**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное   
учреждение высшего образования   
«Санкт-Петербургский политехнический университет Петра Великого»**

**ИНСТИТУТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

|  |  |
| --- | --- |
|  | **Допустить к защите**  Заместитель директора по  учебно-методической работе  \_\_\_\_\_\_\_\_\_\_\_ \_\_Е. Г. Конакина\_\_\_\_\_  (Подпись) (И.О.Ф.)  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |
|  |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Тема Разработка программного обеспечения HDMapCreator для создания и редактирования карт беспилотного транспорта

специальность 09.02.07 группа 42919/1

Студент (ка) Cластникова А. Д.

(подпись) (ФИО)

Руководитель Панченко М. С.

(подпись) (ФИО)

Санкт-Петербург

2022

СОДЕРЖАНИЕ

[Введение 3](#_Toc104849720)

[1 Общая часть 5](#_Toc104849721)

[1.1 Описание предприятия 5](#_Toc104849722)

[1.2 Анализ предметной области 6](#_Toc104849723)

[1.3 Анализ рынка существующих решений 19](#_Toc104849724)

[1.4 Обоснование и выбор методики, технологии и инструментальных средств разработки программного продукта 23](#_Toc104849725)

[2 Специальная часть 26](#_Toc104849726)

[2.1 Постановка задачи 26](#_Toc104849727)

[2.2 Архитектура и описание структуры разрабатываемого приложения 27](#_Toc104849728)

[2.3 Объектно-ориентированное проектирование приложения 30](#_Toc104849729)

[2.4 Разработка пользовательского приложения 31](#_Toc104849730)

[2.5 Текст программы 40](#_Toc104849731)

[2.6 Тестирование разработанного приложения 40](#_Toc104849732)

[2.7 Оценка качества 42](#_Toc104849733)

[3 Экономическая часть 43](#_Toc104849734)

[3.1 Область применения программного продукта и его преимущества перед аналогичным программным продуктом 43](#_Toc104849735)

[3.2 Трудоемкость разработки программного продукта, квалификация исполнителя и его оклад 43](#_Toc104849736)

[3.3 Расчет затрат на разработку 45](#_Toc104849737)

[3.4 Расчет затрат на внедрение 50](#_Toc104849738)

[3.5 Расчет цены и прибыли 55](#_Toc104849739)

[4 Техника безопасности и охрана труда 58](#_Toc104849740)

[4.1 Анализ безопасности и охрана труда 58](#_Toc104849741)

[4.2 Расчет искусственного освещения рабочего места в ООО «НПО «СтарЛайн»» 60](#_Toc104849742)

[4.3 Электробезопасность в ООО «НПО «СтарЛайн»» 62](#_Toc104849743)

[4.4 Пожарная безопасность в ООО «НПО «СтарЛайн»» 64](#_Toc104849744)

[Заключение 64](#_Toc104849745)

[Список использованных источников 68](#_Toc104849746)

[Приложение А 69](#_Toc104849747)

[Приложение Б 71](#_Toc104849748)

# ВВЕДЕНИЕ

С каждым годом в России и в мире все больше развиваются технологии автоматизации транспортных средств. Наивысшей ступенью автоматизации является автономный беспилотный автомобиль.

Наиболее заинтересованной в развитии беспилотных технологий стороной является крупный бизнес, испытывающий существенные логистические издержки. На данный момент в этих издержках существенную роль играет человеческий фактор, водитель обязан соблюдать режим работы и отдыха, а также не всегда может управлять транспортным средством, оптимально расходуя топливо.

Многие крупные российские компании развивают беспилотные технологии самостоятельно или поддерживают стартапы. Так компания «Яндекс» выпустила свой первый прототип в 2017 году. Другая российская компания, «НПО «СтарЛайн»», развивает два независимых проекта RICAR и OSCAR. Проект RICAR выступал в финале национального конкурса «Зимний город» и показал лучший результат среди всех участников. Идеологией проекта OSCAR является не только развитие технологий, но и взаимодействия в сфере, поэтому любой желающий может использовать наработки команды, а также участвовать в различных мероприятиях, ими проводимыми.

Законодательство РФ также дополняется, создавая возможность интеграции беспилотных транспортных средств. Так в Постановлении Правительства РФ от 26 ноября 2018 г. N 1415 "О проведении эксперимента по опытной эксплуатации на автомобильных дорогах общего пользования высокоавтоматизированных транспортных средств" были указаны правовые рамки эксперимента на территории г. Москвы, Республики Татарстан, Владимирской, Ленинградской, Московской, Нижегородской, Новгородской и Самарской областей, Чувашской Республики, Ханты-Мансийского и Ямало-Ненецкого автономных округов, Краснодарского края и г. Санкт-Петербурга.

В беспилотных транспортных средствах выделяют следующие важные модули: локализацию, распознавание, прогнозирование, планирование, управление и картография. Именно последний модуль, картография, позволяет внедрить беспилотные транспортные средства в городскую инфраструктуру с соблюдением правил ПДД.

Целью выпускной квалификационной работы является создание десктопного приложения HDMapCreator, предназначенного для создания и редактирования высокоточных карт беспилотных транспортных средств в формате Apollo OPENDrive, разработанного в рамках проекта с открытым исходным кодом Apollo компании Baidu. Выбранный формат обоснован тем, что разработка данного программного обеспечения производилась для проекта OSCAR – открытой программно-аппаратной платформы на базе проекта Apollo.

В соответствии с целью были поставлены следующие задачи:

* 1. провести анализ предметной области;
  2. проанализировать аналогичные программные продукты;
  3. провести проектирование десктопного приложения;
  4. разработать десктопное приложение

# Общая часть

## Описание предприятия

Научно-производственное объединение «СтарЛайн» входит в группу компаний УльтраСтар, занимающихся разработкой систем безопасности с 1988 года. Президент НПО «СтарЛайн» – Аминджанов Темур Асрорович, генеральный директор – Вильский Сергей Владиславович. Миссия компании: «Мы обеспечиваем жителей России и мира высоким уровнем безопасности на основе прогрессивных технологий мирового сообщества» [1].

Основные направления деятельности компании:

1. Разработка и производство автомобильного охранно-телематического оборудования;
2. Разработка телематических сервисов;
3. Комплексное контрактное производство полного цикла;
4. Полный цикл постпродажного обслуживания: сервисные центры во всех крупных городах России и СНГ, Федеральная служба технической поддержки;
5. Беспилотные технологии: проект «Умный Автомобиль» (RICAR) и проект OSCAR.

Проект OSCAR – это открытая программно-аппаратная платформа для создания беспилотного автомобиля 4-го уровня автономности на базе проекта apollo.auto от Baidu, все желающие могут поучаствовать в процессе разработки беспилотника на уровне кода, опробовать свои алгоритмы на реальном автомобиле, оснащенном дорогостоящим оборудованием. На рисунке 1 представлен автомобиль Lexus RX450h, на базе которого реализуется проект OSCAR.



Рисунок 1 – Беспилотный автомобиль проекта OSCAR

В 2019 году были проведены массовые тест-драйвы беспилотного ТС в Санкт-Петербурге, во время которых жителям города представилась уникальная возможность одними из первых протестировать автомобиль будущего, разрабатываемого в рамках открытого проекта OSCAR.

В мае 2020 года беспилотный автомобиль StarLine прошел серию государственных испытаний и получил заключение о соответствии автомобиля требованиям безопасности. Это заключение дает возможность тестирования беспилотного автомобиля на дорогах общего пользования в рамках постановления Правительства №1415.

## Анализ предметной области

Беспилотный автомобиль – это автомобиль, оборудованный системой автоматического управления, способный передвигаться из точки А в точку Б без участия человека.

Согласно классификации Сообщества автомобильных инженеров (SAE) [2], беспилотные автомобили классифицируются по шести уровням автономности (рисунок 2), начиная с нулевого:

1. Level 0 – No Automation (без автоматизации). Водитель должен контролировать все – руль, тормоз и газ. Обычная машина.
2. Level 1 – Driver Assistance (помощь водителю). Автомобиль помогает тормозить или ускоряться. Автомобили с круиз-контролем  можно назвать беспилотными автомобилями первого уровня.
3. Level 2 – Partial Automation (частичная автоматизация). Автомобиль может одновременно контролировать ускорение и торможение, но человек должен следить за ситуацией и быть готовым принять управление.
4. Level 3 – Conditional Automation (условная автоматизация). Автомобиль может полностью управлять движением, но в какой-то момент может попросить принять управление на себя.
5. Level 4 – High Automation (высокая автоматизация). Система умеет все, что умеет уровень 3, но также может справляться с более сложными дорожными ситуациями. В целом оператор не должен предпринимать действий, но, если автомобиль не сможет принять решение, он об этом оповестит оператора и плавно припаркуется на обочине.
6. Level 5 – Full Automation (полная автоматизация). Полная автономия, участие человека не требуется. Машина сама принимает решение в любой ситуации, руль может отсутствовать.



Рисунок 2 – Уровни автономности беспилотного автомобиля

Чтобы приехать в пункт назначения, беспилотный автомобиль должен знать маршрут, понимать окружающую обстановку, соблюдать ПДД и корректно взаимодействовать с пешеходами и другими участниками дорожного движения. Чтобы соответствовать этим требованиям, беспилотник использует следующие органы и технологии:

1. Камеры, позволяющие реализовать визуальное обнаружение объектов, например, дорожной разметки и знаков.
2. Радар, который использует радиоволны для обнаружения препятствий и объектов спереди и сзади и расстояния до них, а также, благодаря допплер-эффекту, позволяет вычислить их радиальные скорости.
3. Лидар – это активный оптический дальномер, который постоянно сканирует окружающее пространство и формирует его трехмерную карту, так называемое облако точек.
4. AI (искусственный интеллект) – «мозги» беспилотного автомобиля. Обрабатывает данные с камер и сенсоров, прогнозирует поведение участников дорожного движения и принимает решения.

На рисунке 3 приведено примерное расположение оборудования в транспортном средстве.

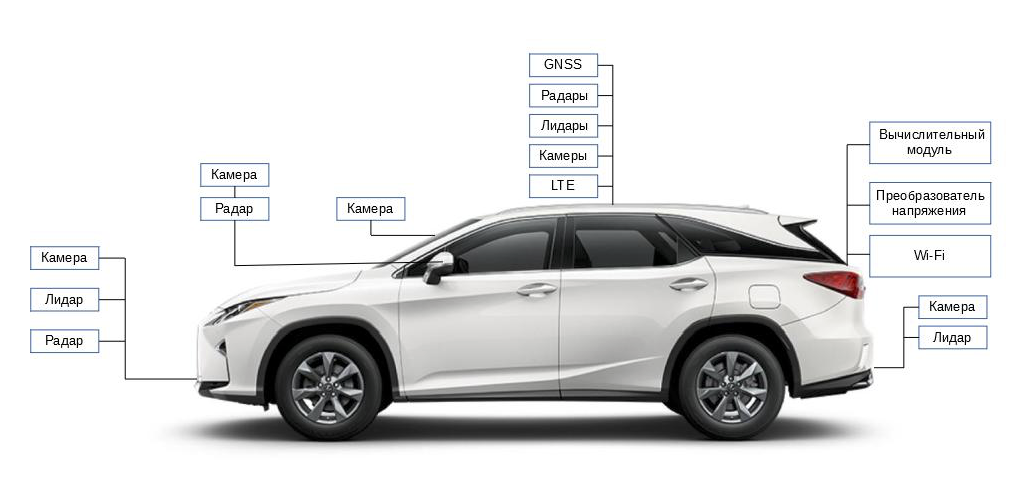


Рисунок 3 – Примерное расположение оборудования на машине

Схему взаимодействия программных модулей приведена на рисунке 4.

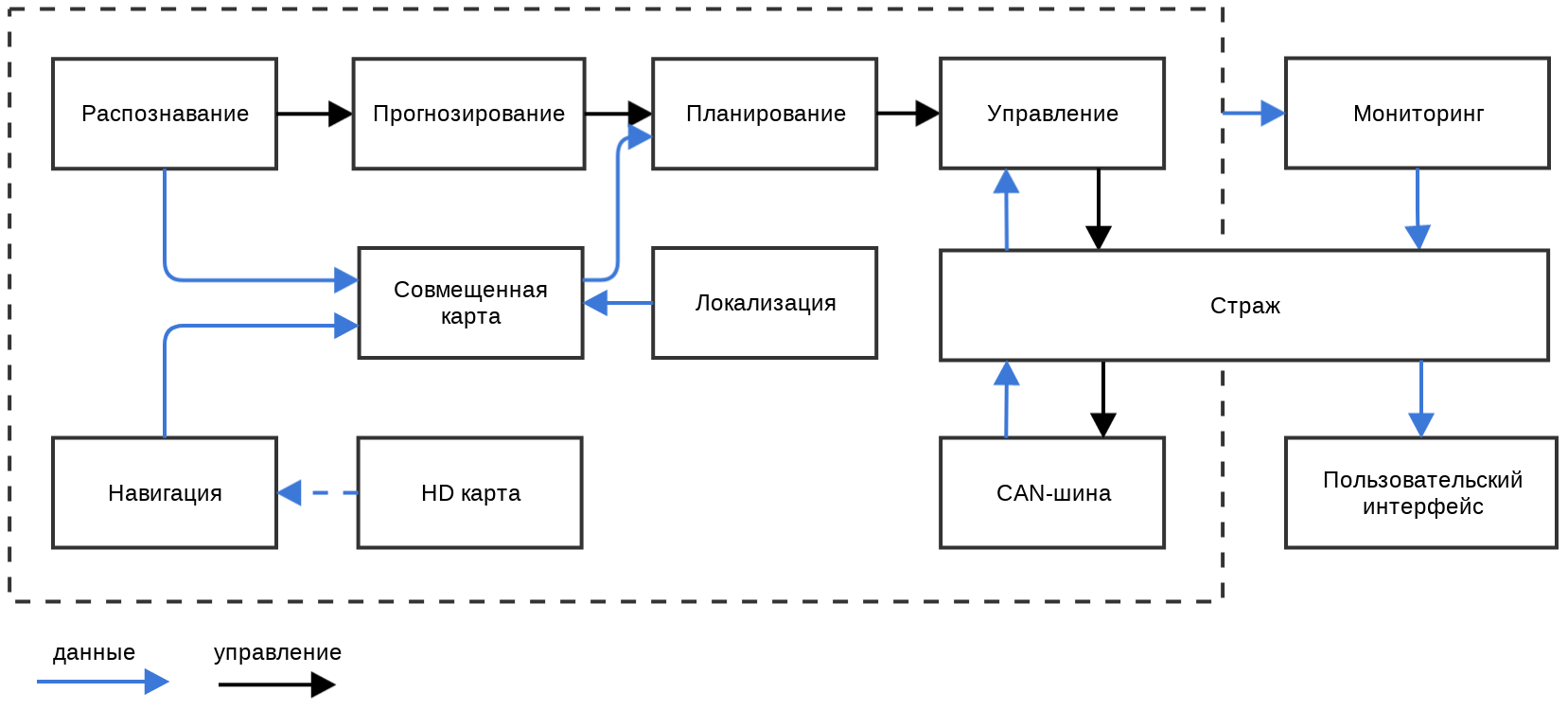


Рисунок 4 – Программные модули проекта OSCAR

Управление автомобилем происходит посредством CAN-шины. CAN-шина – это стандарт промышленной сети, а также система цифровой связи и управления электрическими устройствами автомобиля. Автомобильные устройства подключаются к CAN-шине параллельно для отправки и приема данных. Имея доступ к этому цифровому интерфейсу, можно, например, включать двигатель, открывать двери, складывать зеркала автомобиля с помощью управляющих команд.

Формирование команд для CAN-шины занимается модуль управления, который составляет их в соответствие с построенной модулем планирования траекторией. Модуль планирования в свою очередь ориентируется на данные модуля прогнозирования, который рассчитывает ожидаемые траектории движения других участников дорожного движения, время загорания сигналов светофоров и другие факторы. Модуль прогнозирования основывает свои предсказания на паттернах поведения участников дорожного движения и результатах работы модуля распознавания

Модуль распознавания предоставляет сведения о динамических препятствиях, например, о движущихся рядом автомобилях или перебегающем дорогу человеку. Для этих целей обрабатываются данные радаров, лидаров и камер. С лидаров и радаров в модуль распознавания передаются облака точек (рисунок 5), а с камер фотографии. Данные сперва обрабатываются независимо, а потом объединяются друг с другом и данными локализации. Таким образом на выходе модуля распознавания получены данные о препятствия, сигналах светофора, знаках, а также рассчитаны их координаты в мировой системе координат.

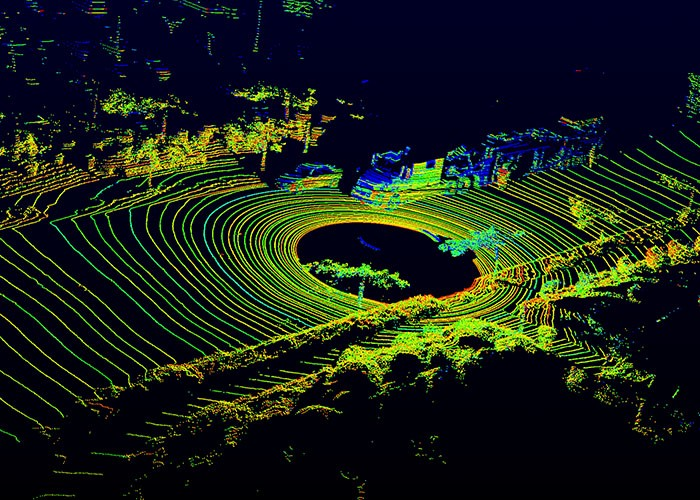


Рисунок 5 – Пример облака точек, полученного с лидара

Локализация позволяет определить начальное положение автомобиля в мировой системе координат, а также следить за отклонением фактического местоположения от запланированного маршрута.

