Петрозаводский государственный университет
Институт математики и информационных технологий

Направление подготовки бакалавриата

09.03.04 Программная инженерия

Отчет по дисциплине «Верификация программного обеспечения»

Система для построения траектории мобильного объекта на полигоне

Выполнил:
студент группы 22407

Д.А.Костин _____
подпись

Руководитель:
К.А.Кулаков

_____
подпись

Итоговая оценка

_____
оценка

Петрозаводск — 2023

Содержание

1 Объект тестирования

1.1 Описание объекта тестирования

Объектом тестирования является система, разработанная на языке программирования *C#* в рамках курсовой работы за 2 и 3 курсы обучения.

В системе:

- Внедрён алгоритм, который по карте местности и показаниям данных о траектории, полученной из векторов перемещения инерциального датчика, исправляет данную траекторию (чтобы она была как можно соответственней реальной).
- Для получения данных о траектории, внедрен алгоритм генерации синтетичиских данных о **траектории** – а именно траекторию, составленную из точек (x, y) с заданным началом (рисунок 2), и строила визуализацию работы алгоритма.

Задачей, решаемой системой визуализации, является по карте местности в плоскости (последовательность точек, составляющих границы препятствий), стартовой и конечной точкам:

1. Сгенерировать (сымитировать) траекторию мобильного объекта (будь то человека, или робота) между стартовой и конечной точками – последовательность координат (x, y) вычисленных из показаний инерциального измерительного датчика.
2. Построить траекторию, которая бы учитывала карту местности и строилась на основании разрабатываемого в системе алгоритма исправления траектории.

Алгоритм, который генерирует исправленную траекторию:

- На вход получает последовательность точек (x,y), карту местности – множество точек ограничивающих область снаружи, назовём планом, и множество множеств точек, задающих многоугольники внутри плана – назовём дырами.
- На выходе алгоритм выдаёт траекторию, которая находится внутри множества точек плана, но вне дыр (множество сложного полигона), строящуюся на основе переданной траектории.

Исходные начальные данные сложного полигона и траектории, идущие вместе с системой, находятся по ссылке:

- https://github.com/DanilaKostin/CourseProject/sample_polygon.txt

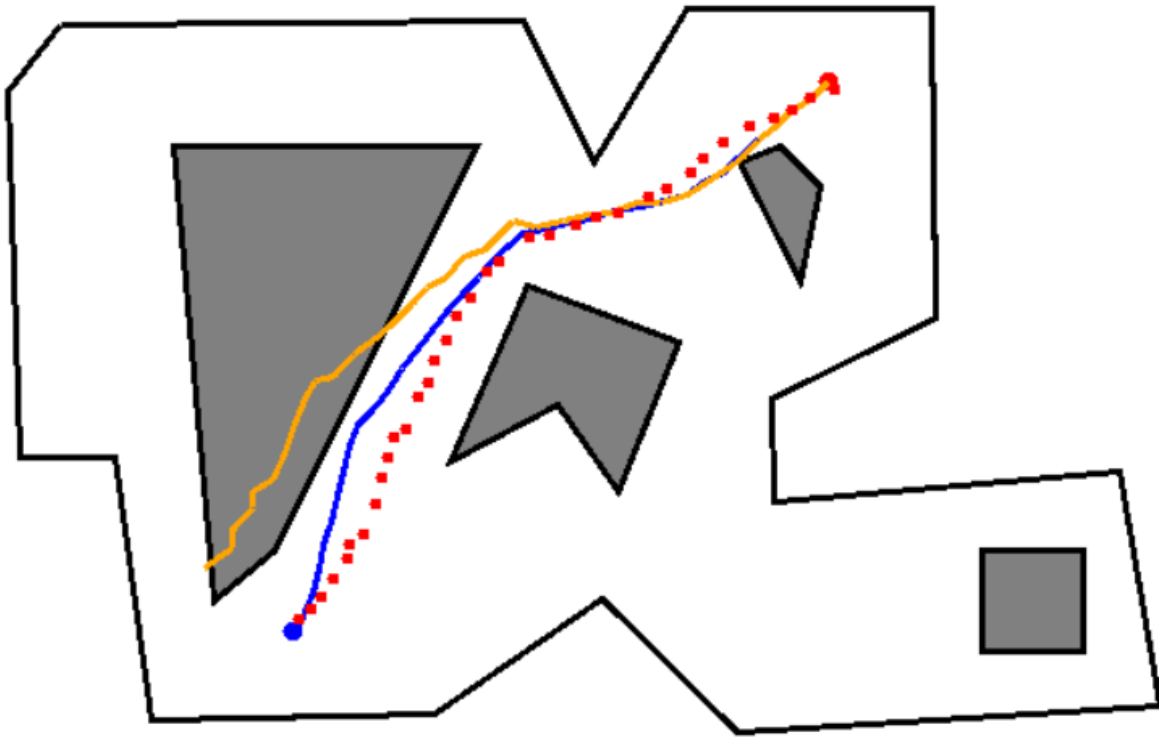


Рис. 1 – Исходные данные, синим – реальная траектория, оранжевым – зафиксированная, красными точками – исправленная

1.2 Функции системы

Система предоставляет следующие функции:

1. Выбор точек начала и конца траектории;
2. Очистка точек начала и конца траектории;
3. Задание коэффициента ошибки сымитированной траектории – максимальное отклонение в градусах каждого звена траектории от реальной;
4. Задание длины шага – максимальная длина вектора перемещения;
5. Построение новой реальной траектории между заданными точками, не проходящей через дыры;

6. Построение исправленной траектории на основе алгоритма;

7. Считывание карты помещения из файла в определенном формате;

1.3 Архитектура системы

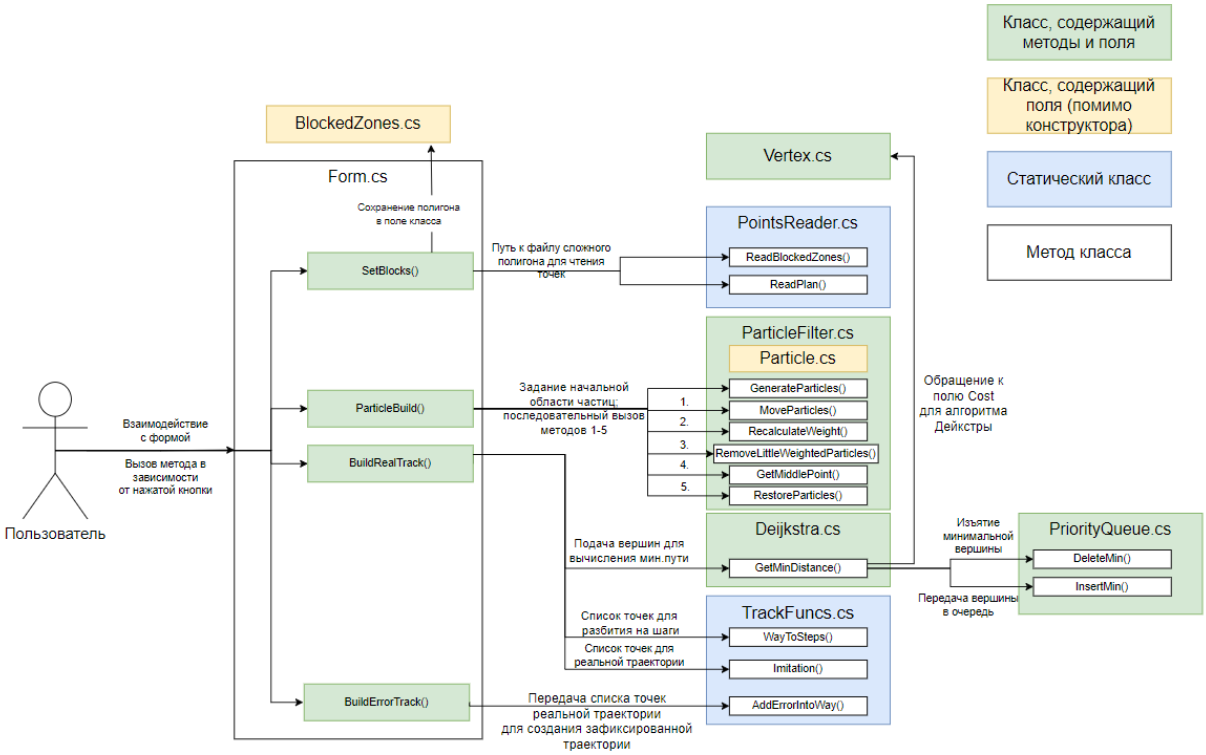


Рис. 2 – Архитектура системы

На схеме рисунка 2:

- 1 – вызов метода MoveParticles для перемещения частиц в направлении зафиксированной траектории;
- 2 – вызов метода RecalculateWeight для пересчёта весов в зависимости от расстояние до точки траектории;
- 3 – вызов метода RemoveLittleWeightedParticles для удаления частиц с малым весом;
- 4 – вызов метода GetMiddlePoint для вычисления центральной точки – очередная точка реальной траектории;
- 5 – вызов метода RestoreParticles для восстановления числа частиц.

1.4 Описание модулей

1.4.1 BlockedZones.cs

Класс, используемый для хранения данных о карте местности. Он хранит в себе следующие поля:

- public List<List<Vertex» startPoints – стартовые границы дыр
- public List<List<Vertex» Points – расширенные границы дыр
- public List<Vertex> startPlan – стартовые границы плана
- public List<Vertex> Plan – отдалённые границы плана
- public Vertex startVert – стартовая вершина пути
- public List<Vertex> endVert – промежуточные точки пути

Метод:

- public BlockedZones(List<Vertex> Plan, List<List<Vertex» Points) – конструктор, устанавливающий соответствующие поля класса.

1.4.2 Vertex.cs

Класс, представляющий собой какую-либо вершину на полигоне.

Поля:

- public Vertex ParentVertex – предыдущая вершина в пути, необходимо для восстановления пути до конечной вершины.
- public bool Visited – флаг посещения вершины алгоритмом Дейкстры.
- public double Cost – цена вершины (в алгоритме Дейкстры).
- public Point Point – точка, соответствующая координатам вершины.
- public Dictionary <Vertex, double> Connections – словарь ключ–вершина, значение – расстояние до неё.

Методы:

- public Vertex(Point p) – конструктор класса, задающий поле Point равным переданному аргументу, инициализирует словарь Dictionary, Cost становиться максимальному числу с точкой.
- public void AddConnection(Vertex t) – добавляет переданную в аргументе вершину в словарь Connections с ключём – расстояние между вершинами, также и для переданной вершины.
- public double FindDistance(Point p1, Point p2) – находит расстояние между двумя переданными точками Point.

1.4.3 PriorityQueue.cs

Класс, реализующий такую структуру данных, как очередь с приоритетом (для алгоритма Дейкстры).

Поля:

- readonly List<Tuple<Vertex, double» PriorityQueue – предсталяет собой кортеж ключ-значение, реализующий хранение очереди.

Методы:

- public void Insert(Tuple<Vertex, double> v) – метод, производящий вставку кортежа (вершина, стоимость) в соответствующее место очереди с приоритетом.
- public Vertex DeleteMin() – метод, извлекающий первый элемент в очереди и перестраивающий очередь.

1.4.4 Deijkstra.cs

Класс, реализующий алгоритм Дейкстры на графе видимости.

Метод:

- public List<Point> GetMinDistance(Vertex start, Vertex end) – метод, возвращающий список точек, составляющих минимальную путь между переданными 2-мя вершинами, в порядке от первой переданной к второй, либо одну точку, если переданные вершины совпадают.

1.4.5 PointsReader.cs

Статический класс, предназначенный для чтения точек сложного полигона.

Методы:

- static public List<List<Vertex» ReadBlockedZones(string FilePath) – метод, читающий переданный в аргументе файл, и возвращающий список списков вершин дыр в формате [int, int], ..., [int, int].
- static public List<Vertex> ReadPlan(string FilePath) – метод, читающий переданный в аргументе файл в формате [int, int], ..., [int, int], и возвращающий список вершин плана.

1.4.6 TrackFuncs.cs

Статический класс, хранящий методы для генерирования траектории, которая могла бы быть зафиксирована инерциальным датчиком.

Методы:

- public List<Point> WayToSteps(List<Point> way, int L) – метод, получающий на вход список точек пути и разбивающий отрезки, составленные последовательными парами этих точек, на части с заданной длиной шага L, и возвращающий этот список точек.
- public List<Point> Imitation(List<Point> p, int d) – метод, получающий на вход последовательность точек и искривляющий траекторию, изменяя угол между отрезками траектории на случайное число от 0 до d градусов, делая её волнообразной (нелинейной), и возвращает данную последовательность, при этом начальная и конечная точки совпадают с соответствующими точками исходной последовательности.
- public List<Point> AddErrorIntoWay(List<Point> way, int e) – метод, получающий на вход последовательность точек, последовательно изменяет углы между отрезками на

некоторый угол – коэффициент ошибки от 0 до e градусов, таким образом возвращает траекторию с смещением – то есть конечная точка не совпадает с конечной точкой исходного списка.

1.4.7 ParticleFilter.cs

Класс, реализующий алгоритм фильтра частиц.

Поля:

- private BlockedZones BlockedZones – поле, хранящее класс сложного полигона.
- public List<Particle> particle – поле, хранящее список частиц.

Методы:

- public ParticleFilter(BlockedZones b) – конструктор класса, запоминающий параметр в поле.
- public void GenerateParticles(int count, int max_x, int max_y) – метод, генерирующий максимум count частиц с координатами от 0, до max_x по X и до max_y по Y и весом в 1/count.
- public void MoveParticles(double length, double angle) – совершает движение всех частиц на расстояние length с начальным поворотом, прибавляемым к ориентации, на угол angle.
- public void RecalculateWeight(Point orientir) – метод, пересчитывающий веса всех частиц относительно переданной точки, если расстояние до частицы превышает 40, то вес равняется 0 – т.к. частица слишком далека от ориентира.
- public void RemoveLittleWeightedParticles(double maxWeight) – метод, удаляющий все частицы с весом, меньшим maxWeigth, при этом вес – принадлежит отрезку от 0 до 1.
- public Point GetMiddlePoint() – возвращает среднюю точку среди всех частиц.
- public void RestoreParticles(Point central_point, int count, int radius) – метод, восстанавливающий число частиц до количества count вокруг точки central_point в радиусе radius.

1.4.8 Particle.cs

Класс, реализующий частицу для фильтра частиц, характеризующуюся весом, координатами и ориентацией.

Поля:

- public Point Point;
- public double Orientation;
- public double Weight;

Методы:

- public Particle(Point point, double orientation, double weight) – конструктор, инициализирующий поля.

2 Стратегия блочного тестирования

Блочное тестирование – то есть тестирование отдельных методов классов отдельно друг от друга, при тестировании описанной системы будет производится при помощи Unit Testing Framework (встроенная в Visual Studio система тестирования), которая достаточно удобна для тестирования проектов на языке $C\#$.

Блочному тестированию подвергнем все методы модулей, участвующие в работе алгоритмов генерации траектории и её исправления:

- PriorityQueue.cs
- Deijkstra.cs
- ParticleFilter.cs
- TrackFuncs.cs
- PointsReader.cs

Не будем нестировать следующие классы:

- Particle.cs, BlockedZones – т.к. он содержит лишь поля и конструктор без вычислений.

Блочные тесты разобьём на следующие классы:

- Позитивный – тест, получающий на вход обычные данные, на которых результат работы очевиден и прост.
- Негативный – тест, получающий на вход данные, которые вызывают особую обработку программой и возврат ошибки/исключения.
- Граничный – тест, получающий на вход граничные данные, которые находятся на границе данных для нормальной работы.

3 Стратегия интеграционного тестирования

При интеграционном тестировании будет проверяется взаимодействие частей системы между собой. Тестирование системы также будет производится при помощи Unit Testing Framework.

Для проверки интеграционного тестирования, будем последовательно вызывать модули и методы, составляющие единый этап в работе алгоритма.

На схеме интеграции на рисунке 5:

- Первая группа методов стоят реальную траекторию.
- Вторая группа – стоит зафиксированную.
- Третья группа – строит исправленную.

4 Стратегия аттестационного тестирования

В ходе аттестационное тестирования будет протестирована работоспособность системы и её возможность осуществлять заявленный функционал в пункте 1.2.

Проверка будет производится человеком, выполняющим действия по заранее заданным инструкциям, который после выполнения действия будет сверяется с заранее заданными

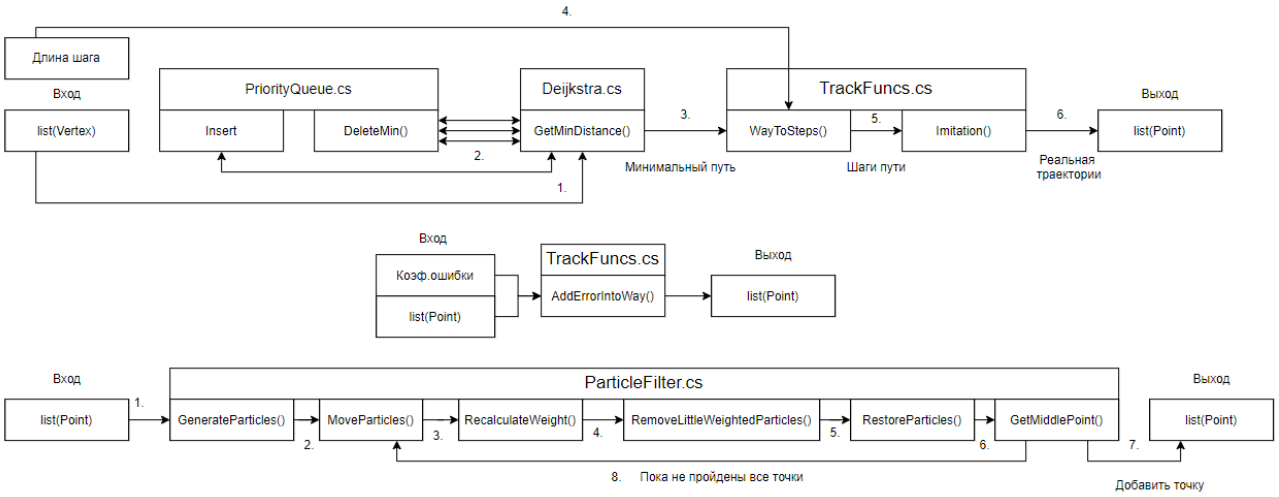


Рис. 3 – Схема интеграции

результатами. Тест считается пройденным, если ожидаемый результат совпадает с фактическим результатом. В противном случае тест считается не пройденным.

5 Стратегия нагрузочного тестирование

В ходе нагрузочного тестирования будет проверяться способность основных компонент алгоритмов выдавать результат работы за конечный период времени без ошибок переполнения памяти.

Тестирование будет производиться путём замера времени построения исправленной траектории на заранее выбранных координатах начала и конца реальной траектории на сложном полигоне из пункта 1.1.

Критерий прохождения зависит от средней оценки времени работы: для 100 входных точек время работы алгоритма должно занимать не более 700 миллисекунд.

При числе точек в количестве, равном n, скорость построение исправленной траектории не должна превышать $500 + n^{1.1}$ миллисекунд, где n – число вершин.

Количество точек	Ожидаемое время
100	700млс.
1000	2600млс.
10000	25.8сек.

6 Блочное тестирование

6.1 PriorityQueue.cs

6.1.1 Метод Insert

Тест Б1, позитивный

Объект тестирования	Метод Insert
Цель	проверка правильности вставки элемента в очередь.
Описание	проверка того, что пустая очередь вставит элемент в вершину.
Начальное состояние	–
Входные данные	Пара ключ-значение (1, Vertex(0,0))
Ожидаемый результат	Поле Queue класса содержит массив из одного элемента – переданной пары.

Тест Б2, позитивный

Объект тестирования	Метод Insert
Цель	проверка правильности вставки элемента в очередь.
Описание	проверка того, что при наличии в очереди элемента в вершине, новый больший элемент получит индекс 1.
Начальное состояние	очередь содержит элемент в вершине, равный (1, Vertex(0,0)).
Входные данные	Пара ключ-значение (2, Vertex(1,1))
Ожидаемый результат	Поле Queue класса содержит массив из двух элементов, на индексе 1 стоит значение с ключом 2.

