|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

по дисциплине: «Компьютерная графика»

Студент Гусаров Аркадий Андреевич

Группа РК6-63Б

Тип задания Лабораторная работа №3-5

Название «Визуализация движения объектов по графу»

Вариант лабораторной работы 2

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Гусаров А.A.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Витюков Ф.A.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2022 г.*

ОГЛАВЛЕНИЕ

[Цель работы 3](#_Toc103086210)

[Задание 3](#_Toc103086211)

[Вводная часть 4](#_Toc103086212)

[Разбор кода 6](#_Toc103086213)

[Результаты работы программы 11](#_Toc103086214)

[Выводы 12](#_Toc103086215)

[Список используемых источников 12](#_Toc103086216)

Цель работы

Цель лабораторной работы – реализовать визуализацию движущегося по графу объекта на основе PhysX Tutorials, используя встроенные визуальные примитивы; для поиска кратчайшего пути из начальной точки графа в конечную, реализовать алгоритм Дейкстры.

Задание

В среде визуализации трёхмерной графики на основе PhysX Tutorials, используя встроенные визуальные примитивы, реализовать визуализацию движущегося по графу объекта.

Для этого задачу декомпозировать на следующие:

а) Генерация графа: вершины графа – точки регулярной прямоугольной сетки в плоскости XY. Для каждой точки задаются случайные смещения offsetX, offsetY, offsetZ, значения которых меньше половины шага сетки. В программе граф хранится в удобной для разработчика форме;

б) Реализация алгоритма Дейкстры;

в) Движение объекта по графу.

* Объект при старте программы появляется в одной из вершин графа.
* Выбирается конечная точка «путешествия» для объекта на графе.
* Ищется кратчайший путь к этой точке.
* Объект продолжает движение до достижения цели.
* Выбирается новая точка назначения.

Объект путешествует по графу бесконечно.

По графу может «путешествовать» несколько объектов. Объекты могут проходить сквозь друг друга, не представляя препятствий для движения.

Объект представляется шаром.

Граф рисуется в виде обычных линий белого цвета.

Найденный кратчайший путь рисуется с помощью стрелок заранее выбранного цвета, отличного от белого.

В настроечном файле должны быть доступны следующие параметры:

graphPointsCountX, graphPointsCountY – количество точек графа по осям;

objectsCount – количество движущихся объектов;

offsetX, offsetY, offsetZ - предельные значения случайных смещений точек графа относительно регулярной сетки;

objectVelocity – скорость движения объекта (в произвольных абстрактных единицах измерения).

Вводная часть

Для реализации программы использовался алгоритм Дейкстры – алгоритм на графах, который находит кратчайшие пути от одной из вершин графа до всех остальных.

Работает он следующим образом:

1. Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a.
2. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки.
3. Работа алгоритма завершается, когда все вершины посещены.

Инициализация:

1. Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности, что говорит о том, что расстояния от a до других вершин пока неизвестны.
2. Все вершины графа помечаются как непосещённые.

Шаги алгоритма:

1. Если все вершины посещены, алгоритм завершается.
2. В противном случае, из ещё не посещённых вершин выбирается вершина u, имеющая минимальную метку.
3. Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, назовём соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом.
4. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину u как посещённую и повторим шаг алгоритма.
5. Алгоритм заканчивает работу, когда все вершины посещены.
6. Если в какой-то момент все непосещённые вершины помечены бесконечностью, то это значит, что до этих вершин нельзя добраться (то есть граф несвязный). Тогда алгоритм может быть завершён досрочно.

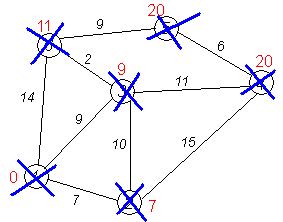


Рисунок 1. Конечно состояние алгоритма, после прохода всех вершин при перемещении из вершины 1 в любую другую

