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

Задания:

1. Разработать централизованный логистический алгоритм (в дальнейшем называемый ЦКА (Центральный Контроль Автопилотов)).

   Задачи алгоритма:

   1.1. раздать маршруты движения каждой подключенной автопилотируемой машине

   1.2. организовать движение без пробок, заторов и т.п. (это не обязано быть похожим на правила дорожного движения)

2. Разработать симуляцию движения машин, для отладки и демонстрации работы алгоритма //FIXME здесь писать в какой проге я это сделаю?

Допущения и условности

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

   Оправдано тем, что около машины будет достаточно свободного места для поворота и скорость перемещения будет до 40 км/ч.

2. Движение 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 секунда (инструкции под новые условия должны появиться уже через секунду).

   Машины перемещаются по квадратам, около каждой машины спереди всегда есть свободный квадрат. На тот момент, когда машина переместиться на следующий квадрат, уже должны быть новые инструкции под новые условия.

4. Машины к системе ЦКА могут подключаться/отключаться внезапно.

Техническое задание для симуляции

в техническом задании требуются определения. Что с этим делать? Где их указать? я указывал их здесь же, это нормально?

1. Создание и редактирование карты на ходу
   1.1. Добавление/удаление зданий
   1.2. Изменение ландшафта
2. Управление потоком машин на ходу

   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.

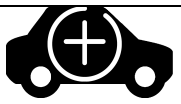
   2.1. Добавление/удаление потоков
   2.2. Регулирование скорости создания машин у потока
   2.3. Поломка машин
   2.4. Назначение точки прибытия машинам в системе ЦКА, создающихся потоком.

   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 создающий	
Добавить поток поглощающий	
Добавить машину	
Удалить (ластик, очистить)	
Перевести в состояние «разрушен»	

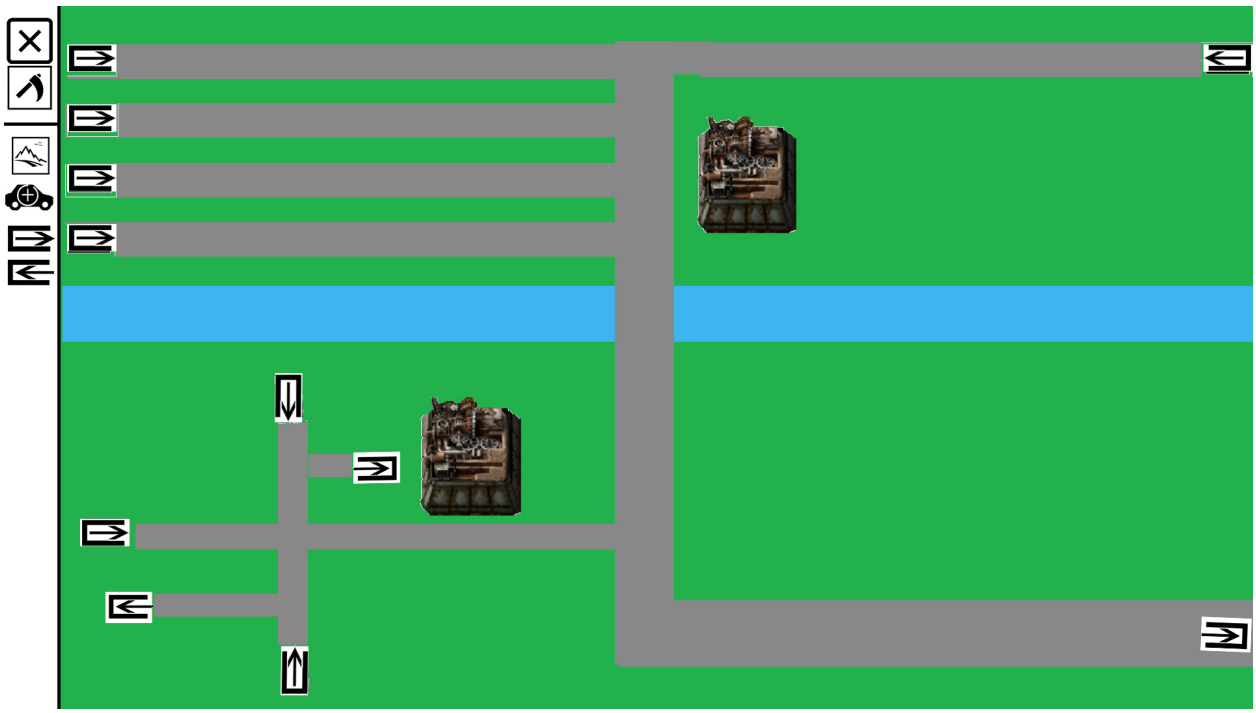


Рисунок 1 Примерный вид графического интерфейса

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

ВВЕДЕНИЕ

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

В обозримом будущем машины по большей части будут управляться автопилотами.

