МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Вятский государственный университет»**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Допущено к защите

Руководитель проекта

\_\_\_\_\_\_\_\_/Долженкова М.Л./

(подпись) (Ф.И.О)

«\_\_»\_\_\_\_\_\_\_\_\_\_2024г.

Разработка функционала для визуализации данных приложения «Радио - Трасса»

Пояснительная записка курсового проекта по дисциплине

«Комплекс знаний бакалавра в области программного и аппаратного обеспечения вычислительной техники»

ТПЖА.09.03.01.487 ПЗ

Разработал студент группы ИВТ-41 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Птахова А.М./

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М.Л./

Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М.Л./

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Долженкова М.Л. /

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Чистяков Г.А./

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Кошкин О.В./

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Скворцов А.А./

(подпись)

Киров 2024

РЕФЕРАТ

Птахова А.М.. Разработка функционала для визуализации данных приложения «Радио-Трасса». ТПЖА.09.03.01.487ПЗ: Курс. проект/ ВятГУ, каф. ЭВМ; рук. Долженкова М.Л. – Киров, 2024. - ПЗ 37с., 17 рис., 2 прил.

РАДИОТРАССА, ОБЛАСТЬ ПОКРЫТИЯ РАДИОТРАССЫ, 3D ВИЗУАЛИЗАЦИЯ, QT, C++.

Функционал, разработанный в рамках данного курсового проекта – расчет и построение области покрытия радиотрассы, 3D визуализация рельефа местности.

В ходе выполнения курсового проекта был выполнен анализ предметной области, проектирование и разработка программного обеспечения.

# ОГЛАВЛЕНИЕ

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум | Подпись | Дата |
| Разраб | | Птахова А. М |  |  | Разработка функционала для визуализации данных приложения «Радио-Трасса» | Литера | | | Лист | Листов |
| Пров | | Долженкова М. Л. |  |  |  |  |  | 3 | 37 |
|  | |  |  |  |  | | | | |
|  | |  |  |  |
| Реценз. | |  |  |  |

[ОГЛАВЛЕНИЕ 3](#_Toc163673818)

[ВВЕДЕНИЕ 5](#_Toc163673819)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc163673820)

[1.1 Техническое задание 6](#_Toc163673821)

[1.1.1 Наименование приложения 6](#_Toc163673822)

[1.1.2 Краткая характеристика области применения 6](#_Toc163673823)

[1.1.3 Цели добавления функционала 6](#_Toc163673824)

[1.1.4 Требования к новому функционалу 6](#_Toc163673825)

[1.3.5. Требования к программной документации 7](#_Toc163673826)

[1.3.6. Стадии и этапы разработки 7](#_Toc163673827)

[1.2 Расчет области покрытия радиотрассы 8](#_Toc163673828)

[2. АНАЛИЗ РАЗРАБОТАННЫХ КЛАССОВ ПРИЛОЖЕНИЯ 10](#_Toc163673829)

[2.1 Анализ разработанных классов 10](#_Toc163673830)

[2.1.1 Рассмотрение циркуляции потока данных 10](#_Toc163673831)

[2.1.2 Рассмотрение диаграммы классов 12](#_Toc163673832)

[2.2 Изменение классов приложения 13](#_Toc163673833)

[2.2.1 Рефакторинг 13](#_Toc163673834)

[2.2.2 Изменение алгоритма запроса тайла 15](#_Toc163673835)

[3. ОБЛАСТЬ ПОКРЫТИЯ РАДИОТРАССЫ 16](#_Toc163673836)

[3.1 Процесс построения области покрытия радиотрассы 16](#_Toc163673837)

[3.2 Разработка алгоритмов функционирования 17](#_Toc163673838)

[3.2.1 Алгоритм расчета опорных точек 17](#_Toc163673839)

[3.2.2 Определения типа трассы 19](#_Toc163673840)

[3.2.3 Закраска фрагмента области 21](#_Toc163673841)

[3.3 Программная реализация 24](#_Toc163673842)

[4. 3D ВИЗУАЛИЗАЦИЯ 25](#_Toc163673843)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 4 |
| Изм. | Лист | № докум | Подпись | Дата |

[4.1 Выбор инструментов для создания 3D модели 25](#_Toc163673844)

[4.2 Анализ классов модуля Qt Data Visualization 25](#_Toc163673845)

[4.3 Разработка собственных классов 27](#_Toc163673846)

[ЗАКЛЮЧЕНИЕ 30](#_Toc163673847)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 31](#_Toc163673848)

[Приложение А 32](#_Toc163673849)

[Приложение Б 35](#_Toc163673850)

# ВВЕДЕНИЕ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 5 |
| Изм. | Лист | № докум | Подпись | Дата |

В ходе выполнения предыдущего курсового проекта было разработано приложение для расчета и построения радиотрассы. Основным недостатком этого приложения была ограниченность предоставляемого функционала.

Эта ограниченность заключалось в том, что пользователь имел возможность построить радиотрассу только для координат, заранее заданных им. В случае если пользователь хотел бы определить область, в пределах которой он мог бы рассчитывать на хороший сигнал, то ему бы потребовалось выполнить многократный расчет радиотрассы, изменяя координаты приемника сигнала и не изменяя координаты источника сигнала. Другими словами, отсутствовала возможность автоматизированного расчета и отображения области покрытия радиотрассы.

Кроме того, не была реализована возможность создание 3D модели рельефа для области карты, задействованной при построении радиотрассы. Данная функция необходима для наглядности, позволяющей оценить правильность построения профиля высот, а также само построения радиотрассы.

Исходя из всего выше сказанного, целью данного курсового проекта является разработка и добавление нового функционала к существующему приложению, включающего в себя 3D визуализацию области, задействованной при построении радиотрассы, и создание области покрытия радиотрассы.

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 6 |
| Изм. | Лист | № докум | Подпись | Дата |

На первом этапе разработки было сформулировано техническое задание, в котором были более подробно описаны требования к новому функционалу. Кроме того, был выполнен разбор алгоритма расчета области покрытия радиотрассы, с целью получения сведений о том, какие данные необходимы для реализации данного алгоритма.

## 1.1 Техническое задание

### 1.1.1 Наименование приложения

Полное наименование приложения: приложения для расчета и построения радиотрассы

Сокращенное наименование приложения: «Радио-Трасса»

### 1.1.2 Краткая характеристика области применения

Приложение предназначено для получения информации о возможности создания канала связи между объектами.

### 1.1.3 Цели добавления функционала

Целью добавление нового функционала является предоставление пользователю возможности определения зоны обслуживания передающих станций радиосвязи, а также визуализации данных, полученных в ходе выполнения расчетов, в том числе и рельефа для обрабатываемой области.

Требования к конечному пользователю не предъявляются.

### 1.1.4 Требования к новому функционалу

Функционал, реализующая расчет и отображение области покрытия радиотрассы, должна обеспечивать возможность выполнения следующих функций:

1) Выбора/ввода только одной координаты – точки источника;

2) Задание длины радиуса для области покрытия;

3) Расчет и отображение области покрытия радиотрассы.

