Acceleration techniques

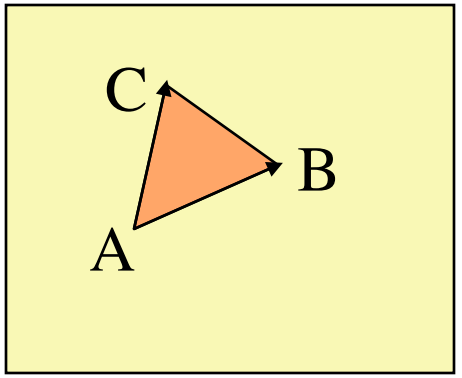
- Reduce per pixel cost
  - Reduce resolution
  - Back-face culling
- Reduce per vertex cost
  - T-strips and fans
  - Frustum culling
  - Simplification
  - Texture map
- Reduce both
  - Occlusion culling

Rendering costs

- Per vertex
  - Lighting calculations
  - Perspective transformation
  - Slopes calculation
- Per pixel
  - Interpolations of z, c **(and leading trailing edges, amortized)**
  - Read, compare, right to update the z-buffer and frame buffer

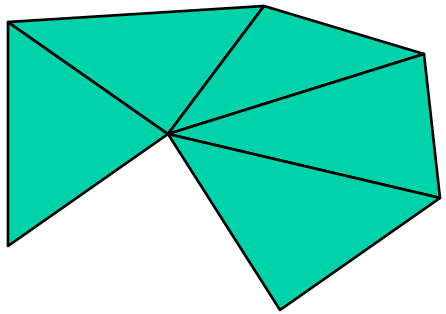
Back-face culling

- Done in screen space by hardware
- Test sign of $(x_B-x_A)\,(y_C-y_A) - (x_C-x_A)\,(y_B-y_A)$

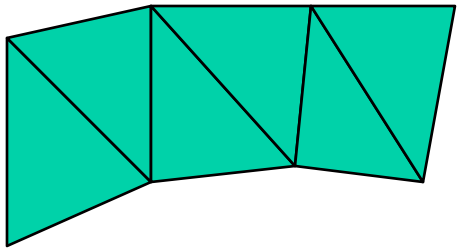


T-strips and fans

- Reuse 2 vertices of last triangle



fan



strip

Was important when hardware had only 3 registers

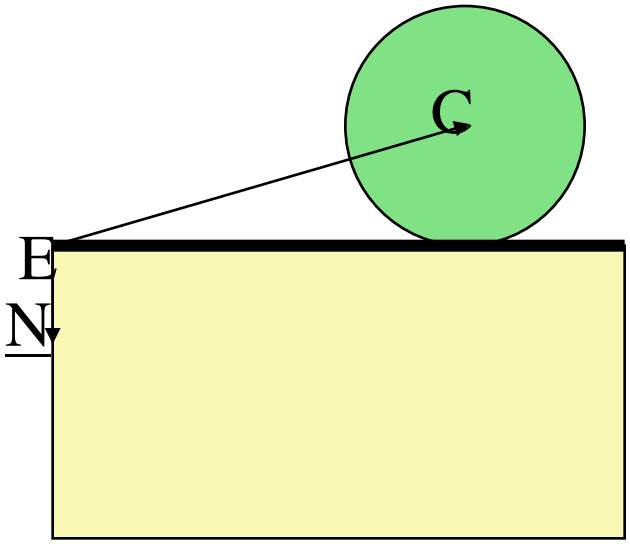
Saves nearly 2/3 of vertex processing

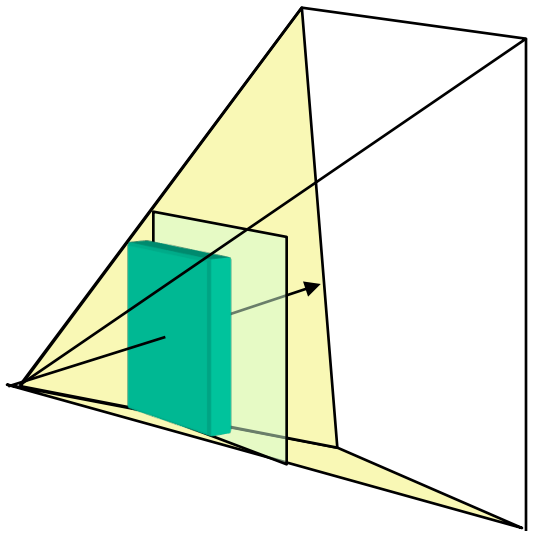
A vertex is still processed twice on average

Less important in modern graphics boards

Frustum culling

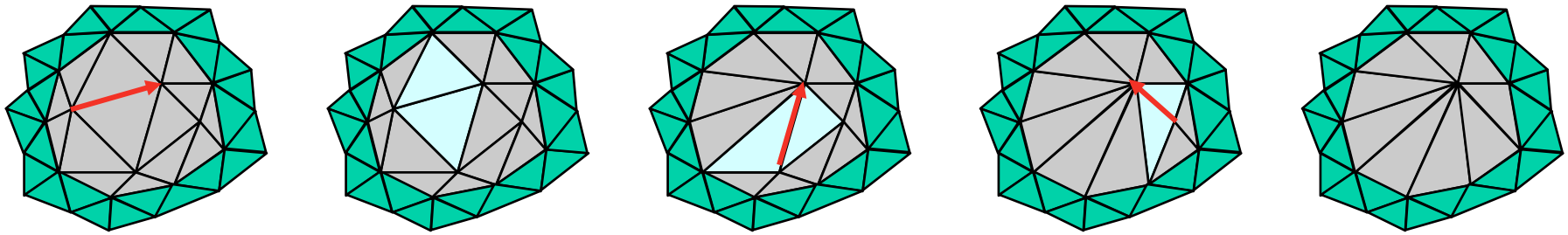
- Build a tight sphere around each object
  - Center it at C=((x_{max}+x_{min})/2, (y_{max}+y_{min})/2, (z_{max}+z_{min})/2)
  - Compute radius r as min||CV|| for all vertices V
- Test the sphere against each half-space of the frustum
  - **If N•(EC)<-r then sphere is outside**

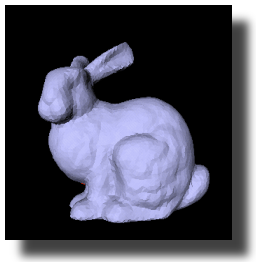


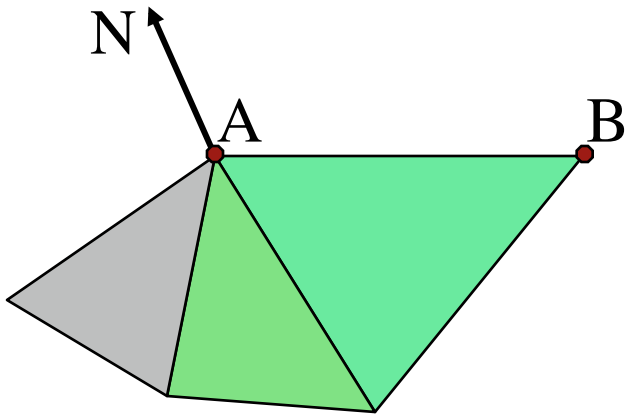


Simplification

- Collapse edges

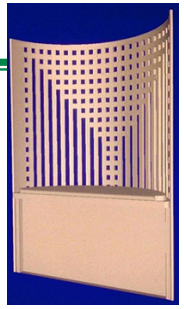




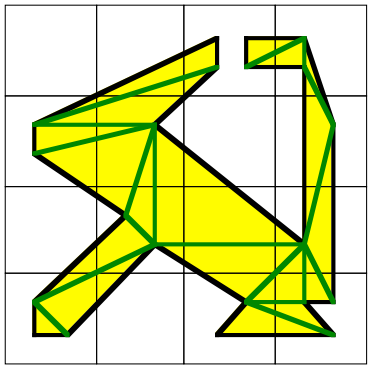


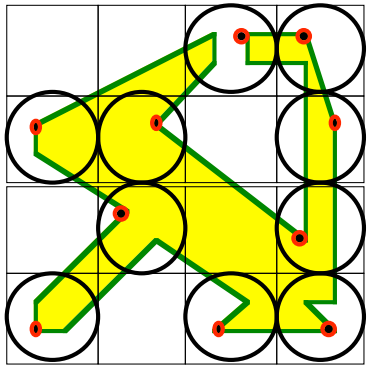
Collapse A to B if $|N \bullet (AB)| < e$ Or use better error estimates

