Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и механики

Кафедра прикладной математики

Отчет

по проекту *«Там, где море»*

команды «T»

Выполнили: студенты групп 13641/1,2,3,4

|  |
| --- |
| Лысикова Наталья |
| Зефиров Артем |
| Пестова Наталья |
| Выменец Максим |
| Цуканов Александр |
| Тхакушинова Рузанна |

Санкт-Петербург

2017

Содержание

[1. Общие сведения о проекте 3](#_Toc480619065)

[2. Техническое задание 4](#_Toc480619066)

[3. Модель команды разработчиков и распределение ролей 6](#_Toc480619067)

[4. Управление рисками 8](#_Toc480619068)

[5. План работы 9](#_Toc480619069)

[6. Артефакты 12](#_Toc480619070)

[6.1. Модель предметной области 12](#_Toc480619071)

[6.2. Варианты использования 12](#_Toc480619072)

[6.3. Диаграмма взаимодействия 13](#_Toc480619073)

[6.4. Диаграмма размещения 13](#_Toc480619074)

[6.5. Интерфейс 14](#_Toc480619075)

[6.7. Диаграмма классов 15](#_Toc480619076)

[6.6. Радарное изображение 16](#_Toc480619077)

[7. Контакты 18](#_Toc480619078)

# 1. **Общие сведения о проекте**

**Название проекта**: Транзас

**Оракул**: Крашенниников Сергей Вениаминович (требования) + Ф.А. Новиков (проектирование, реализация)

**Риски**: (см. раздел 4. Управление рисками )

**Vision/Scope**:

* Взаимодействие лица, представляющего интересы заказчика, с менеджером проекта.
* Представление фронтменом проекта отчета по всем этапам реализации проекта.
* Взаимодействие преподавателя с любым участником команды по вопросам проектирования и реализации.

**Цели:**

1. Реализовать проект в соответствии с техническим заданием (см. раздел 2.Техническое задание)
2. Получение всеми участниками команды опыта разработки прикладного программного обеспечения и опыта командного взаимодействия.

**Критерии:**

* Оценка преподавателя курса.
* Реакция заказчика
* Вдохновленное настроение участников команды по окончании работы над проектом.

**Протокол / Репозиторий:** GitHub.

**Коммуникация:**

1. Внутри команды: очные встречи, удаленное общение (vk.com), динамический план реализации

2. С заказчиком: предоставление прототипа программного продукта, финальная презентация проекта.

**Дисциплина:** В случае несвоевременного выполнения поставленной задачи участником команды при условии отсутствия на то уважительной причины дисциплинарное взыскание баллов.

**Технологии**: Qt, UML, С++.

**Модель команды**: (см. раздел 3. Модель команды разработчиков)

# 2. **Техническое задание**

**Краткое описание проекта**

Заказчик проекта – международная компания «Транзас», производитель высокотехнологичного оборудования, программного обеспечения и системной интеграцией для морской отрасли.

Основная цель проекта «Там, где море» заключается в создании утилиты, предназначенной для обработки изображений, получаемых с радара и последующем анализе данных с целью предоставления информации о движении наблюдаемых при помощи радара объектов.

**Решаемые задачи**

В файловую систему поступают с различной частотой файлы, содержащие радарное изображение. В одном файле содержится одно радарное изображение.

Радарное изображение – цифровое изображение круга, содержащее на нейтральном фоне контуры объектов. Каждое изображение снабжено географической координатой своего центра, а также на каждом изображении будет указано время текущее.

Проведена первичная обработка изображений, то есть устранен аппаратный шум.

Каждый файл имеет уникально имя в соответствие с установленным форматом.

Требуется определить:

1. для каждого изображения контуры пятен объектов (вопросы, связанные с форматом представления данных)

2. по последовательности изображений набор движущихся объектов с указанием характера и параметров движения.

**Функциональные и количественные ограничения**

Вывод времени осуществляется в шкале UTC с указанием часового пояса.

Вывод координат осуществляется в стандартных географических координатах (градусы, минуты, секунды, доли секунд).

Ограничение на размер и количество входных файлов определяется размером оперативной памяти исполняющего компьютера. Формат изображения определяется возможностями радара. Требования по размеру использованной памяти будут уточнены позже.

