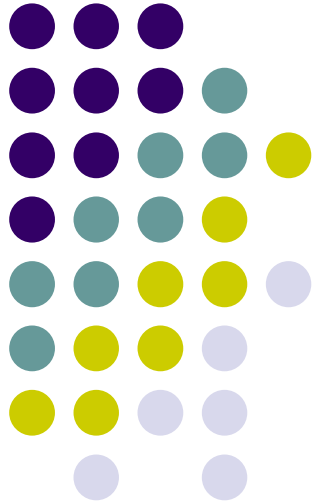
Computer Graphics (CS 4731)
Clipping and Culling

Joshua Cuneo

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

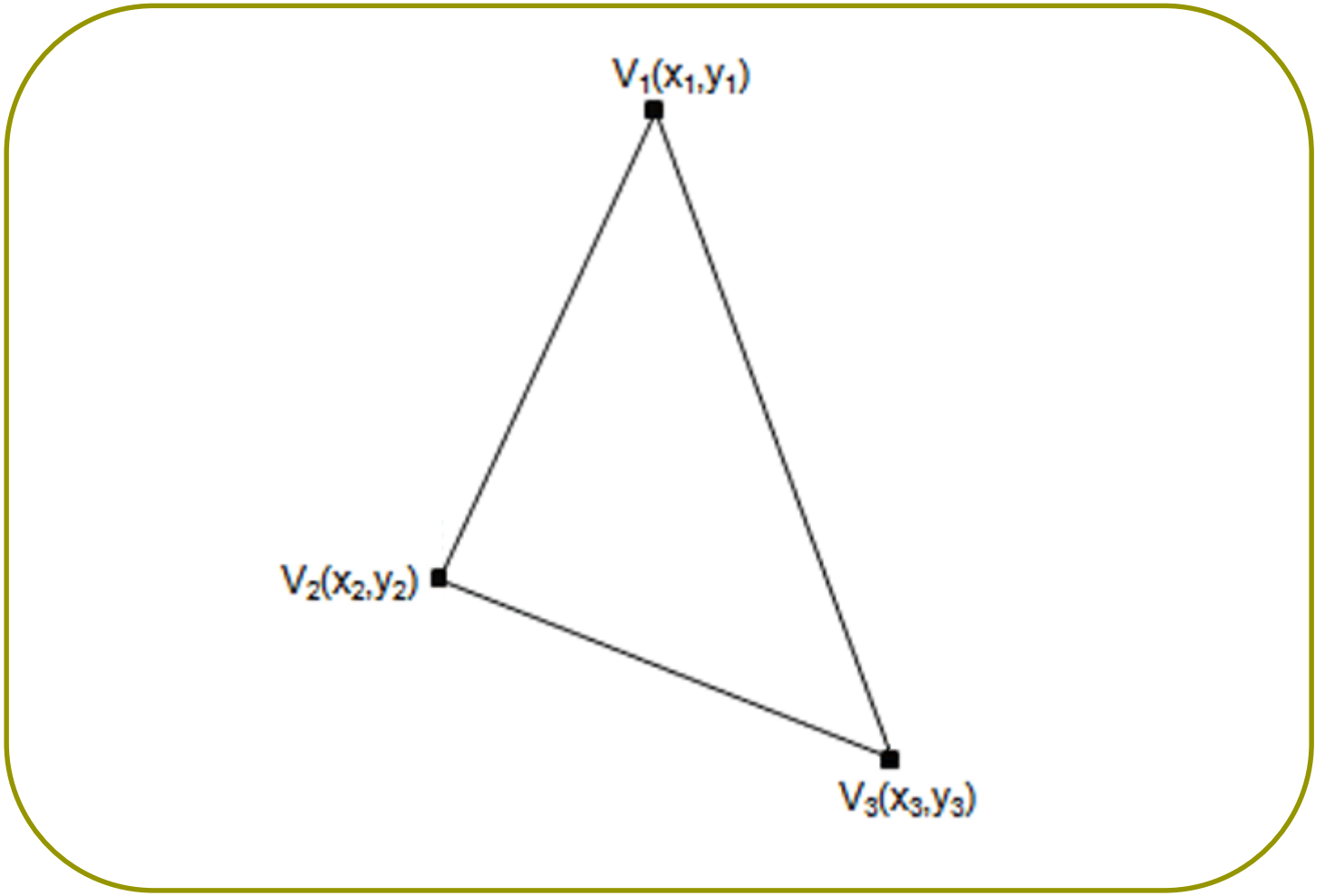


Slides Courtesy of Emmanuel Agu (WPI) and Matt Stoker (Utah)

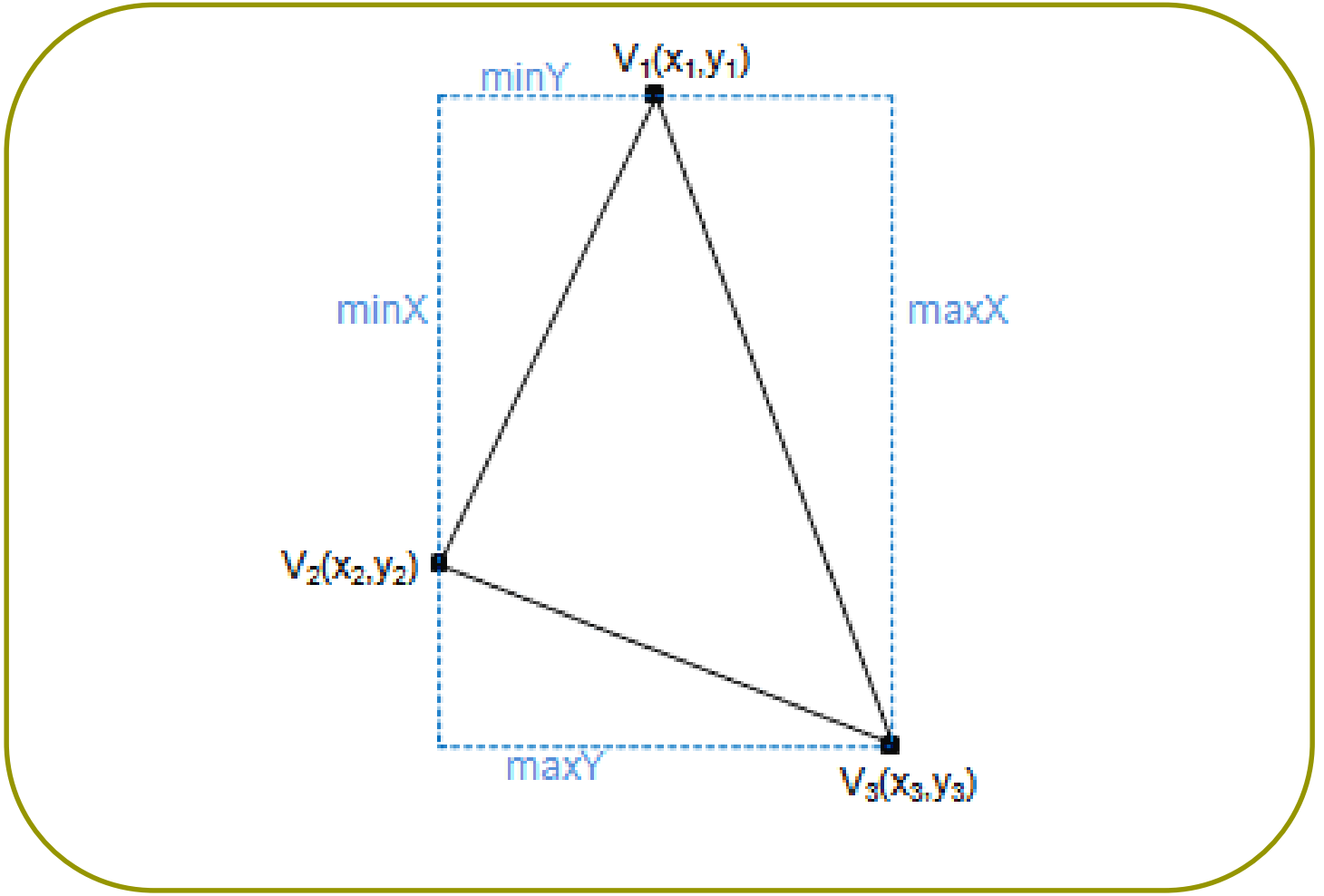
Think-Pair-Share

What's the problem and how can we improve it?

Screen



Bounding Boxes!