Тест Б3, позитивный

Объект тестирования	Метод Insert
Цель	проверка правильности вставки элемента в очередь.
Описание	проверка того, что при наличии в очереди левой ветки двоичной кучи, новый элемент вставится в правую.
Начальное состояние	очередь содержит следующие элементы (1, Vertex(0,0)), (2, Vertex(0,0)).
Входные данные	Пара ключ-значение (3, Vertex(2,2))
Ожидаемый результат	Поле Queue класса содержит массив из трех элементов, на индексе 2 стоит значение с ключом 3.

Тест Б4, позитивный

Объект тестирования	Метод Insert
Цель	проверка правильности вставки элемента в очередь.
Описание	проверка того, переданная пара с значением, которое имеется в очереди, вставится после элемента с данным значением в очереди.
Начальное состояние	очередь содержит следующие элементы (1, Vertex(0,0)), (2, Vertex(1,1)), (3, Vertex(2,2)).
Входные данные	Пара ключ-значение (1, Vertex(-1,-1))
Ожидаемый результат	Поле Queue класса содержит массив из четырёх элементов, на индексе 0 стоит значение с ключом 1, на индексе 1 – стоит ключ 1, на индексе 2 – ключ 3, а на индексе 3 – ключ 2.

6.1.2 Метод DeleteMin

Тест Б5, позитивный

Объект тестирования	Метод DeleteMin
Цель	проверка правильности извлечения элемента.
Описание	проверка того, извлечётся элемент в вершине.
Начальное состояние	очередь содержит следующие элемент (1, Vertex(0,0)), (2, Vertex(0,2)), (3, Vertex(0,3))
Входные данные	–
Ожидаемый результат	Поле Queue класса содержит массив (2,Vertex(0,2)) (3, Vertex(0,3)), возвращается элемент (1, Vertex(0,0))

Тест Б6, граничный

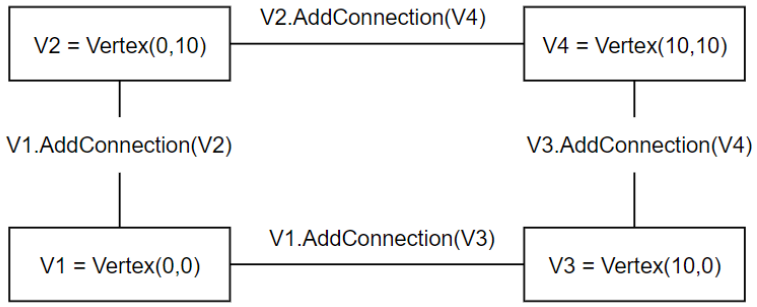
Объект тестирования	Метод DeleteMin
Цель	проверка правильности извлечения элемента.
Описание	проверка при одинаковых ключах.
Начальное состояние	очередь содержит следующие элемент (1, Vertex(0,0)), (1, Vertex(0,2)), (1, Vertex(0,3))
Входные данные	–
Ожидаемый результат	Поле Queue класса содержит массив (1,Vertex(0,2)) (1, Vertex(0,3)), возвращается элемент (1, Vertex(0,0))

Тест Б7, негативный

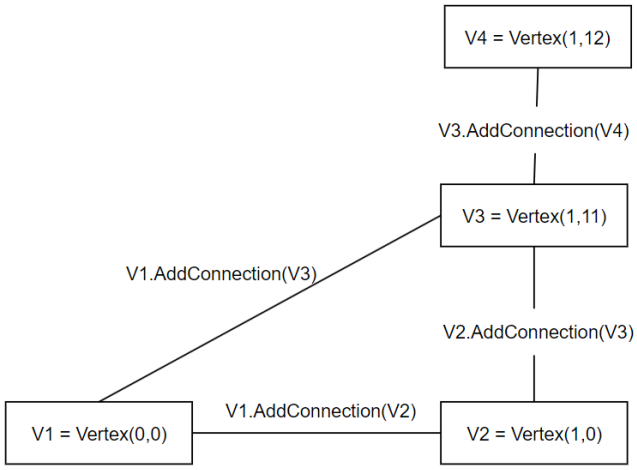
Объект тестирования	Метод DeleteMin
Цель	проверка правильности извлечения элемента.
Описание	проверка при пустой очереди.
Начальное состояние	очередь ничего не содержит
Входные данные	–
Ожидаемый результат	Exception('Очередь пуста')

6.2 Deijkstra.cs

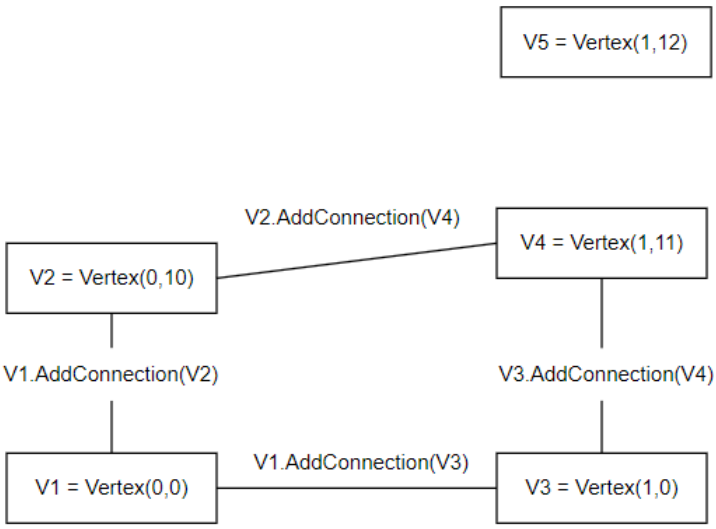
Тест Б8, позитивный

Объект тестирования	Метод GetMinDistance
Цель	проверка правильности нахождения кратчайшего пути.
Описание	проверка простого случая, когда вершина кратчайший путь – в соседней вершине.
Начальное состояние	 1. Создание объектов классов вершин. 2. Вызов методов AddConnection()
Входные данные	V1, V2
Ожидаемый результат	Список [V1, V2].

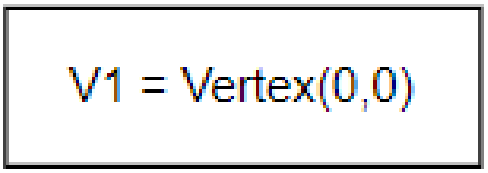
Тест Б9, позитивный

Объект тестирования	Метод GetMinDistance
Цель	проверка правильности нахождения кратчайшего пути.
Описание	проверка случая, когда кратчайший путь в соседней вершине, но кратчайшее дерево не идёт в неё на 1 шаге.
Начальное состояние	 1. Создание объектов классов вершин. 2. Вызов методов AddConnection()
Входные данные	V1, Vertex4
Ожидаемый результат	Список [V1, V3, V4].

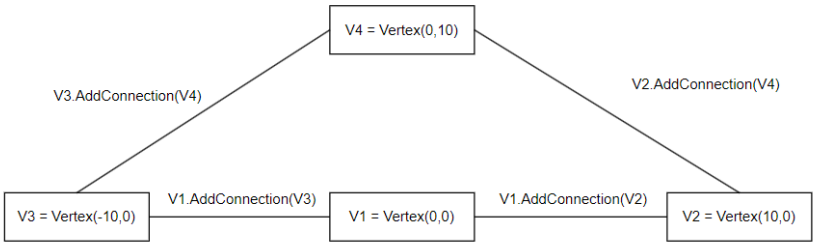
Тест Б10, негативный

Объект тестирования	Метод GetMinDistance
Цель	проверка реакции на случаи разных компонент связности.
Описание	проверка случая, нет пути в конечную вершину
Начальное состояние	 1. Создание объектов классов вершин. 2. Вызов методов AddConnection()
Входные данные	V1, V5
Ожидаемый результат	Exception(’Нет пути между вершинами’).

Тест Б11, граничный

Объект тестирования	Метод GetMinDistance
Цель	проверка реакции на частные случаи.
Описание	проверка случая с 1 вершиной
Начальное состояние	 1. Создание объектов классов вершин. 2. Вызов методов AddConnection(V1, V1)
Входные данные	V1, V1
Ожидаемый результат	Список [V1].

Тест Б12, позитивный

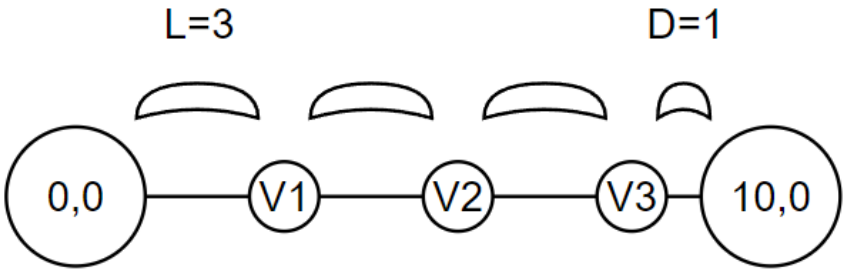
Объект тестирования	Метод GetMinDistance
Цель	проверка реакции на частные случаи.
Описание	проверка случая с двумя путями одинаковой длины
Начальное состояние	 1. Создание объектов классов вершин. 2. Вызов методов AddConnection() сначала для вершин V1, V2, V4, затем для V1, V3, V4.
Входные данные	V1, V4
Ожидаемый результат	Список [V1, V2, V4].

6.3 TrackFuncs.cs

6.3.1 Метод WayToSteps

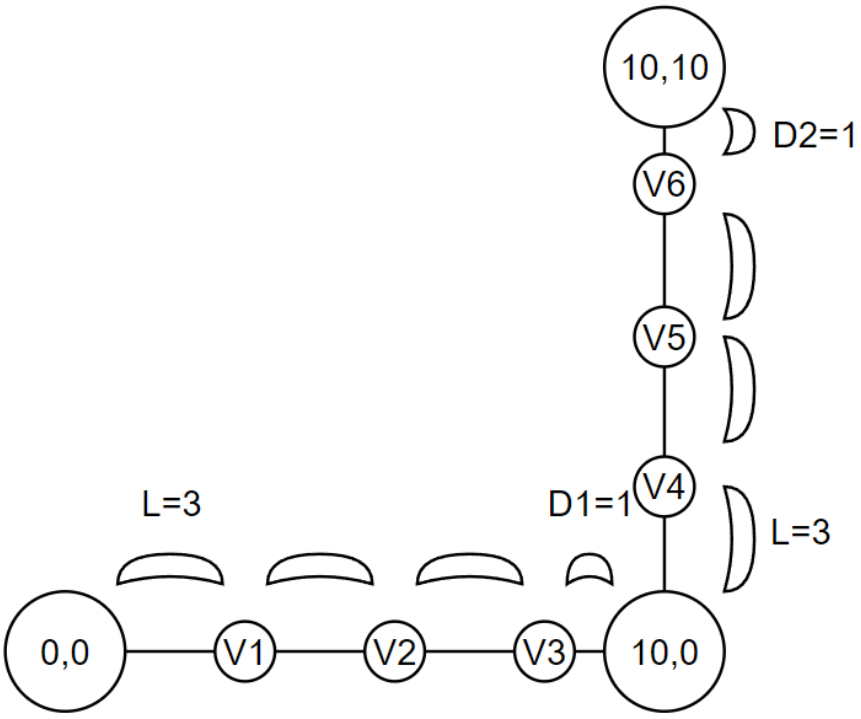
Тест Б13, позитивный

Объект тестирования	Метод WayToSteps
Цель	проверка правильности разбития ломаной на шаги.
Описание	проверка случая для отрезка
Входные данные	Vertex(0,0), Vertex(10, 0), L=3 (длина шага)
Ожидаемый результат	Список точек [Point(0,0), V1=Point(3,0), V2=Point(6,0), V3=Point(9,0), Point(10,0)].



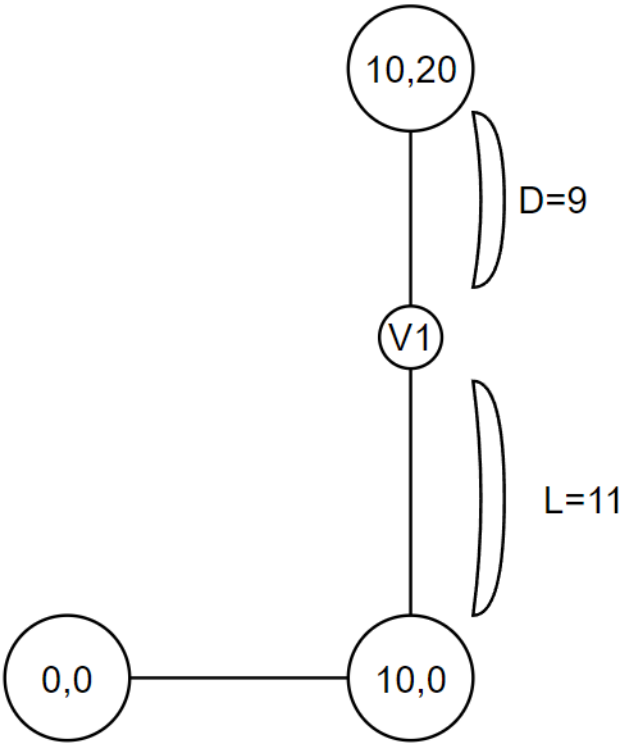
Тест Б14, позитивный

Объект тестирования	Метод WayToSteps
Цель	проверка правильности разбития ломаной на шаги.
Описание	проверка случая для ломаной
Входные данные	[Vertex(0,0), Vertex(10, 0), Vertex(10, 10)], L=3 (длина шага)
Ожидаемый результат	Список точек [Point(0,0), V1=Point(3,0), V2=Point(6,0), V3=Point(9,0), Point(10,0), V4=Point(10, 3), V5=Point(10, 6), V6=Point(10, 9), Point(10,10)]



Тест Б15, позитивный

Объект тестирования	Метод WayToSteps
Цель	проверка правильности разбития ломаной на шаги.
Описание	проверка случая, когда шаг длиннее отрезка ломаной
Входные данные	[Vertex(0,0), Vertex(10, 0), Vertex(10, 20)], L=11 (длина шага)
Ожидаемый результат	Список точек [Point(0,0), Point(10,0), V1=Point(10,11), Point(10,20)]



Тест Б16, негативный

Объект тестирования	Метод WayToSteps
Цель	проверка частного случая.
Описание	проверка случая с 1 точкой
Входные данные	[Vertex(0,0)], L=1 (длина шага)
Ожидаемый результат	Exception('Нужно минимум 2 точки')

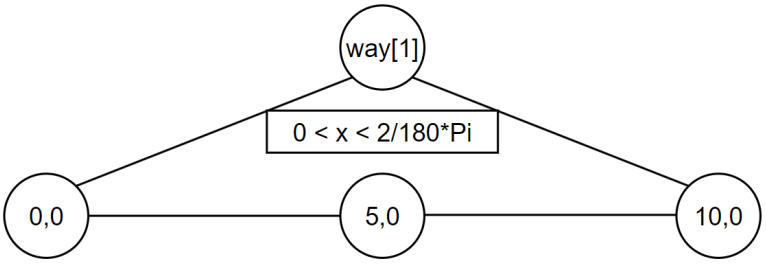
Тест Б17, граничный

Объект тестирования	Метод WayToSteps
Цель	проверка частного случая.
Описание	проверка случая с совпадающей точкой
Входные данные	[Vertex(0,0), Vertex(0,0)], L=1 (длина шага)
Ожидаемый результат	[Point(0,0), Point(0,0)]

6.3.2 Метод Imitation

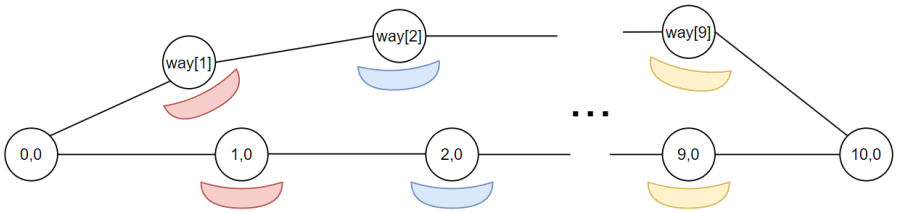
Тест Б18, позитивный

Объект тестирования	Метод Imitation
Цель	проверка имитации траектории.
Описание	проверка случая с 3 точками
Входные данные	p=[Point(0,0), Point(5,0), Point(10,0)], d=2
Ожидаемый результат	Массив way: way[0] == Point(0,0), way[-1] == Point(10,0), len(way) == 3, 0 < |Atan2({p[0],p[1]},{p[1],p[2]}) - Atan2({way[0],way[1]},{way[1],way[2]})| <= 2/180*Pi



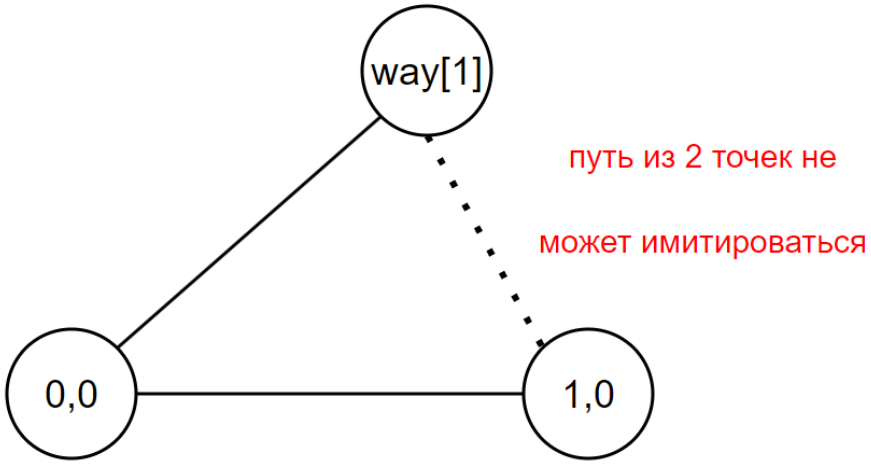
Тест Б19, позитивный

Объект тестирования	Метод Imitation
Цель	проверка имитации траектории.
Описание	проверка случая с 11 точками
Входные данные	p=[Point(0,0), Point(1,0), Point(2,0), ..., Point(10,0)], d=2
Ожидаемый результат	Массив way: way[0] == Point(0,0), way[-1] == Point(10,0), len(way) == 11, для всех i от 1 до 8: 0 < |Atan2({p[i-1],p[i]},{p[i],p[i+1]}) - Atan2({way[i-1],way[i]},{way[i],way[i+1]})| <= 2/180*Pi



Тест Б20, негативный

Объект тестирования	Метод Imitation
Цель	проверка имитации траектории.
Описание	проверка случая с 2 точками
Входные данные	p=[Point(0,0), Point(1,0)], d=2
Ожидаемый результат	Exception('Для имитации необходимо минимум 3 точки')



Тест Б21, негативный

Объект тестирования	Метод Imitation
Цель	проверка имитации траектории.
Описание	проверка случая с 3 точками, 2 из которых совпадают
Входные данные	p=[Point(0,0), Point(1,0), Point(1,0)], d=2
Ожидаемый результат	Exception('Для имитации траектории не должны иди подряд совпадающие точки')

Тест Б22, негативный

Объект тестирования	Метод Imitation
Цель	проверка имитации траектории.
Описание	проверка случая с отрицательным максимальным углом поворота
Входные данные	p=[Point(0,0), Point(1,0), Point(2,0)], d=-2
Ожидаемый результат	Exception('Максимальный угол поворота должен быть положительным')

6.3.3 Метод AddErrorIntoWay

Тест Б23, позитивный

Объект тестирования	Метод AddErrorIntoWay
Цель	проверка имитации траектории.
Описание	проверка случая с 3 точками
Входные данные	way=[Point(0,0), Point(5,0), Point(10,0)], e=2
Ожидаемый результат	Массив error_way: , len(way) == 3, |Atan2({error_way[0],error_way[1]},{error_way[1],error_way[2]}) - Atan2({way[0],way[1]},{way[1],way[2]})| <= 2/180*Pi

Тест Б24, негативный

Объект тестирования	Метод AddErrorIntoWay
Цель	проверка имитации траектории.
Описание	проверка случая с отрицательным максимальным углом ошибки
Входные данные	way=[Point(0,0), Point(5,0), Point(10,0)], e=-1
Ожидаемый результат	Exception('Максимальный угол ошибки должен быть положительным')

