Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

09.03.01 Информатика и вычислительная техника
код и наименование направления подготовки

ОТЧЕТ
по преддипломной практике

по направлению 09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) – «Электронно-вычислительные машины, комплексы, системы и сети», квалификация – бакалавр,
программа академического бакалавриата,
форма обучения – очная, год начала подготовки (по учебному плану) – 2016

Выполнил:
студент гр. ИВ-622
«23» мая 2020 г. _____ /Тимофеев Д.А./

Оценка «_____»

Руководитель практики
от университета
с.п., Кафедры ВС
«23» мая 2020 г. _____ /Гонцова А.В./

Новосибирск 2018

ПЛАН-ГРАФИК ПРОВЕДЕНИЯ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ

Тип практики: преддипломная практика
Способ проведения практики: стационарная
Форма проведения практики: дискретно по периодам проведения практики

Тема ВКР: Разработка алгоритма централизованного управления автомобилями для систем автоведения.

Содержание практики

Наименование видов деятельности	Дата (начало – окончание)
Постановка задачи на практику, определение конкретной индивидуальной темы, формирование плана работ. Водный инструктаж по технике безопасности (охране труда, пожарной безопасности).	03.02.20-20.02.20
Работа с библиотечными фондами, сбор и анализ материалов по теме практики. Поиск имеющегося опыта по планированию движения.	20.03.20-29.02.20
Придумывание различных ситуаций и задач, при движении машин (препятствия, ландшафт и т.п.). Разработка алгоритма. Разработка архитектуры проекта в виде UML диаграмм.	20.02.20-20.03.20
Выполнение работ в соответствии с составленным планом	20.03.20-06.04.20
Сдача лабораторных работ и экзаменов за семестр	06.04.20-06.05.20

Согласовано:

Руководитель практики
от университета
с.п., Кафедры ВС _____ /Гонцова А.В./

ЗАДАНИЕ НА ПРЕДДИПЛОМНУЮ ПРАКТИКУ

Кратко:

1. Разработать централизованный логистический алгоритм, который:
   1.1. раздает маршруты движения для каждой подключенной машины
   1.2. организует движение без пробок, заторов и т.п. (это не обязано быть похожим на правила дорожного движения)
2. Симуляция движения машин, для отладки и демонстрации работы этого алгоритма

Понятия

1. Автопилот – автопилотируемая машина, следующая заданному маршруту (машина под управлением какой-либо системы автоведения).
2. ЦКА – Центральный Контроль Автопилотов (система, раздающая маршруты для машин, под управлением автопилотов). Система организующая эффективный дорожный трафик.
3. ТПД – Точка Прямого Доступа – это точка, в которую можно попасть по прямой линии, не встретив препятствий, без необходимости поворачивать.
4. Поток – место, где генерируются или исчезают машины. На карте будет обозначаться схематично. Является симуляцией въезда/выезда из города, входа/выхода из подземной парковки и т.п.

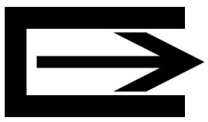


Рисунок 1 Поток создания машин, обозначение на карте

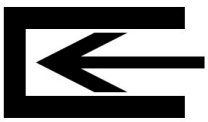


Рисунок 2 Поток поглощения машин, обозначение на карте

Техническое задание в целом

1. Допущения и условности
   1.1. Машина может поворачивать «уголком» (ехала прямо, и вдруг повернула в сторону на 90).

   Я это оправдываю тем, что около машины есть некоторое свободное расстояние всегда и поворот на маленьких скоростях занимает не так уж и много места. Да и большинство машин ездит на скорости до 40 км/ч.
   1.2. Не обязано быть похожим на Правила Дорожного Движения.

Техническое задание для алгоритма

1. Пока есть возможность доехать до точки прибытия, ЦКА должен вести туда машины.
2. ЦКА должен успешно преодолевать
   2.1. Статичные препятствия (здания т.п.)
   2.2. Динамические препятствия (неподконтрольные машины, разрушающиеся/появляющиеся внезапно здания, светофоры с кнопкой и т.п.)
3. Время отклика системы: 1 шаг (инструкции под новые условия должны появиться уже на следующем шаге).

   Машины перемещаются по квадратикам, около каждой машины спереди всегда есть свободный квадратик. На тот момент, когда машина переместиться на следующий квадратик, уже должны быть новые инструкции под новые условия.
4. Машины к системе ЦКА могут подключаться/отключаться внезапно.

