|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Расчетно-графическое задание | | |
| по дисциплине « Методы построения и анализа алгоритмов » | | |
|  | | |
|  | | |
|  | Факультет | ПМИ |
|  | Группа | ПМИ-11 |
| Студент | БаРКОВСКИЙ ИЛЬЯ Баллы: |
| Вариант | 4 Подпись: |
|  | Дата: |
|  |  |
| Преподаватель | Щукин Георгий Анатольевич |
|  |  |
| Новосибирск  2022 | | |

**Задача**

Мобильный робот движется по плоскости. Робот имеет форму круга радиусом r с координатой центра (Cx, Cy); робот может двигаться в любом направлении без ограничений, с некоторой заданной постоянной скоростью. Также на плоскости находятся объекты-препятствия. Заданы начальная и конечная координаты робота Init=(Initx, Inity) и Goal=(Goalx, Goaly), дополнительно может быть задано несколько промежуточных координат.

Задача: найти кратчайший (по возможности) путь для робота из начальной координаты в конечную, не приводящий к столкновению робота с препятствиями. При наличии дополнительных координат путь должен проходить через каждую дополнительную координату минимум по одному разу.

Вариант: RRT, прямоугольники, без промежуточных координат

**Использованные алгоритмы:**

Graph - класс граф (<https://github.com/georgy-schukin/mpiaa-py/blob/master/mpiaa/graphs/graph.py>, был доработан под текущую задачу)

Функция Nearest - возвращает вершину (конфигурацию) из графа G, ближайшую к конфигурации X. Если оказывается, что X близка не к одной вершине, а к ребру (т.е. что ближайшая к X конфигурация лежит на ребре графа), то в этом месте ребро делится на две части и в граф добавляется новая вершина, которая затем возвращается.

Функция Steer - возвращает новую конфигурацию Z из Cfree, лежащую на отрезке XY и ближайшую к Y, такую что CollisionFree(X, Z). Отрезок был исследован с фиксированным шагом, также имеется реализация без шага.

Функция RRT

Алгоритм Коэна-Сазерланда (

Отдельное описание и тесты: <https://github.com/eternalowo/Cohen-Sutherland-Algorithm>  
<https://ru.wikipedia.org/wiki/Алгоритм_Коэна_—_Сазерленда>

<https://en.wikipedia.org/wiki/Cohen–Sutherland_algorithm>

)

Алгоритм Дейкстры (

Адаптирован под текущий класс граф и задачу: <https://waksoft.susu.ru/2021/10/21/realizacziya-algoritma-dejkstry-na-python/>

<https://ru.wikipedia.org/wiki/Алгоритм_Дейкстры>

)

Функция RandomSample - возвращает новую случайную конфигурацию из Cfree.

**Разработанные алгоритмы:**

Класс Rectangle

Функция p

Функция scalar\_mult

Функция get\_vector

Функция get\_distance

Функция get\_height\_point

Функция get\_shortest\_distance

Все описания функций представлены в виде комментариев к коду

**Возможности интерфейса:**

Кнопка Add QInit позволяет добавить координаты робота (ниже – поле для ввода координат).

Кнопка Add QGoal позволяет добавить координаты конечной точки (ниже– поле для ввода координат).

Кнопка Set Iterations позволяет задать кол-во итераций (ниже– поле для ввода кол-ва итераций).

Кнопка Add Obstacle позволяет добавить препятствие (ниже– поле для ввода координат нижней и верхней точки диагонали прямоугольника).

Кнопка Remove Obstacles позволяет удалить все препятствия со сцены.

Кнопка Start Algorithm запускает выполнение алгоритма RRT.

