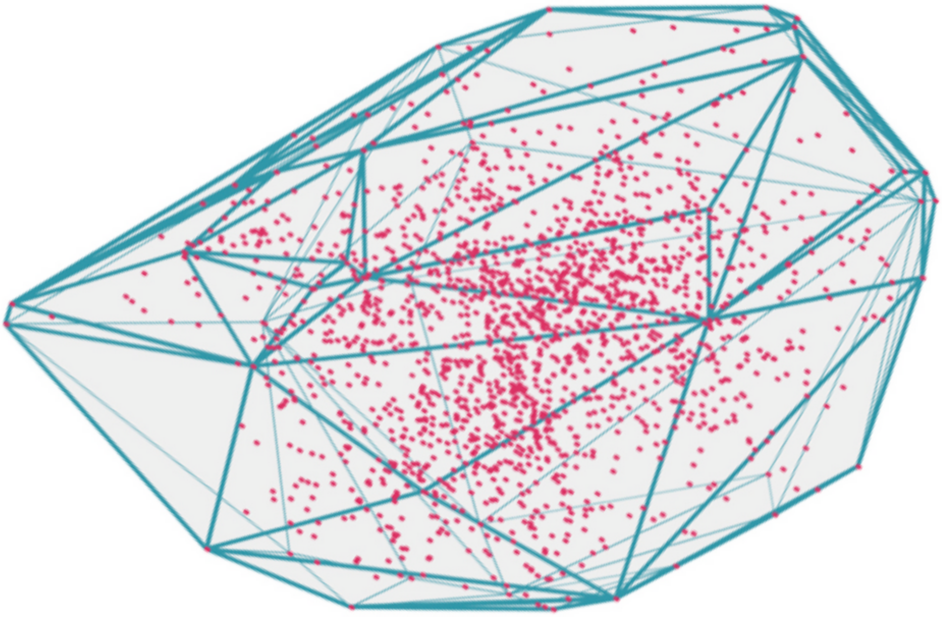
Построение выпуклой 3d-оболочки



Алгоритм, использующий метод «заворачивания подарка», строит выпуклую оболочку множества S, состоящего из n точек в трехмерном пространстве за время $O(hn)$, где h – это количество граней выпуклой оболочки.

Основная идея этого метода состоит в последовательной генерации граней выпуклой оболочки.

Алгоритм состоит из двух стадий: инициализации и серии шагов «заворачивания».

