`tmesh_t` File Format

$\langle red \rangle$ $\langle green \rangle$ $\langle blue \rangle$

$\langle vertex\ count \rangle$

$\langle x_1 \rangle$ $\langle y_1 \rangle$ $\langle z_1 \rangle$ $\langle x_2 \rangle$ $\langle y_2 \rangle$ $\langle z_2 \rangle$ $\ldots$

$\langle face\ count \rangle$

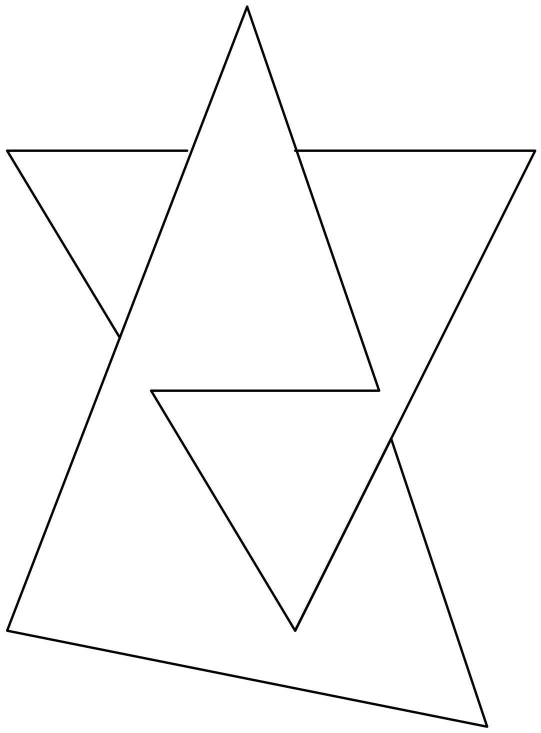
$\langle v_{1,1} \rangle$ $\langle v_{1,2} \rangle$ $\langle v_{1,3} \rangle$

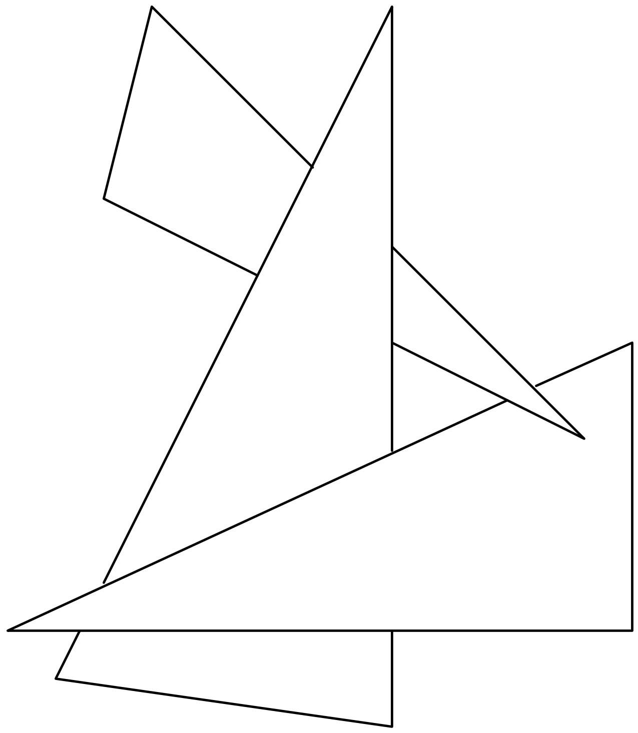
$\langle v_{2,1} \rangle$ $\langle v_{2,2} \rangle$ $\langle v_{2,3} \rangle$

$\langle v_{3,1} \rangle$ $\langle v_{3,2} \rangle$ $\langle v_{3,3} \rangle$ $\ldots$

Three Dimensional Rendering

- For a single convex object, backface removal is sufficient.
- For objects that can be placed in a *occlusion-compatible order*, the painter's algorithm combined with depth sorting is sufficient.
- For complex, interpenetrating objects, we must use Warnock's algorithm or Z buffering.





Painter's Algorithm

Sort all of the potentially visible polygons by decreasing z value.