Техническое задание для симуляции

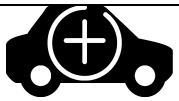
1. Создание и редактирование карты на ходу
   1.1. Добавление/удаление зданий
   1.2. Изменение ландшафта
2. Управление появляющимся потоком машин на ходу
   2.1. Добавление/удаление потоков
   2.2. Регулирование скорости порождения машин у потока
   2.3. Поломка машин
   2.4. Назначение точки прибытия машинам в система ЦКА, создающихся потоком.

   Поток будет создавать машины, которые будут передаваться в ЦКА с заранее установленной точкой прибытия.
3. Ввод и вывод машин под контроль ЦКА
4. Управление маршрутами отдельных машин
5. Возможность конкретной машине задать свой маршрут.
6. Пешеходные переходы с кнопкой требования прохода.

Техническое задание для графического интерфейса

Графический интерфейс будет очень схематичным.

1. Панель управления (слева)

Название:	**Внешний вид:**
Изменить ландшафт	
Добавить поток порождающий	
Добавить поток поглощающий	
Добавить машину	
Удалить (ластик, очистить)	
Перевести в состояние «разрушен»	

2. Свойства объектов редактировать
    2.1. Свойства машин
        2.1.1. Подключить к ЦКА, назначив точку прибытия
        2.1.2. Ввести собственный маршрут.
    2.2. Свойства порождающего потока
        2.2.1. Скорость генерации автомобилей
        2.2.2. Точка прибытия в ЦКА
3. Панель список системных слоев (внутренних объектов), при помощи которых идет построение маршрутов.