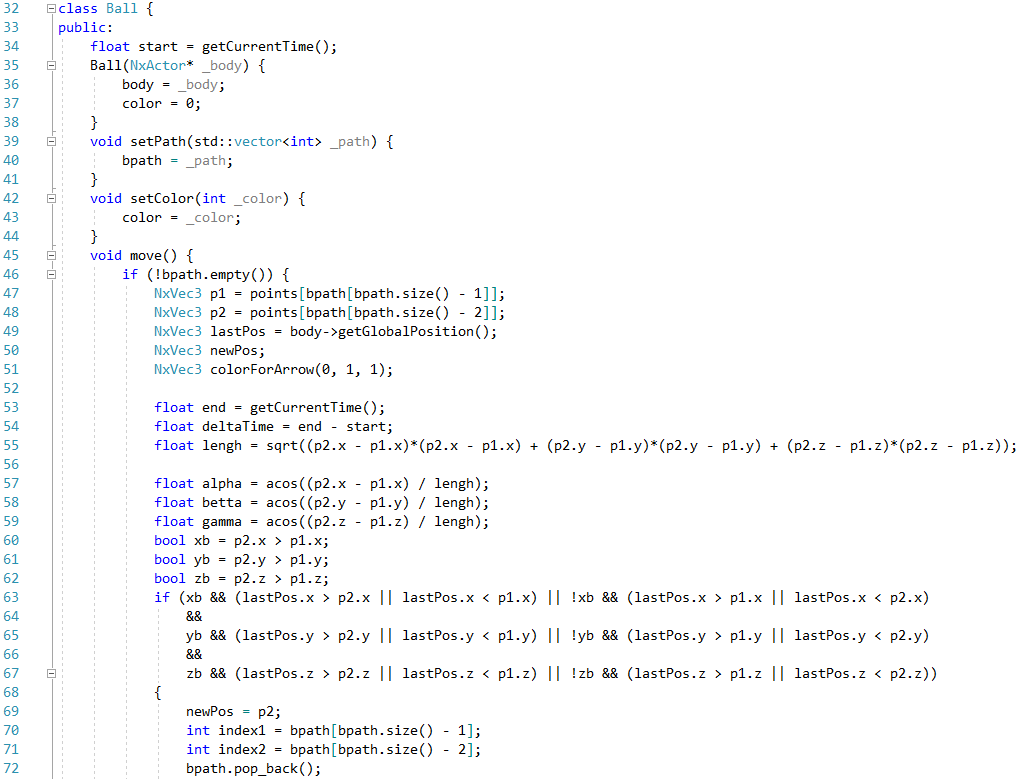
Результат работы алгоритма виден на рисунке 1: кратчайший путь от вершины 1 до 2-й составляет 7, до 3-й — 9, до 4-й — 20, до 5-й — 20, до 6-й — 11.

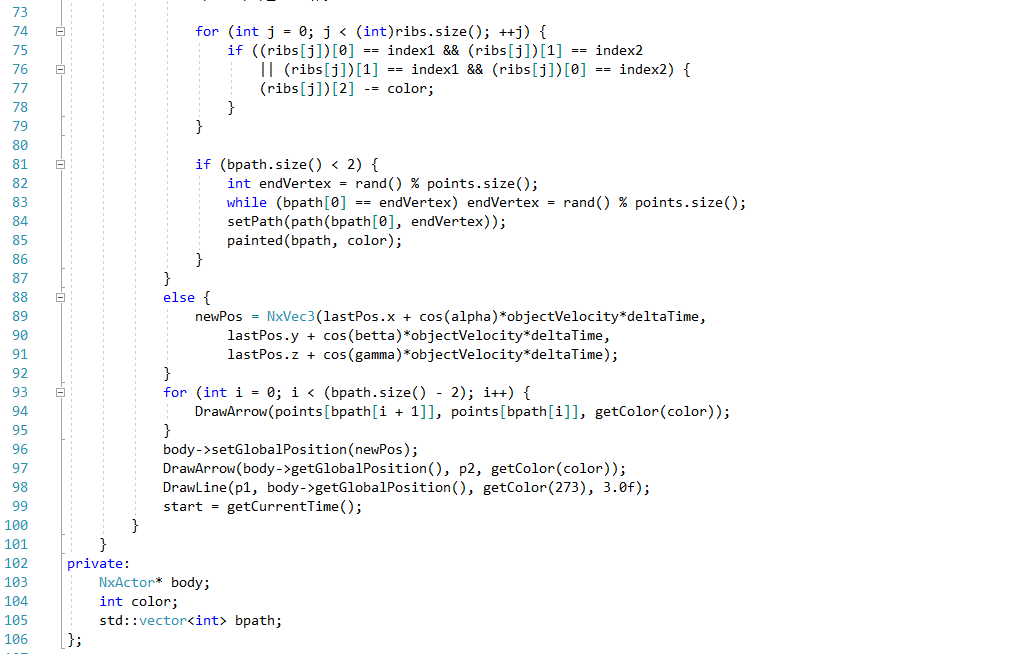
Для алгоритма Дейкстры мы используем одномерный массив (вектор). Изначально в этом массиве значения всех вершин, кроме начальной – бесконечность, значение начальной – 0.