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

Более наглядно тонкости проблемы пробок можно увидеть в 5-минутном видео на русском языке по ссылке (https://youtu.be/xwTMmdeLRKI). (как лучше дать ссылку?)

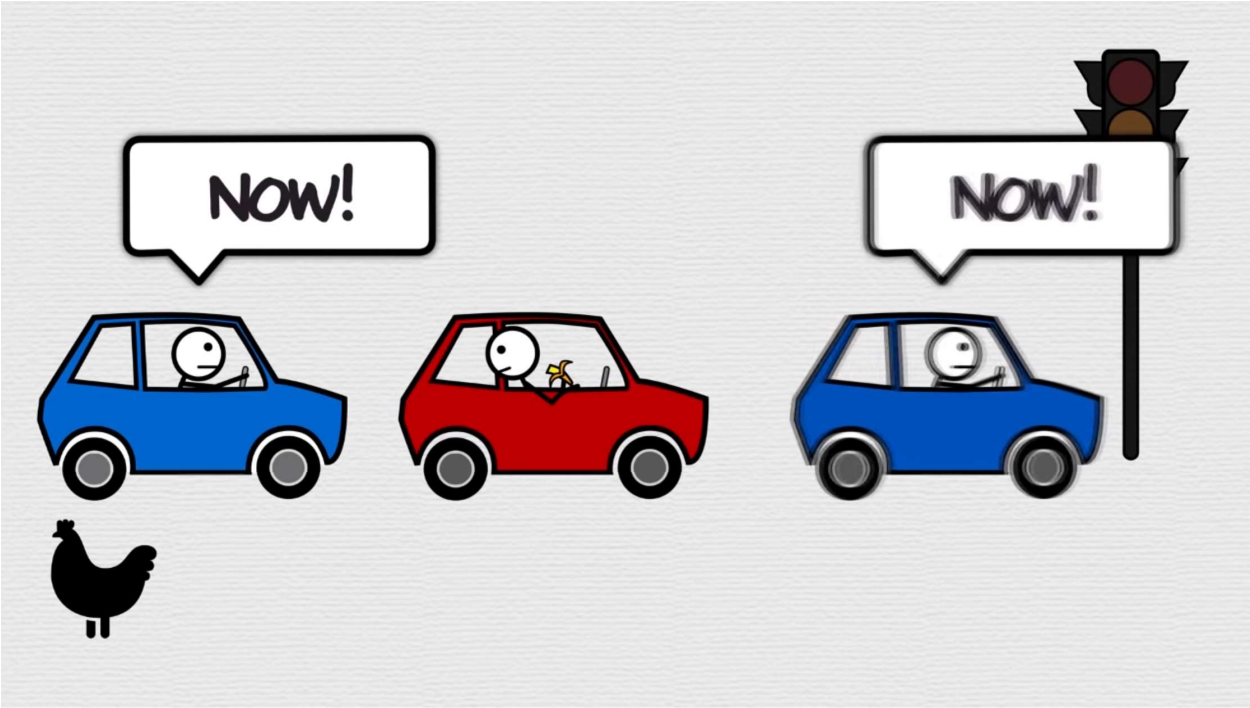


Рисунок 2 Человеческий фактор [https://youtu.be/xwTMmdeLRKI?t=25]

