Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра Телекоммуникационных систем и вычислительных средств (ТСиВС)

9.03.01 Программное обеспечение
мобильных систем
(очная форма обучения)

КУРСОВАЯ РАБОТА

по дисциплине «Визуальное программирование»

Выполнил:

студент ИВТ,

гр. ИА-132 _____/ Бугаев В.М./

«__» _____ 2023 г. (подпись)

Проверил:

Старший преподаватель каф. ТС и ВС _____/ Ахпашев.Р.В/

«__» _____ 2023 г. (подпись)

Новосибирск 2023

Используемые технологии:

- Язык программирования - C++
- Среда разработки – Qt Creator

Цель курсовой работы: Повысить навыки владения языком программирования C++ и навыки работы в среде разработки Qt Creator. По итогу – разработать программу, рассчитывающую и отображающую распространение и затухание сигнала на реальных координатах местности.

Задачи:

- Изучить основы 3D-визуализации в Qt Creator.
- Разработать алгоритм для моделирования распространения сигнала.
- Разработать алгоритм для моделирования затухания сигнала.
- Интегрировать алгоритмы распространения и затухания сигнала в 3D-визуализацию.
- Тестировать и оптимизировать работу программы.

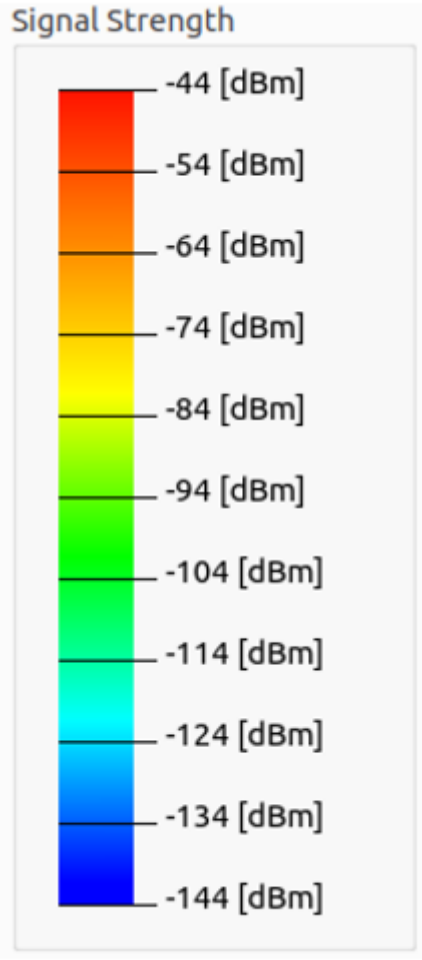
Шаг 1:

На первом шаге нашей курсовой работы создадим тепловую пиксельную карту для модели распространения радиосигнала

1. Создать окно приложения при помощи класса QMainWindow;

2. Создать пиксельную карту (**не менее 1000x1000 пикселей**);

3. Случайным образом установить Точку доступа (базовую станцию) 5G New Radio.

4. Отрисовать тепловую карту (закрасить каждый пиксель определенным цветом) для модели распространения радиосигнала из спецификации 3GPP TR 38.901, пункт 7.4 (выбрать случайную модель из таблицы) [1].
5. Один из возможных выборов градиента:
   i. 

