EECS 487: Interactive Computer Graphics

Lecture 6:

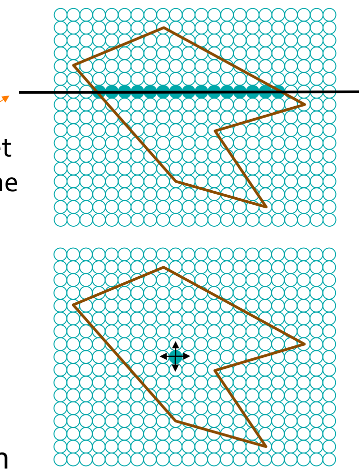
- Triangle rasterization
- Polygon clipping

Polygon Rasterization

Takes shapes like triangles and determines which pixels to set

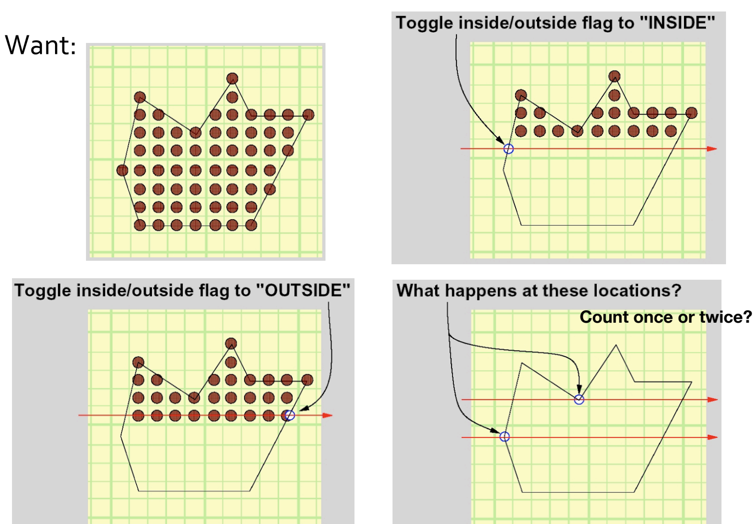
1. Polygon scan-conversion
   - sweep the polygon by scan line, set the pixels whose center is inside the polygon for each scan line
2. Polygon fill
   - select a pixel inside the polygon
   - grow outward until the whole polygon is filled

We'll only look at scan-conversion



Merrello8

Odd-Parity Polygon Rasterization



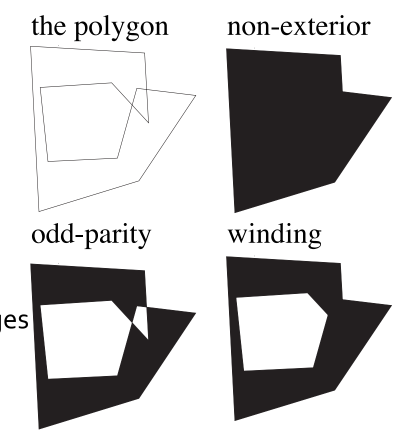
O'Brieno8

What's Inside?

Tests for what's inside:

- odd-parity rule:
  odd-crossing means inside
- non-exterior rule: a point is inside if every ray to infinity intersects the polygon
- winding rule:
  inside if walking along the edges encircles a point ≥ 1 times

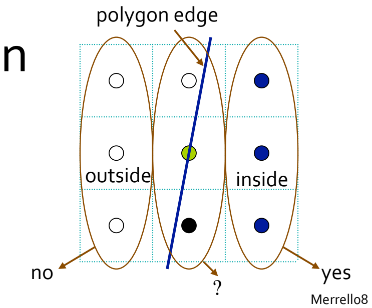
which is right?
rather arbitrary . . .



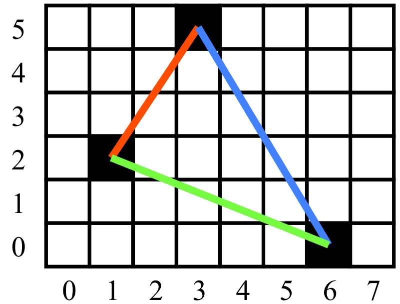
How to ensure correct scan conversion of a polygon?