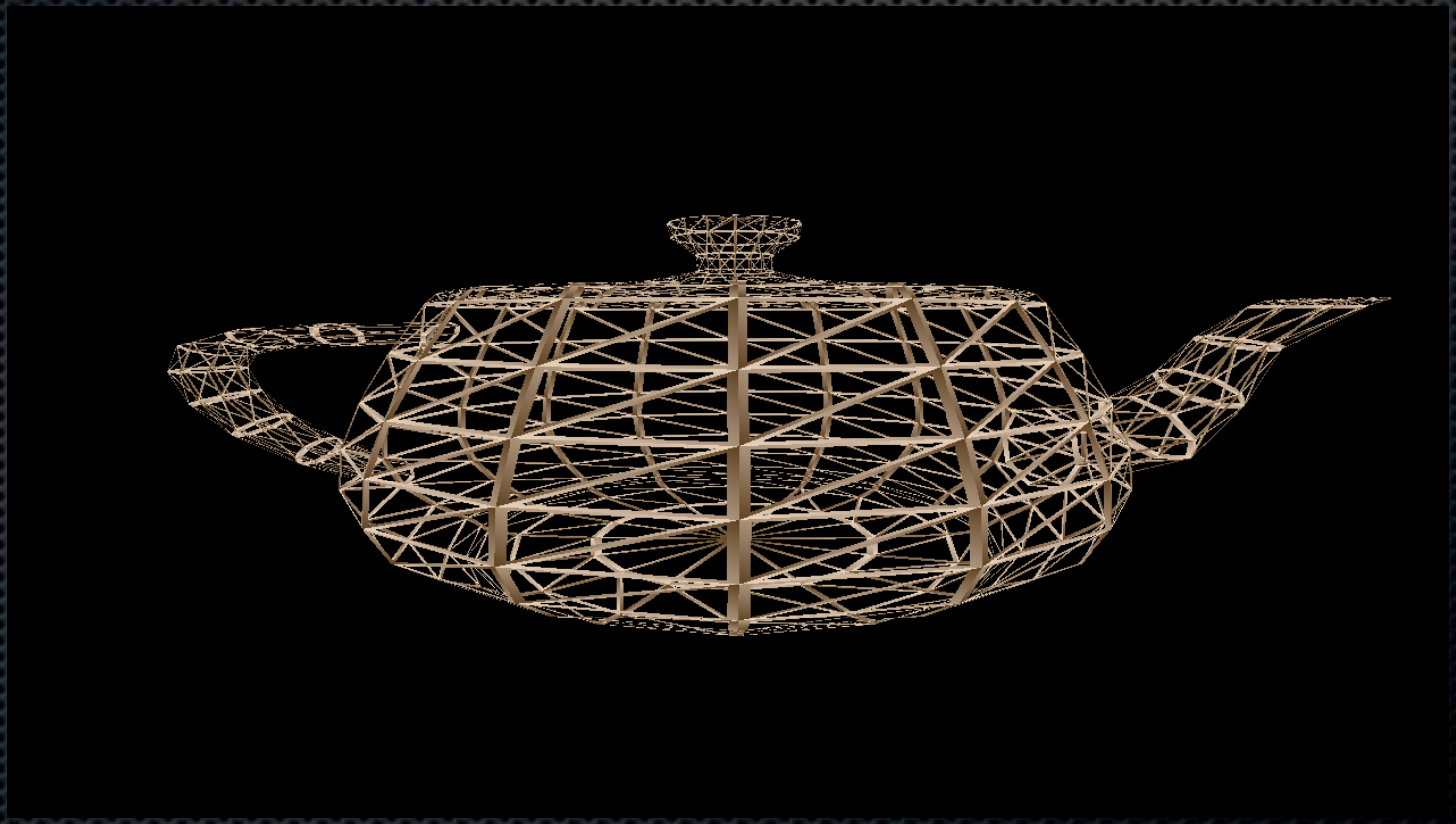


Cull (verb)

to reduce or control the size of by removal

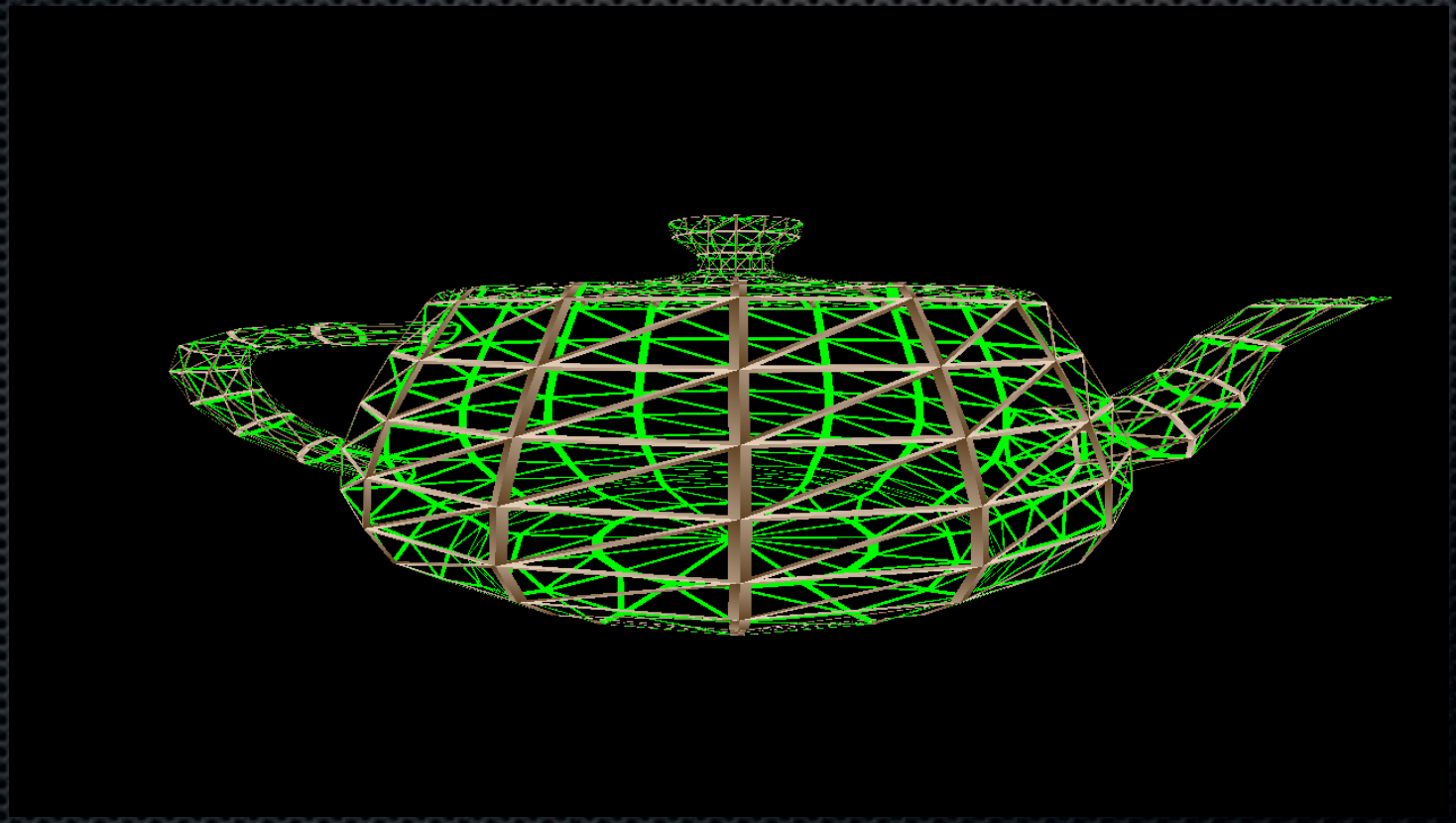
Cull (verb)

to reduce or control the size of by removal



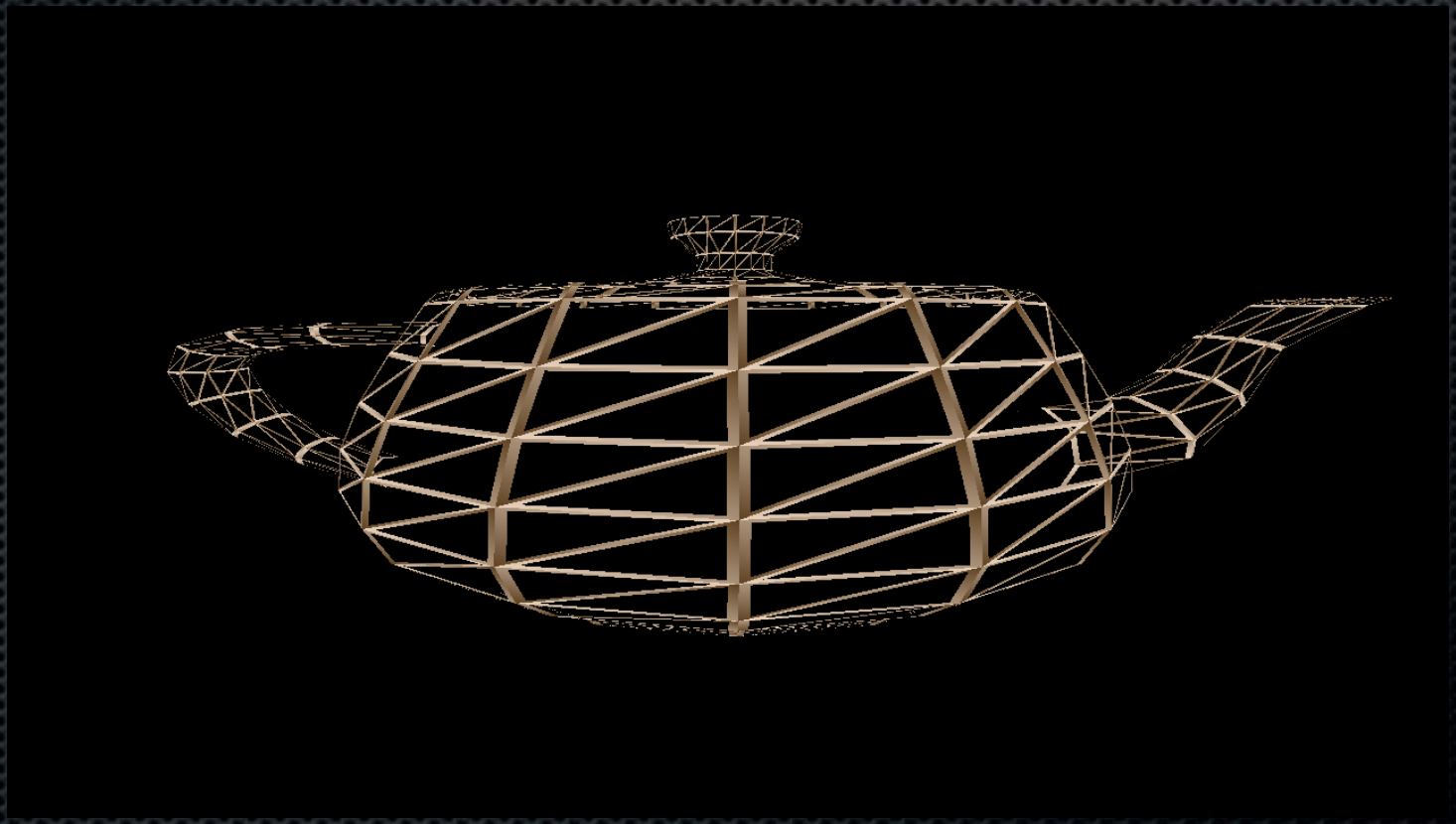
Cull (verb)