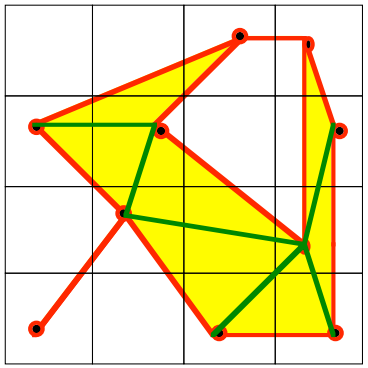
Vertex clustering (Rossignac-Borrel)

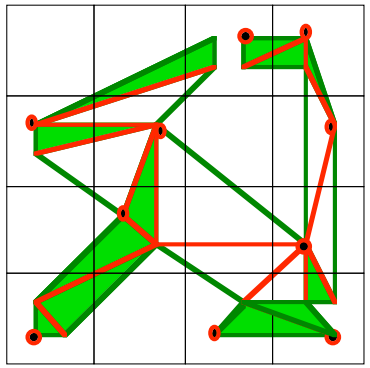


- Subdivide box around object into grid of cells
- Coalesce vertices in each cell into one “attractor”
- Remove degenerate triangles
  - More than one vertex in a cell
  - *Not needed for dangling edge or vertex*

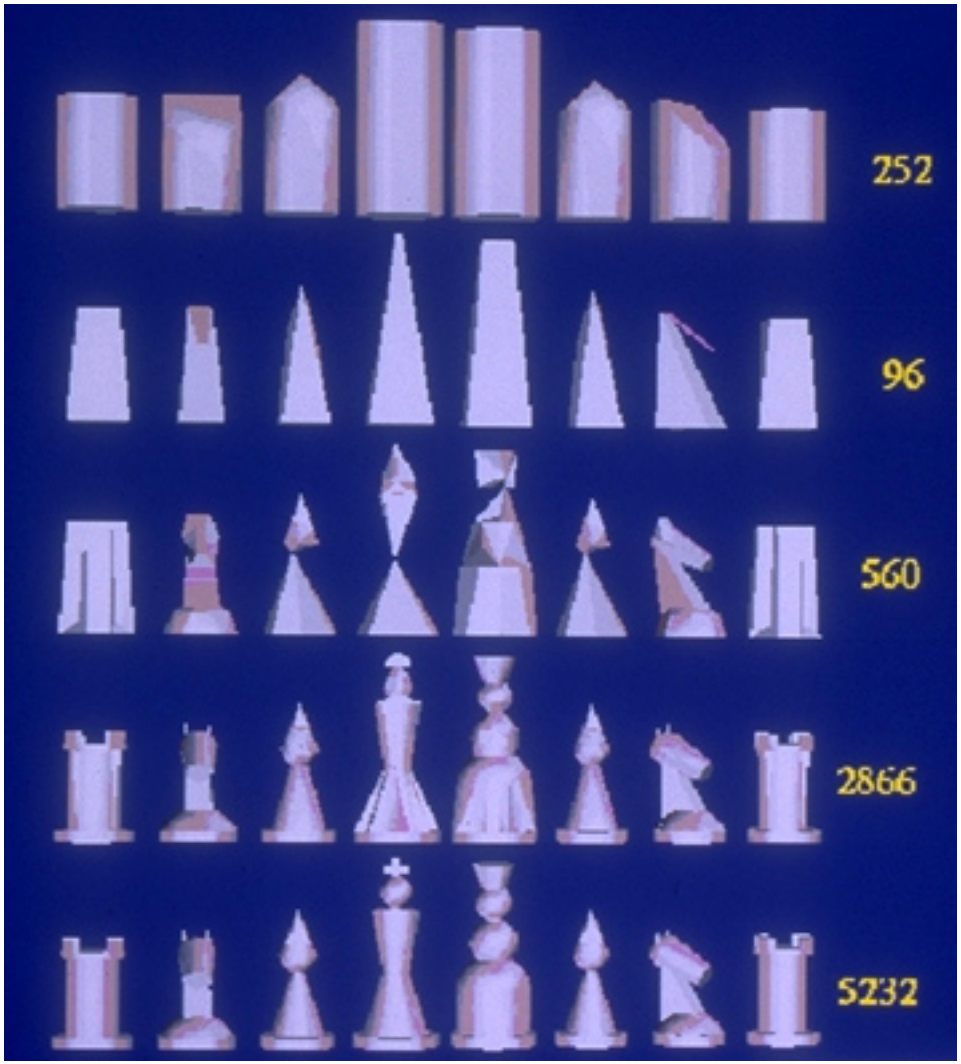






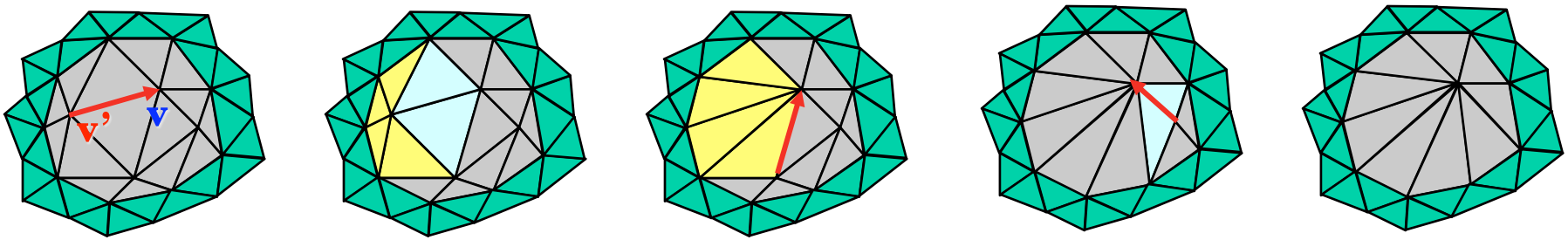


Vertex clustering example



Estimating edge-collapse error

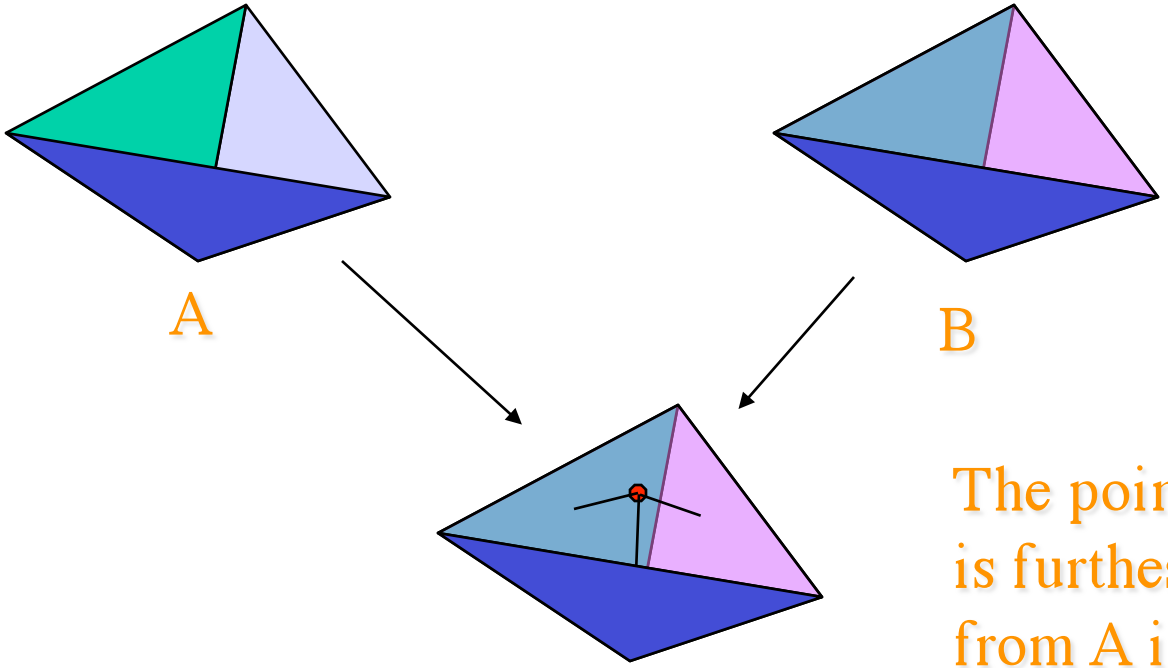
- What is the error produced by edge collapses?



- Volume of the solid bounded by old and new triangles
  - requires computing their intersection
- Exact Hausdorff distance between old and new triangles
  - expensive to compute precisely
- Distance between v and planes of all triangles incident on collapsed vertices
  - one dot product per plane (expensive, but guaranteed error bound)
- Quadratic distance between v and planes old triangles
  - evaluate one quadratic form representing all the planes (cheap but not a bound)
  - leads to closed form solution for computing the optimal V' (in least square sense)

Hausdorff distance between T-meshes

- Expensive to compute,
  - because it can occur away from vertices and edges!