Triangle Rasterization

Two questions:
- which pixel to set?
- what color to set each pixel to?

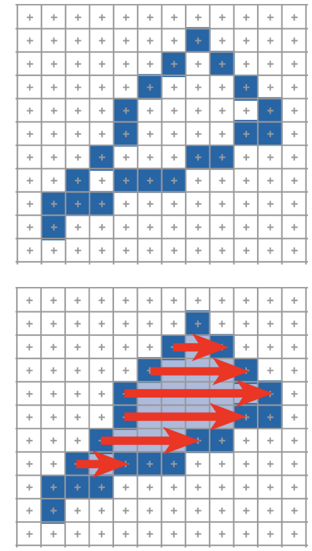


How would you rasterize a triangle?
1. Edge-walking
2. Edge-equation
3. Barycentric-coordinate based



Edge Walking

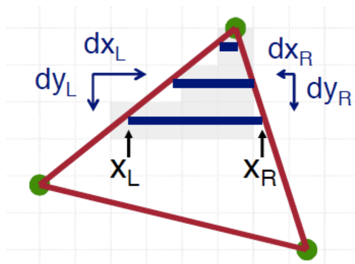
Idea:
- scan top to bottom in scan-line order
- “walk” edges: use edge slope to update coordinates incrementally
- on each scan-line, scan left to right (horizontal span), setting pixels
- stop when bottom vertex or edge is reached



Durand09

Edge Walking

```
void edge_walking(vertices T[3])
{
  for each edge pair of T {
    initialize x_L, x_R;
    compute dx_L/dy_L and dx_R/dy_R;
    for scanline at y {
      for (int x = x_L; x <= x_R; x++) {
        set_pixel(x, y);
      }
    }
    x_L += dx_L/dy_L;
    x_R += dx_R/dy_R;
  }
}
```

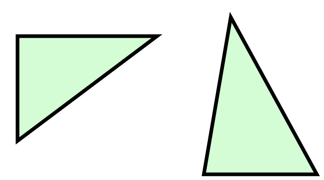


Funkhouser09

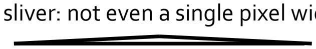
Edge Walking

Advantage: very simple

Disadvantages:
- very serial (one pixel at a time) ⇒ can’t parallelize
- inner loop bottleneck if lots of computation per pixel
- special cases will make your life miserable
  - horizontal edges: computing intersection causes divide by 0!

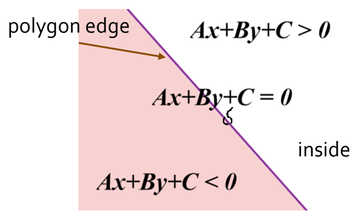


- sliver: not even a single pixel wide



Edge Equations

1. compute edge equations from vertices
   - orient edge equations: let negative halfspaces be on the triangle's exterior (multiply by -1 if necessary)
2. scan through each pixel and evaluate against all edge equations
3. set pixel if all three edge equations > 0

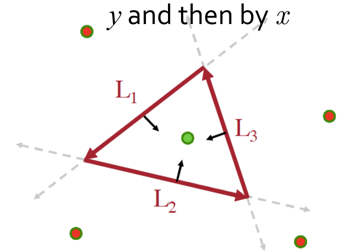


Luzano&Popovic01

Edge Equations

```
void edge_equations(vertices T[3])
{
  bbox b = bound(T);
  foreach pixel(x, y) in b {
    inside = true;
    foreach edge line Li of Tri {
      if (Li.A*x+Li.B*y+Li.C < 0) {
        inside = false;
      }
    }
    if (inside) {
      set_pixel(x, y);
    }
  }
}
```

can be rewritten to update the L's incrementally by y and then by x



Hanrahan/Funkhouser09/Durand08

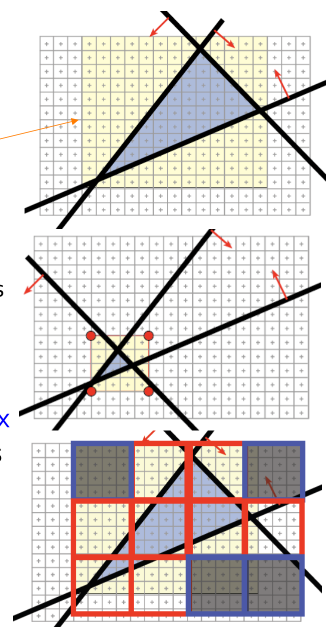
Edge Equations

Can we reduce #pixels tested?