to reduce or control the size of by removal



Back-Face Culling

Drew 37% (382 of 1024) of triangles in scene

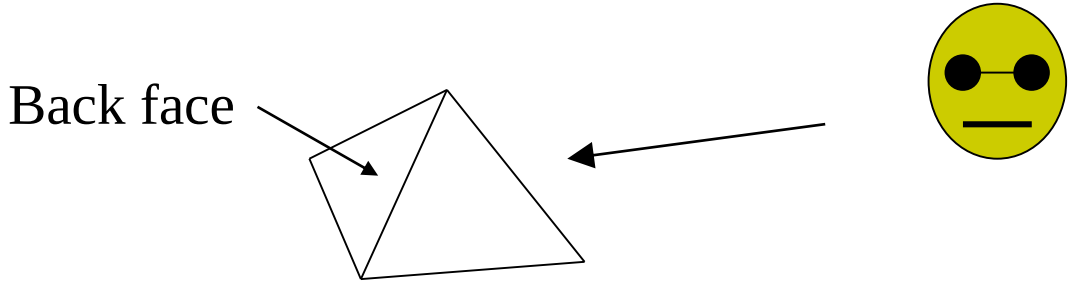


Back-Face Culling

Drew 37% (382 of 1024) of triangles in scene

Back Face Culling

- **Back faces:** faces of opaque object that are “pointing away” from viewer
- **Back face culling:** do not draw back faces (saves resources)

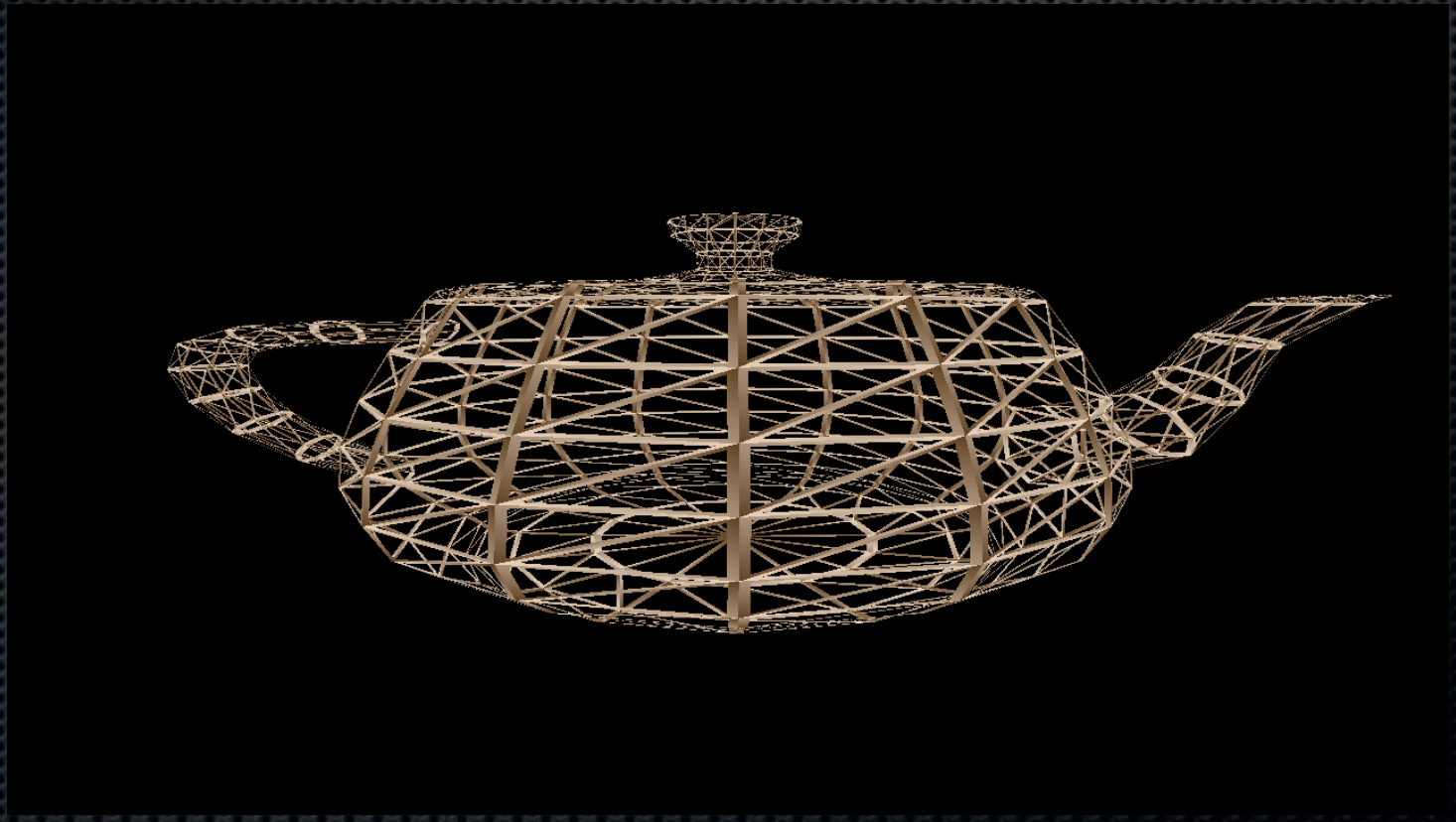


Back-Face Culling

Which triangles are back-facing?