Тест Б25, граничный

Объект тестирования	Метод AddErrorIntoWay
Цель	проверка имитации траектории.
Описание	проверка случая с двумя точками
Входные данные	way=[Point(0,0), Point(5,0)], e=2
Ожидаемый результат	Массив error_way: |Atan2({way[0],error_way[1]}) - Atan2({way[0],way[1]},)| <= 2/180*Pi

6.4 ParticleFilter.cs

6.4.1 Метод GenerateParticles(int count, int start_x, int start_y, int max_x. int max_y)

Тест Б26, позитивный

Объект тестирования	Метод GenerateParticles
Цель	проверка параметров сгенерированный частиц.
Описание	проверка простого случая на 100 частицах
Входные данные	count=100, start_x = 0, start_y = 0, max_x = 100, max_y = 100
Ожидаемый результат	Поле particle содержит 100 элементов класса Particle, координаты $Particle_i$: $0 < Particle_i.Point.X < 100$ и $0 < Particle_i.Point.Y < 100$, Particle.Weight = 0.01

Тест Б27, позитивный

Объект тестирования	Метод GenerateParticles
Цель	проверка параметров сгенерированной частицы.
Описание	проверка правильности вычисления веса при 1 частице
Входные данные	count=1, start_x = 0, start_y = 0, max_x = 1, max_y = 1
Ожидаемый результат	Поле particle содержит 1 элемент класса Particle, координаты Particle: $0 <$ Particle.Point.X < 1 и $0 <$ Particle.Point.Y < 1, Particle.Weight = 1

Тест Б28, негативный

Объект тестирования	Метод GenerateParticles
Цель	проверка частных случаев.
Описание	проверка ошибки при отрицательном количестве частиц
Входные данные	count=-1, start_x = 0, start_y = 0, max_x = 1, max_y = 1
Ожидаемый результат	Exception('Количество частиц должно быть не меньше 1')

Тест Б29, граничный

Объект тестирования	Метод GenerateParticles
Цель	проверка частных случаев.
Описание	проверка случая при координатах равных 0
Входные данные	count=10, start_x = 0, start_y = 0, max_x = 0, max_y = 0
Ожидаемый результат	Поле particle содержит 10 элементов класса Particle, координаты Particle_i: Particle_i.Point.X = 0 и Particle_i.Point.Y = 0, Particle.Weight = 0.1

Тест Б30, позитивный

Объект тестирования	Метод GenerateParticles
Цель	проверка частных случаев.
Описание	проверка случая при max_x < start_x и max_y < start_y
Входные данные	count=10, start_x = 10, start_y = 10, max_x = 0, max_y = 0
Ожидаемый результат	Поле particle содержит 10 элементов класса Particle, координаты $Particle_i$: $0 < Particle_i.Point.X < 10$ и $0 < Particle_i.Point.Y < 10$, Particle.Weight = 0.1

6.4.2 Метод MoveParticles(double length, double angle)

Тест Б31, позитивный

Объект тестирования	Метод MoveParticles
Цель	проверка движения частицы.
Описание	проверка простого движения без поворота
Начальное состояние	Поле particle содержит частицу Particle(Point(0,0), 0, 1)
Входные данные	length=10, angle = 0
Ожидаемый результат	Поле particle содержит 1 элемент класса Particle, координаты Particle: $Particle_0.Point.X = 10$ и $Particle_0.Point.Y = 0$, Particle.Weight = 1, Particle.Orientation = 0

Тест Б32, позитивный

Объект тестирования	Метод MoveParticles
Цель	проверка движения частиц.
Описание	проверка простого движения без поворота
Начальное состояние	Поле particle содержит частицы [Particle(Point(0,0), 0, 0.5), Particle(Point(1,1), 0, 0.5)]
Входные данные	length=10, angle = 0
Ожидаемый результат	Поле particle содержит 2 элемента класса Particle, координаты Particle: $Particle_0$.Point.X = 10 и $Particle_0$.Point.Y = 0, Particle.Weight = 0.5, Particle.Orientation = 0; $Particle_1$.Point.X = 11 и $Particle_1$.Point.Y = 1, Particle.Weight = 0.5, Particle.Orientation = 0

Тест Б33, позитивный

Объект тестирования	Метод MoveParticles
Цель	проверка движения частицы.
Описание	проверка простого движения с поворотом
Начальное состояние	Поле particle содержит частицу [Particle(Point(0,0), 0, 1)]
Входные данные	length=10, angle = Pi/2
Ожидаемый результат	Поле particle содержит 1 элемент класса Particle, координаты Particle: $Particle_0$.Point.X = 0 и $Particle_0$.Point.Y = 10, $Particle_0$.Weight = 1, $Particle_0$.Orientation = 0.5*Pi

Тест Б34, позитивный

Объект тестирования	Метод MoveParticles
Цель	проверка движения частиц.
Описание	проверка простого движения с поворотом
Начальное состояние	Поле particle содержит частицы [Particle(Point(0,0), 0, 0.5), Particle(Point(0,0), Pi, 0.5)]
Входные данные	length=10, angle = Pi/2
Ожидаемый результат	Поле particle содержит 2 элемента класса Particle, координаты Particle: $Particle_0$.Point.X = 0 и $Particle_0$.Point.Y = 10, Particle.Weight = 0.5, Particle.Orientation = 0.5*Pi; $Particle_1$.Point.X = 0 и $Particle_1$.Point.Y = -10, Particle.Weight = 0.5, Particle.Orientation = 1.5*Pi

Тест Б35, позитивный

Объект тестирования	Метод MoveParticles
Цель	проверка частного случая движения частиц.
Описание	проверка движения при повороте на большой угол
Начальное состояние	Поле particle содержит частицу [Particle(Point(0,0), 0, 1)]
Входные данные	length=10, angle = Pi*100.5
Ожидаемый результат	Поле particle содержит 1 элемент класса Particle, координаты Particle: $Particle_0$.Point.X = 0 и $Particle_0$.Point.Y = 10, Particle.Weight = 1, Particle.Orientation = 0.5*Pi;

Тест Б36, негативный

Объект тестирования	Метод MoveParticles
Цель	проверка частного случая движения частиц.
Описание	проверка движения при негативном расстоянии
Начальное состояние	Поле particle содержит частицу [Particle(Point(0,0), 0, 1)]
Входные данные	length=-10, angle = 0
Ожидаемый результат	Поле particle содержит 1 элемент класса Particle, координаты Particle: $Particle_0$.Point.X = -10 и $Particle_0$.Point.Y = 0, Particle.Weight = 1, Particle.Orientation = 0;

6.4.3 Метод RemoveLittleWeightedParticles(double maxWeight)

Тест Б37, позитивный

Объект тестирования	Метод RemoveLittleWeightedParticles
Цель	проверка удаления частиц с малым весом.
Описание	проверка удаления при параметре $\in$ (0,1)
Начальное состояние	Поле particle содержит частицу [Particle(Point(0,0), 0, 0.1), Particle(Point(0,0), 0, 0.2), Particle(Point(0,0), 0, 0.7)]
Входные данные	maxWeight = 0.2
Ожидаемый результат	Поле particle содержит [Particle(Point(0,0), 0, 0.7)]

Тест Б38, позитивный

Объект тестирования	Метод RemoveLittleWeightedParticles
Цель	проверка частных случаев
Описание	проверка оставления частиц при параметре, равном 0.
Начальное состояние	Поле particle содержит частицу [Particle(Point(0,0), 0, 0.1), Particle(Point(0,0), 0, 0.2), Particle(Point(0,0), 0, 0.7)]
Входные данные	maxWeight = 0
Ожидаемый результат	Поле particle содержит [Particle(Point(0,0), 0, 0.1), Particle(Point(0,0), 0, 0.2), Particle(Point(0,0), 0, 0.7)]

Тест Б39, негативный

Объект тестирования	Метод RemoveLittleWeightedParticles
Цель	проверка частных случаев
Описание	проверка оставления частиц при отрицательном параметре
Начальное состояние	Поле particle содержит частицу [Particle(Point(0,0), 0, 0.1), Particle(Point(0,0), 0, 0.2), Particle(Point(0,0), 0, 0.7)]
Входные данные	maxWeight = -1
Ожидаемый результат	Exception('Вес должен принадлежать отрезку [0,1]')

Тест Б40, граничный

Объект тестирования	Метод RemoveLittleWeightedParticles
Цель	проверка частных случаев
Описание	проверка удаления всех частиц при большом параметре
Начальное состояние	Поле particle содержит частицу [Particle(Point(0,0), 0, 0.1), Particle(Point(0,0), 0, 0.2), Particle(Point(0,0), 0, 0.7)]
Входные данные	maxWeight = 1
Ожидаемый результат	Поле particle содержит []

6.4.4 Метод GetMiddlePoint()

Тест Б41, позитивный

Объект тестирования	Метод GetMiddlePoint
Цель	проверка простых случаев
Описание	проверка нахождения центральной точки при 1 частице
Начальное состояние	Поле particle содержит частицу [Particle(Point(0,0), 0, 1)]
Входные данные	–
Ожидаемый результат	Point(0,0)

Тест Б42, позитивный

Объект тестирования	Метод GetMiddlePoint
Цель	проверка простых случаев
Описание	проверка нахождения центральной точки при 2 частицах
Начальное состояние	Поле particle содержит частицу [Particle(Point(0,0), 0, 0.5), Particle(Point(10,0), 0, 0.5)]
Входные данные	–
Ожидаемый результат	Point(5,0)

Тест Б43, позитивный

Объект тестирования	Метод GetMiddlePoint
Цель	проверка частных случаев
Описание	проверка нахождения центральной точки при 3 частицах, 2 из которых одинаковы
Начальное состояние	Поле particle содержит частицу [Particle(Point(0,0), 0, 1/3), Particle(Point(0,0), 0, 1/3), Particle(Point(10,0), 0, 1/3)]
Входные данные	–
Ожидаемый результат	Point(3.(3),0)

Тест Б44, негативный

Объект тестирования	Метод GetMiddlePoint
Цель	проверка частных случаев
Описание	проверка исключения при отсутствии частиц
Начальное состояние	Поле particle содержит []
Входные данные	–
Ожидаемый результат	Exception('Нет частиц для нахождения центральной точки')

6.4.5 Метод RestoreParticles(Point central_point, int count, int radius)

Тест Б45, позитивный

Объект тестирования	Метод RestoreParticles
Цель	проверка простого восстановления частиц
Описание	проверка при отсутствии частиц в поле particle
Начальное состояние	Поле particle содержит []
Входные данные	central_point=Point(0,0), count=10, radius = 10
Ожидаемый результат	Поле particle содержит 10 элементов класса Particle: $Particle_i.Point.X^2 + Particle_i.Point.Y^2 \in [-10, 10]$

Тест Б46, позитивный

Объект тестирования	Метод RestoreParticles
Цель	проверка простого восстановления частиц
Описание	проверка наличии частиц в поле particle
Начальное состояние	Поле particle содержит [Particle(Point(0,0), 0, 1)]
Входные данные	central_point=Point(0,0), count=2, radius = 2
Ожидаемый результат	Поле particle содержит 2 элемента класса Particle: [Particle(Point(0,0), 0, 1), $Particle_1$.Point.X^2 + $Particle_i$.Point.$Y^2 \in$ [-2, 2]

Тест Б47, позитивный

Объект тестирования	Метод RestoreParticles
Цель	проверка частных случаев
Описание	проверка при наличии большего числа частиц в поле particle
Начальное состояние	Поле particle содержит [Particle(Point(0,0), 0, 1), Particle(Point(0,0), 0, 1), Particle(Point(0,0), 0, 1)]
Входные данные	central_point=Point(0,0), count=1, radius = 2
Ожидаемый результат	Поле particle содержит [Particle(Point(0,0), 0, 1), Particle(Point(0,0), 0, 1), Particle(Point(0,0), 0, 1)]

Тест Б48, негативный

Объект тестирования	Метод RestoreParticles
Цель	проверка частных случаев
Описание	проверка при отрицательном параметре количества
Начальное состояние	Поле particle содержит [Particle(Point(0,0), 0, 1)]
Входные данные	central_point=Point(0,0), count=-1, radius = 2
Ожидаемый результат	Exception('Число восстанавливаемых частиц должно быть положительным')

Тест Б49, негативный

Объект тестирования	Метод RestoreParticles
Цель	проверка частных случаев
Описание	проверка при отрицательном радиусе
Начальное состояние	Поле particle содержит [Particle(Point(0,0), 0, 1)]
Входные данные	central_point=Point(0,0), count=2, radius = -2
Ожидаемый результат	Exception('Радиус не может быть отрицательным')

6.4.6 Метод RecalculateWeight(Point orientir))

Тест Б50, позитивный

Объект тестирования	Метод RecalculateWeight
Цель	проверка пересчёта веса
Описание	проверка при частице при расстоянии от ориентира до неё менее 40
Начальное состояние	Поле particle содержит [Particle(Point(0,0), 0, 1)]
Входные данные	orientir=Point(10,10)
Ожидаемый результат	Поле particle содержит [Particle(Point(0,0), 0, 1)]

Тест Б51, позитивный

Объект тестирования	Метод RecalculateWeight
Цель	проверка пересчёта веса
Описание	проверка при двух частицах при расстоянии от ориентира до неё менее 40, одна ближе к ориентиру в два раза, чем другая
Начальное состояние	Поле particle содержит [Particle(Point(5,0), 0, 0.5), Particle(Point(10,0), 0, 0.5)]
Входные данные	orientir=Point(0,0)
Ожидаемый результат	Поле particle содержит [Particle(Point(5,0), 0, 0.53), Particle(Point(10,0), 0, 0.47)]

Тест Б52, граничный

Объект тестирования	Метод RecalculateWeight
Цель	проверка пересчёта веса при граничных случаях
Описание	проверка при двух частицах при расстоянии от ориентира до неё менее 40, одна совпадает с ориентиром
Начальное состояние	Поле particle содержит [Particle(Point(5,0), 0, 0.5), Particle(Point(10,0), 0, 0.5)]
Входные данные	orientir=Point(5,0)
Ожидаемый результат	Поле particle содержит [Particle(Point(5,0), 0, 0.57), Particle(Point(10,0), 0, 0.43)]

Тест Б53, граничный

Объект тестирования	Метод RecalculateWeight
Цель	проверка пересчёта веса при граничных случаях
Описание	проверка при двух частицах при выходе частицы за границу от ориентира
Начальное состояние	Поле particle содержит [Particle(Point(5,0), 0, 0.9), Particle(Point(40,0), 0, 0.1)]
Входные данные	orientir=Point(0,0)
Ожидаемый результат	Поле particle содержит [Particle(Point(5,0), 0, 1), Particle(Point(10,0), 0, 0)]

Тест Б54, граничный

Объект тестирования	Метод RecalculateWeight
Цель	проверка пересчёта веса при граничных случаях
Описание	проверка при двух частицах при выходе всех частицы за границу от ориентира
Начальное состояние	Поле particle содержит [Particle(Point(45,0), 0, 0.5), Particle(Point(40,0), 0, 0.5)]
Входные данные	orientir=Point(0,0)
Ожидаемый результат	Поле particle содержит [Particle(Point(45,0), 0, 0), Particle(Point(40,0), 0, 0)]

Тест Б55, негативный

Объект тестирования	Метод RecalculateWeight
Цель	проверка пересчёта веса при отсутствии частиц
Описание	проверка возврата исключения при отсутствии частиц
Начальное состояние	Поле particle содержит []
Входные данные	orientir=Point(0,0)
Ожидаемый результат	Exception('Частицы отсутствуют')

6.5 PointsReader.cs

6.5.1 Метод ReadBlockedZones

Тест Б56, позитивный

Объект тестирования	Метод ReadBlockedZones
Цель	проверка считывания точек дыр
Описание	проверка при правильном формате
Начальное состояние	Файл zones.txt имеет следующее содержимое: "[0, 0], [10, 0], [5, 10] \n [10, 10], [20, 20], [15, 15]"
Входные данные	file_path="zones.txt"
Ожидаемый результат	[[Point(0, 0), Point(10, 0), Point(5, 10)],[Point(10, 10), Point(20, 20), Point(15, 15)]]

Тест Б57, негативный

Объект тестирования	Метод ReadBlockedZones
Цель	проверка считывания точек дыр
Описание	проверка при правильном формате, но пересечении
Начальное состояние	Файл zones.txt имеет следующее содержимое: "[0, 0], [10, 0], [5, 10] \n [0, 0], [20, 20], [5, 10]"
Входные данные	file_path="zones.txt"
Ожидаемый результат	Exception("Дыры пересекаются")

Тест Б58, негативный

Объект тестирования	Метод ReadBlockedZones
Цель	проверка считывания точек дыр
Описание	проверка при правильном формате, но пересечении в точке
Начальное состояние	Файл zones.txt имеет следующее содержимое: "[0, 0], [10, 0], [5, 10] \n [0, 0], [-10, -20], [-5, -10]"
Входные данные	file_path="zones.txt"
Ожидаемый результат	Exception("Дыры пересекаются")

Тест Б59, негативный

Объект тестирования	Метод ReadBlockedZones
Цель	проверка считывания точек дыр
Описание	проверка при не правильном формате
Начальное состояние	Файл zones.txt имеет следующее содержимое: "[0, 0] [10, 0], [5, 10] \n [0, 0], [-10, -20], [-5, -10]"
Входные данные	file_path="zones.txt"
Ожидаемый результат	Exception("Не верный формат")

Тест Б60, негативный

Объект тестирования	Метод ReadBlockedZones
Цель	проверка считывания точек дыр
Описание	проверка при самопересечении
Начальное состояние	Файл zones.txt имеет следующее содержимое: "[0, 0], [10, 0], [5, 10], [5, -10]"
Входные данные	file_path="zones.txt"
Ожидаемый результат	Exception("Дыра имеет самопересечение")

Тест Б61, негативный

Объект тестирования	Метод ReadBlockedZones
Цель	проверка считывания точек дыр
Описание	проверка при вырожденности дыры
Начальное состояние	Файл zones.txt имеет следующее содержимое: "[0, 0], [10, 0], [5, 0]"
Входные данные	file_path="zones.txt"
Ожидаемый результат	Exception("Дыра вырождена")

Тест Б62, негативный

Объект тестирования	Метод ReadPlan
Цель	проверка считывания точек дыр
Описание	проверка при отсутствии файла
Начальное состояние	Файла zones.txt – нет
Входные данные	file_path="zones.txt"
Ожидаемый результат	Exception("Файла zones.txt не существует")

6.5.2 Метод ReadPlan

Тест Б63, позитивный

Объект тестирования	Метод ReadPlan
Цель	проверка считывания точек плана
Описание	проверка при верном формате
Начальное состояние	Файл plan.txt имеет следующее содержимое: "[0, 0], [10, 0], [10, 10], [0, 10]"
Входные данные	file_path="plan.txt"
Ожидаемый результат	[Point(0, 0), Point(10, 0), Point(10, 10), Point(0, 10)]

Тест Б64, негативный

Объект тестирования	Метод ReadPlan
Цель	проверка считывания точек плана
Описание	проверка при не верном формате
Начальное состояние	Файл plan.txt имеет следующее содержимое: "[0, 0], [10, 0], [10, 10], [0, 10] \n [-10, -10]"
Входные данные	file_path="plan.txt"
Ожидаемый результат	Exception('Не верный формат')

Тест Б65, негативный

Объект тестирования	Метод ReadPlan
Цель	проверка считывания точек плана
Описание	проверка при вырождении
Начальное состояние	Файл plan.txt имеет следующее содержимое: "[0, 0], [10, 0], [5,0]"
Входные данные	file_path="plan.txt"
Ожидаемый результат	Exception('План вырожден')

Тест Б66, негативный

Объект тестирования	Метод ReadPlan
Цель	проверка считывания точек плана
Описание	проверка при самопересечении
Начальное состояние	Файл plan.txt имеет следующее содержимое: "[0, 0], [10, 0], [5, 10], [5, -10]"
Входные данные	file_path="plan.txt"
Ожидаемый результат	Exception("План имеет самопересечение")

Тест Б67, негативный

Объект тестирования	Метод ReadPlan
Цель	проверка считывания точек плана
Описание	проверка при отсутствии файла
Начальное состояние	Файла plan.txt – нет
Входные данные	file_path="plan.txt"
Ожидаемый результат	Exception("Файла plan.txt не существует")

7 Интеграционное тестирование

7.1 Группа 1

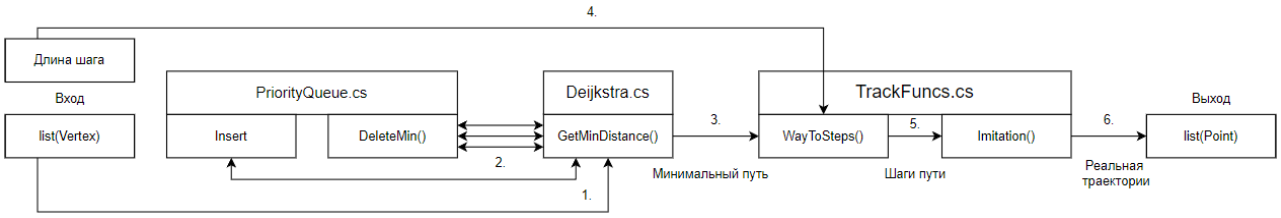
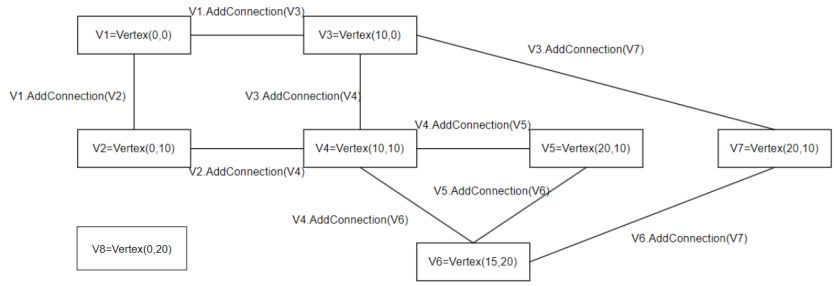


Рис. 4 – Схема интеграции 1 группа

Объект тестирования	Интеграция: группа 1
Цель	проверка группы методов, производящих генерацию реальной траектории
Сценарий выполнения для группы	

1. List<Vertex> V инициализирован по схеме выше.
2. Вызов метода GetMinDistance, передавая в него входные данные.
3. Записать возвращаемое значение в переменную List<Point> points.
4. Вызов метода WayToSteps(points, d=2).
5. Записать возвращаемое значение в переменную List<Point> steps.
6. List<Point> steps взят из предыдущего шага;вызов метода Imitation(points, e=15); записать возвращаемое значение в переменную List<Point> way.