**Требования к эффективности**

Скорость обработки входных файлов должна быть не меньше скорости их поступления. Обработанное изображение кроме пятен объектов будет также содержать информацию с координатами изображения, текущего времени, скорости и направления движения объектов.

**Требования к точности**

Точность определения координат и скоростей объектов зависит от качества поступающих радарных изображений, их количества, а также от внешних условий, при которых осуществлялись наблюдения. В зависимости от этих параметров координаты объектов должны содержать доверительный интервал для уровня вероятности 90%.

**Требования к аппаратному и программному обеспечению**

Утилита должна быть разработана для программных платформ, имеющих возможность выполнения кода, написанного на C++. Это включает:

* выполнение на 32- и 64-битных ОС Linux и Windows (win8/win10) для ПК;
* статическую и динамическую линковку с программными комплексами на C, C++.

**Требования к использованию утилиты**

Утилита должна быть полностью документирована, а также предоставлять в режиме runtime информацию по ее использованию.

**Словарь терминов**

*Изображение* – BMP файл, соответствующий BMP формату.

*Радарное изображение* – цифровое изображение круга, содержащего на нейтральном фоне контуры объектов и цифровой шум.

*Серия изображений* – набор радарных изображений.

*Кадр* – фрагмент действительности, ограниченной картинной плоскостью.

*Точка* – отображение объектов на картинной плоскости.

*Пятна* – набор точек на плоскости, отображающей размер объекта, в пределах заданного масштаба.

*Пиксель* – наименьший логический элемент двумерного изображения.

*Цифровой шум* – дефект изображения, получаемых с радара.

*Объект изображения* – набор пикселей, которые не являются цифровым шумом.

*Скорость* – изменение положения объектов на серии изображений.

*Размер объекта* – число пикселей, которые он занимает на изображении.

*Координаты объекта* – среднее арифметическое координат его пикселей

*Реестр* – список изображений для обработки.

*Разрешение изображения* – величина, определяющая количество пикселей на площади изображения.

# 3. **Модель команды разработчиков и распределение ролей**

В качестве модели команды разработчиков была выбрана иерархическая модель. К преимуществам данной модели относятся

* Возможность четкого распределения обязанностей
* Легкость расширяемости команды

К недостаткам данной модели стоит отнести

* Несколько большую трудоемкость незапланированных коммуникаций между участниками команды по сравнению с другими моделями (бригада главного программиста, модель равных)

Поскольку размер команды Т небольшой, то недостатки иерархической модели не должны оказать значительного влияния. Поэтому участники команды считают, что выбранная модель позволит достичь наибольшей производительности командной работы.

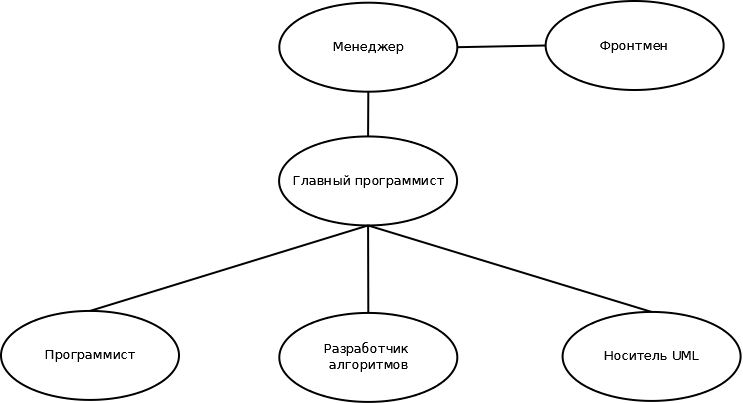


Рисунок 1 – Схема взаимодействия команды Т

Распределение ролей в команде Т осуществляется в соответствие с компетентностями ее участников. Однако в процессе работы над проектом возможны будут корректировки ролей из соображений равенства вклада, вносимого в проект каждым из участников. Распределение ролей приведено в табл. 1.