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

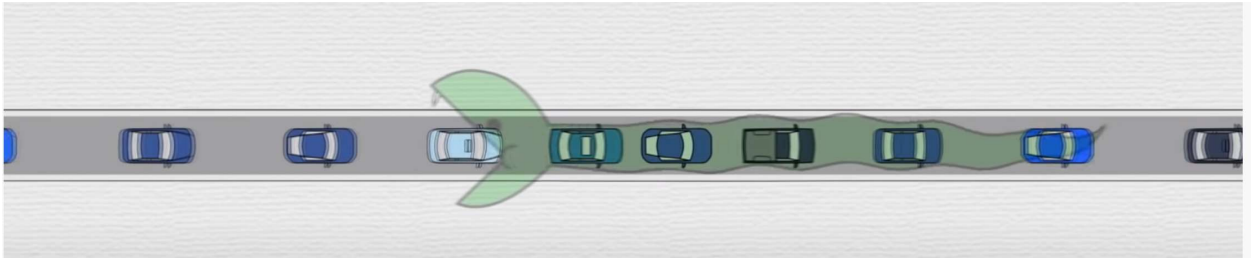


Рисунок 3 Фантомный перекресток. [https://youtu.be/xwTMmdeLRKI?t=109]

Из-за плохой человеческой координации могут возникать пробки даже при движении на круглой замкнутой трассе без препятствий, без светофоров с несколькими машинами.

Пробки могут возникать при отсутствии препятствий, даже когда водители специально пытаются скоординироваться и не создавать пробок. Как бы водители не старались, у них не получается двигаться организованно более 5 минут. (ссылка на видеозапись проведенного эксперимента https://vk.com/video2399234_136124429)

//FIXME ссылку на доказательство в списке литературы указать, или здесь?

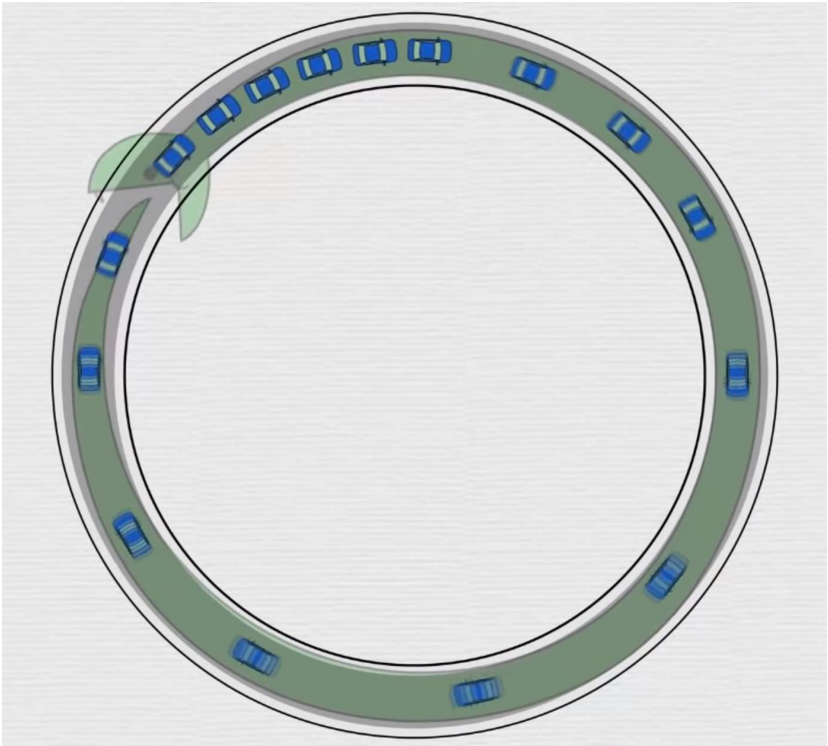


Рисунок 4 Пробка на кольцевой трассе без препятствий с небольшим количеством машин. [https://youtu.be/xwTMmdeLRKI?t=117]

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 5 Поток создания машин, обозначение на карте

Рисунок 6 Поток поглощения машин, обозначение на карте

ОСНОВНАЯ ЧАСТЬ

Описание алгоритма ЦКА (Центрального Контроля Автопилотов)

Алгоритм был продуман, но не реализован в коде на данный момент.

Алгоритм раздает маршруты автопилотируемым машинам, чтобы обеспечить эффективное движение на дороге без пробок.

Краткое описание алгоритма

Пространственно-временной A-Star с ориентацией на заранее построенный граф эталонных маршрутов.

Граф эталонных маршрутов построен при помощи алгоритма Флойда (но возможно будет использоваться алгоритм Дейкстры).

