Содержание проекта:

**GridParameters.hpp** - класс содержит все параметры облака.

**Surface.hpp** - класс обеспечивает хранение поверхности облака точек и работу с ним. Хранение поверхности происходит в виде хеш-таблицы, где ключ - пара {x,y}, значение - z-координата точки поверхности. Для простоты вывода, содержимое хеш-таблицы конвертируются в массив точек.

**SphereCutter.hpp** - класс отвечающий за обработку отрезка движения сферы и работу с поверхностью.

**SurfaceWriter.hpp** - класс для вывода точек поверхности облака в файл

**Unordered\_map.hpp** и **Unordered\_set.hpp** - заголовочные файлы пользовательским определением хеш-таблиц.

Главным отличиями от предыдущего (боксового) решения являются:

1) Отказ от хранения всего облака точек в пользу лишь верхнего слоя.

Раньше хранение осуществлялось в массиве, теперь в хеш-таблице, для обеспечения быстрого поиска и доступа к элементам, по вычислительной сложности сопоставимой с массивом.

2) Новый более оптимальный алгоритм для нахождения пути. О нем далее.

3) Поддержка шага сетки отличного от 1, за счет другого подхода к решению.

Порядок действий:

Во-первых, я сразу отсеиваю случаи, когда сфера не пересекает поверхность. За это отвечает простое условие: если длина векторов **a** и **b** (Рис. 1)больше радиуса и вектора со-направлены (что означает точка начала и конца пути находятся “по одну сторону” поверхности) то поверхность **НЕ** пересекается.

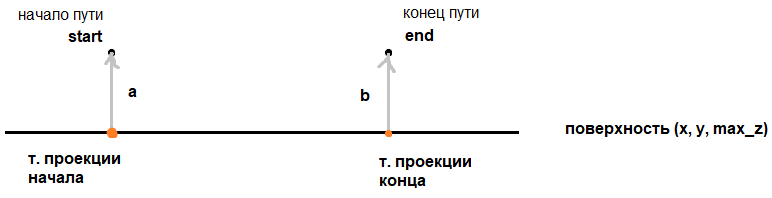


Рис. 1

В остальных случаях поверхность пересекается и переходим в след. этапу.

Нахождение **проекции пути** на поверхность облака.

Положим путь сферы имеет начало в точке s(x, y, z) и конец e (x, y, z).

Тогда проекция равна пути сферы с точками начала и конца на поверхности ps(s.x, s.y, max\_z) и pe(e.x, e.y, max\_z) – где max\_z - максимальное положительное значения оси Z (т.е поверхность).

В алгоритм передаем точку которая лежит на проекции (точка ps подходит). Т.е помещаем в стек.

Сам алгоритм:

1) Берем точку и удаляем её из стека.

2) Проверяем на принадлежность к облаку и оригинальному пути сферы (помним, что в прошлом шаге сфера гарантированно пересекает поверхность) и помещаем в выходной массив.

3) В плоскости x,y проверяем соседние для текущей точки: если сосед лежит на пути и не посещен заранее, то помещаем его в стек.

Посещенные точки добавляются в независимую хеш-таблицу.

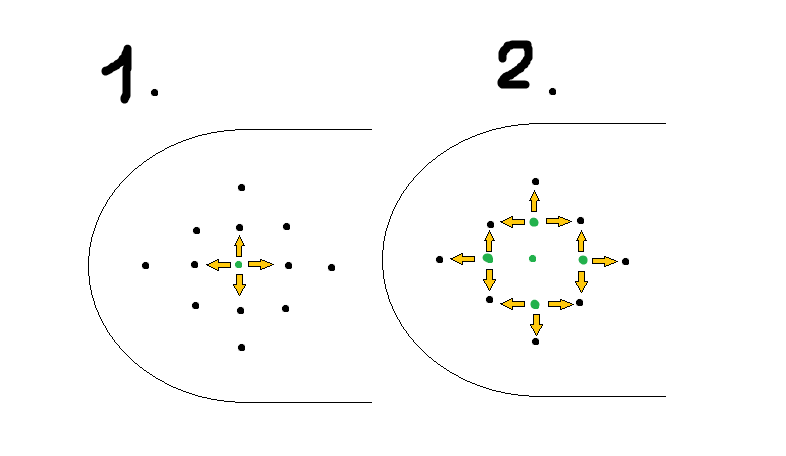


Рис. 2 Иллюстрация работы алгоритма.

И так до тех пор, пока весь путь не будет пройден. Таким я максимально избегаю точки, которые не лежат на пути и не требуют обработки.

Затем для каждой точки проекции пути я ищу нижнюю границу, где прошла сфера.

Особенность моего решения является применение хеш-таблицы, где ключ - столбец (x,y), значение - z-координата **границы** поверхности. По умолчания равна max\_z.

Таким образом для каждой точки проекции не составляет труда найти текущее значение границы на поверхности.

Если новое значение меньше текущего, то перезаписываем.

И так для всех входящих отрезков пути.

Для определения принадлежности точки к пути сферы можно выделить 2 этапа:

1. **Проверка углов между вектором образованными данной точкой и точкой начала/конца пути и вектором движения**

start, end – точки начала и конца пути соответственно.

p – произвольная точка в облаке.

v- вектор движения

Проверка осуществляется сравнением углов θ0 и θ1, представленных на Рис. 3.

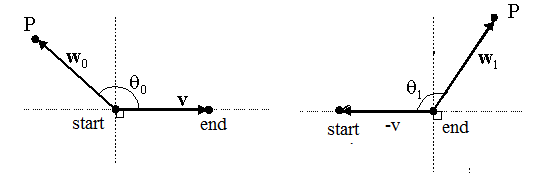


Рис 3. Углы между вектором, направленным к точке и вектором движения

Если θ0 или θ1 меньше нуля то точка лежит за пределами вектора движения и просто сравниваем длину соответствующего вектора с радиусом сферы.

Иначе:

1. **Найти расстояние от точки до вектора движения.**

Вычисляется из свойств векторного произведения и свойств параллелограмма (Рис. 4).

Площадь параллелограмма S равна длине вектора, образованному произведению векторов, на котором основан параллелограмм.

**w = s – p** (p - проверяемая точка, s - точка начала пути сферы)

**v = e - s** (s - точка начала пути сферы, e - точка конца пути сферы, v - вектор движения)

S = |w x v|

Так же площадь параллелограмма равна произведению его стороны на высоту проведенную к этой стороне.

S = |v|\*d

Из этого следует формула вычисления высоты:

d = |w x v| / |v|

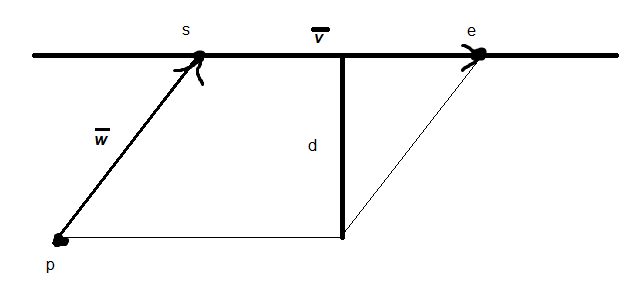


Рис. 4. Иллюстрация формулы вычисления расстояния от точки до прямой