Maze_solver

Описание

Эта программа для нахождения пути из лабиринта. Она принимает бмп изображение и возвращает файл output.bmp с нарисованным путём. Помимо всего прочего в репозитории лежат инструменты для отладки.

Принцип работы

алгоритм

1. Побайтово считывает изображение
2. Формирует матрицы для лабиринта
3. Волновым алгоритмом находит путь
4. Рисует его
5. Сохраняет output.bmp

считывание

Алгоритм использует побайтовое считывание. Сначала он берёт 14 байтов шапки, оттуда смотрит количество байтов в DIB и считывает их. Для чтения он использует unsigned char, которые хранятся в одном байте и принимают значения от 0 до 255 (если одно число закодированно несколькими байтами, то мы используем сдвиг на 8 бит влево для каждого последующего символа). Через это мы находим ширину, высоту, и колличество байт на пиксель, потом вычисляем паддинг (паддинг ставиться после пикселя, для приведения количества пикселей в ширине к кратному 4 числу).

формирование матриц

Затем он считывает попиксельно байты для хранения цветов и сохраняет их для output, а также формирует три матрицы для волнового алгоритма (булеву со стенками, булеву для посещённых клеток, с интами для отметки расстояний). Благодаря этому для формирования вывода нам не нужно с нуля генерить файл, только поменять цвета пикселей на пути из лабиринта (а шапка и DIB от старого файла подойдут к этим данным).

Также здесь алгоритм запоминает старт и финиш.

Волновой алгоритм

Задание рекомендовало использовать волнвой алгоритм для нахождения пути, ведь для его использования нам не нужно дополнительно обрабатывать изображение и, например, переводить его в тот же граф. Этот алгоритм помимо всего прочего ишет кратчайший путь, что хорошо.

Сначала мы обозначаем координаты старта за 0. Затем для каждого из его четырёх (верх, нииз, лево, право) соседей обозначаем из координаты как координаты предыдущего +1.

Для координат я использую собственный класс Point, для которого я добавил метод neighboors (возвращает 4 соседей) и поля отцовских координат.

Пометка происходит через очередь.

- Берётся её начало
- От начала берутся соседи и добавляются в конец очереди
- Маркируем по алгоритму
- Удаляем начало
  Это помогает нам равномерно распространять волны, предотвращая уходы в о одну конкретную сторону и маркирование клетки большим чем нужно числом.

Рисование и сохранение

Далее остаётся от координат финиша по убывающим на 1 клеткам пройтись к началу и в сохранённой нами матрице пикселей заменить цвет по всем координатам на цвет пути(Красный)

Потом мы сохраняем побайтово изображение

- вставляем шапку
- вставляем DIB
- вставляем матрицу пикселей