Last time

- Graphics Pipeline
Today

☐ Clipping

☐ In-class Middle-Term
  ■ Thursday, Nov. 5
  ■ Close-book exam
  ■ Notes on 1 page of A4 or Letter size paper
  ■ To-know list available online
Clipping

- Parts of the geometry to be rendered may lie outside the view volume
- *Clipping* removes parts of the geometry that are outside the view
- Best done in canonical space *before perspective divide*
  - Before dividing out the homogeneous coordinate
Clipping Terminology

- Clip region: the region we wish to restrict the output to

- Geometry: the thing we are clipping
  - Only those parts of the geometry that lie inside the clip region will be output

- Clipping edge/plane: an infinite line or plane and we want to output only the geometry on one side of it
  - Frequently, one edge or face of the clip region
Clipping

- In hardware, clipping is done in canonical space before perspective divide
  - Before dividing out the homogeneous coordinate
- Clipping is useful in many other applications
  - Building BSP trees for visibility and spatial data structures
  - Hidden surface removal algorithms
  - Removing hidden lines in line drawings
  - Finding intersection/union/difference of polygonal regions
  - 2D drawing programs: cropping, arbitrary clipping
- We will make explicit assumptions about the geometry and the clip region
  - Assumption depend on the algorithm
Types of Geometry

- *Points* are clipped via inside/outside tests
  - Many algorithms for this task, depending on the clip region
- Two main algorithms for clipping polygons exist
  - Sutherland-Hodgman
  - Weiler that we will not talk about in our class
Clipping Points to View Volume

☐ A point is inside the view volume if it is on the “inside” of all the clipping planes
  ▪ The normals to the clip planes are considered to point inward, toward the visible region

☐ Now we see why clipping is done in canonical view space
  ▪ For instance, to check against the left plane:
  ▪ X coordinate in 3D must be > -1
  ▪ In homogeneous screen space, same as: \( x_{\text{screen}} > -w_{\text{screen}} \)

☐ In general, a point, \( p \), is “inside” a plane if:
  ▪ You represent the plane as \( n_x x + n_y y + n_z z + d = 0 \), with \( (n_x, n_y, n_z) \) pointing inward
  ▪ And \( n_x p_x + n_y p_y + n_z p_z + d > 0 \)
Clipping Point to Line

\[-x - y + 1 = 0\]

(0.4, 0.4)
Sutherland-Hodgman Clip

- Clip polygons to convex clip regions
- Clip the polygon against each edge of the clip region in turn
  - Clip polygon each time to line containing edge
  - Only works for convex clip regions (Why? Example that breaks?)
To clip a polygon to a line/plane:

- Consider the polygon as a list of vertices.
- One side of the line/plane is considered inside the clip region, the other side is outside.
- We are going to rewrite the polygon one vertex at a time - the rewritten polygon will be the polygon clipped to the line/plane.
- Check start vertex: if “inside”, emit it, otherwise ignore it.
- Continue processing vertices as follows...
Sutherland-Hodgman (3)

<table>
<thead>
<tr>
<th>Inside</th>
<th>Outside</th>
<th>Inside</th>
<th>Outside</th>
<th>Inside</th>
<th>Outside</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
<td><img src="image5.png" alt="Diagram" /></td>
<td><img src="image6.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- Output p
- Output i
- No output
- Output i and p
Sutherland-Hodgman (4)

- Look at the next vertex in the list, $p$, and the edge from the last vertex, $s$, to $p$. If the...
  - polygon edge crosses the clip line/plane going from out to in: emit crossing point, $i$, next vertex, $p$
  - polygon edge crosses clip line/plane going from in to out: emit crossing, $i$
  - polygon edge goes from out to out: emit nothing
  - polygon edge goes from in to in: emit next vertex, $p$
Inside-Outside Testing

- Lines/planes store a vector pointing toward the inside of the clip region - the inward pointing normal
  - Could re-define for outward pointing
- Dot products give inside/outside information
- Note that \( \mathbf{x} \) (a vector) is any point on the clip line/plane

\[
\mathbf{n} \cdot (\mathbf{s} - \mathbf{x}) < 0
\]
\[
\mathbf{n} \cdot (\mathbf{i} - \mathbf{x}) = 0
\]
\[
\mathbf{n} \cdot (\mathbf{f} - \mathbf{x}) > 0
\]
Finding Intersection Pts