Для позиционирования беспилотного автомобиля ошибка гражданского GPS-приемника слишком велика, и поэтому в проекте OSCAR используется сверхточная спутниковая навигация. Для этого на автомобиль был установлен GNSS-RTK приемник, который использует более совершенные техники позиционирования и, таким образом, достигает сантиметровой точности позиционирования только за счет спутниковой навигации.

Но во время движения автомобиль может оказаться в тоннеле или под мостом, где сигналы от спутников слишком слабы либо отсутствуют вовсе. Из-за этого точность GNSS-RTK приемника упадет, либо позиционирование окажется вовсе невозможным. В этих случаях автомобиль будет уточнять свое положение, используя показания одометра, акселерометра, лидаров и камер. Такие алгоритмы, которые объединяют данные с нескольких разнородных источников, чтобы уменьшить неопределенность, называются алгоритмами Sensor Fusion.

Другим немаловажным источником данных является картография. Она позволяет существенно оптимизировать работу модулей распознавания и планирования. Модуль карт сообщает модулю распознавания, какой объект ожидается в данной области видимости, что упрощает классификацию статических окружающих объектов. Например, это позволяет искать светофоры только на тех кадрах, мировые координаты которых близки к координатам светофора на карте.

В картографии выделяют SD-карты и HD-карты.

1. SD-карты – карты обычной точности (небольшой ошибки принято считать отклонение равное 1 м), использующиеся для гражданской навигации.
2. HD-карты (high-definition maps) – высокоточные 3D карты, используемые для управления беспилотным автомобилем, содержащие детали, обычно не отображаемых на традиционных картах. Такие карты могут быть точны вплоть до сантиметров. На рисунке 6 приведен пример визуализации HD-карты.

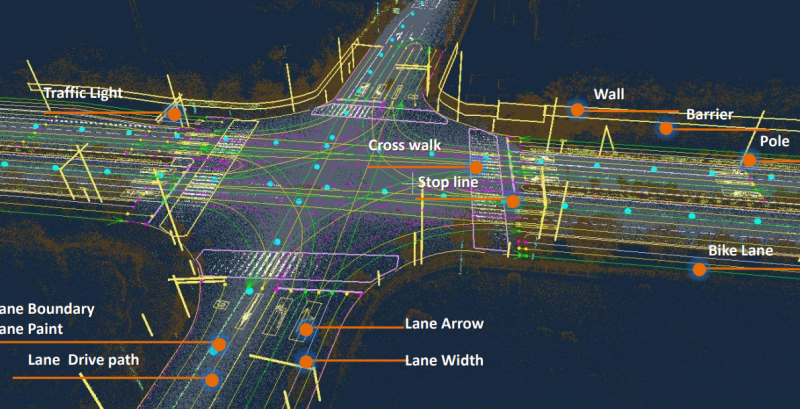


Рисунок 6 – Пример визуализации HD-карты

HD-карты могут быть созданы с помощью наборов данных, собираемых с сенсоров таких как радары, лидары, камеры и GPS, сконструированы с помощью аэрофотосъёмки (см. пример на рисунке 7) или созданы в специализированных программах.



Рисунок 7 – Аэрофотосъемка

При автоматизированном создании карт лидар и камеры работают вместе, затем данные с устройств совмещаются, таким образом возможно не только распознать (классифицировать) окружающие объекты, но и вычислить расстояние до них в 3D пространстве. Пример автоматизированного создания HD-карты представлен на рисунке 8.



Рисунок 8 – Распознавание объектов для автоматизированного создания карт

Высокоточные карты для беспилотных автомобилей обычно включают следующие элементы: разметка и форма дороги, светофоры, дорожное ограждение. Достижение высокой точности – одна из важнейших задач создания HD-карт реальных дорог. Для оценки качества (точности) карт рассматривают две метрики:

1. Глобальная точность (позиционирование относительно плоскости Земли);
2. Локальная точность (позиционирование в отношении окружающих дорожных элементов).

В зонах с хорошим приемом GPS сигнала возможно достичь глобальной точности меньше 3 см отклонения, используя сигнал спутника и поправки с базовой станции.

В зонах блокирования GPS сигнала неточность возрастает в течение продвижения по этой территории, достигая наибольшего значения в ее середине. Максимальная ошибка GPS может быть выражена в процентном соотношении к пройденной дистанции в зоне блокирования GPS сигнала – это значение обычно меньше 0.5%.

По мнению компании Lyft (публичная компания, в том числе развивающие технологии беспилотных автомобилей) HD-карты организуются на 5 слоях (см. рисунок 9): базовая карта (карта стандартных определений), геометрическая карта, семантическая карта, карта приоритетов и карта реального времени.



Рисунок 9 – Структура HD-карт, предлагаемая компанией Lyft.

Геометрическая карта состоит из сырых данных лидара, камер, GPS и IMU (инерциальный измерительный модуль – гироскопическое устройство, предназначенное для стабилизации отдельных предметов или приборов, а также для определения угловых отклонений предметов). На выходе данные представляются облаком точек, которые обрабатываются для выявления объектов, которые будут хранится на геометрической карте.

Семантическая карта создается из геометрической посредством добавления семантических объектов. Они могут быть 2D и 3D, например, границы полос, пересечения объектов, парковочные места, стоп-знаки, светофоры и т.д. Семантические объекты могут содержать информацию о скоростном режиме, ограничение смены полосы движения и другое.

Карта приоритетов содержит динамические данные и данные о человеческом поведение. Например, порядок смены сигналов светофора, среднее время ожидания на светофоре, вероятность обнаружения занятого парковочного места, средняя скорость автомобилей в зоне парковки и т.д.

Карта реального времени – один из наиболее важных слоев, на котором данные о ситуации на дороге обновляются в реальном времени.

Были рассмотрены наиболее распространенные форматы HD-карт.

**ASAM OpenDRIVE** формат – открытый формат описания инфраструктуры дорог, в основе которого лежит XML синтаксис. Данные, сохраняемые в формате ASAM OpenDRIVE, могут описывать геометрию дорог, полос и объектов, например дорожных знаков, а также свойства элементов дороги, например светофоров.

Главной причиной разработки формата ASAM OpenDRIVE была необходимость создать формат для описания дорожной инфраструктуры, который может быть использован для создания симуляций, применяемых для разработки и проверки ADAS (усовершенствованной системы помощи водителю). С помощью формата ASAM OpenDRIVE данные о дорожной инфраструктуре могут быть использованы в разных симуляторах. Предоставление стандартизированного формата описания дорожной инфраструктуры позволяет снизить стоимость создания и преобразования карт из одного формата в другой.

В формате ASAM OpenDRIVE дороги и полосы описываются геометриями (geometrics): прямыми линиями (кривизна равна нулю), дугами (кривизна ненулевое постоянное число), спиралями (кривизна изменяется линейно), кубическими полиномами.

Одним из основных компонентов карты в формате ASAM OpenDRIVE является элемент «Дорога», которая состоит из нескольких частей: базовая линия (наиболее важная часть дороги), полосы и связи с другими дорогами (road linkage). Базовая линия определяет геометрию дороги. Поле полос описывает разные полосы дороги, их количество и ширину. В поле «связи с другими дорогами» описываются соединения с другими дорогами, но для более сложных пересечений используется специальная запись «Соединение» («Junction»). На рисунке 10 приведен пример дороги с соединением.

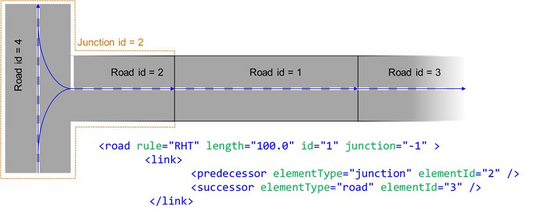


Рисунок 10 – Пример дороги с соединением.

Наиболее важными атрибутами полос являются ширина и тип каждой границы полосы. Ширина может быть указана в метрах или более детально описана полиномом 3-ей степени. Кроме местоположения границ полоса также может содержать информацию о дорожной разметке, материале, доступности полосы (например, выделенная для общественного транспорта).

На дорогах водитель получает большую часть информации с помощью дорожных знаков, светофоров, дорожной разметки. Формат ASAM OpenDRIVE также предоставляет возможность отобразить эту информацию. Каждая дорога может иметь множество связанных с ней дорожных объектов. Эти объекты определяются набором атрибутов, описывающих их расположение, форму и другие важные детали. Например, светофоры – динамические объекты, которые могут менять свое состояние. Они, как и другие дорожные объекты, представляются как тип объекта, в полях которого хранятся данные о текущих состояниях атрибутов.

На рисунке 11 представлена структура формата ASAM OpenDRIVE.

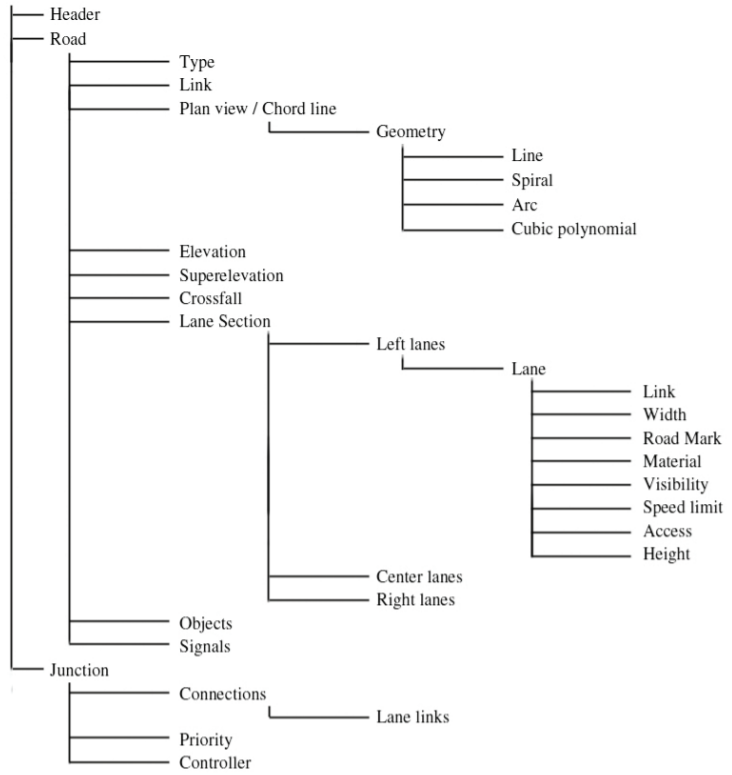


Рисунок 11 – Структура формата ASAM OpenDRIVE

**Apollo OpenDRIVE** формат – открытый формат описания структуры дорог, разрабатываемый компанией Baidu в рамках open-source проекта Apollo. Формат Apollo OpenDRIVE является модифицированной версией формата ASAM OpenDRIVE. Общая структура формата не изменена, но в формате Apollo OpenDRIVE изменено несколько фундаментальных деталей, например, представление дорог.

Одним из фундаментальных элементов стандартов OpenDRIVE являются геометрии объектов. В то время как функции этого элемента остаются неизменны, в спецификации Apollo OpenDRIVE приводится другая структура для него. Также в формате Apollo OpenDRIVE геометрии используются чаще, чем в ASAM OpenDRIVE, они также применяются для описания границ полос и центральной линии. Геометрии Apollo OpenDRIVE определяются набором точек, а не отдельным объектом. На рисунке 12 приведен структура геометрий.

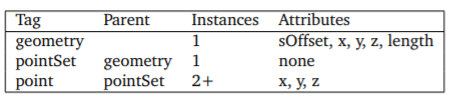


Рисунок 12 – Структура геометрий

Описание формата хранится в protobuf файлах и содержит в себе следующие элементы: дороги, полосы, соединения полос, пешеходные переходы, парковочные зоны, светофоры, знаки «Стоп» и другие.

Была подробнее рассмотрена структура дорог (рисунок 13). Атрибут Road Link позволяет описать отношения между ближайшими дорогами, указать, являются ли они предыдущими, последующими или параллельными в отношении текущей описываемой дороги. В атрибуте Road Lanes перечисляются ссылки на объекты полос дороги, а в Road Objects и Road Signals ссылки на находящиеся на текущей дороге дорожные объекты (знаки, пешеходные переходы и т.д.) и светофоры соответственно.

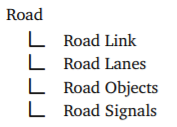


Рисунок 13 – Структура объекта дорога

Объект «Полоса» состоит из геометрического описания ее правой и левой границ, а также центральной линии. Более подробная структура объекта «Полоса» представлен на рисунке 14.

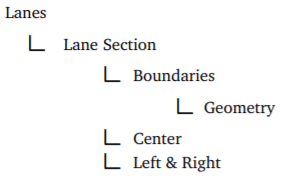


Рисунок 14 – Структура объекта полоса

## Анализ рынка существующих решений

Навигационные (гражданские) карты развиваются длительное время. Некоторые компании, поставляющие гражданские карты, заинтересовались нишей продуктов для беспилотного транспорта. Помимо крупных компаний, как TomTom, также есть некоторое количество стартапов, которые заинтересованы в создании HD-карт или реализации других продуктов, с ними связанных. Были рассмотрены следующие компании.

Американская компания **comma.ai**, наиболее популярным продуктом которой является – система автоматизированного управления автомобилем с открытым исходным кодом, также предлагает приобрести 65 TB видеоданных и данных автомобиля (CAN, IMU, сырые данные GNSS) или скачать пробный набор данных объемом 1 ТБ. Эти данные могут быть использованы для обучения или валидации модулей создания HD-карт.

17 августа 2018 года comma.ai опубликовали статью «HD Maps for the Masses», где рассказали о новом сервисе «comma maps». Данный сервис был заявлен как проект с открытым исходным кодом. К сожалению, он не был опубликован, поэтому оценить планируемый функционал мы можем только благодаря превью, опубликованном в той же статье, а также некоторым материалам в их официальном блоге. Компания заявляла, что данный сервис позволит отобразить пройденный путь, разметку дорог, а также некоторые элементы окружения.

На рисунке 15 приведен интерфейс «comma maps». Анализируя его, можно сделать вывод, что данный сервис позволяет только визуализировать собранную информацию, но не модифицировать ее. Также на странице «Полосы» мы видим, что интерфейс отображает лишь расположение разметки, но не ее тип.



Рисунок 15 – Сервис «comma maps»

Компания **TomTom** — нидерландская компания, поставщик устройств для автомобильной и персональной GPS-навигации, которые также разрабатывают вспомогательные сервисы для беспилотных автомобилей. Одним из них является сервис «ADAS Map», который позволяет получить информацию о наклоне дороги, максимальной разрешенной скорости, ее изгибах и количестве полос. Данный сервис подходит для автомобилей первого и второго уровня автоматизации. Для автомобилей второго уровня и выше TomTom предлагает более продвинутые карты «HD Map with RoadDNA», позволяющие получить более подробные данные о разметке дорог, их геометрии, дорожных знаках, соединении дорог и прочем. RoadDNA – это набор слоев локализации на HD-картах компании TomTom, который позволяет достичь более точную локализацию беспилотного автомобиля.

Также компания TomTom разработала сервис «AutoStream», который позволяет получать последние версии HD и ADAS карт с сервера TomTom. К сожалению, примеров HD-карт компании TomTom нет в открытом доступе, также нет пробных версий их сервисов. На рисунке 16 приведена визуализация этапов распознавания для построения HD-карт компании TomTom.

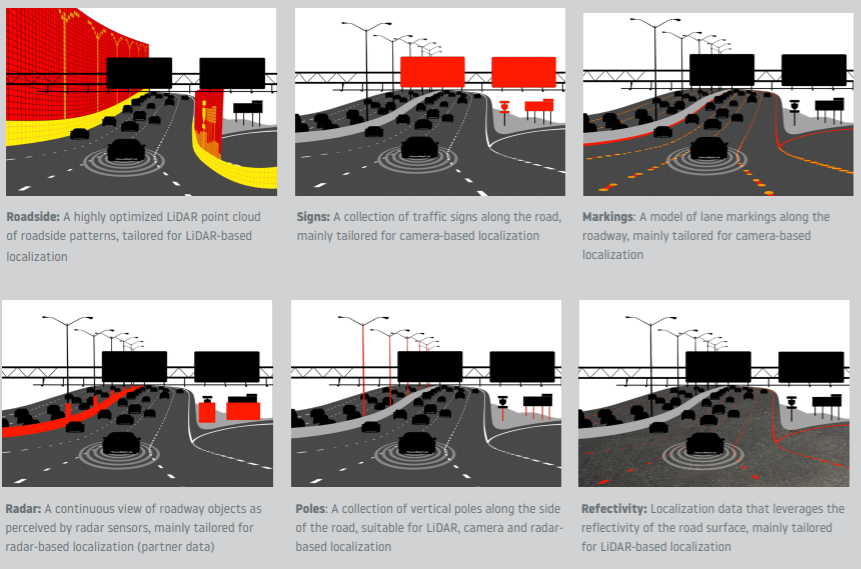


Рисунок 16 – Визуализация этапов распознавания для построения HD-карт компании TomTom.

