерфейс Python для работы с форматом DXF (файл обмена чертежами), разработанный Autodesk. `ezdxf` позволяет разработчикам читать и изменять существующие чертежи DXF или создавать новые чертежи DXF.

Для работы с данной библиотекой требуется версия Python не ниже, чем 3.7.

### 1.4 Фреймворк PyQt5

PyQt5 — это полный набор привязок Python для Qt пятой версии. Он реализован в виде более 35 модулей расширения и позволяет использовать

Python в качестве альтернативного языка разработки приложений для C++ на всех поддерживаемых платформах, включая iOS и Android.

В PyQt5, как и в Qt5, используется механизм слотов и сигналов. Если происходит какое-то событие, генерируется сигнал. Например, событием может быть нажатие на кнопку, или установка соединения с базой данных [3].

В свою очередь слот — это обычный метод, который мы связываем с сигналом. Он сработает только тогда, когда генерируется связанный с ним сигнал [4].

### 1.5 Геометрия расчётов

В нашем приложении нам понадобятся некоторые геометрические формулы, а именно: уравнение прямой, уравнение эллипса, барицентрические координаты. Все они нужны нам для того, чтобы найти пересечение объектов с негабаритным грузом, провозимым по железной дороге.

Чтобы найти пересечение двух прямых требуется приравнять их уравнения друг к другу:

$$k_1 x + b_1 = k_2 x + b_2$$

Если $k_1 = k_2$, то прямые параллельны и не пересекаются.

Если прямые не параллельны перенесём $k_2 x$ в левую часть, а $b_1$ в правую:

$$k_1 x - k_2 x = b_2 - b_1$$

Выразим $x$ из данного выражения:

$$x = \frac{b_2 - b_1}{k_1 - k_2}$$

После вычисления значения $x$ подставим его в уравнение любой из двух прямых, таким образом найдя значение $y$. Теперь определив лежит ли найденная точка на каждом из отрезков, мы сделаем вывод пересекаются ли нужные нам отрезки.

Для нахождения пересечения прямой и эллипса требуется решить следующее квадратное уравнение:

$$(b_r^2 + a_r^2 k^2)x^2 + 2(a_r^2 k(b_1 - y_0) - b_r^2 x_0)x + b_r^2 x_0^2 + a_r^2 (b_1 - y_0)^2 - a_r^2 b_r^2 = 0$$

Решив данное квадратное уравнение, мы найдём пересечение эллипса и прямой, заданной двумя точками отрезка. После проверки полученных (полученной) точек (точки) на принадлежность отрезку, сделаем вывод о том, пересекает ли данный отрезок эллипс или нет.

Для нахождения пересечения дуги и прямой дополнительно к решению уравнения эллипса нам понадобится найти угол:

$$\alpha = \arccos \frac{x - x_0}{a},$$
$$\alpha = \arcsin \frac{y_0 - y}{b},$$

Проверим вхождение угла в диапазон углов дуги, и если угол входит в диапазон, то отрезок пересекает дугу, в противном случае нет.

## 2 Практическая часть

В практической части опишем все модификации, внесённые в первый модуль комплекса приложений для оптимизации перевозки негабаритных грузов, а также о разработке второго модуля.

### 2.1 Структура приложения

Первый модуль был разработан в прошлом году. Однако, при разработке нового функционала были выявлены следующие существенные недостатки изначальной структуры приложения:

- интерфейс и слоты форм лежали в одном файле, что существенно усложняло работу с кодом;
- слишком длинные файлы с кодом, которые также можно было разделить;
- некоторые элементы располагались далеко друг от друга, хотя логически это было неверно;
- поскольку приложение большое, было непонятно, данные какого типа передавать в методы.

В связи с наличием этих недостатков было решено переработать структуру проекта. В первую очередь необходимо было разделить интерфейс, слоты и функции, которые используются в слотах в 3 разных файла. Также было необходимо изменить структуру пакетов приложения. Кроме того, везде понадобилось добавить сигнатуры в методы, что потребовало изменения иерархии некоторых классов.