1. compute a bounding box: $x_{min}, y_{min}, x_{max}, y_{max}$ of triangle
2. compute edge equations from vertices
   - orient edge equations: let negative halfspaces be on the triangle's exterior (multiply by -1 if necessary)
   - can be done incrementally per scan line
3. scan through *each* pixel in bounding box and evaluate against all edge equations
4. set pixel if all three edge equations > 0

Hierarchical bounding boxes

- how to quickly exclude a bounding box?



Durand09

What Color the Pixel?

Both edge walk and edge equations tell you if a pixel is inside a triangle, but how to color the pixel?

As with coloring a line, want pixel color to be an interpolation of triangle's vertex colors

How much is the color of a pixel influenced by each vertex? How do we weigh each vertex's color in setting the color of a pixel?

Linear, Affine, Convex Combinations

Given points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots, \mathbf{p}_n$
and coefficients $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$:

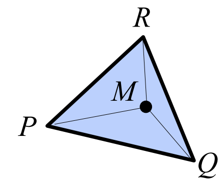
- linear combination:
  $\alpha_1\mathbf{p}_1 + \alpha_2\mathbf{p}_2 + \alpha_3\mathbf{p}_3 + \ldots + \alpha_n\mathbf{p}_n$
- affine combination:
  linear combination with $\alpha_1 + \alpha_2 + \alpha_3 + \ldots + \alpha_n = 1$
- convex combination:
  affine combination with each $\alpha_i \geq 0$

More generally, the convex combinations of a set of points is the convex hull of those points

- geometric interpretation?

  The smallest convex polygon such that all points in the given set are either internal or boundary points

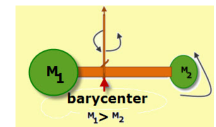
Triangle Rasterization with Barycentric Coordinates



A triangle is a set of all possible convex combinations (convex hull) of three points

- $\triangle PQR = \{\alpha P + \beta Q + \gamma R$ for all $\alpha, \beta, \gamma \geq 0$, and $\alpha + \beta + \gamma = 1\}$

  ⇒ a point $M = \alpha P + \beta Q + \gamma R$ is inside the triangle iff $\alpha + \beta + \gamma = 1$, and $\alpha, \beta, \gamma \geq 0$



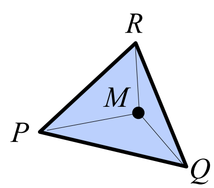
[VanCleave]

- works in any dimension!
- α, β, γ are called the barycentric coordinates
  - barus: Greek for "heavy"
    barycenter: the balance point between two objects
  - incidentally, the interpolating parameters t and $(1 - t)$ in the parametric equation of a 2D line are also barycentric coordinates

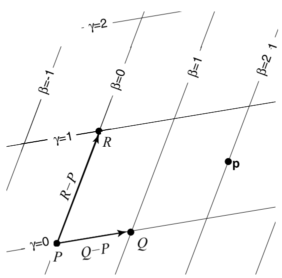
Geometric interpretation?