При 3D визуализации рельефа области функция должна реализовывать построение 3D модели на основе имеющихся значений высот для данной области.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 7 |
| Изм. | Лист | № докум | Подпись | Дата |

В состав технических средств должен входить ПК, включающий в себя:

1) 32 или 64 разрядный процессор с тактовой частотой не меньше 1.0

ГГц;

2) дисплей;

3) не менее 1 Гб оперативной памяти;

4) не менее 100 мегабайт свободного дискового пространства;

5) клавиатура и мышь.

Системные программные средства, используемые программой, должны быть представлены ОС Windows 7/8/10;

Программа должна обеспечивать взаимодействие с пользователем по средствам графического интерфейса и предоставлять возможность выполнять наиболее часто используемые операции с помощью сочетаний клавиш на клавиатуре.

### 1.3.5. Требования к программной документации

Состав программной документации должен включать в себя:

1) техническое задание;

2) руководство пользователя;

3) исходный код.

### 1.3.6. Стадии и этапы разработки

Разработка должна быть приведена в 3 стадии:

1) разработка технического задания;

2) проектирование;

3) внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и разработки технического задания.

На стадии проектирования необходимо выполнить следующие этапы:

1) разработка программы;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 8 |
| Изм. | Лист | № докум | Подпись | Дата |

2) разработка программной документации;

3) испытание программы.

На этапе внедрения выполняется передача программы заказчику.

## 1.2 Расчет области покрытия радиотрассы

Алгоритм расчета области покрытия радиотрассы следующий:

1) выбирается точка – источник сигнала;

2) задается радиус области покрытия;

3) вычисляются точки, лежащие на окружности, образуемой областью покрытия;

4) полученные точки разбивают область видимости на фрагменты. Пример разбиения области видимости на фрагменты представлен на рисунке 1.1.

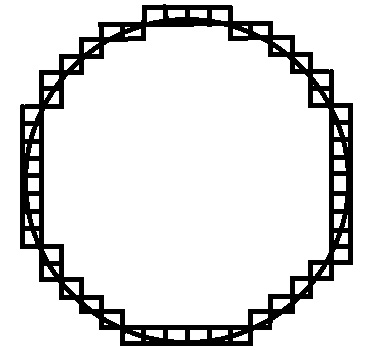


Рисунок 1.1 – Разбиение области видимости на фрагменты

5) для каждой вычисленной точкой и источником сигнала строится радиотрасса;

6) для каждой построенной радиотрассы определяется её тип: открытая, полуоткрытая, полузакрытая, закрытая;

7) каждый фрагмент области видимости закрашивается в цвет, соответствующий типу трассы;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 9 |
| Изм. | Лист | № докум | Подпись | Дата |

8) уменьшается значение радиуса на заданный шаг;

9) повтор п. 3) – 8) до тех пор, пока значение радиуса не станет равным нулю.

На основании рассмотренного алгоритма для расчета области покрытия радиотрассы был сделан вывод, что данная задача сводится к многократному решению задачи о построении радиотрассы между 2-мя точками. Следовательно, для ее реализации необходимо наличие следующих данных:

1) Координаты 2х точек – источника и приемника;

2) Массив точек, принадлежащих маршруту радиотрассы;

3) Массив тайлов для определения значения высот для каждой точки маршрута;

4) Массив высот, описывающих профиль рельефа;

5) Массив высот для линии сигнала;

6) Массив значений для первой зоны Френеля.

7) Массив тайлов и массив высот также необходим и для выполнения 3D визуализации рельефа местности.

Следовательно, для добавления нового функционала к существующему приложению «Радио-Трасса», состоящего из визуализации области покрытия радиотрассы и рельефа самой местности, необходимо наличие определенных данных, которые были описаны выше.

# 2. АНАЛИЗ РАЗРАБОТАННЫХ КЛАССОВ ПРИЛОЖЕНИЯ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 10 |
| Изм. | Лист | № докум | Подпись | Дата |

На текущем этапе разработки был выполнен анализ классов приложения, включающий в себя построение циркуляции потоков данных внутри приложения и его диаграмму классов. В результате был сделан вывод о возможности получения некоторого набора данных, необходимых для реализации нового функционала. Также был произведен рефакторинг и изменение алгоритма запроса тайла с целью уменьшения времени работы последнего.

## 2.1 Анализ разработанных классов

### 2.1.1 Рассмотрение циркуляции потока данных

Как уже было сказано ранее, анализ подразумевает под собой построение движения данных внутри приложения. На рисунке 2.1 представлена циркуляция данных в приложении при расчете профиля высот.

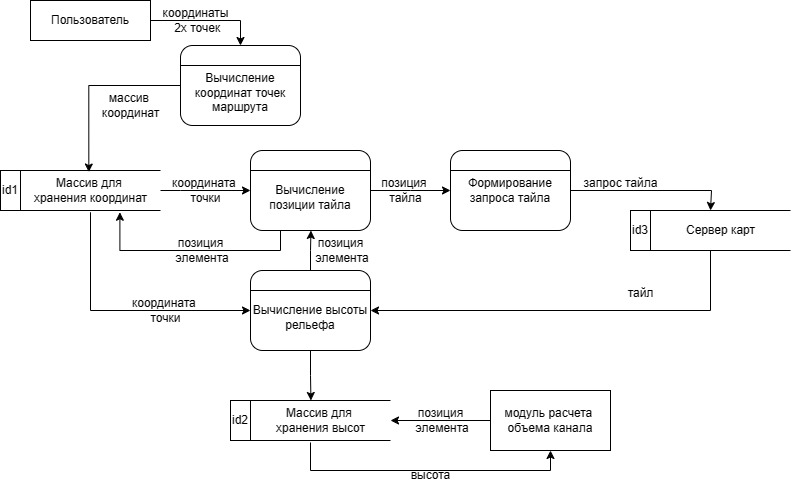


Рисунок 2.1 – Циркуляция потоков данных при расчете профиля высот

На рисунке можно заметить, что в приложение уже существуют массивы для хранения высот и координат точек маршрута. Значит, есть возможность получить эти данные для их последующего использования в новых функциях.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 11 |
| Изм. | Лист | № докум | Подпись | Дата |

На рисунке 2.2 представлена циркуляция потоков данных в приложении при вычислении объема радиоволнового канала между объектами.