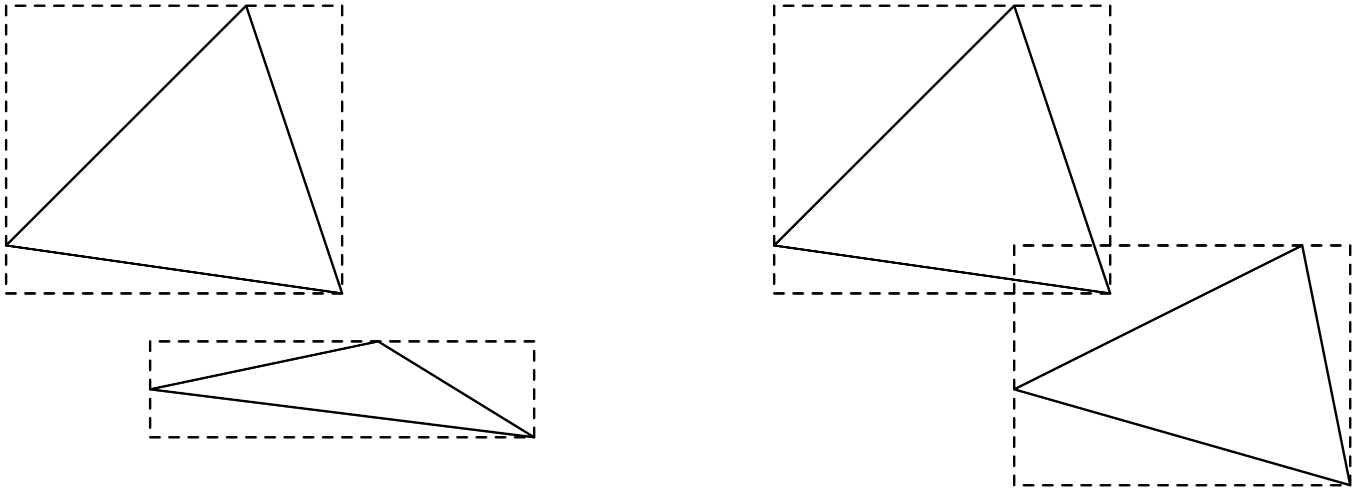
Draw them in this order.

Polygons in front of other polygons will be drawn later, so they will be visible, and they will occlude the polygons behind them.

Depth Sorting

1. **Perspective Depth Transformation**
2. **Overlap Tests**
   (a) Bounding rectangles
   (b) Edge intersection tests
       Bounding rectangles
       Parallel edges
       Intersecting edges
   (c) Complete containment
       Bounding rectangles
       Triangle-inside
3. **Depth Comparison**
   (a) Bounding z intervals
   (b) z comparison at a common point.
       Break ties using additional points.

Overlap Testing With Bounding Rectangles



Bounding rectangle:

$$\begin{aligned} left &= \min\{x_1, x_2, x_3\} \\ right &= \max\{x_1, x_2, x_3\} \\ bottom &= \min\{y_1, y_2, y_3\} \\ top &= \max\{y_1, y_2, y_3\} \end{aligned}$$

Overlap Testing by Edge Intersection

If two triangles overlap then either one is contained in the other or at least two edges of each triangle intersect.

$$\text{Let} \qquad edge_1 = (x_1, y_1) \longrightarrow (x_2, y_2) \qquad edge_2 = (x_3, y_3) \longrightarrow (x_4, y_4)$$

For each pair of edges, test whether they intersect.

- **Bounding rectangles.** If the bounding rectangles of the two edges are disjoint, then the edges do not intersect.

- **Parallel edges.** Compute
  $D = (x_3 - x_4)(y_1 - y_2) - (x_1 - x_2)(y_3 - y_4)$
  If $|D| < \epsilon$ the lines are parallel, so they do not intersect.

- **Intersecting Edges.** If $|D| > \epsilon$, compute
  $s = \left[(x_3 - x_4)(y_1 - y_3) - (x_1 - x_3)(y_3 - y_4)\right]/D$
  $t = \left[(x_1 - x_2)(y_1 - y_3) - (x_1 - x_3)(y_1 - y_3)\right]/D$
  If $0 < (s, t) < 1$ then the intersection point is between the endpoints.

Containment

- **Bounding rectangle test:** Test whether the center of one triangle is contained in the bounding rectangle of the other. If not, then the triangle cannot be contained.

- **Vertex position:** If the edges do not intersect, one triangle might be completely contained in another. We can test this by determining that all three vertexes of one triangle lie on the "inside" of all three edges of the other triangle. Take the vertexes of triangle 1 in CW order to be (x_1, y_1) and (x_2, y_2). Let (x, y) be a vertex of triangle 2. Define

$$f(x, y) = (x - x_1)(y_2 - y_1) - (x_2 - x_1)(y - y_1)$$

  If $f(x, y) > 0$, then the vertex is on the "inside" side of the edge joining (x_1, y_1) to (x_2, y_2).

