Петрозаводский государственный университет
Институт математики и информационных технологий

09.03.04 - Программная инженерия

Отчет по дисциплине «Верификация программного обеспечения»

Сглаживание траектории мобильного объекта на графе

Выполнил:
студент 4 курса группы 22407 группы

Д. А. Устинов _____
подпись

Руководитель:
К.А. Кулаков, к.ф.–м.н., доцент

_____
подпись

Итоговая оценка

_____
оценка

Петрозаводск — 2023

Содержание

1 Объект тестирования **4**

1.1 Описание системы 4

1.2 Архитектура 6

2 Стратегия тестирования **7**

2.1 Описание модулей 7

2.1.1 Модуль генерации пути 7

2.1.2 Модуль очереди с приоритетом 7

2.1.3 Модуль имитации траектории 8

2.1.4 Модуль реализующий фильтр частиц. 8

2.1.5 Модуль построения траектории на основе оценочных точек из фильтра частиц 9

2.2 Стратегия аттестационного тестирования 10

2.3 Стратегия блочного тестирования 10

2.4 Стратегия интеграционного тестирования 10

2.5 Стратегия специального тестирования 11

3 Детальный план тестов **12**

3.1 Блочное тестирование 12

3.2 Аттестационное тестирование 32

3.3 Интеграционное тестирование 37

3.4 Специальное тестирование 48

4 Журнал тестирования **50**

4.1 Аттестационное тестирование 50

4.2 Блочное тестирование 52

4.3 Интеграционное тестирование 65

4.4 Специальное тестирование 71

5 Журнал ошибок **72**

6 Примеры тестов **82**

7 Покрытие тестами **85**

8 Общее описание тестов **86**

9 Итог **87**

10 Ссылки на фактические результаты тестов **88**

1 Объект тестирования

1.1 Описание системы

Программная система – сглаживание траектории мобильного объекта на графе. Она включает в себя генерацию кратчайшего пути, разбиение пути на траекторию мобильного объекта, построение траектории с накапливающейся ошибкой и последующее сглаживание отклоненной траектории с ошибкой с помощью фильтра частиц.

Генерация пути будет происходить с помощью алгоритма Дейкстры, то есть поиск кратчайшего пути из точки А в точку Б. В свою очередь алгоритм Дейкстры использует очередь с приоритетом.

Датчик IMU, фиксируя местоположение объекта, чаще всего накапливает некоторую ошибку. Поэтому построение траектории будет осуществляться с той точки зрения, как её бы зафиксировал инерциальный измерительный блок. То есть будем имитировать траекторию с некоторой постоянно растущей ошибкой, как по длине, так и по углу.

Сглаживание ошибочной траектории будет происходить на основе фильтра частиц. Также в процессе сглаживания траектории может получится такая ситуация, что траектория получится прыгающей с ребро на ребро, в таком случае необходимо создавать дополнительные наборы частиц, которые будут откладываться на остальные смежные ребра. Данный метод включает также определение попала ли точка на отрезок (отложение перпендикуляра).

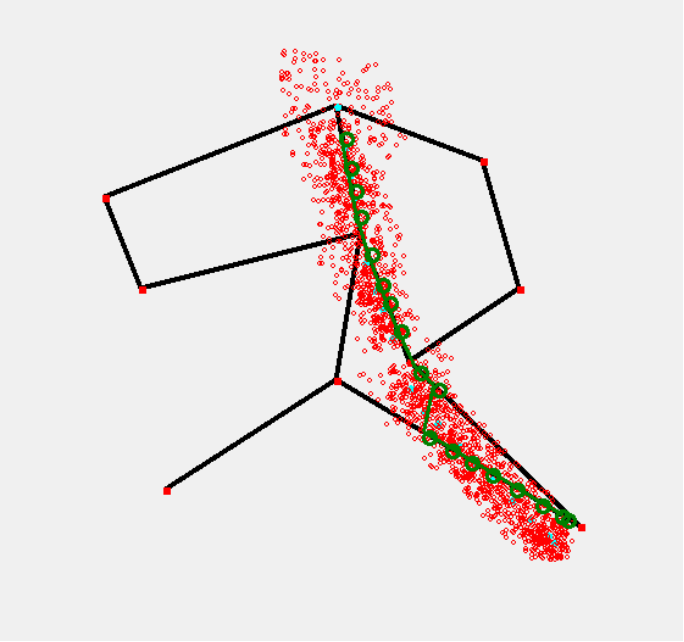


Рис. 1 – Прыгающая траектория

Таким образом, приложение представляет собой следующий функционал:

1. Генерация кратчашего пути на основе алгоритма Дейкстры;
2. Извлечение элемента из очереди с приоритетом;
3. Имитация траектории с пользовательской ошибкой;
4. Создание дополнительных частиц;
5. Работа фильтра частиц;
6. Построение сглаженной траектории на основе оценочных точек, получаемых из фильтра частиц;
7. Визуальное отображение полученных в ходе алгоритма траекторий.

Исходные датасеты:

1. Граф №1 состоит из следующих точек: (X = 300, Y = 75), (X = 110, Y = 150), (X = 420, Y = 120), (X = 320, Y = 180), (X = 140, Y = 225), (X = 450, Y = 225), (X = 300, Y = 300), (X = 360, Y = 285), (X = 160, Y = 390), (X = 500, Y = 420).

   Список связности:

   (X = 300, Y = 75): (X = 110, Y = 150), (X = 420, Y = 120), (X = 320, Y = 180).

   (X = 110, Y = 150): (X = 140, Y = 225), (X = 300, Y = 75).

   (X = 420, Y = 120): (X = 450, Y = 225), (X = 300, Y = 75).

   (X = 320, Y = 180): (X = 140, Y = 225), (X = 300, Y = 75), (X = 300, Y = 300), (X = 360, Y = 285).

   (X = 140, Y = 225): (X = 110, Y = 150), (X = 300, Y = 75), (X = 320, Y = 180).

   (X = 450, Y = 225): (X = 160, Y = 390), (X = 420, Y = 120).

   (X = 300, Y = 300): (X = 160, Y = 390), (X = 500, Y = 420), (X = 320, Y = 180).

   (X = 360, Y = 285): (X = 450, Y = 225), (X = 500, Y = 420), (X = 320, Y = 180).

   (X = 160, Y = 390): (X = 300, Y = 300).

   (X = 500, Y = 420): (X = 300, Y = 300), (X = 360, Y = 285).

2. Граф №2 состоит из следующих точек: (X = 300, Y = 25), (X = 450, Y = 150), (X = 225, Y = 175), (X = 50, Y = 250), (X = 325, Y = 350).

1.2 Архитектура

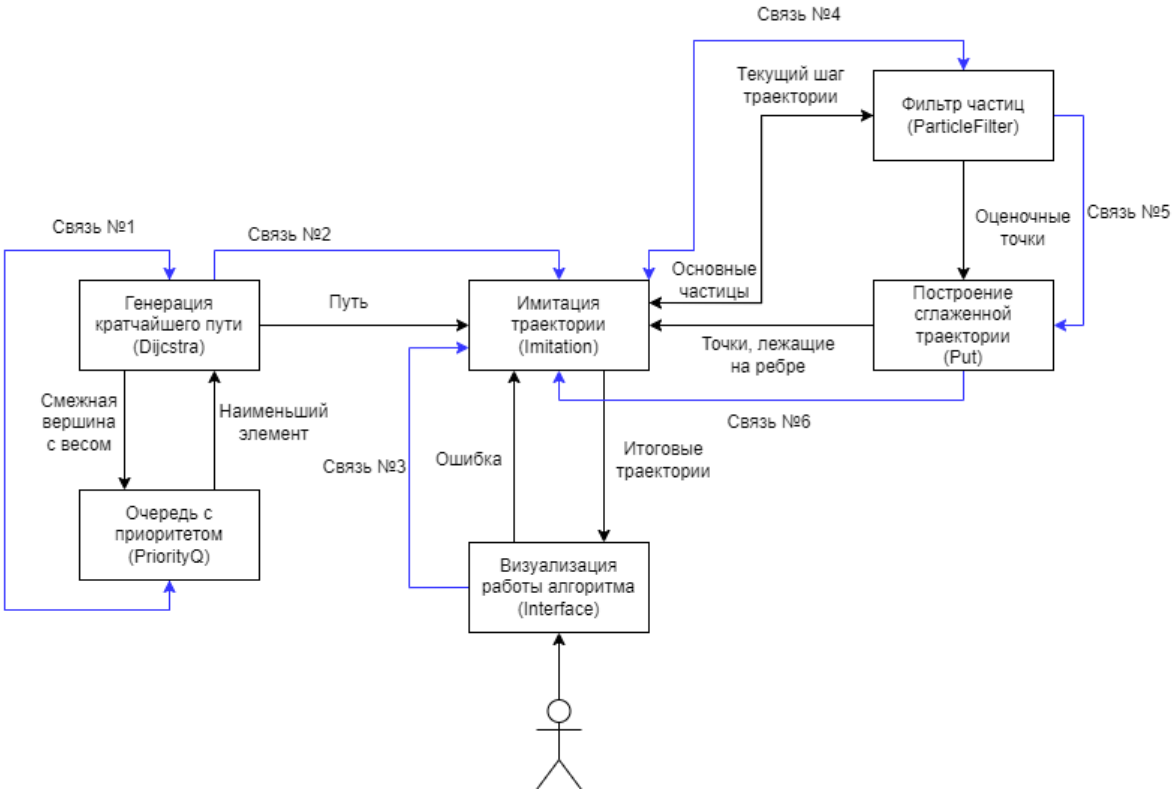


Рис. 2 – Архитектура

Описание связей указаны в главе 2.4.

2 Стратегия тестирования

2.1 Описание модулей

2.1.1 Модуль генерации пути

Модуль Dijcstra отвечает за генерацию кратчайшего пути из точки А в точку Б:

1. Метод AlgorithmDeicstra(Point start, Point end, Dictionary<Point, List<Point» spisok) для получения кратчайшего пути.

   На вход подаются следующие параметры: стартовая точка, конечная точка и словарь, который содержит смежные между собой вершины.

   Метод возвращает: (List<Point>) кратчайший путь из точки А в точку Б.

2. Метод Cost(Point p1, Point p2), определяющий вес, то есть длину ребра.

   На вход подаются следующие параметры: две точки, между которыми нужно определить длину ребра.

   Метод возвращает: (double) вес точки.

2.1.2 Модуль очереди с приоритетом

Модуль PriorityQ:

1. Метод Insert(Tuple<Point, double> met) для добавления элемента в очередь.

   На вход подаются следующие параметры: кортеж, содержащий информацию о передаваемой в метод точки.

2. Метод DeleteMin() – извлечение из очереди точки с минимальным весом.

   Метод возвращает: (Tuple<Point, double>) точку с минимальным весом.

3. Метод Swap(int child, int parent) для смены мест двух элементов очередь.

   На вход подаются следующие параметры: индексы двух элементов, которые нужно поменять местами.

4. Метод Count() – определяет, есть ли элементы в очереди.

   Метод возвращает: True – если очередь пустая, False – если в очереди есть элементы.

2.1.3 Модуль имитации траектории

Модуль Imitation отвечает за разбиение кратчайшего пути на траекторию и последующую имитацию этой траектории:

1. Метод MobileObject(List<Point> path) для разбиения кратчайшего пути на траекторию.
   На вход подаются следующие параметры: кратчайший путь.
   Метод возвращает: (List<Point>) траекторию мобильного объекта.

2. Метод ImitationPath(List<Point> points) – метод, строящий траекторию с ошибкой, сглаженную траекторию.
   На вход подаются следующие параметры: траектория мобильного объекта.
   Метод возвращает: (List<Dictionary<Point, List<Particle»>) Итоговые траектории.

2.1.4 Модуль реализующий фильтр частиц.

Модуль ParticleFilter отвечает за работу фильтра частиц:

1. Метод Initialisation(double count, Point StartPoint) для инициализации набора частиц вокруг заданной (стартовой) точки.
   На вход подаются следующие параметры: число частиц, которое необходимо создать, точка вокруг которой необходимо создать набор частиц.
   Метод возвращает: (List<Particle>) массив частиц.

2. Метод Prediction(Point StartPoint, double length, double angle) – метод, прогнозирующий движение частиц.
   На вход подаются следующие параметры: текущую точку, неверно зафиксированной траектории, длина "шага"(расстояние между двумя соседними точками ошибочной траектории), угол, на который ошибочно откладывается новая точка из траектории.
   Метод возвращает: (List<Particle>) Обновленный набор частиц.

3. Метод MeasurementUpdate(Point StartPoint, List<Particle> newParticles) – метод, определяющий вес каждой частицы.

На вход подаются следующие параметры: текущую точку, неверно зафиксированной траектории, предсказанный набор частиц из метода Prediction.

Метод возвращает: (List<Particle>) Обновленный набор частиц с новым весом.

4. Метод Resambling(Point StartPoint, List<Particle> newParticles) – метод, удаляющий частицы с малым весом.

   На вход подаются следующие параметры: текущую точку, неверно зафиксированной траектории, Обновленный набор частиц с новым весом.

   Метод возвращает: (List<Particle>) Обновленный набор частиц, который не содержит частицы с малым весом.

5. Метод AddParticles(int count, List<Particle> newParticles) – метод, восполняющий количество частиц в наборе.

   На вход подаются следующие параметры: текущую точку, неверно зафиксированной траектории, Обновленный набор частиц с новым весом.