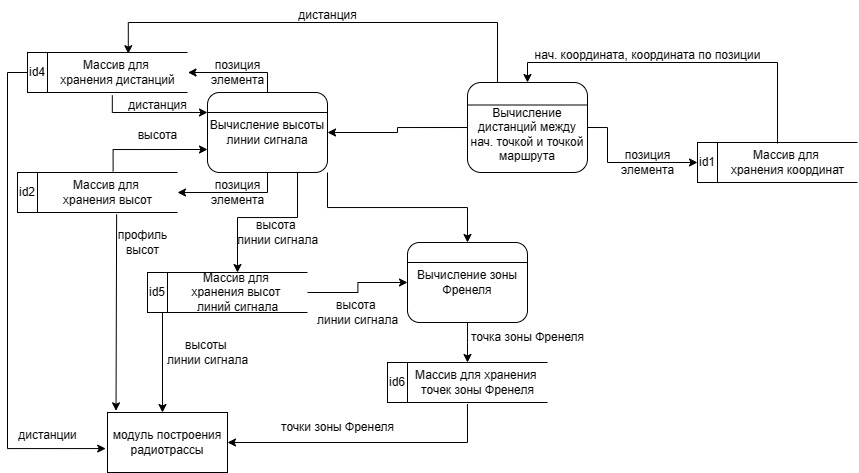


Рисунок 2.2 - циркуляция потоков данных в приложении при вычислении объема радиоволнового канала между объектами

На рисунке 2.2 видно, что в приложении существует возможность получения данных о высоте линии сигнала и точках зоны Френеля, так как для них реализованы массивы.

В результате анализа циркуляции потока данных внутри приложения был сделан вывод о возможности получения практически всех необходимых данных для реализации нового функционала, за исключением массива тайлов, участвовавших при расчетах.

Недостаток, который был выявлен при анализе, - неэффективность алгоритма, выполняющего расчет профиля высот. Как видно из рисунка 2.1 вычисление позиции тайла и его запрос происходит для каждой точки маршрута радиотрассы. В результате происходит многократный запрос одинаковых тайлов. Такая последовательность действий приведет к неэффективности всего алгоритма расчета профиля высот, заключающаяся в увеличении времени его выполнения, для большого набора обрабатываемых данных. Следовательно, необходимо изменить последовательность действий, уменьшив количество запросов одного тайла.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 12 |
| Изм. | Лист | № докум | Подпись | Дата |

Для того, чтобы изменить последовательность действий при расчете профиля высот, необходимо рассмотреть диаграмму классов текущего приложения, необходимую для определения зависимостей между классами.

### 2.1.2 Рассмотрение диаграммы классов

Диаграмма классов существующего приложения представлена на рисунке 2.3.

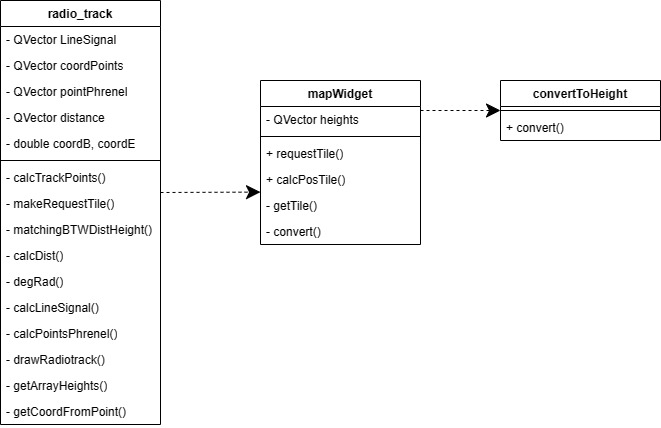


Рисунок 2.3 – Диаграмма классов существующего приложения

Как видно из рисунка внутри классов реализованы методы, выполняющие абсолютно разные функции. Такой подход к проектированию ухудшает дальнейшую масштабируемость приложения. Кроме того, классы имеют высокую степень связности. Такой вывод был сделан на основании циркуляции потоков данных, представленных на рисунке 2.1, и диаграмме классов, представленной на рисунке 2.3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 13 |
| Изм. | Лист | № докум | Подпись | Дата |

В результате выполненного анализа классов приложения был сделан вывод о необходимости выполнения рефакторинга с целью достижения низкой связности и высокого сцепления.

## 2.2 Изменение классов приложения

### 2.2.1 Рефакторинг

Диаграмма классов после рефакторинга представлена на рисунке 2.4.

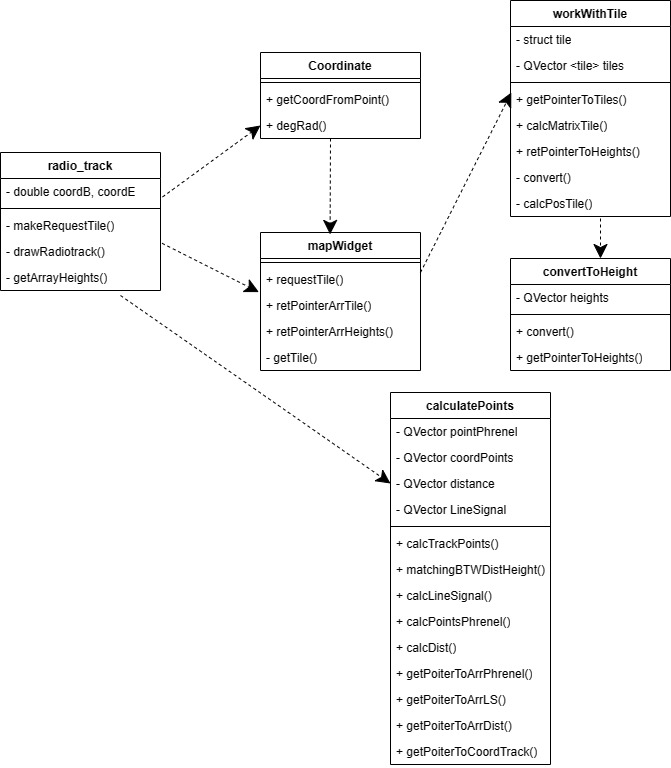


Рисунок 2.4 – Диаграмма классов после рефакторинга

Как видно из рисунка, все методы, а также переменные для хранения данных, относящихся к расчетам различных точек, вынесены в отдельный класс – calculatePoint. Всё что касается работы с координатами: выделение градусов, минут и секунд из дробного значения; перевод градусы в радианы и тд, также вынесены в отдельный класс. Вся работа с тайлами тоже вынесена в отдельный класс. Помимо методов для работы с тайлами в класс была добавлена структура tile, имеющая следующие описание:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 14 |
| Изм. | Лист | № докум | Подпись | Дата |

struct tile

{

int pos\_x;

int pos\_y;

QPixmap pix\_map:

};

Более подробно стоит рассмотреть функцию calcMatrixTile, выполняющую расчет массива тайлов. Листинг кода, содержащий описание функции приведен ниже:

void workWithTile::calcMatrixTiles(const QPointF& coord\_begin,

const QPointF& coord\_end, int zoom)

{

// перевод географических координат точки в позицию тайла

QPoint coord\_tile\_begin = convertCoordDisplayToPosTile(coord\_begin, zoom);

QPoint coord\_tile\_end = convertCoordDisplayToPosTile(coord\_end, zoom);

//нахождение минимальной позиции тайла

findMinCoord(coord\_tile\_begin, coord\_tile\_end);

//перебор всех позиций тайлов в пределах вычисленной

for (auto i = coord\_tile\_begin.x(); i <= coord\_tile\_end.x(); i++)

{

for (auto j = coord\_tile\_begin.y(); j <= coord\_tile\_end.y(); j++)

{

tile new\_tile;

new\_tile.pos\_x = i;

new\_tile.pos\_y = j;

saveTile(new\_tile); }}}

### 2.2.2 Изменение алгоритма запроса тайла

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 15 |
| Изм. | Лист | № докум | Подпись | Дата |

Был доработан алгоритм запроса тайлов для построения профиля высот. Циркуляция потоков данных с учетом нового алгоритма представлена на рисунке 2.5.