Тест И1. Позитивный	
Описание	Проверка группы интеграции при связном графе.
Входные данные	V1, V7
Ожидаемый результат	Проверка осуществляется по алгоритму: • Рассмотреть все i от 1 до len(way)-1; • Для кажого вектора $\{way[i-1], way[i]\}$ и $\{way[i], way[i+1]\}$: • Проверить для всех i, что $\sum Atan2(\{way[i-1], way[i]\}, \{way[i], way[i+1]\}) < 15$, way[0] == Point(0,0), way[-1] == Point(20,10)
Тест И2. Негативный	
Описание	Проверка группы интеграции при отсутствии пути до вершины.
Входные данные	V1, V8
Ожидаемый результат	Exception('Нет пути до вершины')
Тест И3. Негативный	
Описание	Проверка группы интеграции при случаи совпадающих вершин.
Входные данные	V1, V1
Ожидаемый результат	Exception('Начальная и конечные вершины должны отличаться')
Тест И4. Граничный	
Описание	Проверка группы интеграции при нулевых коэффициентах ошибок.
Входные данные	V1, V7, e=0 в сценарии.
Ожидаемый результат	Проверить, что: все точки way принадлежат ломаной – Point(0,0), Point(10,0), Point(20,10), len(way) > 3, и way[0] == Point(0,0), way[-1] == Point(20,10)

Тест И5. Граничный	
Описание	Проверка группы интеграции при нулевых коэффициентах ошибок и шаге пути, превышающем расстояние между вершинами на графе.
Входные данные	V1, V7, d=100, e=0 в сценарии.
Ожидаемый результат	way = [Point(0,0), Point(10,0), Point(20,10)]

7.2 Группа 2

Рис. 5 – Схема интеграции 2 группа

Объект тестирования	Интеграция: группа 2
Цель	проверка группы методов, производящих генерацию зафиксированной траектории
Сценарий выполнения для группы	1. Задание входного массива точек List<Point> way и коэффициента ошибки e. 2. Вызов метода AddErrorIntoWay с передачей тестовых входных данных. 3. Записать возвращаемое значение в переменную List<Point> error_way.
Тест И6. Позитивный	
Описание	Проверка группы интеграции при случаи вершин на одной прямой.
Входные данные	way=[Point(0,0), Point(5,0), Point(10,0)], e=2
Ожидаемый результат	Массив error_way: len(error_way) == 3, |Atan2({error_way[0],error_way[1]},{error_way[1], error_way[2]}) - Atan2({way[0],way[1]},{way[1],way[2]})| <= 2/180*Pi
Тест И7. Позитивный	
Описание	Проверка группы интеграции при использовании выходных данных предыдущей группы.
Входные данные	Массив way, полученный в тесте №1 группы 1, e = 2
Ожидаемый результат	• Для всех i от 1 до len(error_way)-1: • Найти Atan({error_way[i-1], error_way[i]}, {error_way[i], error_way[i+1]}) • Проверить, что выражение по модулю < 2/180*Pi; error _way[0] == way[0], len(error_way) == len(way).

Тест И8. Граничный	
Описание	Проверка группы интеграции при двух точках.
Входные данные	Point(0,0), Point(5,0), e = 2
Ожидаемый результат	Массив error_way: len(error_way) == 2, |Atan2({way[0],error_way[1]}) - Atan2({way[0],way[1]},)| <= 2/180*Pi
Тест И9. Негативный	
Описание	Проверка группы интеграции при отрицательном коэффициенте ошибки.
Входные данные	Point(0,0), Point(5,0), e = -2
Ожидаемый результат	Exception('Максимальный угол ошибки должен быть положительным')

7.3 Группа 3

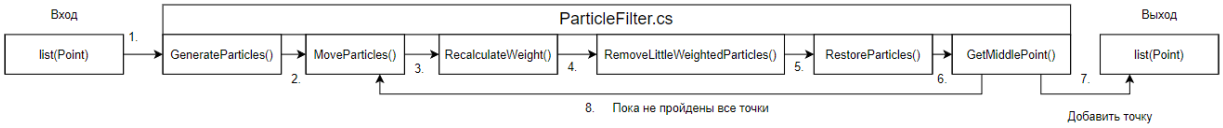
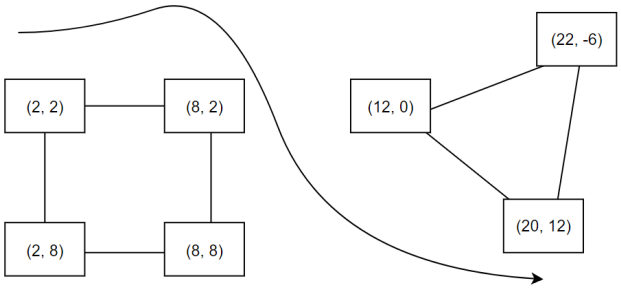


Рис. 6 – Схема интеграции 3 группа

Объект тестирования	Интеграция: группа 3
Цель	проверка последовательного применения методов алгоритма восстановления пути
Сценарий выполнения для группы	 1. Создать класс blockedZones(points, plan), где points = [p1, p2], где p1 – список из точек первого квадрата, p2 – второго; plan = [Point(-10,-10), Point(-10, 100), Point(100, 100), Point(100, -10)] 2. Создать класс ParticleFilter(blockedZones) 3. Вызов метода GenerateParticles(100, 20, 20) 4. Для всех i от 1 до len(error_way)-1: вызов метода MoveParticles(Atan(error_way[i], error_way[i+1]), len(error_way[i],error_way[i+1])), где error_way – входной массив. Результат в поле particles. 5. Вызов метода RecalculateWeight(error_way[i+1]); результат в поле particles. 6. RemoveLittleWeightedParticles(0.001); результат в поле particles. 7. GetMiddlePoint(); сохранить результат в список list<Point> track. 8. Вызвать RestoreParticles(GetMiddlePoint(), 100, 10)

Тест И10. Позитивный	
Описание	Проверка группы интеграции при пути, проходящем вдоль препятствия.
Входные данные	error_way = [Point(0,0), Point(4,0), Point(9,0), Point(10, 3), Point(12, 6)]
Ожидаемый результат	Список track содержит len(error_way) точек, при этом для любой точки point из track: point не принадлежит области blockedZones.Points и blockedZones.Plan.
Тест И11. Позитивный	
Описание	Проверка группы интеграции при пути, проходящем внутри препятствия у его границы.
Входные данные	error_way = [Point(0,0), Point(2,2.1), Point(4,3), Point(6.2, 2.5), Point(8, 1.5)]
Ожидаемый результат	Список track содержит len(error_way) точек, при этом для любой точки point из track: point не принадлежит области blockedZones.Points и blockedZones.Plan, при этом error_way[i].Y $<$ 2.
Тест И12. Позитивный	
Описание	Проверка группы интеграции при пути, проходящем внутри препятствия у его границы и выходящем из него.
Входные данные	error_way = [Point(0,0), Point(2,2.1), Point(4,3), Point(6.2, 2.5), Point(7.5, 3.5), Point(7.9, 6.5), Point(8.5, 8)]
Ожидаемый результат	Список track содержит len(error_way) точек, при этом для любой точки point из track: point не принадлежит области blockedZones.Points и blockedZones.Plan, при этом error_way[i].Y $<$ 2 либо error_way[i].X $>$ 8.

Тест И13. Негативный	
Описание	Проверка группы интеграции при пути, проходящем внутри препятствия без возможности обхода препятствия.
Входные данные	error_way = [Point(3,1.5), Point(3,3), Point(3,5), Point(3, 7)]
Ожидаемый результат	Exception('Не возможно построить исправленную траекторию').

8 Аттестационное тестирование

8.1 Требование «Выбор точек начала и конца»

Тест А1. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала».
Описание теста	Проверка правильности работы выбора точек. 1. Нажатие в окне отрисовки в область белого цвета внутри области ломаной.
Входные данные	–
Ожидаемый результат	В координате нажатия нарисуется круг красного цвета.

Тест А2. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала».
Описание теста	Проверка невозможности задания точки внутри сложного полигона. 1. Нажатие в окне отрисовки в область серого цвета внутри области ломаной.
Входные данные	–
Ожидаемый результат	В координате нажатия не произойдёт ничего.
Тест А3. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку конца».
Описание теста	Проверка правильности работы выбора точек. 1. Нажатие в окне отрисовки в область белого цвета внутри области ломаной.
Входные данные	–
Ожидаемый результат	В координате нажатия нарисуется круг синего цвета.

Тест А4. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку конца».
Описание теста	Проверка невозможности задания точки внутри сложного полигона. 1. Нажатие в окне отрисовки в область серого цвета внутри области ломаной.
Входные данные	–
Ожидаемый результат	В координате нажатия не произойдёт ничего.
Тест А5. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала».
Описание теста	Проверка невозможности задания точки вне плана. 1. Нажатие в окне отрисовки в область, вовне области ломаной.
Входные данные	–
Ожидаемый результат	В координате нажатия не произойдёт ничего.

Тест А6. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку конца».
Описание теста	Проверка невозможности задания точки вне плана. 1. Нажатие в окне отрисовки в область, вовне области ломаной.
Входные данные	–
Ожидаемый результат	В координате нажатия не произойдёт ничего.
Тест А7. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Установить точку начала»;
Описание теста	Проверка при отсутствии загруженного сложного полигона. 1. Нажатие в область отрисовки.
Входные данные	–
Ожидаемый результат	Выведется сообщение в виде диалогового окна об отсутствии загруженного полигона.

8.2 Требование «Очистка точек начала и конца траектории»

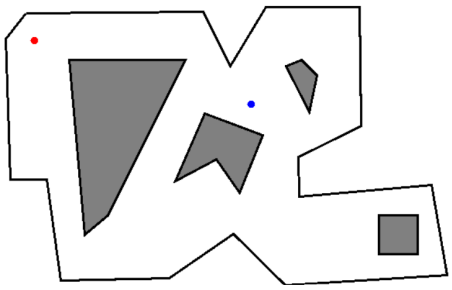
Тест А8. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала». 5. нажать в белой области внутри сложного полигона.
Описание теста	Проверка удаления одной точки – начала. 1. Нажатие кнопку «Очистить точки».
Входные данные	–
Ожидаемый результат	Из области сложного полигона удалится поставленная точка начала – красный круг.

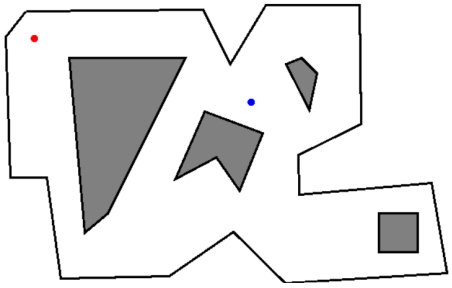
Тест А9. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку конца». 5. нажать в белой области внутри сложного полигона.
Описание теста	Проверка удаления одной точки – конца. 1. Нажатие кнопку «Очистить точки».
Входные данные	–
Ожидаемый результат	Из области сложного полигона удалится поставленная точка начала – синий круг.

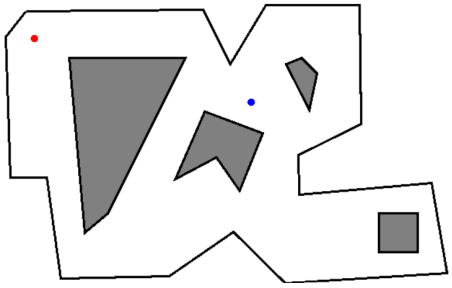
Тест А10. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку конца». 5. нажать в белой области внутри сложного полигона. 6. нажать кнопку «Установить точку начала». 7. нажать в белой области внутри сложного полигона.
Описание теста	Проверка удаления всех поставленных точек. 1. Нажатие кнопку «Очистить точки».
Входные данные	–
Ожидаемый результат	Из области сложного полигона удалятся все точки – исчезнет синий и красный круг.

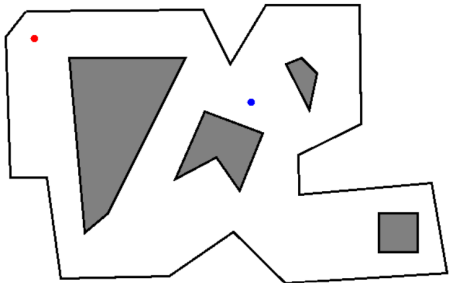
Тест А11. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»;
Описание теста	Проверка при отсутствии точек. 1. Нажатие кнопку «Очистить точки».
Входные данные	–
Ожидаемый результат	Область сложного полигона останется без изменений.
Тест А12. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Показать полигон»;
Описание теста	Проверка при отсутствии загруженного сложного полигона. 1. Нажатие кнопку «Очистить точки».
Входные данные	–
Ожидаемый результат	Выведется сообщение в виде диалогового окна об отсутствии загруженного полигона.

8.3 Требование «Задание коэффициента ошибки сымитированной траектории»

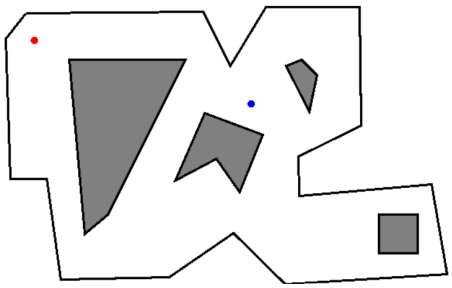
Тест А13. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной, в координатах (красная точка). 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной, в координатах (синяя точка). 
Описание теста	Проверка влияния коэффициента на траекторию. 1. В поле «Коэффициент ошибки» ввести число M. 2. В поле «Длина шага» ввести число N. 3. Нажать кнопку «Построить траекторию».
Входные данные	M=2, N=2
Ожидаемый результат	Построится 2 траектории, синяя будет соединять красную и синюю точки и лежать строго внутри сложного полигона, оранжевая будет отходить от синей.

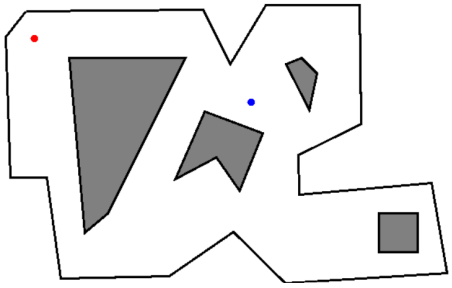
Тест А14. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной, в координатах (красная точка). 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной, в координатах (синяя точка). 
Описание теста	Проверка разных коэффициентов на траекториях. 1. В поле «Коэффициент ошибки» ввести число M_1. 2. В поле «Длина шага» ввести число N. 3. Нажать кнопку «Построить траекторию». 4. В поле «Коэффициент ошибки» ввести число M_2. 5. Нажать кнопку «Построить траекторию».
Входные данные	M_1=2, N=2, M_2=10
Ожидаемый результат	Построится 4 траектории: две синих будет соединять красную и синюю точки и лежать строго внутри сложного полигона, и 2 оранжевых, последняя из которых будет сильней отходить в сторону в сравнении с 1 оранжевой.

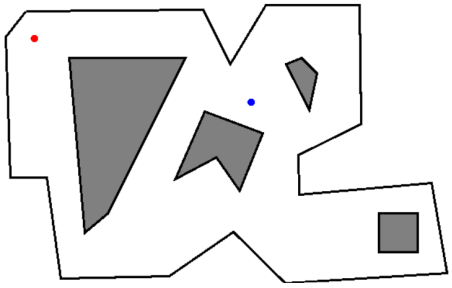
Тест А15. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной, в координатах (красная точка). 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной, в координатах (синяя точка). 
Описание теста	Проверка не верного коэффициента. 1. В поле «Коэффициент ошибки» ввести значение M.
Входные данные	M=-10
Ожидаемый результат	Откроется диалоговое окно с сообщением, что нельзя ввести отрицательный коэффициент.

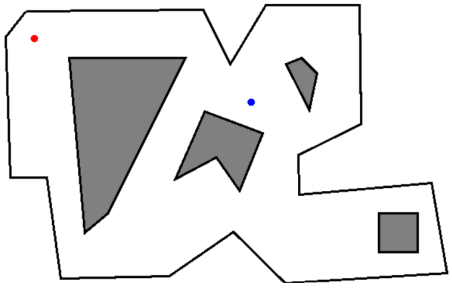
Тест А16. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной, в координатах (красная точка). 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной, в координатах (синяя точка). 
Описание теста	Проверка не верного коэффициента. 1. В поле «Коэффициент ошибки» ввести значение M.
Входные данные	M=91
Ожидаемый результат	Откроется диалоговое окно с сообщением, что нельзя ввести коэффициент больше чем 90.

8.4 Требование «Задание длины шага»

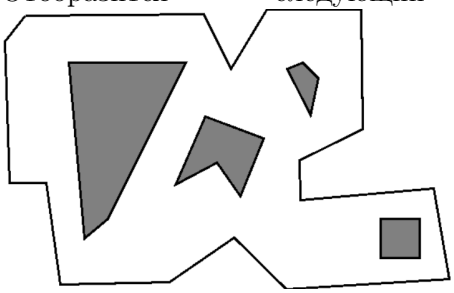
Тест А17. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной, в координатах (красная точка). 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной, в координатах (синяя точка). 
Описание теста	Проверка построения траектории с правильным шагом. 1. В поле «Коэффициент ошибки» ввести число M. 2. В поле «Длина шага» ввести число N. 3. Нажать кнопку «Построить траекторию».
Входные данные	M=2, N=2
Ожидаемый результат	Построится 2 траектории, синяя будет соединять красную и синюю точки и лежать строго внутри сложного полигона, оранжевая будет отходить от синей; число звеньев превосходит 2.