   Метод возвращает: (List<Particle>) Обновленный набор частиц, который не содержит частицы с малым весом.

6. Метод MeasurePosition(List<Particle> newParticles) – метод, оценивающий текущее состояние фильтра частиц, то есть рассчитывается взвешенная сумма позиций всех частиц.

   На вход подаются следующие параметры: набор частиц.

   Метод возвращает: (List<Particle>) Взвешенная сумма позиций всех частиц (оценочная точка).

2.1.5 Модуль построения траектории на основе оценочных точек из фильтра частиц

Модуль Put определяющий, какому ребру исходного графа принадлежит точка из метода MeasurePosition:

1. Метод GetDistance(Point currentAveragePoint) для определения расстояния от точки до ребер.

   На вход подаются следующие параметры: текущая точка, от которой необходимо рассчитать расстояние до всех ребер графа.

   Метод возвращает: (Dictionary<Tuple<Point, Point>, double>) словарь, содержащий ребра и расстояния, от переданной в метод оценочной точки до этих дуг.

2. Метод GetPointOnEdge() – метод, откладывающий точку на ребре с минимальным весом, из метода GetDistance.

   Метод возвращает: (List<Point>) сглаженная траектория мобильного объекта.

2.2 Стратегия аттестационного тестирования

Аттестационное тестирование будет проводиться вручную. В ходе тестирования будут проверяться следующие требования:

1. Воможность визуализации двух исходных графов;
2. Возможность очищения области отрисовки графов;
3. Возможность ввода пользовательской ошибки;
4. Построение сглаженной траектории на основе оценочных точек, получаемых из фильтра частиц.

2.3 Стратегия блочного тестирования

Для выполнения блочного тестирования будут использоваться библиотека NUnit. Блочные тесты предполагают тестирование отдельных функций и модулей программы отдельно друг от друга. Будут протестированы все методы, из раздела «Описание модулей».
В модуле Dijcstra будут протестированы все методы.
В модуле PriorityQ не будут протестированы методы Swap и Count, ввиду их тривиальности.
В модуле Imitation будут протестированы все методы.
В модуле ParticleFilter будут протестированы все методы, кроме Update и Resampling, ввиду их тривиальности.
В модуле Put будут протестированы все методы.

2.4 Стратегия интеграционного тестирования

При интеграционном тестировании проверяется взаимодействие между отдельными блоками программы.

Связи:

Связь №1 описывает взаимосвязь между модулями Dijcstra и PriorityQ, а именно передачу вершины типа Point (int X, int Y) с весом типа double из Dijcstra в PriorityQ

(вызов метода Insert(Tuple<Point, double> met) в методе AlgorithmDeicstra(Point start, Point end, Dictionary<Point, List<Point» spisok)) и извлечение и передачу наименьшей вершины тип Point (int X, int Y) из PriorityQ в Dijcstra (вызов метода DeleteMin в методе AlgorithmDeicstra(Point start, Point end, Dictionary<Point, List<Point» spisok)).

Связь №2 описывает взаимосвязь между модулями Dijcstra и Imitation, а именно передачу массива вершины типа List<Point> из Dijcstra в Imitation (вызов метода AlgorithmDeicstra(start, Point end, Dictionary<Point, List<Point» spisok) и передача найденных вершин в метод MobileObject(List<Point> path)) и обработка массива вершин пути и преобразования их в новый массив (разбиение на шаги) типа List<Point>.

Связь №1 и №2 будут протестированы вместе и будут представлять один блок интеграции.

Связь №3 описывает взаимосвязь между модулями Interface и Imitation, а именно заполнение поля ввода типа TextBox (пользовательской ошибки) и влияение этой ошибки на имитацию траектории.

Связь №4 описывает взаимосвязь между модулями ParticleFilter и Imitation, а именно передачу вершины типа Point (int X, int Y), double step (длины шага) и double angle (угол направления) на каждой итерации цикла из Imitation в ParticleFilter.

Связь №5 описывает взаимосвязь между модулями ParticleFilter и Put, а именно передачу массива оценочных точек типа List<Particle> из ParticleFilter в Put и получение точки типа Point(int X, int Y), которая принадлежит какому-то ребру из исходного графа (Датасет).

Связь №6 описывает взаимосвязь между модулями Put и Imitation, а именно передачу вершины типа Point(int X, int Y) из Put в Imitation и формирование траектории в виде массива типа List<Point>.

Связь №3, №4, №5 и №6 будут протестированы вместе и будут представлять один блок интеграции.

2.5 Стратегия специального тестирования

В ходе нагрузочного тестирования будет проверять время работы разработанного комплекса алгоритмов.

Тестирование будет производится путем замерения времени выполнения соответствующих методов для решения исходной задачи, затем это затраченное время будет сравниваться с эталонным.

Эталонное время подбиралось на основе тестирования алгоритмов

Ниже перечислены количество частиц (точек) и соответствующее максимальное время, за которое должна отработать программа:

1. 1000 точек – 50 млс;
2. 5000 точек – 100 млс;
3. 10000 точек – 150 млс.

3 Детальный план тестов

3.1 Блочное тестирование

Номер теста	$Б_1$
Объект тестирования	Модуль генерации кратчайшего пути (метод AlgorithcDeicstra(Point start, Point end, Dictionary<Point, List<Point» spisok))
Цель тестирования	Проверить генерацию кратчайшей траектории между двумя точками
Входные данные	Стартовая точка: Point (int X = 300, int Y = 75), конечная точка: Point (int X = 500, int Y = 420), словарь смежных ребер: (Dictionary<Point, List<Point»(датасет Граф №1))
Ожидаемый результат	Кратчайший путь массив точек List<Point>:[Point(int X = 300, int Y = 75); Point(int X = 320, int Y = 180); Point (int X = 360, int Y = 285); Point (int X = 500, int Y = 420)]
Тип	**Позитивный**

Номер теста	$Б_2$
Объект тестирования	Модуль генерации кратчайшего пути (метод Cost(Point p1, Point p2))
Цель тестирования	Проверить правильность расчета расстояния между двумя точками
Входные данные	Стартовая точка: Point (int X = 300, int Y = 75), конечная точка ребра Point (int X = 110, int Y = 150)
Ожидаемый результат	Расстояние между точками: double (204,26698215815497)
Тип	**Позитивный**

Номер теста	$Б_3$
Объект тестирования	Модуль очереди с приоритетом (метод Insert(Tuple<Point, double> met))
Цель тестирования	Проверить создается ли очередь с приоритетом и добавляется ли туда вершина
Входные данные	Кортеж, содержащий точку и вес: Tuple(Point(int X = 300, int Y = 75), 0.0)
Ожидаемый результат	Очередь содержит в себе 1 элемент(Tuple(Point(int X = 300, int Y = 75), 0.0)): int 1
Тип	**Позитивный**

Номер теста	$Б_4$
Объект тестирования	Модуль очереди с приоритетом (метод DeleteMin())
Цель тестирования	Проверить удаляется ли из очереди с приоритетом точка с наименьшим весом
Входные данные	Очередь с приоритетом, содержащая кортежи точек с весом Tuple(Point(int X = 300, int Y = 75), 0.0), Tuple(Point(int X = 500, int Y = 120), 10.0)
Ожидаемый результат	Извлеченный кортеж: Tuple(Point(int X = 300, int Y = 75), 0.0
Тип	**Позитивный**

Номер теста	$Б_5$
Объект тестирования	Модуль очереди с приоритетом (метод DeleteMin())
Цель тестирования	Проверить корректно ли будет извлечен элемент из очереди с отрицательным весом
Входные данные	Кортеж, содержащий точку и вес: Tuple(Point(int X = 300, int Y = 75), -120.0)
Ожидаемый результат	Выведется сообщение: "Неверный формат элемента!"
Тип	**Негативный**

Номер теста	$Б_6$
Объект тестирования	Модуль очереди с приоритетом (метод DeleteMin())
Цель тестирования	Проверить корректно ли будет извлечен элемент из очереди с отрицательными координатами
Входные данные	Кортеж, содержащий точку и вес: Tuple(Point(-300, 75), 10.0)
Ожидаемый результат	Выведется сообщение: "Неверный формат элемента!"
Тип	**Негативный**

Номер теста	$Б_7$
Объект тестирования	Модуль генерации кратчайшего пути с приоритетом (метод Cost(Point p1, Point p2))
Цель тестирования	Проверить рассчитывается ли вес (длину ребра) из двух точек с отрицательными координатами
Входные данные	Стартовая точка: Point(int X = -300, int Y = -75), Point(int X = -110, int Y = -150)))
Ожидаемый результат	Выведется сообщение: "Неверный формат точки!"
Тип	**Негативный**

Номер теста	$Б_8$
Объект тестирования	Модуль генерации кратчайшего пути с приоритетом (метод AlgorithmDeicstra(Point start, Point end, Dictionary<Point, List<Point» spisok))
Цель тестирования	Проверить рассчитывается ли кратчаший путь если туда переданы точки (стартовая и конечная) с отрицательными координатами
Входные данные	Генерация кратчайшего пути, вызывается метод AlgorithmDeicstra(Point(int X = -300, int Y = 75), Point(int X = 500, int Y = 420)), Dictionary<Point, List<Point»(датасет Граф №1))
Ожидаемый результат	Выведется сообщение: "Неверный формат точки!"
Тип	**Негативный**

Номер теста	$Б_9$
Объект тестирования	Модуль имитации пути (метод MobileObject(List<Point> path))
Цель тестирования	Разбивается ли путь на траекторию мобильного объекта
Входные данные	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 320, int Y = 180), Point(int X = 360, int Y = 285), Point(int X = 500, int Y = 420)]
Ожидаемый результат	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180), Point(int X = 338, int Y = 227), Point(int X = 352, int Y = 264), Point(int X = 360, int Y = 285), Point(int X = 395, int Y = 319), Point(int X = 426, int Y = 349), Point(int X = 451, int Y = 373), Point(int X = 482, int Y = 403), Point(int X = 500, int Y = 420)]. Значения могут разниться до 50 координат по каждой из осей, ввиду использования рандомайзера
Тип	**Позитивный**

Номер теста	$Б_{10}$
Объект тестирования	Модуль имитации пути (метод MobileObject(List<Point> path))
Цель тестирования	Разбивается ли путь на траекторию, если в переданном в метод массиве точки имеют отрицательные координаты
Входные данные	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = -320, int Y = 180), Point(int X = 360, int Y = 285), Point(int X = 500, int Y = 420)]
Ожидаемый результат	Выведется сообщение: "Неверный формат точки!"
Тип	**Негативный**

Номер теста	$Б_{11}$
Объект тестирования	Модуль имитации пути (метод ImitationPath(Point Start, Point End, ParticleFilter ParticleFilter, Point lastP, List<Point> points))
Цель тестирования	Сглаживается ли ошибочная траектория по отношению к графу
Входные данные	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]
Ожидаемый результат	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 301, int Y = 84), Point(int X = 307, int Y = 112), Point(int X = 310, int Y = 130), Point(int X = 313, int Y = 146)]. Значения могут разниться до 50 координат по каждой из осей, ввиду использования рандомайзера.
Тип	**Позитивный**

Номер теста	$Б_{12}$
Объект тестирования	Модуль имитации пути (метод ImitationPath(Point Start, Point End, ParticleFilter ParticleFilter, Point lastP, List<Point> points))
Цель тестирования	Сглаживается ли сымитированная траектория с ошибкой, если передан массив с точками, имеющими отрицательные координаты
Входные данные	Массив точек List<Point>: [Point(int X = -300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]
Ожидаемый результат	Выведется сообщение: "Неверный формат точки!"
Тип	**Негативный**

Номер теста	$Б_{13}$
Объект тестирования	Модуль реализующий фильтр частиц (метод Initialisation(double count, Point StartPoint, int ogrX, int ogrY))
Цель тестирования	Проверить создаются ли частицы вокруг переданной точки (point StartPoint)
Входные данные	Количество точек: double count = 5, Стартовая точка: Point(int X = 300, int Y = 75)
Ожидаемый результат	Массив частиц List<Particle>: [Point(int X = 310, int Y = 77), Point(int X = 303, int Y = 85), Point(int X = 289, int Y = 97), Point(int X = 280, int Y = 78), Point(int X = 317, int Y = 78)] Значения могут разниться до 30 координат по каждой из осей, ввиду использования рандомайзера.
Тип	**Позитивный**

Номер теста	$Б_{14}$
Объект тестирования	Модуль реализующий фильтр частиц (метод Initialisation(double count, Point StartPoint, int ogrX, int ogrY))
Цель тестирования	Проверить создаются ли частицы вокруг переданной точки (point StartPoint), если количество частиц имеет отрицательное значение
Входные данные	Количество точек: double count = -5, Стартовая точка: Point(int X = 300, int Y = 75)
Ожидаемый результат	Выведется сообщение: "Некорректный формат или количество точек!"
Тип	**Негативный**

Номер теста	$Б_{15}$
Объект тестирования	Модуль реализующий фильтр частиц (метод Initialisation(double count, Point StartPoint, int ogrX, int ogrY))
Цель тестирования	Проверить создаются ли частицы вокруг переданной точки (point StartPoint), если переданная точка имеет некорректный формат (отрицательные координаты)
Входные данные	Количество точек: double count = 5, Стартовая точка: Point(int X = -300, int Y = 75)
Ожидаемый результат	Выведется сообщение: "Некорректный формат или количество точек!"
Тип	**Негативный**

