Науково-практичний звіт на тему

НАЙКОРОТШІ ШЛЯХИ НА МНОЖИНІ ПЕРЕШКОД У 2D

М. С. Кушніренко, студентка 3 курсу, групи ІПС-31

Анотація. У роботі запропоновано узагальнений метод побудови шляхів на заданій множині h-перешкод, для кожного розташування запитних точок S і T. В даному випадку застосовується граф видимості для пошуку всіх можливих варіантів шляхів між двома заданими точками. Шуканий шлях між запитними точками є найкоротшим в графі видимості, вершинами якого є точки S і T, а також полігон вершин перешкод. До знайденої множини шляхів застосовано алгоритм Дейкстри.

Abstract. In the paper we proposes a generalized method for constructing paths on a given set of h-obstacles for each location of query points S and T. In this case, a visibility graph is used to search for all possible path variants between two given points. The required path between the query points is the shortest in the visibility graph, the vertices of which are the points S and T, as well as the polygon of the vertices of the obstacles. The Dijkstra's algorithm was applied to the found set of paths.

1 Вступ

Постановка проблеми. В роботі розглядається один із підходів розв'язання задачі пошуку найкоротших шляхів на множині перешкод у. Розв'язки цієї задачі мають велике теоретичне і практичне значення, особливо, що стосується застосування. Наприклад, одним із таких застосувань розробка ігор. Для якісної розробки гри, у якій герой рухається по карті, де крім нього є й інші предмети, розробнику потрібно вирішувати задачу пошуку шляху між перешкодами, що знаходяться на його шляху. На жаль, не завжди приділяється достатньо уваги даному питанню. Як результат ми отримуємо «всемогутніх» героїв, що можуть проходити крізь стіни, інші предмети та один крізь одного, що значно знижує задоволеність користувачів продуктом.

Сучасним застосуванням графів видимості є планування руху роботів на просторі з перешкодами, обчислення положення радіоантен або як засіб в архітектурі та містобудуванні при аналізі видимості.[1]

Аналіз останніх досліджень. На сьогоднішній день задача обчислення найкоротшого шляху у зваженому графі розглянута у багатьох класичних книгах, присвячених графічним алгоритмам, де описані алгоритм Дейкстри [2] й інші розв'язки.

Також набула популярності геометрична версія проблеми найкоротшого шляху. В цій роботі використовується ідея Лі [3], або як його іще називають хвильовий алгоритм. Існують й інші підходи, для вирішення даної проблеми, але вони мають гірший час виконання. Це пояснюється тим, що алгоритми, які обчислюють найкоротший шлях спочатку конструюючи увесь граф видимості, приречені щонайменше на квадратичний час роботи в найгіршому випадку, оскільки граф видимості може мати квадратичне число ребер. Доволі довго такий час вважався найкращий. Мітчелл [4] був першим, хто зламав квадратичний бар'єр: він показав, що найкоротший шлях може бути обчислений за час $O(n^{5/3} + \varepsilon)$. Згодом він покращив час роботи свого алгоритму до $O(n^{3/2} + \varepsilon)$ [5]. В той же час вели свої дослідження в цій сфері Хершбергеру і Сурі [6, 7]. Їм вдалося досягти оптимального часу для алгоритму за O(n(log(n))).

Новизна та ідея. В розглядуваній роботі запропоновано підхід, який узагальнює задачу пошуку найкоротшого шляху на множині перешкод у 2D та алгоритм її розв'язання.

Мета статті. Розробити узагальнений метод побудови найкоротшого шляху на множині перешкод у 2D із застосуванням графа видимості для пошуку всіх можливих шляхів між двома заданими точками в евклідовому просторі.

2 Основна частина.

Сформулюємо геометричну постановку задачі пошуку найкоротшого шляху на множині перешкод у 2D.

Постановка задачі пошуку найкоротшого шляху на множині перешкод [8]. Нехай дано h непересічних багатокутників - полігональних перешкод. Нехай існує умовний агент, що переміщається в цій же площині, він представлений точкою. Нехай дана стартова точка S і кінцева точка T. Нам необхідно

побудувати найкоротший маршрут з точки S в точку T, при цьому, будемо вважати що агент може знаходитися на кордоні перешкоди, але не всередині неї.

2.1.Побудова розв'язку задачі пошуку найкоротшого шляху на множині перешкод.

Можна довести, що оптимальний шлях - це кусково-лінійна ламана, вершини якої збігаються з вершинами перешкод.