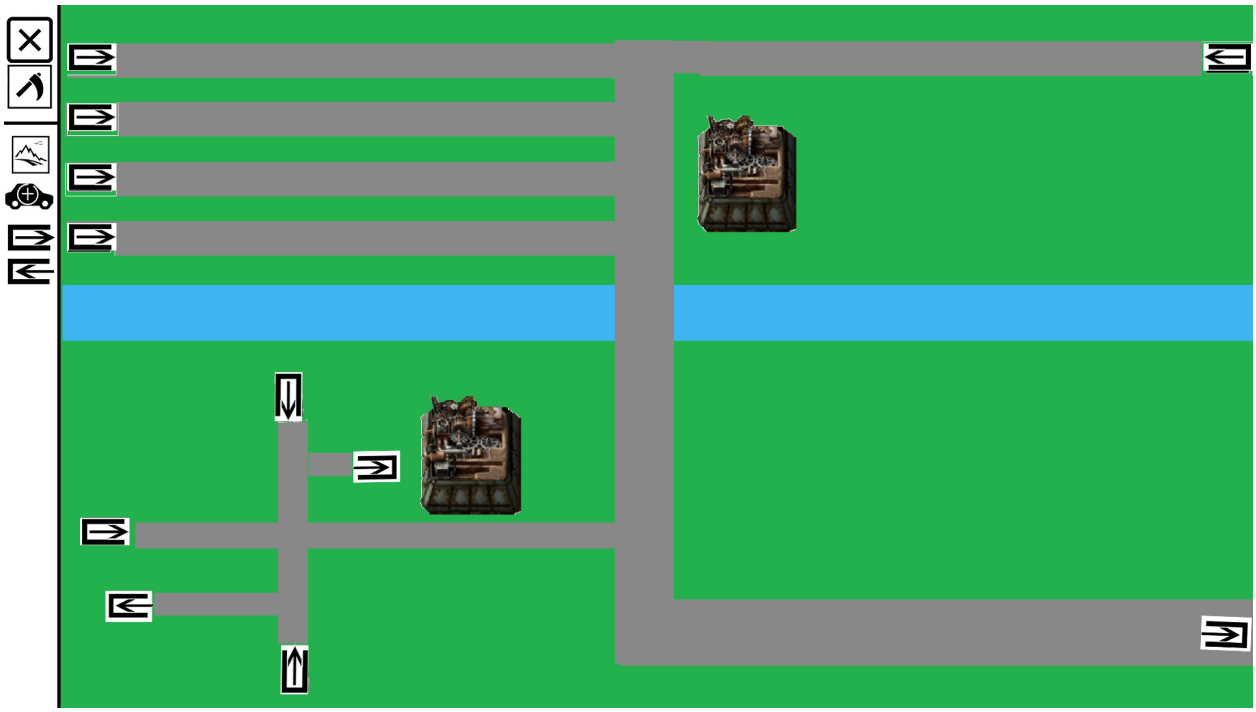


Рисунок 3 Примерный вид графического интерфейса

Значение цветов на карте

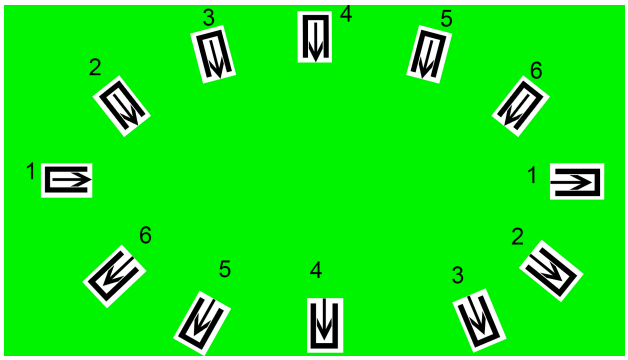
1. Зеленый – трава, зона обычной проходимости.
2. Синий – река (непроходимо для обычных машин)
3. Серый – асфальт, зона самой высокой проходимости
4. Картинка здания – здание, непроходимо
5. Черный – непроходимое препятствие

Техническое задание для алгоритма (конкретные краеугольные ситуации)

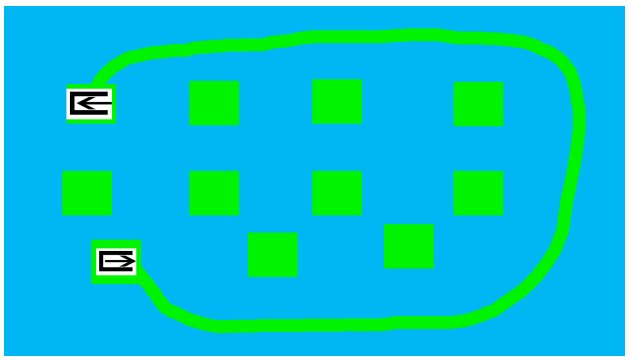
Пока алгоритм придумывал, накопилось много интересных ситуаций для тестов. Вот некоторые из них.