Номер теста	$Б_{16}$
Объект тестирования	Модуль реализующий фильтр частиц (метод Prediction(int count, Point StartPoint, double length, double angle))
Цель тестирования	Проверить обновляются ли список частиц на основании переданной длины шага и угла поворота
Входные данные	Стартовая точка: Point(int X = 300, int Y = 75), длина: double length = 30, угол: double angle = 0.009. Перед вызовом самого метода Prediction, необходимо вызвать метод Initialisation(5, Point(int X = 300, int Y = 75), -30, +30) для первоначальной генерации частиц.
Ожидаемый результат	Массив частиц List<Particle>: Point(int X = 257, int Y = 33), Point(int X = 291, int Y = 74), Point(int X = 306, int Y = 72), Point(int X = 301, int Y = 41), Point(int X = 332, int Y = 117). Значения могут разниться до 30 координат по каждой из осей, ввиду использования рандомайзера.
Тип	**Позитивный**

Номер теста	$Б_{17}$
Объект тестирования	Модуль реализующий фильтр частиц (метод Prediction(int count, Point StartPoint, double length, double angle))
Цель тестирования	Проверить обновляются ли список частиц (на этапе предсказания), если заданная точка имеет отрицательные координаты
Входные данные	Стартовая точка: Point(int X = -300, int Y = 75), длина: double length = 30, угол: double angle = 0.009. Перед вызовом самого метода Prediction, необходимо вызвать метод Initialisation(5, Point(int X = 300, int Y = 75), -30, +30) для первоначальной генерации частиц.
Ожидаемый результат	Выведется сообщение: "Некорректный формат точки или длину шага!"
Тип	**Негативный**

Номер теста	$Б_{18}$
Объект тестирования	Модуль реализующий фильтр частиц (метод Prediction(int count, Point StartPoint, double length, double angle))
Цель тестирования	Проверить обновляются ли список частиц (на этапе предсказания), если длина шага имеет отрицательное значение
Входные данные	Стартовая точка: Point(int X = 300, int Y = 75), длина: double length = -30, угол: double angle = 0.009. Перед вызовом самого метода Prediction, необходимо вызвать метод Initialisation(5, Point(int X = 300, int Y = 75), -30, +30) для первоначальной генерации частиц.
Ожидаемый результат	Выведется сообщение: "Некорректный формат точки или длину шага!"
Тип	**Негативный**

Номер теста	$Б_{19}$
Объект тестирования	Модуль реализующий фильтр частиц (метод MeasurementUpdate(int count, Point StartPoint, List<Particle> newParticles))
Цель тестирования	Проверить имеет ли ближайшая частица к стартовой (переданной в метод) точке наибольший вес среди остальных частиц
Входные данные	Стартовая точка: Point(int X = 300, int Y = 75), список частиц с весом List<Particle>: [Particle(double 0.2, Point(int X = 310, int Y = 77)), Particle(double 0.2, Point(int X = 303, int Y = 85)), Particle(double 0.2, Point(int X = 289, int Y = 97)), Particle(double 0.2, Point(int X = 280, int Y = 78)), Particle(double 0.2, Point(int X = 317, int Y = 78))]
Ожидаемый результат	Ближайшая частица в стартовой: Particle(double 0.2152087032784733, Point(int X = 310, int Y = 77))
Тип	**Позитивный**

Номер теста	$Б_{20}$
Объект тестирования	Модуль реализующий фильтр частиц (метод MeasurementUpdate(int count, Point StartPoint, List<Particle> newParticles))
Цель тестирования	Проверить имеет ли ближайшая частица к стартовой (переданной в метод) точке наибольший вес среди остальных частиц, если точка, вокруг которой перерасчитывается вес, имеет отрицательные координаты
Входные данные Ожидаемый результат	Стартовая точка: Point(int X = -300, int Y = 75), список частиц с весом List<Particle>: [Particle(double 0.2, Point(int X = 310, int Y = 77)), Particle(double 0.2, Point(int X = 303, int Y = 85)), Particle(double 0.2, Point(int X = 289, int Y = 97)), Particle(double 0.2, Point(int X = 280, int Y = 78)), Particle(double 0.2, Point(int X = 317, int Y = 78))] Выведется сообщение: "Некорректный формат точки!"
Тип	**Негативный**

Номер теста	$Б_{21}$
Объект тестирования	Модуль реализующий фильтр частиц (метод MeasurementUpdate(int count, Point StartPoint, List<Particle> newParticles))
Цель тестирования	Проверить имеет ли ближайшая частица к стартовой (переданной в метод) точке наибольший вес среди остальных частиц, если переданные частицы имеют отрицательные координаты
Входные данные	Стартовая точка: Point(int X = 300, int Y = 75), список частиц с весом List<Particle>: [Particle(double 0.2, Point(int X = -310, int Y = 77)), Particle(double 0.2, Point(int X = 303, int Y = 85)), Particle(double 0.2, Point(int X = 289, int Y = 97)), Particle(double 0.2, Point(int X = 280, int Y = 78)), Particle(double 0.2, Point(int X = 317, int Y = 78))]
Ожидаемый результат	Выведется сообщение: "Некорректный формат точки!"
Тип	**Негативный**

Номер теста	$Б_{22}$
Объект тестирования	Модуль реализующий фильтр частиц (метод AddParticles(int count, List<Particle> newParticles))
Цель тестирования	Проверить, добавляются ли частицы до исходного размера массива частиц (исходный размер будет составлять 15 элементов (int count = 15))
Входные данные	Количество частиц, которое необходимо восстановить: int count = 10, список частиц с весом List<Particle>: [Particle(double 0.2, Point(int X = 310, int Y = 77)), Particle(double 0.2, Point(int X = 303, int Y = 85)), Particle(double 0.2, Point(int X = 289, int Y = 97)), Particle(double 0.2, Point(int X = 280, int Y = 78)), Particle(double 0.2, Point(int X = 317, int Y = 78))]
Ожидаемый результат	Восстановленное количество частиц: int count = 15
Тип	**Позитивный**

Номер теста	$Б_{23}$
Объект тестирования	Модуль реализующий фильтр частиц (метод AddParticles(int count, List<Particle> newParticles))
Цель тестирования	Проверить, добавляются ли частицы до исходного размера массива частиц, если указано отрицательное количество частиц
Входные данные	Количество частиц, которое необходимо восстановить: int count = -10, список частиц с весом List<Particle>: [Particle(double 0.2, Point(int X = 310, int Y = 77)), Particle(double 0.2, Point(int X = 303, int Y = 85)), Particle(double 0.2, Point(int X = 289, int Y = 97)), Particle(double 0.2, Point(int X = 280, int Y = 78)), Particle(double 0.2, Point(int X = 317, int Y = 78))]
Ожидаемый результат	Выведется сообщение: "Некорректный количество точек!"
Тип	**Негативный**

Номер теста	$Б_{24}$
Объект тестирования	Модуль реализующий фильтр частиц (метод AddParticles(int count, List<Particle> newParticles))
Цель тестирования	Проверить, добавляются ли частицы до исходного размера массива частиц, если указаны отрицательные координаты у переданных частиц
Входные данные	Количество частиц, которое необходимо восстановить: int count = 10, список частиц с весом List<Particle>: [Particle(double 0.2, Point(int X = -310, int Y = 77)), Particle(double 0.2, Point(int X = 303, int Y = 85)), Particle(double 0.2, Point(int X = 289, int Y = 97)), Particle(double 0.2, Point(int X = 280, int Y = 78)), Particle(double 0.2, Point(int X = 317, int Y = 78))]
Ожидаемый результат	Выведется сообщение: "Некорректный формат точки!"
Тип	**Негативный**

Номер теста	$Б_{25}$
Объект тестирования	Модуль реализующий фильтр частиц (метод MeasurePosition(int count, Point StartPoint, List<Particle> newParticles))
Цель тестирования	Проверить, рассчитывается ли оценочная точка на основе переданного списка частиц
Входные данные	Список частиц с весом List<Particle>: [Particle(double 0.21, Point(int X = 310, int Y = 77)), Particle(double 0.23, Point(int X = 303, int Y = 85)), Particle(double 0.13, Point(int X = 289, int Y = 97)), Particle(double 0.22, Point(int X = 280, int Y = 78)), Particle(double 0.18, Point(int X = 317, int Y = 78))]
Ожидаемый результат	Оценочная точка: Point(int X = 291, int Y = 79)
Тип	**Позитивный**

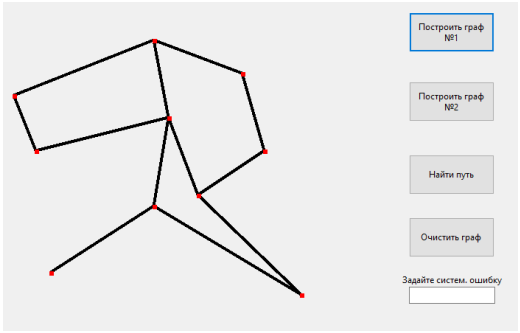
Номер теста	$Б_{26}$
Объект тестирования	Модуль реализующий фильтр частиц (метод MeasurePosition(int count, Point StartPoint, List<Particle> newParticles))
Цель тестирования	Проверить, рассчитывается ли оценочная точка на основе переданного списка частиц, если они имеют отрицательные координаты
Входные данные	Список частиц с весом List<Particle>: [Particle(double 0.21, Point(int X = -310, int Y = 77)), Particle(double 0.23, Point(int X = 303, int Y = 85)), Particle(double 0.13, Point(int X = 289, int Y = 97)), Particle(double 0.22, Point(int X = 280, int Y = 78)), Particle(double 0.18, Point(int X = 317, int Y = 78))]
Ожидаемый результат	Выведется сообщение: "Некорректный формат точки!"
Тип	**Негативный**

Номер теста	$Б_{27}$
Объект тестирования	Модуль построения траектории на основе оценочных точек (метод GetPointOnEdge())
Цель тестирования	Проверить, откладываются ли (добавляются ли) точки на выбранное ребро
Входные данные	Глобальный массив PointsFromFilter: List<Point>(Point (int X = 350, int Y = 79))
Ожидаемый результат	Глобальный массив PointsFromFilter: List<Point>(Point (int X = 251, int Y = 180))
Тип	**Позитивный**

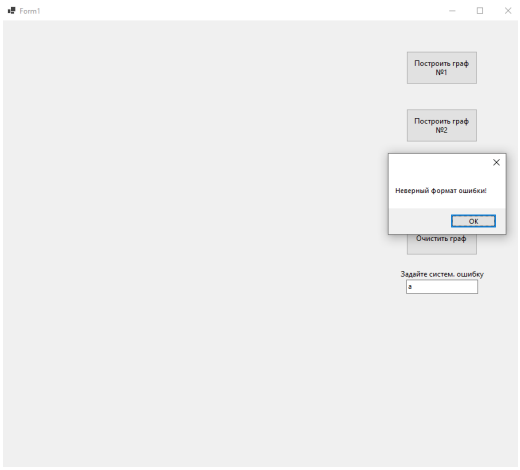
Номер теста	$Б_{28}$
Объект тестирования	Модуль построения траектории на основе оценочных точек (метод GetDistance(Point currentAveragePoint))
Цель тестирования	Проверить, рассчитываются ли расстояние между оценочной точкой и всеми ребрами исходного графа
Входные данные	Оценочная точка: Point currentAveragePoint(int X = 315, int Y = 90), датасет №1
Ожидаемый результат	Количество сформированных расстояний будет равно 10 (такое же, как и количество связей из датасета)
Тип	**Позитивный**

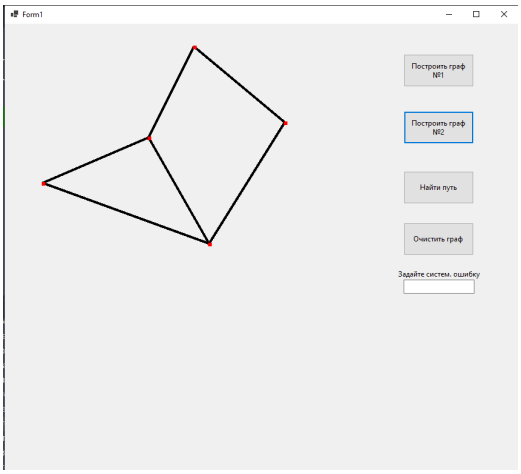
Номер теста	$Б_{29}$
Объект тестирования	Модуль построения траектории на основе оценочных точек (метод GetDistance(Point currentAveragePoint))
Цель тестирования	Проверить, рассчитываются ли расстояние между оценочной точкой и всеми ребрами исходного графа, если оценочная точка имеет отрицательные координаты
Входные данные	Оценочная точка: Point currentAveragePoint(int X = -315, int Y = 90), датасет №1
Ожидаемый результат	Выведется сообщение: "Некорректный формат точки"
Тип	**Негативный**

