МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет» Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет

по производственной практике на АО «Научно-производственное предприятие «Знак», г. Киров

Выполнил студент группы ИВТ-41 /Птахова А.М. /

Руководитель практики от ВятГУ /Клюкин В.Л. / Руководитель практики от предприятия / /

Киров 2024

|  |  |
| --- | --- |
| Дата | Вид деятельности |
| 05.02.2024 | Получение задания на практику |
| 06.02.2024 – 10.04.2024 | Выполнение задания |
| 11.04.2024 – 14.04.2024 | Подготовить и оформить отчет |

Введение

Производственная практика является неотъемлемой и важной частью учебного процесса. В ходе неё не только закрепляются теоретические знания, полученные во время обучения, но и отрабатываются необходимые практические умения для эффективной работы в профессиональной сфере, осваивается инструментарий для работы и взаимодействия в команде, оттачиваются коммуникативные навыки для общения с коллегами.

Данный документ представляет собой отчет по производственной практике, проходившей в период с 5 февраля по 14 апреля АО «Научно-производственное предприятие «Знак», г. Киров.

1. Общие сведения о предприятии

АО «НПП «Знак» был создан 19 сентября 1991 года.

Физический адрес: 610000, Российская Федерация, Кировская область, г. Киров, ул. Спасская, д. 30.

Акционерное общество (АО) – это хозяйственное общество, уставный капитал которого разделён на определённое число акций. Акционерное общество несет ответственность по всем своим обязательствам.

Сегодня это предприятие занимается производством радиолокационной, радионавигационной аппаратуры и радиоаппаратуры дистанционного управления.

2. Описание выполненной работы

В ходе выполнения практика 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.

2.1 Процесс построения области покрытия радиотрассы

Процесс построения области покрытия радиотрассы представлен на рисунке 1.

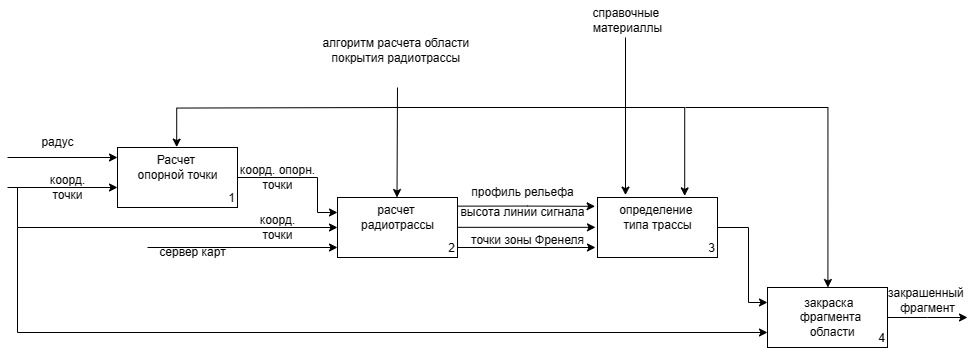


Рисунок 1 – Процесс построения области покрытия радиотрассы

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.

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.

2.2 Разработка алгоритмов функционирования

2.2.1 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 2 изображена граница области видимости при использовании формулы для плоскости.

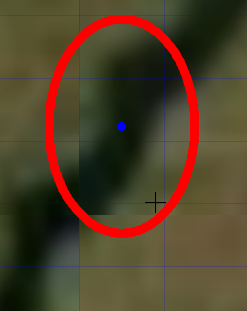


Рисунок 3.2 – Границы области видимости при использовании формулы для плоскости

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))

Граница области видимости при использовании формулы для сферической поверхности представлена на рисунке 3

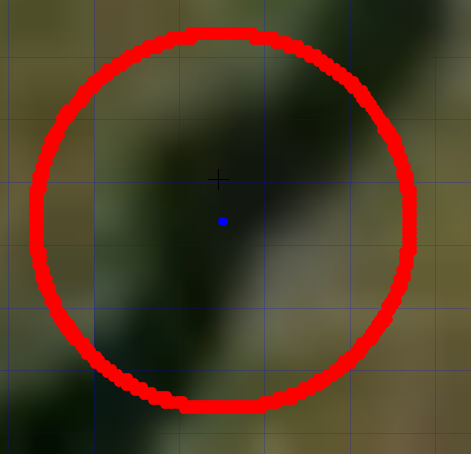


Рисунок 3.3 - Граница области видимости при использовании формулы для сферической поверхности

2.2.2 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 4

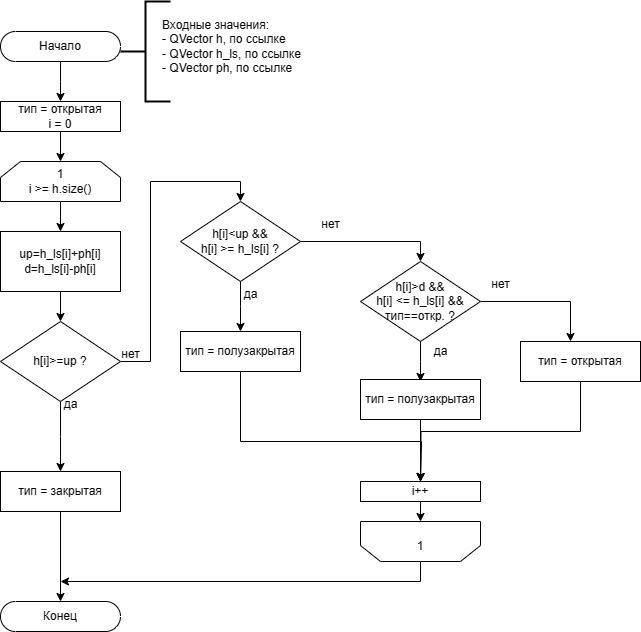


Рисунок 4 – Схема алгоритма определения типа трассы

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.