The point of B that is furthest away from A is closest to 3 points on A that are inside faces

Measuring Error with Quadrics (Garland)

- Given plane P, and point v, we can define a quadric distance Q(v) = $D(v,P)^2$
  - P goes through point p and has normal N
  - D(v,P) = pv•N
  - Q(p) = $(pv•N)^2$, which is a quadratic polynomial in the coordinatges of v
  - If v=(x,y,z), Q(p)=$a_{11}x2+a_{22}y2+a_{33}z2+2a_{23}yz+2a_{13}xz+2a_{12}xy+2b_1x+2b_2y+2b_3z+c$
- Q may be written represented by 3x3 matrix A,a vector b,and a scalar c
- Q(v) may be evaluated as v•(Av)+2b•v+c

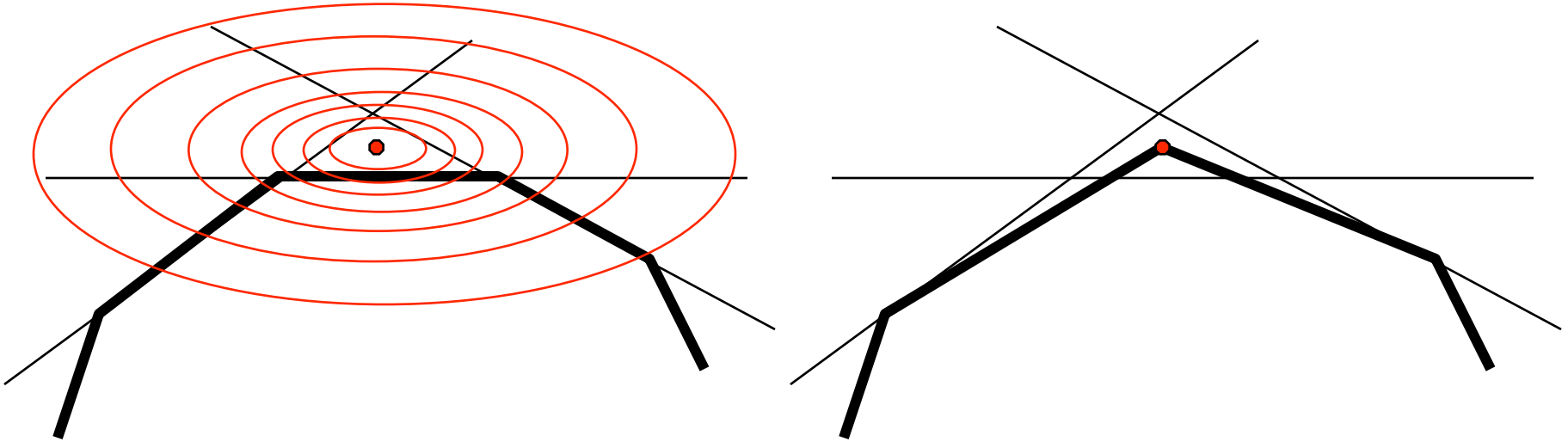
$$Q(\mathbf{v}) = \mathbf{v}^T\mathbf{A}\mathbf{v} + 2\mathbf{b}^T\mathbf{v} + c$$

  - or in matrix form as:

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + 2\begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + c$$

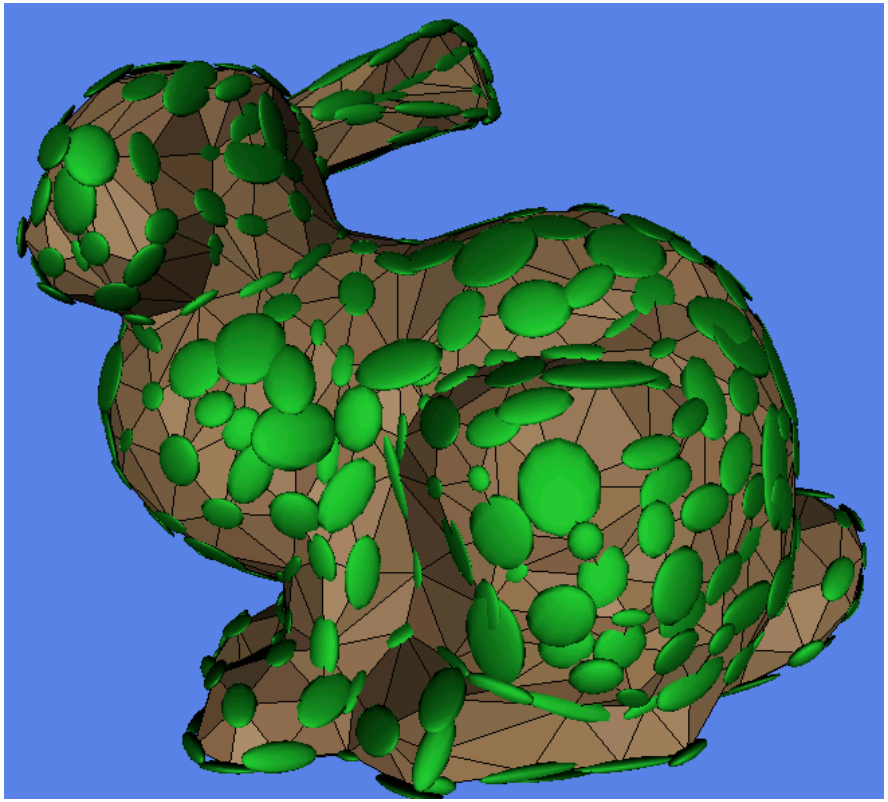
Optimal position for **v**

- Pick v to minimize average distance to planes Q(v)
  - set $dQ(v)/dv = 0$, which yields: $v = -A^{-1} b$



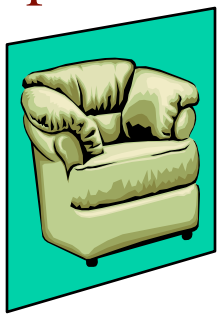
Q(**v**) measures flatness

- Isosurfaces of Q
  - Are ellipsoids
  - Stretch in flat direction
  - Have eigenvalues proportional to the principal curvatures



Texture map

- Replace distant details with their images, pasted on simplified shapes or even no flat bill-boards