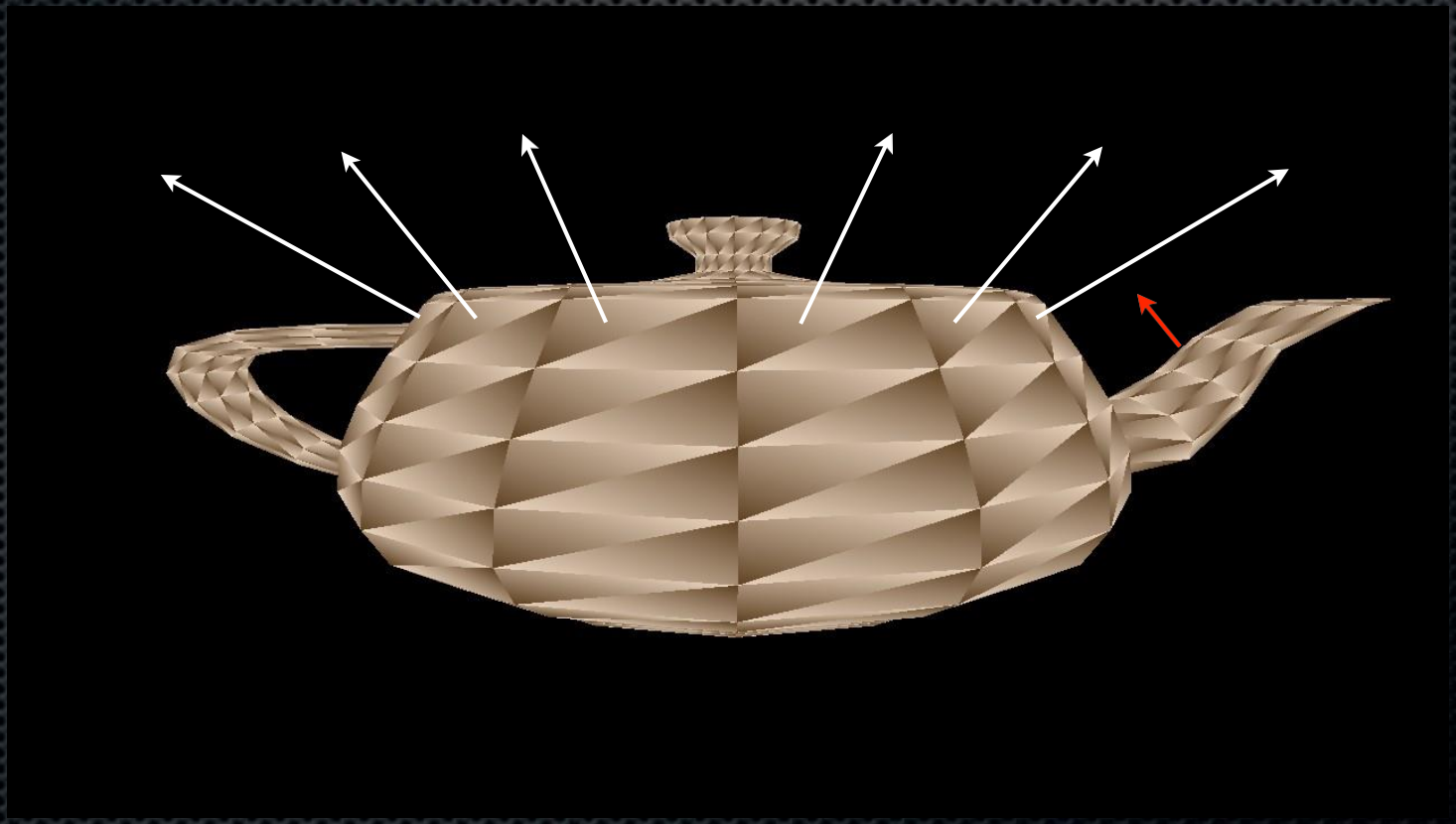
Think-Pair-Share

How can we determine which triangles are back-facing?



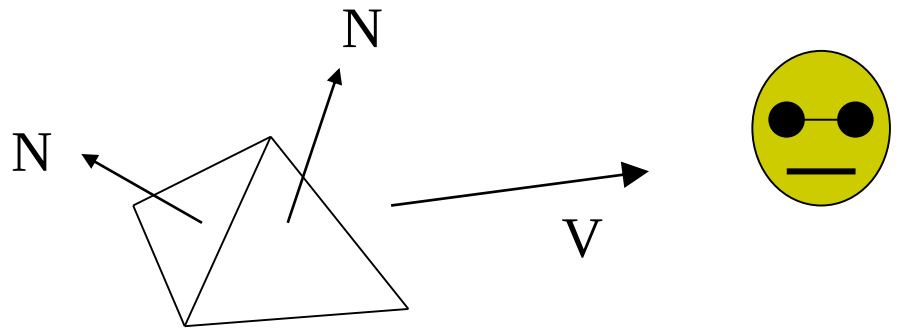
Back-Face Culling

Triangle normal of back-facing triangles points “away”



Back Face Culling

- Goal: Test if a face F is is backface
- How? Form vectors
  - View vector, V
  - Normal N to face F



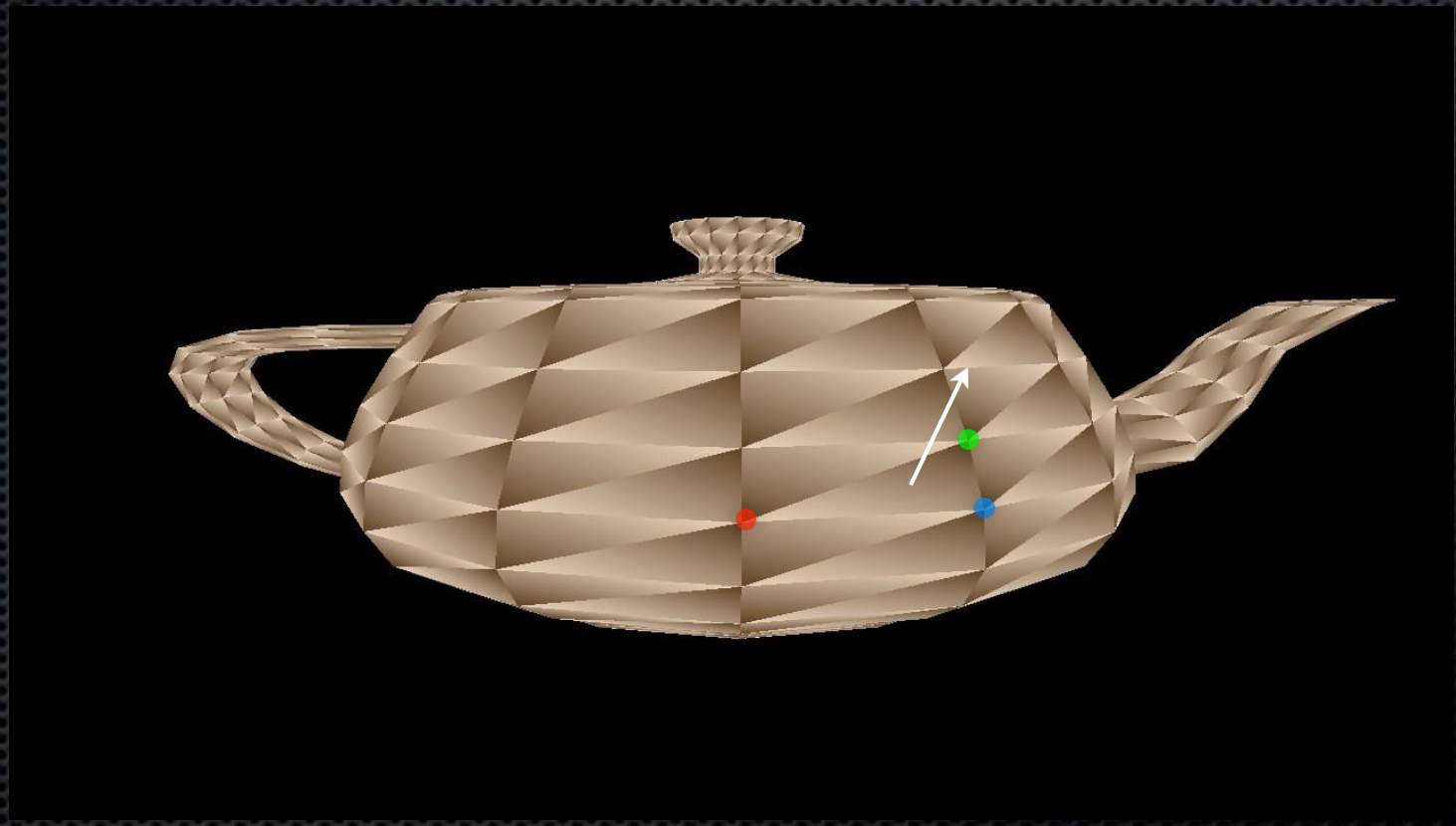
Backface test: F is backface if N.V < 0 why??

Back-Face Culling

What is the triangle's normal?