1. Статические трассы

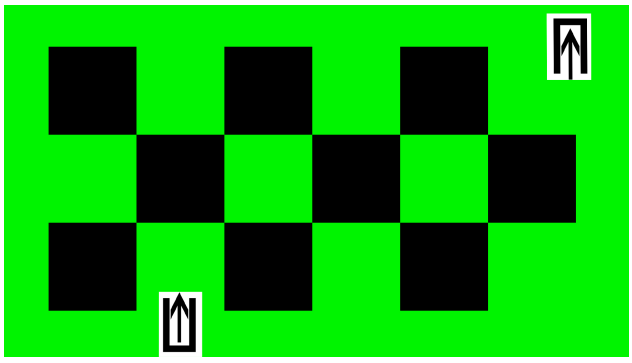
1.1. Часы



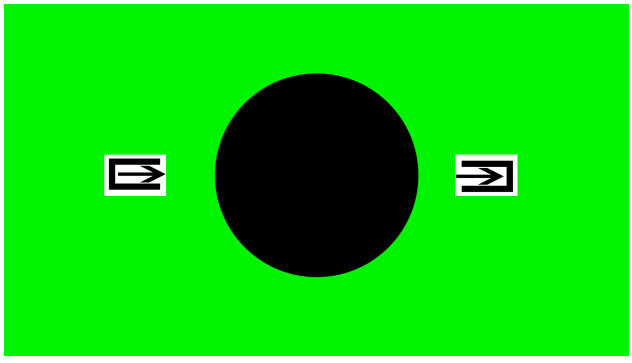
1.2. Острова



1.3. Шахматы

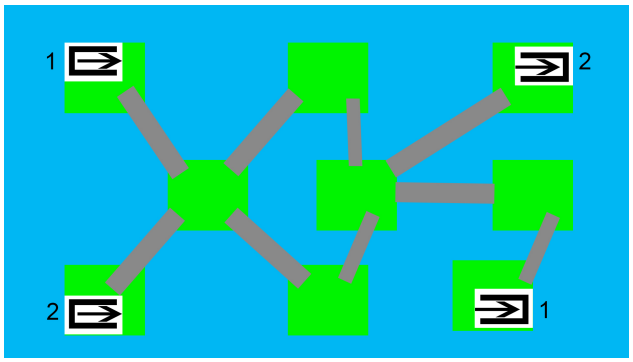


1.4. Обход большого камня

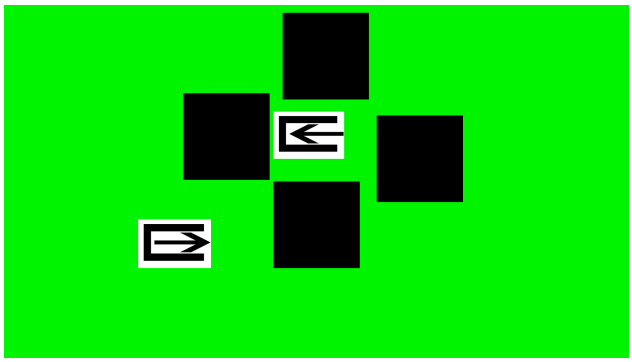


1.5. Длинный мост

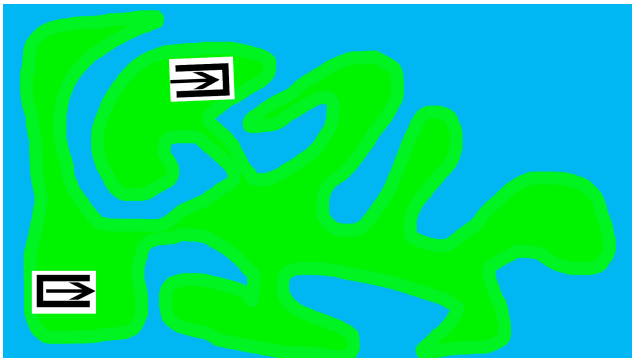
1.6. Много мостов



1.7. Проверка проходимости по размеру в ущелье



1.8. Амебное пятно



ВВЕДЕНИЕ

Централизованное управление автомобильным трафиком сделает движение более эффективным, устранит проблему пробок, позволит быстрее перемещаться.

В обозримом будущем машины будут по большей части управляться автопилотами.

Проблема пробок (и многие другие логистические проблемы) возникают вследствие плохо согласованного движения машин. Правила движения решают эти проблемы, но не исключают человеческий фактор (эгоистичное поведение, нарушение правил, медленная реакция). Централизованная раздача маршрутов автопилотам под управлением программы решает эти проблемы.

Проблему пробок невозможно решить другими способами, кроме как централизованный контроль и управление машины автопилотом. Меньше препятствий, перекрестков, более широкие дороги толком не помогут.

Более наглядно тонкости проблемы пробок можно увидеть на видео по ссылке в приложении под названием «Ты знаешь откуда возникают пробки?».