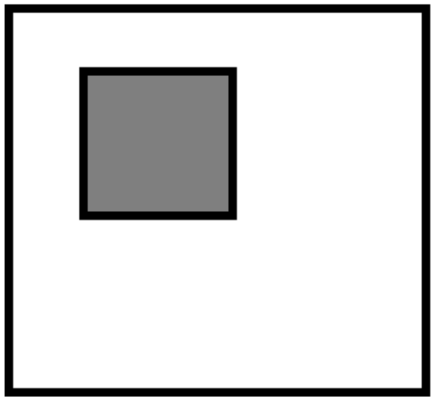
Тест А18. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной, в координатах (красная точка). 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной, в координатах (синяя точка). 
Описание теста	Проверка 2 разных коэффициентов на траекториях. 1. В поле «Коэффициент ошибки» ввести число M. 2. В поле «Длина шага» ввести число N_1. 3. Нажать кнопку «Построить траекторию». 4. В поле «Длина шага» ввести число N_2. 5. Нажать кнопку «Построить траекторию».
Входные данные	N_1=2, M=2, N_2=10
Ожидаемый результат	Построится 4 траектории: две синих будет соединять красную и синюю точки и лежать строго внутри сложного полигона, и 2 оранжевых. Количество звеньев в первой паре траекторий будет больше, чем во второй – т.е. длина звеньев будет меньше, чем во второй.

Тест А19. Граничный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной, в координатах (красная точка). 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной, в координатах (синяя точка). 
Описание теста	Проверка большого коэффициента на траектории. 1. В поле «Коэффициент ошибки» ввести число M. 2. В поле «Длина шага» ввести число N. 3. Нажать кнопку «Построить траекторию».
Входные данные	N=1000, M=2
Ожидаемый результат	Построится 2 траектории, синяя будет соединять красную и синюю точки и лежать строго внутри сложного полигона, оранжевая будет отходить от синей. При этом, число звеньев траектории будет равно 2.

Тест А20. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной, в координатах (красная точка). 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной, в координатах (синяя точка). 
Описание теста	Проверка не верного коэффициента. 1. В поле «Длина шага» ввести значение N.
Входные данные	N=-10
Ожидаемый результат	Откроется диалоговое окно с сообщением, что нельзя ввести отрицательную длину шага.

8.5 Требование «Считывание карты помещения из файла в определенном формате»

Тест А21. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1);
Описание теста	Проверка загрузки сложного полигона. 1. нажать кнопку «Показать полигон»;
Входные данные	–
Ожидаемый результат	Отобразится следующий сложный полигон: 

Тест А22. Граничный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл pol.txt;
Описание теста	Проверка загрузки сложного полигона с минимальным числом дыр. 1. нажать кнопку «Показать полигон»;
Входные данные	pol.txt содержит следующее: [10, 10], [60,10], [60,60], [10,60] \n [20,20], [30,20], [30,40], [20,40]
Ожидаемый результат	Отобразится следующий сложный полигон: 

Тест А23. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл pol.txt;
Описание теста	Проверка выдачи ошибки при самопересекающемся плане. 1. нажать кнопку «Показать полигон»;
Входные данные	pol.txt содержит следующее: [10, 10], [60,10], [60,60], [10,-60],\n
Ожидаемый результат	Отобразится ошибка, что файл содержит самопересекающийся план.
Тест А24. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл pol.txt;
Описание теста	Проверка выдачи ошибки при вырожденном плане. 1. нажать кнопку «Показать полигон»;
Входные данные	pol.txt содержит следующее: [10, 10], [60,10], [30,10] \n
Ожидаемый результат	Отобразится ошибка, что файл содержит вырожденный план.

Тест А25. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл pol.txt;
Описание теста	Проверка выдачи ошибки при самопересекающейся дыре. 1. нажать кнопку «Показать полигон»;
Входные данные	pol.txt содержит следующее: [10, 10], [60,10], [60,60], [10, 60],\n [20, 20], [30, 30], [20,30], [30, 20]
Ожидаемый результат	Отобразится ошибка, что файл содержит самопересекающуюся дыру.
Тест А26. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл pol.txt;
Описание теста	Проверка выдачи ошибки при вырожденной дыре. 1. нажать кнопку «Показать полигон»;
Входные данные	pol.txt содержит следующее: [10, 10], [60,10], [60,60], [10, 60],\n [20, 20], [30, 30], [10,10]
Ожидаемый результат	Отобразится ошибка, что файл содержит вырожденную дыру.

Тест А27. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл pol.txt;
Описание теста	Проверка выдачи ошибки при не верном формате. 1. нажать кнопку «Показать полигон»;
Входные данные	pol.txt содержит следующее: 10, 10, [60,10], [60,60], [10, 60],\n [20, 20], [30, 30], [20,30], [30, 20]
Ожидаемый результат	Отобразится ошибка, что файл содержит не верный формат данных.
Тест А28. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл pol.txt;
Описание теста	Проверка выдачи ошибки при не верных данных. 1. нажать кнопку «Показать полигон»;
Входные данные	pol.txt содержит следующее: [10f, 10d], [60,10], [60,60], [10, 60],\n [20, 20], [30, 30], [10,10]
Ожидаемый результат	Отобразится ошибка, что файл содержит не верный формат данных.

8.6 Требование «Построение новой реальной траектории между заданными точками, не проходящей через дыры»

Тест А29. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной. 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной. 6. В поле «Коэффициент ошибки» ввести число 2. 7. В поле «Длина шага» ввести число 2.
Описание теста	Проверка построения разных реальных траекторий. 1. Нажать кнопку «Построить траекторию». 2. Ещё раз нажать кнопку «Построить траекторию».
Входные данные	–
Ожидаемый результат	Построится 2 траектории синего цвета, соединяющих красную и синюю точки и лежащих строго внутри сложного полигона, при этом эти траектории будут отличаться; а также будут построены 2 траектории оранжевого цвета, отходящих от соответствующей траектории синего цвета.

Тест А30. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. В поле «Коэффициент ошибки» ввести число 2. 3. В поле «Длина шага» ввести число 2.
Описание теста	Проверка выдачи ошибки при отсутствии загруженного сложного полигона. 1. Нажать кнопку «Построить траекторию».
Входные данные	–
Ожидаемый результат	Отобразится открыться диалоговое окно с сообщением, что нельзя построить траектории, если не загружен сложный полигон.
Тест А31. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной. 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной.
Описание теста	Проверка выдачи ошибки при не задании коэффициентов траектории. 1. Нажать кнопку «Построить траекторию».
Входные данные	–
Ожидаемый результат	Отобразится ошибка, что нельзя построить траектории, если не введены коэффициенты траектории.

Тест А32. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»;
Описание теста	Проверка выдачи ошибки при не задании начальной и конечной точек траектории. 1. Нажать кнопку «Построить траекторию».
Входные данные	–
Ожидаемый результат	Отобразится ошибка, что не заданы точки траектории.

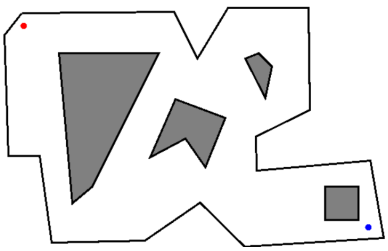
8.7 Требование «Построение исправленной траектории на основе алгоритма»

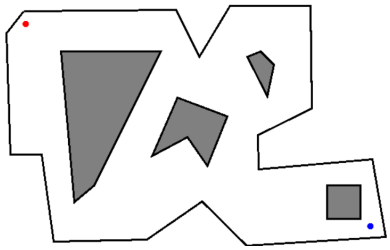
Тест А33. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной. 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной. 6. В поле «Коэффициент ошибки» ввести число 2; в поле «Длина шага» ввести число 2. 7. Нажать кнопку «Построить траекторию».
Описание теста	Проверка построения исправленной траекторий. 1. Нажать кнопку «Фильтр частиц».
Входные данные	–
Ожидаемый результат	Построится траектория, состоящая из красных точек, которая будет проходить ближе к траектории синего цвета по сравнению с траекторией оранжевого цвета.

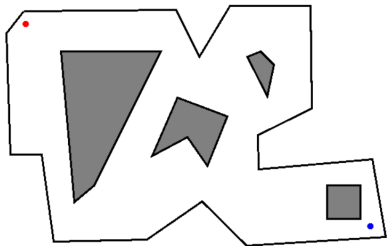
Тест А34. Позитивный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. нажать кнопку «Установить точку начала» и нажать в белой области внутри области, ограниченной ломаной. 5. нажать кнопку «Установить точку конца» и нажать в белой области внутри области, ограниченной ломаной. 6. В поле «Коэффициент ошибки» ввести число 2; в поле «Длина шага» ввести число 2. 7. Нажать кнопку «Построить траекторию».
Описание теста	Проверка построения отличающихся траекторий. 1. Нажать кнопку «Фильтр частиц». 2. Нажать кнопку «Построить траекторию». 3. Ещё раз нажать кнопку «Фильтр частиц».
Входные данные	–
Ожидаемый результат	Построится 2 траектории, состоящих из красных точек, которая будет проходить ближе к своей траектории синего цвета, но они будут отличаться.

Тест А35. Негативный	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»;
Описание теста	Проверка выдачи ошибки при отсутствии зафиксированной траектории. 1. Нажать кнопку «Фильтр частиц».
Входные данные	–
Ожидаемый результат	Отобразится ошибка, что не построена реальная и соответствующая ей зафиксированная траектория.

9 Нагрузочное тестирование

Тест Н1.	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. выбрать начальную и конечные точки в соответствии с рисунком ниже. 5. Задать длину шага равной 10. 6. Задать коэффициент ошибки, равный 2. 
Описание теста	Проверка времени выполнения алгоритма на 100 точках. 1. Нажать кнопку «Фильтр частиц».
Входные данные	–
Ожидаемый результат	Траектория построится максимум за 700 миллисекунд.

Тест Н2.	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. выбрать начальную и конечные точки в соответствии с рисунком ниже. 5. Задать длину шага равной 1. 6. Задать коэффициент ошибки, равный 2. 
Описание теста	Проверка выполнения алгоритма на 1000 точках. 1. Нажать кнопку «Фильтр частиц».
Входные данные	–
Ожидаемый результат	Траектория построится максимум за 2600 миллисекунд.

Тест Н3.	
Начальное состояние	1. Запустить приложение системы; 2. нажать кнопку «Загрузить» и выбрать файл sample_polygon.txt (ссылка на файл в пункте 1.1); 3. нажать кнопку «Показать полигон»; 4. выбрать начальную и конечные точки в соответствии с рисунком ниже. 5. Задать длину шага равной 1. 6. Задать коэффициент ошибки, равный 2. 
Описание теста	Проверка выполнения алгоритма на 10000 точках. 1. Нажать кнопку «Фильтр частиц».
Входные данные	–
Ожидаемый результат	Траектория построится максимум за 25.8 секунд.

10 Журнал тестирования

Журнал тестирования содержит таблицу, информирующую о ходе произведения тестов, описанных в главах «Блочное тестирование», «Нагрузочное тестирование», «Аттестационное тестирование» и «Интеграционное тестирование».

Выделенный красным цветом текст в столбце «Итог» – ссылка на ошибку в «Журнале ошибки». Также в столбце «№» в «Журнале ошибок» имеется ссылка на само содержание теста из «Плана тестирования». (Обратите внимание, обратно из «Плана тестирования нет ссылки).

10.1 Блочное тестирование

10.1.1 Класс PriorityQueue

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод Insert					
Б1	1) 25.11.2023	Проверка того, что пустая очередь вставит элемент в вершину.	Поле Queue класса содержит массив из одного элемента – (1,Vertex(0,0)).	Поле Queue класса содержит массив из одного элемента – (1,Vertex(0,0)).	Пройден
Б2	1) 25.11.2023	Проверка того, что при наличии в очереди элемента в вершине, новый больший элемент получит индекс 1.	Поле Queue класса содержит массив (1,Vertex(0,0)), (2,Vertex(1,1))	Поле Queue класса содержит массив (1,Vertex(0,0)), (2,Vertex(1,1))	Пройден
Б3	1) 25.11.2023	Проверка того, что при наличии в очереди левой ветки двоичной кучи, новый элемент вставится в правую.	Поле Queue класса содержит массив из двух элементов, на индексе 1 стоит значение (1, Vertex(-1,-1)).	Поле Queue класса содержит массив из двух элементов, на индексе 1 стоит значение (1, Vertex(-1,-1))	Пройден
Б4	1) 25.11.2023	Проверка того, переданная пара с значением, которое имеется в очереди, вставится после элемента с данным значением в очереди.	1) Поле Queue класса содержит массив из четырёх элементов, на индексе 0 стоит значение с ключом 1, на индексе 1 – стоит ключ 1, на индексе 2 – ключ 3, а на индексе 3 – ключ 2.	Поле Queue класса содержит массив из четырёх элементов, на индексе 0 стоит значение с ключом 1, на индексе 1 – стоит ключ 1, на индексе 2 – ключ 3, а на индексе 3 – ключ 2.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод DeleteMin					
Б5	1) 26.11.2023	Проверка того, извлечётся элемент в вершине.	Поле Queue класса содержит массив (2,Vertex(0,2)) (3, Vertex(0,3)), возвращается элемент (1, Vertex(0,0)).	Поле Queue класса содержит массив (2,Vertex(0,2)) (3, Vertex(0,3)), возвращается элемент (1, Vertex(0,0))	Пройден
Б6	1) 26.11.2023	Проверка при одинаковых ключах.	Поле Queue класса содержит массив (1,Vertex(0,2)) (1, Vertex(0,3)), возвращается элемент (1, Vertex(0,0)).	Поле Queue класса содержит массив (1,Vertex(0,2)) (1, Vertex(0,3)), возвращается элемент (1, Vertex(0,0)).	Пройден
Б7	1) 26.11.2023	Проверка возврата при пустой очереди	Exception('Очередь пуста').	Exception('Argument OutOfRange Exception')	Не пройден.
Б7	2) 27.11.2023	Проверка при пустой очереди	Exception('Очередь пуста').	Exception('Очередь пуста').	Пройден

10.1.2 Класс Deijkstra

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод GetMinDistance					
Б8	1) 26.11.2023	Проверка простого случая, когда вершина кратчайший путь – в соседней вершине.	Список [V1, V2].	Список [V1, V2].	Пройден
Б9	1) 26.11.2023	Проверка случая, когда кратчайший путь в соседней вершине, но кратчайшее дерево не идёт в неё на 1 шаге.	Список [V1, V3, V4].	Список [V1, V3, V4].	Пройден
Б10	1) 26.11.2023	Проверка случая, нет пути в конечную вершину	Exception('Нет пути между вершинами').	Exception ('Превышено время ожидания выполнения теста "TestMethod3".')	Не пройден
Б11	1) 26.11.2023	Проверка случая с 1 вершиной	Список [V1].	Список двух вершин [V1, V1]	Не пройден
Б12	1) 27.11.2023	Проверка случая с двумя путями одинаковой длины	Список [V1, V2, V4].	Список [V1, V2, V4].	Пройден
Б10	2) 27.11.2023	Проверка случая, нет пути в конечную вершину	Exception('Нет пути между вершинами').	Exception('Нет пути между вершинами').	Пройден
Б11	2) 27.11.2023	Проверка случая с 1 вершиной	Список [V1].	Список [V1].	Пройден

10.1.3 Класс TrackFuncs

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод WayToSteps					
Б13	1) 26.11.2023	Проверка случая для отрезка	Список точек [Point(0,0), V1=Point(3,0), V2=Point(6,0), V3=Point(9,0), Point(10,0)].	Список точек [Point(0,0), V1=Point(3,0), V2=Point(6,0), V3=Point(9,0), Point(10,0)].	Пройден
Б14	1) 26.11.2023	Проверка случая для ломаной	Список точек [Point(0,0), V1=Point(3,0), V2=Point(6,0), V3=Point(9,0), Point(10,0), V4=Point(10, 3), V5=Point(10, 6), V6=Point(10, 9), Point(10,10)]	Список точек [Point(0,0), V1=Point(3,0), V2=Point(6,0), V3=Point(9,0), Point(10,0), V4=Point(10, 3), V5=Point(10, 6), V6=Point(10, 9), Point(10,10)]	Пройден
Б15	1) 26.11.2023	Проверка случая, когда шаг длиннее отрезка ломаной	Список точек [Point(0,0), Point(10,0), V1=Point(10,11), Point(10,20)]	Список точек [Point(0,0), Point(10,0), V1=Point(10,11), Point(10,20)]	Пройден
Б16	1) 26.11.2023	Проверка случая с 1 точкой	Exception('Нужно минимум 2 точки')	список [Point(0,0), Point(0,0)]	Не пройден
Б17	1) 26.11.2023	Проверка случая с совпадающей точкой	[Point(0,0), Point(0,0)]	[Point(0,0), Point(0,0)]	Пройден
Б16	2) 27.11.2023	Проверка случая с 1 точкой	Exception('Нужно минимум 2 точки')	Exception('Нужно минимум 2 точки')	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод Imitation					
Б18	1) 28.11.2023	Проверка случая с 3 точками	Массив way: way[0] == Point(0,0), way[-1] == Point(10,0), len(way) == 3, 0 < |Atan2({p[0],p[1]}, {p[1],p[2]}) - Atan2({way[0],way[1]}, {way[1],way[2]})| <= 2/180*Pi.	way[0] == Point(0,0), way[-1] == Point(10,0), len(way) == 3, 0 < |Atan2({p[0],p[1]}, {p[1],p[2]}) - Atan2({way[0],way[1]}, {way[1],way[2]})| <= 2/180*Pi.	Пройден
Б19	1) 28.11.2023	Проверка случая с 10 точками	Массив way: way[0] == Point(0,0), way[-1] == Point(10,0), len(way) == 10, для всех i от 1 до 8: 0 < |Atan2({p[i-1],p[i]},{p[i],p[i+1]}) - Atan2({way[i-1],way[i]}, {way[i],way[i+1]})| <= 2/180*Pi	way[0] == Point(0,0), way[-1] == Point(10,0), len(way) == 10, для всех i от 1 до 8: 0 < |Atan2({p[i-1],p[i]},{p[i],p[i+1]}) - Atan2({way[i-1],way[i]}, {way[i],way[i+1]})| <= 2/180*Pi	Пройден
Б20	1) 28.11.2023	Проверка случая с 2 точками	Exception('Для имитации необходимо минимум 3 точки')	Список [Point(0,0)]	Не пройден
Б21	1) 28.11.2023	Проверка случая с 3 точками, 2 из которых совпадают	Exception('Для имитации траектории не должны иди подряд совпадающие точки')	Exception('Divide ByZero Exception')	Не пройден
Б22	1) 28.11.2023	Проверка случая с отрицательным максимальным углом поворота	Exception ('Максимальный угол поворота должен быть положительным')	Exception ('Максимальный угол поворота должен быть положительным')	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод Imitation					
Б20	2) 29.11.2023	Проверка случая с 2 точками	Exception('Для имитации необходимо минимум 3 точки')	Exception('Для имитации необходимо минимум 3 точки')	Пройден
Б21	2) 29.11.2023	Проверка случая с 3 точками, 2 из которых совпадают	Exception('Для имитации траектории не должны иди подряд совпадающие точки')	Exception('Для имитации траектории не должны иди подряд совпадающие точки')	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод AddErrorIntoWay					
Б23	1) 28.11.2023	Проверка случая с 3 точками	Массив error_way: , len(error_way) == 3, |Atan2({error_way[0], error_way[1]}, {error_way[1], error_way[2]}) - Atan2({way[0], way[1]}, {way[1], way[2]})| <= 2/180*Pi	len(error_way) == 3, |Atan2({error_way[0], error_way[1]}, {error_way[1], error_way[2]}) - Atan2({way[0], way[1]}, {way[1], way[2]})| <= 2/180*Pi	Пройден
Б24	1) 28.11.2023	Проверка случая с отрицательным максимальным углом ошибки	Exception('Максимальный угол ошибки должен быть положительным')	Exception('Максимальный угол ошибки должен быть положительным')	Пройден
Б25	1) 28.11.2023	Проверка случая с двумя точками	Массив error_way: |Atan2({way[0], error_way[1]}) - Atan2({way[0], way[1]},)| <= 2/180*Pi	|Atan2({way[0], error_way[1]}) - Atan2({way[0], way[1]},)| <= 2/180*Pi	Пройден