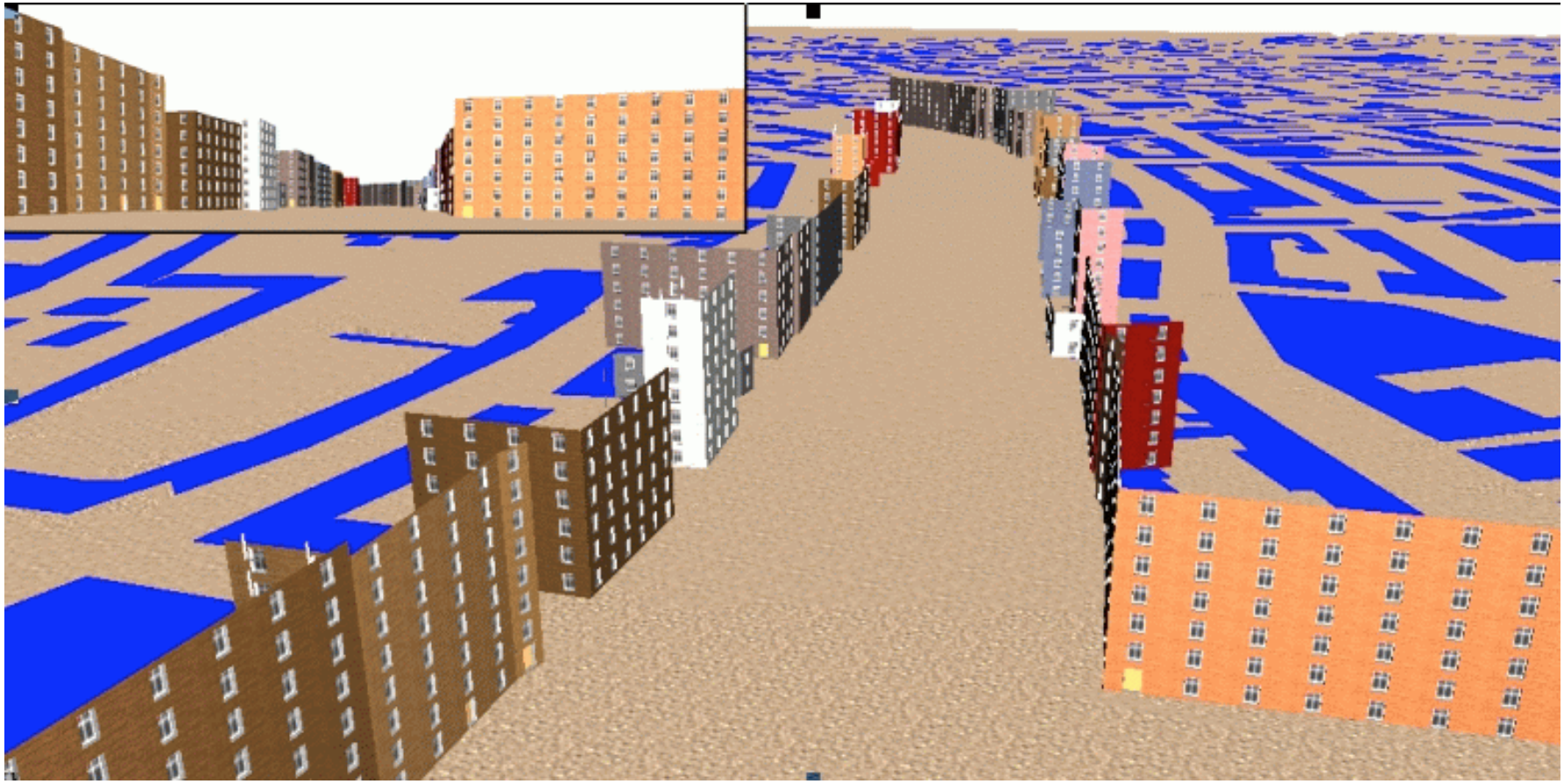


Texture mapping

```
textureMode(NORMALIZED);
PImage maya = loadImage(maya.jpg");
beginShape(QUADS);
texture(maya);
 vertex(-1,  1,  1, 1, 1);
 vertex( 1,  1,  1, 0, 1);
 vertex( 1, -1,  1, 0, 0 );
 vertex(-1, -1,  1, 1, 0);
…
endShape();
```

Occlusion culling in urban walkthrough

- From Peter Wonka
- Need only display a few facades--the rest is hidden



Shadows / visibility duality

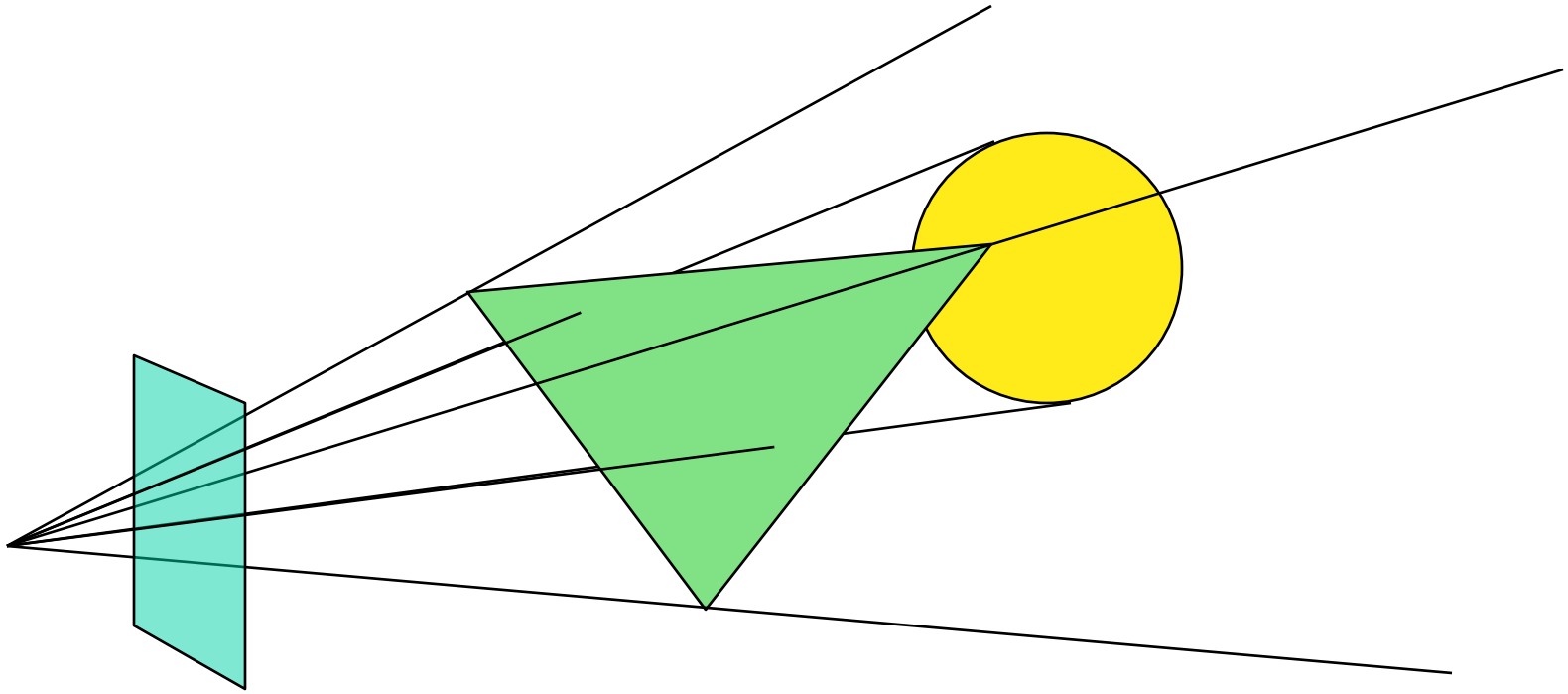
- **Point light source**
  - Shadow (volume)**: set of points that do not see the light**
- **From-point visibility**
  - Occluded space**: set of points hidden from the viewpoint**

- **Area light source (assume flat for simplicity)**
  - Umbra**: set of points that do not see any light**
  - Penumbra**: set of points that see a portion of the light source**
  - Clearing**: set of points that see the entire light source**

- **From-cell visibility (cell C)**
  - Hidden space**: set of points hidden from all points in C**
  - Detectable space**: set of points seen from at least one point in C**
  - Exposed space**: set of points visible from all points in C**

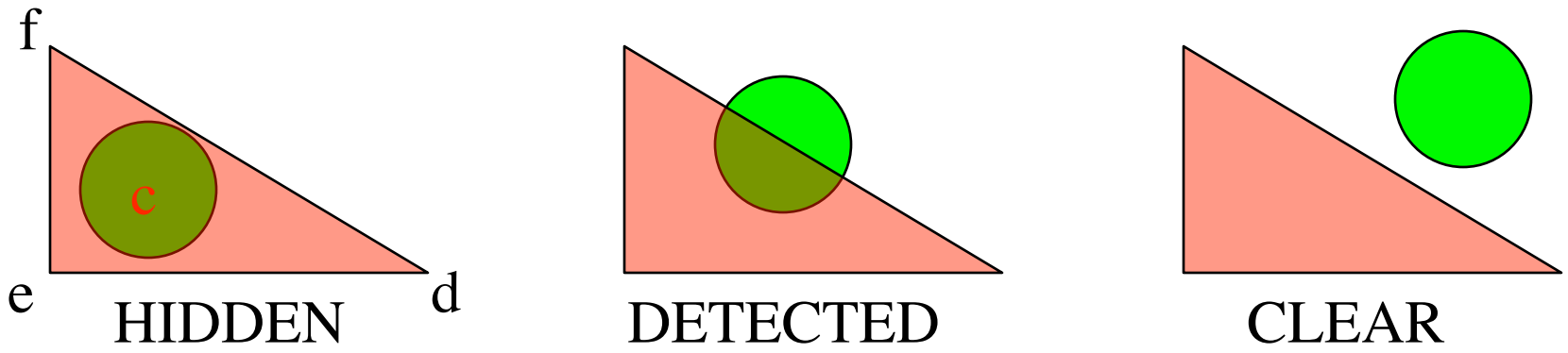
Occlusion culling

- Is ball containing an object hidden behind the triangle?



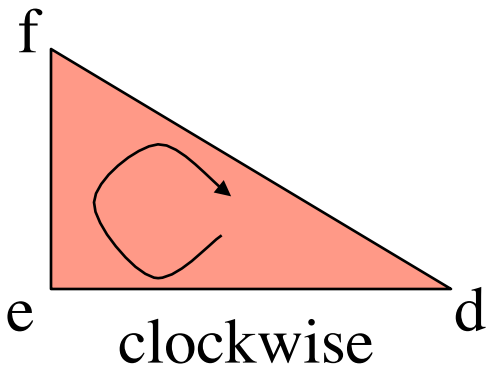
From-point visibility

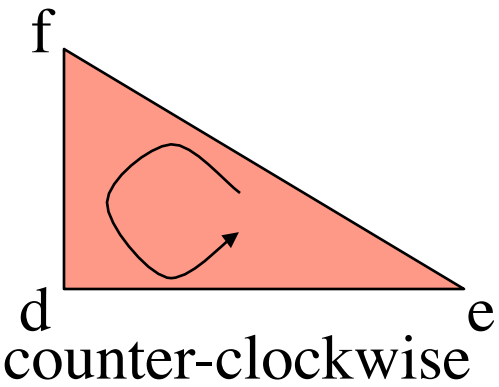
- The object is in a ball B(**c**,r) of center **c** and radius r
- The potential occluder is a triangle with vertices **d**, **e**, and **f**
- Assume viewer is at point **a**
- Write the geometric formulae for testing whether the moon is hidden behind the triangle



When does a triangle appear clockwise?