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

Описание частей алгоритма ЦКА

1. **Пространственно-временной**

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

2. **A-Star**

A-Start делает обход дерева вариантов маршрута 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 приблизительно определяет стоимость пути до точки прибытия.

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

3. Алгоритм Флойда

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

Псевдокод алгоритма Флойда.

На каждом шаге алгоритм генерирует матрицу $W, w_{ij} = d_{ij}^{k}$ Матрица W содержит длины кратчайших путей между всеми вершинами графа. Перед работой алгоритма матрица W заполняется длинами рёбер графа (или запредельно большим M, если ребра нет).

```
for k = 1 to n
  for i = 1 to n
    for j = 1 to n
      W[i][j] = min(W[i][j], W[i][k] + W[k][j])
```

Шаги алгоритма ЦКА (пример работы) - Индексирование карты

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

1) Разбиваем карту на квадратные сектора
2) Каждый сектор покрыт ТПД (точками прямого доступа).

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

Шаги алгоритма ЦКА (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.

   «Ворота» - это проход через наиболее узкое место. Крайние точки непроходимой зоны проецируются на прямую эталонного маршрута.

   Прямой маршрут машины поворачивается, чтобы попасть в «ворота».

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

Реализация

Создана гибкая архитектура.

В UML диаграмме 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 функций у классов не циклична. ConsoleManager это Controller, Render – это View, все остальные классы это Model.

