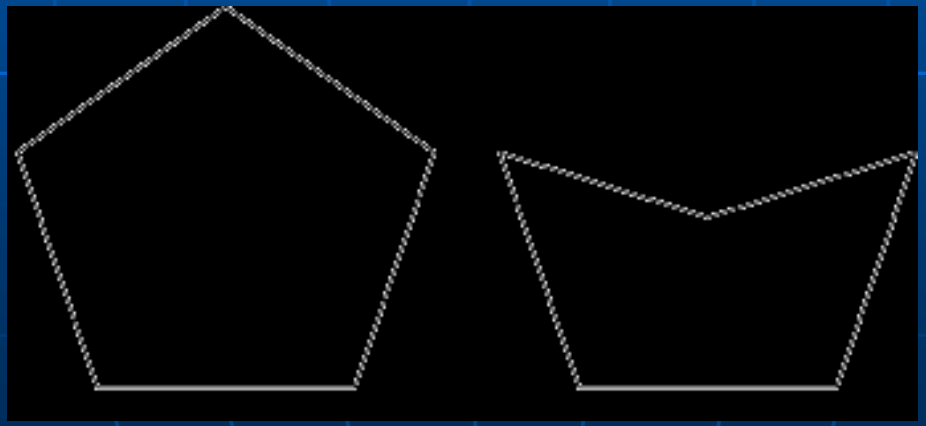
Convex Hull
(Изпъкнала обвивка)

Какво е изпъкнала обвивка?

- Ако си представим забити върху плоскост гвоздеи и около тях опънем ластик, той ще образува форма, образувана от най-външните гвоздеи и съдържаща останалите. Формата се нарича – изпъкнала обвивка.
- Изпъкналата обвивка на множество от точки X в реално векторно пространство V е най-малката обвивка съдържаща X.
- Един обект е обвивка ако за всяка двойка от точки в този обект, всяка точка от свързващата ги отсечка принадлежи на обекта.
- Ако един полигон има ъгъл 180 или повече градуса, то той не е обвивка.



Основни понятия

- Ако дадено множество съдържа една точка, то тази точка е собствената си обвивка.
- Обвивката на две точки е свързващата ги отсечка.
- Обвивката на три точки, нележащи на една права е триъгълникът образуван от тях. Ако те лежат на една права, то обвивката е отсечката свързваща 2-те крайни точки.
- Крайна точка (extreme) е такава, която лежи върху обвивката, т.е. тя е връх на полигона.
- И т.н.

- Съществуват различни алгоритми за намиране на решение за изпъкнала обвивка и те са с различна сложност.

Jarvis March

- Познат като техника за „опаковане на подаръците“.

- Започва се от точка от обвивката.
- Обратно на часовниковата стрелка, се обхождат всички точки за да се намери най-големия ъгъл от текущата крайна точка до следващата точка от обвивката.
- Точките от обвивката се намират в реда на тяхното участие.

- Пример: http://www.cs.princeton.edu/courses/archive/spr10/cos226/demo/ah/JarvisMarch.html

Реализация

- За да се намери най-големия ъгъл, трябва да се знаят последните две точки p1 и p2 от обвивката.
- Приемаме, че p1 и p2 са познати, ъгълът между тези две точки и всяка от останалите точки от областта се пресмята. Точката, за която той е максимален, се в част от обвивката.
- За да стане това имаме две отсечки p0p1 и p1p2. Търсим ъгъла при p1. За целта намираме произведението (p1 – p0) x (p2 – p0). Максималното произведение, отговаря за търсената точка.
- Операцията се повтаря докато не се стигне до първата точка и обвивката не се затвори.

Ефективност

- Proposed by R.A. Jarvis in 1973
- $O(nh)$ complexity, with n being the total number of points in the set, and h being the number of points that lie in the convex hull.
- This implies that the time complexity for this algorithm is the number of points in the set multiplied by the number of points in the hull
- The worst case for this algorithm is denoted by $O(n2)$, which is not optimal.
- Favorable conditions in which to use the Jarvis march include problems with a very low number of total points, or a low number of points on the convex hull in relation to the total number of points.

Graham's Scan

- Точките се сортират по нарастване на техните y-координати. При равенство по-малката x-координата се поставя първа.
- Останалите точки се сортират по нарастване на ъгъла спрямо x-оста, точката P и точката от оставащото множество.
- Следователно първи при сортирането ще са точките най-вдясно с най-малки ъгли, а точките най-вляво ще са последни.