Barycentric Coordinates

Think of $\overrightarrow{PQ}$ and $\overrightarrow{PR}$ as (non-orthogonal) basis of the plane defined by the three vertices of $\triangle PQR$



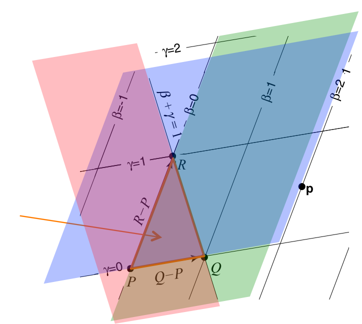
Any point $\mathbf{p}$ in the plane is an affine combination:
$\mathbf{p} = P + \beta(Q - P) + \gamma(R - P) = (1 - \beta - \gamma)P + \beta Q + \gamma R$



Barycentric Coordinates

For the triangle, we're only interested in the points where $\beta + \gamma \leq 1$ and $\beta \geq 0$ and $\gamma \geq 0$

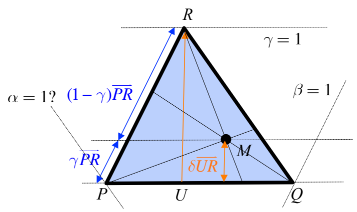
- for $\alpha + \beta + \gamma = 1 \Rightarrow 0 \leq \alpha \leq 1$
  and we have ourselves a convex combination:
  $\Delta PQR = \{\alpha P + \beta Q + \gamma R,$
  $\forall\, \alpha, \beta, \gamma \mid$
  $\alpha + \beta + \gamma = 1$ and $\alpha, \beta, \gamma \geq 0\}$



- for the mathematically inclined, where are the points when $\alpha + \beta + \gamma \neq 1$?

Barycentric Coordinates

Given a point in triangle, how to compute its γ (or α, β)?



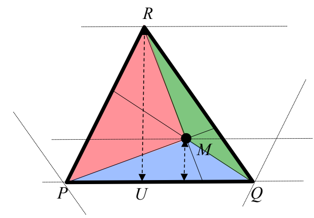
1. using the implicit line equation of the triangle edges: let $\hat{\mathbf{n}}_{\overline{PQ}} = \overline{UR}/\|\overline{UR}\|$ be the normalized normal of $\overline{PQ}$, $\delta\overline{UR}$ is a projection of $\gamma\overline{PR}$ (and of $\overline{PM}$) onto $\hat{\mathbf{n}}_{\overline{PQ}}$ then:

$$\|\delta\overline{UR}\| = \hat{\mathbf{n}}_{\overline{PQ}} \cdot \gamma\overline{PR} = \hat{\mathbf{n}}_{\overline{PQ}} \cdot \overline{PM}$$

$$\gamma = \frac{\hat{\mathbf{n}}_{\overline{PQ}} \cdot \overline{PM}}{\hat{\mathbf{n}}_{\overline{PQ}} \cdot \overline{PR}} = \frac{f_{\overline{PQ}}(M)}{f_{\overline{PQ}}(R)}$$

2. using the signed area of the triangle and sub-triangles

Barycentric Coordinates



Think of $\overline{PQ}$ as the base of both $\triangle PQR$ and $\triangle PQM$

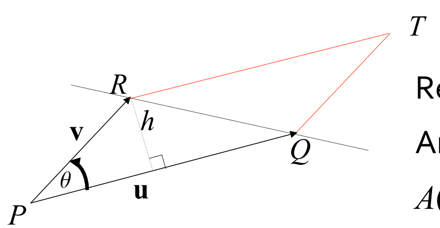
If P, Q, R are not collinear (on a line, degenerate triangle), the signed area $S(\triangle PQR) \neq 0$, and we can find (α, β, γ) for M as:

$$\alpha = \frac{S(\triangle MQR)}{S(\triangle PQR)},\ \beta = \frac{S(\triangle MRP)}{S(\triangle PQR)},\ \gamma = \frac{S(\triangle MPQ)}{S(\triangle PQR)}$$

What's a signed area?

How do you compute the signed area of a triangle?

Computing the Area of a Triangle



Recall $\sin\theta = h / \|\mathbf{v}\|$

Area computation:

$$\begin{aligned} A(PQTR) &= \|\mathbf{u}\|\ h \\ &= \|\mathbf{u}\|\ \|\mathbf{v}\| \sin\theta \\ &= \|\mathbf{u}\times\mathbf{v}\| \\ A(\triangle PQR) &= \tfrac{1}{2}\ \|\mathbf{u}\times\mathbf{v}\| \end{aligned}$$

Works for any triangle

What is $0.5|(\mathbf{q}-\mathbf{p})\times(\mathbf{r}-\mathbf{p})|$ where $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$ are points in 3D?

