**Assignment No. 05**

5. Implement Cohen Sutherland polygon clipping method to clip the polygon with respect the

viewport and window. Use mouse click, keyboard interface

|  |
| --- |
| **Aim** |
| Implement Cohen Sutherland Hodgman algorithm to clip any given polygon. Provide the vertices of the polygon to be clipped and pattern of clipping interactively. |

|  |  |
| --- | --- |
| **Objective(s)** | |
| **1** | Implement Cohen Sutherland lgorithm to clip any given line . |
| **2** | Implement Sutherland Hodgman algorithm to clip any given polygon. |
| **3** | Provide the vertices of the polygon to be clipped and pattern of clipping interactively |

|  |
| --- |
| **Theory** |
| **Line Clipping – Cohen Sutherland**  In computer graphics, '***line clipping'*** is the process of removing lines or portions of lines outside of an area of interest. Typically, any line or part thereof which is outside of the viewing area is removed.  The Cohen–Sutherland algorithm is a computer graphics algorithm used for line clipping. The algorithm divides a two-dimensional space into 9 regions (or a three-dimensional space into 27 regions), and then efficiently determines the lines and portions of lines that are visible in the center region of interest (the viewport).  The algorithm was developed in 1967 during flight simulator work by Danny Cohen and Ivan Sutherland  The design stage includes, excludes or partially includes the line based on where:   * Both endpoints are in the viewport region (bitwise OR of endpoints == 0): trivial accept. * Both endpoints share at least one non-visible region which implies that the line does not cross the visible region. (bitwise AND of endpoints != 0): trivial reject. * Both endpoints are in different regions: In case of this nontrivial situation the algorithm finds one of the two points that is outside the viewport region (there will be at least one point outside). The intersection of the outpoint and extended viewport border is then calculated (i.e. with the parametric equation for the line) and this new point replaces the outpoint. The algorithm repeats until a trivial accept or reject occurs.   The numbers in the figure below are called outcodes. The outcode is computed for each of the two points in the line. The outcode will have four bits for two-dimensional clipping, or six bits in the three-dimensional case. The first bit is set to 1 if the point is above the viewport. The bits in the 2D outcode represent: Top, Bottom, Right, Left. For example the outcode 1010 represents a point that is top-right of the viewport. Note that the outcodes for endpoints **must** be recalculated on each iteration after the clipping occurs.   |  |  |  | | --- | --- | --- | | 1001 | 1000 | 1010 | | 0001 | 0000 | 0010 | | 0101 | 0100 | 0110 |   **Sutherland Hodgman Polygon Clipping**  It is used for clipping polygons. It works by extending each line of the convex *clip polygon* in turn and selecting only vertices from the *subject polygon* those are on the visible side.  An algorithm that clips a polygon must deal with many different cases. The case is particularly note worthy in that the concave polygon is clipped into two separate polygons. All in all, the task of clipping seems rather complex. Each edge of the polygon must be tested against each edge of the clip rectangle; new edges must be added, and existing edges must be discarded, retained, or divided. The algorithm begins with an input list of all vertices in the subject polygon. Next, one side of the clip polygon is extended infinitely in both directions, and the path of the subject polygon is traversed. Vertices from the input list are inserted into an output list if they lie on the visible side of the extended clip polygon line, and new vertices are added to the output list where the subject polygon path crosses the extended clip polygon line.  This process is repeated iteratively for each clip polygon side, using the output list from one stage as the input list for the next. Once all sides of the clip polygon have been processed, the final generated list of vertices defines a new single polygon that is entirely visible. Note that if the subject polygon was concave at vertices outside the clipping polygon, the new polygon may have coincident (i.e. overlapping) edges – this is acceptable for rendering, but not for other applications such as computing shadows.  The following example illustrate a simple case of polygon clipping    Sutherland and Hodgman's polygon-clipping algorithm uses a divide-and-conquer strategy: It solves a series of simple and identical problems that, when combined, solve the overall problem. The simple problem is to clip a polygon against a single infinite clip edge. Four clip edges, each defining one boundary of the clip rectangle, successively clip a polygon against a clip rectangle.  Note the difference between this strategy for a polygon and the Cohen-Sutherland algorithm for clipping a line: The polygon clipper clips against four edges in succession, whereas the line clipper tests the outcode to see which edge is crossed, and clips only when necessary.  **Steps of Sutherland-Hodgman's polygon-clipping algorithm**   * Polygons can be clipped against each edge of the window one at a time. Windows/edge intersections, if any, are easy to find since the X or Y coordinates are already known. * Vertices which are kept after clipping against one window edge are saved for clipping against the remaining edges. * Note that the number of vertices usually changes and will often increases. * We are using the Divide and Conquer approach * After clipped by the right and bottom clip boundaries.   The original polygon and the clip rectangle.  http://www.cc.gatech.edu/grads/h/Hao-wei.Hsieh/Haowei.Hsieh/pic7-1.gif  http://www.cc.gatech.edu/grads/h/Hao-wei.Hsieh/Haowei.Hsieh/pic7-2.gifhttp://www.cc.gatech.edu/grads/h/Hao-wei.Hsieh/Haowei.Hsieh/pic7-3.gifhttp://www.cc.gatech.edu/grads/h/Hao-wei.Hsieh/Haowei.Hsieh/pic7-4.gif  http://www.cc.gatech.edu/grads/h/Hao-wei.Hsieh/Haowei.Hsieh/pic7-5.gif  Clipping polygons would seem to be quite complex. A single polygon can actually be split into multiple polygons .The Sutherland-Hodgman algorithm clips a polygon against all edges of the clipping region in turn. The algorithm steps from vertex to vertex, adding 0, 1, or 2 vertices to the output list at each step. |

|  |
| --- |
| **Input** |
| Enter Line Coordinates , Polygon and Window |

|  |
| --- |
| **Output** |
| Clipped Line and Clipped Polygon |

|  |
| --- |
| **Lab. Based FAQ** |
| 1. What is clipping? 2. What do you mean by interior and exterior clipping? 3. Explain how exterior clipping is useful in multiple window environments 4. The Sutherland-Hodgman algorithm can be used to clip lines against a non rectangular boundary. What uses might this have? What modifications to the algorithm would be necessary? What restrictions would apply to the shape of clipping region? 5. Explain why Sutherland-Hodgman algorithm works only for convex clipping regions? |

|  |
| --- |
| **Lab practice Assignments** |
| 1. Draw a polygon for any vertex perform clipping. 2. Implement a program to implement Line Clipping Algorithm using Liang Barsky Algorithm 3. Modify the Liang-Barsky line clipping algorithm to polygon clipping. |