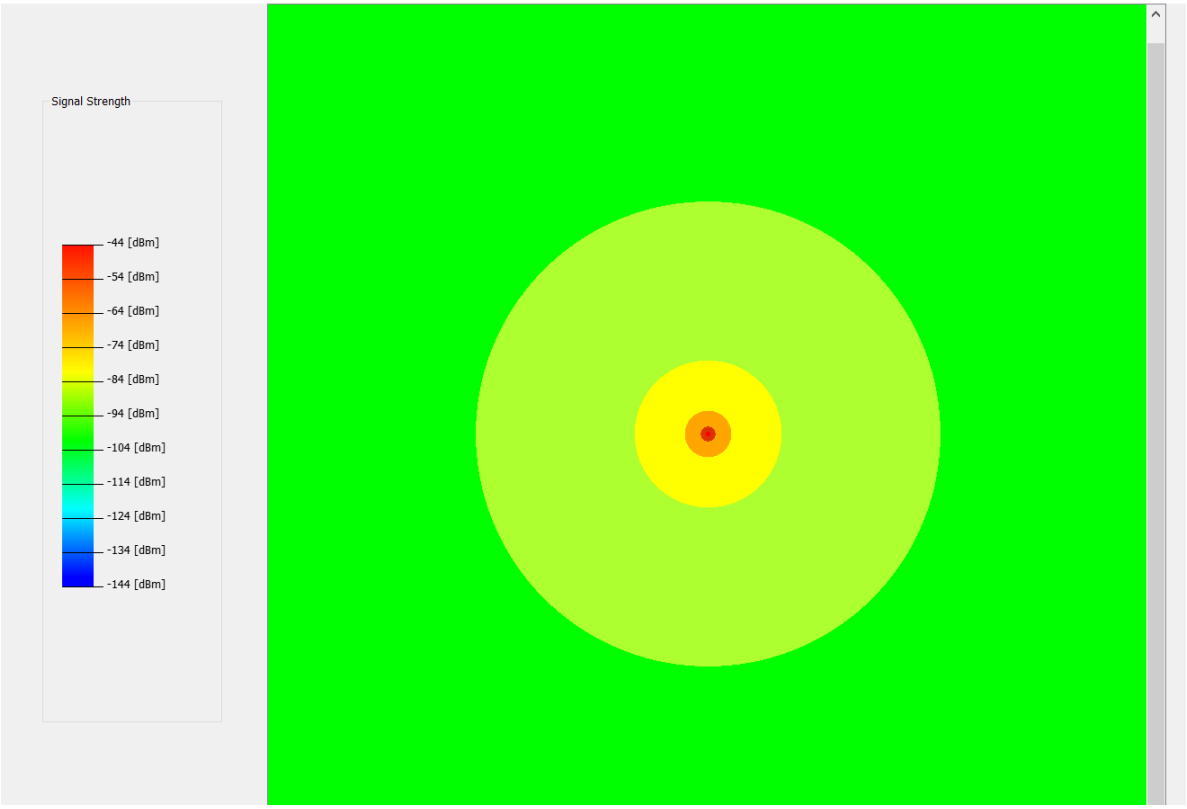
6. Несущая частота (fc) может быть в пределах от 2.1 [GHz] до 6 [GHz].

7. Дистанцию между пикселями определить самим. Например, 1 пиксель = 1 метру, или 10 метрам….зависит от сценария планирования.

Для отрисовки карты используется класс QPixmap, который представляет собой изображение, которое может быть отображено в виджетах Qt. В вашем коде создается объект QPixmap с размерами SIZE_MAP_X на SIZE_MAP_Y. Затем на этом изображении рисуются различные элементы с помощью объекта QPainter, который предоставляет функции для рисования на изображении. После завершения рисования изображение добавляется в сцену с помощью метода addPixmap() класса QGraphicsScene .

Отрисовка градиента: Градиент отрисовывается с помощью класса QLinearGradient, который представляет собой линейный градиент цвета. В вашем коде создается объект QLinearGradient с определенными цветами на разных уровнях. Затем этот градиент используется для заполнения прямоугольника с помощью объекта QPainter. Затем этот прямоугольник добавляется в виджет QLabel, который затем добавляется в группу QGroupBox

В итоге получаем следующее изображение :



Шаг 2:

1. Случайным образом установить на PixMap препятствия (круги, прямые, прямоугольники и т.д.) заранее определяя тип материала:

a. Стеклопакет

b. IRR стекло

c. Бетон

d. Дерево/гипсокартон

2. Используя алгорит Брезенхэма, найти количество препятствий на пути луча от точки доступа до каждого пикселя. Посчитать количество найденных препятствий.
3. На основании спецификации 3GPP TR 38.901 (7.4.3.1 O2I building penetration loss) посчитать степень затухания радиосигнала при прохождении через препятствие [1]. Вычесть полученное затухание от основного бюджета канала.
4. Закрасить соответствующим образом каждый пиксель.

Определим материалы –

1. Стеклопакет
2. IRP стекло
3. Бетон
4. Дерево

Для каждого материала определим соответствующий цвет и коэффициент поглощения сигнала, который будем использовать для формулы расчета затухания сигнала.

