Chapter 4

Polygons

revised March 20, 2013

See the website for details on how the programming assignment will be graded. Each lab specifies a complete list of filenames to be used in the lab; use these names so that the grader can find your work.

It is important that you complete each step before going on to the next, as the exercises build upon one another. You will find it helpful to diagram the action of each method or function as you go along. If you have difficulty in some step, DO NOT proceed before resolving it. You will not be able to fully appreciate the remaining content of the lab and you are likely to compound the problem.

This lab will acquaint the student with Michener's integer-based circle drawing algorithm and direct the creation of a visual program which will implement this algorithm.

4.1 The Polygon Edges

When a vertex list for a polygon is specified, it is understood that the vertices will always be listed in counter-clockwise order around the outside of the polygon. Consider the polygon whose vertices are as follows:

$$
\begin{aligned}
v_0 &: \ (100, 600) \\
v_1 &: \ (200, 200) \\
v_2 &: \ (300, 300) \\
v_3 &: \ (500, 300) \\
v_4 &: \ (600, 400) \\
v_5 &: \ (700, 100) \\
v_6 &: \ (900, 500) \\
v_7 &: \ (400, 700)
\end{aligned}
$$

When these vertices are plotted, a convex polygon with eight edges is realized.

	v_{start}	v_{end}	Δx	Δy
e_0 :	$(100, 600)$	$(200, 200)$	$+100$	-400
e_1 :	$(200, 200)$	$(300, 300)$	$+100$	$+100$
e_2 :	$(300, 300)$	$(500, 300)$	$+200$	0
e_3 :	$(500, 300)$	$(600, 400)$	$+100$	$+100$
e_4 :	$(600, 400)$	$(700, 100)$	$+100$	-300
e_5 :	$(700, 100)$	$(900, 500)$	$+200$	$+400$
e_6 :	$(900, 500)$	$(400, 700)$	-500	$+200$
e_7 :	$(400, 700)$	$(100, 600)$	-300	-100

The polygon is to be filled one row of pixels at a time, from the bottom to the top. An *active edge list* (AEL) will be maintained during the course of the polygon fill; its contents will change from row to row, always consisting of those edges which signal either a start or a stop in the filling of the current row. An edge will begin contributing to the AEL when the row of its y_{lower} is reached and stop contributing when its y_{upper} is reached. Observe that any horizontal edge may be omitted from consideration for contribution to the AEL as the edges adjacent to it can signify the appropriate start and stop to fill it; consequently, edge e_2 is dropped from the example. (Furthermore, if edge e_2 were not dropped, how would it contribute to the AEL?)

	v_{start}	v_{end}	Δx	Δy	y_{lower}	y_{upper}
e_0 :	$(100, 600)$	$(200, 200)$	$+100$	-400	200	600
e_1 :	$(200, 200)$	$(300, 300)$	$+100$	$+100$	200	300
e_3 :	$(500, 300)$	$(600, 400)$	$+100$	$+100$	300	400
e_4 :	$(600, 400)$	$(700, 100)$	$+100$	-300	100	400
e_5 :	$(700, 100)$	$(900, 500)$	$+200$	$+400$	100	500
e_6 :	$(900, 500)$	$(400, 700)$	-500	$+200$	500	700
e_7 :	$(400, 700)$	$(100, 600)$	-300	-100	600	700

Note that the length of the AEL at any time must always be even; for each consecutive pair of edges on the AEL, the first directs filling to start and the second directs filling to stop. This raises the question of how the two edges incident at a vertex such as v_0 contribute to the AEL as compared to the contribution of the two edges incident at vertex v_7. The AEL for the scan line passing through v_0 cannot have entries for the three mathematically intersecting edges e_0, e_6, and e_7; in particular, only one of e_0 and e_7 can contribute a starting position to the AEL. This problem may be addressed by considering whether or not each polygon vertex is a *relative maximum*: vertex v_0 is not a relative maximum while vertex v_7 is. Armed with a means of differentiating between the two, one of the two edges incident at each non-relative maximum may be shortened; this will have the effect of leaving an even number of edges to contribute to the AEL for that vertex's row. In order to determine relative maxima, consider the *rising* or *falling* nature of the edge as reflected by the edge's Δy.

	v_{start}	v_{end}	Δx	Δy	y_{lower}	y_{upper}	rising/falling
e_0 :	$(100, 600)$	$(200, 200)$	$+100$	-400	200	600	$falling$
e_1 :	$(200, 200)$	$(300, 300)$	$+100$	$+100$	200	300	$rising$
e_3 :	$(500, 300)$	$(600, 400)$	$+100$	$+100$	300	400	$rising$
e_4 :	$(600, 400)$	$(700, 100)$	$+100$	-300	100	400	$falling$
e_5 :	$(700, 100)$	$(900, 500)$	$+200$	$+400$	100	500	$rising$
e_6 :	$(900, 500)$	$(400, 700)$	-500	$+200$	500	700	$rising$
e_7 :	$(400, 700)$	$(100, 600)$	-300	-100	600	700	$falling$