Таблица 11. Распределение ролей в команде Т

|  |  |
| --- | --- |
| Роль | Участник |
| Фронтмен | Н. Лысикова |
| Менеджер | Р. Тхакушинова |
| Носитель UML | Н. Пестова |
| Разработчик алгоритмов | А. Зефиров |
| Главный программист | М. Выменец |
| Программист | А. Цуканов |

# 4. **Управление рисками**

**Риск**

Риском проекта называется любое событие или условие, которое может оказать негативное влияние на ход выполнения проекта. Под управлением рисками понимается процесс их предсказания, анализа и эффективной превентивной работы над ними. Далее приведен список выявленных рисков и планы по уменьшению их отрицательного влияния.

**Технологические риски**

К технологическим рискам можно отнести несоответствие запланированного инструментария поставленной задаче; такой риск минимален при условии неизменного состояния технического задания.

**Риски, связанные с персоналом**

В связи с особенностью данного проекта могут потребоваться знания работы алгоритмов обработки изображений, а также узкопрофильные знания предметной области. Это накладывает большую ответственность на разработчика алгоритмов и программистов. На данном этапе работы над проектом оценить данный риск сложно.

**Организационные риски**

К организационным рискам относят возникновение ошибок планирования.

Отведение одной из ролей в команде для менеджера позволяет смягчить последствия ошибок планирования, поскольку в таком случае участник команды, исполняющий эту роль, может контролировать качество работы в режиме реального времени, а значит применять технику адаптивного менеджмента, тем самым смягчая и даже нивелируя эффект ошибок планирования.

**Риски, связанные с требованиями заказчика**

Мы рассчитываем на то, что поставленные требования в техническом задании не изменятся вплоть до завершения работы над проектом. Любые корректировки, касающиеся расширения функциональности, могут быть рассмотрены как постановка задач для нового проекта.

# **5. П**лан работ**ы**

Формирование плана-графика проекта позволяет эффективно спланировать работу над проектом. Результатом его формирования является календарный план работ, позволяющий учитывать этапы и результаты работы над проектом, определять длительность работ и распределять обязанности участников. В таблице 2 показ план-график первого этапа работы над проектом.

Таблица 22. План-график первого и второго этапа (18.02.2017- 25.04.2017)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Роль | Задача | Артефакт | Сроки | Результат | Комментарии |
| Менеджер | Составить план реализации проекта «Там, где море» | План реализации  проекта (.doc файл) | 4.03 | выполнено |  |
| Уточнить окончательное ТЗ | Модернизация плана реализации проекта | 11.03 | выполнено | ТЗ может уточняться в дальнейшем, но требования по функциональности и представлению данных останутся неизменными |
| Фронтмен | Представить план проекта на занятии | - | 11.03 | выполнено |  |
| Разработчик алгоритмов | Проанализировать алгоритмы  обработки изображений | - | 11.03 | выполнено |  |
| Главный программист | Осмыслить программную архитектуру и используемые технологий | - | 11.03 | выполнено |  |
| Носитель UML | Разработать диаграмму деятельности, описывающий весь процесс работы утилиты | диаграмма взаимодействия | 11.03 | выполнено |  |
| Носитель UML и Главный программист | Разработать диаграмму размещения исходных файлов проекта | диаграмма размещения | 11.03 | выполнено |  |
| Разработчик алгоритмов | Освоение библиотеки OpenCV. Тестирование ее на реальных изображениях. | Отчет перед командой о результатах | 18.03 | выполнено |  |
| Разработчик алгоритмов | Написание модуля обработки изображений, предоставляющий API остальным модулям. | Библиотека обработки изображений | 31.03 | выполнено | Перенос сроков на неделю в связи с отсутствием исходных данных |
| Разработчик алгоритмов | Документирование API |  | 1.04 | выполнено |  |
| Разработчик алгоритмов | Написание модуля, формирующего log-файла |  | 8.04 | выполнено |  |
| Разработчик алгоритмов | Написание модулей обработки изображения | алгоритмический модуль | 22.04 |  |  |
| Главный программист | Освоение Qt | - | 18.03 | выполнено |  |
| Главный программист | Проектирование архитектуры программы. Написание оберток модулей. | Программный модуль | 31.03 | выполнено | Перенос сроков на неделю в связи с отсутствием исходных данных |
| Главный программист | Написание модулей взаимодействия с интерфейсом и с алгоритмами обработки изображений. | Программные модули | 1.04 | выполнено |  |
| Главный программист | Объединение модулей проекта | Модули проекта | 22.04 | выполнено |  |
| Фронтмен | Освоение Git | - | 18.03-  25.03 | выполнено |  |
| Менеджер | Освоение Git | - | 18.03-  25.03 | выполнено |  |
| Носитель UML | Создание интерфейса программы | Файлы формата .qml с интерфейсом | 31.03 | выполнено | Перенос сроков на неделю в связи с отсутствием исходных данных |
| Менеджер | Создание модели предметной области | Диаграмма МПО | 25.03 | выполнено |  |
| Носитель UML | Создание диаграммы использования | Диаграмма ВИ | 25.03 | выполнено |  |