**MathWorks** – американская компания, специализируется на разработке программного обеспечения для математических вычислений и имитационного моделирования. Одним из их продуктов является «RoadRunner» - интерактивный редактор, позволяющий создавать 3D сцены для симуляций и тестирования автоматизированных систем управления ТС. С помощью него на сцене можно нанести дорожную разметку, знаки, светофоры, ограждения, здания и другие 3D объекты. «RoadRunner» также поддерживает визуализацию лидарных облаков точек, аэрофотосъемки, GIS информации. 3D сцены могут быть экспортированы в следующие форматы: FBX, glT, OpenFlight, OpenSceneGraph, OBJ, USD.

Также сцена может быть преобразованы в HD-карту следующих форматов: OpenDRIVE, Apollo OpenDRIVE, формат NVIDIA DRIVE Sim и другие. Для данного продукта можно запросить пробную версию, запрос был направлен, но ответ не получен (в том числе автоматический).

На рисунке 17 продемонстрирован интерфейс ПО «RoadRunner».



Рисунок 17 – Интерфейс «RoadRunner»

В таблице 1 приведены результаты сравнения приведенных выше продуктов.

Таблица 1 – Таблица сравнения продуктов HD-картографии.

|  |  |  |  |
| --- | --- | --- | --- |
| **Продукт**  **Пункт  сравнения** | **Comma maps (comma.ai)** | **HD Map with RoadDNA (TomTom)** | **RoadRunner (MathWorks)** |
| Статус продукта | Неизвестен | Закончен | Закончен |
| Есть пробная версия | Нет | Нет | Да |
| Есть открытый доступ | Нет | Нет | Нет |
| Модифицируемость | Неизвестно | Нет | Нет |
| Визуализация данных | Да | Да | Да |
| Позволяет редактировать карту | Нет | Нет | Да |
| Поддерживает формат Apollo OpenDRIVE | Нет | Нет | Да |

Проанализировав все выше сказанное, можно сделать вывод, что большинство продуктов, доступных на данный момент, это готовые HD-карты. Наиболее покрыты Европа и США, данные по российским дорогам в свою очередь практически отсутствуют и их качество неудовлетворительно. Таким образом карты невозможно редактировать. Также рассмотренные продукты имеют закрытый исходный код, поэтому их невозможно доработать под собственные нужды.

## Обоснование и выбор методики, технологии и инструментальных средств разработки программного продукта

Для решения задачи по разработке графического редактора высокоточных карт беспилотного транспорта необходимо провести анализ методик, технологий и инструментальных средств разработки.

В разработке десктопных приложений выделяют 6 основных парадигм (подходов):

1. Императивное программирование
2. Декларативное программирование
3. Структурное программирование
4. Функциональное программирование
5. Логическое программирование
6. Объектно-ориентированное программирование

Для разработки ПО HDMapCreator был выбран объектно-ориентированный подход. Данный подход позволяет решить следующие проблемы:

1. уменьшение сложности программного обеспечения;
2. повышение надежности программного обеспечения;
3. обеспечение возможности модификации отдельных компонентов программного обеспечения без изменения остальных его компонентов;
4. обеспечение возможности повторного использования отдельных компонентов программного обеспечения.

Основные понятия данного подхода включают в себя понятие **класс** – это абстракция множества сущностей реального мира, объединенных общностью структуры и поведения. **Объект** – это элемент класса, то есть абстракция определенной сущности.

После выбора подхода необходимо выбрать язык программирования. К нему предъявляются следующие требования:

1. поддержка Protocol buffer;
2. входит в набор технологий, используемых проектом OSCAR;
3. высокая скорость разработки;

Согласно первому требованию нам подходят следующие языки: C++, C#, Dart, Go, Java, Kotlin, Python. Из перечисленных языков программирования в проекте OSCAR используются: C++, Python. Был выбран язык программирования Python, так как он имеет более высокую скорость разработки. Для реализации графического интерфейса был выбран фреймворк Qt, а именно библиотека PyQt [3,4]. Данный фреймворк обладает значительным количеством виджетов (выпадающие списки, поля ввод, диалоговые окна и т.д.), которые можно настраивать в соответствии с дизайн-проектом приложения, а также расширять их функциональные возможности с помощью сигналов – уведомлений о взаимодействии, посылаемых виджетами.

Также необходимо подобрать оптимальные инструментальные средства разработки.

Интегрированная среда разработки (ИСР) – это система программных средств, используемая программистам для разработки программного обеспечения. В английском языке такая среда называется Integrated development environment или сокращённо IDE.

ИСР обычно включает в себя текстовый редактор, компилятор, интерпретатор, средства автоматизации разработки и сборки программного обеспечения, отладчик и средства для интеграции с системами управления версиями. Большинство современных ИСР предназначены для разработки программ на нескольких языках программирования.

Система управления версиями (английская аббревиатура CVS) - программное обеспечение для облегчения работы с изменяющейся информацией. Она позволяет хранить несколько версий одного и того же документа, при необходимости, возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение и многое другое.

Были рассмотрены наиболее популярные среды разработки, имеющие поддержку языка программирования Python:

1. Atom – ИСР, разработанная компанией GitHub на основе платформы Electron. Благодаря этому данную ИСР можно в значительной степени настраивать посредством установки расширений. Имеет открытый исходный код и распространяется бесплатно.
2. Visual Studio Code (VS Code) – аналог Atom, разработанный компанией Microsoft, так же основан на платформе Electron. Обладает рядом преимуществ, например, реализовано удобное сравнение файлов, подсказки в текстовом редакторе более качественные благодаря технологии IntelliSense, и, если в Atom поиск реализован по конкретному файлу, то в Visual Studio Code есть также поиск по всем файлам одновременно. Visual Studio Code имеет как браузерную, так и десктопную версию. Имеет открытый исходный код и распространяется бесплатно.
3. PyCharm – ИСР, разработанная компанией JetBrains специально для языка Python на основе IntelliJ IDEA. Данная ИСР имеет функции контроля качества кода в соответствии с требованиями PEP8, автодополнения и автоматического рефакторинга. Данная среда разработки распространяется в двух лицензиях: профессиональной и лицензии сообщества. Они отличаются стоимостью и набором функций.

Исходя из перечисленных выше характеристик была выбрана ИСР Visual Studio Code.

# Специальная часть

## Постановка задачи

Опишем более детально мотивацию разработки HDMapCreator. Сфера беспилотного транспорта находится в фазе активного роста, но если мы будем говорить об HD-картографии, то стоит уточнить, что она только в начале своего развития. На данный момент представлено достаточно мало коммерческих решений для создания HD-карт, а решений с открытой лицензией нет вовсе.

Главными требованиями проекта OSCAR к картографическому приложению является возможность как создавать, так и редактировать карты, поддержка формата Apollo OpenDRIVE, модифицируемость приложения, удобный пользовательский интерфейс, который не будет требовать специфических знаний и умений. Основываясь на требовании модифицируемости, было решено разрабатывать приложение самостоятельно. Целью разработки является создание удобного инструмента, который позволит быстро создавать и редактировать HD-карты для проекта OSCAR.

Классы объектов, которые должны быть реализованы в HDMapCreator, являются:

1. Дороги и полосы
2. Знаки «Стоп»
3. Светофоры
4. Парковочные места
5. Перекрестки
6. Стоп-линии
7. Точки интереса
8. Пешеходные переходы

Введем общее обозначение этих классов – дорожные объекты. Каждый дорожный объект состоит из уникального идентификатора, информационных полей и набора точек, его образующих.

HDMapCreator должен обладать следующей базовой функциональностью:

1. Возможность чтения данных из файлов TXT и BIN в соответствии с форматом Apollo OpenDRIVE;
2. Возможность создания дорожных объектов;
3. Возможность редактирования дорожного объекта:
   1. Изменение количества точек и их положения;
   2. Изменение уникального идентификатора;
   3. Изменение информационных полей;
4. Возможность сохранения карты в TXT файл в формате Apollo OpenDRIVE, а файлов проекта в JSON файл;
5. Возможность экспортировать карту, т.е. подготовить routing-файл и файлы симуляции;

Языком интерфейса был выбран английский язык, так как рассматривается возможность публикации данного программного продукта с открытой лицензией.

## Архитектура и описание структуры разрабатываемого приложения

Данное приложение имеет модульную архитектуру [5,6]. Модули можно разделить на функциональные и модули хранения информации. Межмодульное взаимодействие организовано посредством пакетов данных, которые могут содержать также команды, изображения. Архитектура HDMapCreator приведена на рисунке 18.

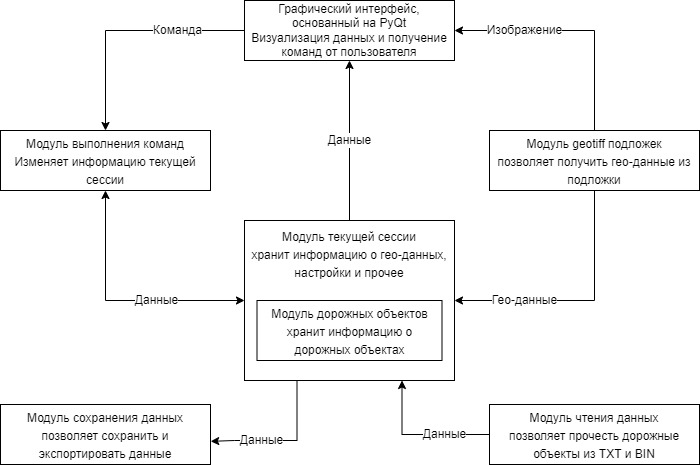


Рисунок 18 – Архитектура HDMapCreator

Модуль текущей сессии хранит следующие данные:

1. Гео-данные. GeoTiff - открытый формат представления растровых данных в формате TIFF совместно с метаданными о географической привязке (геореференцированный растр). В нашем случае это коэффициент масштабирование и координаты левого верхнего угла в системе координат UTM. Система координат UTM – применяемая в геодезии и картографии система координат, разделяющая Землю на 60 вытянутых в меридиональном направлении зон шириной 6 градусов (максимальная ширина зоны 800 км) и отображающая их по отдельности в равноугольной поперечно-цилиндрической проекции Меркатора.
2. Данные о дорожных объектах, которые могут быть получены из TXT или BIN файлов в соответствие с форматом Apollo OpenDRIVE, а также в результате взаимодействия пользователя с графическим интерфейсом.
3. Настройки приложения: цвет отображаемого текста, количество запоминаемых действий пользователя, определенные пользователем папки хранения данных.

Была составлена диаграмма прецендентов использования, приведенная на рисунке 19 [7,8].

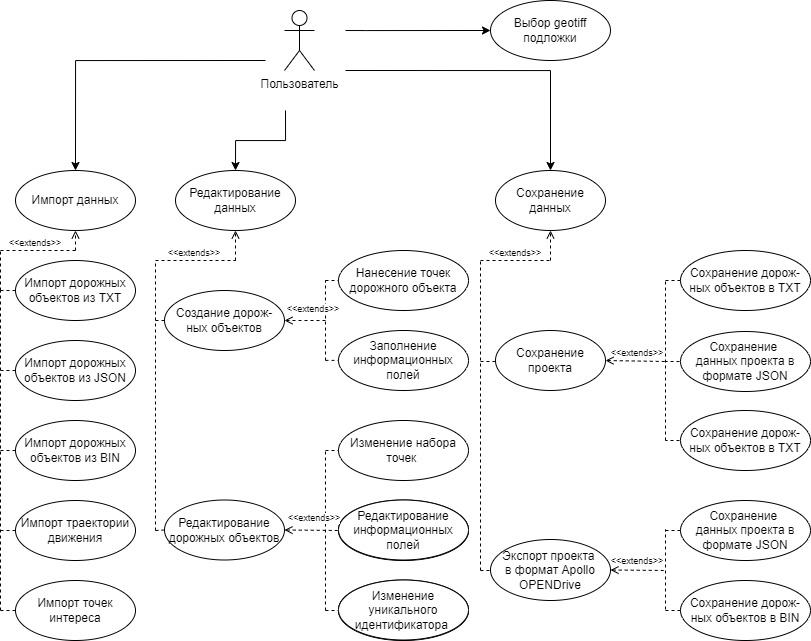


Рисунок 19 – Диаграмма прецендентов использования HDMapCreator.

Пользователю доступен импорт данных из большого числа источников дорожные объекты можно загружать из TXT, BIN и JSON, в последнем также хранятся данные проекта: созданные пользователем папки, настройки отображения и экспорта дорожных объектов. Также можно загрузить данные о точках интереса из файла формата TXT и данное о пройденной траектории для быстрого построения полос.

При создании дорожного объекта пользователь должен заполнить информационные поля, а также нанести точки дорожного объекта (подробнее рассмотрим в разделе «2.3 Объектно-ориентирование проектирование приложения»). Уникальный идентификатор присвоится автоматически, его в последствии можно будет отредактировать, как и другую информацию о дорожном объекте, с помощью графического интерфейса.

Сохранение данных бывает двух типов: простое, при котором дорожные объекты сохраняются в формате TXT и JSON, и экспортирование, когда помимо простого сохранения также выполняется создание файлов симуляции и маршрутизации.

## Объектно-ориентированное проектирование приложения

Все дорожные объекты реализуют интерфейс Shape, который включает в себя функцию отрисовки дорожного объекта в окне графического редактора, функцию парсинга данных из формата Google protobuf, функцию приведения данных к формату Google protobuf, функцию заполнения полей дорожного объекта с помощью данных в формате json и функцию приведения данных к формату json.

Связь Overlap (пересечение) позволяет сообщить, что некоторые дорожные объекты связаны. Следующие типы дорожных объектов могут быть включены в связь Overlap (пересечение): полосы, перекрестки, пешеходные переходы, знаки «Стоп», светофоры и парковочные места.

Для дорожного объекта Lane (полоса) также определены две другие связи: Neighbor, связывающая соседние полосы, и Relation, позволяющая определить предка и потомка.

Объектно-ориентированное проектирование приложение приведено в виде диаграммы классов дорожных объектов на рисунке 20 [7,8].

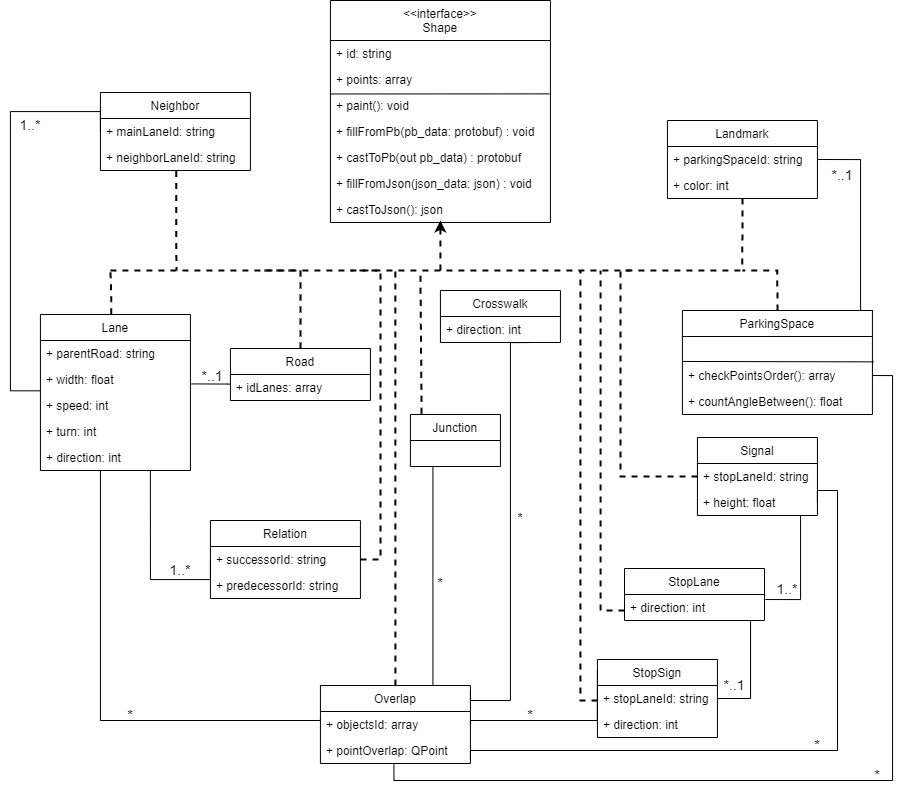


Рисунок 20 – Диаграмма классов дорожных объектов

## Разработка пользовательского приложения

Перед разработкой графического интерфейса были проанализированы популярные графические редакторы. Результаты анализа приведены ниже.

Adobe Photoshop – многофункциональный графический редактор, разрабатываемый и распространяемый компанией Adobe Systems. У данного программного продукта реализована удобная панель инструментов, выделенная на рисунке 21, было решено реализовывать меню «Создание дорожных объектов» по образу и подобию.

