

Московский государственный технический университет имени Н.Э.Баумана  
Факультет «Робототехника и комплексная автоматизация»  
Кафедра «Системы автоматизированного проектирования»

**Отчет по лабораторной работе №1**

Курс «Автоматизация технологического проектирования»

Вариант 1

Работу выполнил:  
Воронина Полина (РК6-37(м))

Проверил:  
к.т.н., доцент кафедры САПР  
Божко Аркадий Николаевич

Москва, 2017г

**Оглавление**

[1. Задача «Планирование перемещений» 3](#_Toc501538526)

[2. Описание алгоритма 3](#_Toc501538527)

[2.1. Граф видимости 3](#_Toc501538528)

[2.2. Алгоритм Дейкстры 4](#_Toc501538529)

[3. Реализация алгоритма 6](#_Toc501538530)

[4. Результат работы программы 6](#_Toc501538531)

[5. Список использованной литературы. 7](#_Toc501538532)

[6. Приложение А. Листинги программного кода. 8](#_Toc501538533)

# Задача «Планирование перемещений»

Под планированием движения понимается поиск бесконфликтного пути для перемещения твердого тела или кинематической конструкции в некотором пространстве. Искомый путь представляет собой непрерывную кривую в конфигурационном пространстве объекта, которая соединяет его начальное и конечное положения, исключает столкновения с препятствиями сцены и удовлетворяет всем установленным ограничениям.

Пусть дано множество непересекающихся многоугольников P — полигональных препятствий. Пусть существует агент, перемещающийся в этой же плоскости. Пусть дана стартовая точка s и конечная точка t. Нам необходимо построить кратчайший маршрут из точки s в точку f, при этом, будем считать что агент может находиться на границе препятствия, но не внутри него.

# Описание алгоритма

## Граф видимости

Один из известных методов организации планирования перемещений основан на применении графов видимости (Visibility Graphs). В простейших случаях граф видимости строится на множестве вершин полигонов, являющихся геометрическими моделями препятствий сцены, и дополняется начальными и конечными точками маршрутов. Все вершины попарно соединяются линейными отрезками, которые принимаются в качестве ребер графа при условии попарной “видимости” инцидентных им вершин и отсутствия пересечения с препятствиями сцены (рис. 1). Сложность построения графа видимости для сцены, представленной полигонами с общим числом вершин n, составляет , что является довольно затратным для применения в индустриальных приложениях. Кроме того, преобразование топологического графа в маршрутную сеть также представляет собой сложную задачу, поскольку ребра графа видимости проходят точно через вершины препятствий, и для успешной навигации требуется коррекция путей или предварительное расширение границ препятствий. Избыточное количество порождаемых маршрутов, крайне нежелательное при моделировании сложных сцен, также является недостатком данного метода. Тем не менее, в двумерном окружении с относительно небольшим числом препятствий данный метод может успешно применяться, например, для решения задач поиска кратчайшего пути в сцене. Более того, существуют попытки использования графов видимости для планирования движения в динамическом окружении.

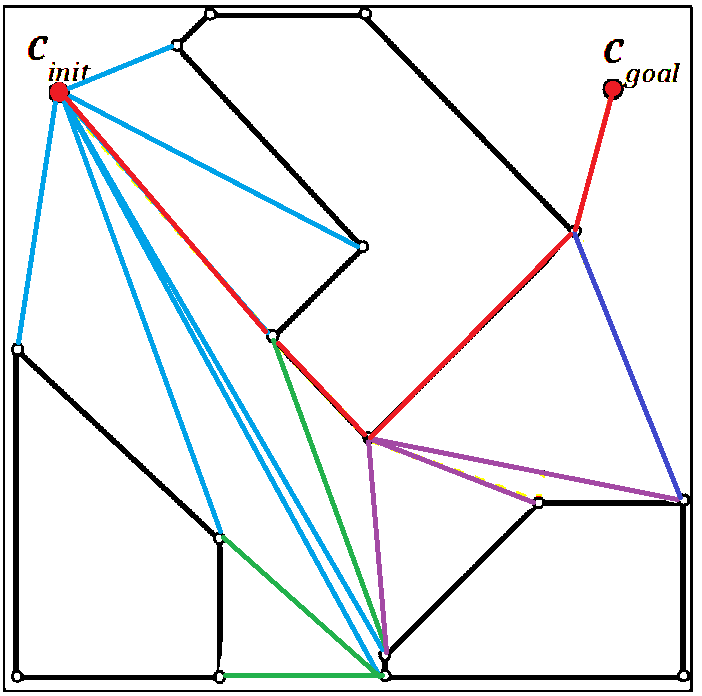


Рисунок 1 – Граф видимости

Граф видимости — это граф взаимной видимости точек пространства, обычно для множества точек и преград на евклидовой плоскости. Любая вершина в графе представляет точку пространства, а любое ребро представляет прямую видимость между точками. То есть, если отрезок прямой, соединяющий две точки пространства, не проходит через какую-либо преграду, в графе будет нарисовано ребро. Если множество точек пространства лежит на прямой, их можно понимать как упорядоченную последовательность. Графы видимости, таким образом, распространяются на область анализа временных рядов.