# **6. Артефакты**

## 6.1. Модель предметной области

Выполнила: Лысикова Наталья 25.03.17.

Корректировка: Лысикова Наталья 12.05.17.

Модель предметной области (рисунок 1) отражает общие сведения о системе, предоставляя необходимые сущности, являющиеся основой для построения диаграммы вариантов использования, пользовательского интерфейса.

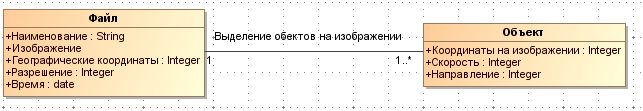


Рисунок 1 – Модель предметной области

На входе программа получает файл изображения из внешнего источника. После обработки, на изображениях выделяются объекты и исключается цифровой шум. В конце обработки получаем характеристику объектов с указанием их координаты и скорости.

## 6.2. Варианты использования

Выполнила: Пестова Наталья 25.03.17

Варианты использования (рисунок 2) отражают действия, которые может выполнить пользователь с программой

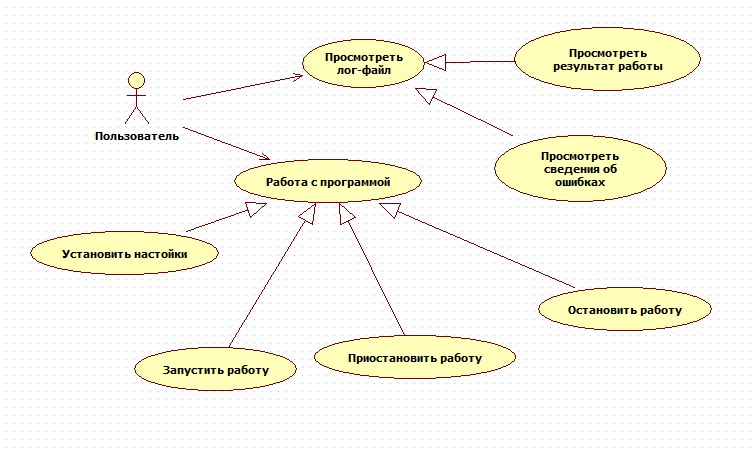


Рисунок 2 – Варианты использования

В данном проекте задействовано одно действующее лицо. Пользователь может работать с программой, а именно установить настройки, запустить обработку изображений, поставить обработку на паузу или остановить обработку. В процессе обработки пользователь может посмотреть лог-файлы, на которых отображены ошибки обработки и характеристики объектов.

## 6.3. Диаграмма взаимодействия

Выполнила: Лысикова Наталья; 11.03.17

Корректировка: Зефиров Артём, Выменец Максим 11.03.17

Диаграмма взаимодействия (рисунок 3) моделирует отношение между объектами системы в рамках технического задания, отображенного в разделе 2.

