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 Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

по дисциплине: «Компьютерная графика»

Студент	Журавлев Н.В.
Группа	РК6-62Б
Тип задания	Лабораторная работа №3-5
Название	«Визуализация движения объектов по графу»

Студент _____ **Журавлев Н.В.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Витюков Ф.А.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2022 г.

Оглавление

Цель работы

Ознакомиться с синтаксисом базовых функций PhysX. Лабораторная работа предполагает создание и визуализацию графа, а также визуализацию движения объектов по графу.

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 выполнения поставленной задачи необходимо выделить несколько аспектов:

1. Генерация графа;
2. Реализация алгоритма Дейкстры;
3. Движение объекта по графу.

Алгоритм Дейкстры находит кратчайшее расстояние от одной из вершин графа до всех остальных. Работает только для графов без рёбер отрицательного веса. Для вычисления кратчайшего пути понадобится одномерный массив (вектор), в который будет записываться стоимость перехода из начальной вершины в другую. Стоимость перехода из начальной вершины в нее же – 0, в остальные – большое число, с которым в дальнейшем будут происходить сравнения. Данное большое число у вершин будет уменьшаться по мере прохождения алгоритма.

Алгоритм Дейкстры следующий:

1. На каждом шаге выбираются не посещённые соседние вершины для текущей вершины (в начальный момент времени – вершина 1).

2. Начинается обход выбранных вершин от меньшей стоимости перехода (веса ребра) к большей. Прибавляем вес ребра к стоимости перехода в текущую вершину и сравниваем с текущей стоимостью выбранной метки (значение в массиве). Если полученная сумма меньше, чем значение в массиве,

то меняем прошлое значение в массиве на полученное значение. Делаем обход оставшихся соседей для текущей вершины.

3. Помечаем вершину, как пройденную. Переходим к следующей вершине.

Разбор кода

Создание графа – рис.1.

```
void GenerateGraph(std::vector<NxVec3>& points, std::vector< std::vector<int> >& ribs, int graphPointsCountX, int graphPointsCountY) {
    int countVert = 0;
    if (graphPointsCountX < 2) return;
    for (int i = 0; i < graphPointsCountX; ++i) {
        for (int j = 0; j < graphPointsCountY; ++j) {
            float randX = dc(offsetX);
            float randZ = dc(offsetZ);
            float randY = dc(offsetY);
            points.push_back(NxVec3(1.0f * j + randX, offsetZ * 2 + randZ, 1.0f *
                i + randY));
        }
    }
    int color = 273;
    for (int i = 0; i < graphPointsCountX - 1; ++i) {
        for (int j = 0; j < graphPointsCountY - 1; ++j) {
            std::vector<int> temp;
            temp.push_back(graphPointsCountY * i + j);
            temp.push_back(graphPointsCountY * i + j + 1);
            temp.push_back(color);
            int temp1, temp2;
            if (temp1 = (rand() % 12)) ribs.push_back(temp);
            temp.clear();
            temp.push_back(graphPointsCountY * i + j);
            temp.push_back(graphPointsCountY * i + j + graphPointsCountY);
            cout << "connecting " << graphPointsCountY * i + j << " with " << graphPointsCountY + j + graphPointsCountY << endl;
            temp.push_back(color);
            if ((temp2 = (rand() % 4)) || !temp1) ribs.push_back(temp);
            temp.clear();
            temp.push_back(graphPointsCountY * i + j);
            temp.push_back(graphPointsCountY * i + j + graphPointsCountY + 1);
            cout << "connecting " << graphPointsCountY * i + j << " with " << graphPointsCountY + j + graphPointsCountY + 1 << endl;
            temp.push_back(color);
            if ((rand() % 2) || !temp1 && !temp2) ribs.push_back(temp);
        }
    }
    for (int j = 0; j < graphPointsCountY - 1; ++j)
    {
        std::vector<int> temp;
        temp.clear();
        temp.push_back(graphPointsCountY * (graphPointsCountX - 1) + j);
        temp.push_back(graphPointsCountY * (graphPointsCountX - 1) + j + 1);
        temp.push_back(color);
        ribs.push_back(temp);
    }
}
```

