Fill Clipping

CS116A

Chris Pollett

Nov. 17, 2004.

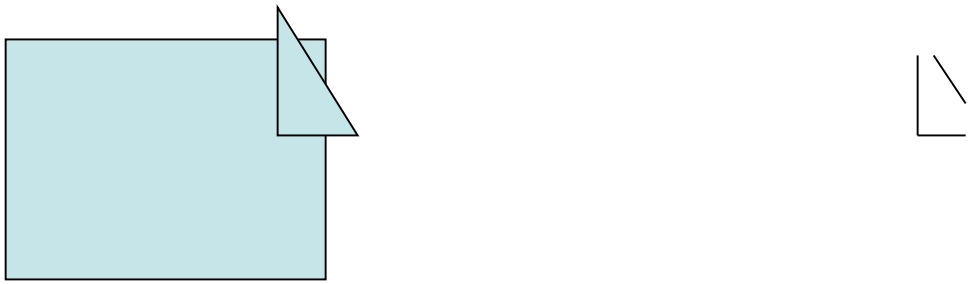
Outline

- Polygon Fill Area Clipping
- Sutherland-Hodgman Polygon Clipping
- Weiler-Atherton Polygon Clipping
- Variation on Clip Windows
- Text Clipping

Polygon Fill Area Clipping

- Since graphics packages usually only support fill areas which are polygons, this is the most important case to get a clip algorithm for.
- Consider the following situation:

after line clipping:



The result is not a polygon so hard to fill.

More Fill Area Observations

- A fill algorithm should maintain the fill area as an entity and check if it can be entirely clipped.
- If can't be clipped, then we need to locate intersection points of the polygon with the clipping rectangle and use these to make a new clipped polygon.
- Finally, we should produce a clipped polygon as a result.

Sutherland-Hodgman Polygon Clipping

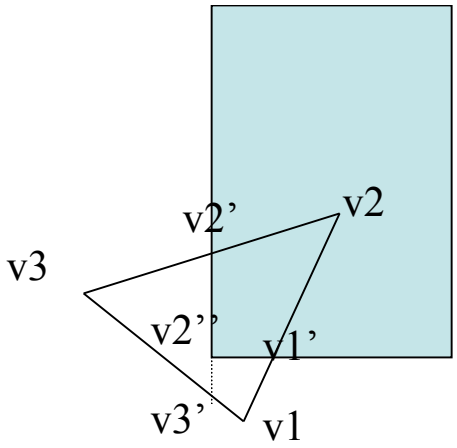
- Works with clipping convex polygons.
- Processes a polygon and produces a vertex list of clipped region to fill.
- Since only produces a single vertex list out, doesn't work for arbitrary concave regions. But it is not hard to modify it.

- The idea of the algorithm is to have four clippers -- one for each side of the clipping region.
- Edges are processed through each of these clippers in turn and output vertices are passed into the next clipper.
- Each clipper has to handle one of four cases when processing a given edge of a polygon …

More S-H Clipping

- The four cases a clipper has to handle when processing an edge are:
  - The first vertex of the edge is outside region but the second vertex is inside. In this case both the intersection of the edge with the border and the second edge are sent to the next clipper.
  - Both vertices are inside the clipping region. Then only the second vertex is sent to the next clipper.
  - The second is out but the first is in -- only the polygon edge intersection is sent to next clipper.
  - Both are out. Don't pass anything to the next clipper.
- The last clipper generates a final vertex list.

Example



{1,2}
{2,3}
{3,1}

{2}
{2'}
{3',1}
Final, list of edges:
{2, 2'}
{2', 3'}
{3', 1}
{1, 2}

{2'}
{3'}
{1}
{2}
Final, list of edges:
{2', 3'}
{3', 1}
{1, 2}
{2, 2'}

{2''}
{}
{1', 2}
{2'}
Final, list of edges:
{2'', 1'}
{1', 2}
{2, 2'}
{2', 2''}

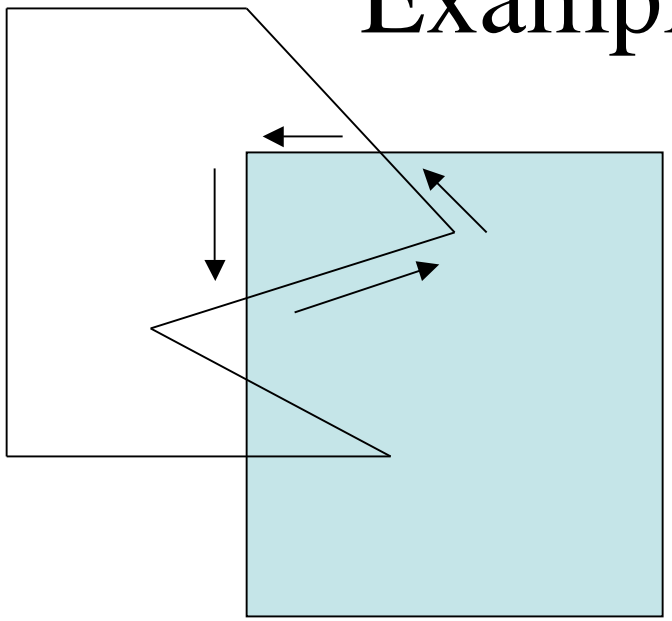
{1'}
{2}
{ 2'}
{2''}
Final, list of edges:
{1', 2}
{2, 2'}
{2', 2''}
{2'', 1'}

- We take the output of the top clipper and fill using it.

Weiler-Atherton Polygon Clipping

- This algorithm works with either convex or concave regions.
- This algorithm can also be used to identify visible surfaces in 3D.
- The idea is to go around edges of the figure in clockwise or counter-clockwise order:
  - Whenever an edge takes us outside the clip region we take a detour along the boundary of the clip region in the same ``direction''.
  - We follow the window boundary then to the next intersection point with the polygon.
  - If this is a new intersection point keep processing edges of figure till see a vertex have seen before.
  - Otherwise, form a vertex list for this area
  - Go back to exit edge and keep processing from there.

Example



After, we'd found this first region we'd continue processing from the edge after the first edge that sent us outside the figure.

Variation on Clip Windows

- Could also use Liang-Barsky to fill.
- Weiler-Atherton can also be adapted to fill even if have polygon shaped clipping windows.
- If clip window has curved boundaries then we could approximate curved boundaries with line segment, then use W-A.

Curve Clipping

What it have a curved fill area we want to clip to a rectangular clipping window?

- Could try to approximate with line segments.

Text Clipping

- Simplest strategy is all or nothing strategy for clipping strings.
- Can also do all or nothing on a character by character basis.
- Finally, can do real clipping of characters using a polygon clipping algorithm for raster fonts. Can do checking of relative positions of individual pixels in the case of bitmap fonts.