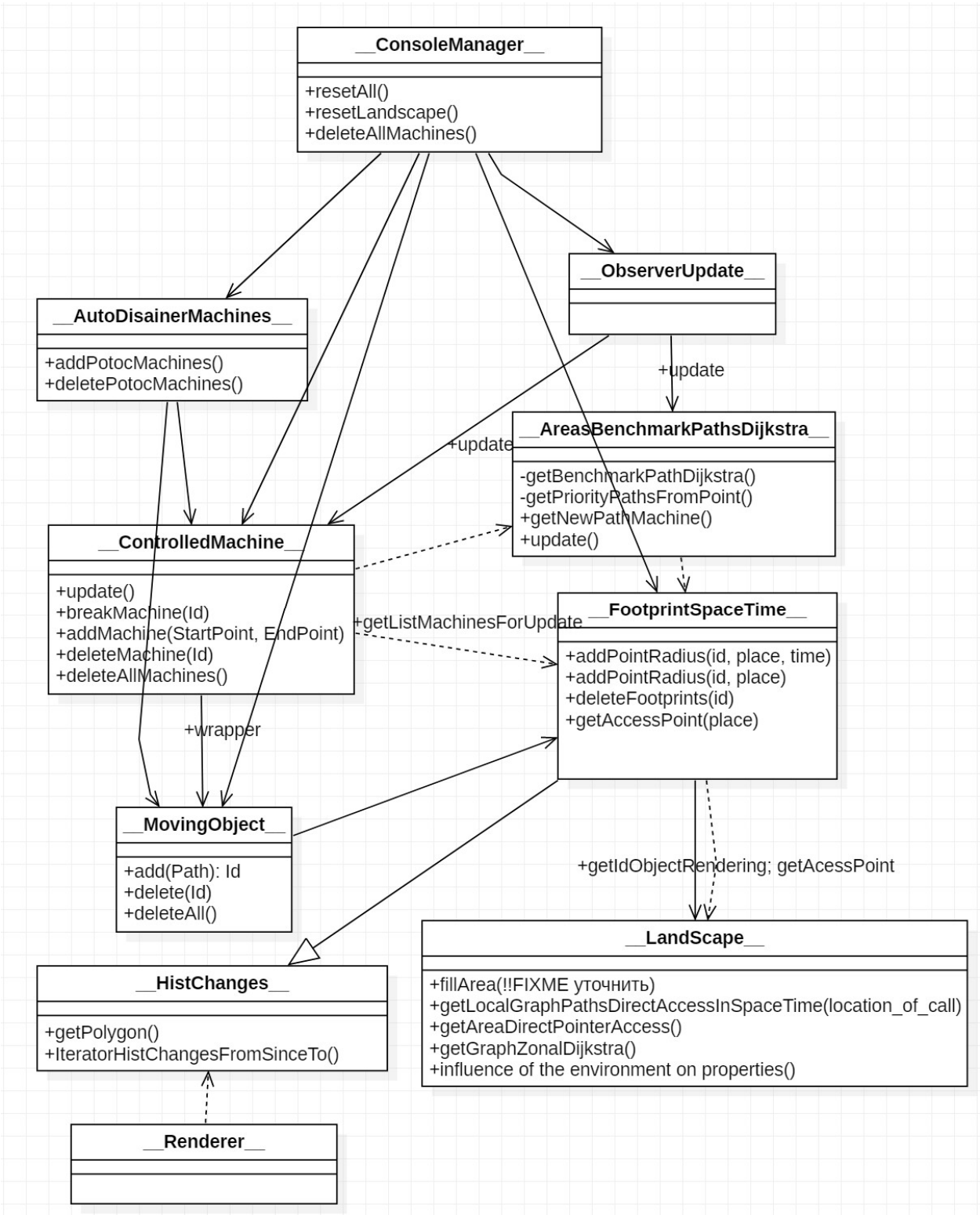


Рисунок 7 UML диаграмма проекта

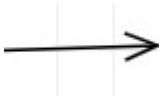


Рисунок 8 Обращение к другому классу с изменением объекта

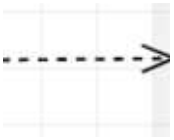


Рисунок 9 Обращение к другому классу, получение данных без изменения объекта

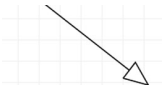


Рисунок 10 Реализация интерфейса

1. **ConsoleManager**

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

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.

3. **MovingObject**

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

4. **AutoDisainerMachines**

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

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).

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.

7. **AreasBenchmarkPaths**

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

8. **LandScape**

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

9. **HistChanges**

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

10. **Render**

Отрисовщик определенной области карты.

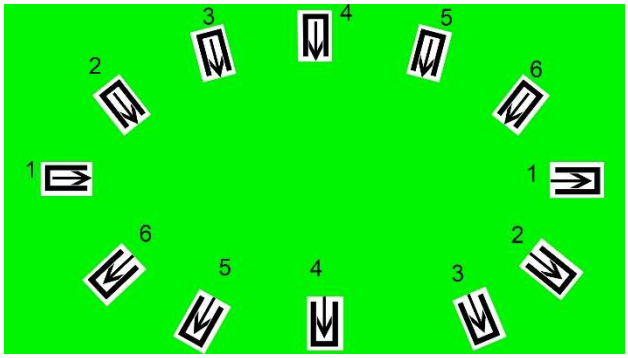
Тестовые карты

Карты, на которых алгоритм должен работать. Карты тестируют краеугольные ситуации, на которых с большой вероятностью проявятся неисправности алгоритма.

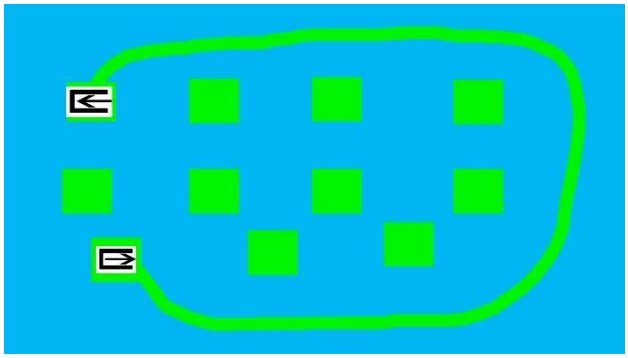
Значение цветов на карте

6. Зеленый – трава, зона обычной проходимости.
7. Синий – река (непроходимо для обычных машин)
8. Серый – асфальт, зона самой высокой проходимости
9. Картинка здания – здание, непроходимо
10. Черный – непроходимое препятствие