Доведення

Частина 1. Оптимальний шлях - кусочно-лінійна ламана

Припустимо, що найкоротший шлях T - не є кусочно-лінійної ламаної. В такому випадку, на шляху T існує така точка p, яка не належить жодному прямому відрізку з T (рисунок 1). Це означає що існує ε-окіл точки p (рисунок 2) в яку не потрапляє одна перешкода. В такому випадку, підшлях T ε , який знаходиться всередині ε-околу, може бути скорочений по хорді, що сполучає точки перетину кордону ε-околу з шляхом T. Раз частина шляху може бути зменшена, значить і весь шлях може бути зменшений, а значить вихідне припущення некоректно. Шлях T - кусочно-лінійна ламана.

Рисунок 1. Точка p на шляху T

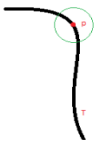

Рисунок 2. ε-окіл точки p

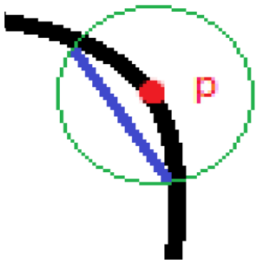

Рисунок 3. Хорда, що є найкоротшим шляхом

Частина 2. Вершини шляху збігаються з вершинами перешкод

Припустимо тепер що вершини шляху T не збігаються з вершинами багатокутників-перешкод. Розглянемо вершину v шляху T. Вершина не може лежати всередині вільного простору, оскільки в іншому випадку, застосувавши аналогічний підхід, ми знайшли б в ε-околі вершини v коротший шлях по хорді (рисунок 4). Значить, вершина v має лежати на кордонах перешкод (в їх вершинах або на ребрах). Але і на ребрах вершина лежати не може, оскільки (знову розглядаємо ε-окіл), ми зможемо зрізати по хорді (рисунок 5).

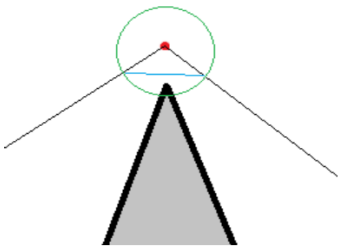

Рисунок 4. ε-окіл вершини шляху, що лежить у вільному просторі

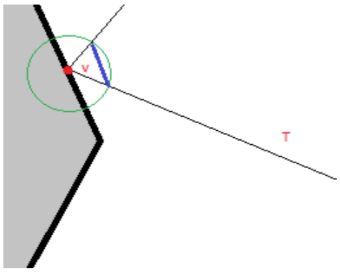

Рисунок 5. ε-окіл вершини шляху, що лежить на ребрі

Вищесказане означає, що вершини кусочно-лінійного шляху T зобов'язані перебувати в вершинах перешкод-багатокутників.

Постановка задачі побудови графа видимості. Будемо називати дві точки взаємовидимими, якщо відрізок, їх з'єднує, не містить в собі внутрішніх точок перешкод-багатокутників.

Для побудови найкоротшого шляху використовується граф видимості. Вершинами графа є вершини перешкод-багатокутників. Ребрами з'єднані тільки взаємовидимі вершини графа.

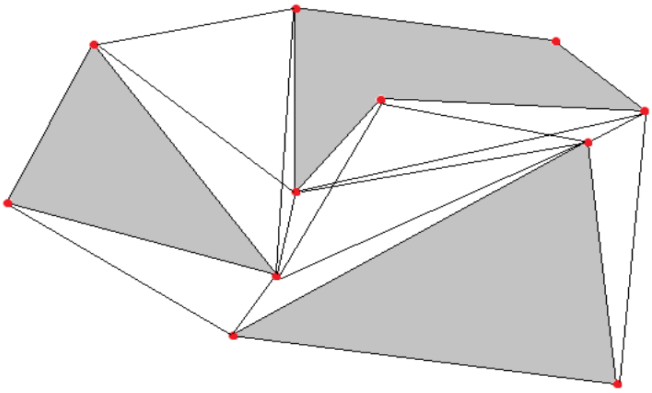

Рисунок 6. Приклад графа видимості.

Припустимо, що ми побудували граф видимості VG. Додамо до графу VG вершини S і T, а також ребра, що з'єднують їх з усіма видимими з них вершинами. Отримаємо розширений граф VG + . Нам залишиться лише знайти найкоротший шлях з вершини S у вершину T в графі VG + . Це можна зробити, застосувавши алгоритм Дейкстри [2]. Залишається лише вирішити задачу побудови графа видимості.