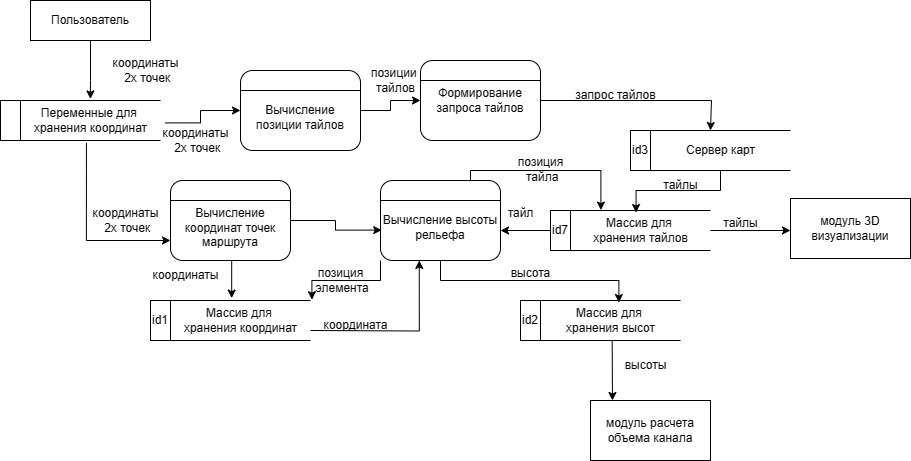


Рисунок 2.5 - Циркуляция потоков данных с учетом нового алгоритма при расчете профиля высот

На рисунке показано, что вычисление позиции тайлов, используемых при расчетах радиотрассы, происходит сразу после того, как пользователь ввел координаты 2х точек. Далее выполняется обращение к серверу карт для запроса всех тайлов. После их получения, они сохраняются в массив.

Когда все тайлы получены и сохранены, выполняется расчет профиля высот. Первоначально происходит вычисление точек маршрута радиотрассы, при расчете которых используются данные 2х точек, которые изначально ввел пользователь. Полученные значения координат точек маршрута записываются в массив. Далее идет вычисление высоты рельефа для каждой из точек. Для этого для каждой точки определяется позиция тайла, выполняется поиск в массиве тайлов элемента с такими значениями, определяется координаты пикселя внутри тайла и его цвет, полученное значение цвета преобразовывается в значение высоты и записывается в массив высот.

В результате выполнения данного этапа разработки было получено приложение, содержащее все необходимые для разработки данные, а также отличающееся временем выполнения расчета профиля высот.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 16 |
| Изм. | Лист | № докум | Подпись | Дата |

# 3. ОБЛАСТЬ ПОКРЫТИЯ РАДИОТРАССЫ

На данном этапе работ был рассмотрен процесс построения области покрытия радиотрассы. На его основе были сделаны выводы о том, какие функции должен содержать в себе этот процесс. Далее при помощи диаграммы классов приложения были выявлены функции, которые уже разработаны и не требуют никакой доработки. Для оставшихся функций были разработаны и програмно реализованы алгоритмы работы.

## 3.1 Процесс построения области покрытия радиотрассы

Процесс построения области покрытия радиотрассы представлен на рисунке 3.1.

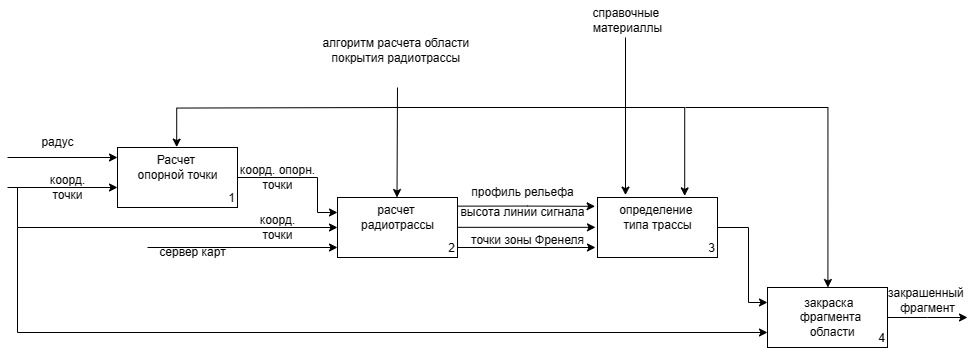


Рисунок 3.1 – Процесс построения области покрытия радиотрассы

Из рисунка видно, что процесс построения области покрытия состоит из 4х основных этапов:

1) расчет опорной точки;

2) расчет радиотрассы между опорной точкой и заданной точкой (источник сигнала);

3) определение типа трассы согласно справочным материалам;

4) закраска фрагмента области покрытия в цвет, соответствующий типу трассы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 17 |
| Изм. | Лист | № докум | Подпись | Дата |

Дополнительно стоит отметить, что этап закраски фрагмента области подразумевает не только закраску фрагмента, но и сам расчет этого фрагмента, заключающегося в расчете координат верхнего левого и нижнего правого краев.

Расчет радиотрассы уже был реализован в приложении. Следовательно, все необходимые функции для выполнения расчета уже были разработаны. В доработке эти функции не нуждаются, так как это уже было сделано на первом этапе работ.

А вот оставшиеся функции никогда до этого не применялись в приложении, поэтому необходимо выполнить их разработку.

## 3.2 Разработка алгоритмов функционирования

### 3.2.1 Алгоритм расчета опорных точек

Расчет опорных точек должен производится на основании двух параметров: координаты точки источника сигнала и радиуса области покрытия радиотрассы. Дополнительным параметром можно использовать угол поворота радиуса относительно центральной точки. Других параметров нет.

Можно предположить, что расчет опорных точек можно выполнить при помощи следующих формул:

где

– координаты центра области покрытия,

– радиус области покрытия,

– угол поворота радиуса относительно центра

Но данная формула применима только для плоскости, потому что расчет выполняется по длине отрезка, а не по длине дуги, как это необходимо при работе с географическими координатами. В результате неправильного расчета область видимости приобретает форму овала, что является неверным. На рисунке 3.2 изображена граница области видимости при использовании формулы для плоскости.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 18 |
| Изм. | Лист | № докум | Подпись | Дата |

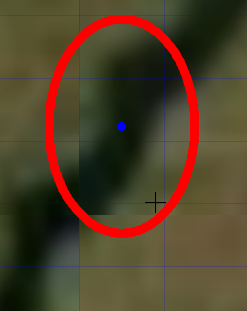


Рисунок 3.2 – Границы области видимости при использовании формулы для плоскости

Как уже было отмечено ранее, правильный расчет возможен в том случае, если он выполняется по длине дуги. Следовательно, необходимо найти формулу для расчета точки через длину дуги.