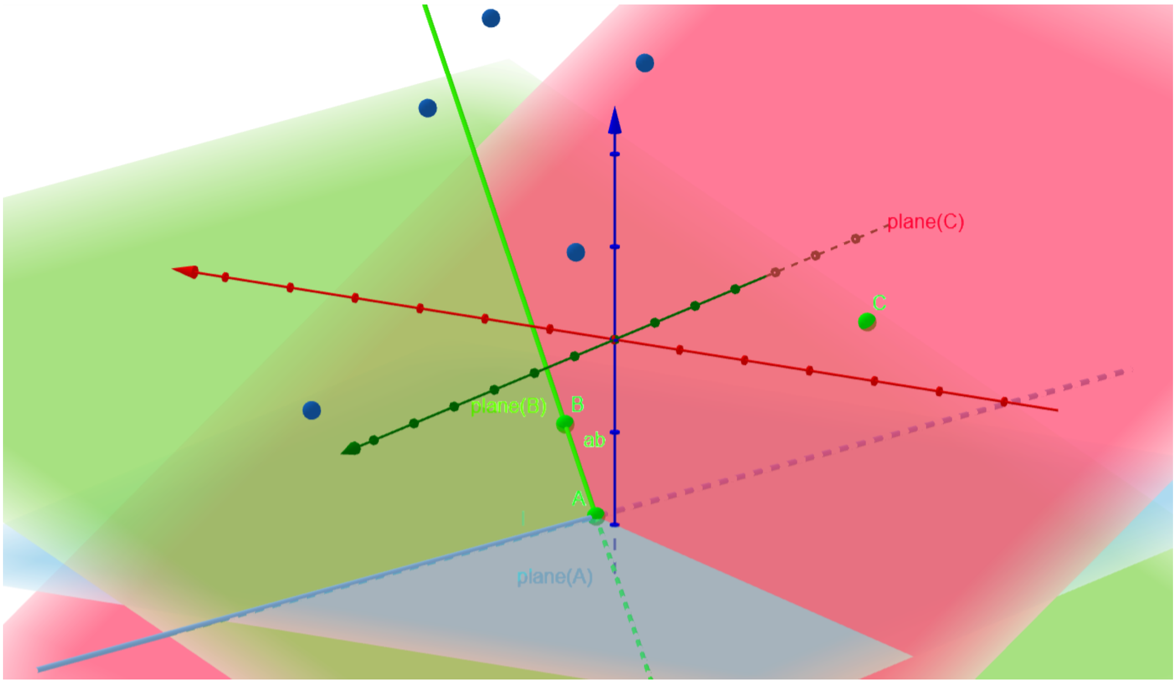
Вход

Set из точек, компаратор сначала сравнивает по z-координате, в случае равенства - по y-координате, и если и в этом случае равенство - по x-координате.

Выход

Через вызовы методов $GetFaces()$ и $GetPoints()$ можно получить вектор граней и set точек соответственно.

Инициализация



В качестве первой вершины оболочки выберем ту, что имеет наименьшую координату по оси Oz, если таких несколько, то выберем имеющую наименьшие координаты по осям Oy и Ox. Обозначим выбранную вершину A.

Рассмотрим плоскость, проходящую через A и перпендикулярную оси Oz, обозначим её $plane(A)$. Рассмотрим произвольную прямую l, лежащую в $plane(A)$ и проходящую через A, тогда некая вершина B, которая вместе с l образует плоскость $plane(B)$ так, чтобы угол между $plane(B)$ и $plane(A)$ был наименьшим, лежит в оболочке. Выберем её в качестве второй вершины оболочки.

Для завершения инициализации осталось найти третью вершину. Обозначим прямую, содержащую ранее выбранные точки A и B, как ab. Тогда некая вершина C, которая вместе с ab образует плоскость $plane(C)$ так, чтобы угол между $plane(C)$ и $plane(B)$ был наименьшим, лежит в оболочке. Выберем её в качестве третьей вершины оболочки.

Итак, мы получили первую грань оболочки - $\triangle ABC$. Определим set ребер $Edges$ и добавим в него все ребра полученного треугольника.

Шаг алгоритма

В цикле на каждом шаге «заворачивания» мы извлекаем из множества $Edges$ одно из ребер e.

Будем поддерживать для каждого ребра массив инцидентных граней, которые уже были добавлены в оболочку. Инцидентных граней у каждого ребра по 2, поэтому если у e они обе уже найдены, то удалим e из $Edges$ и вернемся в начало цикла.

К тому же, ребра в $Edges$ добавляются только после нахождения новой грани, поэтому если мы извлекли ребро из $Edges$, то хотя бы одна инцидентная ей грань уже найдена.

Итак, e инцидентно какой-то уже найденной грани T выпуклой оболочки. Мы находим треугольник U, отличный от T, такой, что он является гранью выпуклой оболочки. По сути, необходимо найти точку S, которая дополнит ребро e до грани U.

Будем хранить грани как два вектора с общим началом. (см. **Figure 1**) Тогда обозначим один из них как AB, а второй - AC. Пусть $\triangle ABC$ - это вышеописанная грань T, AB - рассматриваемое ребро e. В качестве новой вершины S выбираем ту, чтобы между треугольниками $\triangle ABC$ и $\triangle ABS$ был наибольший угол $\in [0, \ \pi]$. Для большей наглядности приоритет выбора вершин в качестве "дополнения до новой грани" к ребру e показан на **Figure 1**, чем больше номер, тем приоритетнее вершина.