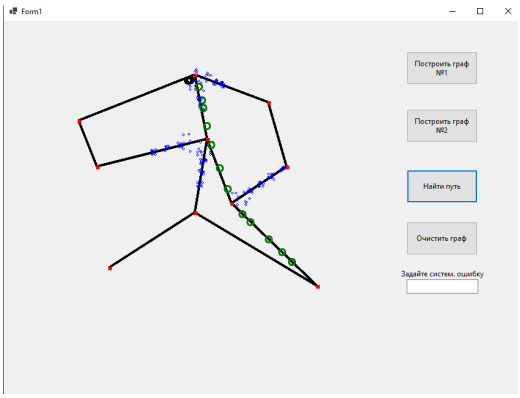
3.2 Аттестационное тестирование

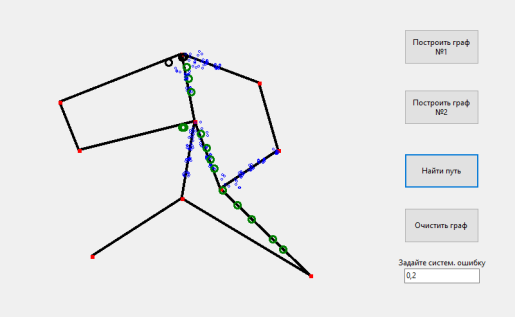
Номер теста	A_1
Объект тестирования	Модуль визуализации
Цель тестирования	Проверить отображение исходного графа по нажатию кнопки
Входные данные	1. Запуск программы (открывается пользовательская форма) 2. Нажатие кнопки «Построить граф №1»
Ожидаемый результат	Визуализация Графа №1 (датасет) на пользовательском интерфейсе 
Тип	**Позитивный**

Номер теста	A_2
Объект тестирования	Модуль визуализации
Цель тестирования	Проверить очищение области на которой отрисовывается граф
Входные данные	1. Запуск программы (открывается пользовательская форма) 2. Нажатие кнопки «Построить граф №1» 3. Нажатие кнопки «Очистить граф»
Ожидаемый результат	Область для рисования очистится
Тип	**Позитивный**

Номер теста	A_3
Объект тестирования	Модуль визуализации
Цель тестирования	Проверить наличие проверки на ввод пользовательской ошибки
Входные данные	1. Запуск программы (открывается пользовательская форма); 2. Ввод некорректной ошибки.
Ожидаемый результат	Предупреждение пользователя о неверном формате ввода ошибки 
Тип	**Негативный**

Номер теста	A_4
Объект тестирования	Модуль визуализации
Цель тестирования	Проверить отображение исходного второго графа по нажатию кнопки
Входные данные	1. Запуск программы (открывается пользовательская форма) 2. Нажатие кнопки «Построить граф №2»
Ожидаемый результат	Визуализация Графа №2 (датасет) на пользовательском интерфейсе 
Тип	**Позитивный**

Номер теста	A_5
Объект тестирования	Модуль визуализации
Цель тестирования	Проверить отображение рассчитанных траекторий
Входные данные	1. Запуск программы (открывается пользовательская форма) 2. Нажатие кнопки «Построить граф №1». 3. Нажатие кнопки «Найти путь»
Ожидаемый результат	Визуализация рассчитанных траекторий с привязкой к графу на пользовательском интерфейсе 
Тип	**Позитивный**

Номер теста	A_6
Объект тестирования	Модуль визуализации
Цель тестирования	Проверить возможность ввести пользовательскую ошибку
Входные данные	1. Запуск программы (открывается пользовательская форма) 2. Нажатие кнопки «Построить граф №1» 3. Ввод ошибки. 4. Нажатие кнопки «Найти путь»
Ожидаемый результат	Визуализация рассчитанных траекторий с привязкой к графу на пользовательском интерфейсе с пользовательской ошибкой 
Тип	**Позитивный**

3.3 Интеграционное тестирование

Блок связей №1

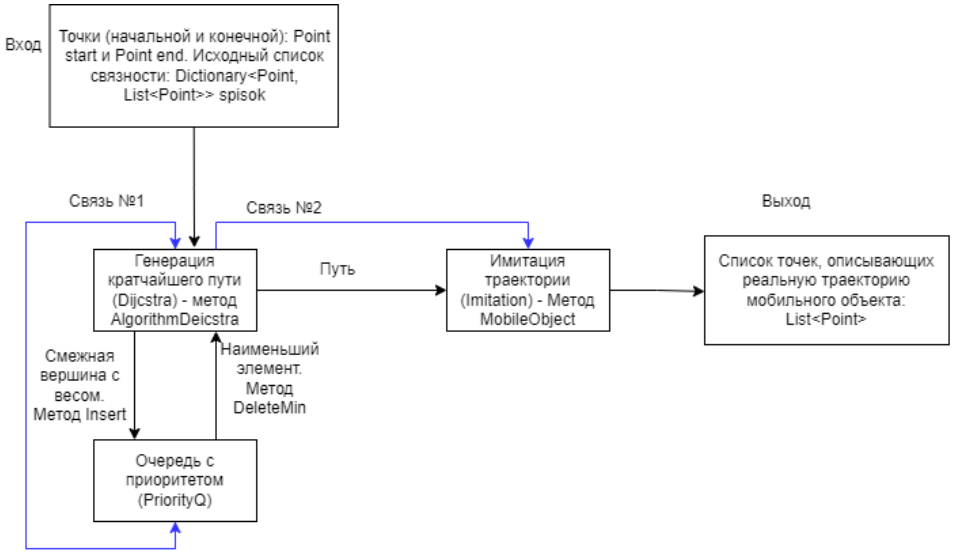


Рис. 3 – Блок связей №1

Номер теста	$И_1$
Объект тестирования	Блок связей №1
Цель тестирования	Проверить, что из сгенерированный кратчайший путь разбивается на реальную траекторию мобильного объекта
Описание тестирования	1. Создается объект класса Dijkstra и вызывается метод AlgorithmDeicstra(Point (int X = 300, 75)), Point (int X = 500, Y = 420)), Dictionary<Point, List<Point» spisok); 2. Создается объект класса Imitation и вызывается метод MobileObject(List<Point>)
Входные данные	Стартовая точка: Point (int X = 300, int Y = 75), конечная точка: Point (int X = 500, int Y = 420), словарь смежных ребер: (Dictionary<Point, List<Point»(датасет Граф №1))
Ожидаемый результат	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180), Point(int X = 338, int Y = 227), Point(int X = 352, int Y = 264), Point(int X = 360, int Y = 285), Point(int X = 395, int Y = 319), Point(int X = 426, int Y = 349), Point(int X = 451, int Y = 373), Point(int X = 482, int Y = 403), Point(int X = 500, int Y = 420)]. Значения могут разниться до 50 координат по каждой из осей, ввиду использования рандомайзера
Тип	**Позитивный**

Номер теста	$И_2$
Объект тестирования	Блок связей №1
Цель тестирования	Проверить, что из сгенерированный кратчайший путь не разбивается на реальную траекторию мобильного объекта, если начальная и конечная точка одинаковые
Описание тестирования	1. Создается объект класса Dijkstra и вызывается метод AlgorithmDeicstra(Point (int X = 300, 75)), Point (int X = 300, Y = 75)), Dictionary<Point, List<Point» spisok); 2. Создается объект класса Imitation и вызывается метод MobileObject(List<Point>)
Входные данные	Стартовая точка: Point (int X = 300, int Y = 75), конечная точка: Point (int X = 300, int Y = 75), словарь смежных ребер: (Dictionary<Point, List<Point»(датасет Граф №1))
Ожидаемый результат	Выведется сообщение: "Начальная и конечная точка совпадают!"
Тип	**Негативный**

Номер теста	$И_3$
Объект тестирования	Блок связей №1
Цель тестирования	Проверить, что из сгенерированный кратчайший путь не разбивается на реальную траекторию мобильного объекта, если переданы точки с отрицательными координатами
Описание тестирования	1. Создается объект класса Dijkstra и вызывается метод AlgorithmDeicstra(Point (int X = -300, 75)), Point (int X = 500, Y = 420)), Dictionary<Point, List<Point» spisok); 2. Создается объект класса Imitation и вызывается метод MobileObject(List<Point>)
Входные данные	Стартовая точка: Point (int X = -300, int Y = 75), конечная точка: Point (int X = 500, int Y = 420), словарь смежных ребер: (Dictionary<Point, List<Point»(датасет Граф №1))
Ожидаемый результат	Выведется сообщение: "Неверный формат точки!"
Тип	**Негативный**

Номер теста	$И_4$
Объект тестирования	Блок связей №1
Цель тестирования	Проверить, что из сгенерированный кратчайший путь не разбивается на реальную траекторию мобильного объекта, если начальная, конечная точка не соответствуют списку связности (1)
Описание тестирования	1. Создается объект класса Dijkstra и вызывается метод AlgorithmDeicstra(Point (int X = 213, int Y = 222)), Point (int X = 500, Y = 420)), Dictionary<Point, List<Point» spisok); 2. Создается объект класса Imitation и вызывается метод MobileObject(List<Point>)
Входные данные	Стартовая точка: Point (int X = 213, int Y = 222), конечная точка: Point (int X = 500, int Y = 420), словарь смежных ребер: (Dictionary<Point, List<Point»(датасет Граф №1))
Ожидаемый результат	Выведется сообщение: "Таких переданных точек нет в списке связности!"
Тип	**Негативный**

Блок связей №2

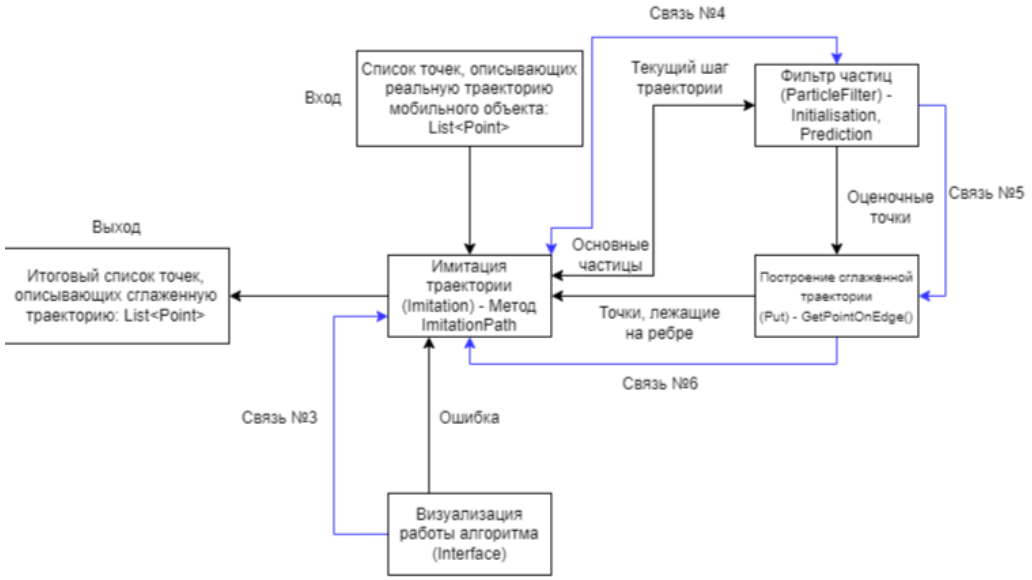


Рис. 4 – Блок связей №2

Номер теста	$И_1$
Объект тестирования	Блок связей №2
Цель тестирования	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу
Описание тестирования	1. Создается объект класса Imitation; 2. Создается объект класса ParticleFilter и вызывается метод Initialisation(int count = 100, Point StartPoint = (int X = 300, int Y = 75), int ogrX = -30, int ogrY = 30); 3. Вызывается метод ImitationPath(Start: Point(int X = 300, int Y = 75), End: Point(int X = 320, int Y = 180), ParticleFilter (на шаге №2), lastP: Point(int X = 300, int Y = 75), List<Point> points) 4. Задается пользовательская ошибка Imitation.Eps = 0.01
Входные данные	Стартовая точка: Point (int X = 300, int Y = 75), конечная точка: Point (int X = 320, int Y = 180), объект класса ParticleFilter на шаге №2, предыдушая точка: Point (int X = 300, int Y = 75), точки, составляющие реальную траекторию: List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]), Imitation.Eps = 0.01
Ожидаемый результат	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 301, int Y = 84), Point(int X = 307, int Y = 112), Point(int X = 310, int Y = 130), Point(int X = 313, int Y = 146)]. Значения могут разниться до 50 координат по каждой из осей, ввиду использования рандомайзера.
Тип	**Позитивный**

Номер теста	$И_2$
Объект тестирования	Блок связей №2
Цель тестирования	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если начальная, конечная и предыдущая точки имеют отрицательные координаты
Описание тестирования	1. Создается объект класса Imitation; 2. Создается объект класса ParticleFilter и вызывается метод Initialisation(int count = 100, Point StartPoint = (int X = 300, int Y = 75), int ogrX = -30, int ogrY = 30); 3. Вызывается метод ImitationPath(Start: Point(int X = -300, int Y = 75), End: Point(int X = 320, int Y = 180), ParticleFilter (на шаге №2), lastP: Point(int X = 300, int Y = 75), List<Point> points) 4. Задается пользовательская ошибка Imitation.Eps = 0.01
Входные данные	Стартовая точка: Point (int X = -300, int Y = 75), конечная точка: Point (int X = 320, int Y = 180), объект класса ParticleFilter на шаге №2, предыдушая точка: Point (int X = 300, int Y = 75), точки, составляющие реальную траекторию: List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]), Imitation.Eps = 0.01
Ожидаемый результат	Выведется сообщение: "Неверный формат точки!"
Тип	**Негативный**