- Use the parametric form for the edge between two points, \( \mathbf{x}_1 \) and \( \mathbf{x}_2 \):
  \[
  \mathbf{x}(t) = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1)t \quad 0 \leq t \leq 1
  \]

- For planes of the form \( x = a \):
  \[
  \mathbf{x}_i = (a, y_1 + \frac{(y_2 - y_1)}{(x_2 - x_1)}(a - x_1), z_1 + \frac{(z_2 - z_1)}{(x_2 - x_1)}(a - x_1))
  \]

- Similar forms for \( y = a \), \( z = a \)
- Solution for general plane can also be found
Inside/Outside in Screen Space

- In canonical view space, clip planes are $x_s = \pm 1$, $y_s = \pm 1$, $z_s = \pm 1$
- Inside/Outside reduces to comparisons before perspective divide

$$-w_s \leq x_s \leq w_s$$
$$-w_s \leq y_s \leq w_s$$
$$-w_s \leq z_s \leq w_s$$
Clipping Lines

- Lines can also be clipped by Sutherland-Hodgman
  - Slower than necessary, unless you already have hardware
- Better algorithms exist
  - Cohen-Sutherland
  - Liang-Barsky
  - Nicholl-Lee-Nicholl (we won’t cover this one - only good for 2D)
Cohen-Sutherland (1)

- Works basically the same as Sutherland-Hodgman
  - Was developed earlier
- Clip line against each edge of clip region in turn
  - If both endpoints outside, discard line and stop
  - If both endpoints in, continue to next edge (or finish)
  - If one in, one out, chop line at crossing pt and continue
- Works in both 2D and 3D for convex clipping regions
Cohen-Sutherland (2)
Cohen-Sutherland - Details

- Only need to clip line against edges where one endpoint is out
- Use *outcode* to record endpoint in/out wrt each edge. One bit per edge, 1 if out, 0 if in.
- Trivial reject:
  - outcode(x1) & outcode(x2) != 0
- Trivial accept:
  - outcode(x1) | outcode(x2) == 0
- Which edges to clip against?
  - outcode(x1) ^ outcode(x2)
Liang-Barsky Clipping

- Parametric clipping - view line in parametric form and reason about the parameter values
  - Parametric form: \( x = x_1 + (x_2-x_1)t \)
  - \( t \in [0,1] \) are points between \( x_1 \) and \( x_2 \)

- Liang-Barsky is more efficient than Cohen-Sutherland
  - Computing intersection vertices is most expensive part of clipping
  - Cohen-Sutherland may compute intersection vertices that are later clipped off, and hence don’t contribute to the final answer

- Works for convex clip regions in 2D or 3D
Parametric Clipping

- Recall, points inside a convex region are inside all clip planes.
- Parametric clipping finds the values of \( t \), the parameter, that correspond to points inside the clip region.
- Consider a rectangular clip region.

\[ \text{Left, } x = x_{\text{min}} \]
\[ \text{Top, } y = y_{\text{max}} \]
\[ \text{Right, } x = x_{\text{max}} \]
\[ \text{Bottom, } y = y_{\text{min}} \]
Parametric Intersection

- Consider line to be infinite
- Find parametric intersections $t_{bottom}, t_{left}, t_{top}, t_{right}$
Entering and Leaving

- Recall, a point is inside a view volume if it is on the inside of every clip edge/plane.
- Consider the left clip edge and the infinite line. Two cases:
  - $t < t_{left}$ is inside, $t > t_{left}$ is outside → leaving
  - $t < t_{left}$ is outside, $t > t_{left}$ is inside → entering
- To be inside a clip plane we either:
  - Started inside, and have not left yet
  - Started outside, and have entered
Entering/Leaving Example

To be inside the clip region, you must have entered every clip edge before you have left any clip edge.
When are we Inside?

- We want parameter values that are inside all the clip planes.
- Any clip plane that we started inside we must not have left yet.
  - First parameter value to leave is the end of the visible segment.
- Any clip plane that we started outside we must have already entered.
  - Last parameter value to enter is the start of the visible segment.
- If we leave some clip plane before we enter another, we cannot see any part of the line.
- All this leads to an algorithm - Liang-Barsky.
Liang-Barsky Sub-Tasks

1. Find parametric intersection points
   ■ Parameter values where line crosses each clip edge/plane

2. Find entering/leaving flags
   ■ For every clip edge/plane, are either entering or leaving

3. Find last parameter to enter, and first one to leave
   ■ Check that enter before leave

4. Convert these into endpoints of clipped segment
1. Parametric Intersection

- Segment goes from \((x_1, y_1)\) to \((x_2, y_2)\):
  \[
  \Delta x = x_2 - x_1 \\
  \Delta y = y_2 - y_1
  \]

- Rectangular clip region with \(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}\)

- Infinite line intersects rectangular clip region edges when:

\[
\begin{align*}
  k & = \frac{p_{\text{top}} - p_{\text{bottom}}}{p_{\text{right}} - p_{\text{left}}} \\
  t_k & = \frac{q_k}{p_k}
\end{align*}
\]

  where

\[
\begin{align*}
  p_{\text{left}} & = -\Delta x \\
  q_{\text{left}} & = x_1 - x_{\text{min}} \\
  p_{\text{right}} & = \Delta x \\
  q_{\text{right}} & = x_{\text{max}} - x_1 \\
  p_{\text{bottom}} & = -\Delta y \\
  q_{\text{bottom}} & = y_1 - y_{\text{min}} \\
  p_{\text{top}} & = \Delta y \\
  q_{\text{top}} & = y_{\text{max}} - y_1
\end{align*}
\]
2. Entering or Leaving?

- When $p_k < 0$, as $t$ increases line goes from outside to inside - entering
- When $p_k > 0$, line goes from inside to outside - leaving
- When $p_k = 0$, line is parallel to an edge
  - Special case: one endpoint outside, no part of segment visible, otherwise, ignore this clip edge and continue

\[ p_{left} = -\Delta x \]
\[ p_{right} = \Delta x \]
\[ p_{bottom} = -\Delta y \]
\[ p_{top} = \Delta y \]
Find Visible Segment $ts$

- Last parameter is enter is $t_{small} = \max(0, \text{entering } t's)$
- First parameter is leave is $t_{large} = \min(1, \text{leaving } t's)$
- If $t_{small} > t_{large}$, there is no visible segment
- If $t_{small} < t_{large}$, there is a line segment
  - Compute endpoints by substituting $t$ values into parametric equation for the line segment
- Improvement (and actual Liang-Barsky):
  - compute $t's$ for each edge in turn (some rejects occur earlier like this)
General Liang-Barsky

- Liang-Barsky works for any convex clip region
  - E.g. Perspective view volume in world or view coordinates
- Require a way to perform steps 1 and 2
  1. Compute intersection $t$ for all clip lines/planes
  2. Label them as entering or exiting
In View Space

\[ \mathbf{x}_{\text{left}} \quad \text{frustum} \quad \mathbf{x}_{\text{right}} \]

\[ \mathbf{x}_1 \quad \text{eye, } \mathbf{e} \quad \mathbf{x}_2 \]
First Step

- Compute inside/outside for endpoints of the line segment
  - Determine which side of each clip plane the segment endpoints lie
  - Use the cross product
  - What do we know if \((\mathbf{x}_1 - \mathbf{e}) \times (\mathbf{x}_{\text{left}} - \mathbf{e}) > 0\) ?
  - Other cross products give other information

- What can we say if both segment endpoints are outside one clip plane?
  - Stop here if we can, otherwise...
Finding Parametric Intersection

- Left clip edge: \( x = e + (x_{left} - e) \ t \)
- Line: \( x = x_1 + (x_2 - x_1) \ s \)
- Solve simultaneous equations in \( t \) and \( s \):
  \[ e_x + (x_{left,x} - e_x) t = x_{1,x} + (x_{2,x} - x_{1,x}) s \]
  \[ e_y + (x_{left,y} - e_y) t = x_{1,y} + (x_{2,y} - x_{1,y}) s \]

- Use endpoint inside/outside information to label as entering or leaving
- Now we have general Liang-Barsky case
General Clipping

- Liang-Barsky can be generalized to clip line segments to arbitrary polygonal clip regions
  - Consider clip edges as non-infinite segments
  - Look at all intersecting $t\$s between 0 and 1

- Clipping general polygons against general clip regions is quite hard: Weiler-Atherton algorithm
Next Time

☐ Rasterization

☐