### 2.2 Вывод информации о грузе

Было решено добавить поля для вывода всех данных о грузе, чтобы пользователь всё видел и при необходимости мог вернуться к начальному этапу и изменить значения.

Для решения этой задачи были внесены некоторые модификации в модуль. Во все всплывающие окна (ввод данных о грузе, ввод параметров примитива и ввод данных о преобразованиях примитива) была добавлена возможность изменять введённую информацию.

Для того, чтобы изменить данные оператор должен снова открыть форму, данные которой он хочет изменить. В форме он увидит те данные, которые ввёл ранее. Эти данные сохранены в нередактируемом виде. Если пользователь захочет их изменить, то он может нажать на кнопку «Изменить».

Кроме того, для наглядности на втором шаге работы первого модуля был добавлен `QGroupBox`, в котором выводятся все введённые данные о грузе.

## 2.3 Модификации в полуавтоматическом вводе

В рамках дипломной работы был модифицирован полуавтоматический способ ввода. Здесь требовалось внести 2 изменения:

— ввести направление движения на схеме в полуавтоматическом вводе и учесть это при расчётах;
— разработать возможность чтения чертежей из файлов формата DXF.

### 2.3.1 Направления движения на схеме

При вводе схем груза мы выбираем 3 вида:

— вид сверху;
— вид сзади;
— вид сбоку.

Вид сзади не имеет каких-либо особенностей в отличие от остальных видов. Вид сбоку и сверху может быть задан в одном из двух направлений: влево или вправо. От этого направления зависят расчёты. По этой причине было предложено ввести функционал, который позволяет определить направление движения и учесть это при расчёте критических точек.

Для этого было введено новое поле для выбора направления. Это поле заполняется только для двух описанных выше проекций. После подтверждения выбора направления на схеме появляется стрелочка, которая показывает это направление.

## 2.4 Структура второго модуля

Прежде чем приступить к написанию второго модуля приложения, была составлена структура приложения.

Рассмотрим данную структуру подробнее. Корневой директорией является `RZHD_APP_module2`. В ней хранятся все артефакты проекта. В данной директории, как и в случае с первым модулем, находятся 2 основные директории:

— `Interface` — хранит все классы, отвечающие за интерфейс приложения и способы взаимодействия с ним;
— `Logic` — хранит классы, которые относятся к логике.

Рассмотрим структуру поддиректории Interface:

1. Ui — директория, которая содержит формы:
   - ChooseLayerRoadsForm.py — форма для выбора слоя дорог;
   - MainWindow.py — основное окно приложения;
   - InfoLoadForm.py — форма, которая содержит параметры груза.
2. Slots — хранит классы со слотами для форм:
   - SlotsChooseLayerRoads.py — слоты для формы для ввода данных о грузе;
   - MainWindow.py — слоты для формы для основного окна приложения;
   - Slots.py — родительский класс для всех классов со слотами. Необходим для проставления сигнатур в методах.
3. Utils — хранит дополнительные классы для работы с формами:
   - Graphics — данная директория содержит в себе два класса для работы с графикой, а именно:
     - GraphicsScene.py — файл с классом сцены для вывода объектов станции;
     - GraphicsView.py — файл с классом окна для просмотра GraphicsScene;
   - CheckItem — класс, который позволяет нам отключать и включать слои, а также выбирать слой дорог;
   - MainWindowFunction.py — класс для работы с окном приложения.

Рассмотрим также структуру поддиректории Logic:

1. DeserialiseClasses — хранит основные десериализованные классы объектов MIF/MID файлов:
   - utils — содержит два вспомогательных класса:
     - Coord.py — класс координат объекта;
     - Layer.py — класс для хранения данных о слое;
   - Figure.py — родительский класс для всех десериализованных классов;
   - MultiFigure.py — родительский класс, для классов содержащих несколько фигур;
   - Arc.py — класс дуга;
   - Point.py — класс точка;

— `Line.py` — класс линия.
— и др.