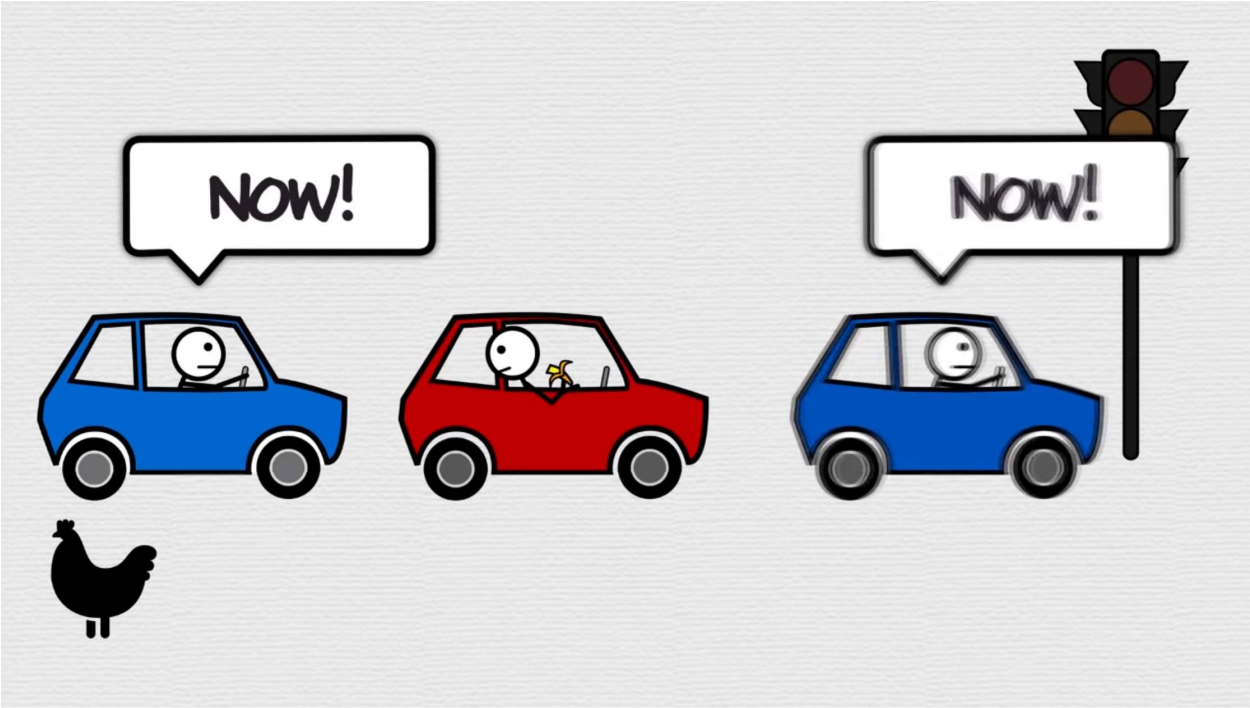


Рисунок 4 Человеческий фактор

При низкой координации движения машин (у людей только так) могут возникать фантомные перекрестки, из-за того, что кто-то притормозил, за ним следующий, а за ним следующий и т.д.

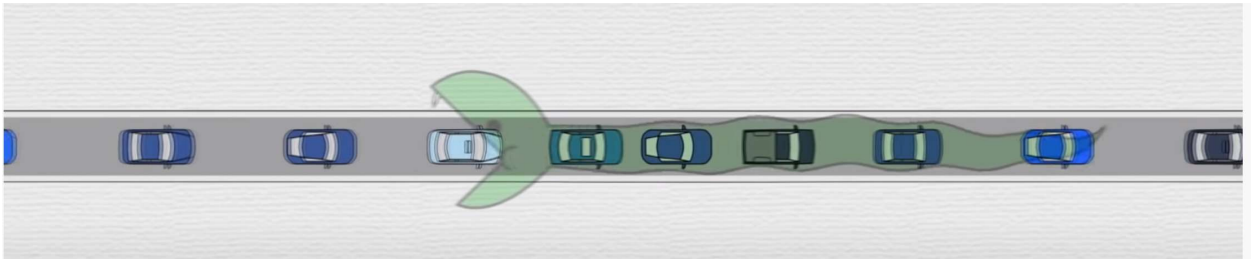


Рисунок 5 Фантомный перекресток.

Были проведены эксперименты. Из-за плохой человеческой координации могут возникать пробки даже при движении на круглой замкнутой трассе без препятствий, без светофоров с несколькими машинами, и где водители специально пытаются скоординироваться и не создавать пробок. Как бы они не старались, у водителей не получается скоординироваться.

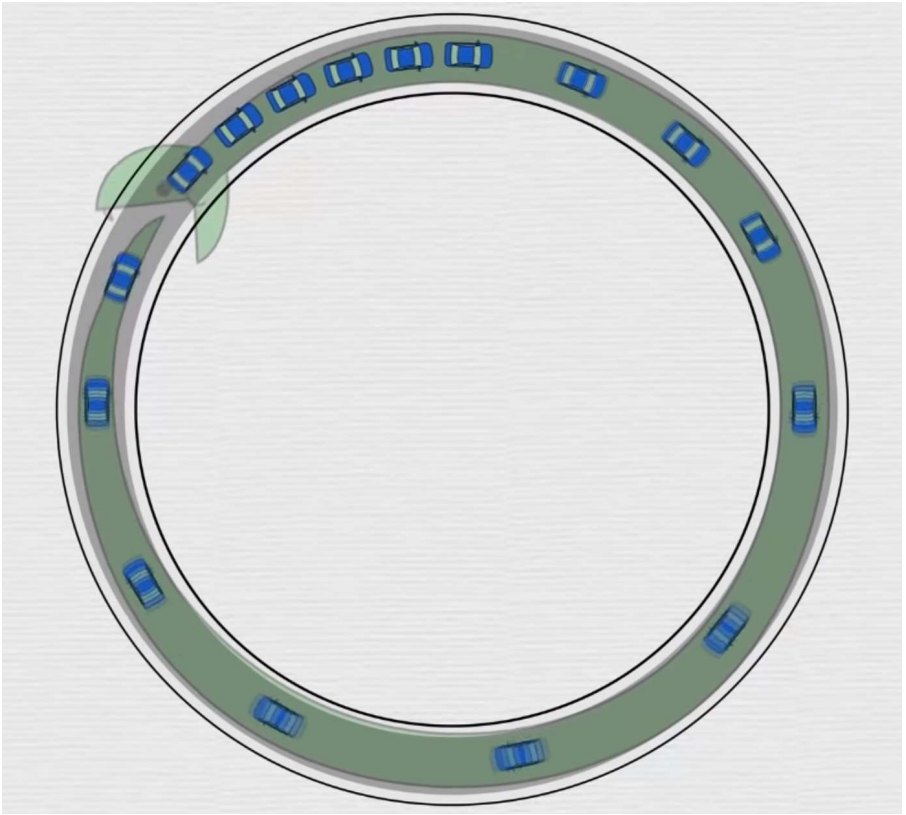


Рисунок 6 Пробка на кольцевой трассе без препятствий с небольшим количеством машин.