Note:

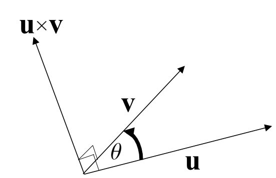
1. the notation $A(\triangle PQR)$ means the vertices of the triangle are visited in the right-handed manner: P to Q to R
2. $A(\triangle PQR) = A(\triangle QRP) = A(\triangle RPQ)$

Cross Product in 3D Review

The cross product $\mathbf{u}\times\mathbf{v}$ is a vector orthogonal to the plane of $\mathbf{u}$ and $\mathbf{v}$ and has length

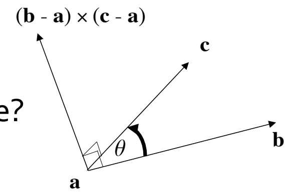
$$\|\mathbf{u}\times\mathbf{v}\| = \|\mathbf{u}\|\ \|\mathbf{v}\| \sin\theta$$

Vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{u}\times\mathbf{v}$ form a right-handed system



Since $\mathbf{u}\times\mathbf{v} \perp \mathbf{u}$ and $\mathbf{u}\times\mathbf{v} \perp \mathbf{v}$, $(\mathbf{u}\times\mathbf{v})\bullet\mathbf{u} = (\mathbf{u}\times\mathbf{v})\bullet\mathbf{v} = 0$

Normal Vectors



What is the normal vector of a plane?
A vector perpendicular to the plane

What is a unit normal?
Normal vector of magnitude one: $\mathbf{n}/||\mathbf{n}||$

If a planar surface contains the points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, a normal can be computed as: $\mathbf{n} = (\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})$

If $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are arranged counter-clockwise, the normal points outward and the surface is the front of the polygon; otherwise, the surface is the back of the polygon and the normal points inward

Cross Product in 3D Review

Since $\mathbf{u}\times\mathbf{v} \perp \mathbf{u}$ and $\mathbf{u}\times\mathbf{v} \perp \mathbf{v}$, $(\mathbf{u}\times\mathbf{v})\bullet\mathbf{u} = (\mathbf{u}\times\mathbf{v})\bullet\mathbf{v} = 0$

Example: $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Cross product can be computed using determinant:

$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} \times \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix} = \begin{vmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \\ u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{vmatrix}$$

$$= \underbrace{\begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}}_{\text{scalar}} \underbrace{\mathbf{e}_x}_{\text{vector}} + \underbrace{\begin{vmatrix} u_2 & u_0 \\ v_2 & v_0 \end{vmatrix}}_{\text{scalar}} \underbrace{\mathbf{e}_y}_{\text{vector}} + \underbrace{\begin{vmatrix} u_0 & u_1 \\ v_0 & v_1 \end{vmatrix}}_{\text{scalar}} \underbrace{\mathbf{e}_z}_{\text{vector}}$$

Cross Product in 3D Review

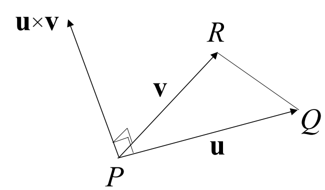
For $\mathbf{u}\times\mathbf{v} = \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} \mathbf{e}_x + \begin{vmatrix} u_2 & u_0 \\ v_2 & v_0 \end{vmatrix} \mathbf{e}_y + \begin{vmatrix} u_0 & u_1 \\ v_0 & v_1 \end{vmatrix} \mathbf{e}_z$

with the standard basis: $\mathbf{e}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{e}_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{e}_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

we get: $\begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} \times \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} u_1v_2 - u_2v_1 \\ u_2v_0 - u_0v_2 \\ u_0v_1 - u_1v_0 \end{bmatrix}$

Example: $\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 4-1 \\ 0-8 \\ 2-0 \end{bmatrix} = \begin{bmatrix} 3 \\ -8 \\ 2 \end{bmatrix}$