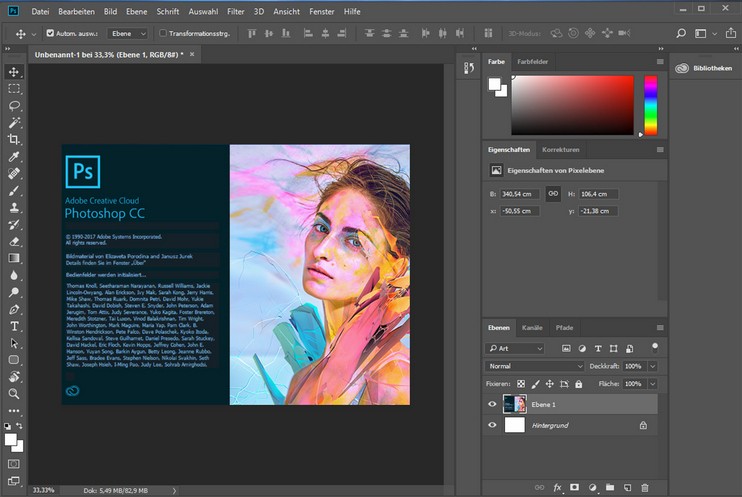


Рисунок 21 – Интерфейс Adobe Photoshop

Blender (рисунок 22) — профессиональное свободное и открытое программное обеспечение для создания трёхмерной компьютерной графики, включающее в себя средства моделирования, скульптинга, анимации, симуляции, рендеринга, постобработки и монтажа видео со звуком, компоновки с помощью «узлов», а также создания 2D-анимаций. В этом программном продукте меня заинтересовала правое боковое меню.

Элемент, отмеченный на рисунке 22 зеленым цветом, стал образцом для панели «Обзор», он позволяет отобразить взаимосвязи между объектами, а определение иконок для различных типов объектов, позволяет их наглядно отделить друг от друга.

Элемент, отмеченный на рисунке 22 оранжевым цветом, отображает подробную информацию об объекте и стал прототипом меню «Подробнее».

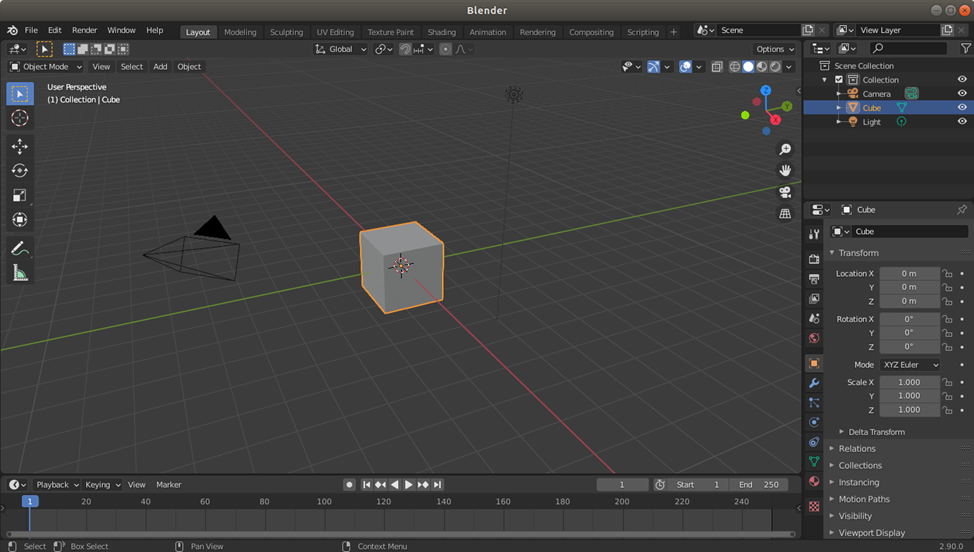


Рисунок 22 – Интерфейс Blender

Далее описана реализация графического интерфейса HDMapCreator. Он состоит из 4 основных панелей (меню).

На панели «Холст», приведенной на рисунке 23, отображаются дорожные элементы, с которыми пользователь может взаимодействовать, например перемещать точки или, вызвав контекстное меню правой кнопкой мыши, добавить новые или удалить.

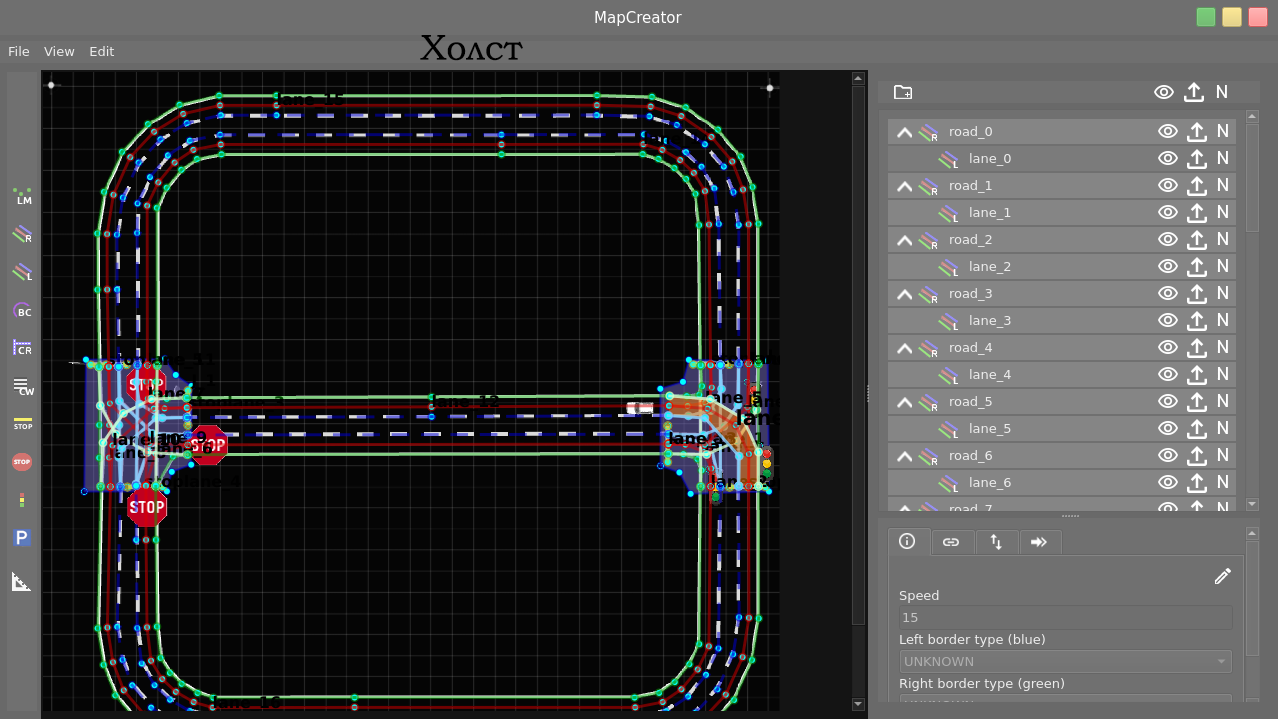


Рисунок 23 – Панель «Холст»

Меню «Создание дорожных объектов», приведенное на рисунке 24, содержит в себе 9 кнопок-иконок для создания дорожных объектов и 2 инструмента: линейку для измерения расстояния в координатной системе мира (результат выводится в м) и инструмент «кривые Безье», позволяющий удобно строить полосы-повороты.



Рисунок 24 – Меню «Создание дорожных объектов»

Меню «Обзор», которое приведено на рисунке 25, отображает имена и типы дорожных объектов, их вложенность в другие дорожные объекты (например, полосы должны быть вложены в дорогу), а также предоставляет доступ к следующим функциональным возможностям: включение/выключение видимости объекта на карте, его уникального идентификатора и разрешение/запрет экспортирования элемента. В случае если у элемента, к которому применяются правила видимости или экспорта, есть дочерние элементы, то к ним будут применены такие же правила. Например, если дорогу road\_236 сделать невидимой, то и полоса lane\_236 тоже будет невидимой, как показано на рисунке 26.

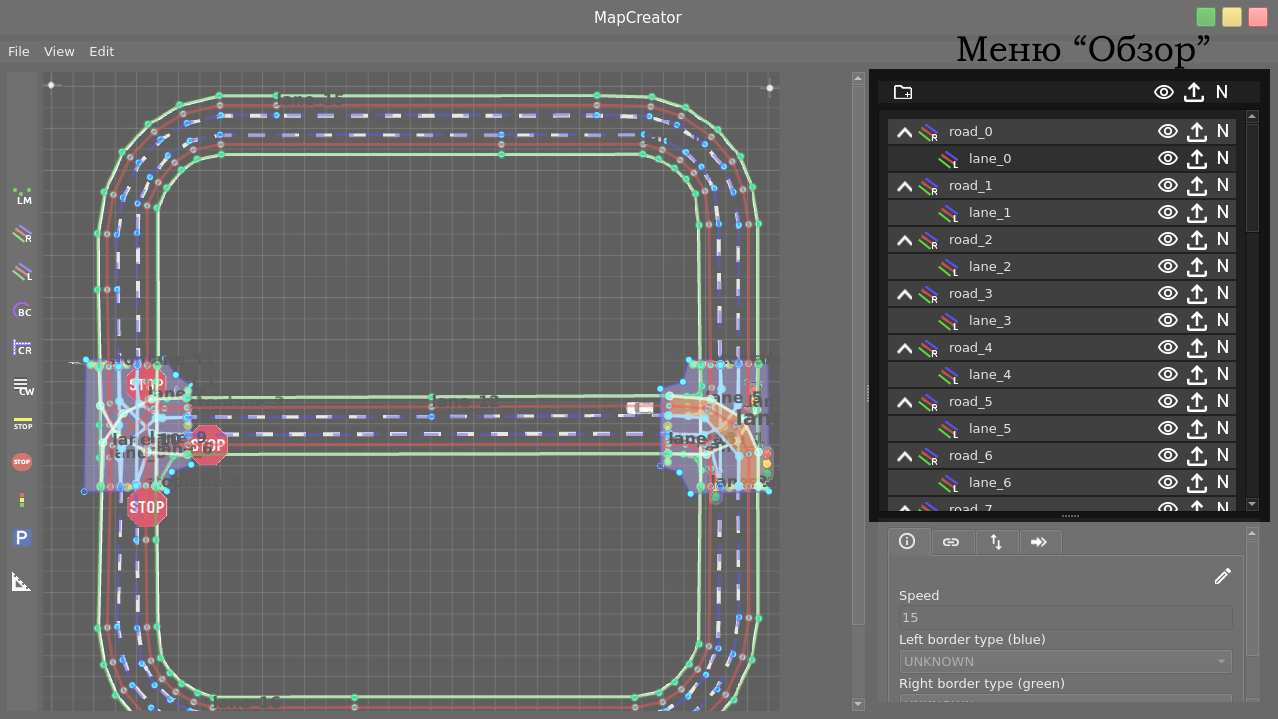


Рисунок 25 – Меню «Обзор»



Рисунок 26 – Пример отображения дорожных элементов в меню «Обзор»

Также у меню «Обзор» есть небольшая панель инструментов, которая позволяет применить правила видимости и экспорта ко всем дорожным элементам, а также предоставляет возможность создавать папки. С помощью папок пользователь может группировать дорожные объекты на свое усмотрение.

Меню «Подробнее» работает в нескольких режимах:

1. Режим создания дорожного элемента. После взаимодействия с кнопками меню «Создание дорожных элементов» на панели «Подробнее» отображаются поля ввода данных о дорожном объекте.
2. Режим просмотра информации. На панели отображаются информационные поля дорожного объекта, а также данные о связях с другими объектами на отдельных вкладках, соответствующих типу связи.
3. Режим редактирования дорожного элемента. На панели отображаются поля с текущими данными дорожного объекта, которые можно редактировать.

Меню «Подробнее» представлено на рисунке 27.

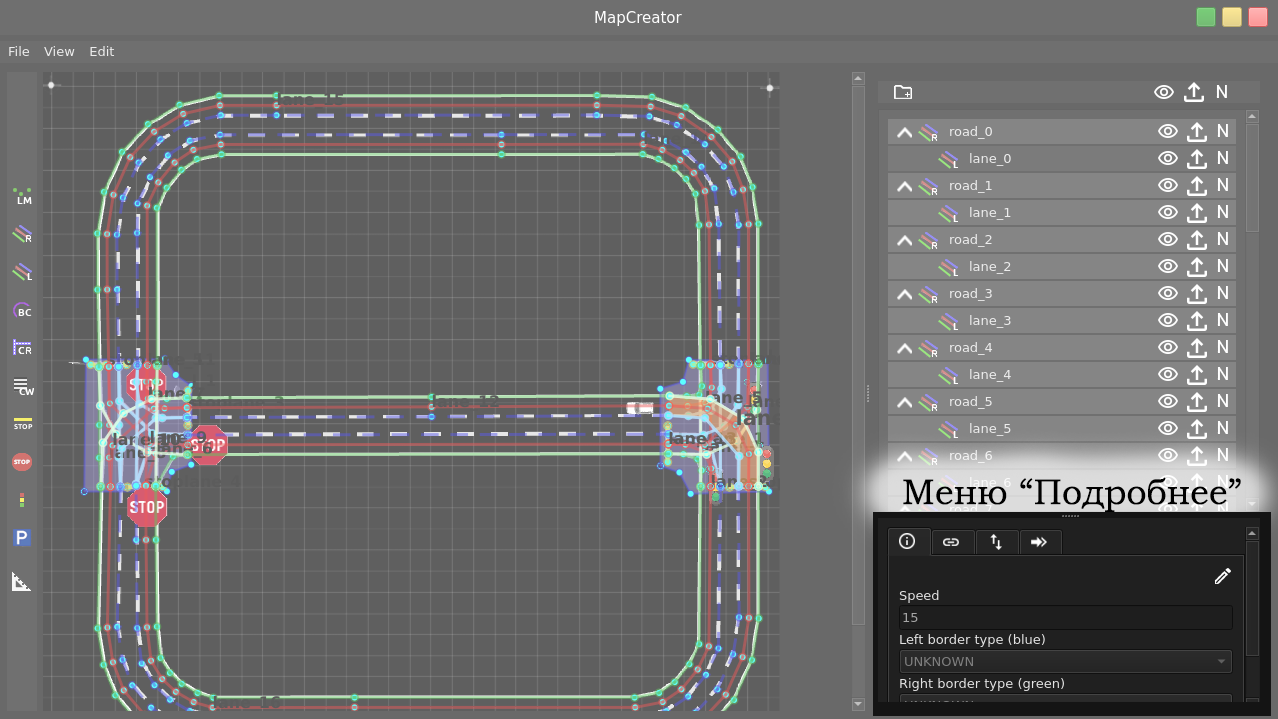


Рисунок 27 – Меню «Подробнее»

Меню «Создание связи с другими дорожными объектами» входит в панель «Подробнее». Каждому из 3 типов связи выделена отдельная вкладка. Добавление и удаление связей производится по одному алгоритму, поэтому рассмотрим на примере вкладки Overlap (пересечение), приведенной на рисунке 28.

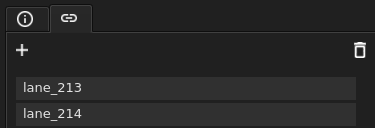


Рисунок 28 – Вкладка «Overlap» объекта типа «Пешеходный переход»

С помощью иконки «Плюс» пользователь может открыть меню создания новой связи. Для удаления объекта «Overlap» необходимо нажать на id элемента (выделится более темным), а затем на иконку мусорного ведра и подтвердить свое действие в диалоговом окне, приведенном на рисунке 29. Чтобы выделить несколько элементов необходимо зажать клавишу «Ctrl» и нажать на все элементы, которые надо удалить.

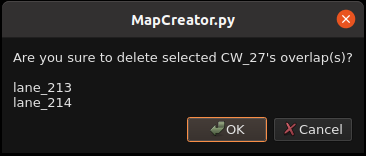


Рисунок 29 – Пример диалога-подтверждения удаления элементов.

Также при взаимодействии с панелью «Холст» у пользователя есть возможность отменить произведенное действие (далее будем обозначать как функцию undo) и вернуть изменения (далее будем обозначать как функцию redo). Также введем обозначение экземпляр A (сокр. от ActionStack) – экземляра класса, в котором реализовано хранение пользовательских данных, а также функции undo и redo. Данный объект будет содержать в себе два списка \_inner\_list (совершенные пользователем действия) и \_inner\_list\_canceled (отмененные пользователем действия).

На рисунке 30 слева на право приведены следующие алгоритмы: алгоритм сохранения изменений, алгоритм отмены изменений (undo) и алгоритм возвращения отмененных изменений (redo). Функция enableActions «размораживает» графический интерфейс и предоставляет пользователю возможность продолжить редактирование.

