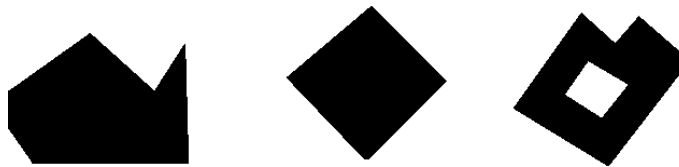


G-III Exercises**1. BSP-Trees**

- (a) Construct a BSP-Tree for the following 2D line segments: 1 : $(2, 2) \rightarrow (6, 6)$; 2 : $(3, 5) \rightarrow (8, 5)$; 3 : $(1, 1) \rightarrow (1, 5)$; 4 : $(2, 5) \rightarrow (3, 6)$; 5 : $(0, 2) \rightarrow (2, 3)$; 6 : $(6, 2) \rightarrow (8, 6)$. Note that these line segments are directed (from the start to the end position as listed above) such that one can distinguish what is on the left and the right of the line similar to polygon orientations in 3D.
- (b) Describe in principle how general 2D polygons like those shown in the figure below can be represented with a BSP tree.

**2. Line Segment - Plane Intersection**

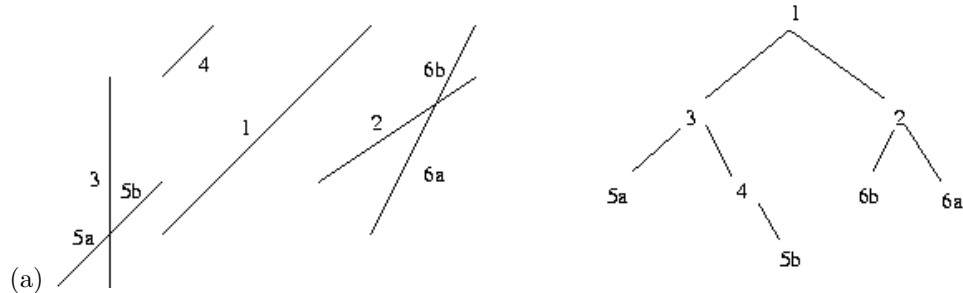
Describe a general method to compute the intersection of a straight line segment with a plane. Assume the line segment and the plane are defined by two or three 3D points respectively. Choose an appropriate representation for the line segment and the plane.

3. Convex Hull Finding

Let p_1, p_2, \dots, p_n be the vertices of a simple, non-convex, polygon 2D P in clockwise or counter-clockwise order. Explain how to modify the incremental algorithm to compute the convex hull of a point set such that it compute the convex hull of the polygon P more efficiently. What is the time order of the modified algorithm?

G-III Exercise Solutions

1. BSP-Trees



(b) The BSP tree can be used to partition the 2D plane into polygonal areas lying fully inside or fully outside of the polygon by recursively partitioning 2D areas into two sub-areas along the polygon edges:

- Each non-leaf node of the BSP tree is formed by the directed line underlying an edge of the polygon. It partitions a 2D area into a left and a right sub-area. Edges of the polygon which intersect the line of a non-leaf node are cut into two.
- A leaf node in the BSP tree represents a sub-area which indicates whether it is fully inside or fully outside of the polygon.

2. Line Segment - Plane Intersection

- Represent the straight line segment between two points p_1, p_2 in parametric form: $l(u) = p_1 + u(p_2 - p_1)$, $u \in [0, 1]$.
- Represent plane implicitly by the equation $x^t n - c = 0$, $x \in \mathbb{R}^3$, where $n \in \mathbb{R}^3$ is the unit normal of the plane and $c \in \mathbb{R}$ is its (directed) distance from the origin. If the plane is given by three non-collinear points q_1, q_2, q_3 , then $n = n_0 / \|n\|$ with $n_0 = (q_2 - q_1) \times (q_3 - q_1)$ and $c = q_1^t n$ (or similar with other point order).
- To compute the intersection solve:

$$(l(u^*))^t n - c = 0$$

$$(p_1 + u^*(p_2 - p_1))^t n - c = 0$$

$$p_1^t n + u^*(p_2 - p_1)^t n - c = 0$$

$$u^* = \frac{c - p_1^t n}{(p_2 - p_1)^t n}$$

- If $(p_2 - p_1)^t n = 0$, there is no intersection (why?)
- Otherwise the intersection between the line and the plane is $p_1 + u^*(p_2 - p_1)$
- If $u^* \in [0, 1]$, the line segment intersects the plane at this point, otherwise there is no intersection.

3. Convex Hull Finding

- Incremental convex hull algorithm (unmodified, reminder):
 - sort the points in P by x coordinate, resulting in ordered sequence p_1, \dots, p_n
 - put the points p_1, p_2 in a list L_1 , with p_1 as the first point
 - for $l = 3$ to n , do
 - * append p_l to L_1
 - * while L_1 contains more than two points and the last three points in L_1 do not make a right turn, do

- delete the middle of the last three points from L_1
 - put the points p_n and p_{n-1} in a list L_2 , with p_n as the first point
 - for $l = n - 2$ downto 1, do
 - * append p_l to L_2
 - * while L_2 contains more than 2 points and the last three points in L_2 do not make a right turn, do
 - delete the middle of the last three points from L_2
 - remove the first and the last point from L_2 and combine L_1 and L_2 to give the result
 - left / right turn speciality: three points which lie on a straight line have to be treated as if they make a left turn to avoid that the middle point of three collinear points is added to the convex hull
- Modification:
 - the ordering of the vertices can be replaced by following the vertices in sequence
 - start vertex has to be a convex vertex p_1 , e.g. the one with the smallest x coordinate
 - label the remaining $n - 1$ vertices in clockwise order, starting at p_1
 - process the vertices in turn by using the right turn check to find the convex hull
- Time order of unmodified algorithm is given by sorting, i.e. it is $O(n \log n)$. The points are already in correct order for the modification, so the loops control the time order for the modified algorithm. The time order of the loops is $O(n)$.