## Алгоритм Дейкстры

После построения графа видимости для поиска наикратчайшего пути использовался алгоритм Дейкстры: Каждой вершине приписывается вес – это вес пути от начальной вершины до данной. Также каждая вершина может быть выделена. Если вершина выделена, то путь от нее до начальной вершины кратчайший, если нет – то временный. Обходя граф, алгоритм считает для каждой вершины маршрут, и, если он оказывается кратчайшим, выделяет вершину. Весом данной вершины становится вес пути. Для всех соседей данной вершины алгоритм также рассчитывает вес, при этом ни при каких условиях не выделяя их. Алгоритм заканчивает свою работу, дойдя до конечной вершины, и весом кратчайшего пути становится вес конечной вершины.

Шаг 1. Всем вершинам, за исключением первой, присваивается вес равный бесконечности, а первой вершине – 0. Все вершины не выделены. Первая вершина объявляется текущей.

Шаг 2. Вес всех невыделенных вершин пересчитывается по формуле: вес невыделенной вершины есть минимальное число из старого веса данной вершины, суммы веса текущей вершины и веса ребра, соединяющего текущую вершину с невыделенной.

Шаг 3. Среди невыделенных вершин ищется вершина с минимальным весом. Если таковая не найдена, то есть вес всех вершин равен бесконечности, то маршрут не существует. Следовательно, выход. Иначе, текущей становится найденная вершина. Она же выделяется.

Шаг 4. Если текущей вершиной оказывается конечная, то путь найден, и его вес есть вес конечной вершины.

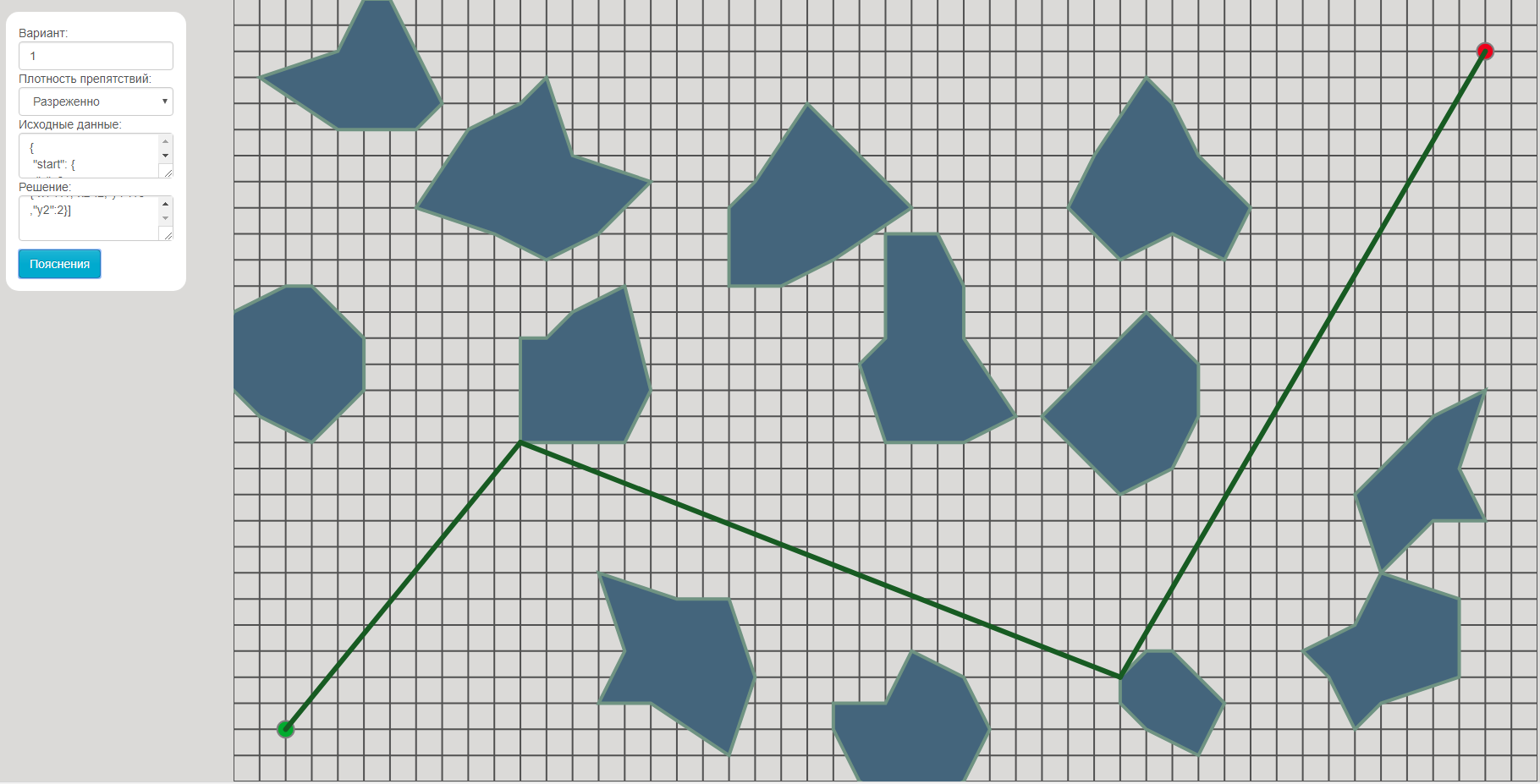
Шаг 5. Переход на шаг 2.