2.2 .Побудова розв'язку задачі побудови графа видимості.

Для кожної вершини знайдемо всі видимі з неї вершини за допомогою методу плоского замітання. Нам потрібно вирішити наступне завдання: на площині дано безліч відрізків (ребер перешкод) і точка p. Знайти усі кінці відрізків, видимі з точки p. Будемо замітати площину обертовим променем з початком в точці p (рисунок 7). Статусом замітаючою прямої будуть відрізки, які її перетинають, впорядковані по зростанню відстані від точки p до точки перетину. Точками подій будуть кінці відрізків.

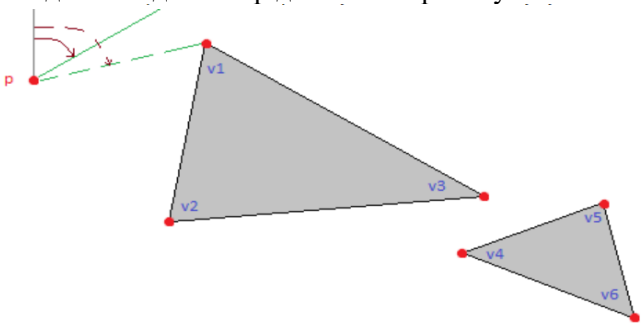

Рисунок 7. Замітання площини обертовим променем

Впорядковані за кутом щодо осі, що проходить через точку p, вершини будуть зберігатися в черзі подій. Ініціалізація статусу займе час O(nlogn), ініціалізація черги подій також займе час O(nlogn). Загальний час роботи алгоритму замітання є O(nlogn). Таким чином, застосувавши алгоритм для кожної вершини графа, ми отримаємо граф видимості за час порядку $O(n^2 \log n)$.

Далі отримаємо список видимих вершин. Перш за все вершини $w \in W$ упорядковуються за кутом між променем vw і вертикальною напіввіссю, що проходить через v. Таке сортування виконується за O(nlogn), наприклад, сортуванням злиттям. Потім відбувається ініціалізація множини видимих вершин (за замовчуванням, така множина порожня). Далі починається замітання площині. В порядку сортування вершин для кожної з них виконується перевірка: чи видно вершину w з вершини v. Так як така перевірка означає наявність перетинів, які зберігаються в збалансованому дереві, вона може бути виконана за O(logn). Якщо вершина видима, необхідно додати її в список видимих вершин. І, нарешті, незалежно від видимості вершини, необхідно змінити статус замітаючої прямої. Для цього для поточної вершини w необхідно видалити зі списку поточних перетинів всі ребра (відрізки), які закінчуються в цій вершині (лежать зліва від прямої vw) і додати всі ребра (відрізки), які в ній починаються (лежать праворуч від прямої vw).

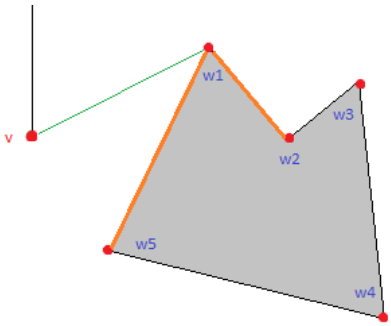

Рисунок 8. Крок 1 – перше оновлення статусу замітаючої прямої.

Отже, на першому етапі (промінь vw1) в дерево додаються два відрізки w1w5 і w1w2, оскільки вони лежать праворуч від променя vw1. Статус замітаючої прямої позначений помаранчевим кольором.

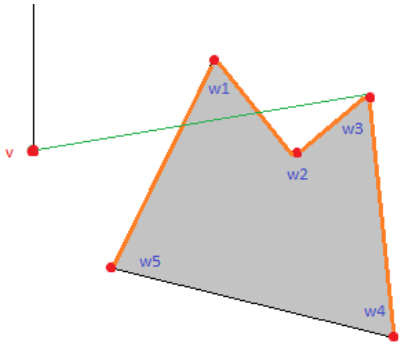

Рисунок 9. Крок 2 – проміжне оновлення статусу замітаючої прямої.

На другому кроці (друга в списку вершина – це вершина w3) додаються ще два ребра в статус: w3w2 і w3w4. Тепер в статусі лежать 4 ребра.