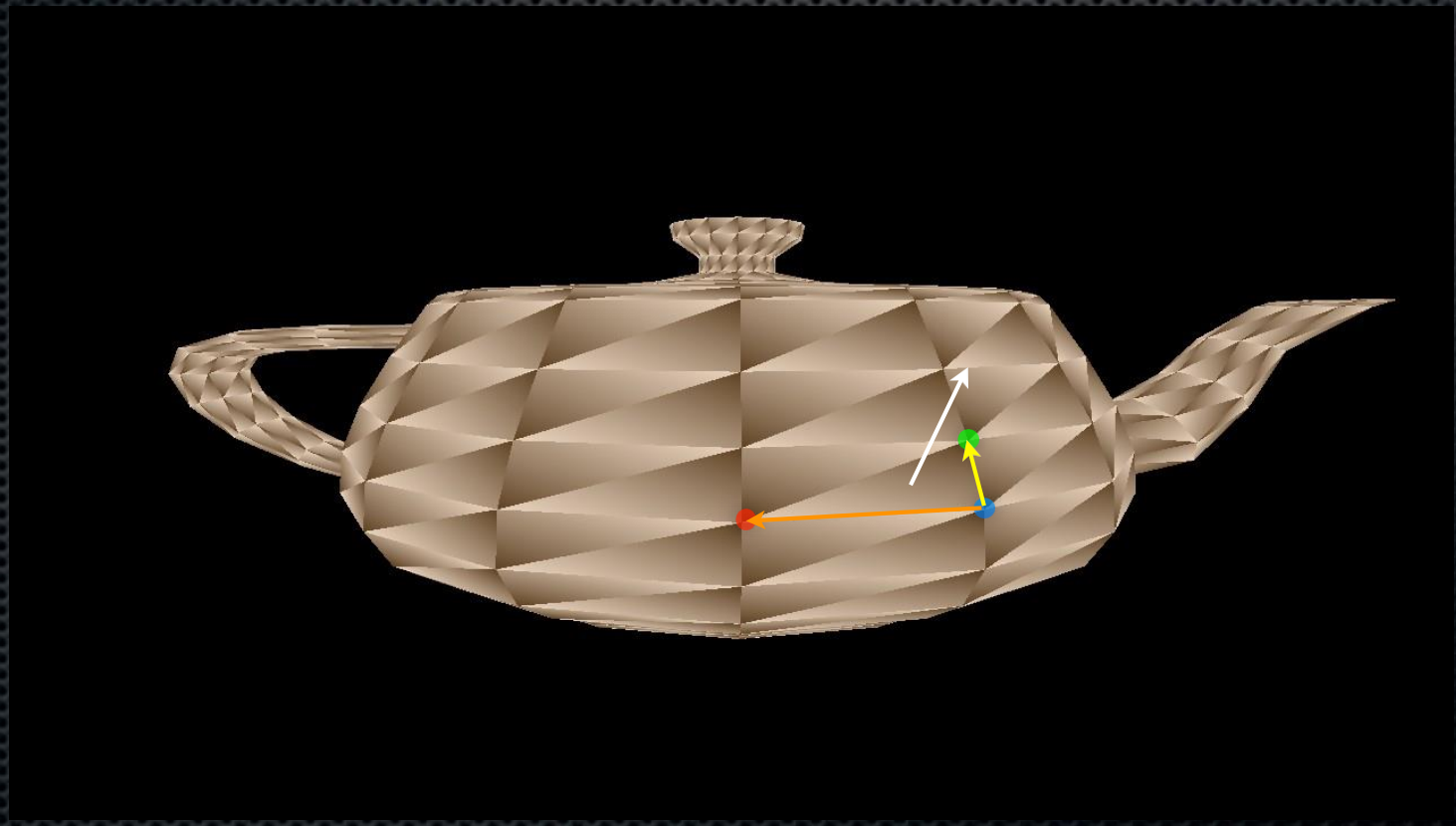
Back-Face Culling

A B C



Back-Face Culling

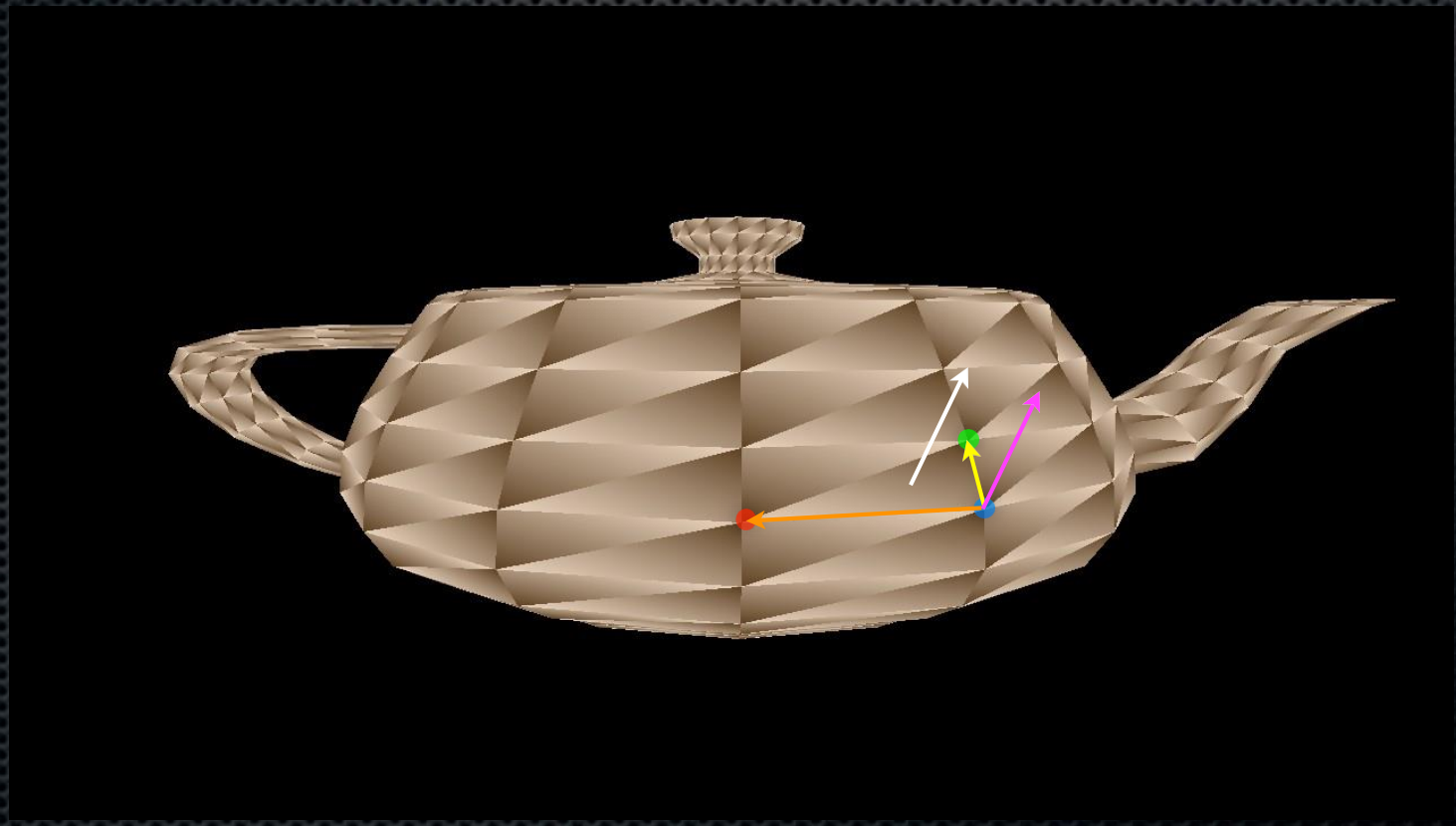
$v_0 = B - A \quad v_1 = C - A$



Back-Face Culling

$v_0 = B - A \quad v_1 = C - A$

$n = v_0 \times v_1$

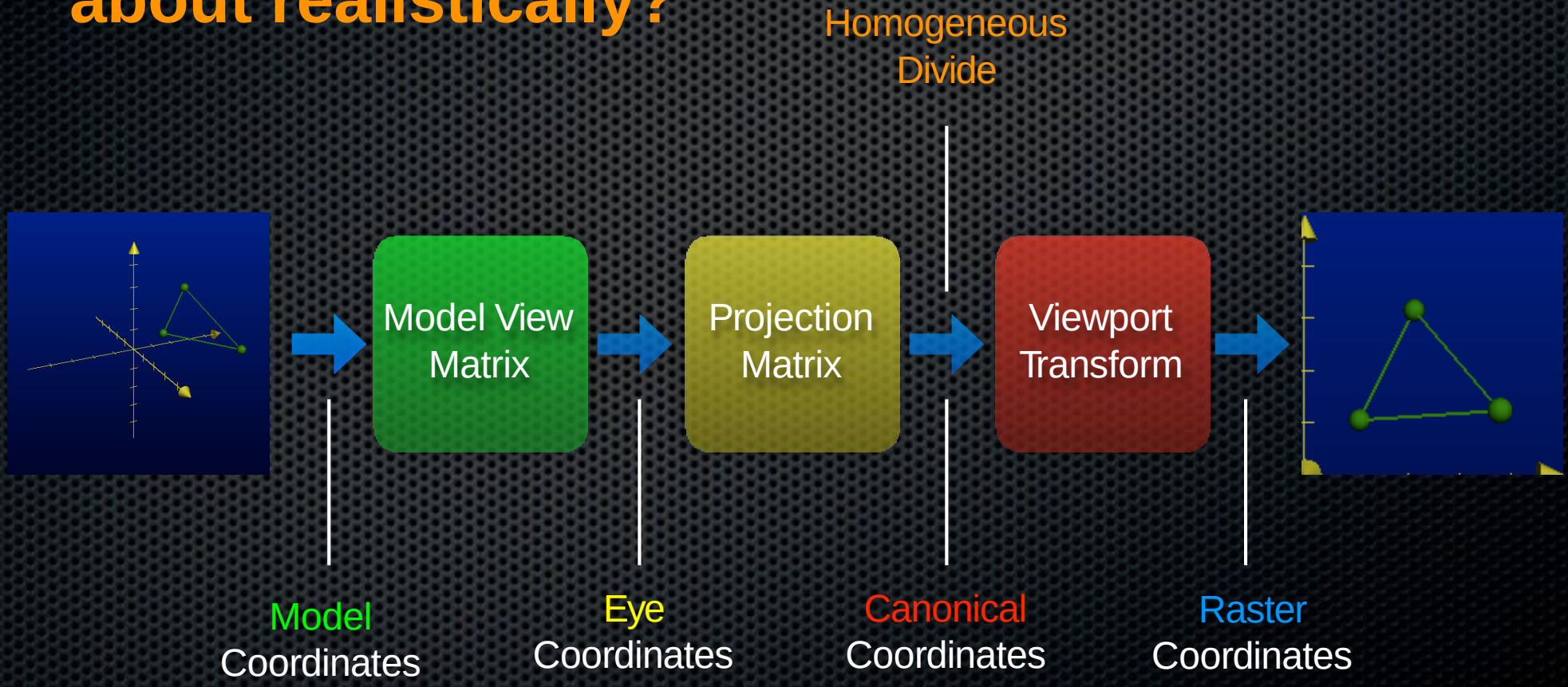


Back-Face Culling

- When to back-face cull?

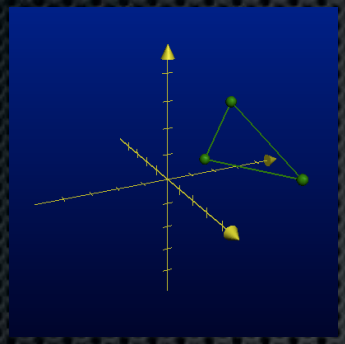
Think-Pair-Share

Ideally, where is the best place to cull? What about realistically?