For reasons which will become clear, subsequent calculations are minimized if the upper end of the lower edge incident at the non-relative maximum is shortened (rather than the shortening the lower end of the upper incident edge). When moving counter-clockwise through the polygon's vertices, there are four possible combinations of rising and falling natures as each vertex is entered and exited: rising and still rising, rising then falling, falling and still falling, and falling then rising. One of these combinations

indicates a vertex which is a relative maximum: rising then falling. Another combination indicates a vertex which is a relative minimum: falling then rising. The other two, rising and still rising as well as falling and still falling, indicate vertices which are neither maxima or minima. Consequently, the upper ends of edges e_1 (rising/rising), e_5 (rising/rising) and e_0 (falling/falling) need to be shortened so that they are no longer incident on their upper endpoints.

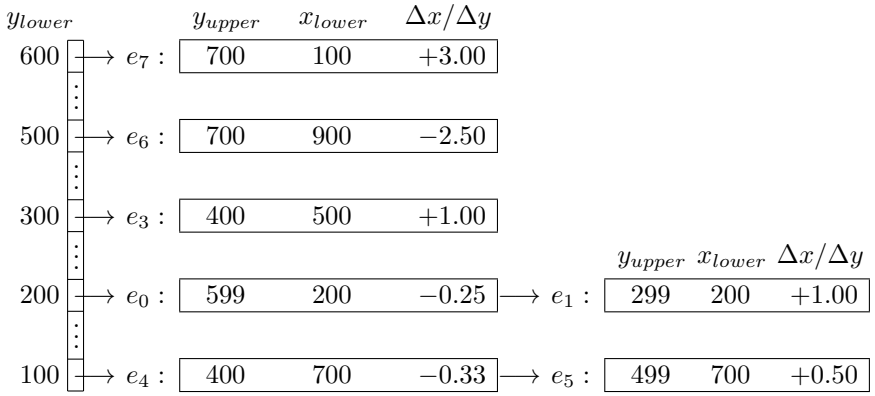
	v_{start}	v_{end}	Δx	Δy	y_{lower}	y_{upper}	rising/falling
e_0 :	$(100, 600)$	$(200, 200)$	$+100$	-400	200	599	$falling$
e_1 :	$(200, 200)$	$(300, 300)$	$+100$	$+100$	200	299	$rising$
e_3 :	$(500, 300)$	$(600, 400)$	$+100$	$+100$	300	400	$rising$
e_4 :	$(600, 400)$	$(700, 100)$	$+100$	-300	100	400	$falling$
e_5 :	$(700, 100)$	$(900, 500)$	$+200$	$+400$	100	499	$rising$
e_6 :	$(900, 500)$	$(400, 700)$	-500	$+200$	500	700	$rising$
e_7 :	$(400, 700)$	$(100, 600)$	-300	-100	600	700	$falling$

As described, each edge contributes to the AEL so long as the current row being filled lies between its y_{lower} and y_{upper}. What it contributes is its x-intercept with that row. For the row of y_{lower}, this is simply the corresponding x_{lower}. For each subsequent row, y increases by one and so the x-intercept changes by the constant amount $\Delta X/\Delta Y$ (in the same way the y changes by $\Delta Y/\Delta X = m$ when x changes by one). Note that the earlier removal of horizontal edges has eliminated the possibility of a divsion by zero here.

	v_{start}	v_{end}	Δx	Δy	y_{lower}	y_{upper}	rising/falling	x_{lower}	$\Delta x/\Delta y$
e_0 :	$(100, 600)$	$(200, 200)$	$+100$	-400	200	599	$falling$	200	-0.25
e_1 :	$(200, 200)$	$(300, 300)$	$+100$	$+100$	200	299	$rising$	200	$+1.00$
e_3 :	$(500, 300)$	$(600, 400)$	$+100$	$+100$	300	400	$rising$	500	$+1.00$
e_4 :	$(600, 400)$	$(700, 100)$	$+100$	-300	100	400	$falling$	700	-0.33
e_5 :	$(700, 100)$	$(900, 500)$	$+200$	$+400$	100	499	$rising$	700	$+0.50$
e_6 :	$(900, 500)$	$(400, 700)$	-500	$+200$	500	700	$rising$	900	-2.50
e_7 :	$(400, 700)$	$(100, 600)$	-300	-100	600	700	$falling$	100	$+3.00$

4.2 The Polygon Edge Table

The *polygon edge table* is constructed from the polygon's edges once those horizontal have been removed and those requiring shortening have had it performed.



... MORE TO FOLLOW ...