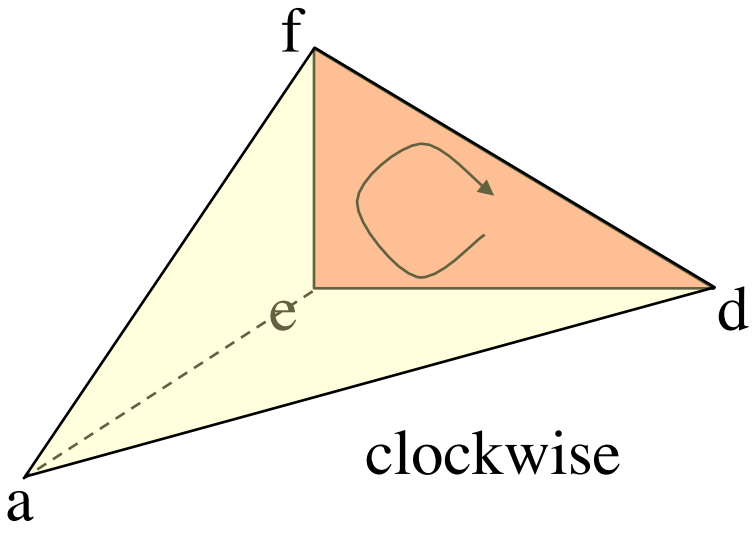
- When do the vertices **d**, **e**, **f**, of a triangle appear clockwise when seen from a viewpoint **a**?

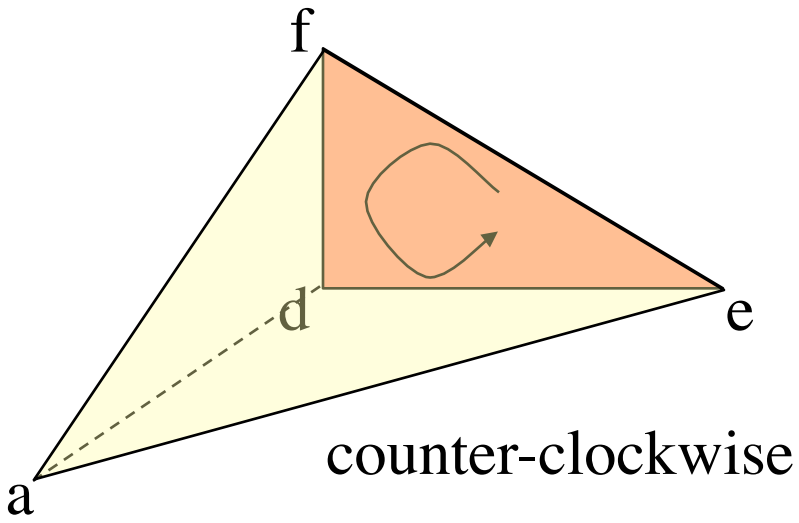




A triangle appears clockwise when

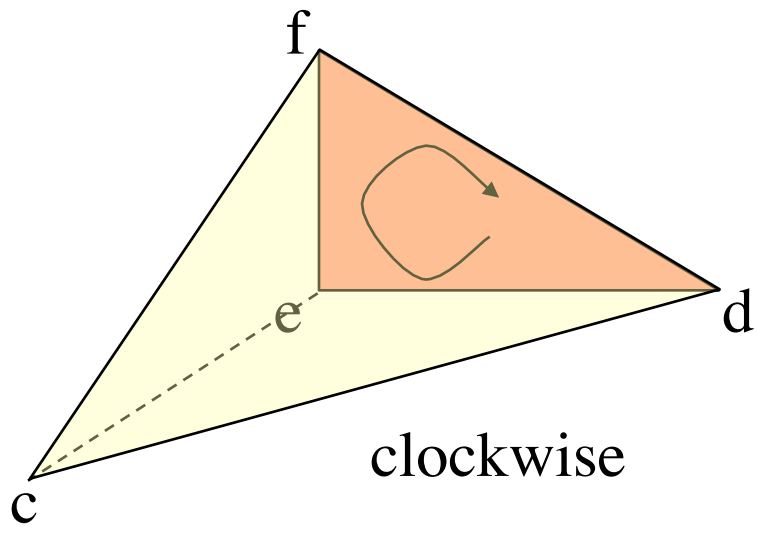
- The vertices **d**, **e**, **f**, of a triangle appear clockwise when seen from a viewpoint **a** when s(**a**,**d**,**e**,**f**) is true
- Procedure s(**a**,**d**,**e**,**f**) {RETURN **ad•(ae×af)**>0}
  - **ad•(ae×af)** is a 3x3 determinant
  - It is called the **mixed product** (or the **triple product**)





Testing a point against a half-space?

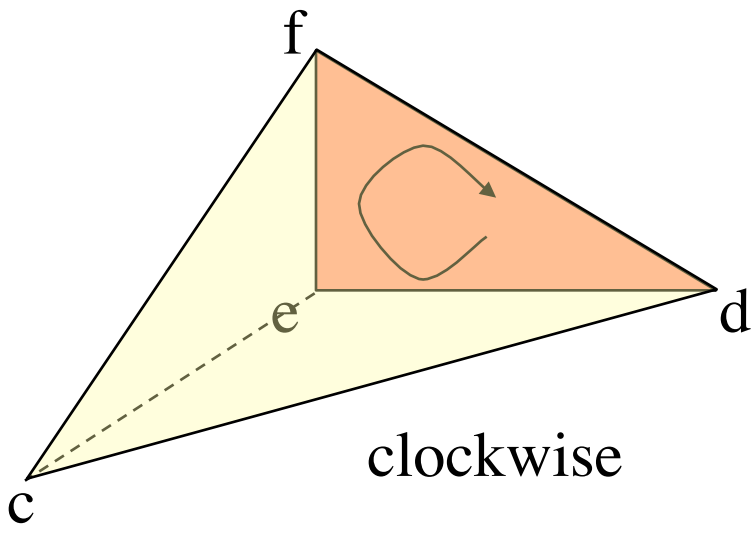
- Consider the plane through points **d**, **e**, and **f**
  - It splits space in 2 half-spaces
- Consider the half-space **H** of all points that see **d**, **e**, **f** clockwise
- How to test whether point **c** is in H?



Testing a point against a half-space

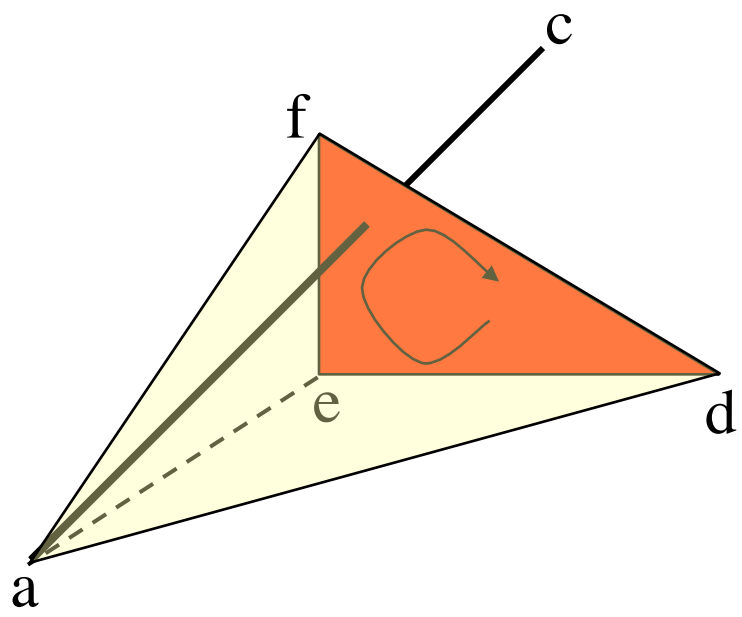
- Consider the plane through points **d**, **e**, and **f**
- Consider the half-space **H** of all points that see **d**, **e**, **f** clockwise
- A point **c** is in H when s(c,d,e,f)