10.1.4 Класс ParticleFilter

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод GenerateParticle					
Б26	1) 28.11.2023	Проверка простого случая на 100 частицах	Поле particle содержит 100 элементов класса Particle, координаты Particle_i: 0 < Particle_i.Point.X < 100 и 0 < Particle_i.Point.Y < 100, Particle.Weight = 0.01	Поле particle содержит 100 элементов класса Particle, координаты Particle_i: 0 < Particle_i.Point.X < 100 и 0 < Particle_i.Point.Y < 100, Particle.Weight = 0.01	Пройден
Б27	1) 28.11.2023	проверка правильности вычисления веса при 1 частице	Поле particle содержит 1 элемент класса Particle, координаты Particle: 0 < Particle.Point.X < 1 и 0 < Particle.Point.Y < 1, Particle.Weight = 1	Поле particle содержит 1 элемент класса Particle, координаты Particle: 0 < Particle.Point.X < 1 и 0 < Particle.Point.Y < 1, Particle.Weight = 1	Пройден
Б28	1) 28.11.2023	Проверка ошибки при отрицательном количестве частиц	Exception(' Количество частиц должно быть не меньше 1')	Exception(' Количество частиц должно быть не меньше 1')	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод GenerateParticle					
Б29	1) 28.11.2023	Проверка случая при координатах равных 0	Поле particle содержит 10 элементов класса Particle, координаты Particle_i: Particle_i.Point.X = 0 и Particle_i.Point.Y = 0, Particle.Weight = 0.1	Поле particle не содержит элементов	Не пройден
Б30	1) 28.11.2023	Проверка случая при max_x < start_x и max_y < start_y	Поле particle содержит 10 элементов класса Particle, координаты Particle_i: 0 < Particle_i.Point.X < 10 и 0 < Particle_i.Point.Y < 10, Particle.Weight = 0.1	Exception('Argument Out Of Range Exception')	Не пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод GenerateParticle					
Б29	2) 29.11.2023	Проверка случая при координатах равных 0	Поле particle содержит 10 элементов класса Particle, координаты Particle_i: Particle_i.Point.X = 0 и Particle_i.Point.Y = 0, Particle.Weight = 0.1	Поле particle содержит 10 элементов класса Particle, координаты Particle_i: Particle_i.Point.X = 0 и Particle_i.Point.Y = 0, Particle.Weight = 0.1	Пройден
Б30	2) 29.11.2023	Проверка случая при max_x < start_x и max_y < start_y	Поле particle содержит 10 элементов класса Particle, координаты Particle_i: 0 < Particle_i.Point.X < 10 и 0 < Particle_i.Point.Y < 10, Particle.Weight = 0.1	Поле particle содержит 10 элементов класса Particle, координаты Particle_i: 0 < Particle_i.Point.X < 10 и 0 < Particle_i.Point.Y < 10, Particle.Weight = 0.1	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод MoveParticles					
Б31	1) 29.11.2023	Проверка простого движения без поворота	Поле particle содержит 1 элемент класса Particle, координаты Particle: Particle_0.Point.X=10 и Particle_0.Point.Y =0, Particle.Weight =1, Particle.Orientation =0	Поле particle содержит 1 элемент класса Particle, координаты Particle: Particle_0.Point.X=10 и Particle_0.Point.Y =0, Particle.Weight =1, Particle.Orientation =0	Пройден
Б32	1) 29.11.2023	Проверка простого движения без поворота на 2 частицах	Поле particle содержит 2 элемента класса Particle, координаты Particle: Particle_0.Point.X =10 и Particle_0.Point.Y =0, Particle.Weight =0.5, Particle.Orientation =0; Particle_1.Point.X =11 и Particle_1.Point.Y =1, Particle.Weight =0.5, Particle.Orientation =0	Поле particle содержит 2 элемента класса Particle, координаты Particle: Particle_0.Point.X =10 и Particle_0.Point.Y =0, Particle.Weight =0.5, Particle.Orientation =0; Particle_1.Point.X =11 и Particle_1.Point.Y =1, Particle.Weight =0.5, Particle.Orientation =0	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод MoveParticles					
Б33	1) 29.11.2023	Проверка простого движения с поворотом	Поле particle содержит 1 элемент класса Particle, координаты Particle: Particle_0.Point.X =0 и Particle_0.Point.Y =10, Particle_0.Weight =1, Particle_0.Orientation =0.5*Pi	Поле particle содержит 1 элемент класса Particle, координаты Particle: Particle_0.Point.X =0 и Particle_0.Point.Y =10, Particle_0.Weight =1, Particle_0.Orientation =0.5*Pi	Пройден
Б34	1) 29.11.2023	Проверка простого движения с поворотом 2 частиц	Поле particle содержит 2 элемента класса Particle, координаты Particle: Particle0.Point.X = 0 и Particle0.Point.Y = 10, Particle.Weight = 0.5, Particle.Orientation = 0.5*Pi; Particle1.Point.X = 0 и Particle1.Point.Y = -10, Particle.Weight = 0.5, Particle.Orientation = 1.5*Pi	Поле particle содержит 2 элемента класса Particle, координаты Particle: Particle0.Point.X = 0 и Particle0.Point.Y = 10, Particle.Weight = 0.5, Particle.Orientation = 0.5*Pi; Particle1.Point.X = 0 и Particle1.Point.Y = -10, Particle.Weight = 0.5, Particle.Orientation = 1.5*Pi	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод MoveParticles					
Б35	1) 29.11.2023	Проверка движения при повороте на большой угол	Поле particle содержит 1 элемент класса Particle, координаты Particle: Particle_0.Point.X = 0 и Particle_0.Point.Y = 10, Particle.Weight = 1, Particle.Orientation = 0.5*Pi	Поле particle содержит 1 элемент класса Particle, Particle.Orientation = 98.5*Pi	Не пройден
Б36	1) 29.11.2023	Проверка движения при негативном расстоянии	Поле particle содержит 1 элемент класса Particle, координаты Particle: Particle_0.Point.X = -10 и Particle_0.Point.Y = 0, Particle.Weight = 1, Particle.Orientation = 0;	Поле particle содержит 1 элемент класса Particle, координаты Particle: Particle_0.Point.X = -10 и Particle_0.Point.Y = 0, Particle.Weight = 1, Particle.Orientation = 0;	Пройден
Б35	2) 30.11.2023	Проверка движения при повороте на большой угол	Поле particle содержит 1 элемент класса Particle, координаты Particle: Particle_0.Point.X = 0 и Particle_0.Point.Y = 10, Particle.Weight = 1, Particle.Orientation = 0.5*Pi	Поле particle содержит 1 элемент класса Particle, координаты Particle: Particle_0.Point.X = 0 и Particle_0.Point.Y = 10, Particle.Weight = 1, Particle.Orientation = 0.5*Pi	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод RemoveLittleWeightedParticles					
Б37	1) 29.11.2023	Проверка удаления при параметре ∈ (0,1)	Поле particle содержит [Particle(Point(0,0), 0, 0.7)]	Поле particle содержит [Particle(Point(0,0), 0, 0.7)]	Пройден
Б38	1) 29.11.2023	Проверка оставления частиц при параметре, равном 0.	Поле particle содержит [Particle(Point(0,0), 0, 0.1), Particle(Point(0,0), 0, 0.2), Particle(Point(0,0), 0, 0.7)	Поле particle содержит [Particle(Point(0,0), 0, 0.1), Particle(Point(0,0), 0, 0.2), Particle(Point(0,0), 0, 0.7)	Пройден
Б39	1) 29.11.2023	Проверка оставления частиц при отрицательном параметре	Exception('Вес должен принадлежать отрезку [0,1]')	Exception('Вес должен принадлежать отрезку [0,1]')	Пройден
Б40	1) 29.11.2023	Проверка удаления всех частиц при большом параметре	Поле particle содержит []	Поле particle содержит []	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод GetMiddlePoint					
Б41	1) 29.11.2023	Проверка нахождения центральной точки при 1 частице	Возвращаемое значение Point(0,0)	Возвращаемое значение Point(0,0)	Пройден
Б42	1) 29.11.2023	Проверка нахождения центральной точки при 2 частицах	Возвращаемое значение Point(5,0)	Возвращаемое значение Point(5,0)	Пройден
Б43	1) 29.11.2023	Проверка нахождения центральной точки при 3 частицах, 2 из которых одинаковы	Возвращаемое значение Point(3.(3),0)	Возвращаемое значение Point(3.(3),0)	Пройден
Б44	1) 29.11.2023	Проверка исключения при отсутствии частиц	Exception('Нет частиц для нахождения центральной точки')	Exception('Divide By Zero Exception')	Не пройден
Б44	2) 30.11.2023	Проверка исключения при отсутствии частиц	Exception('Нет частиц для нахождения центральной точки')	Exception('Нет частиц для нахождения центральной точки')	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод RestoreParticle					
Б45	1) 29.11.2023	Проверка при отсутствии частиц в поле particle	Поле particle содержит 10 элементов класса Particle: Particle_i.Point.X2 + Particle_i.Point.Y2 ∈ [-10, 10]	Поле particle содержит 10 элементов класса Particle: Particle_i.Point.X2 + Particle_i.Point.Y2 ∈ [-10, 10]	Пройден
Б46	1) 29.11.2023	Проверка наличии частиц в поле particle	Поле particle содержит 2 элемента класса Particle: [Particle(Point(0,0), 0, 1), Particle1.Point.X^2 + Particlei.Point.Y^2 ∈ [-2, 2]	Поле particle содержит 2 элемента класса Particle: [Particle(Point(0,0), 0, 1), Particle1.Point.X^2 + Particlei.Point.Y^2 ∈ [-2, 2]	Пройден
Б47	1) 29.11.2023	Проверка при наличии большего числа частиц в поле particle	Поле particle содержит [Particle(Point(0,0), 0, 1), Particle(Point(0,0), 0, 1), Particle(Point(0,0), 0, 1)]	Поле particle содержит [Particle(Point(0,0), 0, 1), Particle(Point(0,0), 0, 1), Particle(Point(0,0), 0, 1)]	Пройден
Б48	1) 29.11.2023	Проверка при отрицательном параметре количества	Exception('Число восстанавливаемых частиц должно быть положительным')	Поле particles не изменилось	Не пройден
Б49	1) 29.11.2023	Проверка при отрицательном радиусе	Exception('Радиус не может быть отрицательным')	Поле particles не изменилось	Не пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод RestoreParticle					
Б48	2) 30.11.2023	Проверка при отрицательном параметре количества	Exception('Число восстанавливаемых частиц должно быть положительным')	Exception('Число восстанавливаемых частиц должно быть положительным')	Пройден
Б49	2) 30.11.2023	Проверка при отрицательном радиусе	Exception('Радиус не может быть отрицательным')	Exception('Радиус не может быть отрицательным')	Пройден
Метод RecalculateWeight					
Б50	1) 21.12.2023	проверка при частице при расстоянии от ориентира до неё менее 40	Поле particle содержит [Particle(Point(0,0), 0, 1)]	Поле particle содержит [Particle(Point(0,0), 0, 1)]	Пройден
Б51	1) 21.12.2023	проверка при двух частицах при расстоянии от ориентира до неё менее 40, одна ближе к ориентиру в два раза, чем другая	Поле particle содержит [Particle(Point(5,0), 0, 0.53), Particle(Point(10,0), 0, 0.47)]	Поле particle содержит [Particle(Point(5,0), 0, 0.53), Particle(Point(10,0), 0, 0.47)]	Пройден
Б52	1) 21.12.2023	проверка при двух частицах при расстоянии от ориентира до неё менее 40, одна совпадает с ориентиром	Поле particle содержит [Particle(Point(5,0), 0, 0.57), Particle(Point(10,0), 0, 0.43)]	Поле particle содержит [Particle(Point(5,0), 0, 0.57), Particle(Point(10,0), 0, 0.43)]	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод RecalculateWeight					
Б53	1) 22.12.2023	проверка при двух частицах при выходе частицы за границу от ориентира	Поле particle содержит [Particle(Point(5,0), 0, 1), Particle(Point(10,0), 0, 0)]	Поле particle содержит [Particle(Point(5,0), 0, 1), Particle(Point(10,0), 0, 0)]	Пройден
Б54	1) 21.12.2023	проверка при двух частицах при выходе всех частицы за границу от ориентира	Поле particle содержит [Particle(Point(5,0), 0, 1), Particle(Point(10,0), 0, 0)]	Поле particle содержит [Particle(Point(5,0), 0, 1), Particle(Point(10,0), 0, 0)]	Пройден
Б55	1) 21.12.2023	проверка возврата исключения при отсутствии частиц	Exception('Частицы отсутствуют')	Exception('Index Out Of Range Exception')	Не пройден
Б55	2) 22.12.2023	проверка возврата исключения при отсутствии частиц	Exception('Частицы отсутствуют')	Exception('Частицы отсутствуют')	Пройден

10.1.5 Класс PointsReader

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод ReadBlockedZones					
Б56	1) 30.11.2023	Проверка при правильном формате	Список списков точек [[Point(0, 0), Point(10, 0), Point(5, 10)],[Point(10, 10), Point(20, 20), Point(15, 15)]]	Список списков точек [[Point(0, 0), Point(10, 0), Point(5, 10)],[Point(10, 10), Point(20, 20), Point(15, 15)]]	Пройден
Б57	1) 30.11.2023	проверка при правильном формате, но пересечении	Exception("Дыры пересекаются")	Exception("Дыры пересекаются")	Пройден
Б58	1) 30.11.2023	проверка при правильном формате, но пересечении в точке	Exception("Дыры пересекаются")	Список [[[0, 0] [10, 0], [5, 10]],[[0, 0], [-10, -20], [-5, -10]]]	Не пройден
Б59	1) 30.11.2023	Проверка при не правильном формате	Exception("Не верный формат")	Exception("Не верный формат")	Пройден
Б60	1) 30.11.2023	Проверка при самопересечении	Exception("Дыра имеет самопересечение")	Exception("Дыра имеет самопересечение")	Пройден
Б61	1) 30.11.2023	проверка при вырожденности дыры	Exception("Дыра вырождена")	Список [[0, 0] [10, 0], [5, 0]]	Не пройден
Б62	1) 30.11.2023	Проверка при отсутствии файла	Exception("Файла zones.txt не существует")	Exception("Файла zones.txt не существует")	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод ReadBlockedZones					
Б58	2) 30.11.2023	Проверка при правильном формате, но пересечении в точке	Exception("Дыры пересекаются")	Exception("Дыры пересекаются")	Пройден
Б61	2) 30.11.2023	Проверка при вырожденности дыры	Exception("Дыра вырождена")	Exception("Дыра вырождена")	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Метод ReadPlan					
Б63	1) 30.11.2023	Проверка при верном формате	Список точек [Point(0, 0), Point(10, 0), Point(10, 10), Point(0, 10)]	Список точек [Point(0, 0), Point(10, 0), Point(10, 10), Point(0, 10)]	Пройден
Б64	1) 30.11.2023	Проверка при не верном формате	Exception('Не верный формат')	Exception('Не верный формат')	Пройден
Б65	1) 30.11.2023	Проверка при вырождении плана	Exception('План вырожден')	Список [[0, 0] [10, 0], [5, 0]]	Не пройден
Б66	1) 30.11.2023	Проверка при самопересечении	Exception('План имеет самопересечение')	Exception('План имеет самопересечение')	Пройден
Б67	1) 30.11.2023	Проверка при отсутствии файла	Exception("Файла plan.txt не существует")	Exception("Файла plan.txt не существует")	Пройден
Б65	2) 01.12.2023	Проверка при вырождении плана	Exception('План вырожден')	Exception('План вырожден')	Пройден

10.2 Интеграционное тестирование

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Группа 1					
И1	1) 1.11.2023	Проверка группы интеграции при связном графе.	Список точек way, такой что: Рассмотреть все i от 1 до len(way)-1; Для кажого вектора $\{way[i-1], way[i]\}$ и $\{way[i], way[i+1]\}$: Проверить для всех i, что $\sum Atan2(\{way[i-1], way[i]\}, \{way[i], way[i+1]\}) < 15$, way[0] == Point(0,0), way[-1] == Point(20,10)	Список точек way, такой что: Рассмотрены все i от 1 до len(way)-1; Для кажого вектора $\{way[i-1], way[i]\}$ и $\{way[i], way[i+1]\}$: Выполнено $\sum Atan2(\{way[i-1], way[i]\}, \{way[i], way[i+1]\}) < 15$, way[0] == Point(0,0), way[-1] == Point(20,10)	Пройден
И2	1) 1.11.2023	Проверка группы интеграции при отсутствии пути до вершины.	Exception('Нет пути до вершины')	Exception('Нет пути до вершины')	Пройден
И3	1) 1.11.2023	Проверка группы интеграции при случаи совпадающих вершин.	Exception('Начальная и конечные вершины должны отличаться')	Exception ('Начальная и конечные вершины должны отличаться')	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Группа 1					
И4	1) 2.11.2023	Проверка группы интеграции при нулевых коэффициентах ошибок.	Все точки way принадлежат ломаной – Point(0,0), Point(10,0), Point(20,10), len(way) > 3, и way[0] == Point(0,0), way[-1] == Point(20,10)	Все точки принадлежат ломаной – Point(0,0), Point(10,0), Point(20,10), len(way) > 3, и way[0] == Point(0,0), way[-1] == Point(20,10)	Пройден
И5	1) 2.11.2023	Проверка группы интеграции при нулевых коэффициентах ошибок и шаге пути, превышающем расстояние между вершинами на графе.	way = [Point(0,0), Point(10,0), Point(20,10)]	[Point(0,0), Point(10,0), Point(20,10)]	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Группа 2					
И6	1) 02.12.2023	Проверка группы интеграции при случаи вершин на одной прямой.	Массив error_way: len(way) == 3, |Atan2 ({error_way[0], error_way[1]}, {error_way[1], error_way[2]}) - Atan2({way[0],way[1]}, {way[1],way[2]})| <= 2/180*Pi	Массив error_way: len(way) == 3, |Atan2 ({error_way[0], error_way[1]}, {error_way[1], error_way[2]}) - Atan2({way[0],way[1]}, {way[1],way[2]})| <= 2/180*Pi	Пройден
И7	1) 02.11.2023	Проверка группы интеграции при использовании выходных данных предыдущей группы.	Для всех i от 1 до len(error_way)-1: Найти Atan({error_way[i-1], error_way[i]}, {error_way[i], error_way[i+1]}) Проверить, что выражение по модулю < 2/180*Pi; error_way[0] == way[0], len(error_way) == len(way).	Для всех i от 1 до len(error_way)-1: Atan({error_way[i-1], error_way[i]}, {error_way[i], error_way[i+1]}) по модулю < 2/180*Pi; error_way[0] == way[0], len(error_way) == len(way).	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Группа 2					
И8	1) 02.12.2023	Проверка группы интеграции при двух точках.	Массив error_way: len(error_way) == 2, |Atan2({way[0], error_way[1]}) - Atan2({way[0], way[1]},)| <= 2/180*Pi	Массив error_way: len(error_way) == 2, |Atan2({way[0], error_way[1]}) - Atan2({way[0], way[1]},)| <= 2/180*Pi	Пройден
И9	1) 03.12.2023	Проверка группы интеграции при отрицательном коэффициенте ошибки.	Exception ('Максимальный угол ошибки должен быть положительным')	Exception ('Максимальный угол ошибки должен быть положительным')	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Группа 3					
И10	1) 03.12.2023	Проверка группы интеграции при пути, проходящем вдоль препятствия.	Список track содержит len(error_way) точек, при этом для любой точки point из track: point не принадлежит области blockedZones.Points и blockedZones.Plan, len(error_way[i], error_way[i+1]) == len(track[i], track[i+1]).	Список track содержит len(error_way) точек, при этом для любой точки point из track: point не принадлежит области blockedZones.Points и blockedZones.Plan, len(error_way[i], error_way[i+1]) == len(track[i], track[i+1]).	Пройден
И11	1) 03.11.2023	Проверка группы интеграции при пути, проходящем внутри препятствия у его границы.	Список track содержит len(error_way) точек, при этом для любой точки point из track: point не принадлежит области blockedZones.Points и blockedZones.Plan, len(error_way[i], error_way[i+1]) == len(track[i], track[i+1]), при этом error_way[i].Y $<$ 2.	Список track содержит len(error_way) точек, для любой точки point из track: point не принадлежит области blockedZones.Points и blockedZones.Plan, len(error_way[i], error_way[i+1]) == len(track[i], track[i+1]), error_way[i].Y $<$ 2.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Группа 3					
И12	1) 03.12.2023	Проверка группы интеграции при пути, проходящем внутри препятствия у его границы и выходящем из него.	Список track содержит len(error_way) точек, при этом для любой точки point из track: point не принадлежит области blockedZones.Points и blockedZones.Plan, len(error_way[i], error_way[i+1]) == len(track[i], track[i+1]), при этом error_way[i].Y < 2 либо error_way[i].X > 8.	len(track) == len(error_way), при этом для любой точки point из track: point не принадлежит области blockedZones.Points и blockedZones.Plan, len(error_way[i], error_way[i+1]) == len(track[i], track[i+1]), при этом error_way[i].Y < 2 либо error_way[i].X > 8.	Пройден
И13	1) 03.12.2023	Проверка группы интеграции при пути, проходящем внутри препятствия без возможности обхода препятствия.	Exception('Не возможно построить исправленную траекторию')	Exception ('Index Out of RangeException ')	Не пройден
И13	2) 05.12.2023	Проверка группы интеграции при пути, проходящем внутри препятствия без возможности обхода препятствия.	Exception('Не возможно построить исправленную траекторию')	Exception('Не возможно построить исправленную траекторию')	Пройден