4.2.1 Building the Edge Table

```
// create an empty edge list for the polygon

for // each vertex around the polygon perimeter

   if // line segment from this vertex to next is not horizontal

      // add it to the polygon's edge list

// create empty edge table array of head pointers to edge lists

for // each edge on the polygon's edge list
{
   if // upper end is not a relative maximum

      // reduce the "y to stop at" by 1

   // add the edge to the "y to start at" list in the edge table
}
```

4.3 Filling with the Edge Table & Active Edge List

```
// create an empty AEL (active edge list)

for // each scan line in the window
{
   // add edge table list for this scan line to AEL in sorted order

   if // AEL is not empty
   {
      // fill pixel runs between pairs of x intercepts

      // delete edges if "y to stop at" has been reached

      // update x intercepts of edges remaining on AEL

   }  // end if

}  // end for
```

4.4 Assignments

Four separate submissions will be made for this laboratory.

4.4.1 Building the Edge List

Write a program to read a set of polygon vertices given in counter-clockwise order around the outside of the polygon, such as the following, from file *cg01-04.txt*.

```
7
10  10
11   6
14  10
16   8
19  11
18  12
12  12
```

Create a circular edge list from these vertices. Horizontal edges should not be added to the list. (Why?) Write the edge list in the following format, which uses the sample data above, to file *debug.txt*.

```
polygon edges prior to adjust of yUppers:
yLower  x intercept    dx/dy  yUpper  delta y
-------------------------------------------
    6      11.0000   -0.2500     10   falling
    6      11.0000    0.7500     10   rising
    8      16.0000   -1.0000     10   falling
    8      16.0000    1.0000     11   rising
   11      19.0000   -1.0000     12   rising
   10      10.0000    1.0000     12   falling
```

Determine which edges have an upper end which is not a relative extremum and truncate the upper end of those edges by one pixel. Write the edge list to the output file again; following is the example edge list after adjustments.

```
polygon edges post adjustment of yUppers:
yLower  x intercept    dx/dy  yUpper  delta y
-------------------------------------------
    6      11.0000   -0.2500      9   falling
    6      11.0000    0.7500     10   rising
    8      16.0000   -1.0000     10   falling
    8      16.0000    1.0000     10   rising
   11      19.0000   -1.0000     12   rising
   10      10.0000    1.0000     12   falling
```

Zip and submit the program as *cg01-04a.zip*.

4.4.2 Building the Edge Table

Continue the program from the previous example. Create an empty edge table as an array of empty linear edge lists, one for each horizontal line of the display. Remove one edge at a time from the polygon's circular edge list and place it on the appropriate linear edge list of the edge table. Write the edge list to the output file. Following is the edge table for the example polygon.

```
Edge Table:
yLower  x intercept    dx/dy  yUpper  x intercept    dx/dy  yUpper
------  ----------------------------  ----------------------------
```

```
 6      11.0000   -0.2500      9      11.0000    0.7500     10
 8      16.0000   -1.0000     10      16.0000    1.0000     10
10      10.0000    1.0000     12
11      19.0000   -1.0000     12
```

Zip and submit the program as *cg01-04b.zip*.

4.4.3 Managing the Active Edge List

Continue the program from the previous example. Create an empty active edge list as a linear edge list. Iterate through each horizontal row of the display. When the row corresponding to a non-empty edge table list is reached, move the edge table list entries onto the active edge list, maintaining a sorted x-intercept order. Write each non-empty active edge list to the text file. If the display were being utilized, the pixels between each pair of entries on the active edge list would be filled. Next, delete those edges for which the y_{upper} value has been reached and increment the $x_{intercept}$ to the next row for the others. Below is the series of non-empty active edge lists for the example polygon.

```
Active Edge List:
   y    x intercept    dx/dy  yUpper  x intercept    dx/dy  yUpper
  ---   ----------------------------  ----------------------------
    6      11.0000   -0.2500      9      11.0000    0.7500     10
    7      10.7500   -0.2500      9      11.7500    0.7500     10
    8      10.5000   -0.2500      9      12.5000    0.7500     10
           16.0000   -1.0000     10      16.0000    1.0000     10
    9      10.2500   -0.2500      9      13.2500    0.7500     10
           15.0000   -1.0000     10      17.0000    1.0000     10
   10      10.0000    1.0000     12      14.0000    0.7500     10
           14.0000   -1.0000     10      18.0000    1.0000     10
   11      11.0000    1.0000     12      19.0000   -1.0000     12
   12      12.0000    1.0000     12      18.0000   -1.0000     12
```

Zip and submit the program as *cg01-04c.zip*.

4.4.4 Implementing the Algorithm on the Display

Modify the program code of the previous section to write pixels on the display corresponding to the active edge list $x_{intercept}$ pairs. Zip and submit the program as *cg01-04d.zip*.