Введение

Каждый день мы прокладываем путь из одной точки в другую, будь то посещение работы, или любого другого места, находящегося вдали от дома. Выбирая оптимальный маршрут, мы учитываем такие факторы как: время, которое мы потратим на дорогу, расстояние, на которое мы собираемся удалиться от начальной точки и, конечно, стоимость. Но один из факторов мы доселе не могли предусмотреть. Мы не могли знать актуальную информацию о пробках на дороге. Эта проблема стала перед человечеством не так давно, но уже представляет большую угрозу для всех планов, которые включают в себя перемещения по городу. По статистике «Яндекс» от 2007 года каждый москвич в среднем проводит 11 часов каждого месяца в пробках. Но с появлением таких технологий как «GPS», «ГЛОНАСС» стало возможным обновлять сведения о состоянии пробок в режиме реального времени. Это позволяет автомобилистам обходить проблемные места на дорогах и, если не полностью, то в большей мере экономить время.

Для достижения максимальной эффективности мы стараемся сократить время и расходы на дорогу. Каждая пробка – серьезная задержка в пути. Гораздо лучше использовать обходной путь, на котором не затруднено движение. Так же немало важно иметь актуальные данные о пробках, иначе желание сократить время может сыграть злую шутку.

Для нахождения оптимального пути необходимо получить сведения со спутника о состоянии и плотности трафика на дорогах. Каждый участок пути имеет несколько состояний, которые отражают наличие или отсутствие пробки. Затем, используя знания о пробках, находится кратчайший путь, который будет исключать проблемные места на дорогах. Такой путь и является оптимальным. Теперь этот путь может использовать автолюбитель, желая не опоздать на работу.

Теория графов может быть применена в экономических задачах. В логистике взаимоотношения между компаниями могут быть выражены в виде графа, в котором вершинами являются предприятия, а ребрами – взаимосвязи. Для поиска оптимального решения часто бывает необходимо найти минимальный путь от одной вершины к другой.

Обоснование цели и задач проекта

Данную тему: «Динамический электронный навигатор», я выбрал из-за большой актуальности проблемы пробок на дорогах. Для себя я поставил цели:

- разработать программный продукт, который бы осуществлял поиск минимального пути в графе для решения указанной проблемы;
- наглядно продемонстрировать его работу на примере смоделированной среды;

- разработать электронное пособие для дальнейшего использования на уроках дискретной математики.

В качестве первоочередной задачи я поставил перед собой изучение теории графов и сконцентрировал внимание на алгоритме фронта волны. Для создания программного продукта я расширил знания языка программирования «Blitz3D», а для моделирования среды – приобрел навыки работы в программе «3Ds MAX».

Методы достижения целей и задач

Для достижения поставленной цели я изучил теоретический материал, связанный с теорией графов, углубил знания в специализированном языке программирования «Blitz3D» и освоил основные операции по работе с 3D-моделированием.

Алгоритм фронта волны - волновой алгоритм поиска пути на карте, алгоритм трассировки. С его помощью можно построить путь, или трассу, между двумя любыми элементами в лабиринте. Из начального элемента распространяется в четырёх направлениях волна. Тот элемент, в который она пришла, образует фронт волны. Элементы первого фронта волны являются источниками вторичных волн.

Элементы второго фронта генерируют волну третьего фронта и так далее. Процесс заканчивается тогда, когда достигается конечный элемент. На втором этапе строится трасса. Построение производится в соответствии с некоторыми правилами:

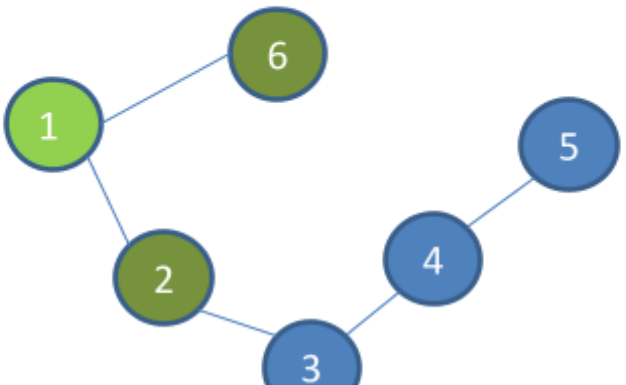
- при построении трассы движение проходит в зависимости от выбранных приоритетов
- путевые координаты уменьшаются при переходе от начального элемента к конечному.