Потери радиосигнала считаем по формуле:

```
double formula(double f, double d) {
   return 28 + 22 * log10(f) + 20 * log10(d);
}
```

Формула, которую вы представили, может быть использована для расчета коэффициента затухания сигнала. Этот коэффициент затухания используется для определения, насколько сигнал ослабляется по мере его распространения в среде. Этот параметр играет важную роль в таких областях, как телекоммуникации, электроника и оптика.
Сигнал отрисовывается с помощью функции drawBresenhamLine(), которая использует алгоритм Брезенхема для отрисовки линии между двумя точками. В этой функции для каждой точки на линии проверяется, какой материал находится на этой точке, и в зависимости от этого выбирается цвет для отрисовки этой точки.

Алгоритм Брезенхема - это быстрый алгоритм для отрисовки линии между двумя точками. Он использует только целочисленное сложение и вычитание, что делает его очень эффективным для использования в компьютерной графике.

Вот как работает алгоритм Брезенхема:

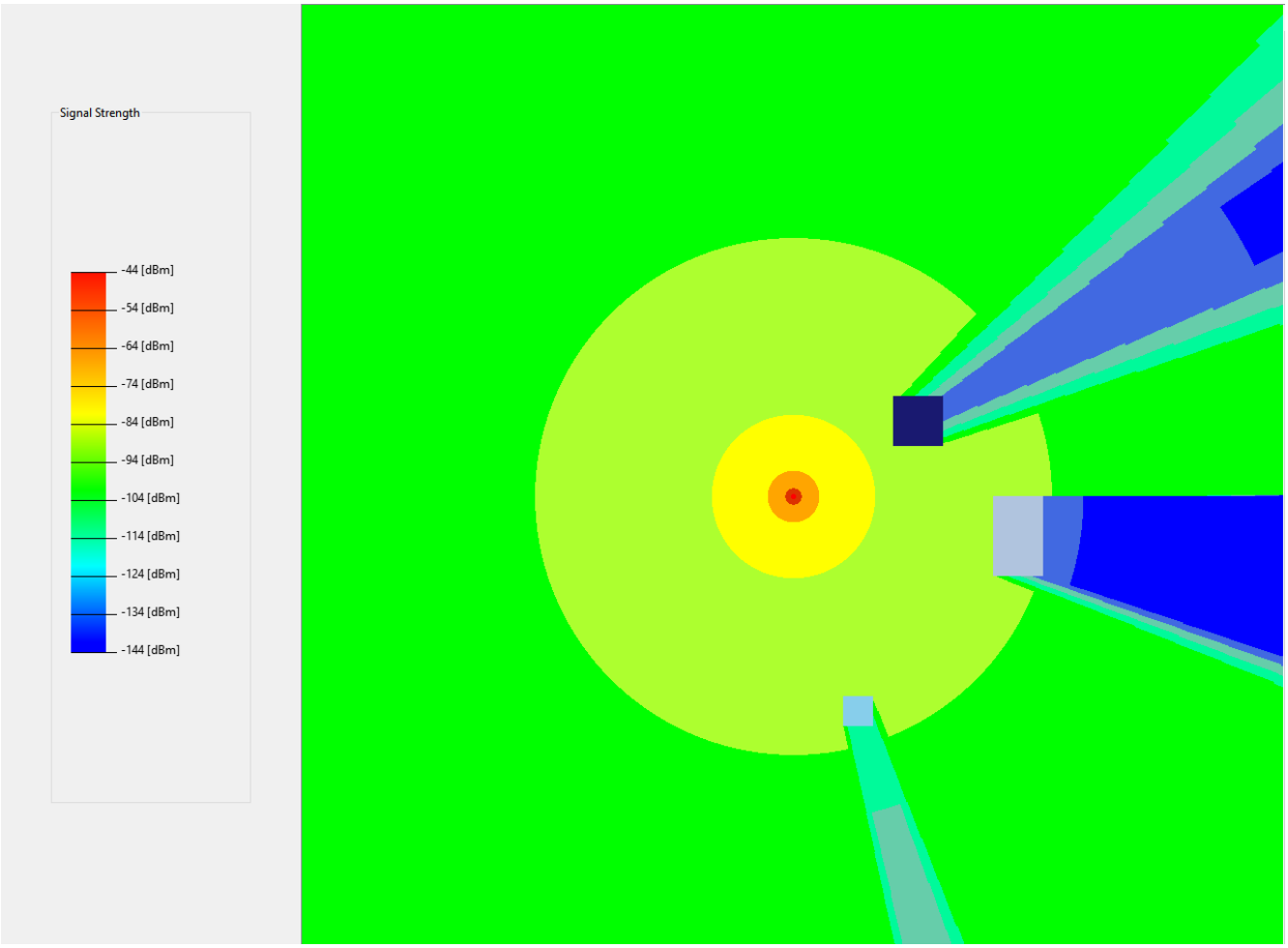
Вычисление наклона: сначала вычисляется наклон линии между двумя точками. Это делается путем вычисления разности координат x и y между двумя точками.