10.3 Аттестационное тестирование

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Выбор точек начала и конца»					
A1	1) 04.12.2023	Проверка правильности работы выбора точки начала путём нажатия в окне отрисовки в область белого цвета внутри области ломаной.	В координате нажатия нарисуется круг красного цвета.	В координате нажатия нарисуется круг красного цвета.	Пройден
A2	1) 04.12.2023	Проверка невозможности задания точки начала внутри сложного полигона путём нажатия в окне отрисовки в область серого цвета внутри области ломаной.	В координате нажатия не произойдёт ничего.	В координате нажатия не произойдёт ничего.	Пройден
A3	1) 04.12.2023	Проверка правильности работы выбора точки конца путём нажатия в окне отрисовки в область белого цвета внутри области ломаной.	В координате нажатия нарисуется круг синего цвета.	В координате нажатия нарисуется круг синего цвета.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Выбор точек начала и конца»					
A4	1) 04.12.2023	Проверка невозможности задания точки конца внутри сложного полигона путём нажатия в окне отрисовки в область серого цвета внутри области ломаной.	В координате нажатия не произойдёт ничего.	В координате нажатия не произойдёт ничего.	Пройден
A5	1) 04.12.2023	Проверка невозможности задания точки начала вне плана путём нажатия в окне отрисовки в область, вовне области ломаной.	В координате нажатия не произойдёт ничего.	В координате нажатия не произойдёт ничего.	Пройден
A6	1) 04.12.2023	Проверка невозможности задания точки конца вне плана путём нажатия в окне отрисовки в область, вовне области ломаной.	В координате нажатия не произойдёт ничего.	В координате нажатия не произойдёт ничего.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Выбор точек начала и конца»					
А7	1) 04.12.2023	Проверка при отсутствии загруженного сложного полигона. путём нажатия в область отрисовки.	Выведется сообщение в виде диалогового окна об отсутствии загруженного полигона.	Вывелась ошибка в ходе выполнения программы Exception('Index OutOfRange')	Не пройден
А7	1) 04.12.2023	Проверка при отсутствии загруженного сложного полигона. путём нажатия в область отрисовки.	Выведется сообщение в виде диалогового окна об отсутствии загруженного полигона.	Вывелось сообщение в виде диалогового окна об отсутствии загруженного полигона.	Пройден

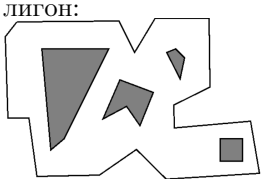
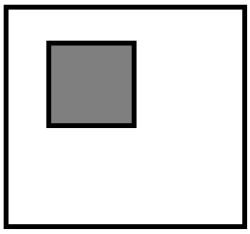
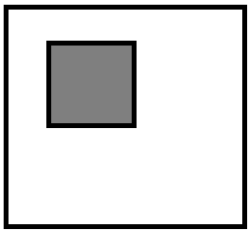
№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Очистка точек начала и конца траектории»					
A8	1) 05.12.2023	Проверка удаления одной точки – начала, нажатием кнопки «Очистить точки».	Из области сложного полигона удалится поставленная точка начала – красный круг.	Из области сложного полигона удалился красный круг.	Пройден
A9	1) 05.12.2023	Проверка удаления одной точки – конца, нажатием кнопки «Очистить точки».	Из области сложного полигона удалится поставленная точка конца – синий круг.	Из области сложного полигона удалился синий круг.	Пройден
A10	1) 05.12.2023	Проверка удаления всех поставленных точек, нажатием кнопки «Очистить точки».	Из области сложного полигона удалятся все точки – исчезнет синий и красный круг.	Из области сложного полигона удалились синий и красный круги.	Пройден
A11	1) 05.12.2023	Проверка при отсутствии точек нажатием кнопки «Очистить точки».	Область сложного полигона останется без изменений.	Область сложного полигона осталась без изменений.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Очистка точек начала и конца траектории»					
A12	1) 05.12.2023	Проверка при отсутствии загруженного сложного полигона нажатием кнопки «Очистить точки».	Выведется сообщение в виде диалогового окна об отсутствии загруженного полигона.	Вывелась ошибка в ходе выполнения программы Exception('Index OutOfRange').	Не пройден
A12	1) 07.12.2023	Проверка при отсутствии загруженного сложного полигона нажатием кнопки «Очистить точки».	Выведется сообщение в виде диалогового окна об отсутствии загруженного полигона.	Выведется сообщение в виде диалогового окна об отсутствии загруженного полигона.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Задание коэффициента ошибки сымитированной траектории»					
A13	1) 07.12.2023	Проверка влияния коэффициента на траекторию.	Построится 2 траектории, синяя будет соединять красную и синюю точки и лежать строго внутри сложного полигона, оранжевая будет отходить от синей.	Построилось 2 траектории, синяя соединяет красную и синюю точки и лежит строго внутри сложного полигона, оранжевая отходит от синей.	Пройден
A14	1) 07.12.2023	Проверка разных коэффициентов на траекториях.	Построится 4 траектории: две синих будет соединять красную и синюю точки и лежать строго внутри сложного полигона, и 2 оранжевых, последняя из которых будет сильней отходить в сторону в сравнении с 1 оранжевой.	Построились 4 траектории: две синих соединяют красную и синюю точки и лежат строго внутри сложного полигона, и 2 оранжевых, последняя из которых сильней отходит в сторону в сравнении с 1 оранжевой.	Пройден
A15	1) 07.12.2023	Проверка не верного коэффициента.	Откроется диалоговое окно с сообщением, что нельзя ввести отрицательный коэффициент.	Не произошло какой-либо реакции.	Не пройден
A16	1) 07.12.2023	Проверка не верного коэффициента (большего 90).	Откроется диалоговое окно с сообщением, что нельзя ввести коэффициент больше чем 90.	Не произошло какой-либо реакции.	Не пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Задание коэффициента ошибки сымитированной траектории»					
A15	2) 08.12.2023	Проверка не верного коэффициента (отрицательного).	Откроется диалоговое окно с сообщением, что нельзя ввести отрицательный коэффициент.	Открылось диалоговое окно с сообщением, что нельзя ввести отрицательный коэффициент.	Пройден
A16	2) 08.12.2023	Проверка не верного коэффициента (большего 90).	Откроется диалоговое окно с сообщением, что нельзя ввести коэффициент больше чем 90.	Откроется диалоговое окно с сообщением, что нельзя ввести коэффициент больше чем 90.	Пройден
Требование «Задание длины шага»					
A17	2) 08.12.2023	Проверка построения траектории с правильным шагом (положительный).	Построится 2 траектории, синяя будет соединять красную и синюю точки и лежать строго внутри сложного полигона, оранжевая будет отходить от синей; число звеньев превосходит 2.	Построилось 2 траектории, синяя соединяет красную и синюю точки и лежит строго внутри сложного полигона, оранжевая отходит от синей; число звеньев превосходит 2.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Задание длины шага»					
A18	1) 08.12.2023	Проверка 2 разных коэффициентов (первый меньше второго) на траекториях.	Построится 4 траектории: две синих будет соединять красную и синюю точки и лежать строго внутри сложного полигона, и 2 оранжевых. Количество звеньев в первой паре траекторий будет больше, чем во второй – т.е. длина звеньев будет меньше, чем во второй.	Построились 4 траектории: две синих соединяют красную и синюю точки и лежат строго внутри сложного полигона, и 2 оранжевых. Количество звеньев в первой паре траекторий больше, чем во второй, длина звеньев меньше, чем во второй.	Пройден
A19	1) 09.12.2023	Проверка большого коэффициента на траектории.	Построится 2 траектории, синяя будет соединять красную и синюю точки и лежать строго внутри сложного полигона, оранжевая будет отходить от синей. При этом, число звеньев траектории будет равно 2.	Построилось 2 траектории, синяя соединяет красную и синюю точки и лежит строго внутри сложного полигона, оранжевая отходит от синей. Число звеньев траектории равно 2.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Задание длины шага»					
A20	1) 08.12.2023	Проверка не верного коэффициента (отрицательный).	Откроется диалоговое окно с сообщением, что нельзя ввести отрицательную длину шага.	Не произошло никакой реакции.	Не пройден
A20	2) 09.12.2023	Проверка не верного коэффициента (отрицательный).	Откроется диалоговое окно с сообщением, что нельзя ввести отрицательную длину шага.	Открылось диалоговое окно с сообщением, что нельзя ввести отрицательную длину шага.	Пройден
Требование «Считывание карты помещения из файла в определенном формате»					
A21	1) 09.12.2023	Проверка загрузки сложного полигона.	Отобразится следующий сложный полигон: 	Отобразился следующий сложный полигон: 	Пройден
A22	1) 09.12.2023	Проверка загрузки сложного полигона с минимальным числом дыр.	Отобразится следующий сложный полигон: 	Отобразился следующий сложный полигон: 	Пройден
A23	1) 09.12.2023	Проверка выдачи ошибки при самопересекающемся плане.	Отобразится ошибка, что файл содержит самопересекающийся план.	Ошибка в ходе выполнения Exception('Файл содержит самопересекающийся план')	Не пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Считывание карты помещения из файла в определенном формате»					
A24	1) 09.12.2023	Проверка выдачи ошибки при вырожденном плане.	Отобразится ошибка, что файл содержит вырожденный план.	Ошибка в ходе выполнения Exception('Файл содержит вырожденный план')	Не пройден
A25	1) 09.12.2023	Проверка выдачи ошибки при самопересекающейся дыре.	Отобразится ошибка, что файл содержит самопересекающуюся дыру.	Ошибка в ходе выполнения Exception('Файл содержит самопересекающуюся дыру')	Не пройден
A26	1) 09.12.2023	Проверка выдачи ошибки при вырожденной дыре.	Отобразится ошибка, что файл содержит вырожденную дыру.	Ошибка в ходе выполнения Exception('Файл содержит вырожденную дыру')	Не пройден
A27	1) 09.12.2023	Проверка выдачи ошибки при вырожденной дыре.	Отобразится ошибка, что файл содержит не верный формат данных.	Отобразилась ошибка, что файл содержит не верный формат данных.	Пройден
A28	1) 09.12.2023	Проверка выдачи ошибки при не верных данных.	Отобразится ошибка, что файл содержит не верный формат данных.	Отобразилась ошибка, что файл содержит не верный формат данных.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Считывание карты помещения из файла в определенном формате»					
A23	2) 10.12.2023	Проверка выдачи ошибки при самопересекающемся плане.	Отобразится ошибка, что файл содержит самопересекающийся план.	Отобразилась ошибка, что файл содержит самопересекающийся план.	Пройден
A24	2) 10.12.2023	Проверка выдачи ошибки при вырожденном плане.	Отобразится ошибка, что файл содержит вырожденный план.	Отобразилась ошибка, что файл содержит вырожденный план.	Пройден
A25	2) 10.12.2023	Проверка выдачи ошибки при самопересекающейся дыре.	Отобразится ошибка, что файл содержит самопересекающуюся дыру.	Отобразилась ошибка, что файл содержит самопересекающуюся дыру.	Пройден
A26	2) 10.12.2023	Проверка выдачи ошибки при вырожденной дыре.	Отобразится ошибка, что файл содержит вырожденную дыру.	Отобразилась ошибка, что файл содержит вырожденную дыру.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Построение новой реальной траектории между заданными точками, не проходящей через дыры»					
A29	1) 10.12.2023	Проверка построения разных реальных траекторий.	Построится 2 траектории синего цвета, соединяющих красную и синюю точки и лежащих строго внутри сложного полигона, при этом эти траектории будут отличаться; а также будут построены 2 траектории оранжевого цвета, отходящих от соответствующей траектории синего цвета.	Построилось 2 траектории синего цвета, соединяющих красную и синюю точки и лежащих строго внутри сложного полигона, при этом эти траектории отличаются; построены 2 траектории оранжевого цвета, отходящие от соответствующей траектории синего цвета.	Пройден
A30	1) 10.12.2023	Проверка выдачи ошибки при отсутствии загруженного сложного полигона.	Отобразится диалоговое окно с сообщением, что нельзя построить траектории, если не загружен сложный полигон.	Ошибка в ходе выполнения Exception('Null Reference Exception')	Не пройден
A31	1) 10.12.2023	Проверка выдачи ошибки при не задании коэффициентов траектории.	Отобразится диалоговое окно с сообщением, что нельзя построить траектории, если не введены коэффициенты траектории.	Отобразится диалоговое окно с сообщением, что нельзя построить траектории, если не введены коэффициенты траектории.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Построение новой реальной траектории между заданными точками, не проходящей через дыры»					
А32	1) 10.12.2023	Проверка выдачи ошибки при не задании начальной и конечной точек траектории.	Отобразится ошибка, что не заданы точки траектории.	Отобразится ошибка, что не заданы точки траектории.	Пройден
А30	2) 11.12.2023	Проверка выдачи ошибки при отсутствии загруженного сложного полигона.	Отобразится диалоговое окно с сообщением, что нельзя построить траектории, если не загружен сложный полигон.	Отобразилось диалоговое окно с сообщением, что нельзя построить траектории, если не загружен сложный полигон.	Пройден
Требование «Построение исправленной траектории на основе алгоритма»					
А33	1) 12.12.2023	Проверка построения исправленной траекторий.	Построится траектория, состоящая из красных точек, которая будет проходить ближе к траектории синего цвета по сравнению с траекторией оранжевого цвета.	Построилась траектория, состоящая из красных точек, которая проходит ближе к траектории синего цвета по сравнению с траекторией оранжевого цвета.	Пройден

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Требование «Построение исправленной траектории на основе алгоритма»					
А34	1) 12.12.2023	Проверка построения отличающихся траекторий.	Построится 2 траектории, состоящих из красных точек, которые будет проходить ближе к своей траектории синего цвета, но они будут отличаться.	Построилось 2 траектории, состоящих из красных точек, которые проходят ближе к своей траектории синего цвета, но они отличаются.	Пройден
А35	1) 12.12.2023	Проверка выдачи ошибки при отсутствии зафиксированной траектории.	Отобразится диалоговое окно с сообщением, что не построена реальная и соответствующая ей зафиксированная траектория.	Отобразилось диалоговое окно с сообщением, что не построена реальная и соответствующая ей зафиксированная траектория.	Пройден

10.4 Нагрузочное тестирование

№	Номер попытки	Описание	Ожидаемый результат	Фактический результат	Итог
Тест №1					
Н1	1) 13.12.2023	Проверка времени выполнения алгоритма на 100 точках.	Траектория построится максимум за 700 миллисекунд.	Траектория построилась за 620 миллисекунд.	Пройден
Н2	1) 13.12.2023	Проверка времени выполнения алгоритма на 1000 точках.	Траектория построится максимум за 2600 миллисекунд.	Траектория построилась за 2120 миллисекунд.	Пройден
Н3	1) 13.12.2023	Проверка времени выполнения алгоритма на 10000 точках.	Траектория построится максимум за 25.8 секунд.	Траектория построилась за 23200 миллисекунд.	Пройден

11 Журнал ошибок

Журнал ошибок содержит информации о том, на каком тесте была обнаружена ошибка – кликабельная ссылка на тест в журнале тестирования, название формируется по правилу:

- <Сокращение класса>.<Сокращение метода>.<Номер теста>.<Попытка>.
- Пример: PQ.D.3.1 – класс PriorityQueue, метод DeleteMin, тест номер 3, попытка 1.
- Либо, если тест не относится к блочному тестированию, то название формируется из названия класса тестов – INT для интеграционных, ATT – для аттестационных, NAG – для нагрузочных, затем указывается номер теста и попытка.

Журнал ошибок содержит ожидаемый результат, результат, полученный в результате запуска теста, и исправление. Исправление содержит меры по исправлению в коде, принятые разработчиком.

Ошибка №1 тест Б7

Ссылка на тест в журнале	PQ.D.3.1
Дата выполнения	26.11.2023
Описание теста	Проверка возврата при пустой очереди.
Ожидаемый результат	Exception('Очередь пуста').
Фактический результат	Exception('Argument OutOfRange Exception')
Исправление	Добавлена проверка длины очереди перед извлечением.

Ошибка №2 тест Б10

Ссылка на тест в журнале	DIJ.GM.3.1
Дата выполнения	26.11.2023
Описание теста	Проверка случая, нет пути в конечную вершину
Ожидаемый результат	Exception('Нет пути между вершинами').
Фактический результат	Exception ('Превышено время ожидания выполнения теста "TestMethod3".')
Исправление	Добавлена обработка ситуации, когда алгоритм не доходит до вершины.

Ошибка №3 тест Б11

Ссылка на тест в журнале	DIJ.GM.4.1
Дата выполнения	26.11.2023
Описание теста	Проверка случая с 1 вершиной
Ожидаемый результат	Список [V1].
Фактический результат	Список двух вершин [V1, V1]
Исправление	Обработка ситуации с одной вершиной в пути.

Ошибка №4 тест Б16

Ссылка на тест в журнале	TF.WTS.4.1
Дата выполнения	26.11.2023
Описание теста	Проверка случая с 1 точкой
Ожидаемый результат	Список Exception('Нужно минимум 2 точки').
Фактический результат	список [Point(0,0), Point(0,0)]
Исправление	Обработка ситуации с одной вершиной в переданном списке.

Ошибка №5 тест Б20

Ссылка на тест в журнале	TF.IM.3.1
Дата выполнения	28.11.2023
Описание теста	Проверка случая с 2 точками
Ожидаемый результат	Exception('Для имитации необходимо минимум 3 точки')
Фактический результат	Список [Point(0,0)]
Исправление	Обработка ситуации с количеством переданных вершин.

Ошибка №6 тест Б21

Ссылка на тест в журнале	TF.IM.4.1
Дата выполнения	28.11.2023
Описание теста	Проверка случая с 3 точками, 2 из которых совпадают
Ожидаемый результат	Exception('Для имитации траектории не должны иди подряд совпадающие точки')
Фактический результат	Exception('Divide ByZero Exception')
Исправление	Обработка ситуации повторяющимися подряд вершинами.

Ошибка №7 тест Б29

Ссылка на тест в журнале	PF.GP.4.1
Дата выполнения	29.11.2023
Описание теста	Проверка случая при координатах равных 0
Ожидаемый результат	Поле particle содержит 10 элементов класса Particle, координаты Particle_i: Particle_i.Point.X = 0 и Particle_i.Point.Y = 0, Particle.Weight = 0.1
Фактический результат	Поле particle не содержит элементов
Исправление	Обработка граничной ситуации – смена оператора строгости на нестрогость.

Ошибка №8 тест Б30

Ссылка на тест в журнале	PF.GP.5.1
Дата выполнения	29.11.2023
Описание теста	Проверка случая при max_x < start_x и max_y < start_y
Ожидаемый результат	Поле particle содержит 10 элементов класса Particle, координаты Particle_i: 0 < Particle_i.Point.X < 10 и 0 < Particle_i.Point.Y < 10, Particle.Weight = 0.1
Фактический результат	Exception('Argument Out Of Range Exception')
Исправление	Обработка ситуации с убывающим порядком координат путём сортировки аргументов.

Ошибка №9 тест Б35

Ссылка на тест в журнале	PF.MP.5.1
Дата выполнения	29.11.2023
Описание теста	Проверка движения при повороте на большой угол
Ожидаемый результат	Поле particle содержит 1 элемент класса Particle, координаты Particle: Particle_0.Point.X = 0 и Particle_0.Point.Y = 10, Particle.Weight = 1, Particle.Orientation = 0.5*Pi
Фактический результат	Поле particle содержит 1 элемент класса Particle, Particle.Orientation = 98.5*Pi
Исправление	Вместо вычитания 2Pi для приведения ориентации к интервалу от 0 до 2Pi, используется остаток от деления.