Номер теста	$И_3$
Объект тестирования	Блок связей №2
Цель тестирования	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если список точек реальной траектории имеют отрицательные координаты
Описание тестирования	1. Создается объект класса Imitation; 2. Создается объект класса ParticleFilter и вызывается метод Initialisation(int count = 100, Point StartPoint = (int X = 300, int Y = 75), int ogrX = -30, int ogrY = 30); 3. Вызывается метод ImitationPath(Start: Point(int X = 300, int Y = 75), End: Point(int X = 320, int Y = 180), ParticleFilter (на шаге №2), lastP: Point(int X = 300, int Y = 75), List<Point> points) 4. Задается пользовательская ошибка Imitation.Eps = 0.01
Входные данные	Стартовая точка: Point (int X = 300, int Y = 75), конечная точка: Point (int X = 320, int Y = 180), объект класса ParticleFilter на шаге №2, предыдушая точка: Point (int X = 300, int Y = 75), точки, составляющие реальную траекторию: List<Point>: [Point(int X = -300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]), Imitation.Eps = 0.01
Ожидаемый результат	Выведется сообщение: "Неверный формат точки!"
Тип	**Негативный**

Номер теста	$И_4$
Объект тестирования	Блок связей №2
Цель тестирования	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если начальная и конечная точка совпадают
Описание тестирования	1. Создается объект класса Imitation; 2. Создается объект класса ParticleFilter и вызывается метод Initialisation(int count = 100, Point StartPoint = (int X = 300, int Y = 75), int ogrX = -30, int ogrY = 30); 3. Вызывается метод ImitationPath(Start: Point(int X = 300, int Y = 75), End: Point(int X = 320, int Y = 180), ParticleFilter (на шаге №2), lastP: Point(int X = 300, int Y = 75), List<Point> points) 4. Задается пользовательская ошибка Imitation.Eps = 0.01
Входные данные	Стартовая точка: Point (int X = 300, int Y = 75), конечная точка: Point (int X = 300, int Y = 75), объект класса ParticleFilter на шаге №2, предыдушая точка: Point (int X = 300, int Y = 75), точки, составляющие реальную траекторию: List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]), Imitation.Eps = 0.01
Ожидаемый результат	Выведется сообщение: "Начальная и конечная точка совпадают!"
Тип	**Негативный**

Номер теста	$И_5$
Объект тестирования	Блок связей №2
Цель тестирования	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если переданная ошибка имеет отрицательное значение
Описание тестирования	1. Создается объект класса Imitation; 2. Создается объект класса ParticleFilter и вызывается метод Initialisation(int count = 100, Point StartPoint = (int X = 300, int Y = 75), int ogrX = -30, int ogrY = 30); 3. Вызывается метод ImitationPath(Start: Point(int X = 300, int Y = 75), End: Point(int X = 320, int Y = 180), ParticleFilter (на шаге №2), lastP: Point(int X = 300, int Y = 75), List<Point> points) 4. Задается пользовательская ошибка Imitation.Eps = -0.01
Входные данные	Стартовая точка: Point (int X = 300, int Y = 75), конечная точка: Point (int X = 300, int Y = 75), объект класса ParticleFilter на шаге №2, предыдушая точка: Point (int X = 300, int Y = 75), точки, составляющие реальную траекторию: List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]), Imitation.Eps = -0.01
Ожидаемый результат	Выведется сообщение: "Некорректное значение пользовательской ошибки!"
Тип	**Негативный**

3.4 Специальное тестирование

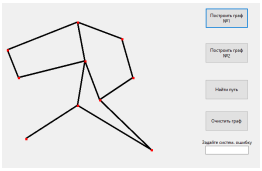
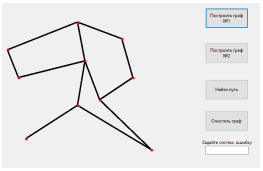
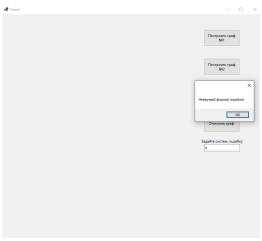
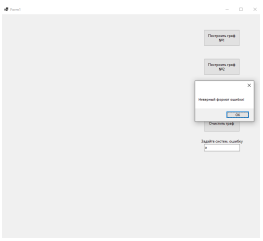
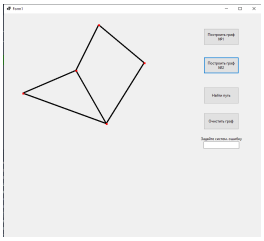
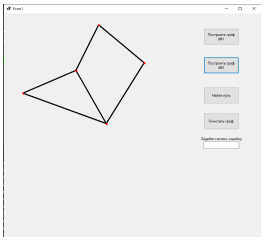
Номер теста	C_1
Объект тестирования	Время выполнения разработанного комплекса алгоритмов для решения исходной задачи
Цель тестирования	Определить время за которое рассчитывается сглаженная траектория
Входные данные	1. Создается объект класса Dijkstra и вызывается метод AlgorithmDeicstra(Point (int X = 300, 75)), Point (int X = 500, Y = 420)), Dictionary<Point, List<Point» spisok); 2. Создается объект класса Imitation и вызывается метод MobileObject(List<Point>) 3. Число точек: int count = 1000
Ожидаемый результат	Комплекс алгоритмов отработает менее чем за 50 млс

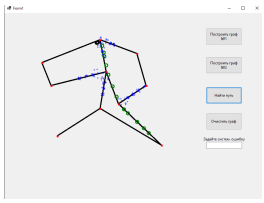
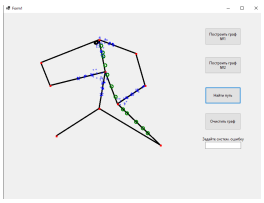
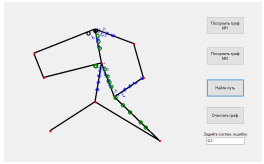
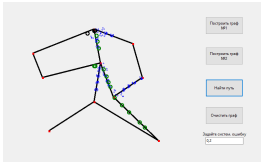
Номер теста	C_2
Объект тестирования	Время выполнения разработанного комплекса алгоритмов для решения исходной задачи
Цель тестирования	Определить время за которое рассчитывается сглаженная траектория
Входные данные	1. Создается объект класса Dijkstra и вызывается метод AlgorithmDeicstra(Point (int X = 300, 75)), Point (int X = 500, Y = 420)), Dictionary<Point, List<Point» spisok); 2. Создается объект класса Imitation и вызывается метод MobileObject(List<Point>) 3. Число точек: int count = 5000
Ожидаемый результат	Комплекс алгоритмов отработает менее чем за 100 млс

Номер теста	C_3
Объект тестирования	Время выполнения разработанного комплекса алгоритмов для решения исходной задачи
Цель тестирования	Определить время за которое рассчитывается сглаженная траектория
Входные данные	1. Создается объект класса Dijkstra и вызывается метод AlgorithmDeicstra(Point (int X = 300, 75)), Point (int X = 500, Y = 420)), Dictionary<Point, List<Point» spisok); 2. Создается объект класса Imitation и вызывается метод MobileObject(List<Point>) 3. Число точек: int count = 10000
Ожидаемый результат	Комплекс алгоритмов отработает менее чем за 150 млс

4 Журнал тестирования

4.1 Аттестационное тестирование

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
A1	1	Проверить отображение исходного графа по нажатию кнопки			Пройден
A2	1	Проверить очищение области на которой отрисовывается граф	Область для рисования очистится	Область для рисования очистится	Пройден
A3	1	Проверить наличие проверки на ввод пользовательской ошибки			Пройден
A4	1	Проверить отображение исходного второго графа по нажатию кнопки			Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
A5	1	Проверить отображение рассчитанных траекторий			Пройден
A6	1	Проверить возможность ввести пользовательскую ошибку			Пройден