Вычисление ошибки: затем вычисляется начальная ошибка, которая равна половине абсолютного значения наклона.

Отрисовка линии: после этого для каждого следующего пикселя вычисляется ошибка. Если ошибка больше или равна половине наклона, то ошибка уменьшается на наклон, и текущий пиксель увеличивается по оси y. Если ошибка меньше половины наклона, то текущий пиксель увеличивается по оси x.

Повторение: Этот процесс повторяется для каждого следующего пикселя, пока не будет достигнут конечный пиксель.

В итоге получаем следующее изображение :



Шаг 3:

Вопросы архитектуры приложения

1. Создать новую ветку нашего Git-проекта.

2. Реализовать собственный класс “Тепловой карты”. Класс должен включать методы:

   1. Вычисления цвета каждого пикселя на основе мощности радиосигнала;

   2. Раскраска тепловой карты по выбранным цветам;

   3. Доп. методы. Смысловые, на свое усмотрение;

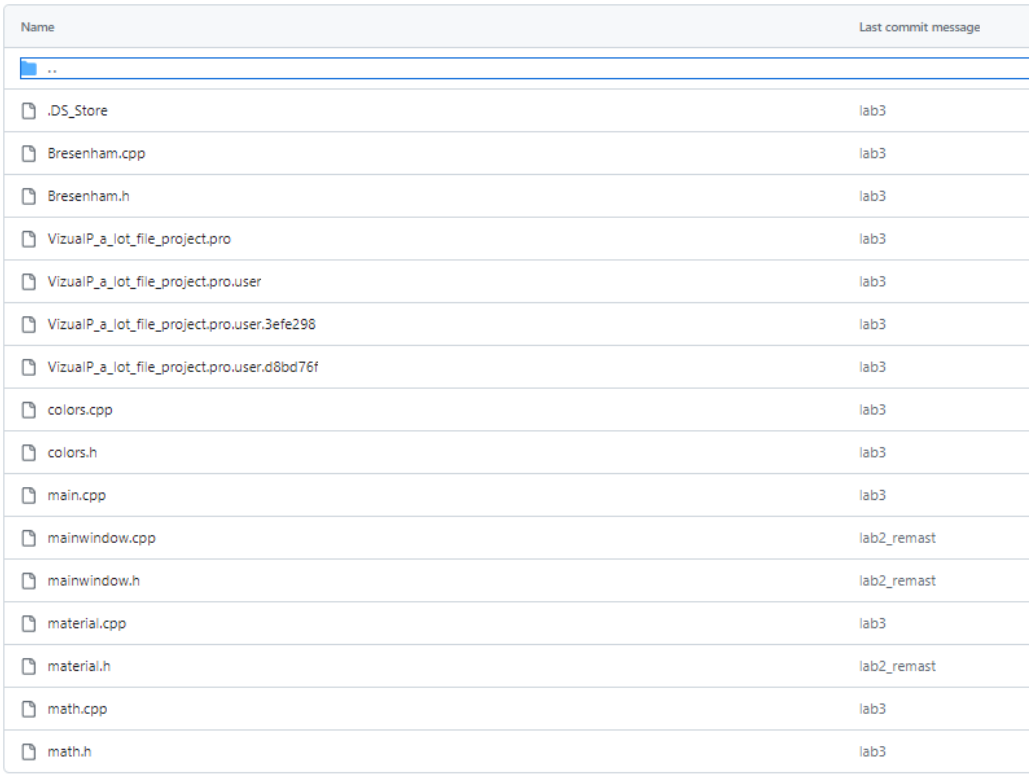
3. Реализовать класс “Propagation Model”, который должен включать в себе все модели по спецификации 3GPP 38.901;

4. Реализовать класс “Материалов”, который будет помогать считать формулу для затухания при прохождении через стены.