Signed Area in 2D



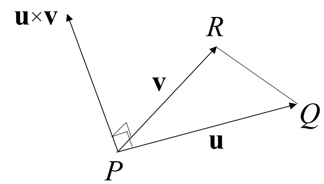
Triangle $\triangle PQR$ in 2D:

$$P = \begin{bmatrix} p_0 \\ p_1 \end{bmatrix}, Q = \begin{bmatrix} q_0 \\ q_1 \end{bmatrix}, R = \begin{bmatrix} r_0 \\ r_1 \end{bmatrix}$$

To compute the area of $\triangle PQR$:
First extend points to 3D
(assume points to be on $z = 0$ plane)

$$\mathbf{u} = \overrightarrow{PQ} = \begin{bmatrix} q_0 - p_0 \\ q_1 - p_1 \\ 0 \end{bmatrix}, \mathbf{v} = \overrightarrow{PR} = \begin{bmatrix} r_0 - p_0 \\ r_1 - p_1 \\ 0 \end{bmatrix}$$

Signed Area in 2D



$$A(\triangle PQR) = \frac{1}{2}\|\mathbf{u}\times\mathbf{v}\|$$

$$\|\mathbf{u}\times\mathbf{v}\| = \left\| \begin{bmatrix} u_0 \\ u_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} v_0 \\ v_1 \\ 0 \end{bmatrix} \right\| = \left\| \begin{bmatrix} 0 \\ 0 \\ u_0v_1 - u_1v_0 \end{bmatrix} \right\|$$

$$= \left|\sqrt{0+0+(u_0v_1-u_1v_0)^2}\right|$$

$$= |u_0v_1 - u_1v_0|$$

$\mathbf{u}\times\mathbf{v}$ is a vector with only the z-element, $||\mathbf{u}\times\mathbf{v}||$ is just the length of the vector

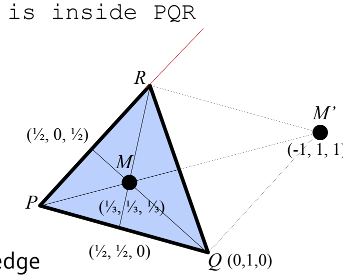
Which is also: $\det\begin{bmatrix} u_0 & v_0 \\ u_1 & v_1 \end{bmatrix} = u_0v_1 - u_1v_0$

except that the determinant preserves the sign, which indicates the direction of $\mathbf{u}\times\mathbf{v}$

Triangle Rasterization with Barycentric Coordinates

```
Given a triangle PQR (non-degenerate)
for each pixel M in the bounding box of PQR {
  check whether M is inside PQR
}
```

For example:

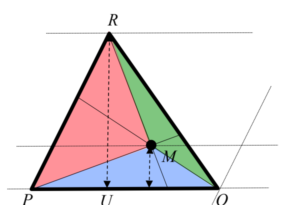


1 coordinate $= 0 \Rightarrow M$ on Δ edge

2 coordinates $= 0$, 1 coordinate $= 1 \Rightarrow M$ on Δ vertex

All points on the $\overline{QM'}$ line are equidistant from the $\overline{PR}$ line and has $\beta = 1$; similarly, all points on the $\overline{RM'}$ line has $\gamma = 1$

Considerations



- To compute the ratio $S(\triangle MPQ)/S(\triangle PQR)$, no need to divide the signed area by 2 …
- If area instead of signed area is computed, to check whether a sub-triangle is of the same sign as the triangle, compute the dot product of the normals, e.g., signof γ = signof $((\overline{MP}\times\overline{MQ})\cdot(\overline{RP}\times\overline{RQ}))$
- $S(\triangle MPQ) = \frac{1}{2} * \left\|\overline{PQ}\right\| * height(M \text{ from } \overline{PQ})$

  $S(\triangle PQR) = \frac{1}{2} * \left\|\overline{PQ}\right\| * height(R \text{ from } \overline{PQ})$

  $$\frac{S(\triangle MPQ)}{S(\triangle PQR)} = \frac{height(M \text{ from } \overline{PQ})}{height(R \text{ from } \overline{PQ})} \text{ but that's} = \frac{f_{\overline{PQ}}(M)}{f_{\overline{PQ}}(R)} \text{ hm} \ldots$$

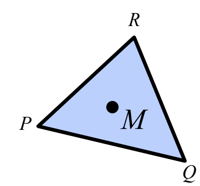
  (which is cheaper to compute?)