2.2.3 Закраска фрагмента области

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 5

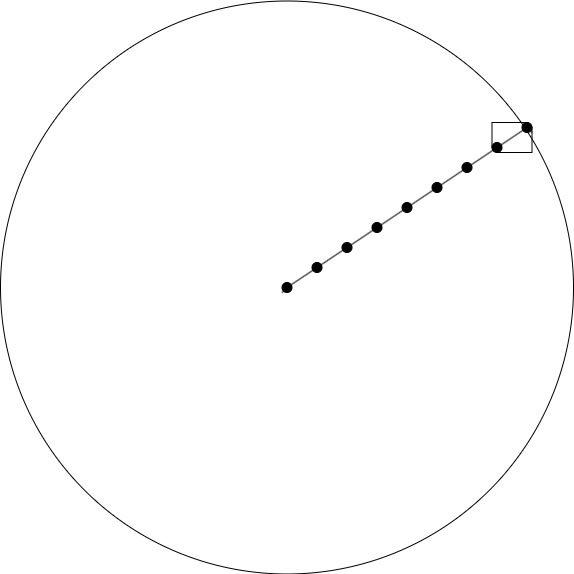


Рисунок 5 – Пример получения границ фрагмента

Результат применения данного алгоритма для расчета области покрытия в радиусе 10 км представлен на рисунке 6.

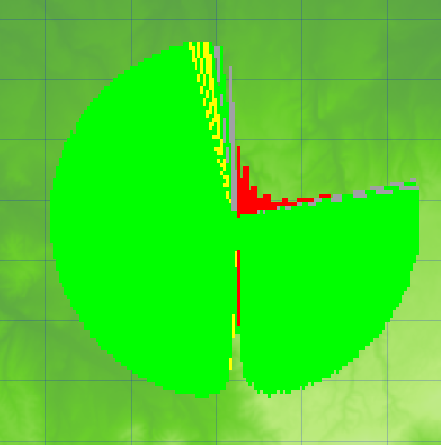


Рисунок 6 – Результат применения алгоритма для области в радиусе 10 км

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 7

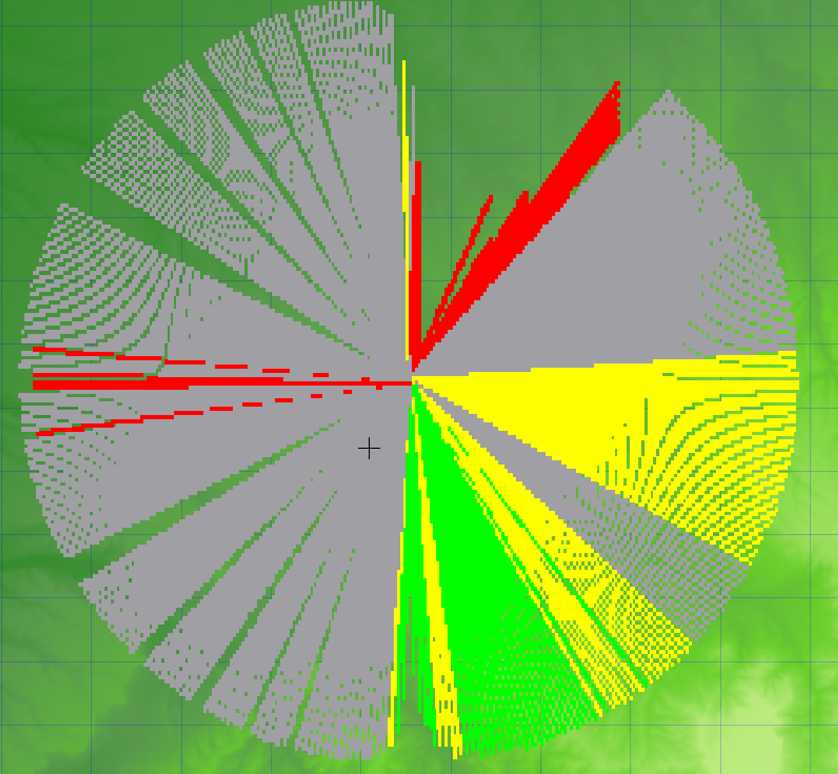


Рисунок 7 - Пример закраски области с просветами при радиусе в 40 км

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.

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 8

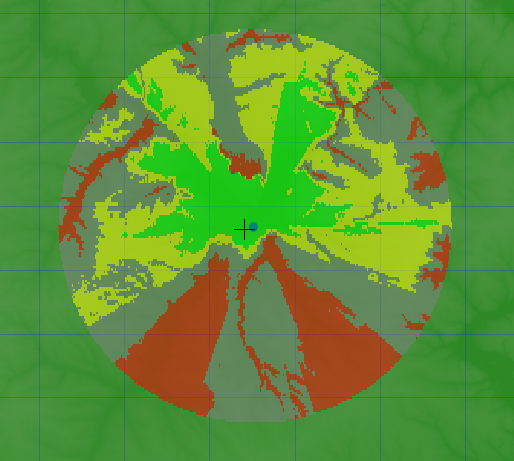


Рисунок 8 - Результат работы алгоритма при разделении области на одинаковые фрагменты

3.3 Программная реализация

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);

}

}

}

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

Приложение А

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

#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;

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() {}