ОСНОВНАЯ ЧАСТЬ

1. Описание алгоритма

1.1. Краткое описание алгоритма

Пространственно-временной A-Star с ориентацией на заранее построенный граф эталонных маршрутов. Граф эталонных маршрутов построен при помощи алгоритма Флойда (хотя может быть будет Дейкстра использоваться).

Под ориентацией имеется ввиду построение приоритета обхода графа, при помощи эталонных маршрутов.

1.2. Описание подалгоритмов

1.2.1. Пространственно-временной

Это прилагательное означает, что алгоритм смотрит в точке не только на наличие постоянного препятствия от ландшафта, но и наличие другой машины, в данный момент времени находящейся в этой точке.

1.2.2. A-Star

A-Start делает обход в глубину по приоритету, основанном на весе узла. Алгоритм оценивает стоимость каждого ближайшего узла и выбирает самый короткий маршрут до узла назначения (минимальная стоимость). Потом процесс повторяется заново, относительно выбранного узла.

Стоимость узла определяется:

1) реальным пройденным расстоянием до узла
2) оценка эвристической функции, которая на глаз определяет стоимость пути до точки прибытия.

   Как правило используют функцию, определяющее прямое расстояние между точками на плоскости. Я буду использовать функцию, ориентирующуюся на заранее построенный граф эталонных кротчайших маршрутов, построенный алгоритмом Флойда.

1.2.3. Алгоритм Флойда

Алгоритм нахождения кратчайших расстояний между всеми парами вершин во взвешенном ориентированном графе. Используется для построения эталонных маршрутов.

1.3. Шаги алгоритма (пример работы) - Индексирование карты

Проводится один раз при инициализации карты, или при весомом изменении ландшафта.

1) **Разбиваем карту на квадратные секторы**
2) **Каждый сектор покрываем ТПД (точки прямого доступа).**

   В из каждого участка карты можно будет попасть в ближайшую точку прямого доступа по прямой линии.
3) **Находим кратчайшие пути между всеми парами вершин**

1.4. Шаги алгоритма (пример работы) – Поиск кратчайшего пути для машины

1) **Нахождение ближайшей ТПД (точки прямого доступа)**
2) **Построить ворота коэффициента наклона (узкие места) Узкие места считаются только от непроходимых препятствий.**

   Оптимальных проход может пролегать и свозь зоны с разной проходимостью. Оптимальный маршрут определяется на стадии индексации карты. Потом надо ориентироваться по нему, остальное мало что значит.
3) **Построить линию стыка двух маршрутов под разным углом**
4) **Проверка ближайшей приоритетной точки на наличие там машины. (и так по всему маршруту, с откатами назад, в случае тупиков)**

2. Реализация

2.1. Создана гибкая архитектура с возможностью делать алгоритмы разного качества, постепенно повышая его, по мере возможностей.

В архитектуре указаны основные классы. Вспомогательные классы не указаны.

Все обращения к классам идут через интерфейсы.

Бизнес-логика программы защищена от всех сторонних ресурсов адаптерами и интерфейсами (не имеет привязанности к конкретным отрисовщикам, системам подачи команд, фреймворкам, базам данных и т.п.).

Система сделана по принципу Model – View – Controller. А именно: цепочка вызовов классов не циклична (в целом).