4.2 Блочное тестирование

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б1	1	Проверить генерацию кратчайшей траектории между двумя точками	List<Point>:[Point(int X = 300, int Y = 75); Point(int X = 320, int Y = 180); Point (int X = 360, int Y = 285); Point (int X = 500, int Y = 420)]	List<Point>:[Point(int X = 300, int Y = 75); Point(int X = 320, int Y = 180); Point (int X = 360, int Y = 285); Point (int X = 500, int Y = 420)]	Пройден
Б2	1	Проверить правильность расчета расстояния между двумя точками	Расстояние между точками: double (204,26698215815497)	Расстояние между точками: double (204,26698215815497)	Пройден
Б3	1	Проверить создается ли очередь с приоритетом и добавляется ли туда вершина	Очередь содержит в себе 1 элемент(Tuple(Point(int X = 300, int Y = 75), 0.0)): int 1	Очередь содержит в себе 1 элемент(Tuple(Point(int X = 300, int Y = 75), 0.0)): int 1	Пройден
Б4	1	Проверить удаляется ли из очереди с приоритетом точка с наименьшим весом	Извлеченный кортеж: Tuple(Point(int X = 300, int Y = 75), 0.0	Извлеченный кортеж: Tuple(Point(int X = 300, int Y = 75), 0.0	Пройден
Б5	1	Проверить корректно ли будет извлечен элемент из очереди с отрицательным весом	Выведется сообщение: "Неверный формат элемента!"	Tuple(Point(int X = 300, int Y = 75), -120.0)	Не пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б5	2	Проверить корректно ли будет извлечен элемент из очереди с отрицательным весом	Выведется сообщение: "Неверный формат элемента!"	Выведется сообщение: "Неверный формат элемента!"	Пройден
Б6	1	Проверить корректно ли будет извлечен элемент из очереди с отрицательными координатами	Выведется сообщение: "Неверный формат элемента!"	Tuple(Point(int X = -300, int Y = 75), 10.0)	Не пройден
Б6	2	Проверить корректно ли будет извлечен элемент из очереди с отрицательными координатами	Выведется сообщение: "Неверный формат элемента!"	Выведется сообщение: "Неверный формат элемента!"	Пройден
Б7	1	Проверить рассчитывается ли вес (длину ребра) из двух точек с отрицательными координатами	Выведется сообщение: "Неверный формат точки!"	double (204,26698215815497)	Не пройден
Б7	2	Проверить рассчитывается ли вес (длину ребра) из двух точек с отрицательными координатами	Выведется сообщение: "Неверный формат точки!"	Выведется сообщение: "Неверный формат точки!"	Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б8	1	Проверить рассчитывается ли кратчаший путь если туда переданы точки (стартовая и конечная) с отрицательными координатами	Выведется сообщение: "Неверный формат точки!"	System.InvalidCastException: Unable to cast object of type'System.String' to type 'System.Double'	Не пройден
Б8	2	Проверить рассчитывается ли кратчаший путь если туда переданы точки (стартовая и конечная) с отрицательными координатами	Выведется сообщение: "Неверный формат точки!"	Выведется сообщение: "Неверный формат точки!"	Пройден
Б9	1	Разбивается ли путь на траекторию мобильного объекта	Массив точек List<Point>: Ожидаемый массив	Массив точек List<Point>: Фактический массив	Пройден
Б10	1	Разбивается ли путь на траекторию, если в переданном в метод массиве точки имеют отрицательные координаты	Выведется сообщение: "Неверный формат точки!"	List<Point>, состоящий из 39 элементов, включая точки с отрицательными координатами	Не пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б10	2	Разбивается ли путь на траекторию, если в переданном в метод массиве точки имеют отрицательные координаты	Выведется сообщение: "Неверный формат точки!"	Сообщение: "Неверный формат точки!"	Пройден
Б11	1	Сглаживается ли ошибочная траектория по отношению к графу	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 301, int Y = 84), Point(int X = 307, int Y = 112), Point(int X = 310, int Y = 130), Point(int X = 313, int Y = 146)]. Значения могут разниться до 50 координат по каждой из осей, ввиду использования рандомайзера.	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 301, int Y = 81), Point(int X = 304, int Y = 101), Point(int X = 308, int Y = 121), Point(int X = 310, int Y = 130)].	Пройден
Б12	1	Сглаживается ли сымитированная траектория с ошибкой, если передан массив с точками, имеющими отрицательные координаты	Выведется сообщение: "Неверный формат точки!"	System.Collections.Generic. KeyNotFoundException: The given key '(X=-300,Y=75, X=320,Y=180)' was not present in the dictionary (отсутствие такой связи в датасете 1)	Не пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б12	2	Сглаживается ли сымитированная траектория с ошибкой, если передан массив с точками, имеющими отрицательные координаты	Выведется сообщение: "Неверный формат точки!"	Сообщение: "Неверный формат точки!"	Пройден
Б13	1	Проверить создаются ли частицы вокруг переданной точки (point StartPoint)	Массив частиц List<Particle>: [Point(int X = 310, int Y = 77), Point(int X = 303, int Y = 85), Point(int X = 289, int Y = 97), Point(int X = 280, int Y = 78), Point(int X = 317, int Y = 78)] Значения могут разниться до 30 координат по каждой из осей, ввиду использования рандомайзера.	Массив частиц List<Particle>: [Point(int X = 274, int Y = 59), Point(int X = 292, int Y = 45), Point(int X = 329, int Y = 53), Point(int X = 310, int Y = 70), Point(int X = 327, int Y = 88)]	Пройден
Б14	1	Проверить создаются ли частицы вокруг переданной точки (point StartPoint), если количество частиц имеет отрицательное значение	Выведется сообщение: "Некорректный формат или количество точек!"	Создаст пустой массив частиц: List<Particle>	Не пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б14	2	Проверить создаются ли частицы вокруг переданной точки (point StartPoint), если количество частиц имеет отрицательное значение	Выведется сообщение: "Некорректный формат или количество точек!"	Выведется сообщение: "Некорректный формат или количество точек!"	Пройден
Б15	1	Проверить создаются ли частицы вокруг переданной точки (point StartPoint), если переданная точка имеет некорректный формат (отрицательные координаты)	Выведется сообщение: "Некорректный формат или количество точек!"	Создаст массив частиц, содержащий точки с отрицательными координатами	Не пройден
Б15	2	Проверить создаются ли частицы вокруг переданной точки (point StartPoint), если переданная точка имеет некорректный формат (отрицательные координаты)	Выведется сообщение: "Некорректный формат или количество точек!"	Выведется сообщение: "Некорректный формат или количество точек!"	Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б16	1	Проверить обновляются ли список частиц на основании переданной длины шага и угла поворота	Массив частиц List<Particle>: Point(int X = 257, int Y = 33), Point(int X = 291, int Y = 74), Point(int X = 306, int Y = 72), Point(int X = 301, int Y = 41), Point(int X = 332, int Y = 117). Значения могут разниться до 30 координат по каждой из осей, ввиду использования рандомайзера.	Массив частиц List<Particle>: Point(int X = 282, int Y = 47), Point(int X = 289, int Y = 85), Point(int X = 317, int Y = 93), Point(int X = 298, int Y = 55), Point(int X = 305, int Y = 109).	Пройден
Б17	1	Проверить обновляются ли список частиц (на этапе предсказания), если заданная точка имеет отрицательные координаты	Выведется сообщение: "Некорректный формат точки или длину шага!"	Создаст массив частиц, содержащий точки с отрицательными координатами	Не пройден
Б17	2	Проверить обновляются ли список частиц (на этапе предсказания), если заданная точка имеет отрицательные координаты	Выведется сообщение: "Некорректный формат точки или длину шага!"	Соощение: "Некорректный формат точки или длину шага!"	Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б18	1	Проверить обновляются ли список частиц (на этапе предсказания), если длина шага имеет отрицательное значение	Выведется сообщение: "Некорректный формат точки или длину шага!"	Создаст массив частиц, содержащий точки с отрицательными координатами	Не пройден
Б18	2	Проверить обновляются ли список частиц (на этапе предсказания), если длина шага имеет отрицательное значение	Выведется сообщение: "Некорректный формат точки или длину шага!"	Соощение: "Некорректный формат точки или длину шага!"	Пройден
Б19	1	Проверить имеет ли ближайшая частица к стартовой (переданной в метод) точке наибольший вес среди остальных частиц	Ближайшая частица в стартовой: Particle(double 0.2152087032784733, Point(int X = 310, int Y = 77))	Particle(double 0.2152087032784733, Point(int X = 310, int Y = 77))	Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б20	1	Проверить имеет ли ближайшая частица к стартовой (переданной в метод) точке наибольший вес среди остальных частиц, если точка, вокруг которой перерасчитывается вес, имеет отрицательные координаты	Выведется сообщение: "Некорректный формат точки!"	Вес не будет рассчитан и будет иметь значение NaN	Не пройден
Б20	2	Проверить имеет ли ближайшая частица к стартовой (переданной в метод) точке наибольший вес среди остальных частиц, если точка, вокруг которой перерасчитывается вес, имеет отрицательные координаты	Выведется сообщение: "Некорректный формат точки!"	Выведется сообщение: "Некорректный формат точки!"	Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б21	1	Проверить имеет ли ближайшая частица к стартовой (переданной в метод) точке наибольший вес среди остальных частиц, если переданные частицы имеют отрицательные координаты	Выведется сообщение: "Некорректный формат точки!"	У частиц с отрицательными координатами будет нулевой вес	Не пройден
Б21	2	Проверить имеет ли ближайшая частица к стартовой (переданной в метод) точке наибольший вес среди остальных частиц, если переданные частицы имеют отрицательные координаты	Выведется сообщение: "Некорректный формат точки!"	Выведется сообщение: "Некорректный формат точки!"	Пройден
Б22	1	Проверить, добавляются ли частицы до исходного размера массива частиц (исходный размер будет составлять 15 элементов (int count = 15))	Восстановленное количество частиц: int count = 15	int count = 15	Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б23	1	Проверить, добавляются ли частицы до исходного размера массива частиц, если указано отрицательное количество частиц	Выведется сообщение: "Некорректный количество точек!"	Будет возвращен исходный массив без рассчитанных весов (NaN)	Не пройден
Б23	2	Проверить, добавляются ли частицы до исходного размера массива частиц, если указано отрицательное количество частиц	Выведется сообщение: "Некорректный количество точек!"	Выведется сообщение: "Некорректный количество точек!"	Пройден
Б24	1	Проверить, добавляются ли частицы до исходного размера массива частиц, если указаны отрицательные координаты у переданных частиц	Выведется сообщение: "Некорректный формат точки!"	Будет возвращен список частиц, юмеющие отрицательные координаты и нерассчитанный вес (NaN)	Не пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б24	2	Проверить, добавляются ли частицы до исходного размера массива частиц, если указаны отрицательные координаты у переданных частиц	Выведется сообщение: "Некорректный формат точки!"	Сообщение: "Некорректный формат точки!"	Пройден
Б25	1	Проверить, рассчитывается ли оценочная точка на основе переданного списка частиц	Оценочная точка: Point(int X = 291, int Y = 79)	Point(int X = 291, int Y = 79)	Пройден
Б26	1	Проверить, рассчитывается ли оценочная точка на основе переданного списка частиц	Выведется сообщение: "Некорректный формат точки!"	Оценочная точка будет иметь нулевые координаты (Point(int X = 0, int Y = 0))	Не пройден
Б26	2	Проверить, рассчитывается ли оценочная точка на основе переданного списка частиц	Выведется сообщение: "Некорректный формат точки!"	Сообщение: "Некорректный формат точки!"	Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
Б27	1	Проверить, откладываются ли (добавляются ли) точки на выбранное ребро	Глобальный массив PointsFromFilter: List<Point>(int X = 251, int Y = 180)	List<Point>(int X = 251, int Y = 180)	Пройден
Б28	1	Проверить, рассчитываются ли расстояние между оценочной точкой и всеми ребрами исходного графа	Количество сформированных расстояний будет равно 10 (такое же, как и количество ребер из датасета)	Количество сформированных расстояний будет равно 10 (такое же, как и количество связей из датасета)	Пройден
Б29	1	Проверить, рассчитываются ли расстояние между оценочной точкой и всеми ребрами исходного графа, если оценочная точка имеет отрицательные координаты	Выведется сообщение: "Некорректный формат точки"	Расстояние будут рассчитаны некорректно (иметь большие значения)	Не пройден
Б29	2	Проверить, рассчитываются ли расстояние между оценочной точкой и всеми ребрами исходного графа, если оценочная точка имеет отрицательные координаты	Выведется сообщение: "Некорректный формат точки!"	Сообщение: "Некорректный формат точки!"	Пройден

4.3 Интеграционное тестирование

Блок №1

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
И1	1	Проверить, что сгенерированный кратчайший путь разбивается на реальную траекторию мобильного объекта	Ожидаемый массив	Фактический массив	Пройден
И2	1	Проверить, что из сгенерированный кратчайший путь не разбивается на реальную траекторию мобильного объекта, если начальная и конечная точка одинаковые	Выведется сообщение: "Начальная и конечная точка совпадают!"	List<Point>: Point(int X = 300, int Y = 75)	Не пройден
И2	2	Проверить, что из сгенерированный кратчайший путь не разбивается на реальную траекторию мобильного объекта, если начальная и конечная точка одинаковые	Выведется сообщение: "Начальная и конечная точка совпадают!"	Выведется сообщение: "Начальная и конечная точка совпадают!"	Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
И3	1	Проверить, что из сгенерированный кратчайший путь не разбивается на реальную траекторию мобильного объекта, если переданы точки с отрицательными координатами	Выведется сообщение: "Неверный формат точки!"	Выведется сообщение: "Неверный формат точки!"	Пройден
И4	1	Проверить, что из сгенерированный кратчайший путь не разбивается на реальную траекторию мобильного объекта, если начальная, конечная точка не соответствуют списку связности (1)	Выведется сообщение: "Таких переданных точек нет в списке связности!"	System.Collections.Generic. KeyNotFoundException: "The given key 'X=500,Y=420' was not present in the dictionary."	Не пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
И4	2	Проверить, что из сгенерированный кратчайший путь не разбивается на реальную траекторию мобильного объекта, если начальная, конечная точка не соответствуют списку связности (1)	Выведется сообщение: "Таких переданных точек нет в списке связности!"	Выведется сообщение: "Таких переданных точек нет в списке связности!"	Пройден

Блок №2

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
И1	1	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 301, int Y = 84), Point(int X = 307, int Y = 112), Point(int X = 310, int Y = 130), Point(int X = 313, int Y = 146)]. Значения могут разниться до 50 координат по каждой из осей, ввиду использования рандомайзера.	Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 301, int Y = 81), Point(int X = 304, int Y = 101), Point(int X = 308, int Y = 121), Point(int X = 310, int Y = 130)].	Пройден
И2	1	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если начальная, конечная и предыдущая точки имеют отрицательные координаты	Выведется сообщение: "Неверный формат точки!"	Выведется сообщение: "Неверный формат точки!"	Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
И3	1	Проверить, что траектория мобильного объекта имитируется и сглаживается по отношению к графу, если точки реальной траектории имеют отрицательные координаты	Выведется сообщение: "Неверный формат точки!"	Выведется сообщение: "Неверный формат точки!"	Пройден
И4	1	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если начальная и конечная точка совпадают	Выведется сообщение: "Начальная и конечная точка совпадают!"	System.Collections.Generic. KeyNotFoundException: "The given key '(X=300,Y=75, X=300,Y=75)' was not present in the dictionary."	Не пройден
И4	2	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если начальная и конечная точка совпадают	Выведется сообщение: "Начальная и конечная точка совпадают!"	Выведется сообщение: "Начальная и конечная точка совпадают!"	Пройден

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
И5	1	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если переданная ошибка имеет отрицательное значение	Выведется сообщение: "Некорректное значение пользовательской ошибки!"	Некорректный список точек: List<Point>	Не пройден
И5	2	Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если переданная ошибка имеет отрицательное значение	Выведется сообщение: "Некорректное значение пользовательской ошибки!"	Выведется сообщение: "Некорректное значение пользовательской ошибки!"	Пройден

4.4 Специальное тестирование

ID	Попытка	Описание	Ожидаемый результат	Фактический результат	Результат
C1	1	Определить время за которое рассчитывается сглаженная траектория	Комплекс алгоритмов отработает менее чем за 50 млс	27 млс	Пройден
C2	1	Определить время за которое рассчитывается сглаженная траектория	Комплекс алгоритмов отработает менее чем за 100 млс	59 млс	Пройден
C3	1	Определить время за которое рассчитывается сглаженная траектория	Комплекс алгоритмов отработает менее чем за 150 млс	100 млс	Пройден

5 Журнал ошибок

Ошибка №1

1. Тест: $Б_7$
2. Цель тестирования: Проверить рассчитывается ли вес (длину ребра) из двух точек с отрицательными координатами
3. Входные данные: Стартовая точка: Point(int X = -300, int Y = -75), Point(int X = -110, int Y = -150)))
4. Ожидаемый результат: Выведется сообщение: "Неверный формат точки!"
5. Фактический результат: double (204,26698215815497)
6. Исправление: Была добавлена проверка на передаваемые значения координат точки

Ошибка №2

1. Тест: $Б_8$
2. Цель тестирования: Проверить рассчитывается ли кратчаший путь если туда переданы точки (стартовая и конечная) с отрицательными координатами
3. Входные данные: Генерация кратчайшего пути, вызывается метод AlgorithmDeicstra(Point(X = -300, int Y = 75), Point(int X = 500, int Y = 420)), Dictionary<Point, List<Point»(датасет Граф №1))
4. Ожидаемый результат: Выведется сообщение: "Неверный формат точки!"
5. Фактический результат: System.InvalidCastException: Unable to cast object of type 'System.String' to type 'System.Double'
6. Исправление: Была добавлена проверка на передаваемые значения координат точки

Ошибка №3

1. Тест: $Б_5$
2. Цель тестирования: Проверить корректно ли будет извлечен элемент из очереди с отрицательным весом

3. Входные данные: Кортеж, содержащий точку и вес: Tuple(Point(int X = 300, int Y = 75), -120.0)
4. Ожидаемый результат: Выведется сообщение: "Неверный формат элемента!"
5. Фактический результат: Tuple(Point(int X = 300, int Y = 75), -120.0)
6. Исправление: Была добавлена проверка на передаваемые значения элемента (вес точки)

Ошибка №4

1. Тест: $Б_6$
2. Цель тестирования: Проверить корректно ли будет извлечен элемент из очереди с отрицательными координатами
3. Входные данные: Кортеж, содержащий точку и вес: Tuple(Point(-300, 75), 10.0)
4. Ожидаемый результат: Выведется сообщение: "Неверный формат элемента!"
5. Фактический результат: Tuple(Point(int X = -300, int Y = 75), 10.0)
6. Исправление: Была добавлена проверка на передаваемые значения элемента (координаты точки)

Ошибка №5

1. Тест: $Б_{10}$
2. Цель тестирования:Разбивается ли путь на траекторию, если в переданном в метод массиве точки имеют отрицательные координаты
3. Входные данные: Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = -320, int Y = 180), Point(int X = 360, int Y = 285), Point(int X = 500, int Y = 420)]
4. Ожидаемый результат: Выведется сообщение: "Неверный формат точки!"
5. Фактический результат: List<Point>, состоящий из 39 элементов, включая точки с отрицательными координатами
6. Исправление: Была добавлена проверка на передаваемые значения точки

Ошибка №6

1. Тест: $Б_{12}$
2. Цель тестирования: Сглаживается ли сымитированная траектория с ошибкой, если передан массив с точками, имеющими отрицательные координаты
3. Входные данные: Массив точек List<Point>: [Point(int X = -300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]
4. Ожидаемый результат: Выведется сообщение: "Неверный формат точки!"
5. Фактический результат: System.Collections.Generic.KeyNotFoundException: The given key '(X=-300,Y=75, X=320,Y=180)' was not present in the dictionary (отсутствие такой связи в датасете 1)
6. Исправление: Была добавлена проверка на передаваемые значения точки

Ошибка №7

1. Тест: $Б_{14}$
2. Цель тестирования: Проверить создаются ли частицы вокруг переданной точки (point StartPoint), если количество частиц имеет отрицательное значение
3. Входные данные: Массив точек List<Point>: [Point(int X = -300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]
4. Ожидаемый результат: Выведется сообщение: "Некорректный формат или количество точек!"
5. Фактический результат: Создаст пустой массив частиц: List<Particle>
6. Исправление: Была добавлена проверка на передаваемое значение количества точек

Ошибка №8

1. Тест: $Б_{15}$
2. Цель тестирования: Проверить создаются ли частицы вокруг переданной точки (point StartPoint), если переданная точка имеет некорректный формат (отрицательные координаты)

3. Входные данные: Количество точек: double count = 5, Стартовая точка: Point(int X = -300, int Y = 75)

4. Ожидаемый результат: Выведется сообщение: "Некорректный формат или количество точек!"