5. Реализовать git push на новую ветку (созданную в пункте 1);

6. Выполнить слияние новой ветки в свою основную.

Разбиваем наш код на многофайловый проект и получаем следующую структуру:



Name	Last commit message
..	
.DS_Store	lab3
Bresenham.cpp	lab3
Bresenham.h	lab3
VizualP_a_lot_file_project.pro	lab3
VizualP_a_lot_file_project.pro.user	lab3
VizualP_a_lot_file_project.pro.user.3efe298	lab3
VizualP_a_lot_file_project.pro.user.d8bd76f	lab3
colors.cpp	lab3
colors.h	lab3
main.cpp	lab3
mainwindow.cpp	lab2_remast
mainwindow.h	lab2_remast
material.cpp	lab3
material.h	lab2_remast
math.cpp	lab3
math.h	lab3

Это необходимо сделать, так как так код станет более читаемым(в любой момент можно будет попасть в нужный участок кода под соответствующим названием)

Для того чтобы создать новую ветку будем использовать команду

git checkout -b lab3

Добавим все файлы в репозиторий

git add .

Создадим commit

git commit -m “lab3”

Заливаем на удаленный репозиторий

git push origin lab3

Для того чтобы объединить ветки перейдем в веку, в которую будем загружать файлы

git checkout Labs

“Мерджим”

git merge lab3

Пушим

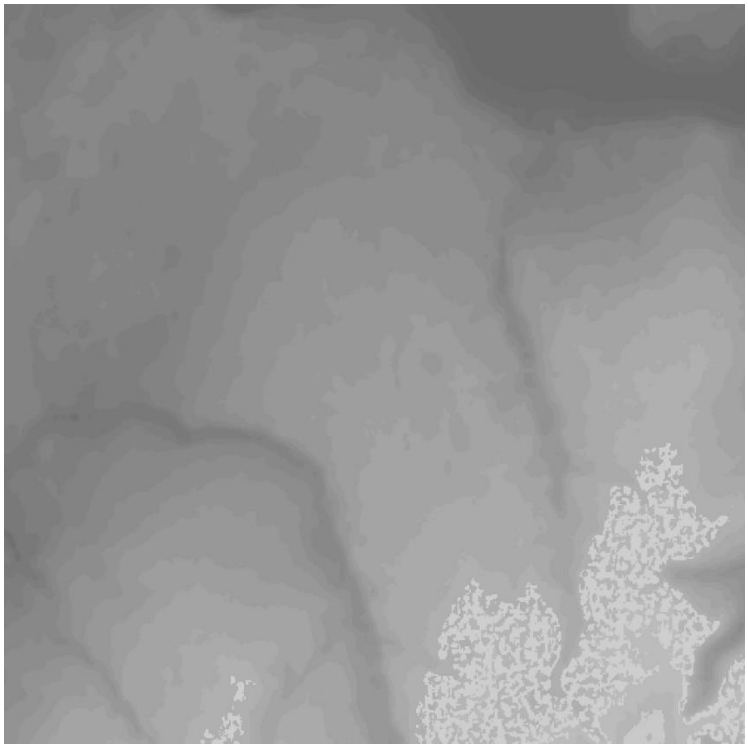
git push origin Labs

Шаг 4:

Реализация 3D-карты с поверхностью Земли.

1. Создать новую ветку нашего Git-проекта.
2. Реализовать 3D-карту поверхности Земли (с учетом высоты над уровнем моря) при помощи Q3DSurface класса;
3. Отобразить результаты шага №1 и №2 на получившейся карте.
4. Реализовать git push на новую ветку (созданную в пункте 1);
5. Выполнить слияние новой ветки в свою основную.

В качестве местности выберем город Бердск.