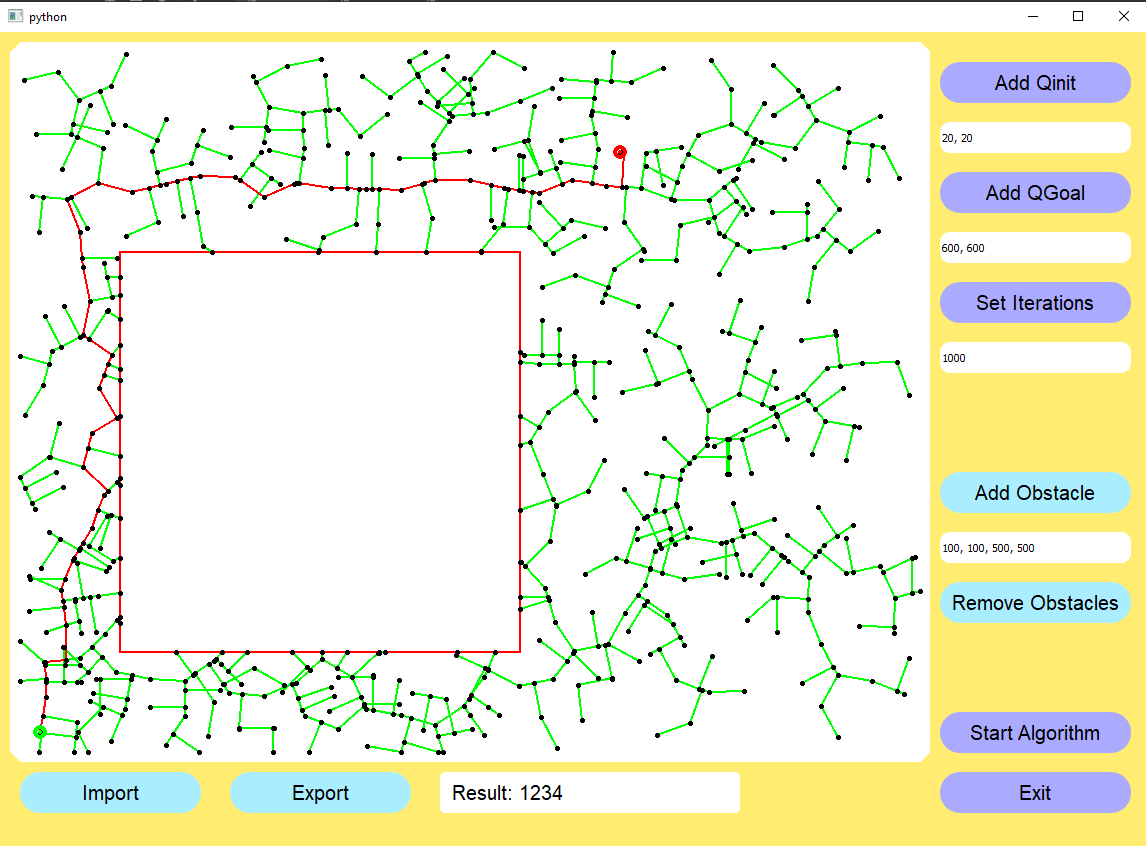
Кнопка Exit выполняет выход из программы.

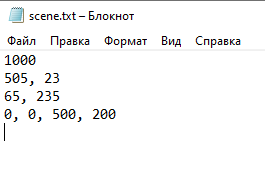
Кнопка Import позволяет выгрузить сцену из txt файла.

Кнопка Export позволяет выгрузить сцену в txt файл (Перед экспортом сцены следует очистить все препятствия, если до этого была другая сцена).

Перед тем как добавлять объекты на сцену – сначала выполняется ввод данных в соответствующие поля.

Пример корректного ввода во все поля и структура txt файла для импорта:





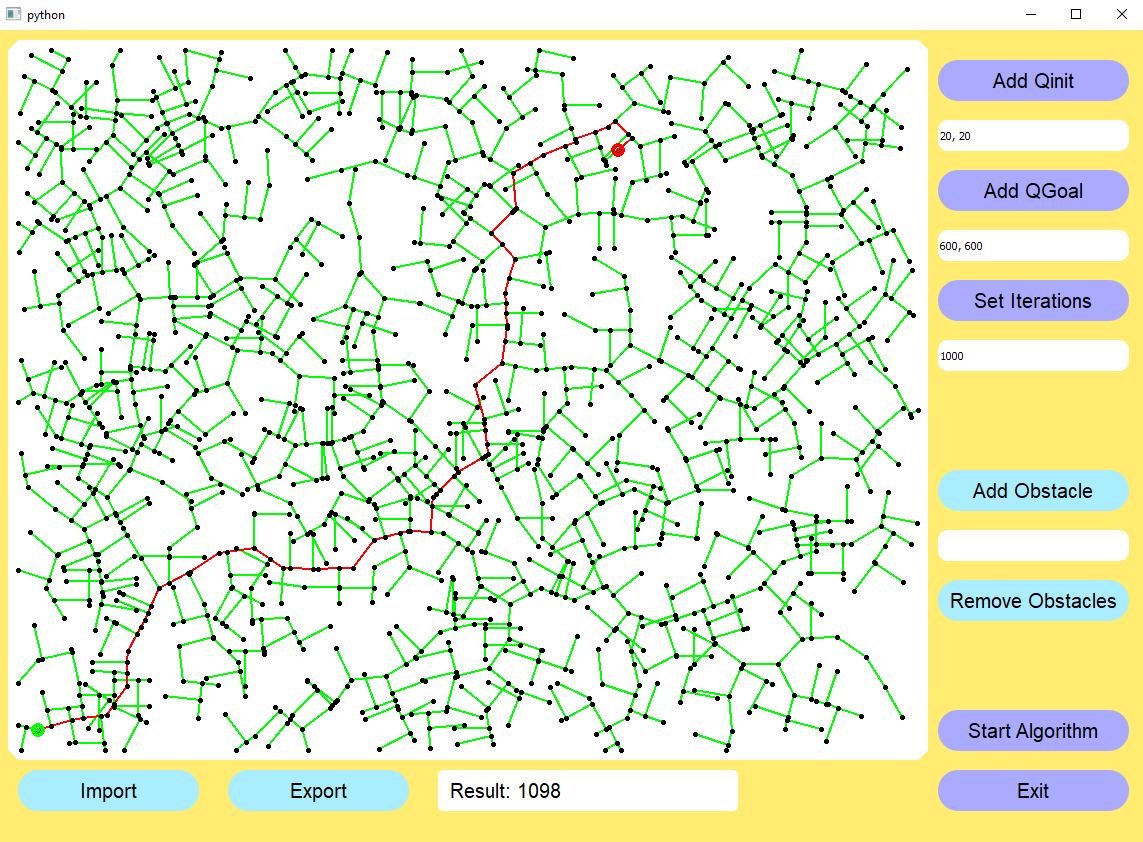
**Тесты (1000 итераций):**

Все сцены и код программы доступны по ссылке: <https://github.com/eternalowo/Randomly-Exploring-Random-Trees>

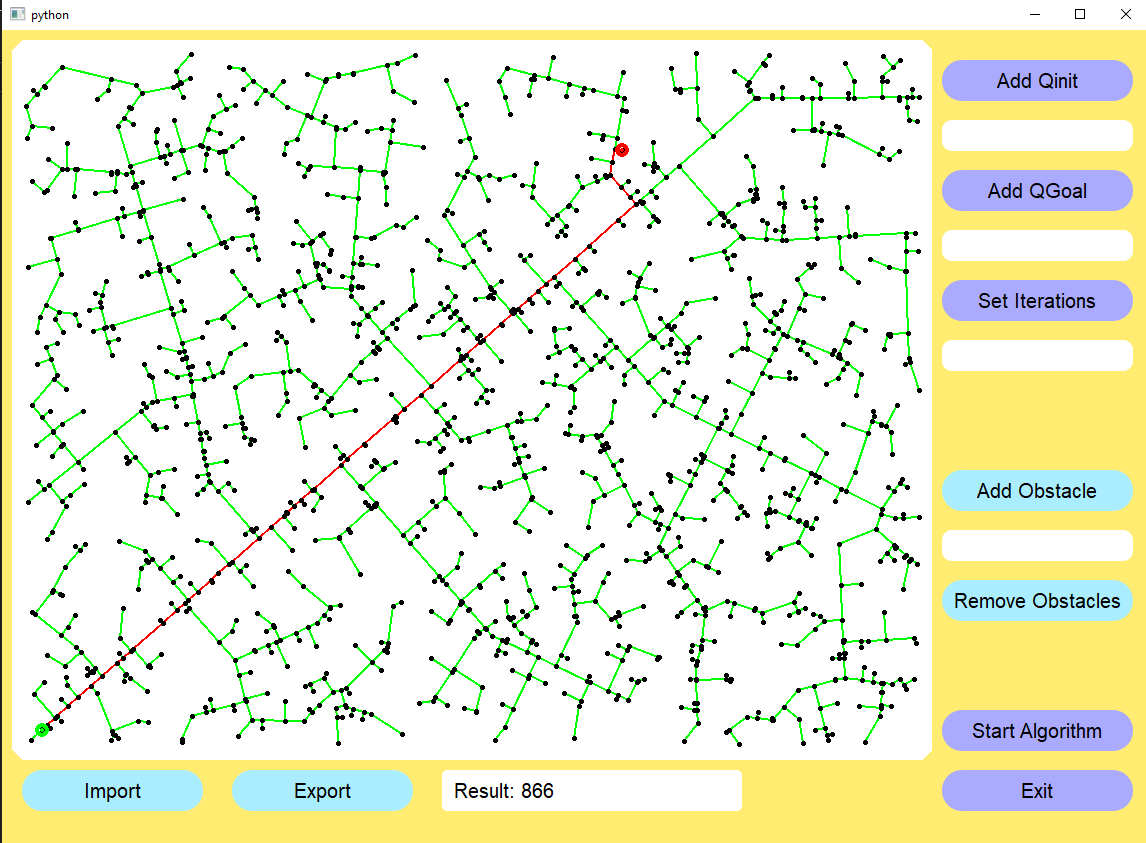
Разные реализации алгоритма лежат в разных ветках репозитория