Эти приоритеты выбираются в процессе разработки. В зависимости от выбора тех или иных приоритетов получаются различные трассы. Но в любом случае длина трассы остается одной и той же.

Проиллюстрируем алгоритм на конкретном примере:

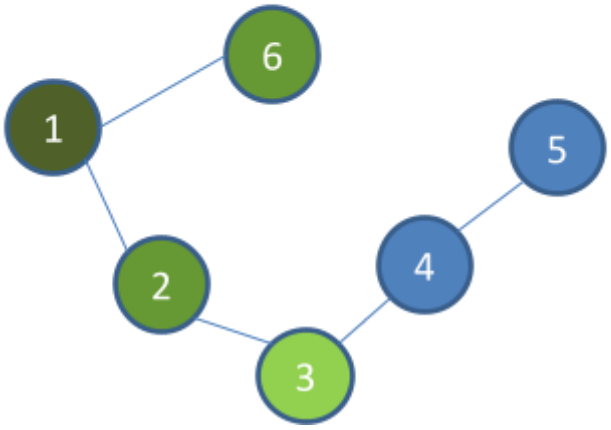
1. Находим вершины (образы) графа, с которыми соединена начальная вершина – фронт волны первого уровня.

FW1(1)={2,6}



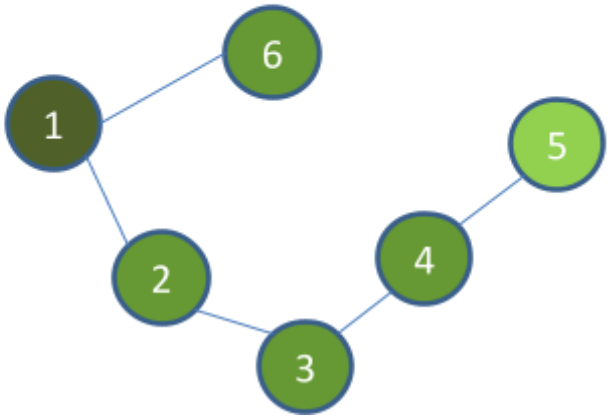
2. Находим точки, которые соединены с найденными в шаге 1, и выбрасываем те вершины, в которых побывали ранее – фронт волны второго уровня.

FW1(1)={2,6}
FW2(1)={3}



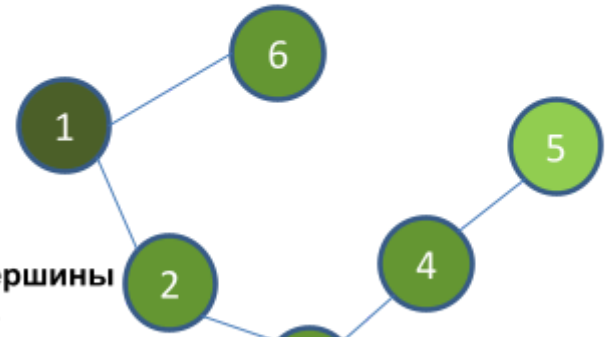
3. Продолжаем делать тоже самое, пока не достигнем конечной вершины.

FW1(1)={2,6}
FW2(1)={3}
FW3(1)={4}
FW4(1)={5}



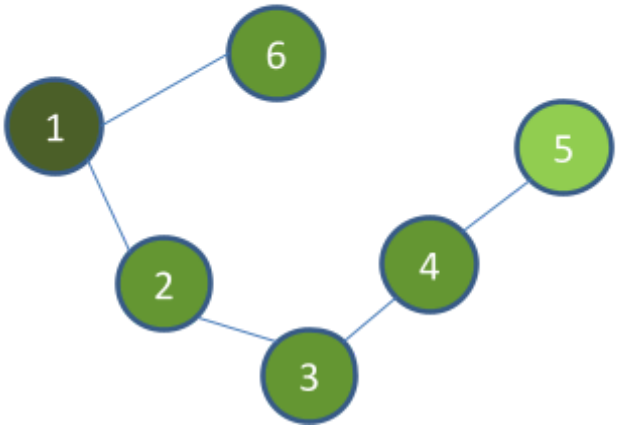
4. Ищем пересечение множества вершин последнего фронта волны с множеством прообразов конечной вершины.

FW1(1)={2,6}
FW2(1)={3}
FW3(1)={4}
FW4(1)={5}
FW3(1)∩{4}={4}



Прообраз некоторой вершины – вершина, из

5. Повторяем этот шаг, пока не дойдем до начальной вершины.



FW1(1)={2,6}
FW2(1)={3}
FW3(1)={4}
FW4(1)={5}
FW3(1)∩{4}={4}
FW2(1)∩{3}={3}
FW1(1)∩{2}={2}

Теперь перейдем от математики к программированию и рассмотрим техническую сторону проекта. Для программной реализации продукта был использован язык программирования и среда «Blitz3D». Среда специализирована для разработки компьютерных игр на базе графической технологии DirectX и отличается простым синтаксисом и широкими возможностями. Язык содержит 588 команд и позволяет создавать несложные 2D и 3D-миры.

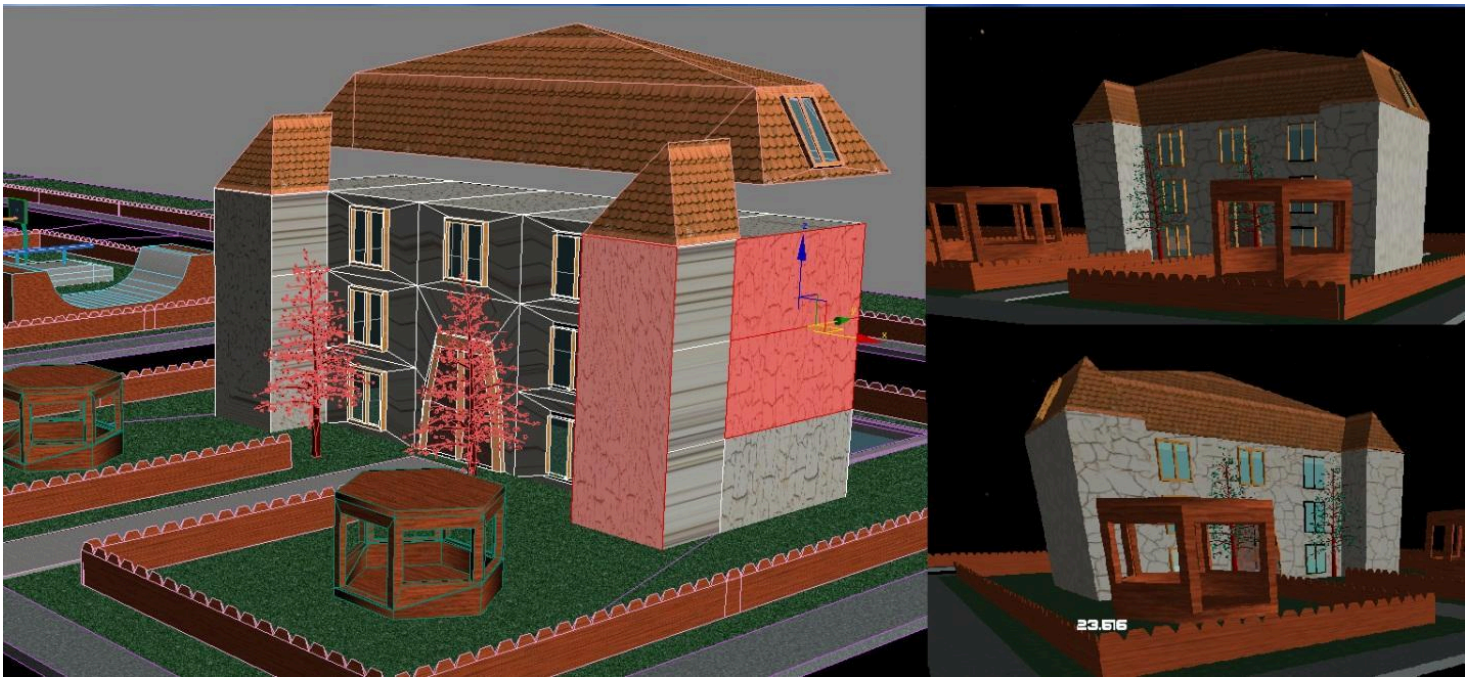
В программе участвует объект-автомобиль и объект-сцена. Сцена состоит из квартала условного города, и представляет собой граф, сторонами которого являются улицы, а вершинами – перекрестки. По этому участку города ездит автомобиль, водителем которого является конечный пользователь программы. Так же на верхней части экрана расположена мини-карта, которая показывает текущее положение автомобиля и состояние пробок. При нажатии на участок пути он блокируется, и программа ищет новый путь, если это возможно.