2. `Readers` — содержит два класса `ReaderJson` и `ReaderMif` для считывания и парсинга `JSON` и `MIF/MID` файлов соответственно;
3. `Load` — содержит единственный файл `Load.py`, в котором описаны данные о грузе;
4. `SplitCheck` — содержит классы для работы с объектами станции:
   — `CheckRoad.py` — класс для проверки пересечения груза с объектами, прилегающими к железной дороге;
   — `Equations.py` — класс с уравнением прямой и уравнением эллипса;
   — `SplitStation.py` — класс для разбиения станции на прямоугольники с объектами.

### 2.5 Парсинг файлов MIF/MID

Для того, чтобы в приложении можно было работать с участком железнодорожного пути, необходимо реализовать считывание файлов форматов MIF и MID. Такие файлы хранят информацию о всех объектах участка пути. В каждом файле MIF и MID хранятся данные об одном слое участка. Файл MIF хранит данные, необходимые для вывода участка пути на экран, а MID хранит дополнительные параметры конкретного объекта.

Начнем с парсинга MIF-файла. Этот файл хранит набор примитивов, описанных в теоретической части. Для реализации считывания примитивов был разработан общий класс `Figure`, который хранит общие поля и методы всех объектов, необходимые для отрисовки их на сцене.

Для каждого объекта есть свой отдельный класс, т. к. каждый класс задаётся своим уникальным набором параметров. При считывании файла проводится проверка на то, какой объект считывается и управление передается соответствующему методу. Метод считывает все необходимые для данного объекта параметры и создаёт объект класса. Вся данная логика описана в файле `ReaderMif.py`.

### 2.6 Парсинг JSON-файла с данными о грузе

Для проверки пропуска груза по участку пути необходимы данные о грузе, полученные в результате работы первого модуля приложений. В первом модуле реализован экспорт этих данных в JSON. Во втором модуле эти данные

необходимо считать и учитывать при обработке.

Для решения данной задачи был создан класс `Load`, который предназначен для хранения информации о грузе. Он разработан специально в том формате, в котором у нас приходит JSON с данными о грузе, чтобы лаконично считать его сразу в этот класс и не создавать лишних обработок.

Само считывание JSON-файла происходит в файле `ReaderJson.py`.

В класс `ReaderJson.py` передаётся название файла. Метод `parse_json()` открывает этот файл и с помощью библиотеки `json` парсит файл в словарь, а далее сразу в объект класса `Load`.

### 2.7 Внешний вид приложения и работа пользователя с ним

В данном пункте рассмотрим интерфейс приложения и взаимодействие оператора с ним.

В левой части стартового окна находится `QGroupBox`, который содержит в себе элементы `QListWidget`, текстовые поля, кнопки и `Qlabel`. `QListWidget` содержит в себе слои станции.

Для того, чтобы посмотреть возможна ли провозка груза по участку пути, оператору требуются данные о грузе, полученные в первом модуле. Для загрузки данных о грузе существует элемент формы `QCommandLinkButton` *«Выбрать слой железных дорог»*. По нажатию на данную кнопку открывается диалоговое окно с возможностью выбора данных о грузе в формате `JSON`.

После того, как данные о грузе загружены в программу, в поля *«Индекс негабаритности на прямой»* и *«Индекс негабаритности на кривой»* будет выведена соответствующая информация. Также при помощи кнопки *«Остальные параметры груза»* можно просмотреть все параметры груза.

Рассмотрим элемент формы `QCommandLinkButton` *«Открыть данные о станции»*. При помощи данной кнопки можно загрузить данные о станции или участке пути.

При помощи элемента `QListWidget` мы имеем возможность включать и отключать слои на сцене. При отключении слоя, данный слой будет убран со сцены, и не будет учитываться при расчёте проходимости груза по пути.

По нажатию на кнопку *«Выбрать слой железных дорог»*, открывается окно. В нём выбирается слой железных дорог, по которым мы пропускаем проверяемый груз. Железнодорожные пути представляют собой либо линии, либо полилинии.

При нажатии на кнопку *«Определить проходимость груза»*, если данные о грузе и станции были загружены и выбран слой с железной дорогой, пути, по которым груз не может пройти, а также объекты, которые мешают пропуску груза, будут перекрашены в красный цвет.