Эту формулу можно получить из сферической теоремы косинусов. Так как сферическая теорема косинусов позволяет найти длину дуги между точками, находящимися на сфере, то для получения точки через длину дуги, необходимо решить обратную задачу, воспользовавшись следствием из этой теоремы. Формула для расчета координаты точки следующая:

φ₂ = arcsin(sin(φ₁) cos(d/R) + cos(φ₁) sin(d/R) cos(α))

λ₂ = λ₁ + atan2(sin(α) sin(d/R) cos(φ₁), cos(d/R) - sin(φ₁) sin(φ₂)), где

φ₁, λ₁ - широта и долгота источника сигнала,

φ₂, λ₂ - широта и долгота приемника сигнала,

α - направление от точки A до точки B (азимут),

R - радиус сферы,

d - длина отрезка по ортодомии.

Граница области видимости при использовании формулы для сферической поверхности представлена на рисунке 3.3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 19 |
| Изм. | Лист | № докум | Подпись | Дата |

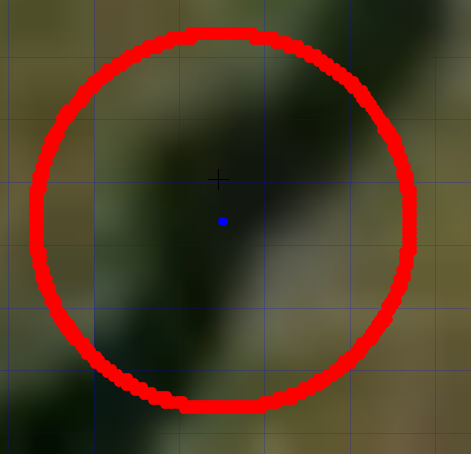


Рисунок 3.3 - Граница области видимости при использовании формулы для сферической поверхности

### 3.2.2 Определения типа трассы

Cуществует 4 типа трассы: открытая, полуоткрытая, полузакрытая и закрытая. Тип определяется согласно тому, как профиль высот пересекает область сигнала вместе с зоной Френеля:

1) если профиль выше верхней границы зоны Френеля – закрытая;

2) если профиль выше линии сигнала, но не пересекает верхнюю границу зоны Френеля – полузакрытая;

3) если профиль ниже линии сигнала и не пересекает нижнюю границу зоны Френеля – полуоткрытая;

4) если профиль ниже верхней границы зоны Френеля – открытая;

Весь процесс определения типа изображен на схеме алгоритма, представленной на рисунке 3.4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 20 |
| Изм. | Лист | № докум | Подпись | Дата |

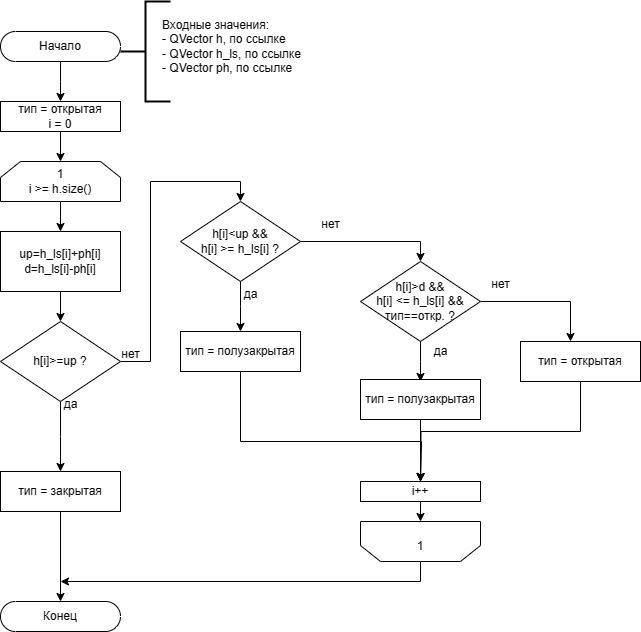


Рисунок 3.4 – Схема алгоритма определения типа трассы

В качестве входных параметров передаются массивы профиля высот, высот линии сигнала и точек зоны Френеля. Далее происходит определение нижней и верхней границы зоны Френеля для каждой точки профиля высот. Для определения верхней границы производится сложение значение высоты линии сигнала со значением расстояния зоны Френеля в этой точке, для нижней – вычитание. Далее происходит сравнение высоты профиля и верхней границы зоны Френеля. Если профиль отказывается выше, то трассе присваивается закрытый тип и алгоритм заканчивает свою работу. Если же нет, не пересекает, то выполняется следующая проверка. И так до тех пор, пока тип трассы не станет закрытым или пока не будут рассмотрен весь профиль высот.

### 3.2.3 Закраска фрагмента области

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 21 |
| Изм. | Лист | № докум | Подпись | Дата |

Ранее уже было сказано, что закраска фрагмента области подразумевает не только саму закраску фрагмента, но и первоначальный расчет координат границ этого фрагмента.

Так как закраска фрагмента связана с определением типа трассы, определение которого в свою очередь связано с расчетом радиотрассы. А расчет радиотрассы включает в себя определение точек маршрута, то можно предположить, что границы фрагмента можно получать из координат последней и предпоследней точек маршрута радиотрассы. Пример получения границ фрагмента представлен на рисунке 3.5

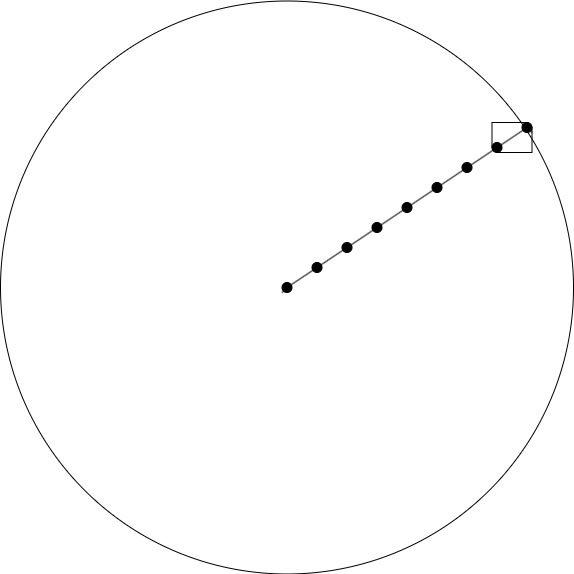
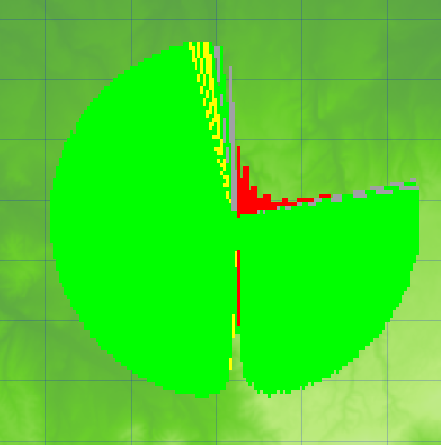