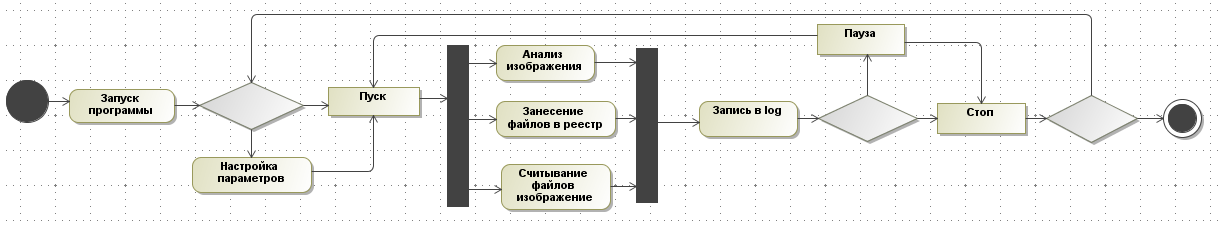


Рисунок 3 – Диаграмма взаимодействия

Диаграмма взаимодействия связана с интерфейсом программы (пункт 6.5). После запуска программы, пользователь может установить настройки или сразу запустить обработку изображения. Программа одновременно выполняет 3 процесса: считывает изображение, добавляет его в очередь, анализирует изображения. Результат обработки записывается в log файлах. Также у пользователя имеется возможность поставить обработку изображения на паузу или остановить изображения. После запуска программы все лог-файлы очищаются.

## 6.4. Диаграмма размещения

Выполнила: Тхакушинова Рузанна 11.03.17

Корректировка: Зефиров Артём, Выменец Максим 11.03.17; Тхакушинова Рузанна 12.05.17

На диаграмме размещения (рисунок 4) отражена реализация компонентов, а также физическое размещение на вычислительных ресурсах. Диаграмма отображает соответствующие блок файлы из программы обработки изображений.

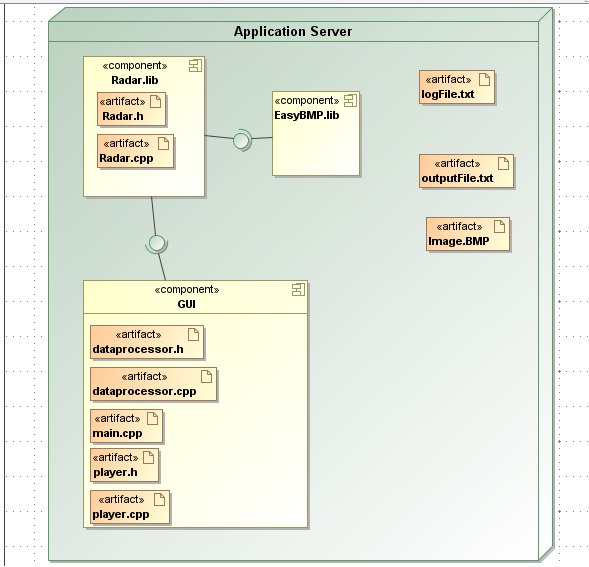


Рисунок 4 – Диаграмма размещения

На рисунке 4 отображена схема размещения. Компонентами диаграммы размещения являются:

1) Gui –интерфейс

* + main.cpp - запуск основного виджета (окна) приложения;
  + player.h,player.cpp - создание и отображение всех виджетов приложения, создание сигнально-слотовых соединений для пересылки событий между объектами - виджетами (кнопками и полями окна настроек) и объектом dataprocessor;
  + dataprocessor.h, dataprocessor.cpp - основной класс, отвечающие за логику приложения, соединенный сигнально-слотовыми соединениями с виджетами интерфейса и вызывающий функции библиотеки обработки изображений.

2) Radar.lib

* Radar.h – объявление атрибутов методов;
* Radar.cpp – чистка изображения от шума и обработка изображений, предоставление характеристик обнаруженных объектов;

3) EasyBMP.lib – библиотека для работы с файлами формата BMP;

4) Image.BMP –получаемое изображение с радара;

5) outputFile.txt – характеристики (координаты и скорости) обнаруженных объектов;

6) logFile.txt – отметки об изменении output-файла и о запуске метода «run», с указанием, какие были входные файлы.