- http://www.cs.princeton.edu/courses/archive/fall08/cos226/demo/ah/GrahamScan.html

Graham's Scan

- Once the points are sorted in order in respect to the anchor point from rightmost to leftmost, the anchor point and the point lying furthest to the right (the first point in the sorted data) must both be on the hull.
- These two points serve as our initial points on the hull and are pushed onto a stack representing hull points.
- From here, for each consecutive point in the sort, it is determined whether a right or left turn is taken from the top two items on the stack to the point. (cross product).
- As long as a left hand turn is made, push the point onto the stack.
- When a right hand turn is made, pop top point off the stack, reevaluate and continue popping as long as right hand turns are made. Then push the new point onto the stack.
- When the initial point is reached again, the points on the stack represent the convex hull.

Pseudocode

```
# Points[1] is the pivot or anchor point
Stack.push(Points[1]);
Stack.push(Points[2]);
FOR I = 3 TO Points.length
   o = Cross_product(Stack.second, Stack.top, Points[i];
   IF o == 0
        THEN
          Stack.pop;
          Stack.push(Points[i]);
   ELSE
        WHILE o <= 0 and Stack.length > 2
          Stack.pop;
          o = Cross_product(Stack.second, Stack.top, Points[i]);
        ENDWHILE
        Stack.push(Points[i]);
   NEXT i
```

Efficiency

- In the Graham Scan, the first phase of sorting points by angle around the anchor point is of time $O(n \log n)$ complexity.
- Phase 2 of this algorithm has a time complexity of $O(n)$.
- The total time complexity for this algorithm is $O(n \log n)$ which is much more efficient than a worse case scenario of Jarvis march at $O(n2)$.

Divide and Conquer

- In this method, two separate hulls are created, one for the leftmost half of the points, and one for the rightmost half.
- To divide in halves, sort by x-coordinates and find the median. If there is an odd number of points, the leftmost half should have the extra point.
- Recursively find the convex hull for the left set of points and the right set of points. This gives hull A and hull B.
- Stitch together the two hulls to form the hull of the entire set.
- http://www.cse.unsw.edu.au/~lambert/java/3d/divideandconquer.html

Implementation

- To stitch the hulls together, the upper and lower tangent lines must be found.
- To find the lower tangent, start with the rightmost point in hull B and the leftmost point in hull A. While the line segment formed between these two points is not the lower tangent line of the hull, figure which of the two points is not at its lower tangent point.
- While A is not the lower tangent point, move around the hull clockwise, that is A=A – 1.
- While B is not at the lower tangent point, move around the hull counterclockwise, that is B = B + 1.
- When the upper and lower tangent lines are found, march around hull A and hull B deleting the points that are now within the merged convex hull.

Efficiency

- The divide and conquer algorithm is also of time complexity $O(n \log n)$.
- This algorithm is often used for cases in three dimensions.
- A technicality of the divide and conquer method lies in how many points are in the set. If the set has less than four points, there is no need to sort the points, but rather for these special cases, determine the hull separately.
- A downside to this algorithm is the overhead associated with recursive function calls.

Quickhull

- Interactive, redraw as you go algorithm.
- Similar to quicksort in efficiency.
- Sort in terms of increasing x-coordinate values.
- Find four extreme points and form a quadrilateral.
- Discard all points that lie within the quadrilateral.
- Each edge of the quadrilateral is then examined to see whether any points lie outside the edge of the current hull.
- A triangle is drawn from the edge to the point that lies the furthest outside of the edge. Any points within this triangle are eliminated because they do not lie on the hull.
- The original edge is removed and new edges are formed with the new point and two closest extreme points.
- Edges added to a bucket from which recursion takes place to find any points that lie outside the edge and drawing new triangles until all points are eliminated other than the vertices of the hull.
- http://www.cs.princeton.edu/courses/archive/spr10/cos226/demo/ah/QuickHull.html

Efficiency

- Proposed by Preparata and Hong in 1977
- Quickhull method exhibits $O(n \log n)$ complexity.
- For unfavorable inputs, the worst case can be $O(n2)$.
- Overall, this algorithm is dependent upon how evenly the points are split at each stage. At times this algorithm can be more useful than Graham's scan, but if the points are not evenly distributed then it is often not.
- Points uniformly distributed in a square can throw out all inner points in only a few iterations.
- The quickhull algorithm saves time in that the quicksort of the points (splitting then into an upper and lower hull) splits the problem into two smaller sub-problems that are solved the same way.
- It also saves time due to the fact that a significant amount of points that lie within the quadrilateral formed can be eliminated immediately.
- The main disadvantage to this algorithm is the overhead associated with recursive function calls.