1. Нет препятствий:

С фиксированным шагом:

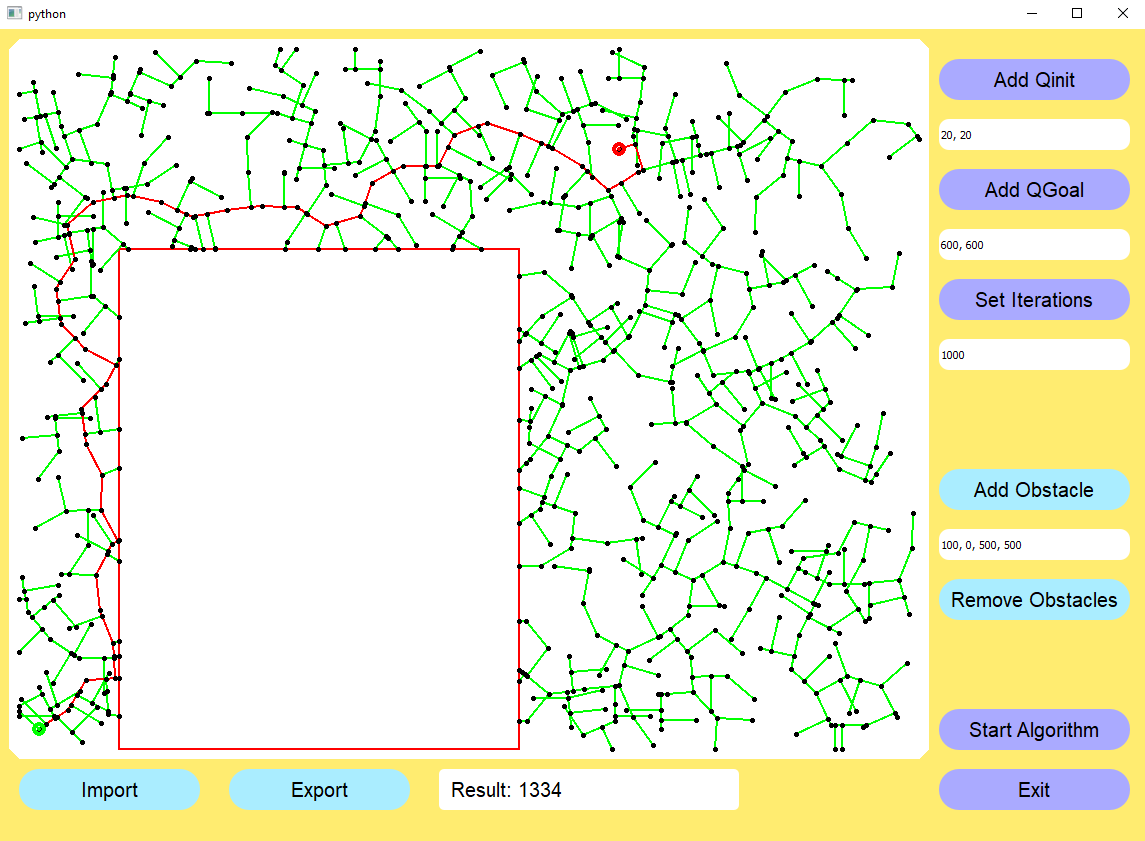


Без фиксированного шага:

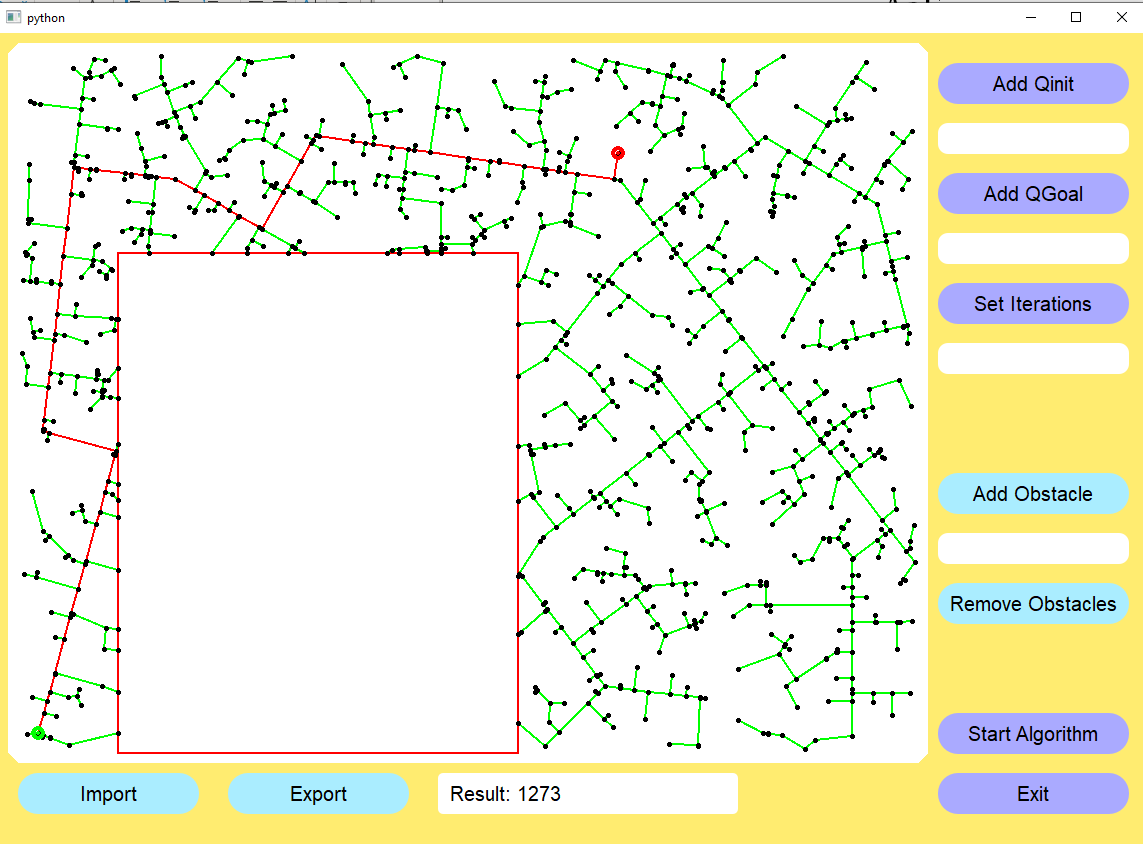


2. С препятствием:

С фиксированным шагом:

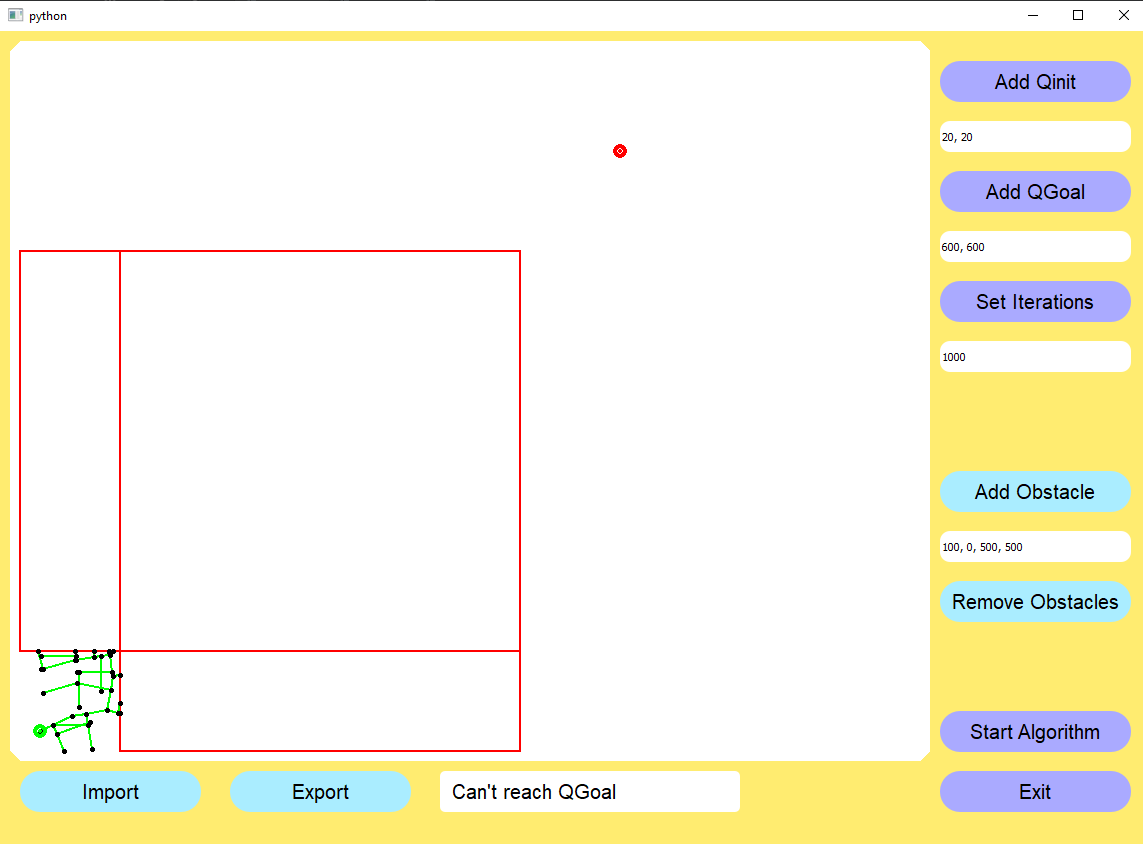


Без фиксированного шага:

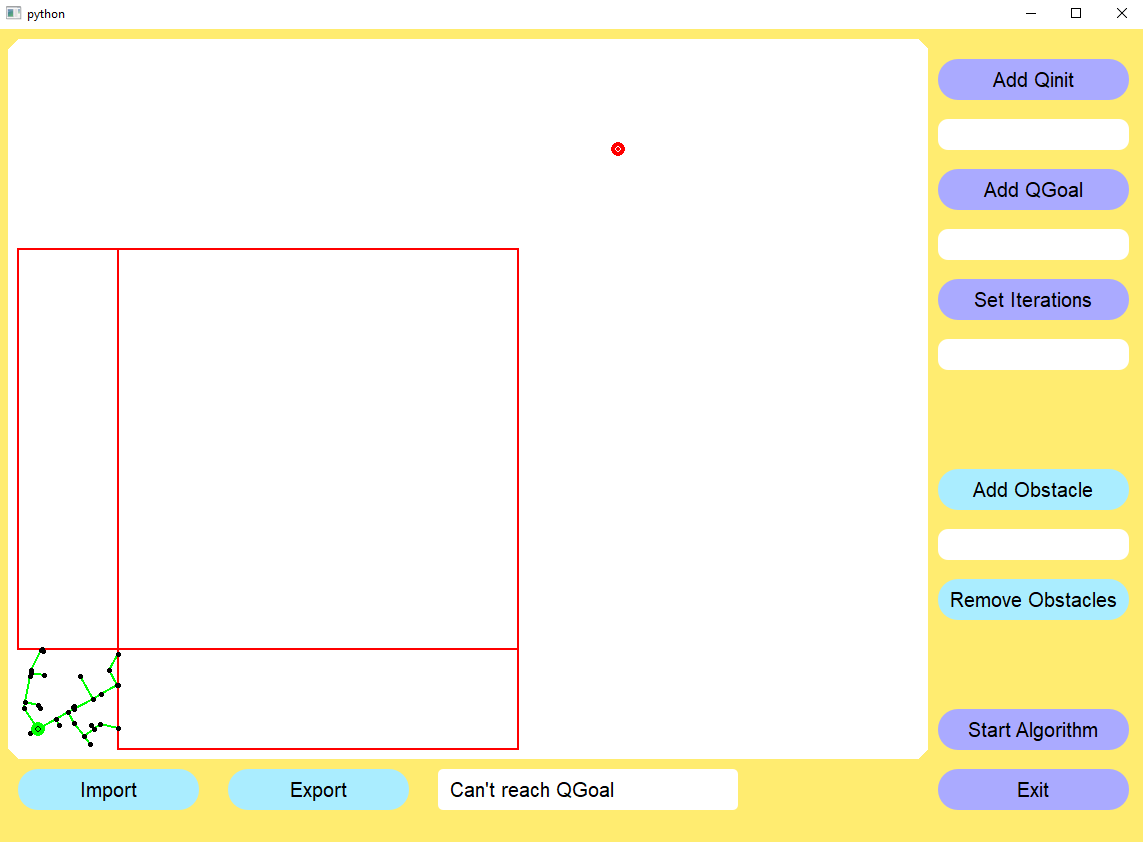


1. Недоступная точка:

С фиксированным шагом:

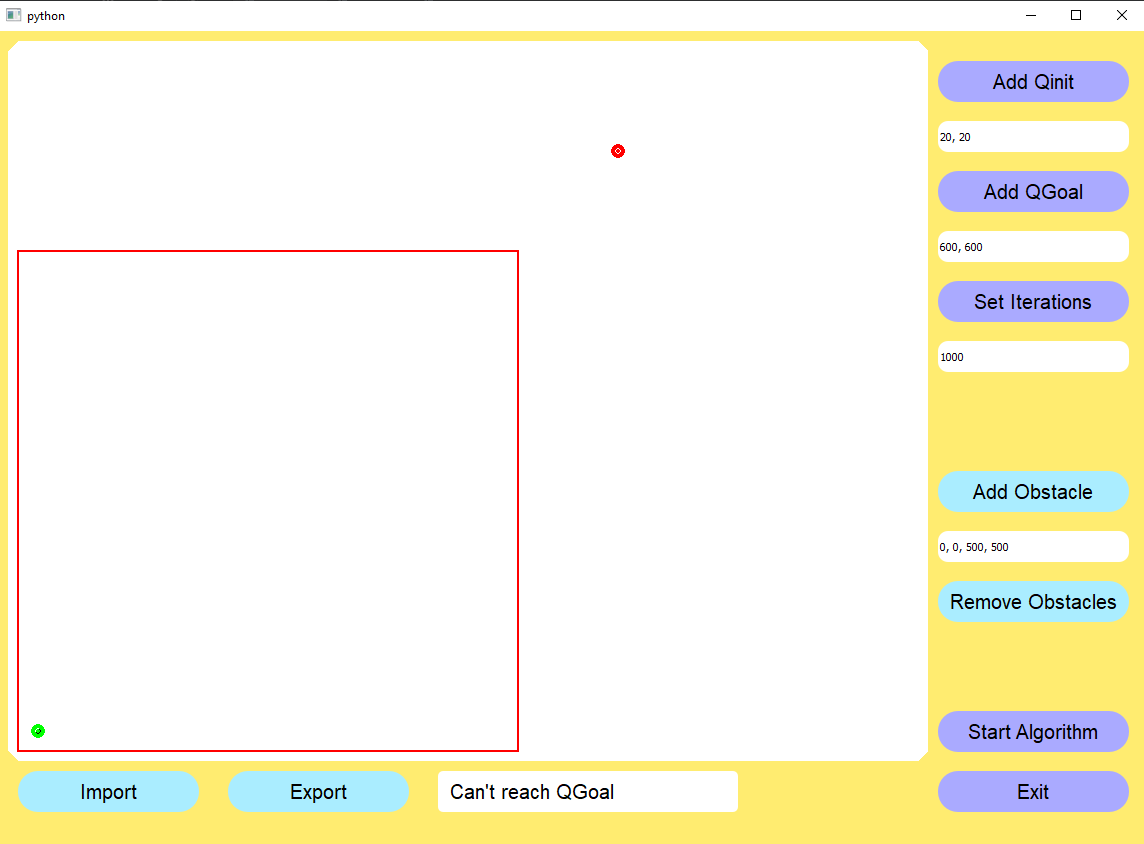


Без фиксированного шага:

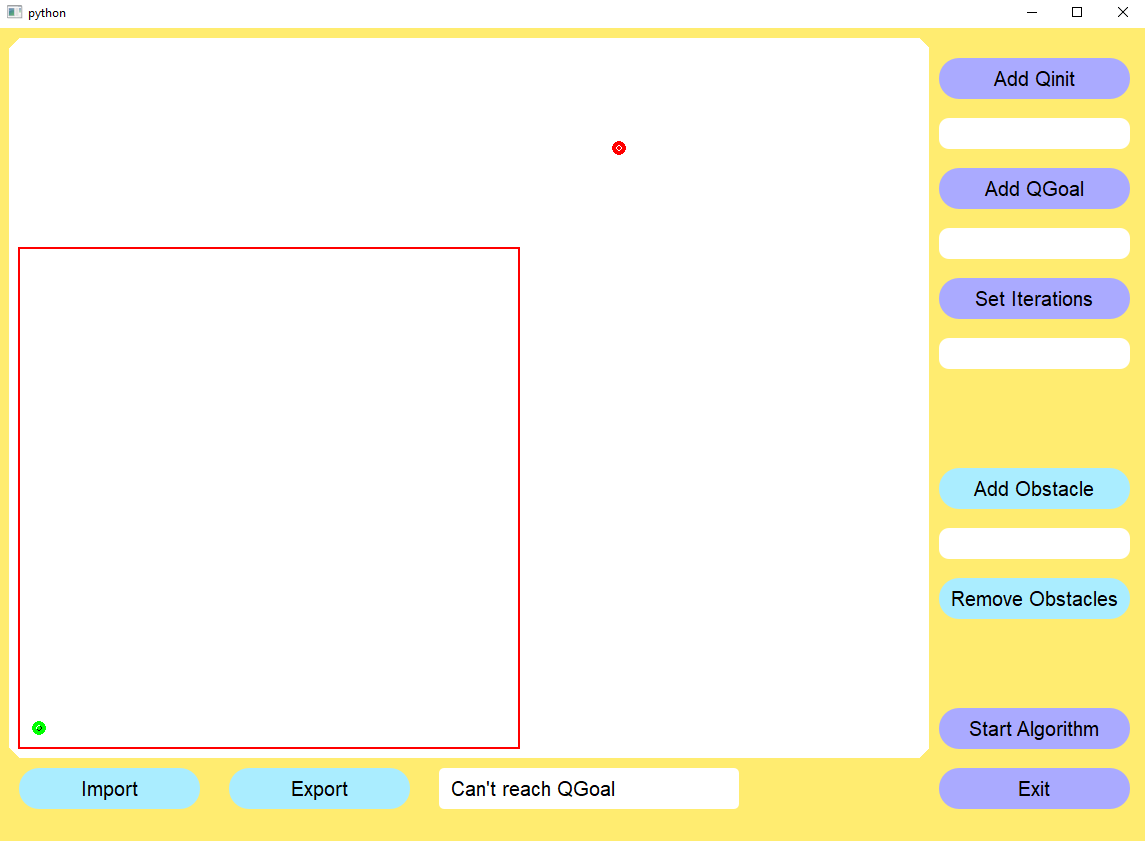


1. Недоступная начальная точка:

С фиксированным шагом:



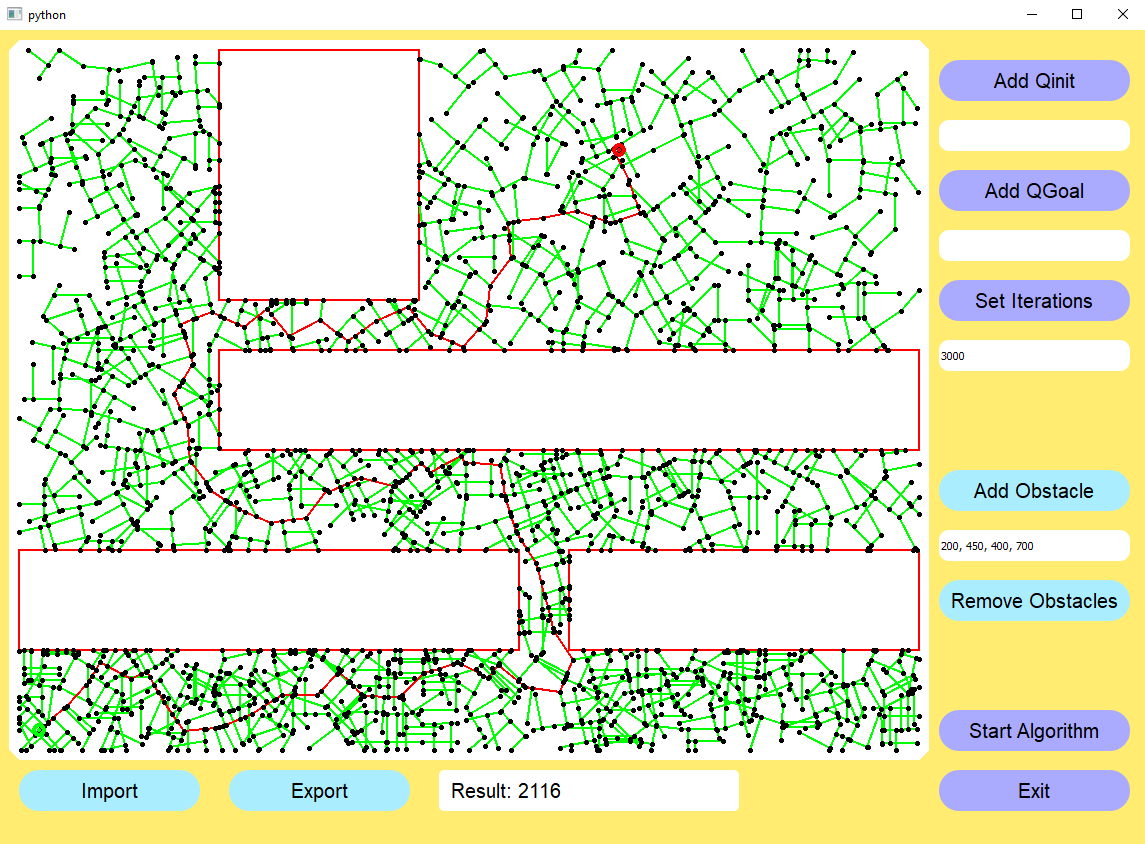
Без фиксированного шага:



1. Сложная конфигурация препятствий:

(1000 итераций зачастую мало, рекомендуется брать больше, например – 3000)

С фиксированным шагом:



Без фиксированного шага:

