Sweep fill
Sweep fill

- Reduces to filling many spans
- Inside/outside parity is relatively straightforward
- Need to compute the spans, then fill
- Need to update the spans for each scan
- Need to implement “inside” rule for ambiguous cases.
**Spans**

- Fill the bottom horizontal span of pixels; move up and keep filling
- Assume we have $x_{\text{min}}$, $x_{\text{max}}$.
- Recall--for non integral $x_{\text{min}}$ (going from outside to inside), **round up** to get first interior point, for non integral $x_{\text{max}}$ (going from inside to outside), **round down** to get last interior point
- Recall--convention for for integral gives a span closed on the left and open on the right
- **Thus**: fill from ceiling($x_{\text{min}}$) up to but not including ceiling($x_{\text{max}}$)
Algorithm

• For each row in the polygon:
  – Throw away irrelevant edges (horizontal ones, ones that we are done with)
  – Obtain newly relevant edges (ones that are starting)
  – Fill spans
  – Update spans

• Issues:
  – what aspects of edges need to be stored?
  – when is an edge relevant/irrelevant?
The next span - 1

- for an edge, have $y=mx+c$
- hence, if $y_n=mx_n+c$, then $y_{n+1}=y_n+1=m(x_n+1/m)+c$
- hence, *if there is no change in the edges*, have:
  $$x += (1/m)$$
The next span - 2

- Horizontal edges are irrelevant (typically would be pruned at the outset)
- Edge becomes relevant when y>=y_{min} of edge
- Edge becomes irrelevant - when y>=y_{max} of edge (note appeal to convention)
Edge 2

Fill using 2 and 3

Fill using 1 and 3

Edge 1

Edge 3
Filling in details -- 1

- For each edge store: x-value, maximum y value of edge, 1/m
  - x-value starts out as x value for $y_{\text{min}}$
  - m is never 0 because we ignore horizontal ones
- Keep edges in a table, indexed by minimum y value (Edge Table==ET)
- Maintain a list of active edges (Active Edge List==AEL).
Filling in details -- 2

• For row = min to row=max
  – AEL=append(AEL, ET(row));  (add edges starting at the current row)
  – remove edges whose ymax=row
    • OK since we are assuming integral coordinates; otherwise one would use ceil(ymax)
  – sort AEL by x-value
  – fill spans
    • parity rule
    • convention for integral $x_{\text{min}}$ and $x_{\text{max}}$
  – update each edge in AEL
    • $x += (1/m)$
Compute the edge table (ET) to begin. Then fill polygon and update active edge list (AEL) row by row.

Format of edge entries

```
ET

<table>
<thead>
<tr>
<th>x</th>
<th>1/m</th>
<th>ymax</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
```

`3 0 5`  
`1 0 3`  
`5 0 5`
AEL just before filling

Row=5
Row=4
Row=3
Row=2
Row=1
Row=0
Format of edge entries

```
| x | 1/m | ymax |
```

ET

```
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
```

```
1 1 4
4 -1 4
```
AEL just before filling

Row=4

Row=3

Row=2

Row=1

Row=0
Comments

- Sort is quite fast, because AEL is usually almost in order.
- Nonetheless, OpenGL limits to convex polygons, so two and only two elements in AEL at any time, and no sorting.
- With additional logic to keep track of what color to use, can fill in many polygons at a time.
- Can be done *without* floating point
Dodging floating point

• $1/m = D_x/D_y$, which is a rational number.
• $x = x_{\text{int}} + x_{\text{num}}/D_y$
• store $x$ as $(x_{\text{int}}, x_{\text{num}})$,
• then $x \rightarrow x + 1/m$ is given by:
  – $x_{\text{num}} = x_{\text{num}} + D_x$
  – if $x_{\text{num}} \geq x_{\text{denom}}$
    • $x_{\text{int}} = x_{\text{int}} + 1$
    • $x_{\text{num}} = x_{\text{num}} - x_{\text{denom}}$

• Advantages:
  – no floating point
  – can tell if $x$ is an integer or not (check $x_{\text{num}} = 0$), and get $\text{truncate}(x)$ easily, for the span endpoints.