Back-Face Culling

if (n.z < 0)
cull

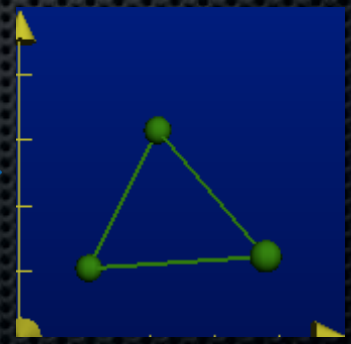


Model View Matrix

Projection Matrix

Homogeneous Divide

Viewport Transform



Model Coordinates

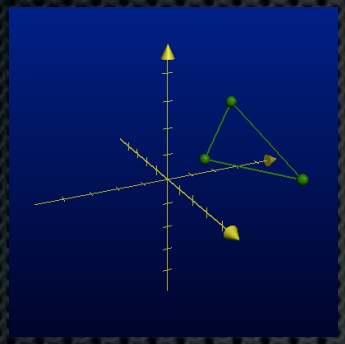
Eye Coordinates

Canonical Coordinates

Raster Coordinates

Back-Face Culling

if (n.z < 0)
cull

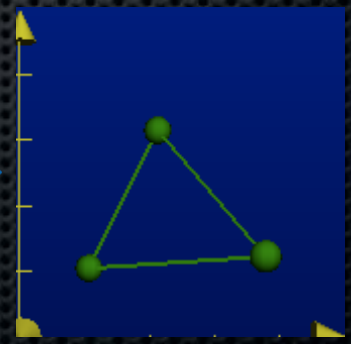


Model View Matrix

Projection Matrix

Homogeneous Divide

Viewport Transform



Model Coordinates

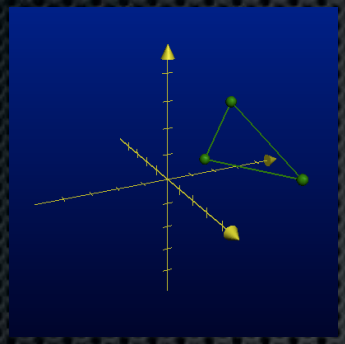
Eye Coordinates

Canonical Coordinates

Raster Coordinates

Back-Face Culling

if ($n \bullet (t_{center} - origin) > 0$)
cull

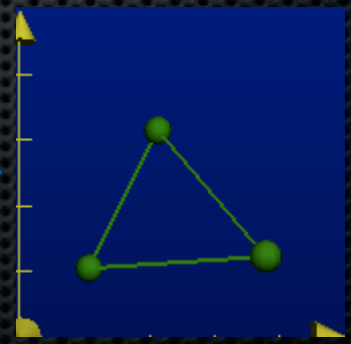


Model View Matrix

Projection Matrix

Homogeneous Divide

Viewport Transform



Model Coordinates

Eye Coordinates

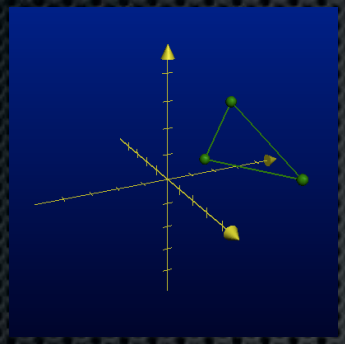
Canonical Coordinates

Raster Coordinates

