Постановка задачи

Необходимо разработать программу, реализующую алгоритм Glavina, который используется для поиска пути из начальной точки в конечную на пространстве с препятствиями. Для проверки реализации необходимо использовать генератор, выполняющий генерацию json-файла с описанием препятствий.

Описание метода

Алгоритм Glavina впервые был описан Бернардом Главина в статье «Solving Findpath by Combination of Goal-Directed and Randomized Search» [1]. Основная идея алгоритма заключается в дополнении метода GDS (Goal-Directed Search) созданием промежуточных подцелей.

Метод GDS работает следующим образом:

1. строится прямая p из текущей точки в конечную;
2. выполняется попытка перемещения по прямой p с заданным шагом s;
3. если новое положение находится в свободном пространстве, то перемещение выполняется;
4. если новое положение находится на препятствии, то выполняется сдвиг по прямой p в обратном направлении до момента пересечения с границей препятствия;
5. выполняется попытка перемещения в точку, лежащую на ортогональной к p прямой на расстоянии s или $-s$ от точки пересечения с границей препятствия;
6. переходим к пункту 1;
7. метод останавливает работу, когда окажется в конечной точке, и застревает, когда шаги 4-5 невозможно выполнить.

На рисунке ниже приведена иллюстрация работы метода GDS:

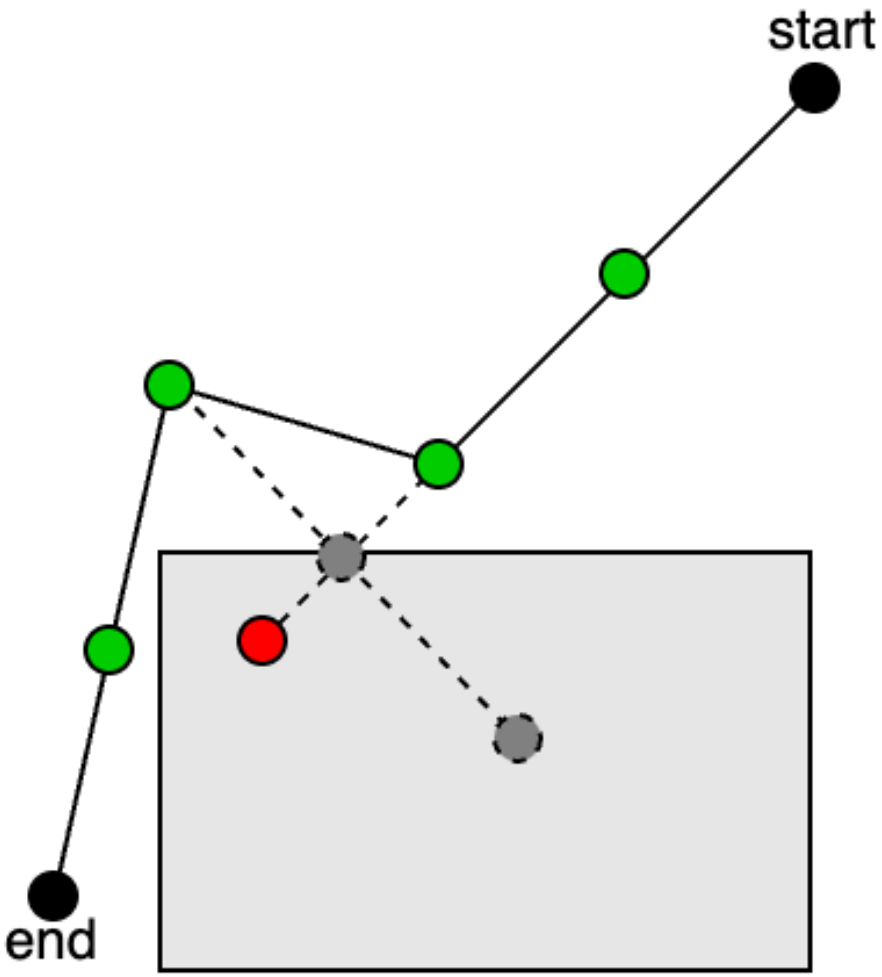

Рисунок 1 – Работа GDS

Алгоритм Glavina работает следующим образом:

1. выполняется попытка поиска пути из начальной точки в конечную при помощи метода GDS;
2. если путь найден, алгоритм завершает работу;
3. если GDS застревает, то точка остановки добавляется в список подцелей, добавляется на граф и соединяется на графе ребром с начальной точкой, а также сохраняется путь из этой точки в начальную;
4. на свободном пространстве генерируется случайная точка r, которая считается подцелью;

5. выполняется попытка найти путь из r при помощи GDS в начальную точку, в конечную точку, а также во все существующие подцели;
6. в случае нахождения пути в начальную точку, в конечную точку, или в существующую подцель r добавляется на граф, а также соединяется ребром с точкой, в которую удалось найти путь.
7. в случае застревания GDS при выполнении пунктов 5-6 точка остановки становится подцелью, заносится в список существующих подцелей, сохраняется путь из этой точки в r, а также точка добавляется в граф и соединяется ребром с r;
8. выполнение алгоритма происходит до тех пор, пока на графе не будет существовать пути из начальной точки в конечную, или пока не будет достигнуто максимальное количество подцелей.

Также стоит отметить, что для метода GDS был введен дополнительный критерий остановки после определенного количества итераций, так как было выяснено, что метод может зацикливаться на определенных участках при попытке обойти препятствие.

Пример зацикливания на одному участке приведен на рисунке ниже. На этом примере выполняется попытка добраться в точку end. После невозможности продолжения движения по прямой из точки 2, будет выполнена попытка обойти препятствие. В точках 2 и 3 перемещение влево невозможно, поэтому выполняется перемещение вправо. В точке 3 уже можно выполнить перемещение влево, поэтому оно выполняется. В результате этого действия происходит переход в точку 5, прямолинейный путь из которой в конечную вершину совпадает с путем из точки 3. Следовательно, из точки 5 переход будет выполнен снова в точку 4, а из нее – в точку 5 и так до бесконечности.

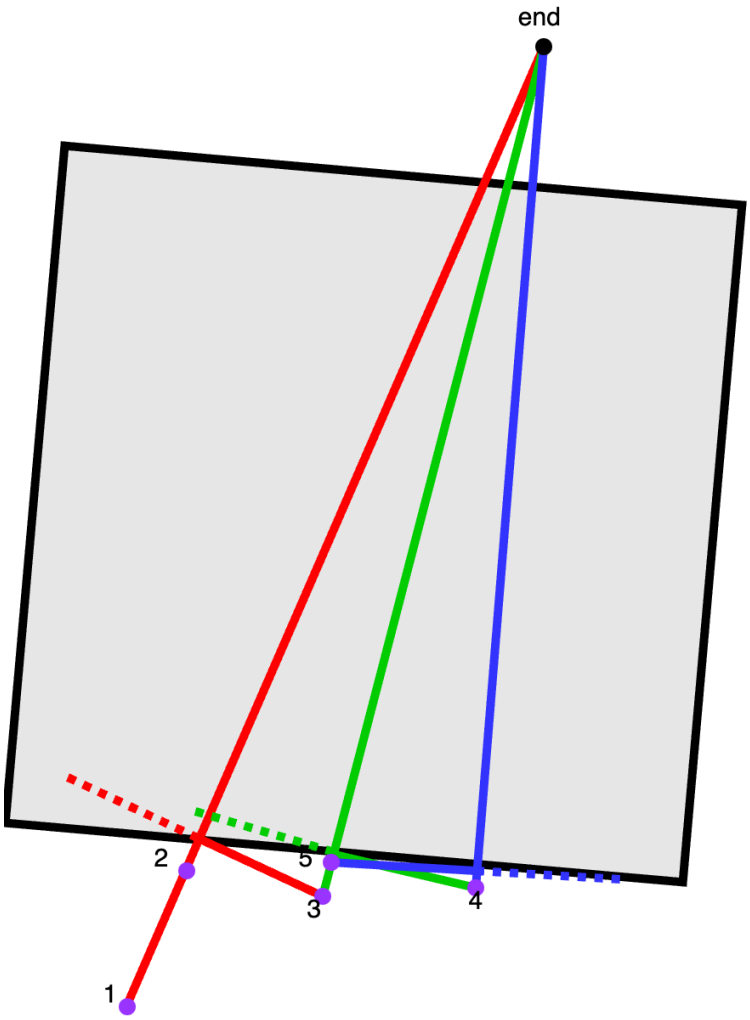

Рисунок 1 – Пример зацикливания алгоритма GDS

Программная реализация

Программа состоит из следующих функций:

1. read_obstacles_data() – выполняет чтение .json файла;
2. create_polygons – выполняет генерацию списка препятствий и объединяет накладывающиеся друг на друга полигоны в единый;
3. gds() – пытается выполнить базовый метод GDS для поиска пути из одной точки в другую, возвращает путь и булевую переменную, обозначающую найден ли путь в исходную точку, или метод застрял;

4. gds_subgoal() – выполняет метод GDS, дополненный генерацией подцелей.

   В программе имеются следующие параметры:

   step – размер шага s;

   max_iterations – количество итераций, которое должно выполниться для остановки метода GDS (проверка на зацикливание на участке);

   max_subgoals – максимальное количесвто подцелей.

Вычислительный эксперимент

Для проверки работа реализованного метода были сгенерированы файлы с описанием пространств с препятствиями. Найденные пути представлены на рисунках 3-5.

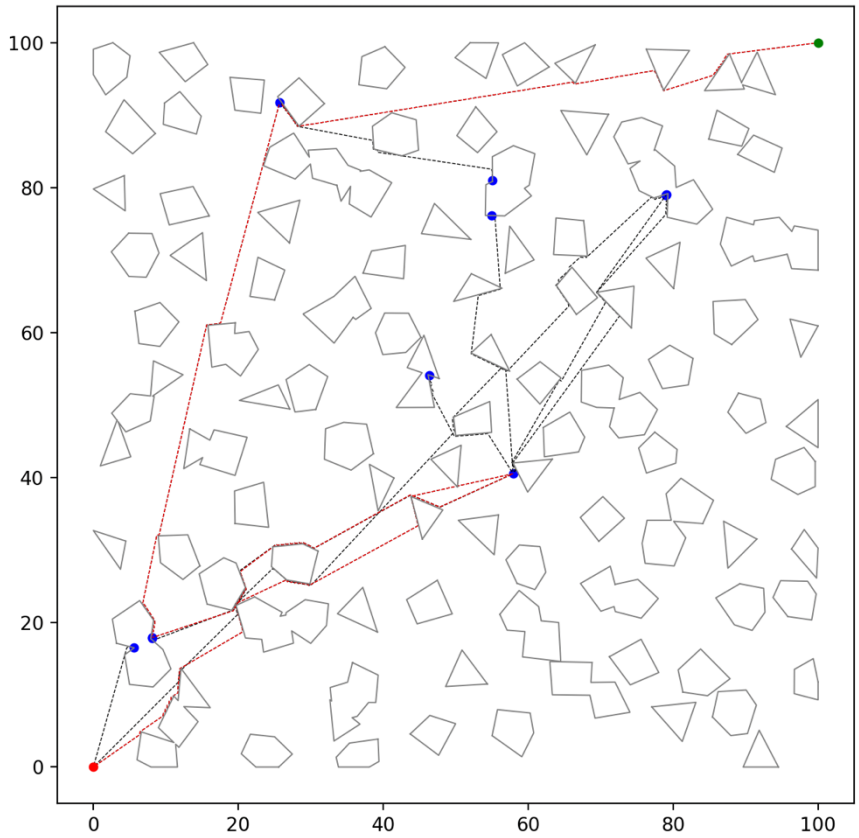

Рисунок 3 – Найденный путь для файла test_1.json

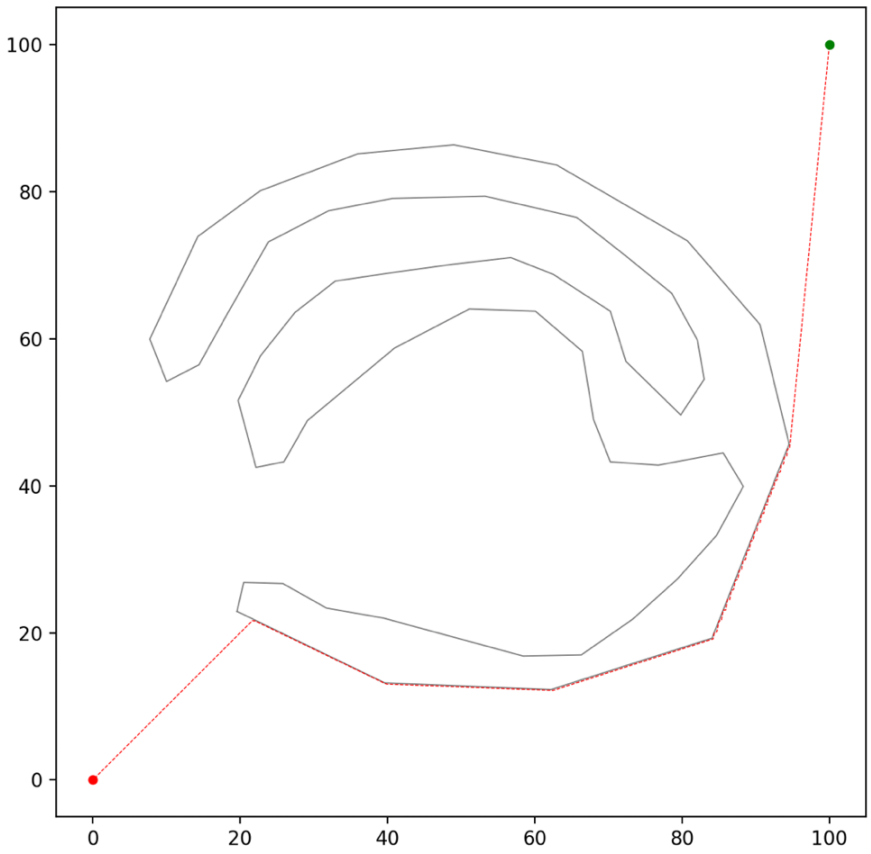

Рисунок 4 – Найденный путь для файла test_2.json

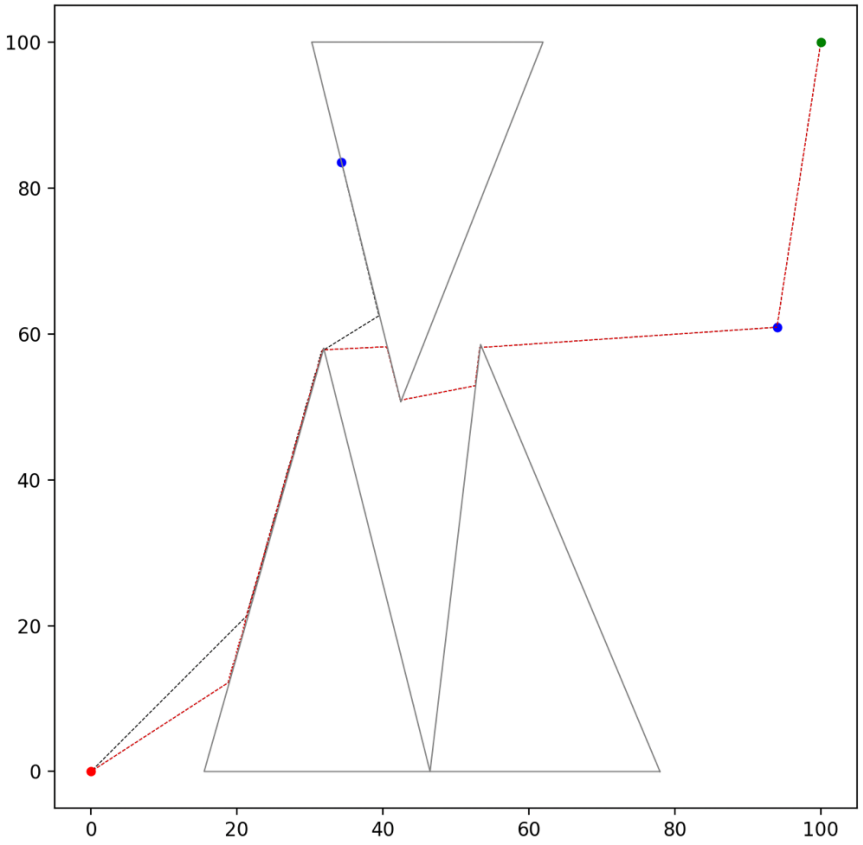

Рисунок 5 – Найденный путь для файла test_3.json

Чтобы выяснить влияние плотности препятствий на алгоритм был выполнен вычислительный эксперимент. Были проведены замеры времени работы и количества удачных нахождений пути. При выполнении эксперимента значение плотности препятствий менялось в диапазоне от 3 до 11, для каждого значения было выполнено 5 запусков программы. Были выставлены следующие параметры алгоритма:

- максимальное количество подцелей: 15;
- шаг: 0,3;
- итераций для завершения GDS: 20000.