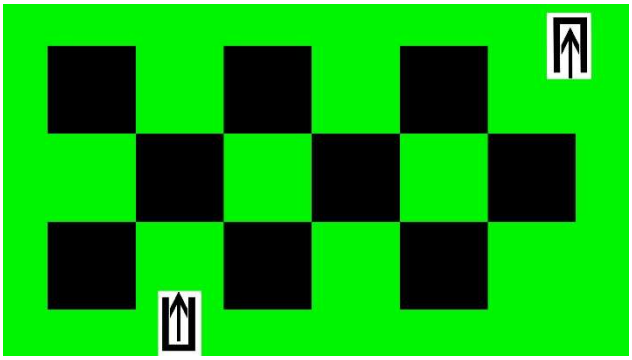
1. Часы



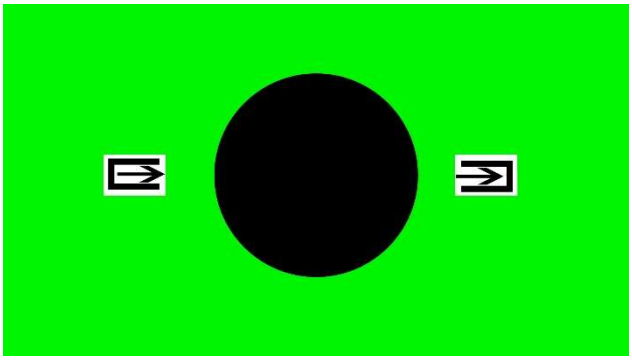
2. Острова



3. Шахматы

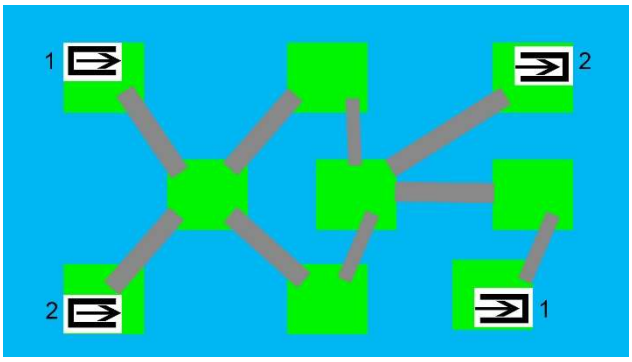


4. Обход большого камня

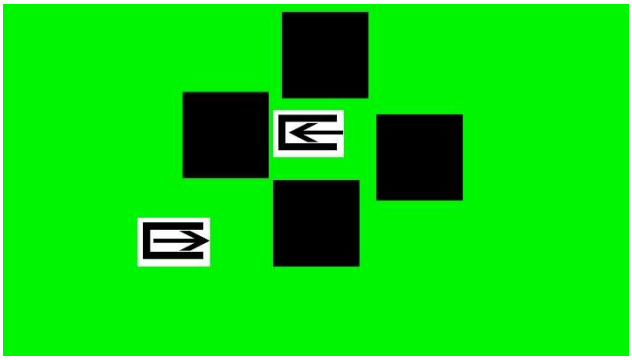


5. Длинный мост

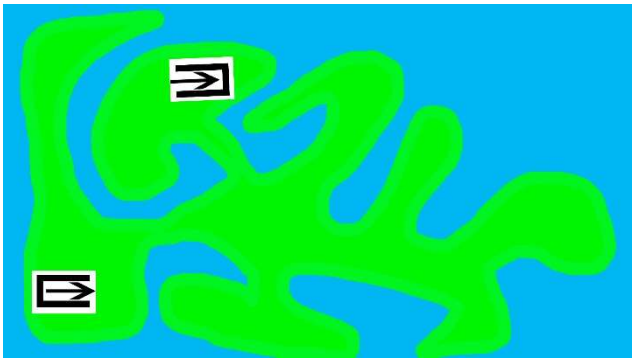
6. Много мостов



7. Проверка проходимости по размеру в ущелье



8. Амебное пятно



Экспериментальная часть

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

Пока что реализовано только перемещение машины по заранее заданному маршруту. Создается объект машины. В класс FootprintSpaceTime прописываются следы движения машины на протяжении всего времени следованию маршруту. Render отрисовывает перемещения машины, запрашивая данные из FootprintSpaceTime.

ЗАКЛЮЧЕНИЕ

Был продуман алгоритм, гибкие интерфейсы классов. Частично создана симуляции перемещения машин (добавлена функция создавать машину с маршрутом, и симуляция ее движения).

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)

8. Видеозапись проведенного эксперимента с движением машин по кругу без препятствий URL: https://vk.com/video2399234_136124429 (дата обращения: 11.05.2020)

ПРИЛОЖЕНИЯ

Иерархия файлов

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