Из спаршенных с сайта точек высот, получим необходимую нам карту высот, чтобы реализовать её в Qt Creator

```
using namespace QtDataVisualization;

Map_3D::Map_3D(int pos_x, int pos_y, int size_x, int size_y, int size_map_x, int
size_map_y)\
    : pos_x(pos_x), pos_y(pos_y), size_x(size_x), size_map_x(size_map_x),
size_map_y(size_map_y)
{
    graph = new Q3DSurface();
    graph->setSelectionMode(QAbstract3DGraph::SelectionRowAndColumn);
    graph->axisY()->setRange(90.0f, 500.0f);
    container = QWidget::createWindowContainer(graph);
    container->setGeometry(pos_x, pos_y, size_x, size_y);
    container->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);
    container->setFocusPolicy(Qt::StrongFocus);

    QVector3D positionOne = QVector3D(0, 0, 0);

    QImage color = QImage(2, 2, QImage::Format_RGB32);
    color.fill(Qt::red);

    QImage layerOneHMap(filename2);
    QHeightMapSurfaceDataProxy *layerOneProxy = new
QHeightMapSurfaceDataProxy(layerOneHMap);
```

```
    layerOneSeries = new QSurface3DSeries(layerOneProxy);
    layerOneSeries->setItemLabelFormat(QStringLiteral("(@xLabel, @zLabel):
@yLabel"));

    layerOneProxy->setValueRanges(0, size_map_x, 0, size_map_y);

    layerOneSeries->setDrawMode(QSurface3DSeries::DrawSurface);
    layerOneSeries->setFlatShadingEnabled(false);

    graph->addSeries(layerOneSeries);
}

QWidget *Map_3D::get_container(){

    return container;
}
void Map_3D::render(char *file_texture){

    layerOneSeries->setTextureFile("");
    layerOneSeries->setTextureFile(file_texture);


}
QSurface3DSeries *Map_3D::get_QSurface3DSeries(){
    return layerOneSeries;
}

void Map_3D::clicked(int type){
    std::cerr<<"Clicked "<<type<<"\n";
}
```

1 Конструктор: Конструктор Map_3D::Map_3D(int pos_x, int pos_y, int size_x, int size_y, int size_map_x, int size_map_y) инициализирует класс с указанными параметрами. Он создает новый объект Q3DSurface, устанавливает режим выбора и корректирует диапазон оси Y. Также создается контейнер QWidget для графика, устанавливается его геометрия и политика размера, а также устанавливается его политика фокусировки. Создается QHeightMapSurfaceDataProxy с изображением карты высот, и создается QSurface3DSeries с этим прокси. Серия добавляется к графику.

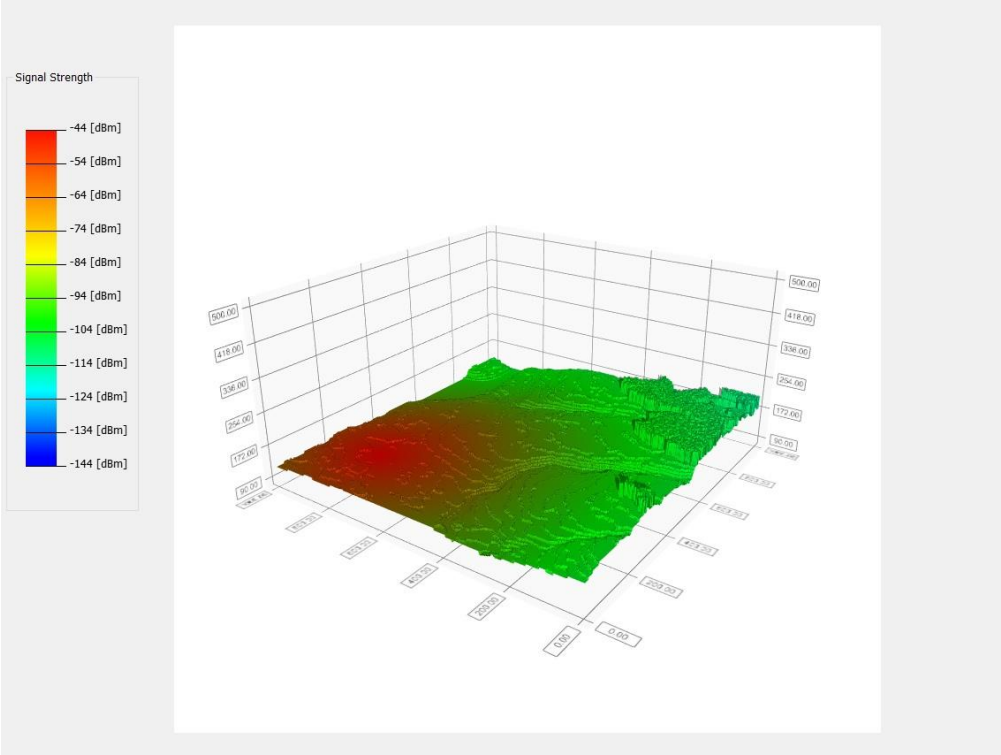
2. get_container(): Эта функция возвращает контейнер QWidget для графика.