На скриншоте изображена среда разработки «Blitz3D» и фрагмент кода программы, который является частью алгоритма фронта волны. Так же, справа, виден результат работы алгоритма. Эти данные отладочные, и конечный

```
For i= 1 To 12 ; Цикл по количеству вершин. Проверяем  вершину на ребро с остальными

If matrix(start,i) =1 And start <> i ; Если есть ребро между вершинами и это не одна и та же вершина
   If matrix(start, finish) = 1 Return ; Если есть ребро между начальной и конечной точкой - выходим

   If DimTest(i) = 0 ; Если в массиве шага еще не было этой вершины
      For k= 1 To 12 ; Ищем последний пустой элемент
         If Last_points(1,k) = 0 ; Если найден пустое место
            Last_points(1,k) = i ; Добавляем в последний элемент

            If matrix(i, finish) =1 Goto back ; Если это конечная вершина - переходим ко второй части
            Steps=Steps+1 ; Наращиваем количество шагов
            Exit ; Выход цикла
         EndIf

      Next
   EndIf
```

пользователь не будет их видеть. В качестве результата он получит обозначения на мини-карте, которым он может следовать.

Так же, для создания 3D-моделей, использовалась программа «3Ds MAX 9», в которой были созданы все объекты, участвующие в программе. После наложения текстур модели были экспортированы в программу, где, впоследствии, из них был сформирован участок города.



Практические, инновационные результаты работы

1. Решение задач имеет прикладной характер.
2. Использование на уроке и внеурочной деятельности.
3. Соединение возможностей дискретной математики и программирования и наглядная демонстрация.

Выводы и рекомендации

Достоинствами работы является возможность редактировать маршрут в реальном времени, графическое представление работы алгоритма, широкая область применения в практике, и, самое важное, актуальность проблемы в повседневной жизни.

В качестве недостатков стоит отметить невозможность расчета времени пути, пренебрежение расстоянием между вершинами и ресурсоемкость графической среды.

Использованная литература

1. Дискретная математика : учебник для студ. учреждений сред. проф. Образования / М. С. Спирина, П.А. Спирин. – 6-е изд., стер. – М.: Издательский центр «Академия», 2010, – 368 с.
2. Программирование игр. Автор - Маниш Сети, Год выпуска: 2006, Издательство – Технический бестселлер.
3. Википедия - http://ru.wikipedia.org/