# Реализация алгоритма

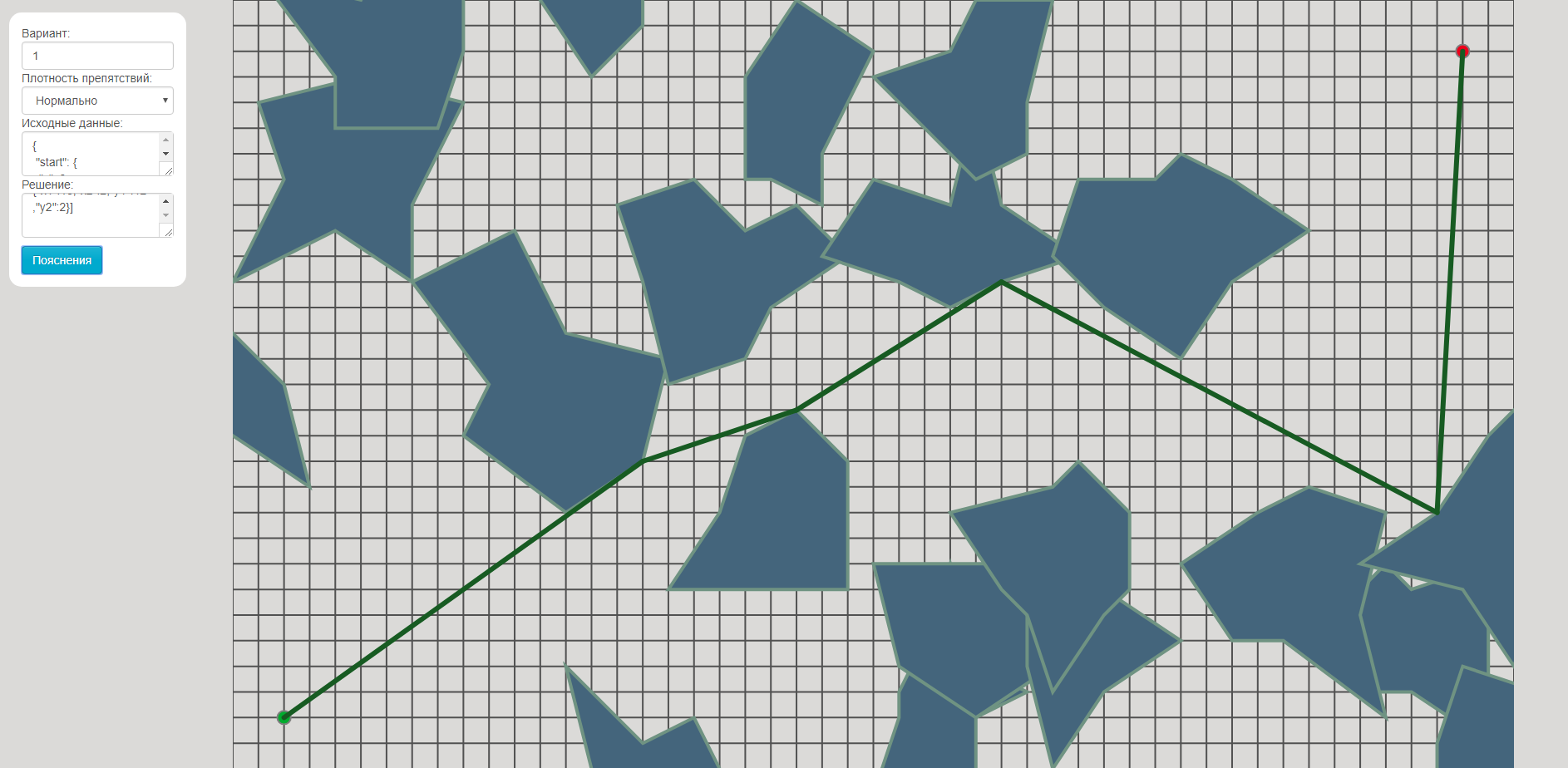
Для реализации алгоритма потребовалось реализовать следующие классы:

# Результат работы программы

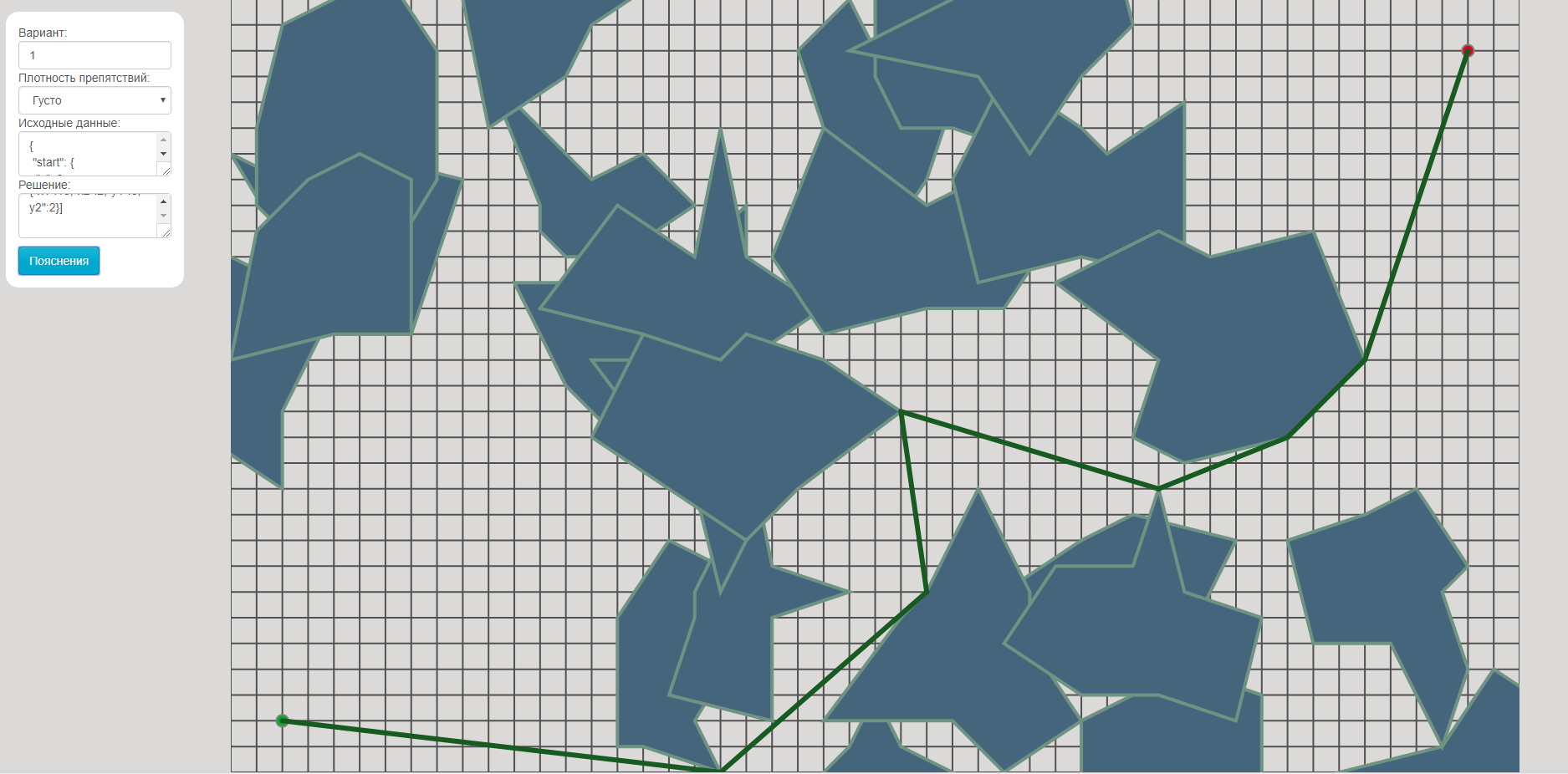
Разреженно 0.669



Нормально 9.246

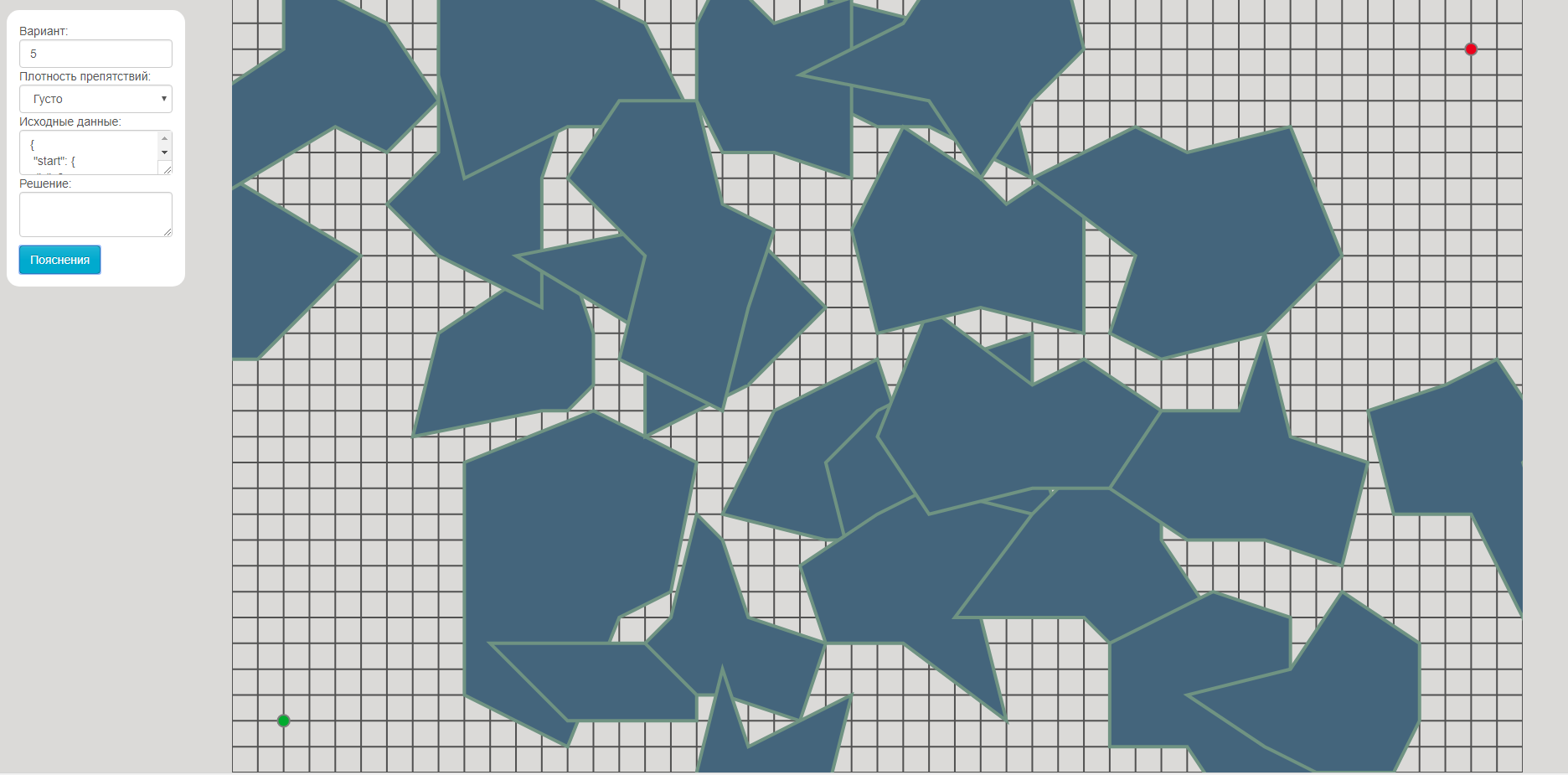


Густо 12.707



Густо 8.377

НЕТ РЕШЕНИЯ



# Список использованной литературы.

# Приложение А. Листинги программного кода.