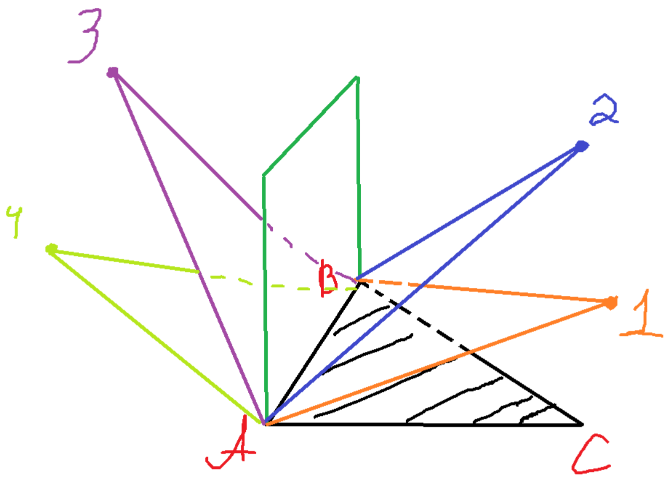


Figure 1: Выбор новой вершины оболочки

Дополнительно построим зеленую перпендикулярную плоскость (Несложно построить нормаль h к грани - возьмем векторное произведение AB и AC, h и AB образуют эту плоскость). Для чего она нужна? Дело в том, что при реализации работа сводится к подсчетам косинусов или синусов углов, и сложно отслеживать, имеем ли мы дело с углом α или $\pi - \alpha$. Но с перпендикулярной плоскостью все становится проще: если вершины, которые мы сравниваем как кандидатов быть вершиной S, лежат в разных полуплоскостях относительно зеленой плоскости, то выберем ту, что лежит полуплоскости, в которой НЕ лежит точка C. Если же вершины в одной полуплоскости, то несложно через косинусы сравнить их (опустим математические детали, это простая геометрия 10го класса).

После нахождения S добавляем два ребра полученного треугольника, отличных от e, в $Edges$, обновляем у всех трех ребер массивы инцидентных граней, и, разумеется, сохраняем новую грань U (будем поддерживать вектор из полученных граней).

Оценка сложности

Так как на каждом шаге мы находим одну точку из S, то трудоемкость шага этого алгоритма равна $O(n)$. Учитывая, что h – это количество граней выпуклой оболочки, мы сделаем не более h шагов. Таким образом, трудоемкость алгоритма «заворачивания подарка» равна $O(hn)$. Алгоритм будет полезен при малых h, так как в случае, когда все точки множества S входят в выпуклую оболочку, этот алгоритм будет иметь квадратичную трудоемкость.

Частичное тестирование

Самым важным элементом всего алгоритма является выбор новой вершины оболочки, описанный выше текстом и на рисунке **Figure 1**. Данный шаг реализован через метод
$IsFirstMoreNear(first,\ second,\ plane)$, который возвращает $true$, если точка $first$ больше подходит на роль "дополняющей до новой грани" относительно грани $plane$, нежели $second$, иначе - $false$. При тестировании данного метода достаточно создать плоскость и 4 вершины, как на **Figure 1**, и убедиться, что при всевозможных попарных сравнениях точка с большим номером является приоритетнее.

Общее тестирование

Рандомно создадим облако точек с диапазоном целочисленных координат от -10000 до 10000 в количестве 100, 1000 и 10000 штук, и создадим для них оболочку.

Для проверки корректности достаточно проверить 2 инварианта: оболочка действительно выпуклая, и все вершины из облака лежат внутри оболочки.

Проверка выпуклости реализована так: для каждой грани проверяем, лежат ли все вершины оболочки в одной полуплоскости относительно плоскости, содержащей эту грань. Если везде - да, то выпуклость есть, иначе - нет.

Проверка принадлежности вершины выпуклой оболочке: идея в том, что если некая вершина A не лежит в оболочке, то найдутся грань F и вершина B из оболочки такие, что A и B лежат в разных полуплоскостях относительно F. Если же A лежит в оболочке, то таких F и B не найдется. Поэтому пройдемся циклом по всем вершинам построенной оболочки, и на каждой итерации будем проходиться циклом по всем граням оболочки и проверять, лежат ли вершина из оболочки и рассматриваемая в одной полуплоскости относительно грани. Если везде - да, то вершина лежит в оболочке, иначе - нет.

Ссылки

Идея алгоритма была заимствована и доработана из работы по ссылке:
http://www.inf.tsu.ru/library/DiplomaWorks/CompScience/2004/Chadnov/diplom.pdf
(см. пункт 2.3.2 Алгоритм «заворачивания подарка», стр. 22-23)