## 6.5. Интерфейс

Выполнил: Зефиров Артём 01.04.17

Корректировка: Пестова Наталья 18.04.17 (дизайн кнопок)

Запуск программы

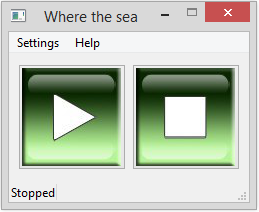


Рисунок 5 – Запуск программы

Выбор настроек

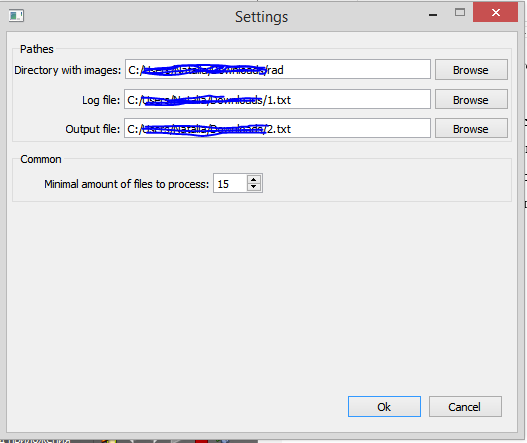


Рисунок 6 – Выбор настроек программы

Запуск обработки изображения

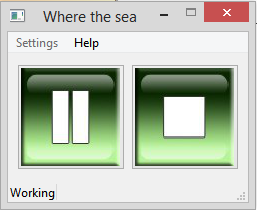


Рисунок 7 – Начало обработки изображения

Приостановка обработки изображения

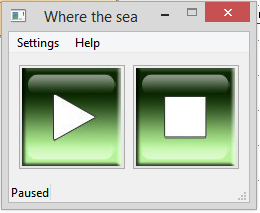


Рисунок 8 – Пауза изображения

Выбор настроек при паузе

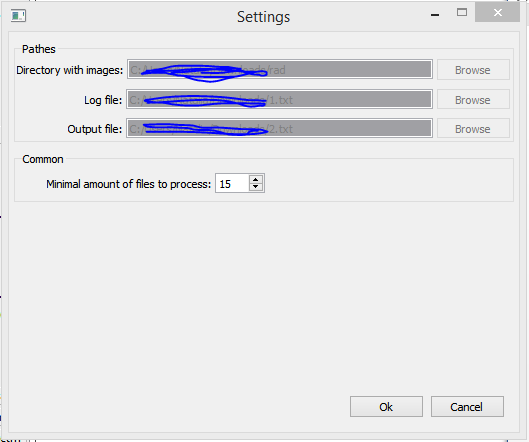


Рисунок 9 – Выбор настроек при паузе

Остановка обработки изображения

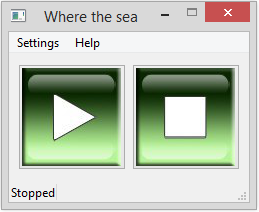


Рисунок 10 – Остановка обработки изображения

Ошибка при указании расположения изображений

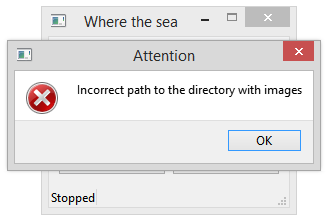


Рисунок 11 – Ошибка указания расположения изображений

Ошибка в указании log и output файлов

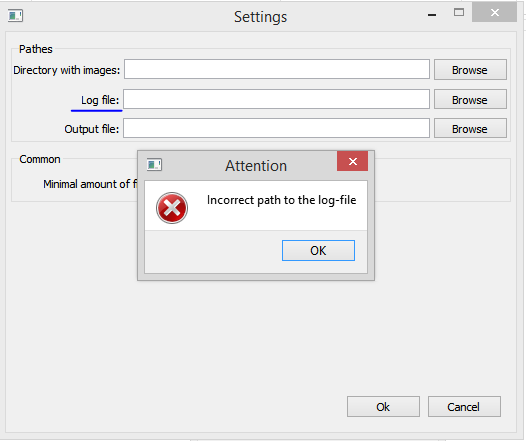
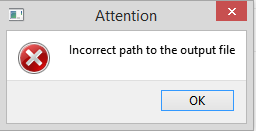