5. Фактический результат: Создаст массив частиц, содержащий точки с отрицательными координатами

6. Исправление: Была добавлена проверка на передаваемый формат точек

Ошибка №9

1. Тест: $Б_{17}$

2. Цель тестирования: Проверить обновляются ли список частиц (на этапе предсказания), если заданная точка имеет отрицательные координаты

3. Входные данные: Стартовая точка: Point(int X = -300, int Y = 75), длина: double length = 30, угол: double angle = 0.009. Перед вызовом самого метода Prediction, необходимо вызвать метод Initialisation(5, Point(int X = 300, int Y = 75), -30, +30) для первоначальной генерации частиц.

4. Ожидаемый результат: Выведется сообщение: "Некорректный формат точки или длину шага!"

5. Фактический результат: Создаст массив частиц, содержащий точки с отрицательными координатами

6. Исправление: Была добавлена проверка на передаваемый формат точек

Ошибка №10

1. Тест: $Б_{18}$

2. Цель тестирования: Проверить обновляются ли список частиц (на этапе предсказания), если длина шага имеет отрицательное значение

3. Входные данные: Стартовая точка: Point(int X = -300, int Y = 75), длина: double length = 30, угол: double angle = 0.009. Перед вызовом самого метода Prediction, необходимо вызвать метод Initialisation(5, Point(int X = 300, int Y = 75), -30, +30) для первоначальной генерации частиц.

4. Ожидаемый результат: Стартовая точка: Point(int X = 300, int Y = 75), длина: double length = -30, угол: double angle = 0.009. Перед вызовом самого метода Prediction, необходимо вызвать метод Initialisation(5, Point(int X = 300, int Y = 75), -30, +30) для первоначальной генерации частиц.

5. Фактический результат: Создаст массив частиц, содержащий точки с отрицательными координатами

6. Исправление: Была добавлена проверка на длину шага

Ошибка №11

1. Тест: $Б_{20}$

2. Цель тестирования: Проверить имеет ли ближайшая частица к стартовой (переданной в метод) точке наибольший вес среди остальных частиц, если точка, вокруг которой перерасчитывается вес, имеет отрицательные координаты

3. Входные данные: Стартовая точка: Point(int X = -300, int Y = 75), список частиц с весом List<Particle>: [Particle(double 0.2, Point(int X = 310, int Y = 77)), Particle(double 0.2, Point(int X = 303, int Y = 85)), Particle(double 0.2, Point(int X = 289, int Y = 97)), Particle(double 0.2, Point(int X = 280, int Y = 78)), Particle(double 0.2, Point(int X = 317, int Y = 78))]

4. Ожидаемый результат: Выведется сообщение: "Некорректный формат точки!"

5. Фактический результат: Вес не будет рассчитан и будет иметь значение NaN

6. Исправление: Была добавлена проверка на передаваемый формат точек

Ошибка №12

1. Тест: $Б_{21}$

2. Цель тестирования: Проверить имеет ли ближайшая частица к стартовой (переданной в метод) точке наибольший вес среди остальных частиц, если переданные частицы имеют отрицательные координаты

3. Входные данные: Стартовая точка: Point(int X = 300, int Y = 75), список частиц с весом List<Particle>: [Particle(double 0.2, Point(int X = -310, int Y = 77)), Particle(double 0.2, Point(int X = 303, int Y = 85)), Particle(double 0.2, Point(int X = 289, int Y =

97)), Particle(double 0.2, Point(int X = 280, int Y = 78)), Particle(double 0.2, Point(int X = 317, int Y = 78))]

4. Ожидаемый результат: Выведется сообщение: "Некорректный формат точки!"

5. Фактический результат: У частиц с отрицательными координатами будет нулевой вес

6. Исправление: Была добавлена проверка на передаваемый формат точек

Ошибка №13

1. Тест: $Б_{23}$

2. Цель тестирования: Проверить, добавляются ли частицы до исходного размера массива частиц, если указано отрицательное количество частиц

3. Входные данные: Количество частиц, которое необходимо восстановить: int count = -10, список частиц с весом List<Particle>: [Particle(double 0.2, Point(int X = 310, int Y = 77)), Particle(double 0.2, Point(int X = 303, int Y = 85)), Particle(double 0.2, Point(int X = 289, int Y = 97)), Particle(double 0.2, Point(int X = 280, int Y = 78)), Particle(double 0.2, Point(int X = 317, int Y = 78))]

4. Ожидаемый результат: Выведется сообщение: "Некорректный количество точек!"

5. Фактический результат: Будет возвращен исходный массив без рассчитанных весов (NaN)

6. Исправление: Была добавлена проверка на передаваемое количество точек

Ошибка №14

1. Тест: $Б_{24}$

2. Цель тестирования: Проверить, добавляются ли частицы до исходного размера массива частиц, если указаны отрицательные координаты у переданных частиц

3. Входные данные: Количество частиц, которое необходимо восстановить: int count = 10, список частиц с весом List<Particle>: [Particle(double 0.2, Point(int X = -310, int Y = 77)), Particle(double 0.2, Point(int X = 303, int Y = 85)), Particle(double 0.2, Point(int X = 289, int Y = 97)), Particle(double 0.2, Point(int X = 280, int Y = 78)), Particle(double 0.2, Point(int X = 317, int Y = 78))]

4. Ожидаемый результат: Выведется сообщение: "Некорректный формат точки!"

5. Фактический результат: Будет возвращен список частиц, юмеющие отрицательные координаты и нерассчитанный вес (NaN)

6. Исправление: Была добавлена проверка на передаваемый формат точек

Ошибка №15

1. Тест: $Б_{26}$

2. Цель тестирования: Проверить, рассчитывается ли оценочная точка на основе переданного списка частиц, если они имеют отрицательные координаты

3. Входные данные: Список частиц с весом List<Particle>: [Particle(double 0.21, Point(int X = -310, int Y = 77)), Particle(double 0.23, Point(int X = 303, int Y = 85)), Particle(double 0.13, Point(int X = 289, int Y = 97)), Particle(double 0.22, Point(int X = 280, int Y = 78)), Particle(double 0.18, Point(int X = 317, int Y = 78))]

4. Ожидаемый результат: Выведется сообщение: "Некорректный формат точки!"

5. Фактический результат: Оценочная точка будет иметь нулевые координаты (Point(int X = 0, int Y = 0))

6. Исправление: Была добавлена проверка на передаваемый формат точек

Ошибка №16

1. Тест: $Б_{29}$

2. Цель тестирования: Проверить, рассчитываются ли расстояние между оценочной точкой и всеми ребрами исходного графа, если оценочная точка имеет отрицательные координаты

3. Входные данные: Оценочная точка: Point currentAveragePoint(int X = -315, int Y = 90), датасет №1

4. Ожидаемый результат: Выведется сообщение: "Некорректный формат точки!"

5. Фактический результат: Расстояние будут рассчитаны некорректно (иметь большие значения)

6. Исправление: Была добавлена проверка на передаваемый формат точек

Ошибка №17

1. Тест: $И_2$
2. Цель тестирования: Проверить, что из сгенерированный кратчайший путь не разбивается на реальную траекторию мобильного объекта, если начальная и конечная точка одинаковые
3. Входные данные: Стартовая точка: Point (int X = 300, int Y = 75), конечная точка: Point (int X = 300, int Y = 75), словарь смежных ребер: (Dictionary<Point, List<Point»(датасет Граф №1))
4. Ожидаемый результат: Выведется сообщение: "Начальная и конечная точка совпадают!"
5. Фактический результат: List<Point>: Point(int X = 300, int Y = 75) - стартовая точка
6. Исправление: Была добавлена проверка на совпадение точек

Ошибка №18

1. Тест: $И_4$
2. Цель тестирования: Проверить, что из сгенерированный кратчайший путь не разбивается на реальную траекторию мобильного объекта, если начальная, конечная точка не соответствуют списку связности (1)
3. Входные данные: Стартовая точка: Point (int X = 213, int Y = 222), конечная точка: Point (int X = 500, int Y = 420), словарь смежных ребер: (Dictionary<Point, List<Point»(датасет Граф №1))
4. Ожидаемый результат: Выведется сообщение: "Таких переданных точек нет в списке связности!"
5. Фактический результат: System.Collections.Generic. KeyNotFoundException: "The given key 'X=500,Y=420' was not present in the dictionary."
6. Исправление: Была добавлена проверка на совпадение точек со списком связности

System.Collections.Generic. KeyNotFoundException: "The given key '(X=300,Y=75, X=300,Y=75)' was not present in the dictionary."

Ошибка №19

1. Тест: $И_4$

2. Цель тестирования: Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если начальная и конечная точка совпадают

3. Входные данные: Стартовая точка: Point (int X = 300, int Y = 75), конечная точка: Point (int X = 300, int Y = 75), объект класса ParticleFilter на шаге №2, предыдушая точка: Point (int X = 300, int Y = 75), точки, составляющие реальную траекторию: List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]), Imitation.Eps = 0.01

4. Ожидаемый результат: Выведется сообщение: "Начальная и конечная точка совпадают!"

5. Фактический результат: System.Collections.Generic. KeyNotFoundException: "The given key 'X=500,Y=420' was not present in the dictionary."

6. Исправление: Была добавлена проверка на совпадение точек

Ошибка №20

1. Тест: $И_5$

2. Цель тестирования: Проверить, что реальная траектория мобильного объекта имитируется и сглаживается по отношению к графу, если переданная ошибка имеет отрицательное значение

3. Входные данные: Стартовая точка: Point (int X = 300, int Y = 75), конечная точка: Point (int X = 300, int Y = 75), объект класса ParticleFilter на шаге №2, предыдушая точка: Point (int X = 300, int Y = 75), точки, составляющие реальную траекторию: List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 305, int Y = 104), Point(int X = 311, int Y = 133), Point(int X = 319, int Y = 175), Point(int X = 320, int Y = 180)]), Imitation.Eps = -0.01

4. Ожидаемый результат: Выведется сообщение: "Некорректное значение пользовательской ошибки!"