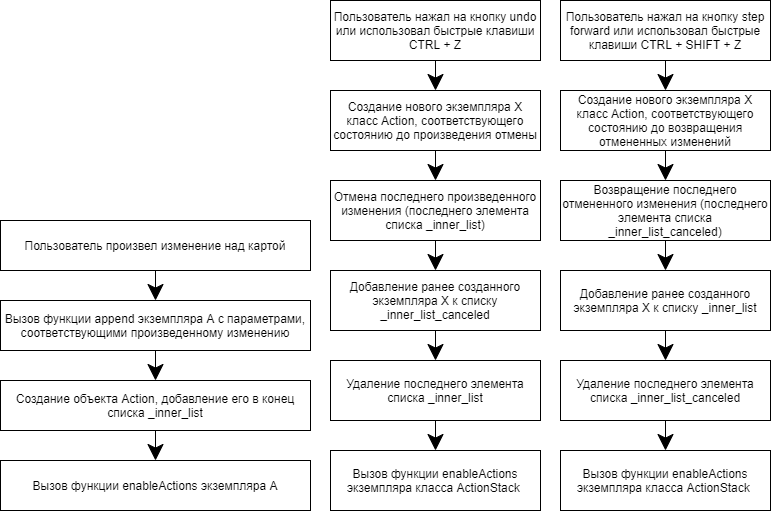


Рисунок 30 – Алгоритмы сохранения изменений, отмены изменений (undo) и возвращения отмененных изменений (redo).

Меню настроек программного продукта состоит из 4 вкладок. Первая вкладка, приведенная на рисунке 31, отвечает за настройки редактора и содержит только одну настройку – цвет отображаемого текста

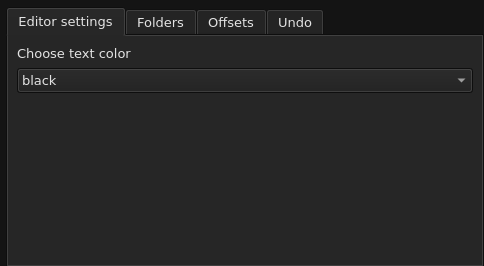


Рисунок 31 – Вкладка «Настройки редактора»

Вторая вкладка, приведенная на рисунке 32, отвечает за определенные пользователем папки. Это позволяет сократить время на поиск нужного файла. Так пользователь может сохранить директории для geotiff-подложек, записей траекторий проезда, для сохранения и экспорта проекта.

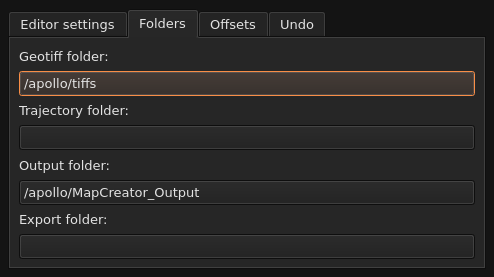


Рисунок 32 – Вкладка «Настройки папок»

Третья вкладка, приведенная на рисунке 33, позволяет пользователю редактировать смещения, загруженные с geotiff-подложки, также можно указать среднюю высоту местности над уровнем моря и поворот подложки.

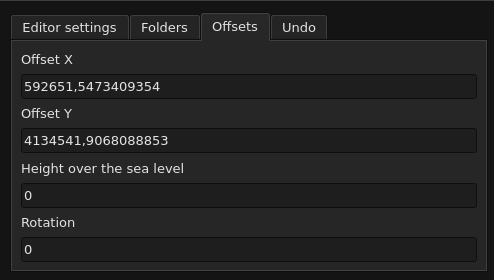


Рисунок 33 – Вкладка «Настройки географических параметров»

Последняя вкладка, приведенная на рисунке 34, позволяет пользователю указать, сколько его действий нужно хранить для функций undo и redo.

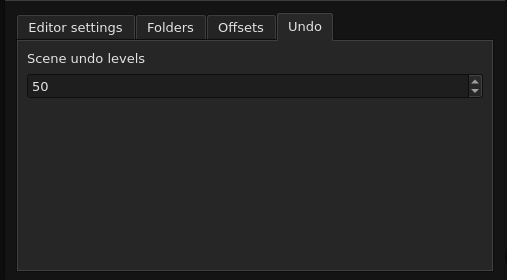


Рисунок 34 – Вкладка «Настройки сохраняемых действий»

## Текст программы

Текст программы представляет собой документ, выполненный машинным способом. Текст программы приведен в приложении Б. Текст программы выполнен в соответствии с нормами языка программирования Python [9, 10].

## Тестирование разработанного приложения

После окончания разработки программного продукта, было проведено его функциональное тестирование. Функциональное тестирование – это вид тестирования, целями которого является проверка способности программного обеспечения решать задачи, необходимые пользователям. Часть набора тестов представлен в таблице 2.

Таблица 2 – Тестирование приложения

| **№** | **Название** | **Шаги** | **Ожидаемый результат** | **Результат** |
| --- | --- | --- | --- | --- |
| 1 | Запуск приложения в Docker-контейнере | Запустить приложение командой oscar mapcreator start внутри Docker-контейнера | Открытие главного меню. Активированные кнопки выбора подложки, загрузки данных, настроек. | Пройден |
| 2 | Запуск приложения вне Docker-контейнере | Запустить приложение командой oscar mapcreator start вне Docker-контейнера | Открытие главного меню. Активированные кнопки выбора подложки, загрузки данных, настроек. | Пройден |
| 3 | Открытие geotiff-подложки | В меню Файл выбрать пункт «Открыть подложку», в диалоговом окне выбрать файл | На холсте графического редактора отображается черно-белая подложка | Пройден |
| 4 | Загрузка данных из TXT файла | В меню Файл выбрать пункт «Импорт из txt», в диалоговом окне выбрать файл | Отображение дорожных элементов в меню «Обзор», на холсте графического редактора | Пройден |
| 5 | Загрузка данных из BIN файла | В меню Файл выбрать пункт «Импорт из BIN», в диалоговом окне выбрать файл | Отображение дорожных элементов в меню «Обзор», на холсте графического редактора | Пройден |
| 6 | Загрузка данных из JSON файла | В меню Файл выбрать пункт «Импорт из JSON», в диалоговом окне выбрать файл | Отображение дорожных элементов в меню «Обзор», на холсте графического редактора | Пройден |
| 7 | Создание дорожного объекта | Выбрать дорожный объект, заполнить информационные поля и нанести точки, подтвердить создание | Отображение нового дорожного элемента в меню «Обзор», на холсте графического редактора | Пройден |
| 8 | Редактирование дорожного объекта | В меню «Обзор» выбрать дорожный элемент, активировать режим редактирования, изменить информацию | В меню «Подробнее» отображается измененная информация. | Пройден |
| 9 | Сохранение проекта | В меню Файл выбрать пункт «Сохранить проект», выбрать папку | В папке появились файлы TXT и JSON | Пройден |
| 10 | Экспорт проекта | В меню Файл выбрать пункт «Экспортировать проект», выбрать папку | В папке появились файлы TXT, JSON и BIN | Пройден |

В ходе тестирования программа справилась со всем поставленными задачами. В каждом тесте результат работы программного продукта соответствовала ожидаемому.

## Оценка качества

Разработанное приложение соответствует требованиям проекта OSCAR и реализует весь необходимый набор функция для быстрого создания высокоточных карт беспилотного транспорта в формате Apollo OpenDRIVE. В данном программном продукте заложена возможность последующего дополнения функциональных возможностей, добавления новых форматов сохранения карт. Относительно имеющегося функционала приложение имеет полную конкурентную способность с аналогичными продуктами.

Графический редактор HDMapCreator соответствует современным требованиям к пользовательскому интерфейсу, он интуитивно понятен и не требует специальных знаний и навыков для использования.

1. **Экономическая часть**
   1. **Область применения программного продукта и его преимущества перед аналогичным программным продуктом**

Приложение предназначено для пользования сотрудниками ООО «НПО «СтарЛайн»» для создания и редактирования высокоточных карт беспилотного транспорта.

Достоинствами программного продукта является возможность чтения данных как из текстовых, таки бинарных файлов, дружелюбный пользовательский интерфейс, а также отсутствие регистрации пользователей, что упрощает работу с приложением и ускоряет процессы создания и редактирования карт.

* 1. **Трудоемкость разработки программного продукта, квалификация исполнителя и его оклад**

Трудоемкость разработки можно определить в таблице 3. Строка «Всего» отображает общую трудоемкость разработки.

Таблица 3 – Трудоемкость разработки программного продукта

| **Наименование этапа** | **Условное обозначение** | **Трудоемкость выполнения этапа, час** |
| --- | --- | --- |
| Описание задания | То | 30 |
| Разработка UML - диаграмм | Тд | 30 |
| Проектирование программы | Тпп | 370 |
| Отладка программы | Топ | 30 |
| Оценка качества программы | Тоц | 20 |
| Оформление документации | Тд | 50 |
| Внедрение программного продукта | Твпп | 176 |
| Всего | Тобщ | 706 |

Разработчики программного продукта представлены в таблице 4.

Таблица 4 – Разработчики программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| **Исполнители** | **Оклад, руб.** | **Часовая тарифная ставка, руб. /час** | **Количество сотрудников** |
| Разработчик-программист | 40 000 | 227,27 | 1 |
| Руководитель  проекта | 70 000 | 397,73 | 1 |
| Инженер по внедрению | 30 000 | 170,45 | 1 |

Часовая тарифная ставка ЧТС, руб./час., определяется исходя из месячного оклада, количества рабочих дней в месяце и продолжительности рабочего по формуле:

(1)

где Ом – оклад исполнителя в месяц, руб. /мес.;

Д – количество рабочих дней в месяце (для расчета Д = 22 раб. дня);

Тс – продолжительность рабочего дня (для расчета Тс = 8час.)

В таблице 5 показана стоимость технических средств для разработки системы.

Таблица 5 – Стоимость технических средств разработки

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование компонента** | **Цена, руб.** | **Количество, шт.** | **Стоимость, руб.** |
| Ноутбук MSI Katana GF76 11UC-679XRU | 82340 | 1 | 82340 |
| Мышь Defender Optimum MB-270 | 179 | 1 | 179 |
| МФУ HP DeskJet 2320 | 3490 | 1 | 3490 |
| Итого: | | | 86009 |

В таблице 6 представлены затраты на расходные материалы.

Таблица 6 – Планируемые затраты на расходные материалы (Р)

| **Затраты** | **Стоимость** | **Количество** | **Сумма, руб.** |
| --- | --- | --- | --- |
| Интернет | 550 руб./месяц | 4 месяца | 2 200 |
| Электричество | 4,85 руб./КВт\*ч | 1593 КВт\*ч | 7726,05 |
| Бумага | 450 руб. | 1 упаковка | 450 |
| Ручка | 80 руб. | 3 штуки | 240 |
| Заправка МФУ | 1890 руб. | 1 раз | 1890 |
| Итого: | | | 12 506,05 |

* 1. **Расчет затрат на разработку**

Исходные данные, связанные с разработкой программного продукта, приведены в таблице 7.

Таблица 7 – Исходные данные

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Оклад разработчика | Ор | 110 000 руб. |
| Время разработки | Трп | 3 месяца |
| Машинное время разработки | Тмч | 2,5 месяца |
| Коэффициент дополнительной заработной платы | Кд | 0,17 |
| Коэффициент страховых взносов | Кст | 0,3 |
| Количество единиц техники | Q | 1 шт. |
| Себестоимость содержания техники | См/ч | 25 руб./час |
| Коэффициент готовности техники | Кгт | 0,95 |
| Число рабочих дней в месяце | ЧРД | 22 дня |
| Продолжительность смены | Тст | 8 часов |
| Коэффициент сменности | Ксм | 1 |
| Коэффициент транспортных расходов | Кт | 0 |
| Коэффициент накладных расходов | Кнр | 0,53 |

Расчет полных затрат на разработку проектного решения (программного продукта) осуществляется по формуле:

Зрп = Зот + Зст + Зэвм + Зсп + Зрм + Знр, (2)

где Зот – затраты на оплату труда разработчика (разработчиков), руб.;

Зст – страховые взносы по оплате труда во внебюджетные фонды, руб.;

Зэвм – затраты, связанные с содержанием вычислительной техники, руб.;

Зсп – затраты на технические средства, руб.;

Зрм – затраты на расходные материалы, необходимые при разработке программного продукта, руб.;

Знр – затраты по накладным расходам, приходящиеся на разработку программного продукта, руб.

* + 1. **Затраты на оплату труда разработчиков (Зот), руб.**

Размер фонда оплаты труда разработчиков (Зот) рассчитывается по формуле:

Зот = Ор Трп (1+Кд), (3)

где Ор – месячный оклад разработчика проектного решения, руб./мес.;

Трп – время разработки проектного решения разработчиком, мес.(час.) включает в себя машинное время работы над проектом (Тмрп);

Кд – коэффициент дополнительной заработной платы разработчика.

* + 1. **Затраты по страховым вносам (Зст), руб.**

Сумма страховых взносов определяется по формуле:

Зст = Кст Зот, (4)

где Кст – коэффициент страховых взносов для расчета отчислений во внебюджетные фонды.

* + 1. **Затраты по содержанию ЭВМ (Зэвм), руб.**

Затраты, связанные с эксплуатацией и содержанием ЭВМ, определяются по формуле:

Зэвм = Тмрп Кгт Q Cм/ч, руб. (5)

где Тмрп – машинное время на разработку проектного решения, час;

Кгт – коэффициент готовности техники;

Q – количество условных единиц, используемой техники;

Cм/ч – стоимость машино-часа, эксплуатации оборудования, руб. в час.

Так как машинное время может измеряться в месяцах, а себестоимость машино-часа за один час, то машинное время необходимо перевести в часы.

Перевод рабочего времени в часы осуществляется по формуле:

Тмрп = Тмч Чрд Тсм Ксм, (6)

где Тмч – рабочее время в месяцах;

Чрд – число рабочих дней в месяце;

Тсм – продолжительность рабочей смены;

Ксм – количество рабочих смен.

* + 1. **Затраты на расходные материалы (Зрм), руб.**

Затраты на расходные материалы необходимые для разработки проектного решения определяются по формуле:

Зрм = , руб (7)

затраты на расходные материалы, руб.;

Кт – коэффициент транспортных расходов.

* + 1. **Затраты по накладным расходам (Знр)**

Затраты по накладным расходам определяются по формуле:

Знр = КнрЗот, руб. (8)

где Кнр – коэффициент накладных расходов, принимается для расчета по данным предприятия;

Зот – затраты по оплате труда, руб.

Таблица 8 – Результаты вычислений

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Оплата труда | Зот | 386 100 руб.; |
| Страховые взносы | Зст | 115 830 руб.; |
| Содержание ЭВМ | Зэвм | 10 450 руб.; |
| Затраты на специальные программы | Зсп | 86 009 руб.; |
| Расходные материалы | Зрм | 12 506,05 руб.; |
| Накладные расходы | Знр | 204 633 руб.; |
| Итого затрат на разработку | Зрп | 815 528,05 руб. |

Произведя вычисления было выявлено, что полные затраты на разработку составляют 815 528,05 рублей. Промежуточные результаты вычислений представлены в таблице 8.

* 1. **Расчет затрат на внедрение**

Исходные данные, связанные с разработкой программного продукта, приведены в таблице 9.

Таблица 9 – Исходные данные

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Оклад инженера по внедрению | Ор | 30000 руб. |
| Время необходимое для внедрения | Трп | 1 месяца |
| Коэффициент дополнительной заработной платы | Кд | 0,17 |
| Коэффициент страховых взносов | Кст | 0,3 |
| Затраты на разработку | Зрп | 815 528,05 руб. |
| Себестоимость содержания техники | См/ч | 25 руб./час |
| Коэффициент готовности техники | Кгт | 0,95 |
| Число рабочих дней в месяце | ЧРД | 22 дня |
| Продолжительность смены | Тст | 8 часов |
| Коэффициент сменности | Ксм | 1 |
| Коэффициент накладных расходов | Кнр | 0,53 |

Затраты на внедрение являются для организации единовременными и формируют величину капиталовложений в данный проект.

Затраты на внедрение проектного решения (Квпр) рассчитываются по формуле:

Квпр = Зрп + Зот + Зст + Зэвм + Знр, руб. (13)

где Зрп – затраты на приобретение программного обеспечения (включают стоимость разработанного ПП, а также других существующих ПП, необходимых для функционирования системы), руб.;

Зот – затраты на оплату туда работников, занятых внедрением проекта, руб.;

Зст – отчисления во внебюджетные фонды с заработной платы работников, занятых внедрением проекта, руб.;

Зэвм – затраты, связанные с эксплуатацией ЭВМ при внедрении проектного решения, руб.;

Знр – накладные расходы, руб.

* + 1. **Затраты на приобретение программного обеспечение (Зпо)**

Для внедрения разработанного проекта не требуется приобретение дополнительных специальных программ.

* + 1. **Затраты на оплату работников, занятых внедрением проекта (Зот)**

Затраты на оплату труда работников, занятых внедрением разработанного проекта, включают в себя заработную плату сотрудников, участвующих непосредственно в установке и отладке системы.

Зотв = Окл. Разработчика \* Те \* (1+Кд), руб. (14)

где Те – трудоемкость внедрения, т.е. время необходимое для внедрения проекта, час., дни, мес.;

Кд – коэффициент дополнительной заработной платы.

* + 1. **Затраты по страховым взносам (Зст)**

Сумма страховых взносов определяется по формуле:

Зст = Кст Зот, (15)

где Кст – коэффициент страховых взносов для расчета отчислений во внебюджетные фонды.

* + 1. **Затраты по содержанию ЭВМ (Зэвм), руб**

Затраты, связанные с эксплуатацией и содержанием ЭВМ, определяются по формуле:

Зэвм = Тмрп Кгт Q Cм/ч, руб. (16)

где Тмрп – машинное время на внедрение проектного решения, час;

Кгт – коэффициент готовности техники;

Q – количество условных единиц, используемой техники;

Cм/ч – стоимость машино-часа, эксплуатации оборудования, руб. в час.