s(**c**,**d**,**e**,**f**) {RETURN **cd**•(**ce**×**cf**)>0}



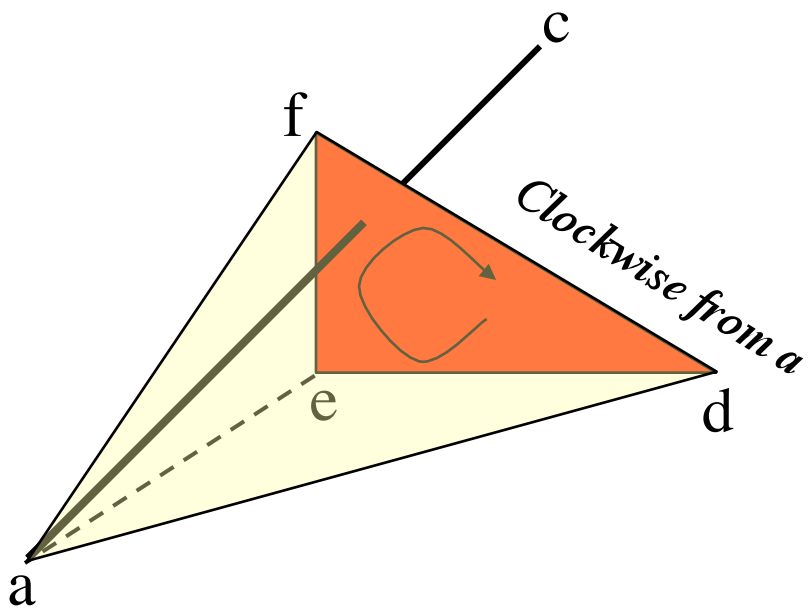
Is a point hidden by a triangle?

- Is point **c** hidden from viewpoint **a** by triangle (**d, e, f**)?
  - Write down all the geometric test you need to perform



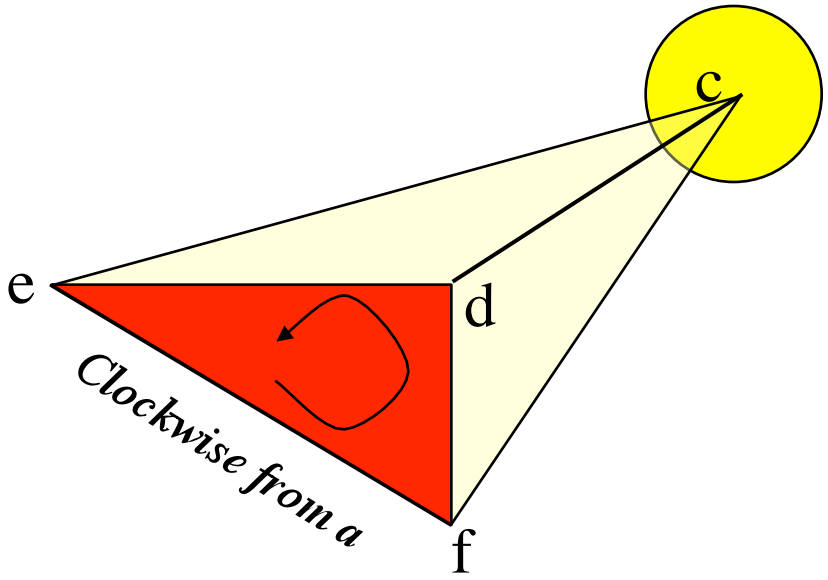
Test whether a point is hidden by a triangle

- To test whether **c** is hidden from **a** by triangle **d**, **e**, **f** do
  - if not s(a,d,e,f) then swap d and e
  - if not s(a,d,e,c) then return false
  - if not s(a,e,f,c) then return false
  - if not s(a,f,d,c) then return false
  - if s(c,d,e,f) then return false
  - return true



Testing a ball against a half-space?

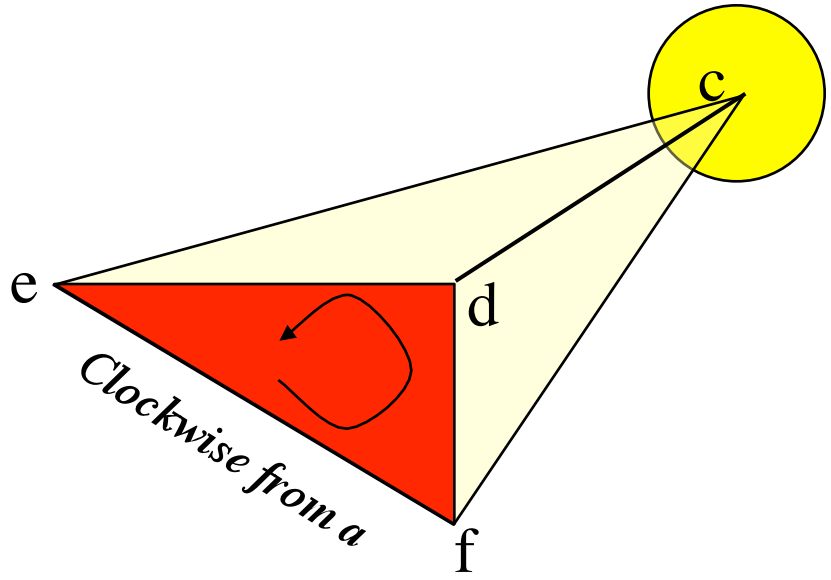
- Consider the half-space **H** of all points that see (**d**, **e**, **f**) clockwise
- When is a ball of center **c** and radius r in **H**?



Testing a ball against a half-space

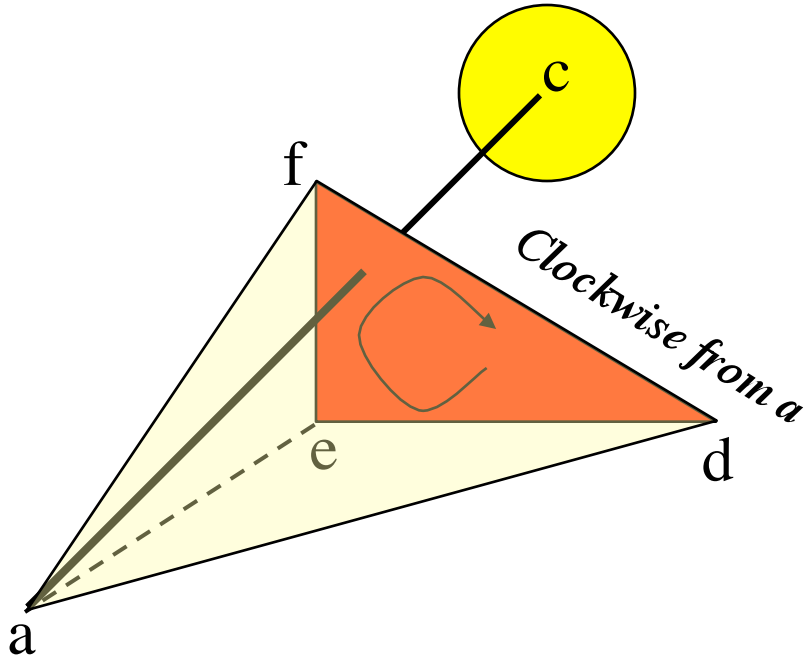
- Consider the half-space **H** of all points that see (**d, e, f**) clockwise
- A ball of center **c** and radius r is in **H** when:

$$\mathbf{ec} \bullet (\mathbf{ed} \times \mathbf{ef}) > r \, \| \, \mathbf{ed} \times \mathbf{ef} \|$$



Is the ball hidden by a triangle?

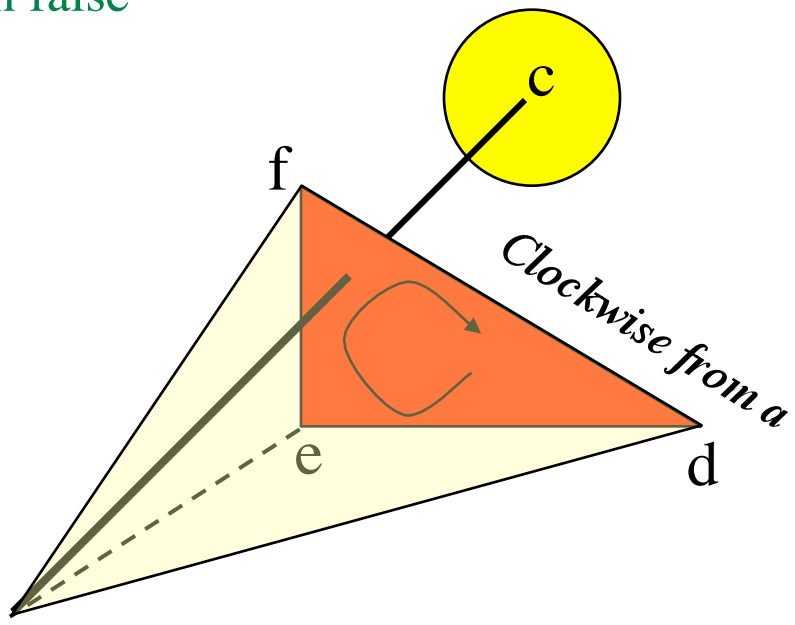
- Is the ball of center **c** and radius r hidden from **a** by triangle with vertices **d**, **e**, and **f**?