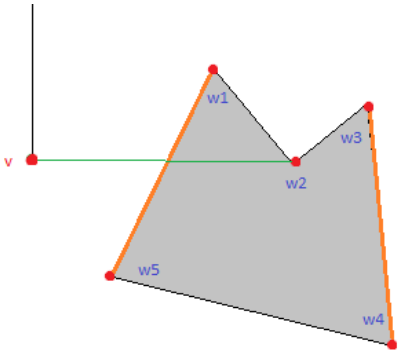

Рисунок 10. Крок 3 – останнє оновлення статусу замітаючої прямої.

Нарешті, на третьому етапі (вершина w2) зі статусу прибираються 2 ребра: w3w2 і w1w2.

Алгоритм.

Алгоритм SHORTEST_PATH (h, S, T) будує найкоротший шлях на множині перешкод.

Вхідні дані. Набір P полігональних перешкод, що не перетинаються, і дві точки S і T у вільному просторі.

Вихідні дані. Найкоротший шлях, що з'єднує S і T.

1. Graph ← VISIBILITY_GRAPH(P ∪ {S, T}).
2. Призначити кожній дузі (v, w) у Graph вагу, яка дорівнює евклідовій довжині відрізка vw.
3. Використати алгоритм Дейкстри для обчислення найкоротшого шляху між S і T$_1$ в Graph.

Алгоритм VISIBILITY_GRAPH(h) будує граф видимості.

Вхідні дані. Набір h полігональних перешкод, які не перетинаються.

Вихідні дані. Граф видимості Graph (h).

1. Ініціалізуємо граф G = (V, E) де V – множина всіх вершин багатокутників в h і E = Ø.
2. Для всіх вершин v ∈ V do W ← VISIBLE_VERTICES(v, h).
3. Для кожної вершини w ∈ W додати дугу (v, w) в E.
4. return G.

Алгоритм VISIBLE_VERTICES(p, h) знаходить взаємовидимі вершини.

Вхідні дані. Набір P полігональних перешкод і точка p, яка не лежить всередині жодної перешкоди.

Вихідні дані. Набір всіх вершин перешкод, які видно з p.

1. Сортування вершин перешкод відповідно до кута за годинниковою стрілкою. Вершини, ближчі до p, повинні йти перед вершинами, віддаленими від p. Нехай w1, ..., wn є відсортованим списком.
2. Нехай ρ – промінь, паралельний додатній осі x, який починається з p. Знайти ребра перешкод, які перетинаються цим променем, і зберегти їх у збалансованому дереві пошуку Tree в тому порядку, в якому вони перетинаються з ρ.
3. W ← Ø.
4. for i ← 1 to n do if VISIBLE(w_i) then додати w_i до W.
5. Вставити в дерево Tree ребра перешкод, інцидентних до w_i, які лежать за годинниковою стрілкою на промені від p до w_i.
6. Видалити з Tree ребра перешкод, інцидентних до w_i, що лежать проти годинникової стрілки на промені від p до w_i.
7. return W.

Алгоритм VISIBLE (w_i) визначає видимість вершини.

Вхідні дані. Вершина з відсортованого списку.

Вихідні дані. Статус видимості даної вершини.

1. if pw_i перетинає внутрішню частину перешкоди, для якої w_i є вершиною then return false.
2. else if i = 1або w_{i-1} не належить сегменту pw_i then шукати в Tree ребро e на крайньому лівому листі.
3. if e існує, а pw_i перетинає e then return false.
4. else then return true.
5. if w_{i-1} не видно then return false.
6. else пошук у Tree для ребра e, що перетинає $w_{i-1}w_i$.
7. if e існує then return false.
8. else then return true.

3 Обґрунтування складності

Теорема 1. Найкоротший шлях між двома точками серед множини полігональних перешкод з n ребрами може бути обчислений за $O(n^2 \log n)$ часу. [9]

Доведення. Визначення графа видимості потребує $O(n^2 \log n)$ часу, де n – загальна кількість ребер перешкод. Кількість дуг Graph обмежена C^{n+2}_2. Отже, присвоєння ваг дугам графа потребує $O(n^2)$ часу. Алгоритм Дейкстри обчислює найкоротший шлях між двома вузлами в графі з k дугами, кожна з яких має невід'ємну вагу за $O(n \log n + k)$ часу. Оскільки $k = O(n^2)$, ми робимо висновок, що загальний час роботи SHORTEST_PATH дорівнює $O(n^2 \log n)$, а час запиту складає $O(\log n)$.

4 Практична частина

Задача пошуку найкоротшого шляху на множині перешкод у 2D реалізована у інтегрованому середовищі розробки Qt Creator. Запуск програми надає її користувачу можливість скористатися інтерфейсом графічного вікна.