Если не выбран слой железных дорог, или груз будет выведено сообщение об ошибке.

## 2.8 Логика работы

### 2.8.1 Разделение на районы

Основной задачей данной работы являлось решение задачи проверки прохождения груза по определённой железной дороге. Требовалось найти относительно оптимальный алгоритм нахождения пересечений объектов с крайней точкой груза. Поскольку объектов на сцене может быть достаточно много, требовалось ограничить область проверки и соответственно количество проверяемых объектов.

Было принято простое и очевидное, но одновременно с этим достаточно эффективное решение: разделить станцию на прямоугольники. Опишем алгоритм более формально:

1. Для начала решим, на сколько прямоугольников требуется разделить область в зависимости от масштаба;
2. Затем создадим словарь, где ключом будет являться кортеж из двух элементов, первый из которых обозначает номер района по горизонтали, а второй по вертикали. Значением будет являться список объектов входящих в данную область;
3. Чтобы определить входит ли объект в область, требуется проверить его координаты. Для каждого объекта алгоритм разный:
   - для линии проверяем пересечения по уравнению прямой с линиями границ областей, и если отрезок пересекает данную границу, то добавляем по соответствующему ключу отрезок в список фигур, содержащихся в данной области;
   - все полигональные объекты рассматриваются как набор линий, где каждая линия проверяется отдельно;
   - для нахождения эллипса в определённой области используем уравнение эллипса и ищем пересечения с линиями границ областей,

которые пересекает данный эллипс;

— для нахождения пересечений дуги используем случай с эллипсом, но дополнительно проверим, входят ли данные пересечения в границу углов дуги.

Теперь мы можем проверять только те объекты, которые находятся в одной области с крайней точкой груза.

### 2.8.2 Проверка прохождения груза по железной дороге

В предыдущем пункте мы рассмотрели алгоритм разделения объектов станции по определенным областям. Теперь нам требуется пропустить груз по каждому пути и проверить задевает ли груз данные объекты:

— построим линии, параллельные железной дороге, отдалённые от неё на нужное расстояние, но без вывода на сцене;

— используя вывод из пункта 1.5 проверим пересечение данных линий с объектами областей, в которые эти линии попали;

— если объект пересекается линией, то вносим его в перечень опасных, и раскрашиваем на сцене в красный цвет;

— для проверки точек используем проверку через барицентрические координаты.

После того, как будут проверены все пути, опасные участки и объекты будут перекрашены в красный цвет.

# ЗАКЛЮЧЕНИЕ

По итогу дипломной работы была достигнута цель, которая заключалась в доработке модуля *«Анализа и формирования данных о грузе»* приложения и разработке нового модуля *«Анализа проходимости груза на участке пути»*, предназначенного для автоматизации пропуска негабаритного груза по участку пути.

В ходе дипломной работы были решены следующие задачи:

- изучена литература, связанная с предметной областью;
- исследованы средства языка Python;
- исследован фреймворк PyQt5;
- внесены доработки в первый модуль приложения в соответствии с требованиями заказчика;
- реализовано считывание схем груза в новом формате;
- изучены средства ГИС;
- проанализированы способы автоматизации пропуска груза по участку пути;
- исследован необходимый математический аппарат для расчётов, необходимых во втором модуле приложения;
- реализован второй модуль приложения, предназначенный для пропуска негабаритных грузов по участку пути (станции).

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1 MapInfo Pro 2019 Руководство пользователя MapInfo Pro. — Stamford CT: Pitney Bowes, 2019. — 1560 с.

2 Документация по DXF [Электронный ресурс]. — URL: https://elima.ru/articles/?id=2 (Дата обращения 05.05.2021). Загл. с экр. Яз. рус.

3 *Harwani, B.* Qt5 Python GUI Programming Cookbook / B. Harwani. — Birmingham: Packt, 2018. — 462 pp.

4 *Шлее, М.* Qt 5.3. Профессиональное программирование на C++ / М. Шлее. — СПб: БХВ-Петербург, 2015. — 928 с.