Subsurface Scattering: Translucency
- light enters and leaves at different locations on the surface
- bounces around inside
- technical Academy Award, 2003
  - Jensen, Marschner, Hanrahan

Subsurface Scattering: Milk vs. Paint

Subsurface Scattering: Marble

Subsurface Scattering: Skin

Reading for This Time
- FCG Chap 12 Graphics Pipeline
  - only 12.1-12.4

News
- Announcement from Jessica
  - www.cutsforcancer.net
- P1 grades posted (by student number)
- P3, H3 out by Wednesday

Correction: Recursive Ray Tracing
RayTrace(r,scene)
obj := FirstIntersection(r,scene)
if (no obj) return BackgroundColor;
else begin
  if ( Reflect(obj) ) then
    reflect_color := RayTrace(ReflectRay(r,obj));
  else
    reflect_color := Black;
  if ( Transparent(obj) ) then
    refract_color := RayTrace(RefractRay(r,obj));
  else
    refract_color := Black;
  return Shade(reflect_color,refract_color,obj);
end;
Clipping Lines To Viewport
- combining trivial accepts/rejects
- trivially accept lines with both endpoints inside all edges of the viewport
- trivially reject lines with both endpoints outside the same edge of the viewport
- otherwise, reduce to trivial cases by splitting into two segments

Trivial Accepts
- big optimization: trivial accept/rejects
- Q: how can we quickly determine whether a line segment is entirely inside the viewport?
- A: test both endpoints

Trivial Rejects
- Q: how can we know a line is outside viewport?
- A: if both endpoints on wrong side of same edge, can trivially reject line

Clipping Lines To Viewport
- naive approach to clipping lines:
  - for each line segment
  - for each edge of viewport
  - find intersection point
  - pick "nearest" point
  - if anything is left, draw it
- what do we mean by "nearest"?
- how can we optimize this?
Cohen-Sutherland Line Clipping
- outcodes
  - 4 flags encoding position of a point relative to top, bottom, left, and right boundary
  - \(OC(p1)=0010\)
  - \(OC(p2)=0000\)
  - \(OC(p3)=1001\)

Cohen-Sutherland Line Clipping
- assign outcode to each vertex of line to test
- line segment: \(p1-p2\)
- trivial cases
  - \(OC(p1)=0\) & \(OC(p2)=0\)
    - both points inside window, thus line segment completely visible (trivial accept)
  - \(OC(p1)=1\) & \(OC(p2)=0\)
    - there is (at least) one boundary for which both points are outside (same flag set in both outcodes)
    - thus line segment completely outside window (trivial reject)

Cohen-Sutherland Line Clipping
- if line cannot be trivially accepted or rejected, subdivide so that one or both segments can be discarded
- pick an edge that the line crosses (how?)
- intersect line with edge (how?)
- discard portion on wrong side of edge and assign outcode to new vertex
- apply trivial accept/reject tests; repeat if necessary

Viewport Intersection Code
- \((x_1, y_1), (x_2, y_2)\) intersect vertical edge at \(x_{right}\)
  - \(y_{intersect} = y_1 + \frac{(x_{right} - x_1)}{m}\)
  - \(m = \frac{(y_2 - y_1)}{(x_2 - x_1)}\)

Viewport Intersection Code
- \((x_1, y_1), (x_2, y_2)\) intersect horizon edge at \(y_{bottom}\)
  - \(x_{intersect} = x_1 + \frac{(y_{bottom} - y_1)}{m}\)
  - \(m = \frac{(y_2 - y_1)}{(x_2 - x_1)}\)
  - \((x_1, y_1), (x_2, y_2)\) intersect horizontal edge at \(y_{bottom}\)
  - \(x_{intersect} = x_1 + \frac{(y_{bottom} - y_1)}{m}\)
  - \(m = \frac{(y_2 - y_1)}{(x_2 - x_1)}\)

Polygon Clipping
- objective
  - 2D: clip polygon against rectangular window
  - or general convex polygons
  - extensions for non-convex or general polygons
  - 3D: clip polygon against parallelepiped

Polygon Clipping
- not just clipping all boundary lines
  - may have to introduce new line segments

Why Is Clipping Hard?
- a really tough case:
  - concave polygon to multiple polygons

Sutherland-Hodgeman Clipping
- basic idea:
  - consider each edge of the viewport individually
  - clip the polygon against the edge equation
  - after doing all edges, the polygon is fully clipped

Sutherland-Hodgeman Clipping
- basic idea:
  - consider each edge of the viewport individually
  - clip the polygon against the edge equation
  - after doing all edges, the polygon is fully clipped

Discussion
- key concepts
  - use outcodes to quickly eliminate/include lines
  - best algorithm when trivial accept/rejects are common
  - must compute viewport clipping of remaining lines
  - non-trivial clipping cost
  - redundant clipping of some lines
  - basic idea, more efficient algorithms exist
Sutherland-Hodgeman Clipping
• basic idea:
  • consider each edge of the viewport individually
  • clip the polygon against the edge equation
  • after doing all edges, the polygon is fully clipped

Clipping Against One Edge
• p[i] inside: 2 cases
  inside
  inside
  output: p

• p[i] outside: 2 cases
  outside
  outside
  output: p, p[i]

Clipping Against One Edge
clipPolygonToEdge( p[n], edge ) {
  for( i= 0 ; i< n ; i++ ) {
    if( p[i] inside edge ) {
      if( p[i-1] inside edge ) output p[i]; // p[i-1]= p[n-1]
        else ( p= intersect( p[i-1], p[i], edge ); output p, p[i];
      )
    else ( // p[i] is outside edge
      if( p[i-1] inside edge ) {
        p= intersect(p[i-1], p[i], edge ); output p;
      }
    )
  }
}

Sutherland-Hodgeman Example
inside
outside
p0
p1
p2
p3
p4
p5
p6
p7

Sutherland-Hodgeman Discussion
• similar to Cohen/Sutherland line clipping
• inside/outside tests: outcodes
• intersection of line segment with edge: window-edge coordinates
• clipping against individual edges independent
  • great for hardware (pipelining)
• all vertices required in memory at same time
  • not so good, but unavoidable
• another reason for using triangles only in hardware rendering

Sutherland-Hodgeman Clipping
• basic idea:
  • consider each edge of the viewport individually
  • clip the polygon against the edge equation
  • after doing all edges, the polygon is fully clipped

Sutherland-Hodgeman Algorithm
• input/output for whole algorithm
• input: list of polygon vertices in order
• output: list of clipped polygon vertices consisting of old vertices (maybe) and new vertices (maybe)
• input/output for each step
  • input: list of vertices
  • output: list of vertices, possibly with changes
• basic routine
  • go around polygon one vertex at a time
  • decide what to do based on 4 possibilities
    • is vertex inside or outside?
    • is previous vertex inside or outside?

Sutherland-Hodgeman Clipping
• basic idea:
  • consider each edge of the viewport individually
  • clip the polygon against the edge equation
  • after doing all edges, the polygon is fully clipped