В будь-який момент часу у користувача є можливість:

1. Задати початкову точку шуканого найкоротшого шляху (Встановити початкову точку).
2. Задати кінцеву точку шуканого найкоротшого шляху (Встановити кінцеву точку).
3. Додати на полотно новий многокутник-перешкоду (Створити перешкоду).

Додати новий многокутник-перешкоду користувач може так (див. рисунок 13):

1. Натиснути на кнопку Створити перешкоду.
2. Додати початкову вершину многокутника на полотно.
3. Додавати нові вершини многокутника за годинниковою стрілкою відносно початкової точки.
4. Натиснути кнопку Завершити створення перешкоди.

Також є можливість відмовитись від побудови даного многокутника, натиснувши на кнопку Скасувати.

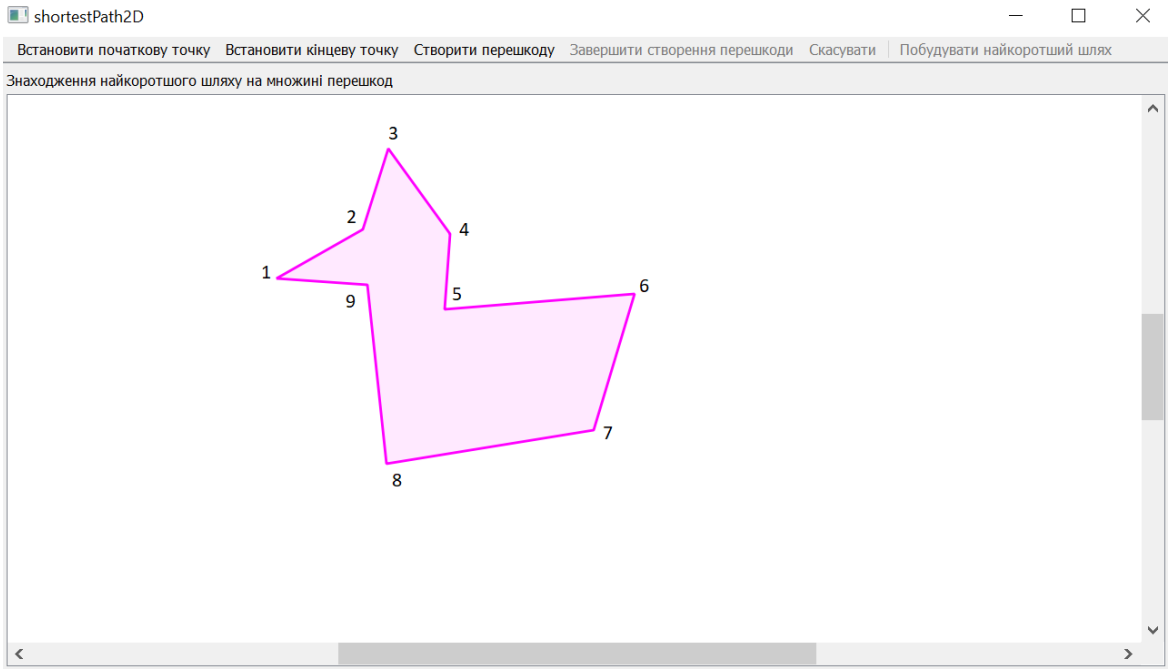

Рисунок 13. Порядок побудови многокутника.

При наявності кінців проектованого шляху на полотні користувач має можливість запустити алгоритм його пошуку кнопкою Побудувати найкоротший шлях (див. рисунок 14).