Back-Face Culling

transform eye to model coordinates

if $(n \bullet (t_{center} - eye) > 0)$
  cull

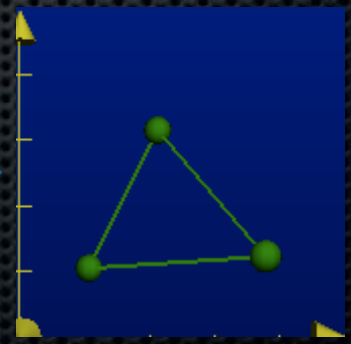


Model View Matrix

Projection Matrix

Homogeneous Divide

Viewport Transform



Model Coordinates

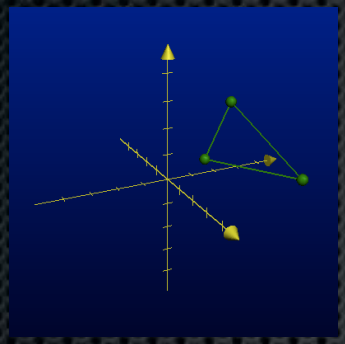
Eye Coordinates

Canonical Coordinates

Raster Coordinates

Back-Face Culling

if ($n \bullet (t_{center} - origin) > 0$)
cull

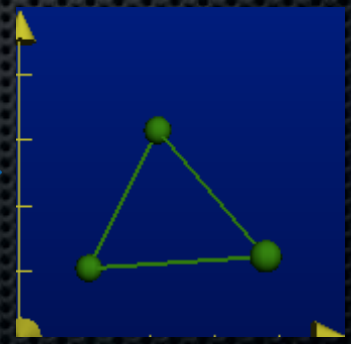


Model View Matrix

Projection Matrix

Homogeneous Divide

Viewport Transform



Model Coordinates

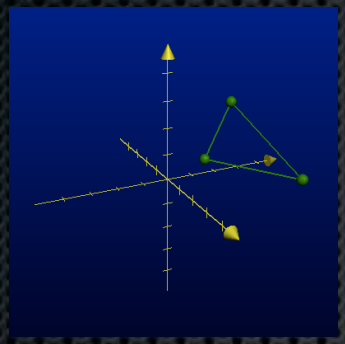
Eye Coordinates

Canonical Coordinates

Raster Coordinates

Back-Face Culling

if ($n \bullet t_{center} > 0$)
cull

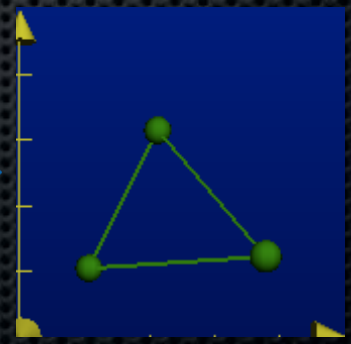


Model View Matrix

Projection Matrix

Homogeneous Divide

Viewport Transform



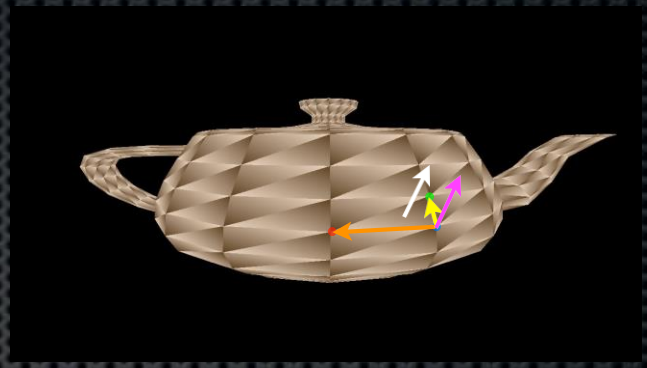