На каждом шаге, мы выбираем не помеченную вершину с наименьшим значением и ищем из неё пути в другие вершины. Далее прибавляем вес ребра к весу первой вершины и сверяем получившееся значение со значением 2-ой вершины в массиве весов. Если сумма меньше, чем значение в массиве, то присваиваем в массив полученное значение. Помечаем вершину, как пройденную. Повторяем алгоритм, пока не дойдём до целевой вершины.

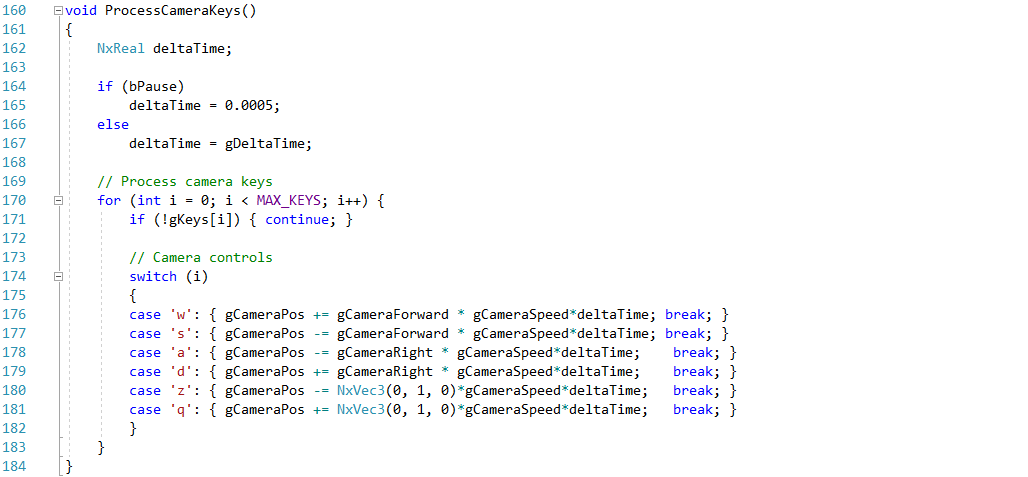
Разбор кода

Инициализация объекта Шар происходит с помощью класса *Ball*:

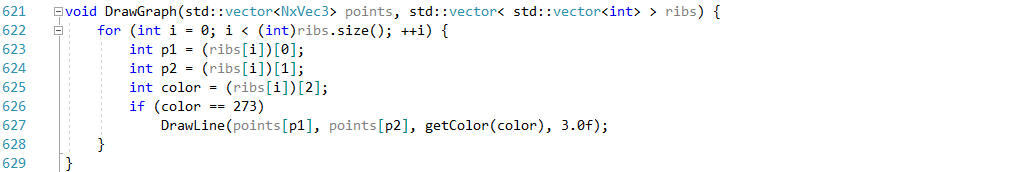




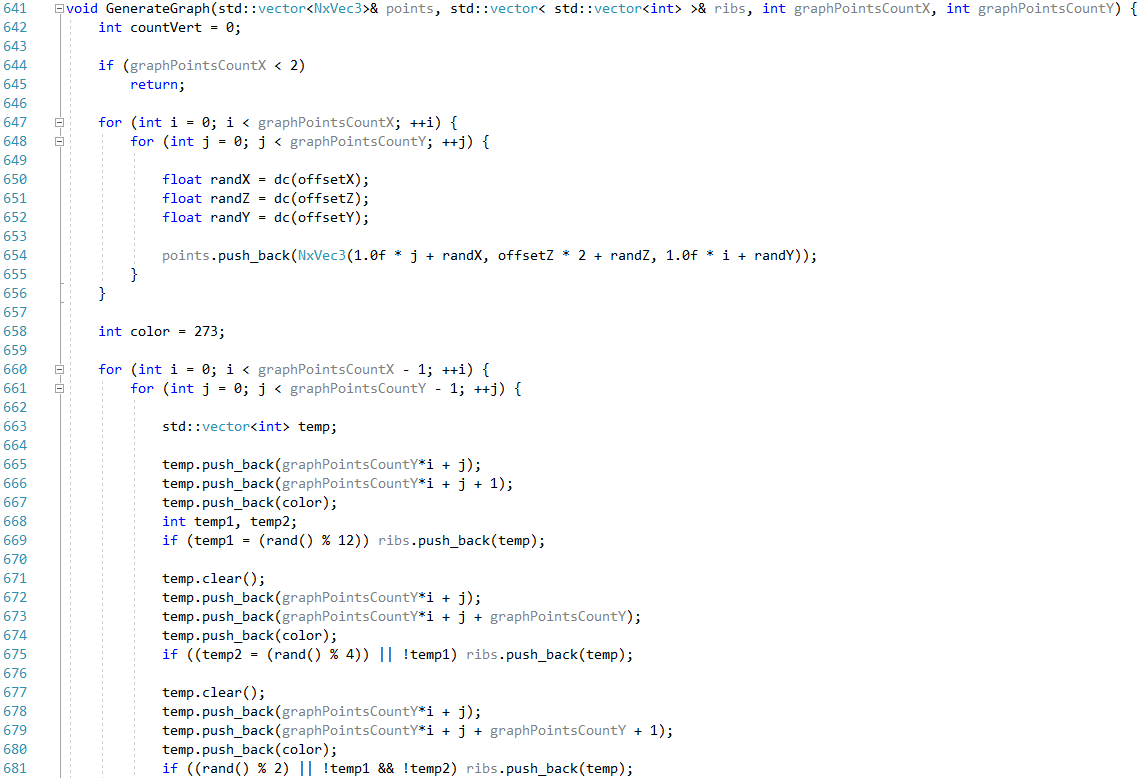
Отслеживание нажатий на клавиши для смены ракурса камеры:

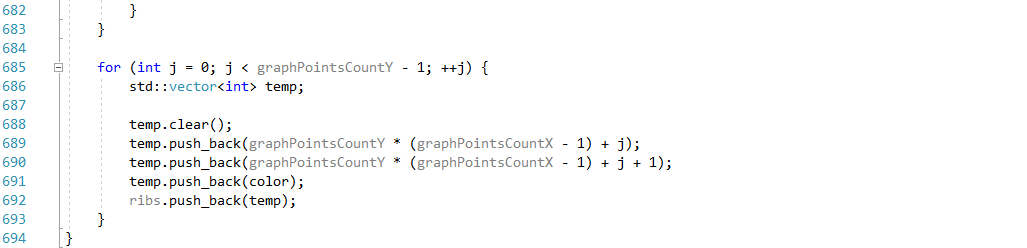


Функция отрисовки графа:

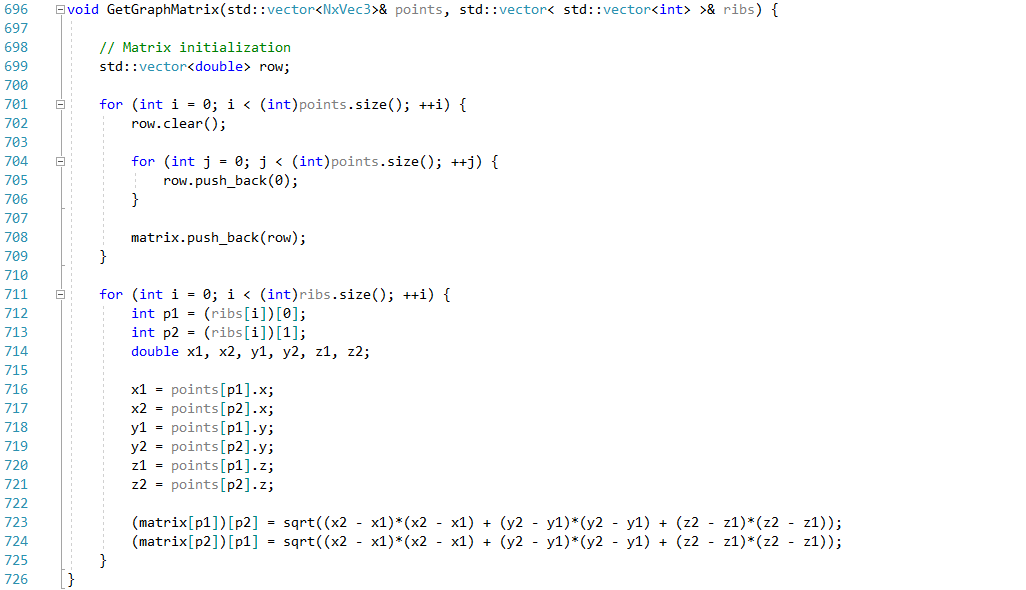


Функция генерации графа:

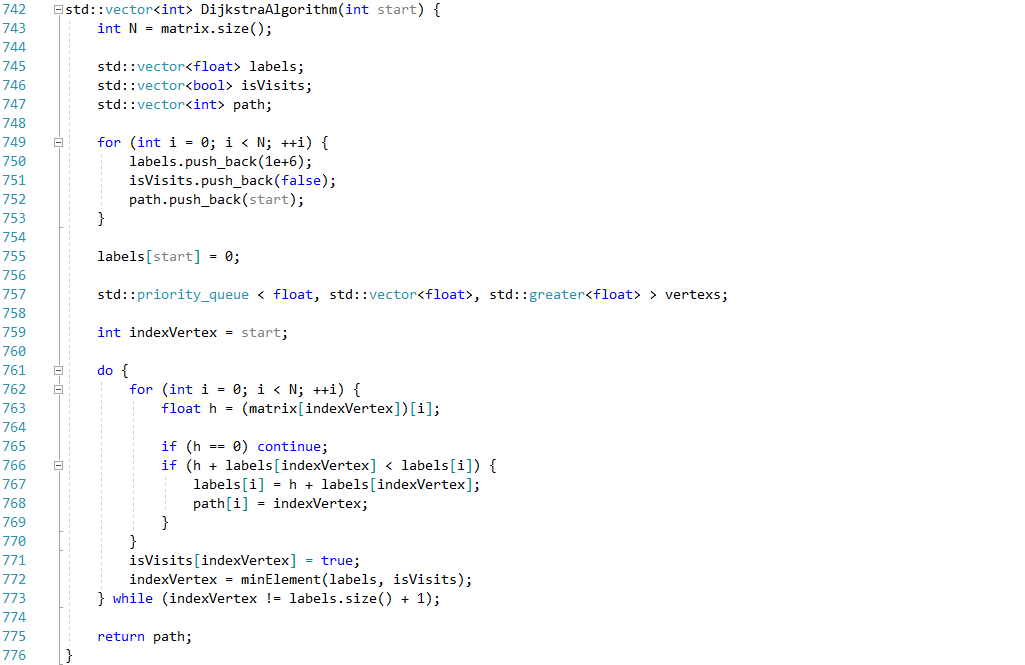




Функция инициализации матрицы, по которой происходит инициализация графа:



Алгоритм Дейкстры реализован в функции *DijkstraAlgorithm*:



Результаты работы программы

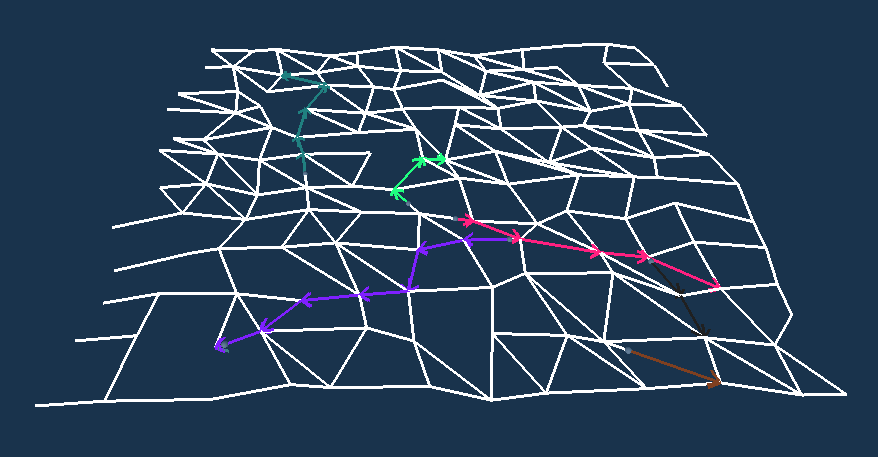


Рисунок 2. Полученная визуализация при graphPointsCountX = 12, graphPointsCountY = 12, objectsCount = 7