Is the ball hidden by a triangle?

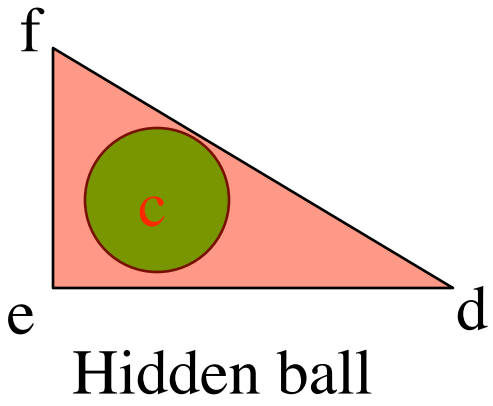
- To test whether the ball of center **c** and radius r hidden from a by triangle with vertices **d**, **e**, and **f** do:
  - if not s(a,d,e,f) then swap d and e
  - if ac • (ad×ae) < r ‖ ad×ae ‖ then return false
  - if ac • (ae×af) < r ‖ ae×af ‖ then return false
  - if ac • (af×ad) < r ‖ af×ad ‖ then return false
  - if ec • (ed×ef) < r ‖ ed×ef ‖ then return false
  - return true

- Explain what each line tests
- Are these test correct?
- Are they sufficient?
- Why?

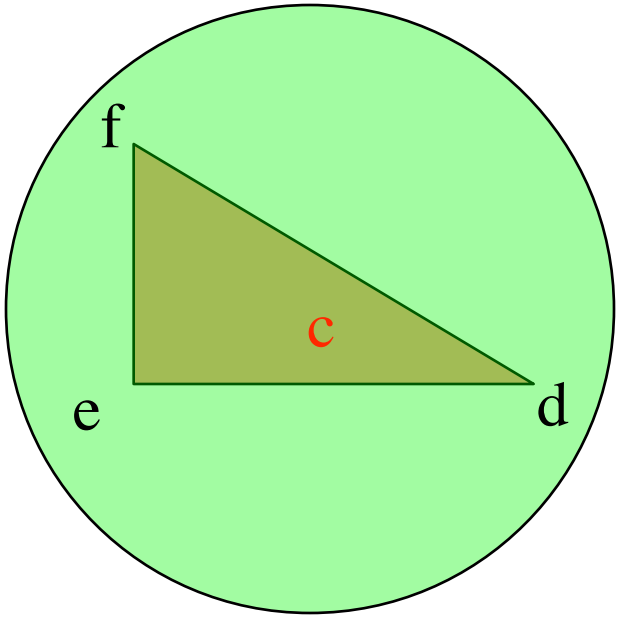


Is the ball **hiding** a triangle?

- Triangle **d**, **e**, **f** is hidden from viewpoint **a** by a ball of center **c** and radius r when its 3 vertices are (hidden).
  - Proof?
  - How to test it?



Hidden ball

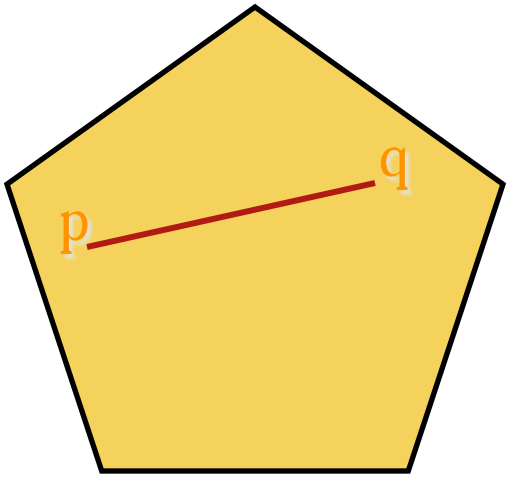


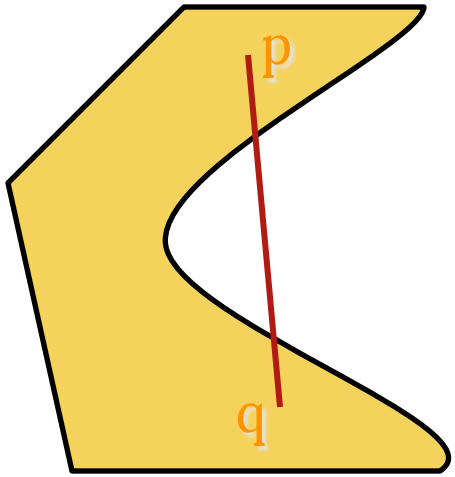
Hidden triangle

When is a planar set A **convex**?

seg(p,q) is the line segment from p to q

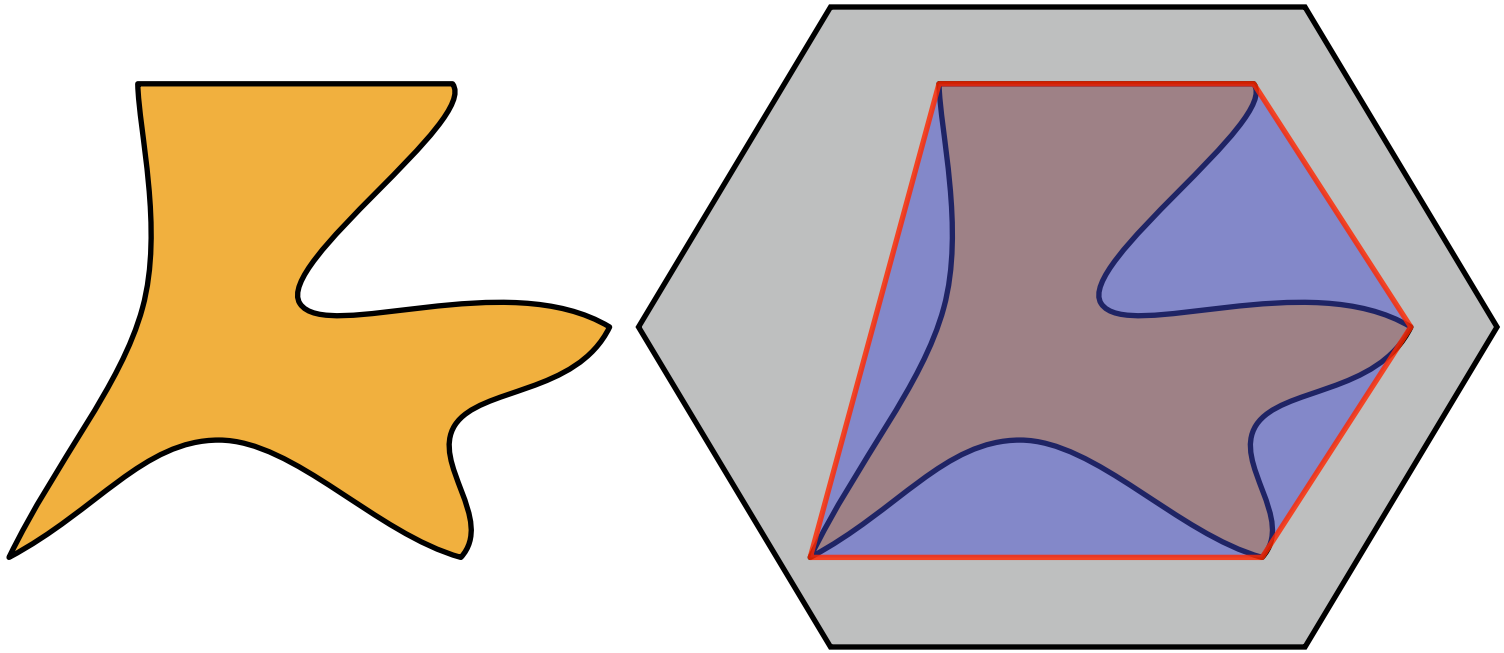
A is convex $\leftrightarrow$ ($\forall p \in A \; \forall q \in A \rightarrow \text{seg}(p,q) \subset A$)





What is the **convex hull** of a set A?

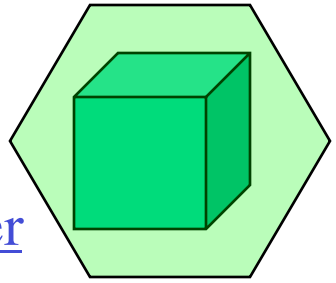
The convex hull H(A) is the intersection of all convex sets that contain A.



Is a **convex** occluder hiding a polyhedron?