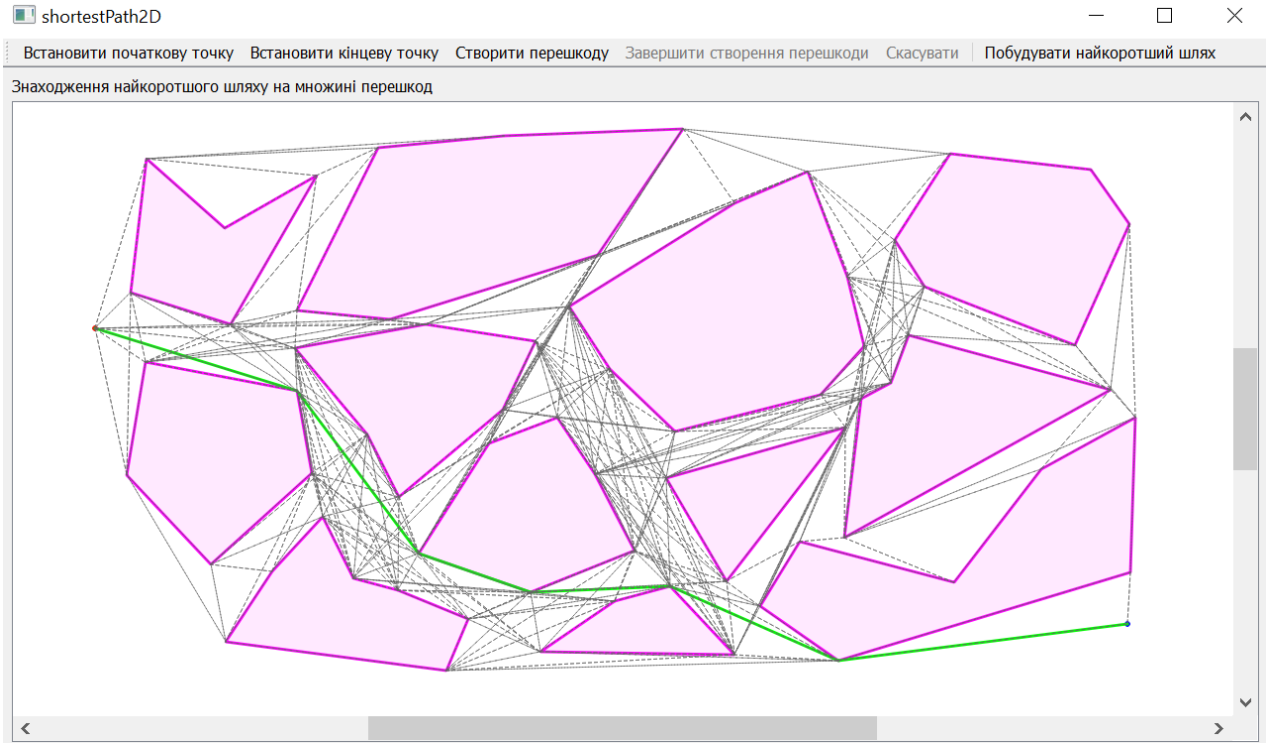

Рисунок 14. Результати роботи програми.

5 Висновки

У роботі реалізовано та описано алгоритм, що дозволяє на заданій множині з h перешкод, сумарна кількість вершин яких рівна n, для кожного розташування запитних точок S і T знайти найкоротший шлях між ними. Для реалізації цього алгоритму було складено геометричну постановку задачі пошуку найкоротшого шляху на множині перешкод у 2D, а також було запропоновано ефективний розв'язок даної проблеми, що включає в себе побудову графу видимості та виконання алгоритму Дейкстри для пошуку найкоротшого шляху у графі між двома вершинами. В процесі виконання лабораторної роботи писався звіт з покроковим оглядів ключових моментів алгоритму. Історія досліджень теми пошуку найкоротшого шляху між двома точками на множині перешкод у евклідовому просторі показує, що цей алгоритм, вперше запропонований Лі, є одним з найбільш надійних та простих у розумінні алгоритмів, що дозволяють розв'язати дану проблему.

Складність такого алгоритму становить $O(n^2 \log n)$, а час запиту $O(\log n)$. Для практичного застосування реалізованого алгоритму було налагоджено зручний україномовний інтерфейс користувача з усіма необхідними для його роботи функціями. Розумінню інтерфейсу сприяють надані знімки екрану.

Список літератури

1. https://en.wikipedia.org/wiki/Visibility_graph
2. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
3. D. T. Lee. Proximity and reachability in the plane. Report R-831, Dept. Elect. Engrg., Univ. Illinois,
4. Urbana, IL, 1978 J. S. B. Mitchell. Shortest paths among obstacles in the plane. In Proc. [[th Annu. ACM Sympos. Comput. Geom., pages 308–317, 1993.
5. J. S. B. Mitchell. Shortest paths among obstacles in the plane. Internat. J. Comput. Geom. Appl., 6:309–332, 1996.
6. J. Hershberger and S. Suri. Efficient computation of Euclidean shortest paths in the plane. In Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci., pages 508–517, 1993.
7. J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. SIAM J. Comput., 28:2215–2256, 1999.
8. https://habr.com/ru/post/199256/
9. Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars. Computational Geometry. Algorithms and Applications. Springer; 3rd (Third) Edition edition (2010).