Рисунок 3.5 – Пример получения границ фрагмента

Результат применения данного алгоритма для расчета области покрытия в радиусе 10 км представлен на рисунке 3.6.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 22 |
| Изм. | Лист | № докум | Подпись | Дата |

Рисунок 3.6 – Результат применения алгоритма для области в радиусе 10 км

Из рисунка видно, что при небольшом значении радиуса область закрашивается равномерно. Но при увеличении значения радиуса область уже начинает закрашиваться неравномерно: появляются просветы. Пример закраски области с просветами при радиусе в 40 км представлен на рисунке 3.7

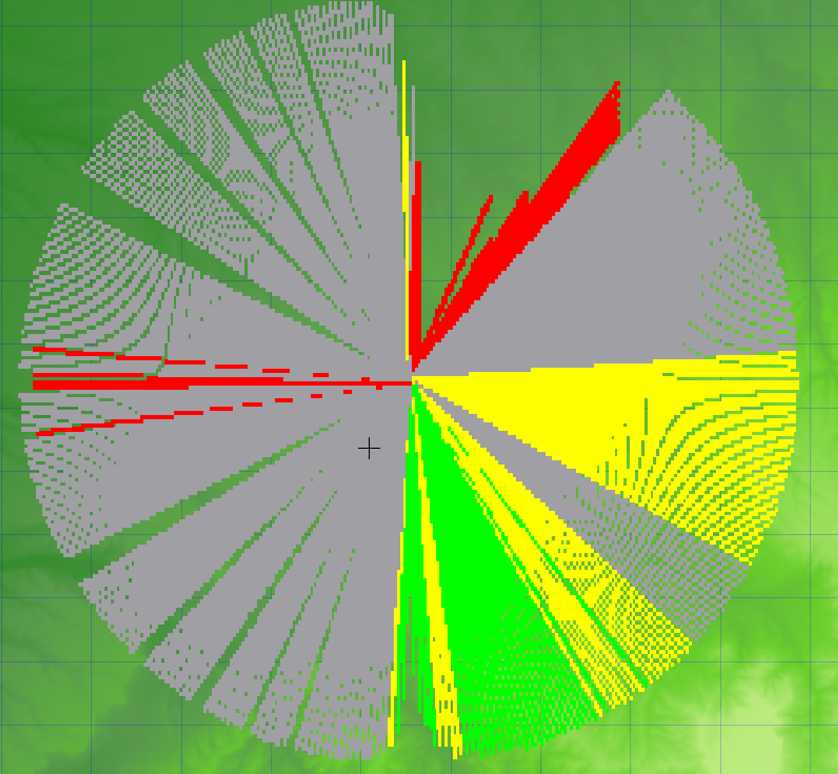


Рисунок 3.7 - Пример закраски области с просветами при радиусе в 40 км

Появление просветов связано с тем, что вычисленные опорные точки и точки маршрута для этих точек разбивают область покрытия на неодинаковые фрагменты. Для решения проблемы с появлением просветов при закраске, необходимо выполнить разбиение области на одинаковые фрагменты. Следовательно, предложенные первоначально алгоритм не подходит и его требует изменить.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 23 |
| Изм. | Лист | № докум | Подпись | Дата |

Тогда алгоритм разбиения области на одинаковые фрагменты следующий:

1) вычисление координат самих крайних опорных точек;

2) определение минимальных и максимальных координат среди координат крайних точек;

3) задание шага;

4) проход по циклу от минимальных до максимальных значений координат с заданным шагом и получение координат границ фрагмента.

Расчет радиотрассы и определения ее типа происходит в момент, когда происходит расчет границ фрагмента. Результат работы алгоритма при разделении области на одинаковые фрагменты приведен на рисунке 3.8

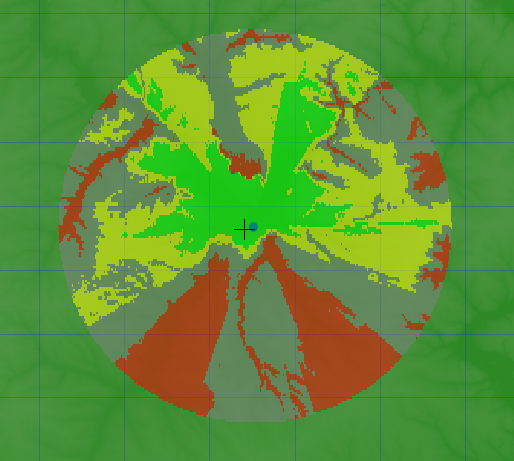


Рисунок 3.8 - Результат работы алгоритма при разделении области на одинаковые фрагменты

## 3.3 Программная реализация

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 24 |
| Изм. | Лист | № докум | Подпись | Дата |

Расчет области покрытия радиотрассы является ничем иным, как многократным расчетом радиотрассы для разных координат точек приемника. В уже доработанном алгоритме расчета радиотрассы запрос тайлов для последующего построения профиля высот используется вычисление массива тайлов для маршрута радиотрассы. Такой подход приведет к дублированию тайлов при вычислении области покрытия, поэтому стоит сделать перегрузку уже имеющейся функции для расчета массива тайлов, изменив набор параметров.

Код приведен ниже:

void workWithTile::calcMatrixTiles(const QPointF& coord\_right,

const QPointF& coord\_up,

const QPointF& coord\_left,

const QPointF& coord\_down, int zoom)

{

QPoint coord\_tile\_left = convertCoordDisplayToPosTile(coord\_left, zoom);

QPoint coord\_tile\_right = convertCoordDisplayToPosTile(coord\_right, zoom);

QPoint coord\_tile\_up = convertCoordDisplayToPosTile(coord\_up, zoom);

QPoint coord\_tile\_down = convertCoordDisplayToPosTile(coord\_down, zoom);

findMinCoord(coord\_tile\_left, coord\_tile\_up);

findMinCoord(coord\_tile\_right, coord\_tile\_down);

for (int i = coord\_tile\_left.x(); i <= coord\_tile\_down.x(); i++)

{

for (int j = coord\_tile\_left.y(); j <= coord\_tile\_down.y(); j++)

{

tile new\_tile;

new\_tile.pos\_x = i;

new\_tile.pos\_y = j;

saveTile(new\_tile);

}

}}

Был добавлен новые классы, содержащий в себе методы для расчета и отрисовки области покрытия радиотрассы на карте. Листинг кода приведен в приложении А.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 25 |
| Изм. | Лист | № докум | Подпись | Дата |

# 4. 3D ВИЗУАЛИЗАЦИЯ

На последнем этапе работ был выполнен выбор инструментов для создания 3D модели. Также был произведен анализ классов, содержащихся в модуле Qt Data Visualization, при помощи которого были разработаны собственные классы, выполняющие 3D визуализацию.

## 4.1 Выбор инструментов для создания 3D модели

Все реализованные ранее части приложения разрабатывалась под Qt. И было бы намного удобнее реализовать новый метод также с использованием Qt. Удобство использование этого фреймворка обусловлена удобством и простой дальнейшей поддержки этого проекта.