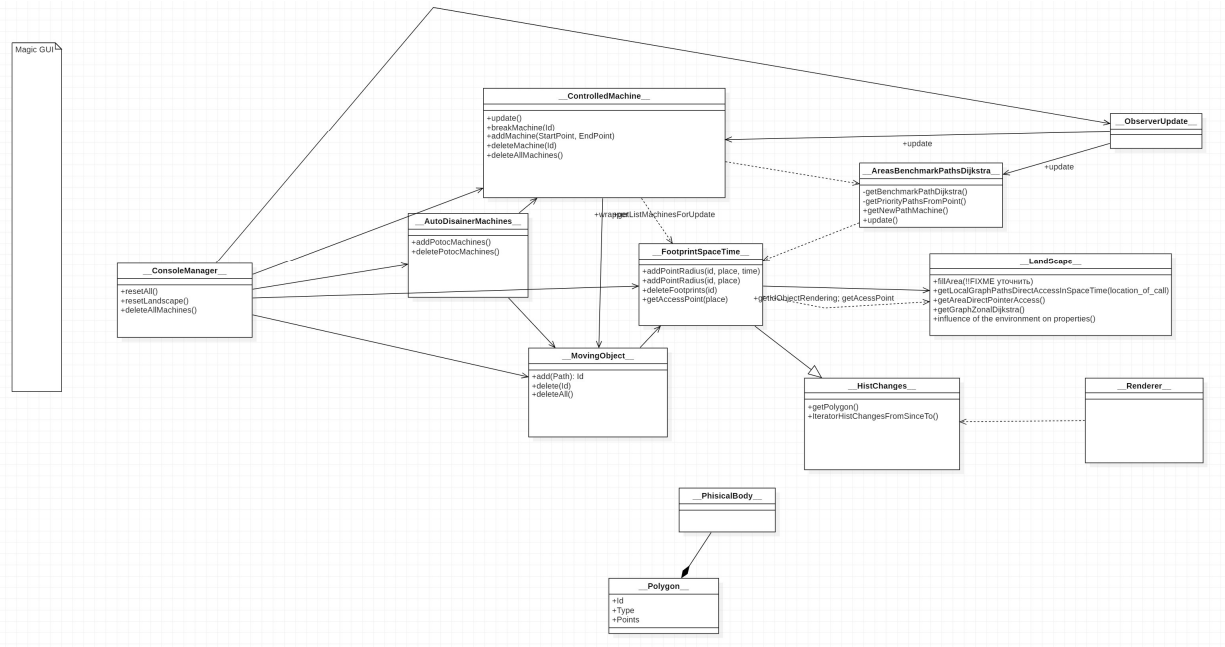


Рисунок 7 Полная UML диаграмма всего проекта

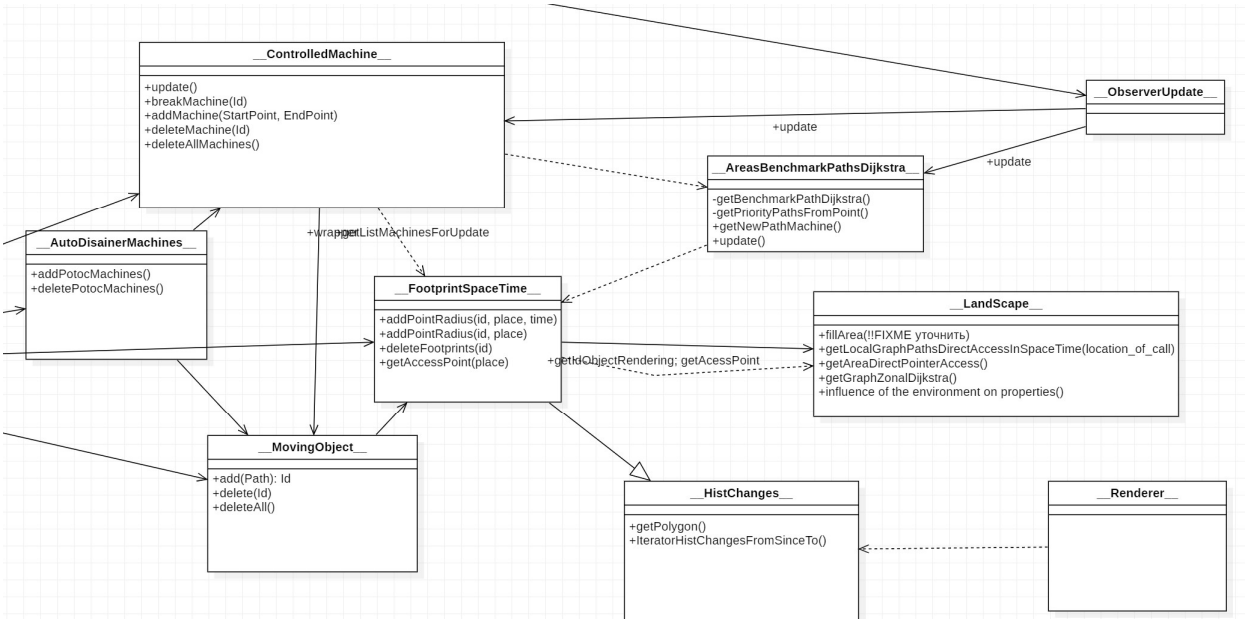


Рисунок 8 Главная часть UML диаграммы проекта

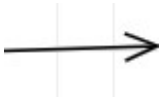


Рисунок 9 Обращение к другому классу с изменением объекта

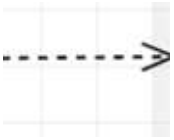


Рисунок 10 Обращение к другому классу, получение данных без изменения объекта

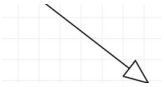


Рисунок 11 Реализация интерфейса

2.1.1. ConsoleManager

Используется управление текстовыми командами. Такой интерфейс был выбран, как максимально гибкий. ConsoleManager перерабатывает текстовые команды в вызовы функций определенных классов.

2.1.2. Observer

Класс Наблюдатель. ConsoleManager после некоторых изменений (например ландшафта) сообщает Наблюдателю о необходимости обновления. Наблюдатель передает оповещение об обновлении соответствующим классам в нужном порядке.

2.1.3. MovingObject

Класс занимается созданием объектов машин, их учетом, отвечает за оставление следов в объекте класса FootprintSpaceTime.

2.1.4. AutoDisainerMachines

Класс отвечает за функционирование потоков, создающих машины с заранее заданной точкой прибытия.

2.1.5. ControlledMachine

Класс отвечает за предоставление маршрута машине из точки A в точку B. Собственно говоря, здесь и будет работать главный логистический алгоритм (см. пункт 1).

2.1.6. FootprintSpaceTime

Пространственно-временные следы. Главный класс в проекте, вокруг которого все работает. Класс отвечает за

1) Хранение информации о положении машины в пространстве и времени
2) Выдачи информации о занятости места определенной машиной
3) Решение конфликтов столкновений машин, зданий и т.п.

2.1.7. AreasBenchmarkPaths

Этот класс отвечает за составление эталонных маршрутов из точек прямого доступа.

2.1.8. LandScape

Подкласс FootprintSpaceTime. Отвечает за выдачу информации о проходимости через статические объекты (неподвижные объекты).

2.1.9. HistChanges

Возвращает изменения за запрошенный промежуток времени. Еще есть функционал выдать все элементы в определенный момент времени.

2.1.10. Render

Отрисовщик определенной области карты.

2.2. Сделана часть функционала симуляции. Квадрат может перемещаться по заранее заданному маршруту. Этот функционал пронизывает всю систему. Осталось сделать алгоритм, составляющий эти инструкции.

Симуляция сделана на JavaFX.

3. Экспериментальная часть

К тестированию алгоритма еще не приступил. Пока что симуляция не закончена.

Пока что есть только перемещение машины по заранее заданному маршруту. Эта функция пронизывает всю архитектуру проекта, реализована на всех уровнях.

ЗАКЛЮЧЕНИЕ

Был продуман алгоритм, гибкие интерфейсы классов. Частично создана симуляции перемещения машин (добавлена функция создавать машину с маршрутом, эта функция пронизывает всю архитектуру проекта).

В дальнейшем будет доделала симуляция (непроходимые препятствия, столкновения машин и т.п.) и будет запрограммирован алгоритм, раздающий маршруты для автопилотов.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Курносов М.Г. Введение в структуры и алгоритмы обработки данных. М. : Автограф, 2015. 12с.

2. Алгоритм поиска A*. URL: https://www.youtube.com/watch?v=AsEC2TJZ3JY (дата обращения: 11.05.2020)

3. Алгоритм Флойда. URL: https://www.youtube.com/watch?v=HwK67u7zaEE (дата обращения: 11.05.2020)

4. Поиск пути в играх. Алгоритм поиска пути A*. URL: https://www.youtube.com/watch?v=gCclsviUeUk (дата обращения: 11.05.2020)

5. Ты знаешь откуда возникают пробки? URL: https://www.youtube.com/watch?v=xwTMmdeLRKI (дата обращения: 11.05.2020)

6. Базовые алгоритмы нахождения кратчайших путей во взвешенных графах / Хабр. URL: https://habr.com/ru/post/119158/ (дата обращения: 11.05.2020)

7. GeorgiaFrankinStain/Centralized_Control_of_Autopilots_diploma: This is my diploma. URL: https://github.com/GeorgiaFrankinStain/Centralized_Control_of_Autopilots_diploma (дата обращения: 11.05.2020)

ПРИЛОЖЕНИЯ

Иерархия файлов

Программный код можно посмотреть по ссылке в списке литературы на GitHub.