- **A polyhedron P is hidden by a convex occluder O if all of its vertices are**
  - Proof**?**
  - Is a **subdivision surface** hidden if its control vertices are?

- **We do not need to test all of the vertices**
  - Only those of a container (minimax box, bounding ball, convex hull)
    - Proof?
  - Only those on the silhouette of a convex container
    - Proof?

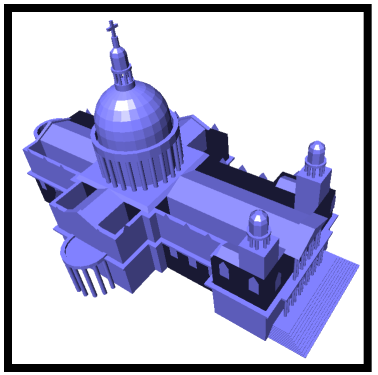


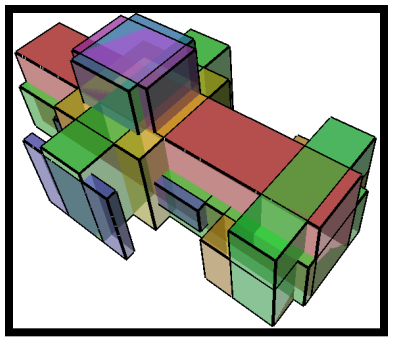
- **What if the occluder is not convex?**
  - Can it still work?

Strategies for non-convex occluders?

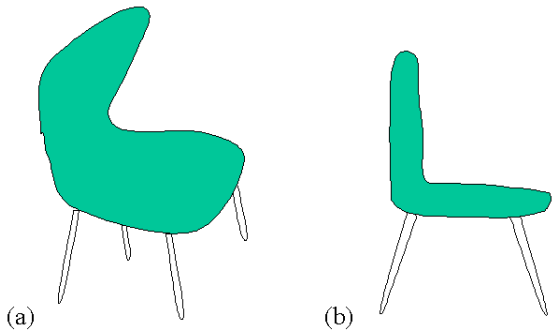
What if O is not convex?

- replace O by an interior ball
  - Not easy to compute a (nearly) largest ball inside O
  - The average of the vertices of O may be outside of O
- replace O by an interior axis-aligned box
  - Not easy to compute a largest box inside
- replace O by a convex set it contains
  - O may be too thin





Andujar, Saona Navazo
CAD, 32 (13), 2000

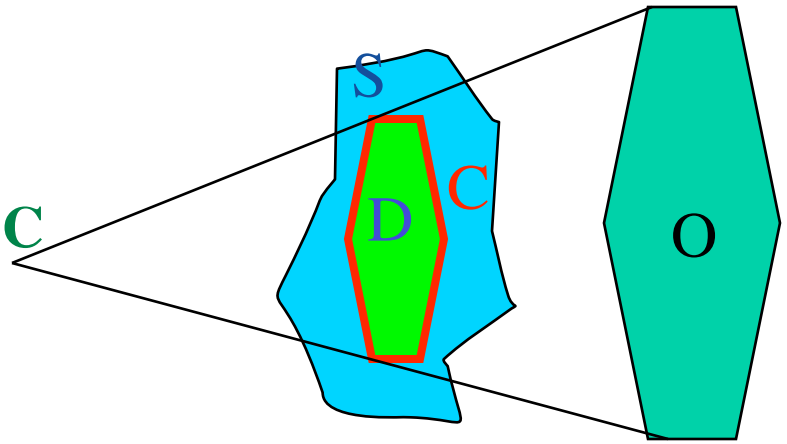


The **Hoop** principle

The occluder does not have to be convex, provided that a portion of its surface appears convex to the viewer and that the object is behind it.

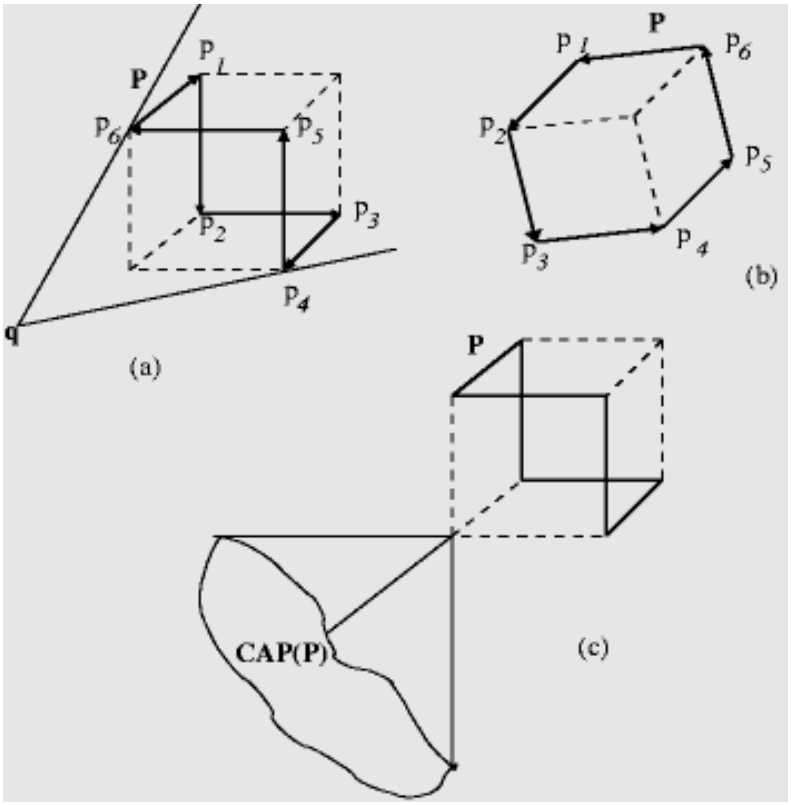
Hoop from non-convex occluders

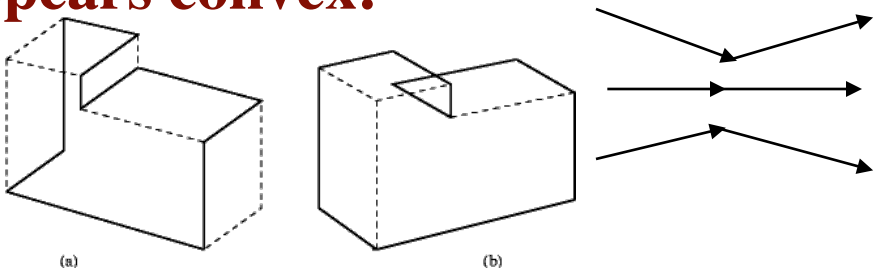
- **Pick a candidate occluder S**
  - **large triangle mesh**
  - **close to the viewer**
- **Select a set D of triangles of S whose border C is a hoop:**
  - **single loop**
  - **that has a convex projection on the screen**
- **Build a shield O**
  - **on plane** behind **D**
  - **bounded by projection of C**
- **O is a convex occluder**
  - **anything hidden by O**
  - **is hidden by D**
  - **and hence by S**
  - **sure?**



When does a 3D polyloop appear convex?

Each pair of consecutive edges defines a linear half-space from where their common vertex appears convex.

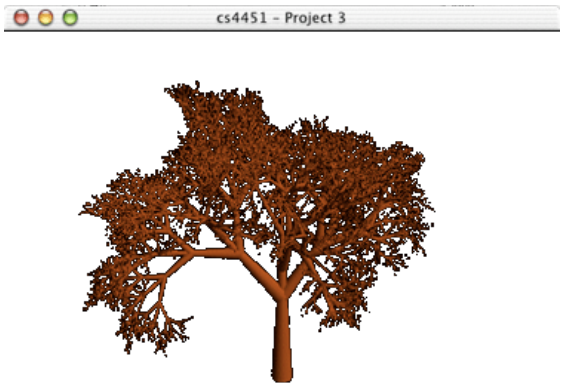




The portion of space from which a loop C of edges appears convex is the intersection of these half-spaces or of their complements.

Is the moon hidden by a tree?

- Can't grow a patch of edge-connected triangles
  - Leaves/branches are small and disjoint
- Use approximate shadow fusion
  - Render tree(s)
  - Mark area of rendered pixels
  - Construct big rectangle inside it (how?)
  - Let M= max(z-buffer inside rectangle)
  - Push back rectangle to z=M
  - Use it as occluder for other objects (behind the tree)
- Accuracy depends on screen resolution
  - Can be improved by multiple passes (zoomed 1/4 of image each)
- Use bounding spheres around objects to test them
  - Conservative part
  - Use a hierarchy of bounding sphere (hierarchical z-buffer)



Examples of questions for exams

- When is it no-longer useful to simplify the model?
- What are the 7 acceleration techniques?
- Which ones reduce per-vertex cost?
- How much savings should you expect from each technique?