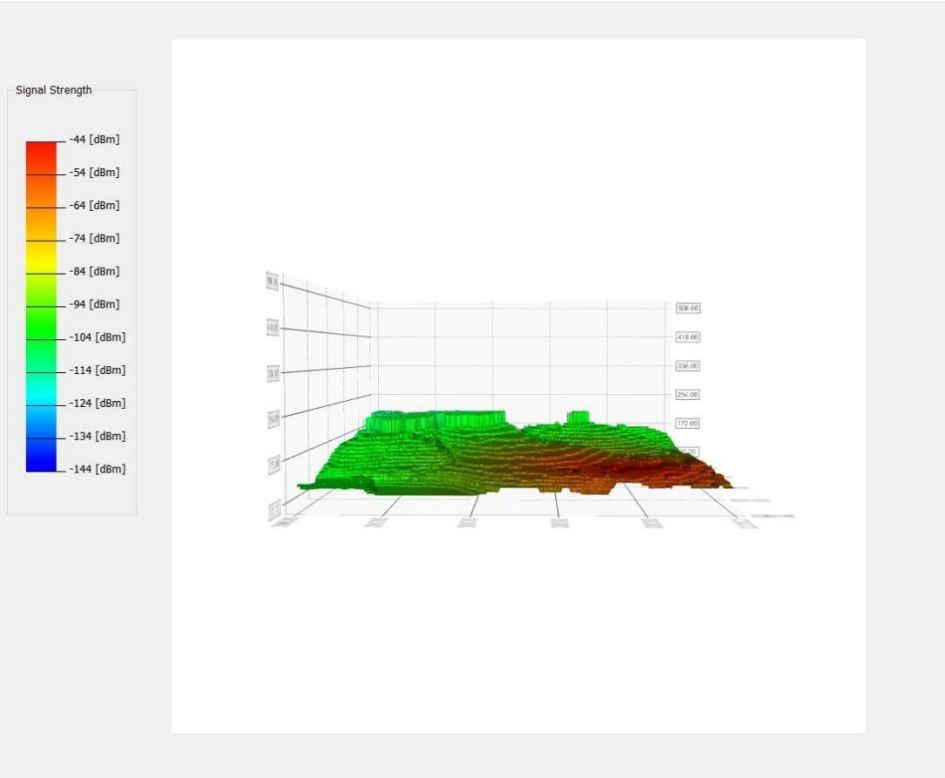
3. render(char *file_texture): Эта функция устанавливает файл текстуры для серии. Сначала она очищает текущий файл текстуры, затем устанавливает его на новый файл.
4. get_QSurface3DSeries(): Эта функция возвращает объект QSurface3DSeries.
5. clicked(int type): Эта функция выводит сообщение в стандартный поток ошибок, указывая, что произошло событие клика.

Класс Q3DSurface используется для рендеринга 3D-графиков поверхности. Он позволяет разработчикам рендерить 3D-графики поверхности и просматривать их, свободно вращая сцену. Визуальные свойства поверхности, такие как режим рисования и затенение, можно контролировать с помощью QSurface3DSeries. Q3DSurface поддерживает выбор, показывая выделенный шарик в точке данных, где пользователь щелкнул левой кнопкой мыши (когда используется обработчик ввода по умолчанию) или выбрал с помощью QSurface3DSeries.

В заключение, этот класс предоставляет способ создания 3D-графика поверхности, добавления к нему серии данных и контроля над ее визуальными свойствами. Он также предоставляет функциональность для обработки событий клика.

В итоге получаем следующее изображение :





Вывод: в ходе данной курсовой работы мы повысили навыки владения языком программирования C++ и улучшили навыки по работе с Qt Creator. Мы научились рассчитывать и моделировать распространение/затухание сигнала, а также поработали с 3д проекцией и познакомились с методом Брезенхэма.

Git проекта

https://github.com/VictorBugaev/Visual_prog/tree/Labs/5semester