Так как машинное время может измеряться в месяцах, а себестоимость машино-часа за один час, то машинное время необходимо перевести в часы.

Перевод рабочего времени в часы осуществляется по формуле:

Тмрп = Тмч Чрд Тсм Ксм, (17)

где Тмч – рабочее время в месяцах;

Чрд – число рабочих дней в месяце;

Тсм – продолжительность рабочей смены;

Ксм – количество рабочих смен.

* + 1. **Затраты по накладным расходам (Знр)**

Затраты по накладным расходам определяются по формуле:

Знр = Кнр Зотв, руб (18)

где Кнр – коэффициент накладных расходов

Зотв – затраты по оплате труда работников, занятых внедрением.

Таблица 10 – Результаты вычислений

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Обозначение** | **Значение** |
| Оплата труда | Зотв | 35 100 руб. |
| Страховые взносы | Зст | 10 530 руб. |
| Содержание ЭВМ | Зэвм | 3 344 руб. |
| Накладные расходы | Знр | 18 603 руб. |
| Программного обеспечение | Зпо | 815 528,05 руб. |
| Итого затрат на внедрение | Зрпв | 883 105,05 руб. |

Произведя вычисления было выявлено, что полные затраты на внедрение составляют 883 105,05 рублей. Промежуточные результаты вычислений представлены в таблице 10.

* 1. **Расчет цены и прибыли**

Исходные данные, связанные с разработкой программного продукта, приведены в таблице 11.

Таблица 11 – Исходные данные

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Обозначение** | **Значение** |
| Коэффициент рентабельности | Кр | 0,25 |
| Коэффициент налога на добавочную стоимость | Кндс | 0,2 |
| Ставка налога | Кнп | 0,2 |
| Стоимость затрат на разработку и внедрение | Квпр | 883 105,05 руб. |

Цена программного продукта, который разработан одной организацией по заказу другой и не предназначен для тиражирования, определяется по формуле:

Цпп = Квпр + Ппл + НДС , руб. (19)

где Ппл – планируемая прибыль рассчитывается по формуле.

Ппл = Квпр × Кр (20)

НДС – налог на добавленную стоимость определяется исходя из Кндс = 0.2 (ставка налога 20%) по формуле

НДС = (Квпр + Ппл)Кндс, руб. (21)

Каждое предприятие с полученной прибыли перечисляет государству налог на прибыль. На сегодня ставка налога 20% (Кнп = 0.2) от полученной прибыли, и определяется по формуле:

НП = Ппл Кнп, руб. (22)

Чистая прибыль, остающаяся в распоряжении предприятия

ЧП = Ппл – НП, руб. (23)

Поступление в бюджет складываются из налога на прибыль и НДС

ПБ = НП + НДС, руб. (24)

Таблица 12 – Результаты вычислений

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Обозначение** | **Значение** |
| Плановая прибыль | Ппл | 220 776,26 руб. |
| Налог на добавочную стоимость | НДС | 220 776,26 руб. |
| Цена программного продукта | Цпп | 1 324 657,57 руб. |
| Налог на прибыль | НП | 44 155,25 руб. |
| Чистая прибыль | ЧП | 176 621,01 руб. |
| Поступление в бюджет | ПБ | 264 931,51 руб. |

Произведя вычисления было выявлено, что чистая прибыль будет составлять 176 621,01 рублей. Промежуточные результаты вычислений представлены в таблице 12.

ВЫВОД: на основании произведенных расчетов, можно сделать вывод, что на разработку продукта будет затрачено 815 528,05 рублей, а с учетом внедрения 883 105,05 руб. Цена готового продукта будет равна 1 324 657,57 рубля, что в свою очередь позволит получить чистую прибыль в размере от 176 621,01 рублей.

1. **Техника безопасности и охрана труда**
   1. **Анализ безопасности и охрана труда**

Рабочее место программиста — это его рабочий кабинет, либо часть помещения, в котором он проводит большую часть работы. Совершенно очевидно, что чем лучше приспособлено рабочее место для выполнения функций программиста, тем производительнее и эффективнее будет его трудовая деятельность.

Программист имеет собственное рабочее место, оборудованное ПЭВМ. Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также – расстоянию спинки до переднего края сиденья.

Рабочая поверхность стола сделана из дерева. На столе размещены ЭВМ, монитор и принтер. Площадь поверхности стола предполагает размещение монитора, устройств ввода/вывода и рабочей зоны с местом расположения оперативной документации.

Кабинет, в котором расположено рабочее место, представляет собой помещение площадью 15 м2, длиной 5 м, шириной 3 м и высотой 3 м.

Для создания благоприятных условий для зрительного восприятия кабинет оформлен в светло-бежевом цвете.

Для создания и поддержания независимо от наружных условий оптимальных значений температуры, влажности, чистоты воздуха, в холодное время используется водяное отопление, в теплое время года применяется кондиционирование воздуха.

Режим работы программиста - односменный с пятидневной рабочей неделей и нормированным рабочим днем. Продолжительность ежегодного отпуска - 28 календарных дней.

При идентификации вредных производственных факторов было установлено, что опасными могут являться:

* шум и вибрация, источником которого является оргтехника;
* электромагнитные поля и излучения от дисплея ПК;
* статическое электричество, накапливаемое на клавиатуре, дисплее, корпусе системного блока;
* электрический ток в электрических сетях;
* запыленность рабочей зоны, так влажная уборка рабочей поверхности проводиться не ежедневно, а только по мере загрязнения;
* статические перегрузки, так как работа программиста связана с неизменной статической позой, иногда неудобной позе (работа за ПК), что может привести к искривлению позвоночника, остеохондрозу, застою в органах малого таза;
* перенапряжение анализаторов, в большей степени зрительного, что может привести к различным нарушениям зрения (снижение остроты, спазм и уменьшение запаса аккомодации), помимо этого повышение зрительной нагрузки может приводить к общему утомлению, возникновению головной боли, ухудшению самочувствия;
* умственное перенапряжение, так как в процессе деятельности программист анализирует и перерабатывает огромное количество административных сетей, разрабатывает программные продукты, поддерживает технику в рабочем состоянии.

В целях выявления вредных и (или) опасных производственных факторов и осуществления мероприятий по приведению условий труда в соответствие государственными нормативными требованиями охраны трудаежегодно проводится аттестация рабочих мест по условиям труда - оценка условий труда на рабочих местах. Аттестации подлежат все имеющиеся в организации рабочие места.

При аттестации рабочих мест проводят оценку условий труда, гигиеническую оценку и оценку травмоопасности рабочих мест. При этом учитывают наличие средств коллективной защиты, обеспеченность работников средствами индивидуальной защиты и определяют эффективность этих средств.

Согласно санитарным правилам СП 2.2.3670-20 "Санитарно-эпидемиологические требования к условиям труда", дисплей должен располагаться на расстоянии 60-70 см, но не менее 50 см от глаз. Между боковыми поверхностями мониторов должно быть не менее 1,2 м. При использовании жидкокристаллических дисплеев на каждый компьютер должно приходиться не менее 4,5 м2 площади. На дисплей ПК не должен попадать прямой солнечный свет во избежание бликов и повышенной нагрузки на зрение.

Поэтому в кабинете, где работает программист, стоит только один компьютер, а комплектующая оргтехника расположена достаточно далеко непосредственно от самого специалиста. Это обусловлено тем, что все копировальные аппараты, принтеры и факс в периоды своей работы выделяют не только вредные токсичные газы, но и шумы и вибрацию. Данные факторы оказывают гораздо меньшее негативное влияние, нежели излучение от компьютера, но также являются вредными и могут нанести вред здоровью сотрудника.

Для снижения напряжения с глаз, а также для снятия мышечного напряжения во время рабочего дня проводится проветривание в течении 10 – 15 минут каждые 1,5 часа.

* 1. **Расчет искусственного освещения рабочего места в ООО «НПО «СтарЛайн»»**

Через зрительный анализатор человек получает около 80% из общего объема информации. Качество поступающей информации во многом зависит от освещения: неудовлетворительное количественно или качественно, оно не только утомляет зрение, но и вызывает утомление организма в целом. Кроме того, нерациональное освещение может явиться причиной травматизма: плохо освещенные опасные зоны, слепящие источники света и блики от них, резкие тени ухудшают видимость настолько, что вызывают полную потерю ориентировки работающих, снижают производительность труда и увеличивают брак продукции. Поэтому необходимо достаточное искусственное освещение для рабочего места.

Условия работы для расчета показателя освещенности рабочего места программиста являются:

* помещение размером 15 квадратных метров;
* высота помещения 3 метра, длина 5 метра, ширина 3 метров;
* лампы люминесцентные (дневного света) в количестве трех штук, новые;
* крупногабаритная мебель отсутствует.

Расчет общего равномерного искусственного освещения горизонтальной рабочей поверхности выполняется методом коэффициента использования светового потока. Световой поток (лм) одной лампы рассчитывается по следующей формуле:

 (25)

где Ен - нормируемая минимальная допустимая освещенность по СП 52.13330.2016, для помещения 2000 лк;

S - площадь освещаемого помещения, м2;

z - коэффициент неравномерности освещения, z = 1,1;

K - коэффициент запаса, учитывающий снижение освещенности в процессе эксплуатации вследствие загрязнения и старения ламп и светильников, а также снижения отражающих свойств поверхностей помещения для кабинета будет равен 1,3;

Nc - число светильников в помещении;

- коэффициент затенения, обычно 1;

=i - коэффициент использования светового потока.

Коэффициент использования светового потока, давший название методу расчета, определяется по индексу помещения i в зависимости от типа светильника и коэффициентов отражения света от потолка, стен и пола:

 (26)

где А, В - длина и ширина помещения в плане, м;

Нс - высота подвеса светильников над рабочей поверхностью, м.

Согласно СП 52.13330.2016 величина полученного светового потока достаточно для данного помещения.

* 1. **Электробезопасность в ООО «НПО «СтарЛайн»»**

При поступлении на работу сотрудник должен пройти инструктаж по технике безопасности и электробезопасности. Работника знакомят с основными правилами по технике безопасности, предлагают внимательно прочитать действующие на предприятии инструкции, поясняя при этом отдельные правила и требования.

Инструктаж по технике безопасности при выполнении конкретной работы проводит руководитель соответствующего производственного участка, показывая инструктируемому правильные безопасные приемы работы.

Повторный инструктаж проводится для рабочих независимо от их квалификации, стажа и опыта работы, не реже 1 раза в 6 месяцев по программе инструктажа на рабочем месте.

При нарушении работающими правил и инструкций по технике безопасности, технологической и производственной дисциплины, а также в случаях изменения технологического процесса или вида работы проводят дополнительные инструктажи.

Все инструктажи оформляются записями в специальном журнале с указанием номеров или шифров инструкций. Журнал о проведении инструктажа хранится у руководителя подразделения.

Работник на рабочем месте должен соблюдать общие меры электробезопасности. При этом запрещается:

* прикасаться к задней панели системного блока при включенном питании;
* допускать попадание влаги на поверхность системного блока, монитора, рабочую поверхность клавиатуры, дисководов, принтеров и др. устройств;
* производить самостоятельно вскрытие и ремонт оборудования;
* пользоваться неисправными электроприборами и электропроводкой;
* ремонтировать электроприборы самостоятельно;
* подвешивать электропровода на гвоздях, металлических и деревянных предметах, перекручивать провод, закладывать провод и шнуры на водопроводные трубы и батареи отопления, вешать что-либо на провода, вытягивать за шнур вилку из розетки;
* прикасаться одновременно к персональному компьютеру и к устройствам, имеющим соединение с землей (радиаторы отопления, водопроводные краны, трубы и т.п.);
* пользоваться самодельными электронагревательными приборами и электроприборами с открытой спиралью;
* наступать на переносимые электрические провода, лежащие на полу.

При работе с электроприборами и оргтехникой (персональные компьютеры, принтеры, сканеры, копировальные аппараты, факсы, бытовые электроприборы, приборы освещения):

* автоматические выключатели и электрические предохранители должны быть всегда исправны;
* Изоляция электропроводки, электроприборов, выключателей, штепсельных розеток, ламповых патронов и светильников, а также шнуров, с помощью которых включаются в электросеть электроприборы, должны быть в исправном состоянии;
* Электроприборы необходимо хранить в сухом месте, избегать резких колебаний температуры, вибрации, сотрясений.

Для подогрева воды пользоваться сертифицированными электроприборами с закрытой спиралью и устройством автоматического отключения, с применением несгораемых подставок.

* 1. **Пожарная безопасность в ООО «НПО «СтарЛайн»»**

Инструктаж по пожарной безопасности проводится по программе, разработанной инженером по охране труда ООО «НПО «СтарЛайн»», с учетом требований стандартов, правил, норм и инструкций о мерах пожарной безопасности. Продолжительность инструктажа устанавливается в соответствии с утвержденной программой. Инструктаж по пожарной безопасности, как правило, проводится совместно с инструктажем по технике безопасности и в те же сроки.

Инструктаж по пожарной безопасности проходят все работники организации, независимо от их образования, стажа работы по данной профессии или должности, временные работники, командированные, обучающиеся и студенты, прибывшие на практику.

О проведении инструктажа по пожарной безопасности работник, проводивший инструктаж, делает запись в журнале учета проведения инструктажей по пожарной безопасности, с обязательной подписью инструктируемого и инструктирующего. При регистрации внепланового инструктажа по пожарной безопасности указывают причину его проведения.

Все работники организаций должны допускаться к работе только после прохождения противопожарного инструктажа, а при изменении специфики работы проходить дополнительное обучение предупреждению и тушению возможных пожаров в порядке, установленном руководителем.

На проведение первичного противопожарного инструктажа необходимо отводить не менее 1 ч. Инструктируемые должны ознакомиться:

* с действующими на объекте правилами пожарной безопасности и инструкциями;
* с производственными участками, наиболее опасными в пожарном отношении, где запрещается курить, применять открытый огонь;
* с возможными причинами возникновения пожара и мерами его предупреждения;
* с практическими действиями в случае возникновения пожара - вызов пожарной помощи, использование первичных средств пожаротушения, место расположения ближайшего телефона и ознакомление с правилами поведения в случае возникновения пожара, эвакуации людей и материальных ценностей.

При первичном инструктаже инструктирующий обязан рассказать о производственных установках с повышенной пожарной опасностью, мерах предотвращения пожаров и загораний, указать место курения, ознакомить вновь поступившего с имеющимися на объекте средствами пожаротушения, показать ближайший телефон и объяснить правила поведения в случае возникновения пожара.

Проведение противопожарного инструктажа в обязательном порядке должно сопровождаться практическим показом способов использования имеющихся на объекте средств пожаротушения (противогазы, респираторы, огнетушители и т.д.).

Проведение противопожарного инструктажа в обязательном порядке должно сопровождаться практическим показом способов использования имеющихся на объекте средств пожаротушения (противогазы, респираторы, огнетушители и т.д.). Техника пожарной безопасности на предприятии ООО «НПО «СтарЛайн»» соблюдена согласно СП 4.13130.2013.

Соблюдение рассмотренных в данном разделе мероприятий по охране труда и технике безопасности в ООО «НПО «СтарЛайн»» позволяет снизить утомляемость и травматизм, повысить производительность труда, обеспечить комфортные условия трудовой деятельности специалиста, работающего в должности программиста.

# ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы был создан программный продукт, состоящий из клиентского десктопного приложения HDMapCreator, предназначенное для создания и редактирования высокоточных карт беспилотного транспорта.

В ходе выполнения работы был проведен анализ предметной области и анализ рынка существующих решений, рассматривающие тенденции развития и наиболее распространенные форматы HD-карт. Также для создания приложения были изучены аналогичные программные решения, на основании чего были дополнены функциональные требования, изначально определенные проектом OSCAR. Полученный продукт прошел тестирование, в ходе которого не было выявлено проблем в его функционировании. Таким образом, поставленные задачи были полностью выполнены.