Рисунок 1. Реализация генерации графа

Реализация алгоритма Дейкстры - рис.2.

```cpp
int minElement(std::vector<float> labels, std::vector<bool> isVisits) {
    float minVal = 1e+6;
    float min = labels.size() + 1;
    for (int i = 0; i < (int)labels.size(); ++i) {
        if (!isVisits[i] && labels[i] < minVal) {
            minVal = labels[i];
            min = i;
        }
    }
    return min;
}
std::vector<int> deicstra(int start) {
    int N = matrix.size();
    std::vector<float> labels;
    std::vector<bool> isVisits;
    std::vector<int> path;
    for (int i = 0; i < N; ++i) {
        labels.push_back(1e+6);
        isVisits.push_back(false);
        path.push_back(start);
    }
    labels[start] = 0;
    std::priority_queue < float, std::vector<float>, std::greater<float> > vertexs;
    int indexVertex = start;
    do {
        for (int i = 0; i < N; ++i) {
            float h = (matrix[indexVertex])[i];
            if (h == 0) continue;
            if (h + labels[indexVertex] < labels[i]) {
                labels[i] = h + labels[indexVertex];
                path[i] = indexVertex;
            }
        }
        isVisits[indexVertex] = true;
        indexVertex = minElement(labels, isVisits);
    } while (indexVertex != labels.size() + 1);
    return path;
}
```

Рисунок 2.Функции реализации алгоритма

Конфигурационный файл – рис.3

```xml
<?xml version="1.0" encoding="utf-8"?>
<root>
<graphPointsCountX>7</graphPointsCountX>
<graphPointsCountY>7</graphPointsCountY>
<objectsCount>7</objectsCount>
<offsetX>0.3</offsetX>
<offsetY>0.3</offsetY>
<offsetZ>0.3</offsetZ>
<objectVelocity>0.75</objectVelocity>
</root>
```

Рисунок 3. Данные в конфигурационном файле

Результаты работы программы

На рис.3 приведен пример результата работы программы для следующих параметров:

graphPointsCountX – 7;

graphPointsCountY – 7;

objectsCount – 7.

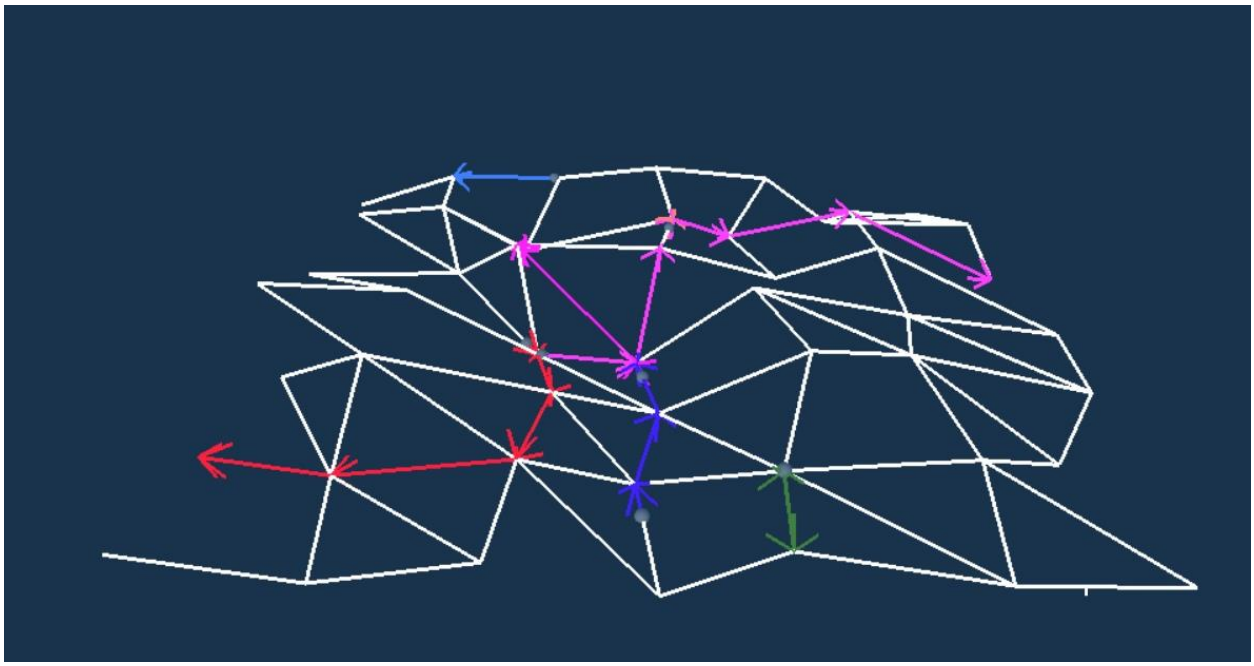

Рисунок 4.Результат исполнения программы

Выводы

При выполнении лабораторной работы были изучены основы PhysX Tutorial и усвоены основные принципы работы с объектами, графами. Освоен и реализован алгоритм Дейкстры для нахождения кратчайшего пути.