Рисунок 12 - Ошибка в указании log и output файлов

## 6.7. Диаграмма классов

Выполнил: Выменец Максим 21.04.17

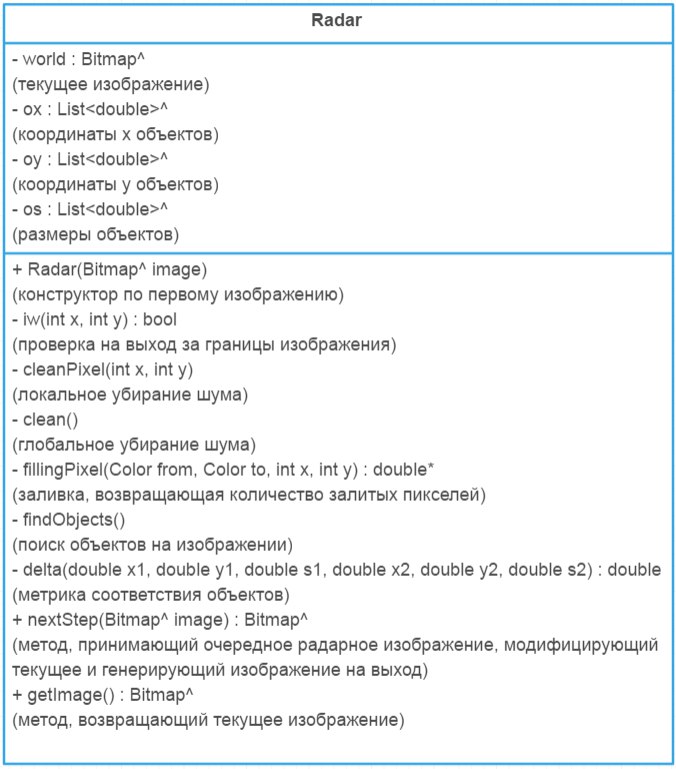


Рисунок 13 – Диаграмма классов

Алгоритм обработки изображений

Выполнил: Выменец Максим 07.05.17

На входе программа получает изображение формата BMP.

**Главный метод ImageProcessing:**

1. Вызов метода **cleanPixel** для всех пикселей p(x,y) изображения поочерёдно.

2. Вызов метода **findObjects**.

3. Если изображение является первым в серии - перейти к шагу 10.

4. Если изображение не первое, то составить матрицу costs. В элементе costs[i][j] находится значение метрики (которая может варьироваться) различия i-го объекта, найденного при предыдущем вызове алгоритма (т.е. методом **findObjects** для предыдущего изображения) и j-го объекта, найденного при этом вызове. Также составить вектора costsI и costsJ, в которых costsI[i] = i и costsJ[j] = j.

5. Найти в матрице costs наименьший элемент, обозначим его costs[minI][minJ]

6. Если costs[minI][minJ] больше константы maxDelta (параметр задается вручную, параметр характеризует ограничение на размер изменения объекта), перейти к шагу 10.

7. Покоординатно вычесть из расположения minJ-о объекта, найденного при этом вызове алгоритма, расположение minI-о объекта, найденного при предыдущем вызове, принять это в качестве скорости объекта.

8. Удалить из матрицы costs строку minI и столбец minJ, из вектора costsI - элемент с индексом minI, из вектора costsJ - элемент с индексом minJ

9. Перейти к шагу 6.

10. Вывести характеристики всех найденных при этом вызове алгоритма объектов, считая их скорость равной нулю по всем координатам, если она неизвестна.

**Метод СleanPixel:**

В данном методе происходит очистка изображений от шума.

1. Если пиксель изображения, для которого вызван метод не является белым или чёрным, завершить выполнение метода.

p[x,y] == White or Black

2. Перебрать соседние пиксели изображения, имеющие одинаковый цвет, сохранить в переменную same число совпадающих с данным пикселем по цвету.

same = sum(p[x,y])

3. Если same меньше двух, обратить цвет пикселя и вызвать метод для соседних пикселей. Например, белые пиксели same, если их меньше 2, будут перекрашены в чёрный цвет.

**Метод findObjects:**

1. Перейти к очередному пикселю изображения, если все пиксели уже перебраны, закончить выполнение метода.

if (p[x+1,y+1] == False) {break}

2. Если пиксель не белый, вернуться к шагу 1.