Также были выставлены следующие параметры при генерации препятствий:

- размер препятствия: 6;
- максимальное количество вершин: 6.

Результаты представлены в таблице и на рисунке 6.

Таблица 1 – Результаты вычислительного эксперимента

Плотность	**Среднее время поиска пути**	**Процент удачных запусков**
3	0.122	100
4	0.036	100
5	0.078	100
6	0.103	100
7	7.246	100
8	13.207	60

9	8.882	100
10	15.302	20

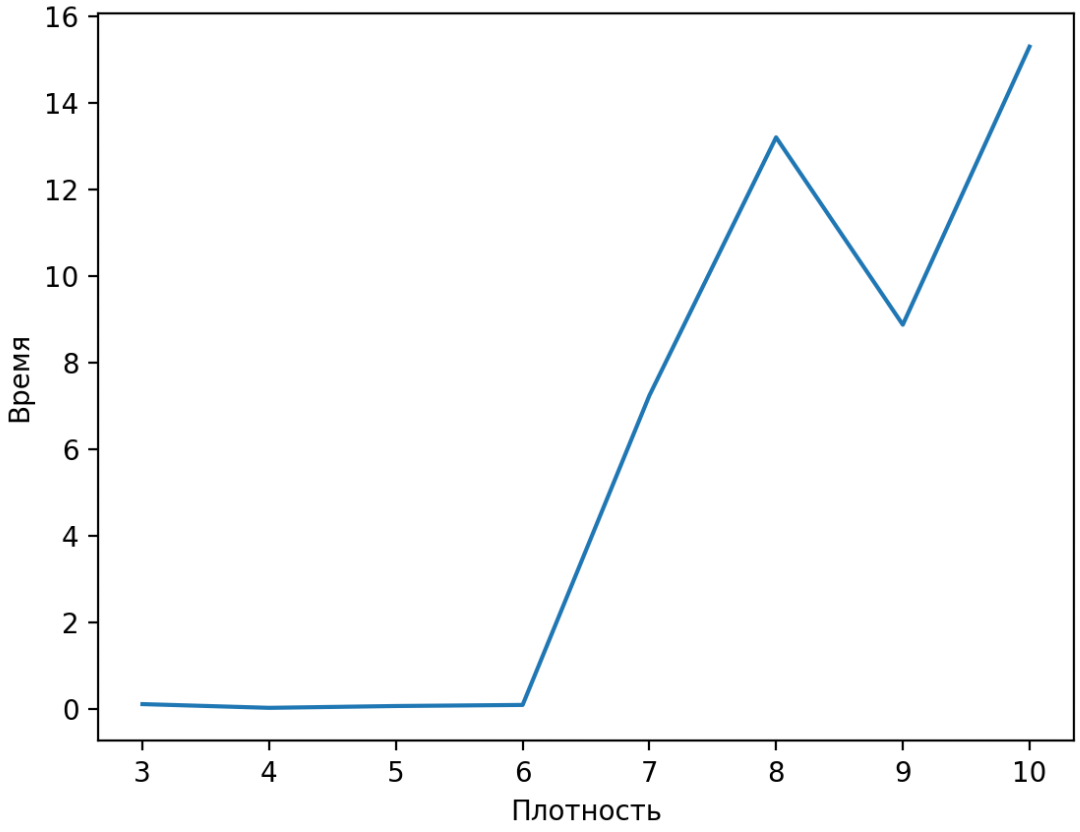

Рисунок 6 – Зависимость среднего времени поиска пути от плотности препятствий

Список использованных источников

1. B. Glavina Solving findpath by combination of goal-directed and randomized search // Proceedings., IEEE International Conference on Robotics and Automation. - Cincinnati, OH, USA: IEEE, 1990. - С. 1718-1723.