Coloring a Triangle

Let:

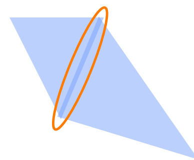
- P, Q, R be the vertices of the triangle
- $\mathbf{c}_P$, $\mathbf{c}_Q$, $\mathbf{c}_R$ their respective colors

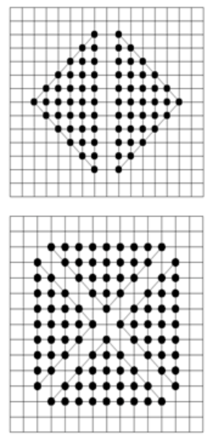
How do we color point M as a blend of the color of the three vertices?



- compute the barycentric coordinates of M
- combine colors using those coordinates: $\mathbf{c}_M = \alpha\,\mathbf{c}_P + \beta\,\mathbf{c}_Q + \gamma\,\mathbf{c}_R$

Shared Edges





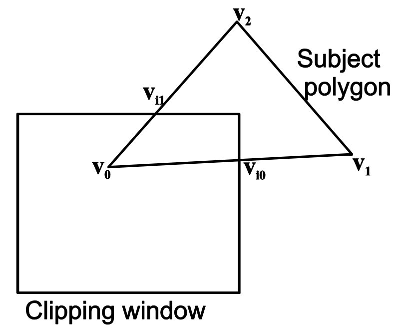
Don't want to double draw the edge, especially if both triangles are transparent

Don't want to skip the edge either!

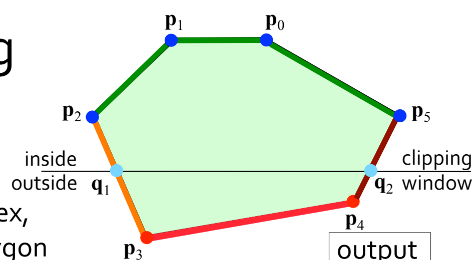
Trick: use an off-screen point, e.g., $(-1, -1)$, and "award" the edge to the triangle whose third (non-shared) vertex is in the same (or opposite) side as the off-screen point (how to determine sided-ness?)

Polygon Clipping

Can we simply reduce the problem to line clipping of the edges?



Polygon Clipping



output
p_1
p_2
q_1
q_2
p_5
p_0

Sutherland-Hodgman "edge walking"

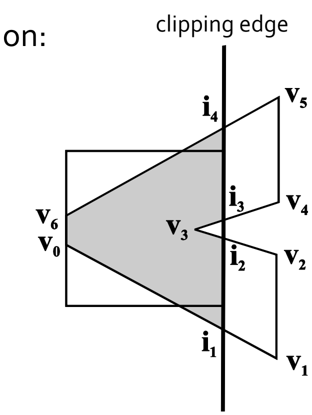
- starting from an inside vertex, traverse the vertices of polygon in order
- do an inside/outside test for each next vertex
  - in-in: output next vertex
  - in-out: output intersection
  - out-out: output nothing
  - out-in: output intersection and next vertex

(there are other more efficient and complex algorithms, e.g., Liang-Barsky parametric space and clipping in 4D)

Polygon Clipping

Also works with concave polygon:

v_k	v_{k+1}	Case	Output
v_0	v_1	i-o	i_1
v_1	v_2	o-o	-
v_2	v_3	o-i	i_2, v_3
v_3	v_4	i-o	i_3
v_4	v_5	o-o	-
v_5	v_6	o-i	i_4, v_6
v_6	v_0	i-i	v_0



then continue with other clipping edges . . .