3. Если пиксель белый, вызвать для него метод **fillingPixel** (см. ниже) с белым цветом в качестве исходного и зелёным в качестве конечного, и сохранить его выходные данные [x,y].

4. Если размер предполагаемого объекта не меньше константы minMonitoredSize (данный параметр задается вручную, параметр характеризует наименьший возможный размер объекта изображения для обработки), то записать его характеристики в список объектов - в качестве каждой координаты следует записать вычисленную методом **fillingPixel** сумму соответствующих координат, разделённую на размер.

5. Если размер предполагаемого объекта меньше константы minMonitoredSize (описание константы п.4), вызвать для текущего пикселя метод **fillingPixel** с зелёным цветом в качестве исходного и чёрным в качестве конечного.

6. Вернуться к шагу 1.

**Метод fillingPixel:**

Метод вызывается для пикселя изображения p[x,y] и принимает два цвета в качестве исходного и конечного. Метод возвращает количество (оно же размер объекта) и сумму координат (для каждой координаты отдельно) перекрашенных пикселей

1. Если цвет пикселя не совпадает с текущим, завершить выполнение метода. В качестве всех выходных значений вернуть нули.

2. Заменить цвет пикселя на конечный, вызвать метод для всех соседних пикселей. Вернувшиеся соответствующие значения сложить.

3. К сумме размеров прибавить единицу, к суммам сумм координат соответствующие координаты текущего пикселя.

4. Вернуть получившиеся значения.

## 6.6. Результат анализа изображения

Выполнил: Выменец Максим – исходное изображение 07.04.17; logfile.txt, outputFile 07.05.17.

**Исходное изображение.**

На исходном изображении (рисунок 14), которое мы получили от радара. В названии каждого файла изображения находится информация о дате изображения, его координата, название. На входном радарном изображении отображен белый шум, который может мешать анализу объектов, находящихся на изображении. В связи с этим одним из существенных этапов обработки изображения является устранение белого шума.

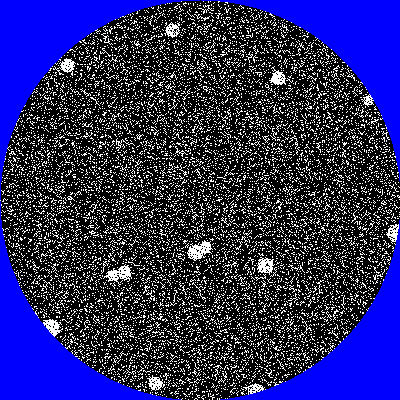


Рисунок 14 – Радарное изображение

Программа получает серию подобных изображений (рисунок 14). После обработки изображений пользователь получает 2 обработанных текстовых файла logfile.txt (рисунок 15) с указанием информации по обработанным изображениям и outputFile.txt (рисунок 16), где указана информация по объектам изображения.

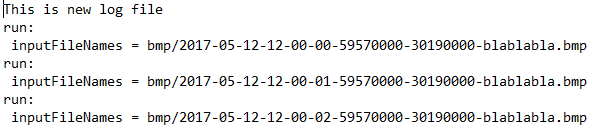


Рисунок 15 – logfile

На рисунке 15 отображена информация об обработанных файлов, которые программа поставила в очередь на обработку.

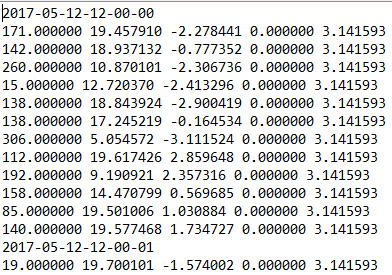


Рисунок 16 – outputFile

В начале изображения отображена дата исходного изображения. После, в 1 столбце указан размер изображения в пикселях. Во 2 столбце – расстояние до объекта из центра изображения по выбранной шкале. В 3 – угол объекта из центра изображения, в радианах. В 4 – скорость объекта, по выбранной шкале. В 5 – Угол движения объекта по направлению вверх, в радианах.

# **7. Контакты**

Выменец Максим Юрьевич:

Тел. 89500376267

e-mail: [kellendros95@yandex.ru](mailto:kellendros95@yandex.ru)