Ошибка №10 тест Б44

Ссылка на тест в журнале	PF.GM.4.1
Дата выполнения	29.11.2023
Описание теста	Проверка исключения при отсутствии частиц
Ожидаемый результат	Exception('Нет частиц для нахождения центральной точки')
Фактический результат	Exception('Divide By Zero Exception')
Исправление	Обработка ситуации, когда частиц нет.

Ошибка №11 тест Б48

Ссылка на тест в журнале	PF.RP.4.1
Дата выполнения	29.11.2023
Описание теста	Проверка при отрицательном параметре количества
Ожидаемый результат	Exception('Число восстанавливаемых частиц должно быть положительным')
Фактический результат	Поле particles не изменилось
Исправление	Обработка ситуации, когда количество восстанавливаемых частиц отрицательно.

Ошибка №12 тест Б49

Ссылка на тест в журнале	PF.RP.5.1
Дата выполнения	29.11.2023
Описание теста	Проверка при отрицательном параметре количества
Ожидаемый результат	Exception('Радиус не может быть отрицательным')
Фактический результат	Поле particles не изменилось
Исправление	Обработка ситуации, когда переданный радиус отрицателен.

Ошибка №13 тест Б58

Ссылка на тест в журнале	PR.RB.3.1
Дата выполнения	30.11.2023
Описание теста	Проверка при правильном формате, но пересечении в точке
Ожидаемый результат	Exception("Дыры пересекаются")
Фактический результат	Список [[[0, 0] [10, 0], [5, 10]],[[0, 0], [-10, -20], [-5, -10]]]
Исправление	Обработка ситуации, когда дыры пересекаются в точке.

Ошибка №14 тест Б61

Ссылка на тест в журнале	PR.RB.6.1
Дата выполнения	30.11.2023
Описание теста	проверка при вырожденности дыры
Ожидаемый результат	Exception("Дыра вырождена")
Фактический результат	Список [[0, 0] [10, 0], [5, 0]]
Исправление	Обработка ситуации, когда точки дыры лежат на одной прямой.

Ошибка №15 тест Б65

Ссылка на тест в журнале	PR.RP.3.1
Дата выполнения	30.11.2023
Описание теста	Проверка при вырождении плана
Ожидаемый результат	Exception('План вырожден')
Фактический результат	Список [[0, 0] [10, 0], [5, 0]]
Исправление	Обработка ситуации, когда точки плана лежат на одной прямой.

Ошибка №16 тест И13

Ссылка на тест в журнале	INTG.3.4
Дата выполнения	03.12.2023
Описание теста	Проверка группы интеграции при пути, проходящем внутри препятствия без возможности обхода препятствия.
Ожидаемый результат	Exception('Не возможно построить исправленную траекторию')
Фактический результат	Exception ('Index Out of RangeException ')
Исправление	Обработка ситуации, когда траектория не может быть построена ввиду препятствия.

Ошибка №17 тест А7

Ссылка на тест в журнале	ATT.1.7
Дата выполнения	04.12.2023
Описание теста	Проверка при отсутствии загруженного сложного полигона. путём нажатия в область отрисовки.
Ожидаемый результат	Выведется сообщение в виде диалогового окна об отсутствии загруженного полигона.
Фактический результат	Вывелась ошибка в ходе выполнения программы Exception('Index OutOfRange')
Исправление	Проверка перед выполнением наличия загруженного сложного полигона.

Ошибка №18 тест А12

Ссылка на тест в журнале	ATT.2.5
Дата выполнения	05.12.2023
Описание теста	Проверка при отсутствии загруженного сложного полигона нажатием кнопки «Очистить точки».
Ожидаемый результат	Выведется сообщение в виде диалогового окна об отсутствии загруженного полигона.
Фактический результат	Вывелась ошибка в ходе выполнения программы Exception('Index OutOfRange').
Исправление	Проверка перед выполнением наличия загруженного сложного полигона.

Ошибка №19 тест А15

Ссылка на тест в журнале	АТТ.3.3
Дата выполнения	07.12.2023
Описание теста	Проверка не верного (отрицательного) коэффициента.
Ожидаемый результат	Откроется диалоговое окно с сообщением, что нельзя ввести отрицательный коэффициент.
Фактический результат	Не произошло какой-либо реакции.
Исправление	Проверка ввода коэффициента на отрицательность.

Ошибка №20 тест А16

Ссылка на тест в журнале	АТТ.3.4
Дата выполнения	07.12.2023
Описание теста	Проверка не верного коэффициента (большего 90).
Ожидаемый результат	Откроется диалоговое окно с сообщением, что нельзя ввести коэффициент больше чем 90.
Фактический результат	Не произошло какой-либо реакции.
Исправление	Проверка ввода коэффициента на превышение больше 90.

Ошибка №21 тест А20

Ссылка на тест в журнале	АТТ.4.4
Дата выполнения	08.12.2023
Описание теста	Проверка не верного коэффициента (отрицательный).
Ожидаемый результат	Откроется диалоговое окно с сообщением, что нельзя ввести отрицательную длину шага.
Фактический результат	Не произошло какой-либо реакции.
Исправление	Проверка ввода коэффициента на отрицательность.

Ошибка №22 тест A23

Ссылка на тест в журнале	ATT.4.4
Дата выполнения	09.12.2023
Описание теста	Проверка выдачи ошибки при самопересекающемся плане.
Ожидаемый результат	Отобразится ошибка, что файл содержит самопересекающийся план.
Фактический результат	Ошибка в ходе выполнения Exception('Файл содержит самопересекающийся план')
Исправление	Проверка плана из файла на самопересечение.

Ошибка №23 тест A24

Ссылка на тест в журнале	ATT.5.4
Дата выполнения	09.12.2023
Описание теста	Проверка выдачи ошибки при вырожденном плане.
Ожидаемый результат	Отобразится ошибка, что файл содержит вырожденный план.
Фактический результат	Ошибка в ходе выполнения Exception('Файл содержит вырожденный план')
Исправление	Проверка плана из файла на вырожденность.

Ошибка №24 тест A25

Ссылка на тест в журнале	ATT.5.5
Дата выполнения	09.12.2023
Описание теста	Проверка выдачи ошибки при самопересекающейся дыре.
Ожидаемый результат	Отобразится ошибка, что файл содержит самопересекающуюся дыру.
Фактический результат	Ошибка в ходе выполнения Exception('Файл содержит самопересекающуюся дыру')
Исправление	Проверка дыр из файла на самопересечение.

Ошибка №25 тест А26

Ссылка на тест в журнале	ATT.5.6
Дата выполнения	09.12.2023
Описание теста	Проверка выдачи ошибки при вырожденной дыре.
Ожидаемый результат	Отобразится ошибка, что файл содержит вырожденную дыру.
Фактический результат	Ошибка в ходе выполнения Exception('Файл содержит вырожденную дыру')
Исправление	Проверка дыр из файла на вырожденность.

Ошибка №26 тест А30

Ссылка на тест в журнале	ATT.6.2
Дата выполнения	10.12.2023
Описание теста	Проверка выдачи ошибки при отсутствии загруженного сложного полигона.
Ожидаемый результат	Отобразится диалоговое окно с сообщением, что нельзя построить траектории, если не загружен сложный полигон.
Фактический результат	Ошибка в ходе выполнения Exception('Null Reference Exception')
Исправление	Проверка на наличие загруженного сложного полигона.

Ошибка №27 тест Б55

Ссылка на тест в журнале	PF.RW.6.1
Дата выполнения	21.12.2023
Описание теста	проверка возврата исключения при отсутствии частиц
Ожидаемый результат	Exception('Частицы отсутствуют')
Фактический результат	Exception('Index Out Of Range Exception')
Исправление	Проверка на наличие частиц в поле particles.

12 Примеры тестов

```
[TestMethod]
public void Test1Positive()
{
    Vertex V1 = new Vertex(new Point(0, 0));
    Vertex V2 = new Vertex(new Point(0, 10));
    Vertex V3 = new Vertex(new Point(10, 0));
    Vertex V4 = new Vertex(new Point(10, 10));
    V1.AddConnection(V2);
    V1.AddConnection(V3);
    V3.AddConnection(V4);
    V2.AddConnection(V4);
    Dijkstra d = new Dijkstra();
    var way = d.GetMinDistance(V1, V2);
    var expected = new List<Point> { new Point(0, 0), new Point(0, 10) };
    Assert.AreEqual(expected.Count, way.Count);
    for (int i = 0; i < way.Count; i++)
    {
        Assert.AreEqual(way[i], expected[i]);
    }
}
```

Рис. 7 – Тест Б1, позитивный. Класс Deijkstra, метод GetMinDistance

```
[TestMethod]
public void Test3Negative()
{
    Vertex V1 = new Vertex(new Point(0, 0));
    Vertex V2 = new Vertex(new Point(0, 10));
    Vertex V3 = new Vertex(new Point(1, 0));
    Vertex V4 = new Vertex(new Point(1, 11));
    Vertex V5 = new Vertex(new Point(1, 12));

    V1.AddConnection(V2);
    V2.AddConnection(V4);
    V1.AddConnection(V3);
    V3.AddConnection(V4);

    Dijkstra d = new Dijkstra();

    var exception = Assert.ThrowsException<KeyNotFoundException>(
    () => d.GetMinDistance(V1,V5));
}
```

Рис. 8 – Проверка на исключение, тест Б3, блочный, негативный, класс Deijkstra, метод GetMinDistance

```
public class Group1
{
    Vertex V1, V2, V3, V4, V5, V6, V7;
    BlockedZones BlockedZones;
    [TestInitialize]
    public void Init()
    {
        BlockedZones = new BlockedZones();
        BlockedZones.startPlan = new List<Vertex> { new Vertex(-10,-10), new Vertex(100, -10),
        new Vertex(100,100), new Vertex(-10, 100)};
        BlockedZones.startPoints = new List<List<Vertex>>{new List<Vertex> {new Vertex(10, 50),
        new Vertex(20, 50), new Vertex(15, 60) }};
        V1 = new Vertex(new Point(0, 0));
        V2 = new Vertex(new Point(0, 10));
        V3 = new Vertex(new Point(10, 0));
        V4 = new Vertex(new Point(10, 10));
        V5 = new Vertex(new Point(20, 10));
        V6 = new Vertex(new Point(15, 20));
        V7 = new Vertex(new Point(20, 10));
        V1.AddConnection(V2);
        V1.AddConnection(V3);
        V2.AddConnection(V4);
        V3.AddConnection(V4);
        V4.AddConnection(V5);
        V4.AddConnection(V6);
        V5.AddConnection(V6);
        V6.AddConnection(V7);
        V3.AddConnection(V7);
    }
```

Рис. 9 – Интеграционные тесты. Конструкция, инициализирующаяся перед 1 группой тестов.

```
[TestMethod]
public void Test1Positive()
{
    Dijkstra d = new Dijkstra();
    var points = d.GetMinDistance(V1, V7);
    var steps = TrackFuncs.WayToSteps(points, 5);

    var way = TrackFuncs.Imitation(steps, BlockedZones);

    double sum = 0.0;
    for (int i = 2; i < way.Count - 2; i++)
    {
        Point vect1 = new Point(way[i].X - way[i - 1].X, way[i].Y - way[i - 1].Y);
        Point vect2 = new Point(way[i + 1].X - way[i].X, way[i + 1].Y - way[i].Y);

        Point s1 = new Point(steps[i].X - steps[i - 1].X, steps[i].Y - steps[i - 1].Y);
        Point s2 = new Point(steps[i + 1].X - steps[i].X, steps[i + 1].Y - steps[i].Y);

        var a1 = Math.Atan2(vect2.Y - vect1.Y, vect2.X - vect1.X);
        var a2 = Math.Atan2(s2.Y - s1.Y, s2.X - s1.X);
        sum += a2 - a1;
        Assert.IsTrue(Math.Abs(sum) < 15 * Math.PI / 180);
    }
    Assert.AreEqual(new Point(0,0), way[0]);
    Assert.AreEqual(new Point(20, 10), way.Last());
}
```

Рис. 10 – 1 группа интеграции, тест И1, позитивный.

```
[TestMethod]
public void Test1Positive()
{
    var error_way = new List<Point> { new Point(0,0), new Point(4,0),
    new Point(9,0), new Point(10,3), new Point(12,6) } ;
    var track = ProcessErrorWay(error_way);
    Assert.AreEqual(error_way.Count-1, track.Count);
    foreach (Point p in track)
    {
        Assert.IsTrue(IsInside(p, BlockedZones.startPlan.Select(t => t.Point).ToList()));
        Assert.IsTrue(!IsInside(p, BlockedZones.startPoints[0].Select(t => t.Point).ToList()));
    }
}
```

Рис. 11 – Интеграционный тест И10 в группе 3

```
public List<Point> ProcessErrorWay(List<Point> error_way)
{
    ParticalFilter.GenerateParticles(1000, 20, 20, BlockedZones.startPoints, BlockedZones.startPlan);
    List<Point> result = new List<Point>();// { error_way[0] };
    for (int i = 1; i < error_way.Count; i++)
    {
        Point e = new Point(error_way[i].X - error_way[i-1].X, error_way[i].Y - error_way[i-1].Y);
        ParticalFilter.MoveParticles(Math.Sqrt(e.X * e.X + e.Y * e.Y), Math.Atan2(e.Y, e.X));
        Console.WriteLine(Math.Atan2(e.Y, e.X));
        ParticalFilter.RecalculateWeight(error_way[i]);
        ParticalFilter.RemoveLittleWeightedParticles(0.0001);
        result.Add(ParticalFilter.GetMiddlePoint());
        ParticalFilter.RestoreParticles(ParticalFilter.GetMiddlePoint(), 1000, 2);
    }
    return result;
}
```

Рис. 12 – Метод, выполняющий все итерации для третьей группы интеграции. Тестовым методам остаётся удостоверить результат.

13 Покрытие тестами

Покрытие тестами производилось в среде Microsoft Visual Studio с использованием расширения ReSharper. Учитываются лишь запрограммированные в фреймворке Unit Testing Framework блочные и интеграционные тесты.

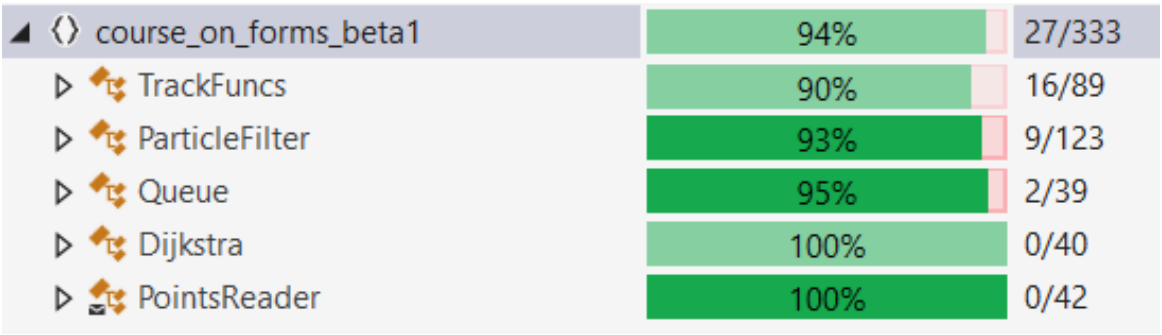


Рис. 13 – Покрытие тестами.

В среднем, тестами покрыто 94% тестируемых модулей, 27 строк из 333 не было покрыто.

14 Общее описание тестов

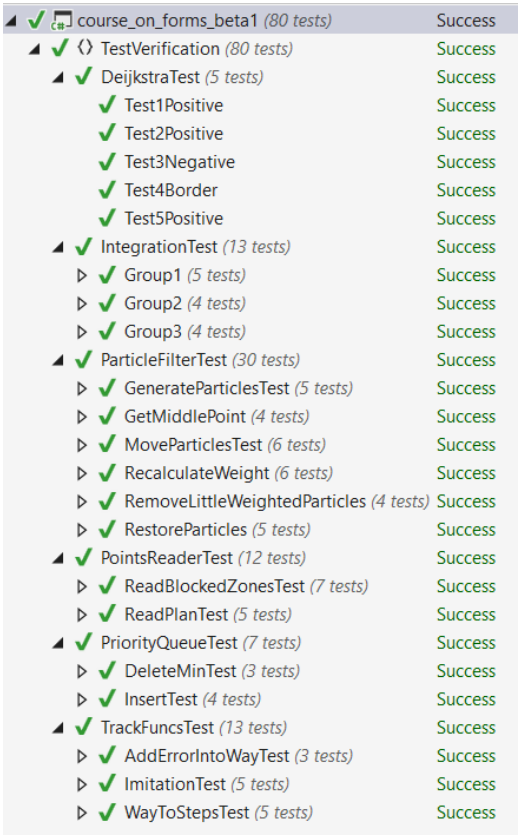


Рис. 14 – Пройденные тесты.

Всего было пройдено 80 теста, тест подразделялся на подгруппы в зависимости от того, имеет ли класс несколько методов для тестирования.

Например, класс Deijkstra имеет лишь 1 метод, поэтому в нём нет никаких подгрупп, тогда как класс ParticleFilter имеет сразу 6 методов, для которых созданы подгруппы, содержащие соответственные тесты из плана тестирования.

Подгруппа	Кол-во тестов	Позитивных	Граничных	Негативных	Исправлений
Класс Deijkstra					
GetMinDistance	5	3	1	1	2
Группы интеграции					
Group1	5	1	2	2	0
Group2	4	2	1	1	0
Group3	4	3	0	1	1
ParticleFilter					
Generate Particles	5	3	1	1	2
Get Middle Point	4	3	0	1	1
Move Particles	6	5	0	1	1
Remove Little Weighted Particles	4	2	1	1	0
Restore Particles	5	3	0	2	2
Restore Particles	6	2	3	1	1
Класс PointsReader					
Read BlockedZones	7	1	0	6	2
ReadPlan	5	1	0	4	1
Класс PriorityQueue					
DeleteMin	3	1	1	1	1
Insert	3	3	0	1	0
Класс TrackFuncs					
AddError IntoWay	3	1	1	1	1
Imitation	5	3	0	2	2
WayToSteps	5	3	1	1	1

Итого:

- 18 найденных и исправленных ошибок в интеграционных и блочных тестах;
- всего 80 теста в интеграционном и блочном тестировании;

- 40 позитивных, 12 граничных, 28 негативных тестов.

Описание аттестационных тестов:

Кол-во тестов	Позитивных	Граничных	Негативных	Исправлений
Требование «Выбор точек начала и конца»				
7	2	0	5	1
Требование «Очистка точек начала и конца траектории»				
5	4	0	1	1
Требование «Задание коэффициента ошибки сымитированной траектории»				
4	2	0	2	2
Требование «Задание длины шага»				
4	2	1	1	1
Требование «Считывание карты помещения из файла в опре- деленном формате»				
8	1	1	6	3
Требование «Построение новой реальной траектории между заданными точками, не проходящей через дыры»				
4	1	0	3	1
Требование «Построение исправленной траектории на основе алгоритма»				
3	2	0	1	0

- Всего 35 аттестационных тестов;
- 14 позитивных, 2 граничных, 19 негативных;
- В ходе тестов обнаруженено 9 ошибок.

Также были проведены нагрузочные испытания с замером времени выполнения работы на различных объемах вершин.

Тест	Количество точек	Ожидаемое время	Результат
H1	100	700млс.	Пройден
H2	1000	2600млс.	Пройден
H3	10000	25.8сек.	Пройден

15 Вывод

По итогу, было протестировано блочными и интеграционными тестами 306 строк кода системы для построения траектории мобильного объекта на сложном полигоне, в ходе которого было обнаружено 18 ошибок, которые были исправлены, и проведено повторное тестирование для удостоверения в том, что ошибки были устранены.

Было проведено аттестационное тестирование, в ходе которого было разработано 35 тестом и обнаружено 9 ошибок, которые также были исправлены.

Нагрузочное тестирование показало, что работа алгоритма выполняется за установленное время.

Таким образом, в ходе тестирования были произведены меры по исправлению ошибок, в связи с чем приложение системы стало работать с меньшим числом ошибок и меньше падать.