CHAPTER 7 – ACTIVE EDGE LIST ALGORITHM

7.1 OVERVIEW

When the Merge/Split algorithm is modified to fill regions on a given scanline at the same time, the regions are no longer independent and the algorithm becomes very similar to the standard scanline approach.

7.2 COMPARISON TO SCANLINE

In the standard scanline algorithm, edges are stored in a bucket sorted edge table. In this data structure there is, for each scanline, a list of edges that begin on the given scanline. The edge data structure contains information needed to rasterize the line and the maximum y value which serves as a stopping point. When a scanline is encountered, any edges in the list for that line are added to the active edge table (AET). In addition, edges in the AET must be checked to determine if they terminate at the given scanline. The AET is maintained in sorted order by x value and pixels are filled between pairs of edges in this table.

In the AEL algorithm presented here, the vertices are stored in a sorted vertex list as in the case of the Merge/Split algorithm. A list of edges much like the AET is maintained for filling purposes. This list is checked for modification only at event points (vertices.) An edge is added, removed, or replaced based on the classification of the vertex serving as the stopping point.
7.3 REGION FILLING

When a START vertex is encountered, the bounding edges are found as in Merge/Split and are added to the active edge list. The edges are the predecessor and successor edges of the START vertex. Horizontal edges are handled the same as in Merge/Split. A STOP vertex indicates the end of a region and the edges are removed.

A SPLIT vertex indicates the region is to split into 2 regions. Bounding edges of the new regions are the bounding edges of the original region and the predecessor and successor edges of the SPLIT vertex. Therefore, two edges are added to the list between the two existing edges for the region. If two SPLIT vertices have the same y value, the second SPLIT will cause the active edge list to again be modified before any of the region is filled.

A MERGE vertex causes the two edges incident to the MERGE vertex to be removed. They will be adjacent in the list. If there are multiple MERGE vertices at the same y value, they are acted upon independently as explained for SPLIT vertices.

When a CONTINUE vertex is encountered, an edge is replaced with the appropriate new edge.

If two vertices are endpoints of a horizontal edge, they will appear consecutively in the sorted vertex list and only the vertex with the smaller x value (the one occurring first) will modify the active edge list.

The Active Edge List Algorithm is presented below. Note that the routine add_edges must search the list for the proper location for the edges to be added. Similarly, remove_edges must find the first of the two edges and remove the adjacent pair. The routine replace_edge must also find the
edge to be replaced. Each of these routines will initialize the edge before it is inserted into the active edge list.

Procedure AEL:
(for n vertices stored in sorted vertex list v[ ])

set ael to empty
i=1
while ( i < n)
case ( v[i].VertexType)
   START:
      e1 = PredEdge( v[i] )
      x1 = v[i].x
      if ( v[i].EdgeType = SUCC_HORIZON ) i = i + 1
      e2 = SuccEdge( v[i] )
      x2 = v[i].x
      if ( x1 != x2) fill(x1,x2,v[i].y)
      add_edges(ael,e1,e2)
   MERGE:
      e1 = PredEdge( v[i] )
      x1 = v[i].x
      if ( v[i].EdgeType = SUCC_HORIZON ) i = i + 1
      e2 = SuccEdge( v[i] )
      x2 = v[i].x
      if ( x1 != x2) fill(x1,x2,v[i].y)
      remove_edges(ael,e1,e2)
   SPLIT:
      e1 = SuccEdge( v[i] )
      x1 = v[i].x
      if ( v[i].EdgeType = PRED_HORIZON ) i = i + 1
      e2 = PredEdge( v[i] )
      x2 = v[i].x
      if ( x1 != x2) fill(x1,x2,v[i].y)
      add_edges(ael,e1,e2)
   STOP:
      e1 = SuccEdge( v[i] )
      x1 = v[i].x
      if ( v[i].EdgeType = PRED_HORIZON ) i = i + 1
      e2 = PredEdge( v[i] )
      x2 = v[i].x
      if ( x1 != x2) fill(x1,x2,v[i].y)
      remove_edges(ael,e1,e2)

Active Edge List Algorithm
(continued on next page)
49

CONTINUE_SUCC:
if ( v[i].EdgeType = NO_HORIZON)
   e1 = PredEdge( v[i] )
   e2 = SuccEdge( v[i] )
else if ( v[i].EdgeType = PRED_HORIZON)
   e1 = PredEdge( v[i+1] )
   e2 = SuccEdge( v[i] )
   fill( v[i+1].x, v[i].x, v[i].y )
   i = i + 1
else
   e1 = PredEdge( v[i] )
   e2 = SuccEdge( v[i+1] )
   fill( v[i].x, v[i+1].x, v[i].y )
   i = i + 1
replace_edge( ael, e1, e2 )

CONTINUE_PRED:
if ( v[i].EdgeType = NO_HORIZON)
   e1 = SuccEdge( v[i] )
   e2 = PredEdge( v[i] )
else if ( v[i].EdgeType = SUCC_HORIZON)
   e1 = SuccEdge( v[i+1] )
   e2 = PredEdge( v[i] )
   fill( v[i+1].x, v[i].x, v[i].y )
   i = i + 1
else
   e1 = SuccEdge( v[i] )
   e2 = PredEdge( v[i+1] )
   fill( v[i].x, v[i+1].x, v[i].y )
   i = i + 1
replace_edge( ael, e1, e2 )
fill_segments( ael, v[i+1] )
i = i + 1

Active Edge List Algorithm

Procedure fill_segments( ael, v )

   while ( current y < v.y – 1)
      for each pair of edges e1 and e2 in ael
         rasterize e1 until e1.y changes
         rasterize e2 until e2.y changes
         fill( e1.x, e2.x, e1.y )

Fill_segments Algorithm