5. Фактический результат: Некорректный список точек: List<Point>

6. Исправление: Была добавлена проверка на пользовательскую ошибку

6 Примеры тестов

Пример блочного положительного теста:

```
[TestMethod]
| Ссылок: 0
public void TestMethodDeleteMinPositive()
{
    var result = Tuple.Create(new Point { X = 300, Y = 75 }, 0.0);
    PriorityQ d = new PriorityQ();
    var tu = Tuple.Create(new Point { X = 300, Y = 75 }, 0.0);
    var tu2 = Tuple.Create(new Point { X = 320, Y = 222 }, 0.0);
    d.Insert(tu);
    d.Insert(tu2);
    var result2 = d.DeleteMin();
    Assert.AreEqual(result, (Tuple<Point, double>)result2);
}
```

Рис. 5 – Блочный тест №$Б_4$

Пример блочного негативного теста:

```
[TestMethod]
| Ссылок: 0
public void TestMethodInsertDeleteMinNegative2()
{
    string result = "Неверный формат элемента!";
    PriorityQ d = new PriorityQ();
    var tu = Tuple.Create(new Point { X = -300, Y = 75 }, 10.0);
    d.Insert(tu);
    var result2 = d.DeleteMin();
    Assert.AreEqual(result, result2);
}
```

Рис. 6 – Блочный тест №$Б_6$

Пример интеграционного положительного теста:

```
    var points = new List<Point>();
    var spisokSmezhnosti = new Dictionary<Point, List<Point>>();

    points.Add(new Point { X = 300, Y = 75 });
    points.Add(new Point { X = 110, Y = 150 });
    points.Add(new Point { X = 420, Y = 120 });
    points.Add(new Point { X = 320, Y = 180 });
    points.Add(new Point { X = 140, Y = 225 });
    points.Add(new Point { X = 450, Y = 225 });
    points.Add(new Point { X = 300, Y = 300 });
    points.Add(new Point { X = 360, Y = 285 });
    points.Add(new Point { X = 160, Y = 390 });
    points.Add(new Point { X = 500, Y = 420 });

    spisokSmezhnosti.Add(points[0], new List<Point> { points[1], points[2], points[3] });
    spisokSmezhnosti.Add(points[1], new List<Point> { points[4], points[0] });
    spisokSmezhnosti.Add(points[2], new List<Point> { points[5], points[0] });
    spisokSmezhnosti.Add(points[3], new List<Point> { points[4], points[6], points[7], points[0] });
    spisokSmezhnosti.Add(points[4], new List<Point> { points[1], points[3] });
    spisokSmezhnosti.Add(points[5], new List<Point> { points[7], points[2] });
    spisokSmezhnosti.Add(points[6], new List<Point> { points[8], points[9], points[3] });
    spisokSmezhnosti.Add(points[7], new List<Point> { points[9], points[5], points[3] });
    spisokSmezhnosti.Add(points[8], new List<Point> { points[6] });
    spisokSmezhnosti.Add(points[9], new List<Point> { points[7], points[6] });

    Dijkstra d = new Dijkstra();

    var p1 = points.First();
    var p10 = points.Last();

    var result1 = (List<Point>)d.AlgorithmDeicstra(p1, p10, spisokSmezhnosti);

    var flag = true;
    var result4 = true;
    var resultList = new List<Point>
    {
        new Point(300, 75),
        new Point(305, 104),
        new Point(311, 133),
        new Point(319, 175),
        new Point(320, 180),
        new Point(338, 227),
        new Point(352, 264),
        new Point(360, 285),
        new Point(395, 319),
        new Point(426, 349),
        new Point(451, 373),
        new Point(482, 403),
        new Point(500, 420)
    };
    Imitation.Putt.Clear();
    Imitation.helpFilters.Clear();
    Imitation n = new Imitation();
    Imitation.GraphPoints2 = points;
    Imitation.ListSmezhnosti2 = spisokSmezhnosti;
    Put.GraphPoints = points;
    Put.ListSmezhnosti = spisokSmezhnosti;


    var result2 = (List<Point>)n.MobileObject(result1);

    for (var i = 0; i < resultList.Count - 1; i++)
    {
        if ((Math.Abs(resultList[i].X - result2[i].X) > 50) || (Math.Abs(resultList[i].Y - result2[i].Y) > 50))
        {
            flag = false;
        }
    }
    Imitation.Putt.Clear();
    Imitation.helpFilters.Clear();
    Assert.AreEqual(result4, flag);
}
```

Рис. 7 – Интеграционный тест №И1 блока связей №1

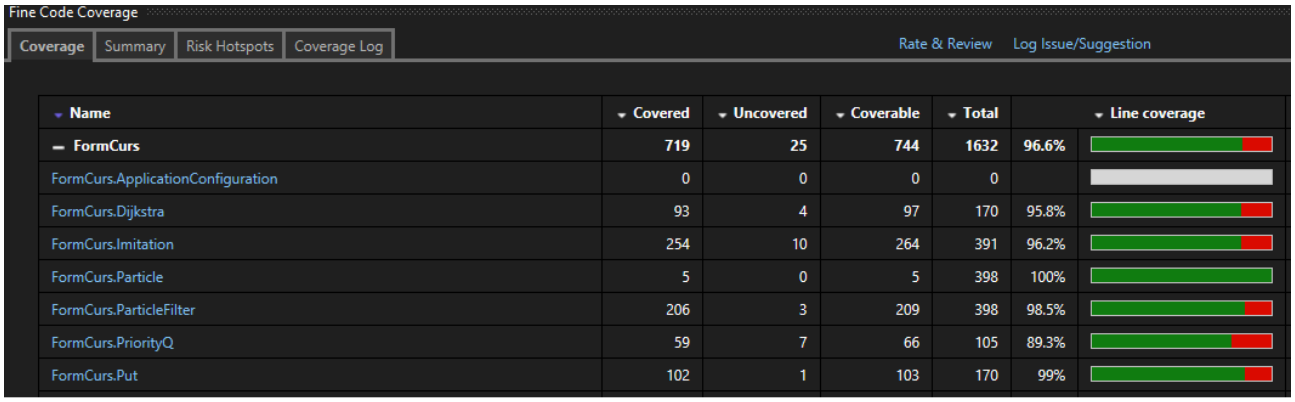
Пример интеграционного негативного теста:

```
{
    var points = new List<Point>();
    var spisokSmezhnosti = new Dictionary<Point, List<Point>>();

    points.Add(new Point { X = 300, Y = 75 });
    points.Add(new Point { X = 110, Y = 150 });
    points.Add(new Point { X = 420, Y = 120 });
    points.Add(new Point { X = 320, Y = 180 });
    points.Add(new Point { X = 140, Y = 225 });
    points.Add(new Point { X = 450, Y = 225 });
    points.Add(new Point { X = 300, Y = 300 });
    points.Add(new Point { X = 360, Y = 285 });
    points.Add(new Point { X = 160, Y = 390 });
    points.Add(new Point { X = 500, Y = 420 });

    spisokSmezhnosti.Add(points[0], new List<Point> { points[1], points[2], points[3] });
    spisokSmezhnosti.Add(points[1], new List<Point> { points[4], points[0] });
    spisokSmezhnosti.Add(points[2], new List<Point> { points[5], points[0] });
    spisokSmezhnosti.Add(points[3], new List<Point> { points[4], points[6], points[7], points[0] });
    spisokSmezhnosti.Add(points[4], new List<Point> { points[1], points[3] });
    spisokSmezhnosti.Add(points[5], new List<Point> { points[7], points[2] });
    spisokSmezhnosti.Add(points[6], new List<Point> { points[8], points[9], points[3] });
    spisokSmezhnosti.Add(points[7], new List<Point> { points[9], points[5], points[3] });
    spisokSmezhnosti.Add(points[8], new List<Point> { points[6] });
    spisokSmezhnosti.Add(points[9], new List<Point> { points[7], points[6] });

    var resultList = new List<Point>
    {
        new Point(300, 75),
        new Point(305, 104),
        new Point(311, 133),
        new Point(319, 175),
        new Point(320, 180)
    };
    Imitation.Putt.Clear();
    Imitation.helpFilters.Clear();
    Imitation n = new Imitation();
    Imitation.GraphPoints2 = points;
    Imitation.ListSmezhnosti2 = spisokSmezhnosti;
    Put.GraphPoints = points;
    Put.ListSmezhnosti = spisokSmezhnosti;

    ParticleFilter part = new ParticleFilter();
    var t = (List<Particle>)part.Initialisation(100, resultList[1], -30, 30);
    var result2 = n.ImitationPath(new Point(300, 75), new Point(300, 75), part, new Point(300, 75), resultList);

    var result3 = Imitation.Putt;

    var rresult = "Начальная и конечная точка совпадают!";
    Imitation.Putt.Clear();
    Imitation.helpFilters.Clear();
    Assert.AreEqual(rresult, result2);
}
```

Рис. 8 – Интеграционный тест №И4 блока связей №2

7 Покрытие тестами

Разработка проекта и тестирование выполнялось в среде Microsoft Visual Studio с использованием расширения Fine Code Coverage. В покрытии учитываются блочные и интеграционные тесты.



Name	Covered	Uncovered	Coverable	Total	Line coverage
FormCurs	719	25	744	1632	96.6%
FormCurs.ApplicationConfiguration	0	0	0	0	
FormCurs.Dijkstra	93	4	97	170	95.8%
FormCurs.Imitation	254	10	264	391	96.2%
FormCurs.Particle	5	0	5	398	100%
FormCurs.ParticleFilter	206	3	209	398	98.5%
FormCurs.PriorityQ	59	7	66	105	89.3%
FormCurs.Put	102	1	103	170	99%

Рис. 9 – Покрытие кода тестами:

В итоге результат составил 96,6%, 25 строк из 744 не были покрыты тестами.

8 Общее описание тестов

Всего было разработано и проведено 41 тест (из них блочных позитивных: 13, блочных негативных: 16, интеграционных позитивных: 2, интеграционных негативных: 7, аттестационных позитивных: 5, аттестационных негативных: 1). Из них в 20 была обнаружена в коде ошибка, в точности в блочных – 16 ошибок, в интеграционных – 4 ошибки.

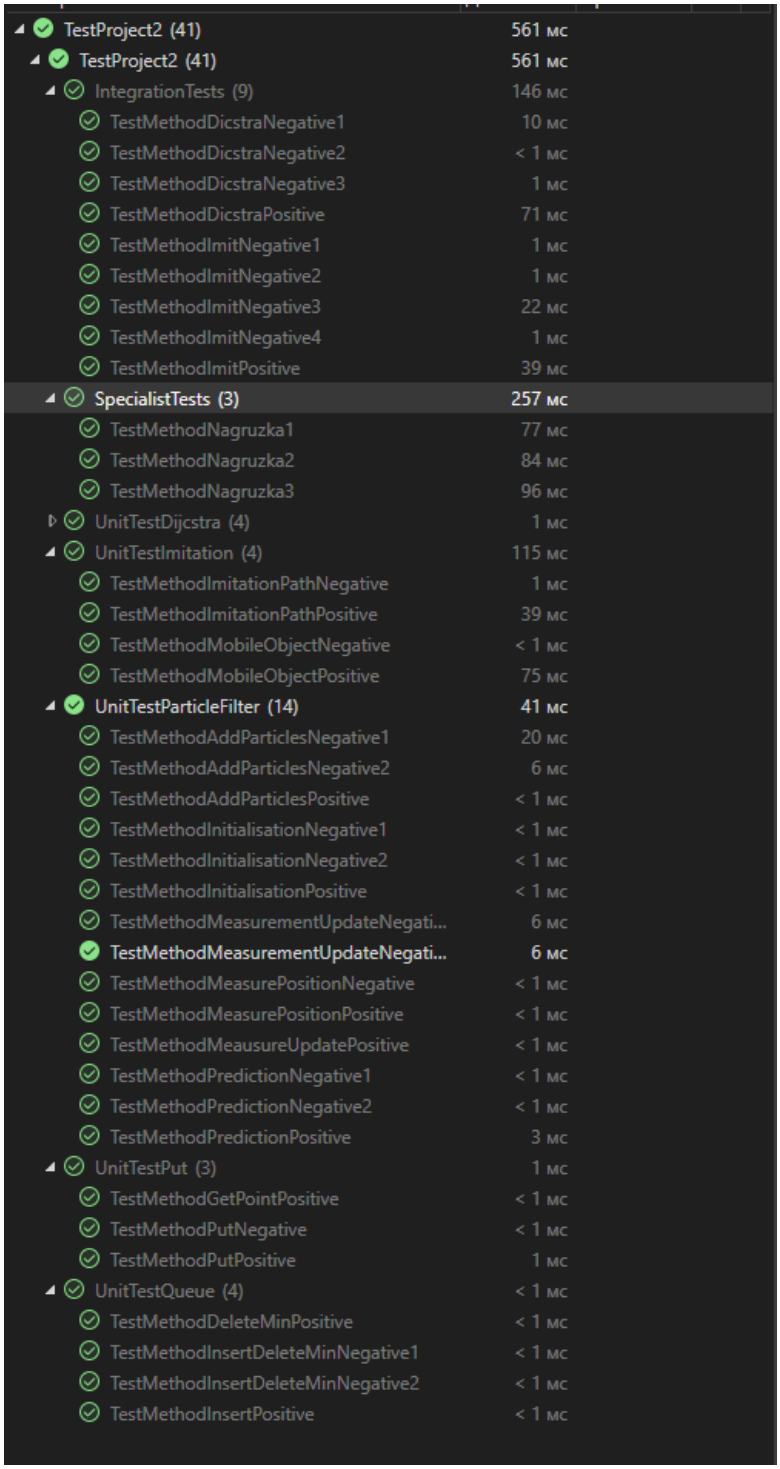


Рис. 10 – Отчет о проведении тестов

9 Итог

В ходе курса «Верификация ПО» было проведено тестирование (41 тест) программы «Сглаживание траектории мобильного объекта на графе», в результате которого были найдены и исправлены ошибки (20 ошибок), которые влияли на корректность работы разработанных алгоритмов.

10 Ссылки на фактические результаты тестов

1. Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 307, int Y = 111), Point(int X =314, int Y = 146), Point(int X = 319, int Y = 174), Point(int X = 320, int Y = 180), Point(int X = 331, int Y = 208), Point(int X = 346, int Y = 249), Point(int X = 359, int Y = 282), Point(int X = 360, int Y = 285), Point(int X = 396, int Y = 320), Point(int X = 428, int Y = 351), Point(int X = 458, int Y = 380), Point(int X = 500, int Y = 420)].

2. Массив точек List<Point>: [Point(int X = 300, int Y = 75), Point(int X = 307, int Y = 111), Point(int X =312, int Y = 139), Point(int X = 318, int Y = 171), Point(int X = 320, int Y = 180), Point(int X = 332, int Y = 211), Point(int X = 349, int Y = 256), Point(int X = 359, int Y = 282), Point(int X = 360, int Y = 285), Point(int X = 391, int Y = 314), Point(int X = 421, int Y = 344), Point(int X = 466, int Y = 380), Point(int X = 500, int Y = 420)].