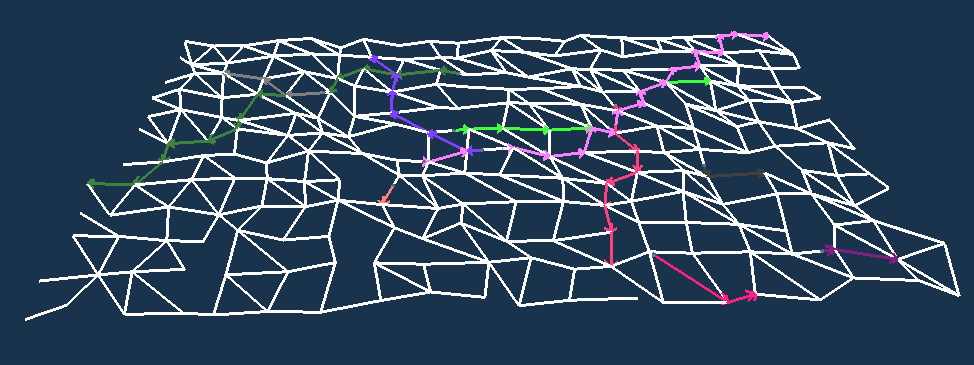


Рисунок 3. Полученная визуализация при graphPointsCountX = 12, graphPointsCountY = 20, objectsCount = 10

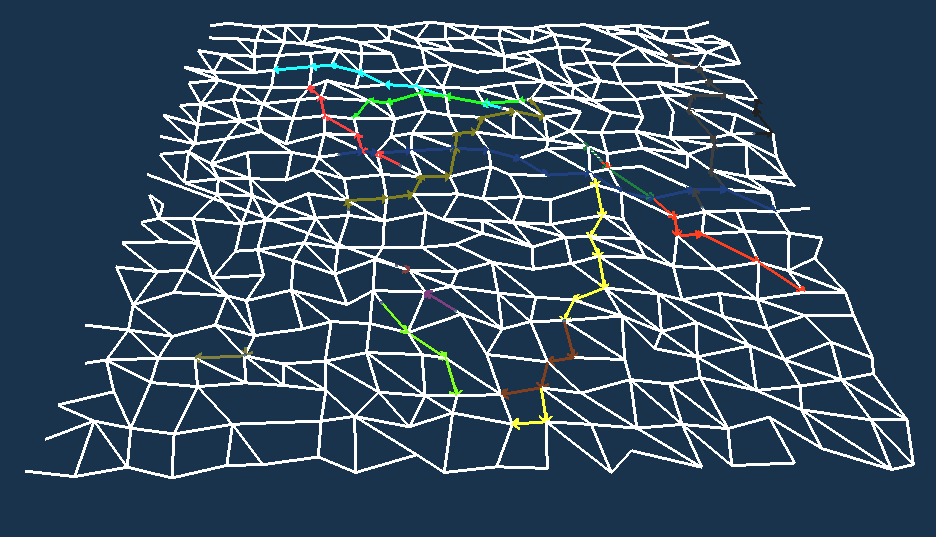


Рисунок 4. Полученная визуализация при graphPointsCountX = 20, graphPointsCountY = 20, objectsCount = 15

Выводы

В данной лабораторной работе была реализована визуализация движущегося по графу объекта на основе PhysX Tutorials с использованием встроенных визуальных примитивов, также был реализован алгоритм Дейкстры для поиска кратчайшего пути из начальной точки графа в конечную.

Список используемых источников

1. Алгоритм Дейкстры // Википедия URL: https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC\_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B (дата обращения: 06.06.2022).
2. Алгоритм Дейкстры. Поиск оптимальных маршрутов на графе // Хабр URL: https://habr.com/ru/post/111361/ (дата обращения: 06.06.2022).
3. PhysX SDK. Rigid Body chapter, lessons 101-116. Nvidia Corporation, 2008.