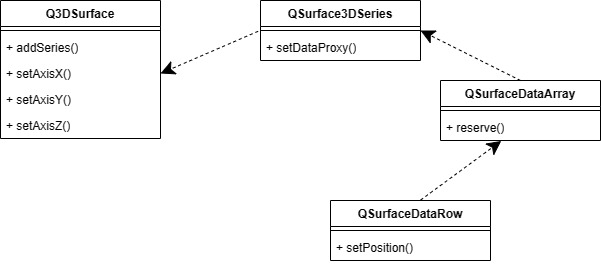
Задача построения 3D модели для какого-то набора данных является довольно распространенной задачей. Следовательно, во фреймворке Qt должны быть уже реализованы модули, позволяющие выполнять 3D визуализацию.

Фреймворк, действительно, содержит 2 модуля, позволяющие выполнить задачу: Qt Data Visualization, Qt OpenGL

Что касается последнего, то с Qt 5 данный модуль считается устаревшим, а со следующей версии он был убран. Так как разрабатываемое приложение планируется поддерживать и дальше, то использование этого модуля для разработки является нецелесообразным.

## 4.2 Анализ классов модуля Qt Data Visualization

Анализ классов для данного модуля производился при помощи рассмотрения диаграммы классов, представленной на рисунке 4.1



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 26 |
| Изм. | Лист | № докум | Подпись | Дата |

Рисунок 4.1 – Диаграмма классов модуля Qt Data Visualization

В модуле Data Visualization для работы с 3D графикой разработан класс Q3DSurface, его основной задачей является рисование 3х-мерной модели вместе с системой координат. Для задания системы координат используются методы setAxisX, setAxisY, setAxisZ. Для рисования 3D модели используется метод addSeries.

Для отрисовки данных их необходимо сформировать и передать обратно в класс Q3DSurface. Для этой передачи данных был реализован класс QSurface3DSeries, содержащий метод setDataProxy. А для формирования данных используется класс QSurfaceDataArray.

Так как любая 3х-мерная модель состоит из набора 3х-мерных данных, то в модуле был разработан класс QSurfaceDataArray, который сохраняет весь набор данных в 3х-мерный массив для дальнейшего использования. Кроме того, был разработан класс QSurfaceDataRow для создания 3х мерной точки, которая впоследствии записывается в 3х-мерный массив.

## 4.3 Разработка собственных классов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 27 |
| Изм. | Лист | № докум | Подпись | Дата |

Диаграмма разработанных классов с использованием классов модуля представлена на рисунке 4.2

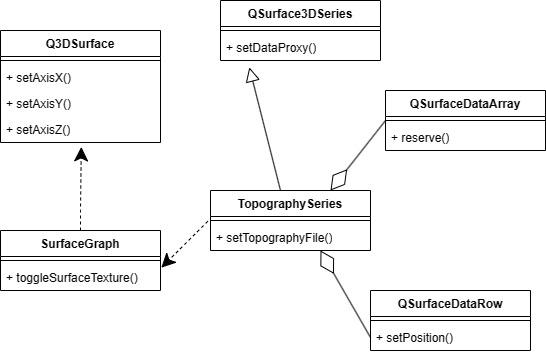


Рисунок 4.2 - Диаграмма разработанных классов с использованием классов модуля

Данные для 3х-мерной модели формируются за счет преобразования каждого тайла, используемого при расчетах, в 3х-мерный массив. То есть координаты пикселя вместе с номером тайла в массиве преобразовываются в координаты X, Z. А значение цвета пикселя, конвертируется в высоту и будет являться координатой Y.

Для формирования данных был разработан класс TopographySeries, который является наследником от класса QSurface3DSeries и в котором добавлен метод setTopographyFile, который и занимается получением данных из тайлов.

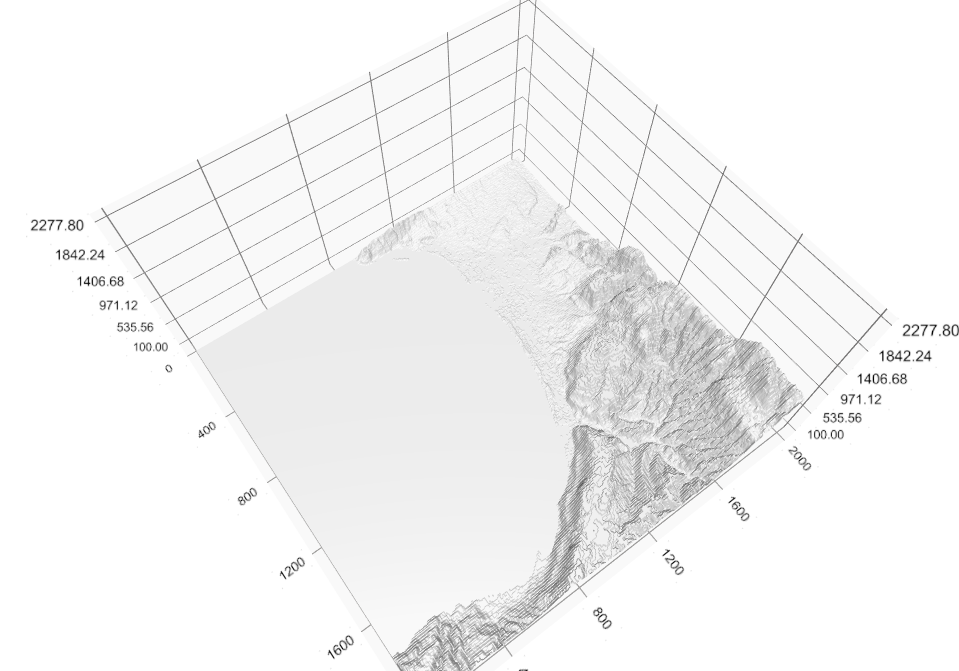
Для создания 3х-мерной точки используется класс QSurfaceDataRow, экземпляр которого создается внутри TopographySeries и в который записывается значение полученная в ходе конвертации значений точка. А для сохранения этой точки в массив используется QSurfaceDataArray, экземпляр которого создается внутри TopographySeries.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 28 |
| Изм. | Лист | № докум | Подпись | Дата |

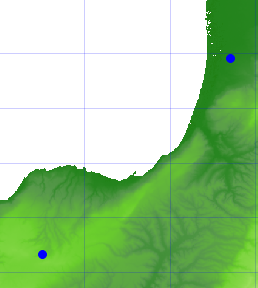
Далее набор данных передается для отрисовки, но так как QSurface3DSeries был изменен через создание наследника - нового класса, то вместо Q3DSurface был разработан новый класс SurfaceGraph, который вызывает методы TopographySeries для получения и формирования данных для 3D модели. Создание координатных осей остается без изменений, поэтому методы для их создания вызываются из экземпляра класса Q3DSurface.

Листинг кода приведен в приложении Б.

Результат построения 3D модели рельефа для области представлен на рисунке 4.3.