Также в ходе выполнения работы была проведена оценка экономической эффективности программного продукта и разработаны мероприятия по охране труда.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. СтарЛайн. О компании. [Электронный ресурс]. – Режим доступа : http://ultrastar.ru/6354, свободный. – Загл. с экрана.
2. HD Maps: New age maps powering autonomous vehicles [Электронный ресурс]. – Режим доступа: https://www.geospatialworld.net/article/hd-maps-autonomous-vehicles/, свободный. – Загл. с экрана.
3. Бизли Д. Python. Подробный справочник. - Пер. с англ. - СПб.: Символ-Плюс, 2010. - 864 с, ил.
4. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. – 2-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2019. – 161 с. – (Бакалавр. Прикладной курс).
5. Трофимов, В. В. Основы алгоритмизации и программирования : учебник для СПО / В. В. Трофимов, Т. А. Павловская ; под ред. В. В. Трофимова. - М. : Издательство Юрайт, 2019. - 137 с
6. Кудрявцев, К. Я. Методы оптимизации : учеб. пособие для вузов / К. Я. Кудрявцев, А. М. Прудников. - 2-е изд. - М. : Издательство Юрайт, 2019. - 140 с.
7. Боггс У. UML и RationalRose / пер. с англ.; У. Боггс, М. Боггс. - М.: Издательство "ЛОРИ", 2000. - 582 с.
8. Rumbaugh J. The Unified Modeling Language. Reference Manual / J.rumbaugh, I. Jacobson, G. Booch. - MA: Addison-Wesley, 1999. - 550 p.
9. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
10. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

Приложение А

(справочное)

**Данные для экономической части**

Таблица A.1 – Исходные данные для расчета затрат на разработку

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Оклад разработчика | Ор | 110 000 руб. |
| Время разработки | Трп | 3 месяца |
| Машинное время разработки | Тмч | 2,5 месяца |
| Коэффициент дополнительной заработной платы | Кд | 0,17 |
| Коэффициент страховых взносов | Кст | 0,3 |
| Количество единиц техники | Q | 1 шт. |
| Себестоимость содержания техники | См/ч | 25 руб./час |
| Коэффициент готовности техники | Кгт | 0,95 |
| Число рабочих дней в месяце | ЧРД | 22 дня |
| Продолжительность смены | Тст | 8 часов |
| Коэффициент сменности | Ксм | 1 |
| Коэффициент транспортных расходов | Кт | 0 |
| Коэффициент накладных расходов | Кнр | 0,53 |

Таблица A.2 – Исходные данные для расчета затрат на внедрение

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Оклад инженера по внедрению | Ор | 30000 руб. |
| Время необходимое для внедрения | Трп | 1 месяца |
| Коэффициент дополнительной заработной платы | Кд | 0,17 |
| Коэффициент страховых взносов | Кст | 0,3 |
| Затраты на разработку | Зрп | 815 528,05 руб. |
| Себестоимость содержания техники | См/ч | 25 руб./час |
| Коэффициент готовности техники | Кгт | 0,95 |
| Число рабочих дней в месяце | ЧРД | 22 дня |
| Продолжительность смены | Тст | 8 часов |
| Коэффициент сменности | Ксм | 1 |
| Коэффициент накладных расходов | Кнр | 0,53 |

Приложение Б

(справочное)

**Исходной код**

**Пример дорожного элемента – StopSign**

from Objects.Shape import Shape

from PyQt5.QtGui import \*

from PyQt5.QtCore import \*

from map\_elements import StopLane

class StopSign(Shape):

    def \_\_init\_\_(self,id):

        super().\_\_init\_\_(id)

        self.id=id

        self.idStopLane=None

        self.direction=2

    def addStopLane(self, stoplane\_id):

        self.idStopLane = stoplane\_id

        if self.widget is not None:

            self.widget.setParentItem(stoplane\_id)

    def to\_dict(self):

        ans = {"id": self.id,

               "visible" : self.visible,

               "can\_export": self.can\_export,

               "show\_name": self.show\_name,

               "idStopLane": self.idStopLane,

               "direction": self.direction

        }

        return ans

    def paint(self, painter, point, image, scaleFactor = None):

        if not self.visible:

            return

        if self.highlighted:

            pen = QPen(self.highlighted\_text\_color) #lineColor

            penWidth = max(1, int(round(2.0 / self.scale)))

            pen.setWidth(penWidth)

            brush = QBrush(self.highlighted\_text\_color) #fillColor

            painter.setPen(pen)

            painter.setBrush(brush)

            polygon = QPolygonF()

            polygon.append(QPoint(point.x() - 3  / scaleFactor, point.y() + 12 / scaleFactor))

            polygon.append(QPoint(point.x() + 12 / scaleFactor, point.y() - 3 / scaleFactor))

            polygon.append(QPoint(point.x() + 28 / scaleFactor, point.y() - 3 / scaleFactor))

            polygon.append(QPoint(point.x() + 43 / scaleFactor, point.y() + 12 / scaleFactor))

            polygon.append(QPoint(point.x() + 43 / scaleFactor, point.y() + 28 / scaleFactor))

            polygon.append(QPoint(point.x() + 28 / scaleFactor, point.y() + 43 / scaleFactor))

            polygon.append(QPoint(point.x() + 12 / scaleFactor, point.y() + 43 / scaleFactor))

            polygon.append(QPoint(point.x() - 3 / scaleFactor, point.y() + 28 / scaleFactor))

            painter.drawPolygon(polygon)

        painter.drawImage(point.x(), point.y(), image)

    def addStopSignToMap(self, map, overlaps, selfMC, pointsLane):

        self.getArgsFromMapCreator(selfMC)

        rotateangle=float(self.rotation)

        x=float(self.offsetX)

        y=float(self.offsetY)#CHANGED

        st=StopLane(self.id,map)

        pointsStopLane=self.get\_decimal(pointsLane)

        points=self.get\_lanexy(pointsStopLane, x, y, rotateangle)

        st.add\_central\_curve(points)

        self.addOverlaps(overlaps,st)

**Пример объекта связи – Neighbor**

class Neighbor():

    LEFT\_FORWARD, LEFT\_REVERSED, RIGHT\_FORWARD, RIGHT\_REVERSED = range(4)

    def \_\_init\_\_(self, id, mainLane, neighbor\_lane, neighbor\_type):

        self.id=id

        self.mainLane=mainLane

        self.neighbor\_lane = neighbor\_lane

        self.neighbor\_type = neighbor\_type

    def get\_type\_str\_for\_main\_lane(self):

        if self.neighbor\_type == self.LEFT\_FORWARD:

            return "LEFT FORWARD"

        elif self.neighbor\_type == self.LEFT\_REVERSED:

            return "LEFT REVERSED"

        elif self.neighbor\_type == self.RIGHT\_FORWARD:

            return "RIGHT FORWARD"

        elif self.neighbor\_type == self.RIGHT\_REVERSED:

            return "RIGHT REVERSED"

    def get\_type\_str\_for\_neighbor\_lane(self):

        if self.neighbor\_type == self.LEFT\_FORWARD:

            return "RIGHT FORWARD"

        elif self.neighbor\_type == self.LEFT\_REVERSED:

            return "LEFT REVERSED"

        elif self.neighbor\_type == self.RIGHT\_FORWARD:

            return "LEFT FORWARD"

        elif self.neighbor\_type == self.RIGHT\_REVERSED:

            return "RIGHT REVERSED"

    def to\_dict(self):

        ans = {"id": self.id,

               "mainLane" : self.mainLane.id,

               "neighbor\_lane": self.neighbor\_lane.id,

               "neighbor\_type": self.neighbor\_type

        }

        return ans

    def from\_dict(self, d):

        self.\_\_dict\_\_.update(d)

**Пример protobuf файла - Map**

syntax = "proto2";

package apollo.hdmap;

import "proto/map/map\_clear\_area.proto";

import "proto/map/map\_crosswalk.proto";

import "proto/map/map\_junction.proto";

import "proto/map/map\_lane.proto";

import "proto/map/map\_overlap.proto";

import "proto/map/map\_signal.proto";

import "proto/map/map\_speed\_bump.proto";

import "proto/map/map\_stop\_sign.proto";

import "proto/map/map\_yield\_sign.proto";

import "proto/map/map\_road.proto";

import "proto/map/map\_parking\_space.proto";

import "proto/map/map\_pnc\_junction.proto";

// This message defines how we project the ellipsoidal Earth surface to a plane.

message Projection {

// PROJ.4 setting:

// "+proj=tmerc +lat\_0={origin.lat} +lon\_0={origin.lon} +k={scale\_factor}

// +ellps=WGS84 +no\_defs"

optional string proj = 1;

}

message Header {

optional bytes version = 1;

optional bytes date = 2;

optional Projection projection = 3;

optional bytes district = 4;

optional bytes generation = 5;

optional bytes rev\_major = 6;

optional bytes rev\_minor = 7;

optional double left = 8;

optional double top = 9;

optional double right = 10;

optional double bottom = 11;

optional bytes vendor = 12;

}

message Map {

optional Header header = 1;

repeated Crosswalk crosswalk = 2;

repeated Junction junction = 3;

repeated Lane lane = 4;

repeated StopSign stop\_sign = 5;

repeated Signal signal = 6;

repeated YieldSign yield = 7;

repeated Overlap overlap = 8;

repeated ClearArea clear\_area = 9;

repeated SpeedBump speed\_bump = 10;

repeated Road road = 11;

repeated ParkingSpace parking\_space = 12;

repeated PNCJunction pnc\_junction = 13;

}

**Функции испорта данных из TXT и BIN**

    def openBinFunc(self):

        self.clearObj()

        if self.imageLabel is None:

            self.open()

            if self.imageLabel is None:

                return

        if self.folders['output']!="":

            fileName, \_ = QFileDialog.getOpenFileName(self, "Open bin map file",

                     self.folders['output'],"Bin files(\*.bin\*)")

        else:

            fileName, \_ = QFileDialog.getOpenFileName(self, "Open bin map file",

                     PATH,"Bin files(\*.bin\*)")

        if fileName is not None and fileName!="" and self.imageLabel is not None:

            getObjectsFromBin(self, fileName)

    def openTxtFunc(self):

        self.clearObj()

        if self.imageLabel is None:

            self.open()

            if self.imageLabel is None:

                return

        if self.folders['output']!="":

            fileName, \_ = QFileDialog.getOpenFileName(self, "Open txt map file",

                     self.folders['output'],"Text files (\*.txt\*);;TEXT (\*.TXT\*)")

        else:

            fileName, \_ = QFileDialog.getOpenFileName(self, "Open txt map file",

                     PATH,"Text files (\*.txt\*);;TEXT (\*.TXT\*)")

        if fileName is not None and fileName!="" and self.imageLabel is not None:

            getObjectsFromTxt(self, fileName)

**Функция удаления дорожного элемента**

def DeleteObjFunc(self):

    if not self.canDelete:

        self.MapCreator.CreateErrorWindow("Please, end editing or creating object, then you'll be able to delete objects", "")

        return

    for shape in self.MapCreator.Lanes:

        if not shape.visible or (shape.is\_border is not None and not shape.parent.visible):

            continue

        index = shape.nearestVertex(self.rightClickPos, self.epsilon/self.scaleFactor)

        index = index or shape.leftBorder.nearestVertex(self.rightClickPos, self.epsilon/self.scaleFactor)

        index = index or shape.rightBorder.nearestVertex(self.rightClickPos, self.epsilon/self.scaleFactor)

        if index is not None:

            self.MapCreator.stack.append(shape, Action.DELETION\_STARTED)

            del\_index=[]

            i=0

            for rel in self.MapCreator.Relations:

                if not (rel.successor is None or rel.predecessor is None) and \

                 (rel.successor.id == shape.id or rel.predecessor.id == shape.id):

                    del\_index.append(i)

                i+=1

            del\_index.sort(reverse=True)

            for i in del\_index:

                self.MapCreator.stack.append(self.MapCreator.Relations[i], Action.DELETED)

                del self.MapCreator.Relations[i]

            for road in self.MapCreator.Roads:

                if shape.id in road.idLanes:

                    road.idLanes.remove(shape.id)

                    if len(road.idLanes)==0:

                        self.MapCreator.stack.append(road, Action.DELETED)

                        del self.MapCreator.Roads[self.MapCreator.Roads.index(road)]

            self.MapCreator.stack.append(shape, Action.DELETED)

            del self.MapCreator.Lanes[self.MapCreator.Lanes.index(shape)]

            self.draggableIndex=None

            self.draggablePoint=None

            self.update()

            return

    for shape in self.MapCreator.Signals:

        if not shape.visible:

            continue

        index = shape.nearestVertex(self.rightClickPos, self.epsilon/self.scaleFactor)

        if index is not None:

            if shape == self.current:

                return

            self.MapCreator.stack.append(shape, Action.DELETED)

            del self.MapCreator.Signals[self.MapCreator.Signals.index(shape)]

            self.draggableIndex=None

            self.draggablePoint=None

            self.update()

            return

    for shape in self.MapCreator.Landmarks:

        if not shape.visible:

            continue

        index = shape.nearestVertex(self.rightClickPos, self.epsilon/self.scaleFactor)

        if index is not None:

            self.MapCreator.stack.append(shape, Action.DELETED)

            del self.MapCreator.Landmarks[self.MapCreator.Landmarks.index(shape)]

            self.draggableIndex=None

            self.draggablePoint=None

            self.update()

            return

    for shape in self.MapCreator.StopLanes:

        if not shape.visible:

            continue

        index = shape.nearestVertex(self.rightClickPos, self.epsilon/self.scaleFactor)

        if index is not None:

            self.MapCreator.stack.append(shape, Action.DELETION\_STARTED)

            for signal in self.MapCreator.Signals:

                if signal.idStopLane == shape.id:

                    self.MapCreator.stack.append(signal, Action.DELETED)

                    del self.MapCreator.Signals[self.MapCreator.Signals.index(signal)]

            for stopsign in self.MapCreator.StopSigns:

                if stopsign.idStopLane == shape.id:

                    self.MapCreator.stack.append(stopsign, Action.DELETED)

                    del self.MapCreator.StopSigns[self.MapCreator.StopSigns.index(stopsign)]

            self.MapCreator.stack.append(shape, Action.DELETED)

            del self.MapCreator.StopLanes[self.MapCreator.StopLanes.index(shape)]

            self.draggableIndex=None

            self.draggablePoint=None

            self.update()

            return

    for shape in self.MapCreator.Crosswalks:

        if not shape.visible:

            continue

        index = shape.nearestVertex(self.rightClickPos, self.epsilon/self.scaleFactor)

        if index is not None:

            self.MapCreator.stack.append(shape, Action.DELETED)

            del self.MapCreator.Crosswalks[self.MapCreator.Crosswalks.index(shape)]

            self.draggableIndex=None

            self.draggablePoint=None

            self.update()

            return

    for shape in self.MapCreator.Junctions:

        if not shape.visible:

            continue

        index = shape.nearestVertex(self.rightClickPos, self.epsilon/self.scaleFactor)

        if index is not None:

            self.MapCreator.stack.append(shape, Action.DELETED)

            del self.MapCreator.Junctions[self.MapCreator.Junctions.index(shape)]

            self.draggableIndex=None

            self.draggablePoint=None

            self.update()

            return

    for shape in self.MapCreator.ParkingSpaces:

        if not shape.visible:

            continue

        index = shape.nearestVertex(self.rightClickPos, self.epsilon/self.scaleFactor)

        if index is not None:

            self.MapCreator.stack.append(shape, Action.DELETION\_STARTED)

            for landmark in self.MapCreator.Landmarks:

                if landmark.parking\_space\_id == shape.id:

                    self.MapCreator.stack.append(landmark, Action.PROPERTY\_CHANGED,

                                                 property\_dict = {'parking\_space\_id': shape.id})

                    landmark.parking\_space\_id = None

            self.MapCreator.stack.append(shape, Action.DELETED)

            del self.MapCreator.ParkingSpaces[self.MapCreator.ParkingSpaces.index(shape)]

            self.draggableIndex=None

            self.draggablePoint=None

            self.update()

            return

**Функция отрисовки дорожных элементов в окне редактора**

def paintEvent(self, event):

    if not self.PixMap:

        return super(Canvas, self).paintEvent(event)

    p = self.\_painter

    p.begin(self)

    p.setFont(QFont('Times', self.font\_size, QFont.Bold))

    p.setRenderHint(QPainter.Antialiasing)

    p.setRenderHint(QPainter.HighQualityAntialiasing)

    p.setRenderHint(QPainter.SmoothPixmapTransform)

    p.scale(self.scaleFactor, self.scaleFactor)

    p.drawPixmap(0, 0, self.PixMap)

    if len(self.previews) > 0:

        for preview in self.previews:

            preview.paint(p)

    if self.Trajectory is not None:

        self.Trajectory.paint(p)

    if self.current is not None:

        self.current.scale = self.scaleFactor

        if type(self.current) is Lane:

            self.current.leftBorder.scale = self.scaleFactor

            self.current.rightBorder.scale = self.scaleFactor

    highlighted\_objects = []

    for junction in self.MapCreator.Junctions:

        if junction.highlighted:

            highlighted\_objects.append(junction)

        else:

            junction.paint(p)

    for parking\_space in self.MapCreator.ParkingSpaces:

        if parking\_space.highlighted:

            highlighted\_objects.append(parking\_space)

        else:

            parking\_space.paint(p)

    for road in self.MapCreator.Roads:

        road.paint(p, self.MapCreator)

    for crosswalk in self.MapCreator.Crosswalks:

        if len(crosswalk.points)!=0:

            if crosswalk.highlighted:

                highlighted\_objects.append(crosswalk)

            else:

                if crosswalk.accepted:

                    if crosswalk.direction==1:

                        image = self.stripesImageHor.scaled(20/self.scaleFactor, 20/self.scaleFactor)

                    elif crosswalk.direction==2:

                        image = self.stripesImageVer.scaled(20/self.scaleFactor, 20/self.scaleFactor)

                    crosswalk.paint(p, image)

                else:

                    crosswalk.paint(p)

    for stoplane in self.MapCreator.StopLanes:

        if stoplane.highlighted:

            highlighted\_objects.append(stoplane)

        else:

            stoplane.paint(p)

    for landmark in self.MapCreator.Landmarks:

        landmark.paint(p)

    for stopsign in self.MapCreator.StopSigns:

        if stopsign.idStopLane is None:

            continue

        stoplane = self.MapCreator.StopLanes.find(stopsign.idStopLane)

        if not stoplane:

            continue

        if stopsign.highlighted:

            highlighted\_objects.append(stopsign)

        else:

            point = stoplane.points[0]

            stopsign.paint(p,point,image=self.imageStopSign.scaled(40/self.scaleFactor, 40/self.scaleFactor))

    for signal in self.MapCreator.Signals:

        if signal.highlighted:

            highlighted\_objects.append(signal)

        else:

            signal.paint(p,image=self.imageSignal.scaled(15/self.scaleFactor, 40/self.scaleFactor, Qt.KeepAspectRatio))

    for lane in self.MapCreator.Lanes:

        if lane.highlighted:

            highlighted\_objects.append(lane)

        else:

            lane.paint(p)

    for obj in highlighted\_objects:

        if type(obj) in [Lane, StopLane, Junction, ParkingSpace]:

            obj.paint(p)

        elif type(obj) == Crosswalk:

            if obj.accepted:

                if obj.direction==1:

                    image = self.stripesImageHor.scaled(20/self.scaleFactor, 20/self.scaleFactor)

                elif obj.direction==2:

                    image = self.stripesImageVer.scaled(20/self.scaleFactor, 20/self.scaleFactor)

                obj.paint(p, image)

            else:

                obj.paint(p)

        elif type(obj) == Signal:

            obj.paint(p,image=self.imageSignal.scaled(15/self.scaleFactor, 40/self.scaleFactor, \

            Qt.KeepAspectRatio), scaleFactor=self.scaleFactor)

        elif type(obj) == StopSign:

            stoplane = self.MapCreator.StopLanes.find(obj.idStopLane)

            point = stoplane.points[0]

            obj.paint(p,point,image=self.imageStopSign.scaled(40/self.scaleFactor, \

            40/self.scaleFactor), scaleFactor = self.scaleFactor)

    if self.Ruler is not None and len(self.Ruler.points)>0:

        self.Ruler.scale=self.scaleFactor

        if len(self.Ruler.points)==2:

            x1=float(Decimal(self.Ruler.points[0].x())\*Decimal(self.geotiffScale)\*self.resizeFactorWidth)

            y1=float(Decimal(self.Ruler.points[0].y())\*Decimal(self.geotiffScale)\*self.resizeFactorHeight)

            x2=float(Decimal(self.Ruler.points[1].x())\*Decimal(self.geotiffScale)\*self.resizeFactorWidth)

            y2=float(Decimal(self.Ruler.points[1].y())\*Decimal(self.geotiffScale)\*self.resizeFactorHeight)

            distance=str(round(float(self.distance\_between(QPointF(x1,y1),QPointF(x2,y2))),2))

        else:

            distance="0"

        self.Ruler.paint(p,distance=distance)

    p.end()

**Функция сохранения проекта**

def SaveTxt(self,dir,export=False):

    filename = dir.split('/')[-1]

    if filename == '':

        return

    version = ask\_for\_apollo\_version(self)

    map = map\_pb2.Map()

    poi = poi\_pb2.POI()

    werent\_exported = []

    map\_dict = {}

    map\_dict['widgets'] = self.TreeWidget.get\_dict()

    map\_dict['lanes'] = []

    for obj in self.Lanes:

        map\_dict['lanes'].append(obj.to\_dict(self))

        if obj.can\_export:

            obj.addLaneToMap(map, self)

        else:

            werent\_exported.append(obj.id)

    map\_dict['stoplane'] = []

    for obj in self.StopLanes:

        map\_dict['stoplane'].append(obj.to\_dict(self))

        if not obj.can\_export:

            werent\_exported.append(obj.id)

    map\_dict['signals'] = []

    for obj in self.Signals:

        map\_dict['signals'].append(obj.to\_dict(self))

        if obj.can\_export:

            stoplane = self.StopLanes.find(obj.idStopLane)

            obj.addSignalToMap(map, self.Overlaps, self, stoplane.points)

        else:

            werent\_exported.append(obj.id)

    map\_dict['stopsigns'] = []

    for obj in self.StopSigns:

        map\_dict['stopsigns'].append(obj.to\_dict())

        if obj.can\_export:

            stoplane = self.StopLanes.find(obj.idStopLane)

            if stoplane:

                points = stoplane.points

                obj.addStopSignToMap(map, self.Overlaps, self, points)

        else:

            werent\_exported.append(obj.id)

    map\_dict['crosswalks'] = []

    for obj in self.Crosswalks:

        map\_dict['crosswalks'].append(obj.to\_dict(self))

        if obj.can\_export:

            obj.addCrosswalkToMap(map, self.Overlaps, self)

        else:

            werent\_exported.append(obj.id)

    map\_dict['junctions'] = []

    for obj in self.Junctions:

        map\_dict['junctions'].append(obj.to\_dict(self))

        if obj.can\_export:

            obj.addJunctionToMap(self, map, self.Overlaps)

        else:

            werent\_exported.append(obj.id)

    map\_dict['parking\_spaces'] = []

    for obj in self.ParkingSpaces:

        map\_dict['parking\_spaces'].append(obj.to\_dict(self))

        if obj.can\_export:

            obj.addParkingSpaceToMap(self, map, self.Overlaps)

        else:

            werent\_exported.append(obj.id)

    map\_dict['landmarks'] = []

    has\_landmark\_to\_save = False

    for obj in self.Landmarks:

        map\_dict['landmarks'].append(obj.to\_dict(self))

        if obj.can\_export:

            ps\_id = obj.parking\_space\_id

            if not self.ParkingSpaces.find(ps\_id).can\_export:

                obj.parking\_space\_id = None

                self.createMessage("Exporting parking space (" + ps\_id +

                 ") was not allowed. Parking space property for landmark (" + obj.id + ") set to None.")

            obj.save(self, poi, version)

            obj.parking\_space\_id = ps\_id

            has\_landmark\_to\_save = True

        else:

            werent\_exported.append(obj.id)

    if has\_landmark\_to\_save:

        file = open(dir+"/default\_end\_way\_point.txt", 'w')

        file.write(str(poi))

        file.close()

    map\_dict['roads'] = []

    for obj in self.Roads:

        map\_dict['roads'].append(obj.to\_dict())

        if obj.can\_export:

            obj.addRoadToMap(map, self)

        else:

            werent\_exported.append(obj.id)

    map\_dict['relations'] = []

    for rel in self.Relations:

        if rel.successor is None or rel.predecessor is None:

            continue

        map\_dict['relations'].append(rel.to\_dict())

    map\_dict['overlaps'] = []

    for overlap in self.Overlaps:

        map\_dict['overlaps'].append(overlap.to\_dict())

        if overlap.can\_export():

            overlap.addOverlapToMap(map, self)

    map\_dict['neighbors'] = []

    for neighbor in self.Neighbors:

        map\_dict['neighbors'].append(neighbor.to\_dict())

    if not export:

        map\_json = json.dumps(map\_dict)

        with open(dir+"/"+filename+'.json', 'w') as file:

            file.write(map\_json)

        with open(self.fileSettings, 'wb') as f:

            pickle.dump((self.offsetX,self.offsetY,self.offsetZ,self.rotation),f)

            pickle.dump(self.folders,f)

            pickle.dump(self.max\_stack\_length,f)

            pickle.dump(Shape.text\_color\_str,f)

    map\_file = open(dir+"/base\_map.txt", 'w')

    map\_file.write(str(map))

    map\_file.close()

    if len(werent\_exported) > 0:

        werent\_exported = "These objects weren't exported:\n" \

        + ", ".join(werent\_exported)

        self.createMessage(werent\_exported)

**Класс для сохранения действий пользователя**

class Action(object):

    CREATION\_STARTED, CREATED, ADDING\_POINT\_STARTED, POINT\_ADDED, POINT\_MOVED,\

    POINT\_DELETED, DELETION\_STARTED, DELETED, RENAMED, PROPERTY\_CHANGED = range(10)

    def \_\_init\_\_(self, obj, action, is\_folder=False, property\_dict=None):

        super().\_\_init\_\_()

        self.obj = obj

        self.action = action

        self.is\_folder = is\_folder

        self.property\_dict = property\_dict

        self.update\_widget\_pos()

        if is\_folder:

            self.id = obj.id

            self.points = []

            self.widget\_pos = self.obj.getPos()

            self.widget\_dict = self.obj.get\_dict()

            self.folder\_children = []

            for child in self.obj.children:

                if child.is\_folder:

                    self.folder\_children.append([child, "folder"])

                else:

                    self.folder\_children.append([child.obj.id, type(child.obj)])

            if self.obj.folder is not None:

                self.parent\_folder = self.obj.folder.id

            else:

                self.parent\_folder = None

        elif type(obj) == Relation:

            self.id = "RELATION\_HAS\_NO\_ID"

            self.points = []

            self.parent\_folder = None

            self.widget\_pos = None

            self.widget\_dict =  None

        else:

            self.id = self.obj.id

            if type(self.obj.points) == WayPointsList:

                self.points = self.obj.points.getInnerListCopy()

            else:

                self.points = []

                for point in self.obj.points:

                    self.points.append(QPointF(point.x(), point.y()))

            if self.obj.widget is not None:

                self.widget\_pos = self.obj.widget.getPos()

                self.widget\_dict = self.obj.widget.get\_dict()

                if self.obj.widget.folder is not None:

                    self.parent\_folder = self.obj.widget.folder.id

                else:

                    self.parent\_folder = None

            else:

                self.widget\_pos = None

                self.widget\_dict = None

                self.parent\_folder = None

    def update\_widget\_pos(self):

        if hasattr(self.obj, 'widget') and self.obj.widget is not None:

            self.widget\_pos = self.obj.widget.getPos()

        else:

            self.widget\_pos = None

    def get\_action\_type\_text(self):

        if self.action == self.CREATED:

            return "CREATED"

        elif self.action == self.ADDING\_POINT\_STARTED:

            return "ADDING\_POINT\_STARTED"

        elif self.action == self.POINT\_ADDED:

            return "POINT\_ADDED"

        elif self.action == self.POINT\_MOVED:

            return "POINT\_MOVED"

        elif self.action == self.POINT\_DELETED:

            return "POINT\_DELETED"

        elif self.action == self.DELETION\_STARTED:

            return "DELETION\_STARTED"

        elif self.action == self.DELETED:

            return "DELETED"

        elif self.action == self.RENAMED:

            return "RENAMED"

**Функция отмены пользовательского действия**

def undo(self):

    cur\_action = self.\_inner\_list[-1]

    obj = cur\_action.obj

    new\_action = Action(obj, cur\_action.action, is\_folder=cur\_action.is\_folder, property\_dict=cur\_action.property\_dict)

    if cur\_action.action in \

    [Action.POINT\_MOVED, Action.POINT\_ADDED, Action.POINT\_DELETED]:

        if type(obj.points) == WayPointsList:

            obj.points.setInnerList(cur\_action.points)

        else:

            obj.points = MyPointsList()

            for point in cur\_action.points:

                obj.points.append(point)

    elif cur\_action.action == Action.ADDING\_POINT\_STARTED:

        return

    elif cur\_action.action == Action.RENAMED:

        if type(obj) == ParkingSpace:

            for landmark in self.MC.Landmarks:

                if landmark.parking\_space\_id == obj.id:

                    landmark.parking\_space\_id = cur\_action.id

        obj.id = cur\_action.id

        if cur\_action.is\_folder:

            obj.setId(cur\_action.id)

        else:

            obj.widget.setId(obj.id)

    elif cur\_action.action == Action.PROPERTY\_CHANGED:

        self.\_inner\_list\_canceled.append(new\_action)

        obj.\_\_dict\_\_.update(cur\_action.property\_dict)

        del self.\_inner\_list[-1]

        return

       elif cur\_action.action == Action.DELETION\_STARTED:

        self.\_inner\_list\_canceled.append(new\_action)

        del self.\_inner\_list[-1]

        return

    elif cur\_action.action == Action.CREATION\_STARTED:

        self.\_inner\_list\_canceled.append(new\_action)

        del self.\_inner\_list[-1]

        return

    elif cur\_action.action == Action.DELETED:

        if cur\_action.is\_folder:

            self.MC.TreeWidget.container.add\_folder\_on\_pos(obj, 0)

        elif type(obj) == Lane:

            self.MC.Lanes.append(obj)

        elif type(obj) == Road:

            self.MC.Roads.append(obj)

        elif type(obj) == Relation:

            self.MC.Relations.append(obj)

        elif type(obj) == Signal:

            self.MC.Signals.append(obj)

        elif type(obj) == StopSign:

            self.MC.StopSigns.append(obj)

        elif type(obj) == StopLane:

            self.MC.StopLanes.append(obj)

        elif type(obj) == Crosswalk:

            self.MC.Crosswalks.append(obj)

        elif type(obj) == Junction:

            self.MC.Junctions.append(obj)

        elif type(obj) == ParkingSpace:

            self.MC.ParkingSpaces.append(obj)

        elif type(obj) == Landmark:

            self.MC.Landmarks.append(obj)

            obj.points.setInnerList(cur\_action.points)

        if cur\_action.widget\_pos is not None:

            if cur\_action.is\_folder:

                obj.container.moveWidget(obj, cur\_action.widget\_pos + 1)

            else:

                obj.widget.container.moveWidgetWithChildren(obj.widget,\

                    cur\_action.widget\_pos)

            cur\_action.update\_widget\_pos()

        if cur\_action.is\_folder:

            obj.setVisible(True)

        elif hasattr(obj, "widget"):

            obj.widget.setVisible(True)

        self.MC.TreeWidget.container.update\_size()

    elif cur\_action.action == Action.CREATED:

        if type(obj) == Lane:

            del self.MC.Lanes[self.MC.Lanes.index(obj)]

        elif type(obj) == Road:

            del self.MC.Roads[self.MC.Roads.index(obj)]

        elif type(obj) == Relation:

            del self.MC.Relations[self.MC.Relations.index(obj)]

        elif type(obj) == Signal:

            del self.MC.Signals[self.MC.Signals.index(obj)]

        elif type(obj) == StopSign:

            del self.MC.StopSigns[self.MC.StopSigns.index(obj)]

        elif type(obj) == StopLane:

            del self.MC.StopLanes[self.MC.StopLanes.index(obj)]

        elif type(obj) == Crosswalk:

            del self.MC.Crosswalks[self.MC.Crosswalks.index(obj)]

        elif type(obj) == Junction:

            del self.MC.Junctions[self.MC.Junctions.index(obj)]

        elif type(obj) == ParkingSpace:

            del self.MC.ParkingSpaces[self.MC.ParkingSpaces.index(obj)]

        elif type(obj) == Landmark:

            obj.points.clear()

            del self.MC.Landmarks[self.MC.Landmarks.index(obj)]

    self.\_inner\_list\_canceled.append(new\_action)

    del self.\_inner\_list[-1]

    if self.\_inner\_list\_canceled[-1].action == Action.DELETED:

        index = self.getDeletionStartIndex(obj)

        if index != len(self.\_inner\_list):

            widget\_dict = self.\_inner\_list[index].widget\_dict

        else:

            widget\_dict = None

        self.undoDeletion(obj)

        if type(obj) == Lane:

            if obj.parentRoad is not None:

                road = self.MC.Roads.find(obj.parentRoad)

                obj.setParentRoad(road.id)

        elif type(obj) == Signal or type(obj) == StopSign:

            if obj.idStopLane is not None:

                obj.addStopLane(obj.idStopLane)

        if self.\_inner\_list\_canceled[-1].is\_folder:

            children = []

            for child in self.\_inner\_list\_canceled[-1].folder\_children:

                if child[1] == "folder":

                    children.append(child[0])

                elif child[1] == Lane:

                    children.append(self.MC.Lanes.find(child[0]).widget)

                elif child[1] == Road:

                    children.append(self.MC.Roads.find(child[0]).widget)

                elif child[1] == Signal:

                    children.append(self.MC.Signals.find(child[0]).widget)

                elif child[1] == StopSign:

                    children.append(self.MC.StopSigns.find(child[0]).widget)

                elif child[1] == StopLane:

                    children.append(self.MC.StopLanes.find(child[0]).widget)

                elif child[1] == Crosswalk:

                    children.append(self.MC.Crosswalks.find(child[0]).widget)

                elif child[1] == Junction:

                    children.append(self.MC.Junctions.find(child[0]).widget)

                elif child[1] == ParkingSpace:

                    children.append(self.MC.ParkingSpaces.find(child[0]).widget)

                elif child[1] == Landmark:

                    children.append(self.MC.Landmarks.find(child[0]).widget)

            obj.children = []

            for child in children:

                obj.addChild(child)

                child.folder = obj

                child.setLevel(obj.level + 1)

        elif cur\_action.parent\_folder is not None:

            folder = self.MC.TreeWidget.container.folders.find(cur\_action.parent\_folder)

            if folder is not None:

                folder.addChild(obj.widget)

                obj.widget.folder = folder

                obj.widget.setLevel(folder.level + 1)

        if type(obj) == Relation:

            pass

        elif self.\_inner\_list\_canceled[-1].is\_folder:

            if widget\_dict is not None:

                obj.update\_icons(widget\_dict)

            else:

                obj.update\_icons(cur\_action.widget\_dict)

        else:

            if widget\_dict is not None:

                obj.widget.update\_icons(widget\_dict)

            else:

                obj.widget.update\_icons(cur\_action.widget\_dict)

    elif self.\_inner\_list\_canceled[-1].action == Action.CREATED:

        self.undoCreation(obj)

    self.MC.imageLabel.update()

    self.MC.TreeWidget.container.update\_size()

    self.enableActions()