Depth Comparison

Goal: Find (x, y, z_1) on triangle A and (x, y, z_2) on triangle B and compare the z's.

Two cases:

1. **Edge Intersection:** Use the intersection point.
2. **Containment:** Use any vertex of the contained triangle.

If $|z_1 - z_2| \geq \epsilon$, we know which triangle is closer.
If $|z_1 - z_2| < \epsilon$ then we must find a second point.

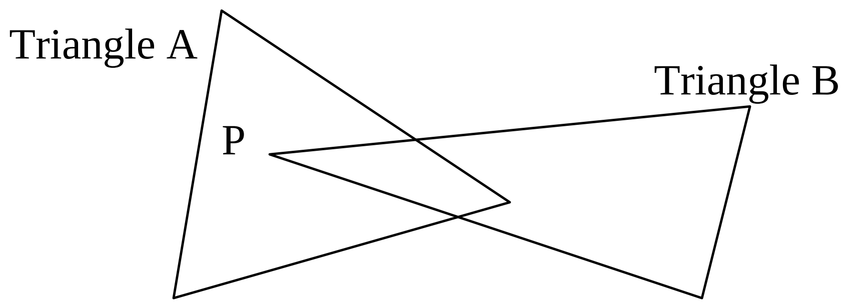
Finding a Second or Third Point

Two cases:

1. **Edge Intersection:** There must exist a second edge crossing. Find and use it.
2. **Containment:** Use another vertex of the contained triangle.

Again, $|z_1 - z_2| < \epsilon$ may occur. In this case, we need a third point.

1. **Edge Intersection:** Search for a third edge intersection. If found, use it.
   If not found, then a vertex of one triangle must be contained in the other, so test for vertex containment.
2. **Containment:** Use the last vertex of the contained triangle.



Depth Comparison

Let (x, y) be the common point.

Let $A_1x + B_1y + C_1z_1 - D_1 = 0$ be the plane equation for triangle 1.

Let $A_2x + B_2y + C_2z_2 - D_2 = 0$ be the plane equation for triangle 2.

We can solve these for z_1 and z_2. Triangle 1 is in front of Triangle 2 iff

$$\frac{D_1 - A_1x - B_1y}{C_1} < \frac{D_2 - A_2x - B_2y}{C_2}.$$

Data Structures

All of the front facing triangles in a scene are converted into `dsrec_t` objects. Each `dsrec_t` has the following fields:

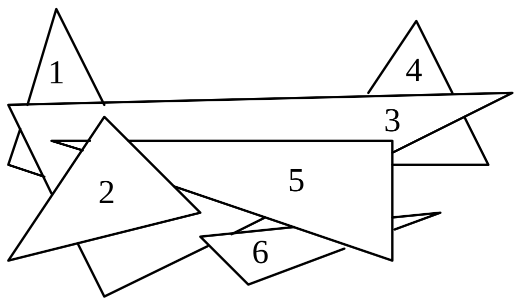
`num_behind` This counts the number of triangles behind this triangle.

`infront` A linked list of triangles in front of us.

`orig` A pointer to the original (untransformed) triangle.

`trans` A pointer to an array of transformed vertexes.

triangle	counter	list
1	0	3 5 2
2	3	
3	2	2 5 6
4	0	3 5
5	4	2
6	1	5



Painter's Algorithm

- Find a triangle with `num_behind` $= 0$. Draw the `orig` triangle (with `solid`).
- Go through the linked list of `infront` triangles, and decrement each of their `num_behind` counters. Set our `num_behind` counter to -1.
- Repeat until all triangles have `num_behind` $= -1$.

Code

```
for (int i = 0; i < ffcounter−1; i++)       // establish depth order
  for (int j = i+1; j < ffcounter; j++)
    if (∗dsrec[i] ^ ∗dsrec[j])       // triangle i overlaps triangle j
    if (∗dsrec[i] < ∗dsrec[j]) {  // triangle i closer than triangle j
      dsrec[i]→num_behind++;
      dsrec[j]→infront = new infront_t(i,dsrec[j]→infront);
    } else {
      dsrec[j]→num_behind++;
      dsrec[i]→infront = new infront_t(j,dsrec[i]→infront);
    }
for (i = 0; i < ffcounter; i++) {             // painter's algorithm
  int j = 0 ;
  while (dsrec[j]→num_behind) j++;             // find null entry
  dsrec[j]→num_behind = −1;
  for (infront_t∗ f = dsrec[j]→infront ; f; f = f→next)
    dsrec[f→tri]→num_behind−−;
  dsrec[j]→orig→solid(this);
}
```