# Cyrus Beck Algorithm
# Line Clipping

## Background:

Cyrus Beck is a line clipping algorithm that is made for convex polygons. It allows line clipping for non-rectangular windows, unlike Cohen Sutherland . It also removes the repeated clipping needed in Cohen Sutherland.

**Input:**

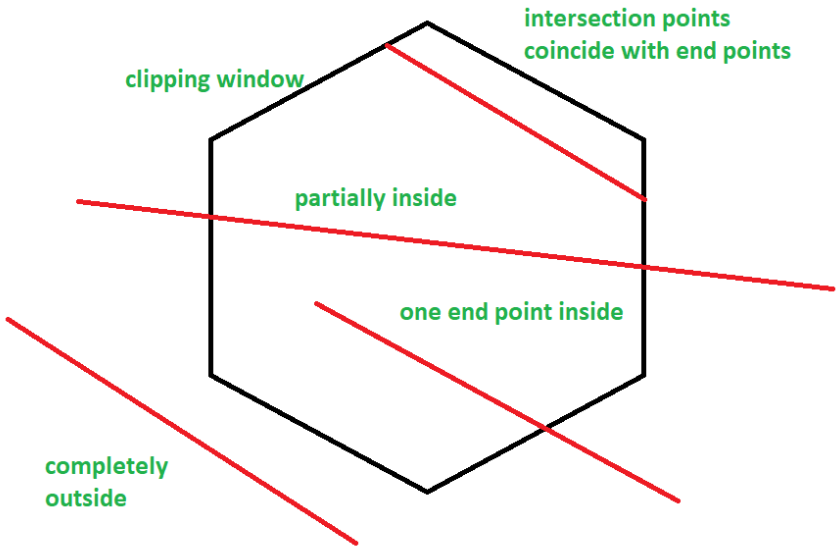
1. **Convex area of interest**
   which is defined by a set of coordinates
   given in a clockwise fashion.
2. **vertices** which are an array of coordinates:
   consisting of pairs (x, y)
3. **n** which is the number of vertices
4. A **line** to be clipped
   given by a set of coordinates.
5. **line** which is an array of coordinates:
   consisting of two pairs, (x0, y0) and (x1, y1)

**Output:**

1. Coordinates of line clipping which is the **Accepted clipping**
2. Coordinates (-1, -1) which is the **Rejected clipping**

## Algorithm:

- Normals of every edge is calculated.
- Vector for the clipping line is calculated.
- Dot product between the difference of one vertex per edge and one selected end point of the clipping line and the normal of the edge is calculated (for all edges).
- Dot product between the vector of the clipping line and the normal of edge (for all edges) is calculated.
- The former dot product is divided by the latter dot product and multiplied by -1. This is ‘t’.
- The values of ‘t’ are classified as entering or exiting (from all edges) by observing their denominators (latter dot product).
- One value of ‘t’ is chosen from each group, and put into the parametric form of a line to calculate the coordinates.
- If the entering ‘t’ value is greater than the exiting ‘t’ value, then the clipping line is rejected.




**Case 1:** The **line is partially inside** the clipping window:

```
1. 0 < tE < tL < 1
2.
3. where tE is 't' value for entering intersection point
4.       tL is 't' value for exiting intersection point
```

**Case 2:** The **line has one point inside or both sides inside** the window **or the intersection points are on the end points of the line**:

$0 \leq t_E \leq t_L \leq 1$

**Case 3:** The line is **completely outside** the window:
$t_L < t_E$

**Pseudocode:**

First, calculate the parametric form of the line to be clipped and then follow the algorithm.

- Choose a point called $P_1$ from the two points of the line ($P_0P_1$).
- Now for each edge of the polygon, calculate the normal pointing away from the centre of the polygon, namely $N_1$, $N_2$, etc.

- Now for each edge choose $P_{Ei}$ (i -> $i^{th}$ edge) (choose any of the vertices of the corresponding edge, eg.: For polygon ABCD, for side AB, $P_{Ei}$ can be either point A or point B) and calculate

$P_0 - P_{Ei}$

- Then calculate

$P_1 - P_0$

- Then calculate the following dot products for each edge:

$N_i \cdot (P_0 - P_{Ei})$

$N_i \cdot (P_1 - P_0)$

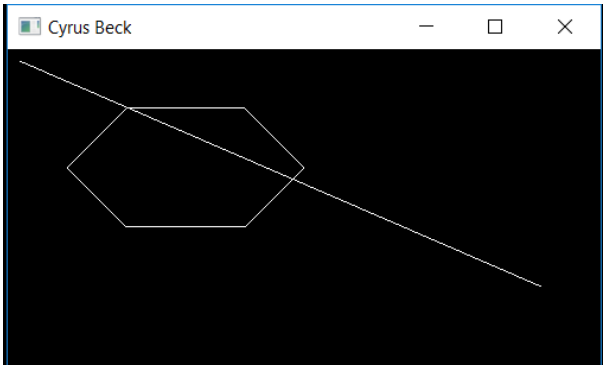
where i -> $i^{th}$ edge of the convex polygon

- Then calculate the corresponding 't' values for each edge by:

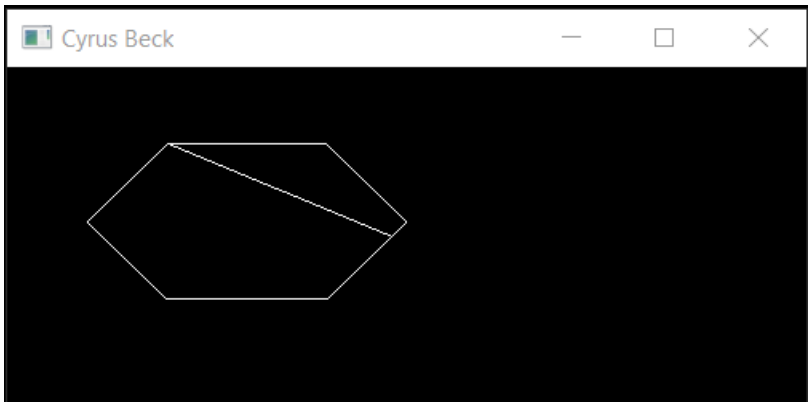
$$t = \frac{N_i \cdot (P_0 - P_{Ei})}{-(N_i \cdot (P_1 - P_0))}$$

- Then club the 't' values for which the $\mathbf{N_i \cdot (P_1 - P_0)}$ came out to be negative and take the minimum of all of them and 1.
- Similarly club all the 't' values for which the $N_i \cdot (P_1 - P_0)$ came out to be positive and take the maximum of all of the clubbed 't' values and 0.
- Now the two 't' values obtained from this algorithm are plugged into the parametric form of the 'to be clipped' line and the resulting two points obtained are the clipped points.

## BEFORE CLIPPING




## AFTER CLIPPING


Cyrus Beck