а)



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 29 |
| Изм. | Лист | № докум | Подпись | Дата |

Б)

Рисунок 4.3 а) - Результат построения 3D модели рельефа для области, б) – область, для которой выполнялось 3D моделирование

# ЗАКЛЮЧЕНИЕ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 30 |
| Изм. | Лист | № докум | Подпись | Дата |

В ходе выполнения данного курсового проекта была выполнена разработка нового функционала для визуализации данных приложения «Радио-Трасса», включающего в себя расчет и отображение области покрытия радиотрассы, а также построение 3х-мерной модели рельефа местности, участвующей в расчетах радиотрассы.

Во время разработки нового функционала был изменен алгоритм для расчета профиля высот, что позволило уменьшить время выполнения расчета на 0.2 секунды. Это повлияло на время выполнения расчета области покрытия радиотрассы: вместо 2 мин расчет стал занимать 30 секунд.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 31 |
| Изм. | Лист | № докум | Подпись | Дата |

1. Документация по C++ [Электронный курс] – режим доступа [https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-160](https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-160%20)

2. ГОСТ «Отчет по НИР» [Электронный курс] – режим доступа <https://www.tsu.ru/upload/medialibrary/235/gost_7.32_2017.pdf>

3. Документация по Qt [Электронный курс] – режим доступа

<https://doc.qt.io/all-topics.html>

4. Дьяков Б.Н. Геодезия. Общий курс: Учеб. пособие для вузов. - Новосибирск: Изд-во Новосиб ун-та, 1993.- 171 С.

5. Определение типа радиотрассы [Электронный курс] - режим доступа <https://habr.com/ru/articles/257731/>

# Приложение А

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 32 |
| Изм. | Лист | № докум | Подпись | Дата |

(обязательное)

**Листинг кода**

Calc\_coverage.h

#include "qpoint.h"

//#include <qvector.h>

#include <earth\_point.hpp>

#include "commonType.h"

namespace track

{

class visiable\_coverage

{

public:

visiable\_coverage() = default;

~visiable\_coverage();

void clearAll();

void makeMapWeight(QVector<QPointF>heights, QVector<QPointF> coord\_track,

QVector<QPointF> heights\_signal,

const QVector<QPointF>& point\_phrenel, QPointF coord);

weight& getMapWeight(int n);

coordCoverageVisible\_& getVisible();

void start();

private:

int number = 0;

test test\_;

weight mapWeight\_;

coordCoverageVisible\_ visible;

};

Calc\_coverage.cpp

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 33 |
| Изм. | Лист | № докум | Подпись | Дата |

#include "calc\_coverage.h"

#include "qmath.h"

#include <fstream>

using namespace track;

void visiable\_coverage::start()

{

number = 0;

test\_.clear();

}

void visiable\_coverage::clearAll()

{

mapWeight\_.clear();

visible.clear();

}

weight& visiable\_coverage::getMapWeight(int n)

{

return test\_.find(n)->second;

}

coordCoverageVisible\_& visiable\_coverage::getVisible()

{

return visible;

}

void visiable\_coverage::makeMapWeight(QVector<QPointF> heights,

QVector<QPointF> coord\_track,

QVector<QPointF> heights\_signal,

const QVector<QPointF>& point\_phrenel, QPointF coord)

{

int type\_track = 4;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 34 |
| Изм. | Лист | № докум | Подпись | Дата |

for (int i = 0; i < coord\_track.size(); i++)

{

if (heights[i].y() > heights\_signal[i].y() + point\_phrenel[i].y())

{

type\_track = 1;

break;

}

if (heights[i].y() <= heights\_signal[i].y() + point\_phrenel[i].y()

&& heights[i].y() > heights\_signal[i].y())

{

type\_track = 2;

}

if (heights[i].y() <= heights\_signal[i].y()

&& heights[i].y() > heights\_signal[i].y() - point\_phrenel[i].y())

{

type\_track < 3 ? type\_track = type\_track : type\_track = 3;

}

if (heights[i].y() <= heights\_signal[i].y() - point\_phrenel[i].y())

{

type\_track < 4 ? type\_track = type\_track : type\_track = 4;

}

}

mapWeight\_.insert({ type\_track, coord });

test\_.insert({ number, mapWeight\_ });

number++;

}

visiable\_coverage::~visiable\_coverage() {}

# Приложение Б

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 35 |
| Изм. | Лист | № докум | Подпись | Дата |

(Обязательное)

**Листинг кода**

#include <QtDataVisualization/QSurface3DSeries>

#include "color\_height.h"

#include "calculateTile.h"

using namespace QtDataVisualization;

using namespace track;

class TopographicSeries : public QSurface3DSeries

{

Q\_OBJECT

public:

explicit TopographicSeries();

~TopographicSeries();

void setTopographyFile(QVector <tile>& tiles\_, float width, float height);

float sampleCountX() { return m\_sampleCountX; }

float sampleCountZ() { return m\_sampleCountZ; }

public Q\_SLOTS:

private:

float m\_sampleCountX;

float m\_sampleCountZ;

converter\_toHeight\* converter;

};

#include "topographyseries.h"

#include <fstream>

#include <math.h>

using namespace QtDataVisualization;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 36 |
| Изм. | Лист | № докум | Подпись | Дата |

//! [0]

// Value used to encode height data as RGB value on PNG file

const float packingFactor = 11983.0f;

//! [0]

TopographicSeries::TopographicSeries()

{

converter = new converter\_toHeight();

setDrawMode(QSurface3DSeries::DrawSurface);

setFlatShadingEnabled(true);

setBaseColor(Qt::white);

}

TopographicSeries::~TopographicSeries()

{

}

void TopographicSeries::setTopographyFile(QVector <tile>& tiles, float width, float height)

{

int diff\_x = tiles[tiles.size() - 1].pos\_x - tiles[0].pos\_x + 1;

int diff\_y = tiles[tiles.size() - 1].pos\_y - tiles[0].pos\_y + 1;

int imageHeight = 256 \* diff\_x;

int imageWidth= 256 \* diff\_y;

double stepZ = 4;

double stepX = 4;

QSurfaceDataArray\* dataArray = new QSurfaceDataArray;

dataArray->reserve(imageHeight);

for (int i = 0; i < imageHeight; i++) {

float z = float(i)\*stepZ;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ТПЖА.09.03.01.487 ПЗ | Лист |
|  |  |  |  |  | 37 |
| Изм. | Лист | № докум | Подпись | Дата |

QSurfaceDataRow\* newRow = new QSurfaceDataRow(imageWidth);

for (int j = 0; j < imageWidth; j++) {

int pos\_x = i / 256;

int pos\_y = j / 256;

int number\_tile = pos\_x\*diff\_y+pos\_y;

QImage heightMapImage(tiles[number\_tile].pix\_map.toImage());

uchar\* bits = heightMapImage.bits();

QColor color(heightMapImage.pixel(i%256, j%256));

int red = color.red();

int green = color.green();

int blue = color.blue();

int height = converter->convert(red, green, blue);

float y = height;

(\*newRow)[j].setPosition(QVector3D(float(j) \* stepX, y, z));

}

\*dataArray << newRow;

}

dataProxy()->resetArray(dataArray);

m\_sampleCountX = float(imageWidth);

m\_sampleCountZ = float(imageHeight);

}