Model Coordinates

Eye Coordinates

Canonical Coordinates

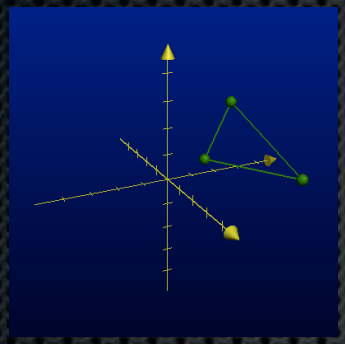
Raster Coordinates

Back-Face Culling



if ($n \bullet t_a > 0$)
cull

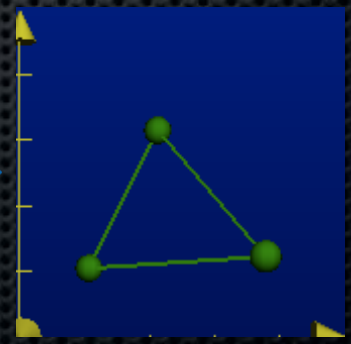
Homogeneous Divide



Model View Matrix

Projection Matrix

Viewport Transform



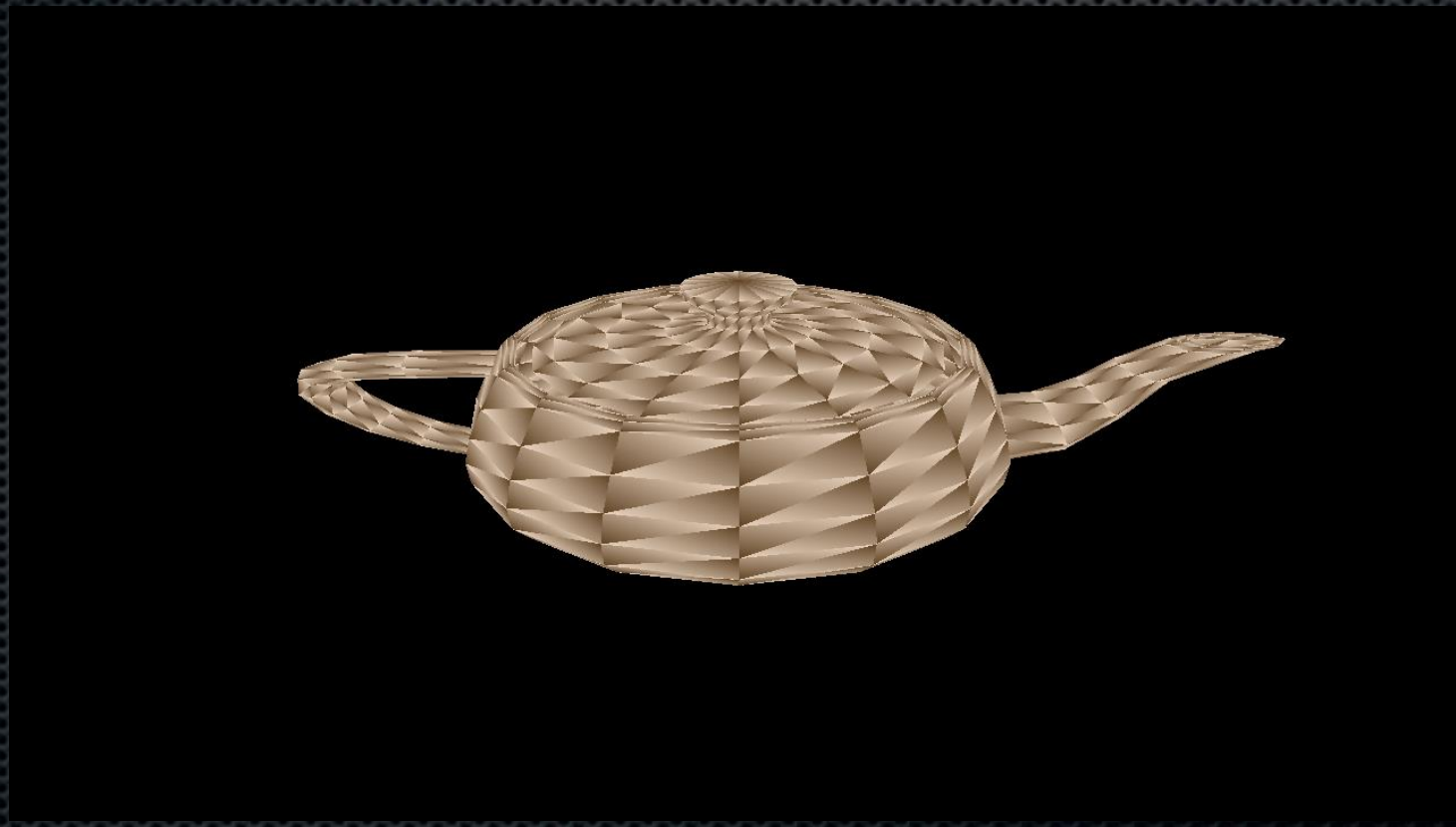
Model Coordinates

Eye Coordinates

Canonical Coordinates

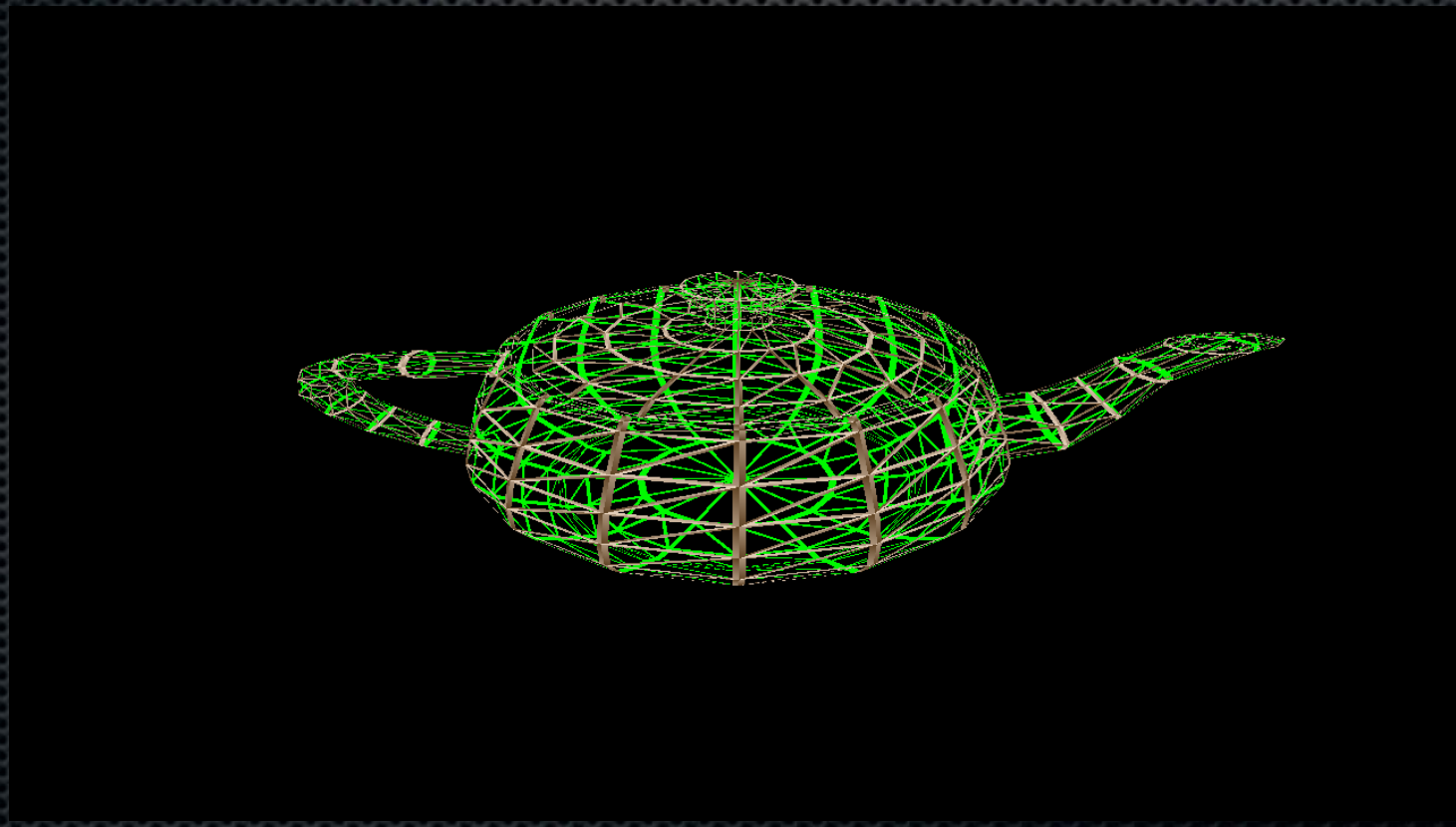
Raster Coordinates

Back-Face Culling Issues



Back-Face Culling Issues

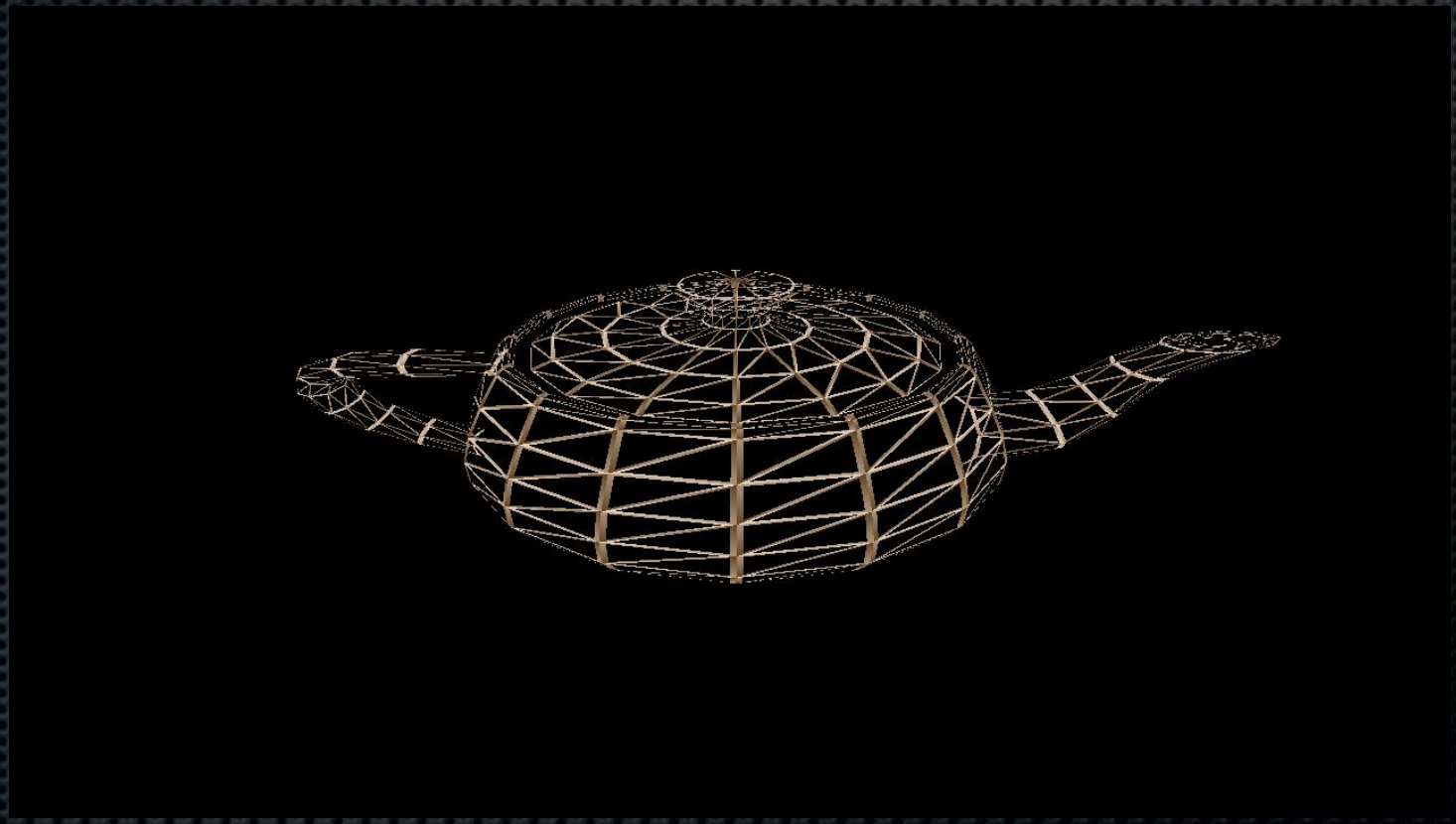
Back-Face Culling Issues



Back-Face Culling Issues

Think-Pair-Share

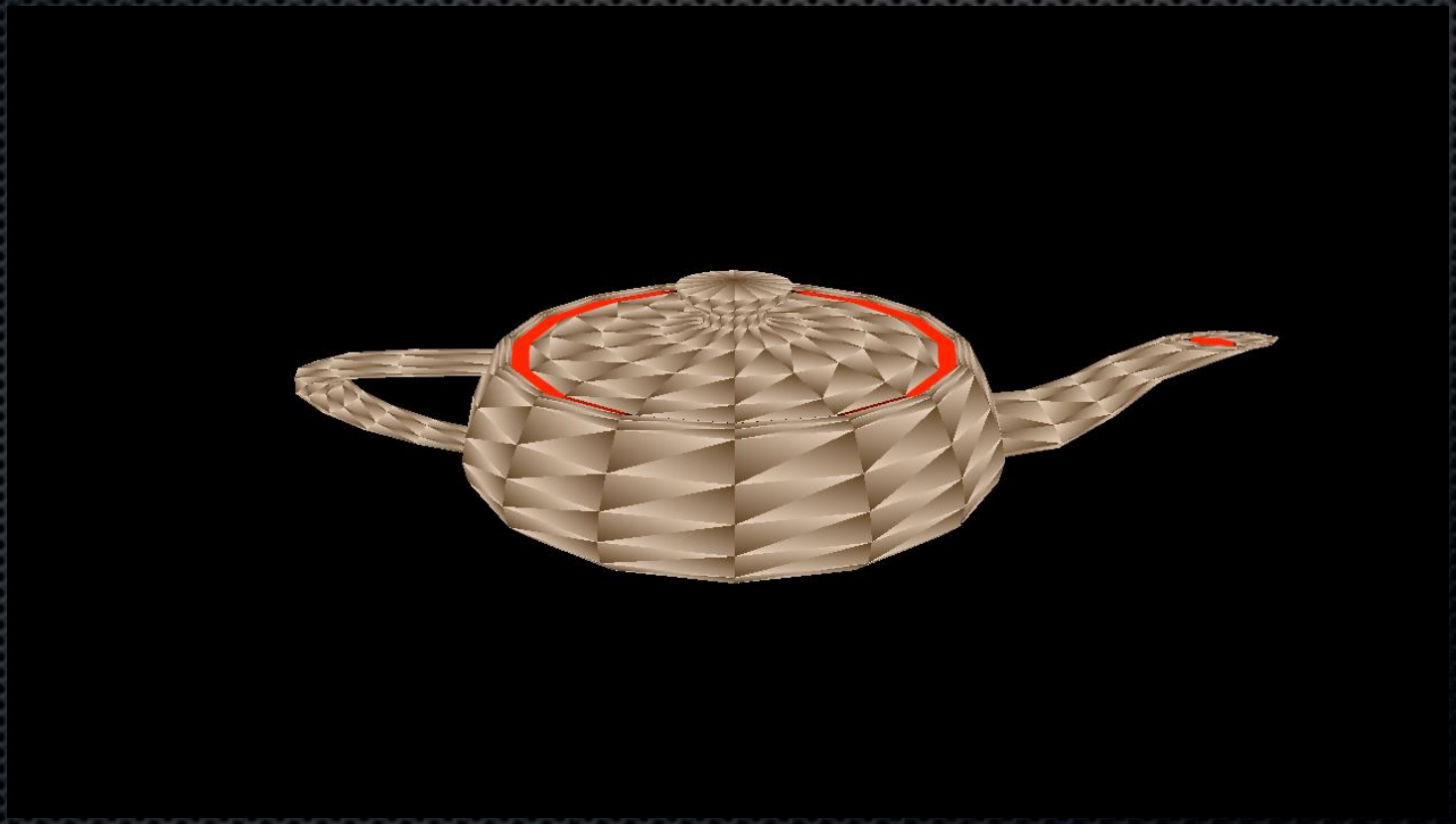
What's the problem here?



Back-Face Culling Issues

Back-Face Culling Issues

Objects drawn with back-face culling need to be solid



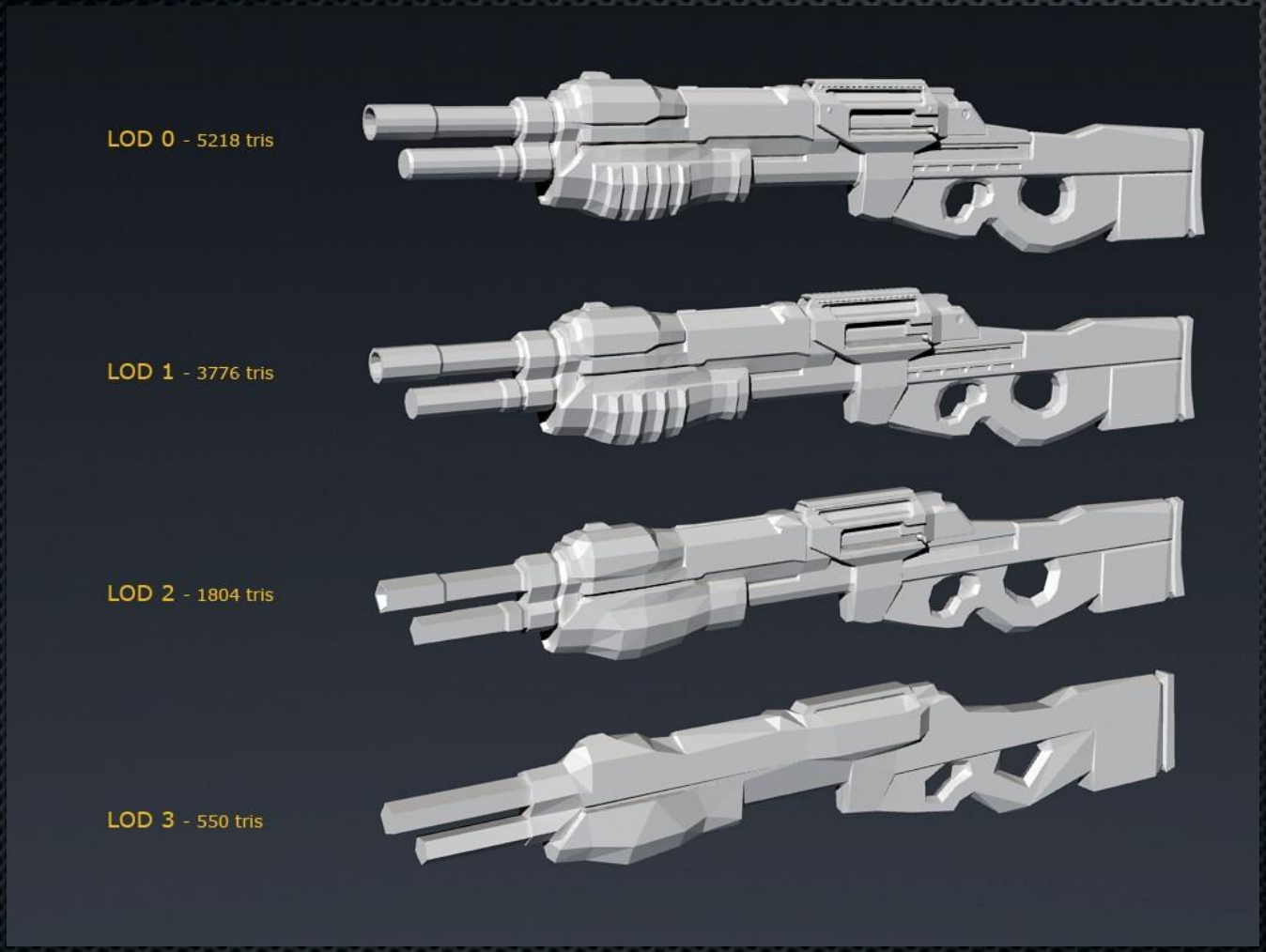
Other Culling Opportunities

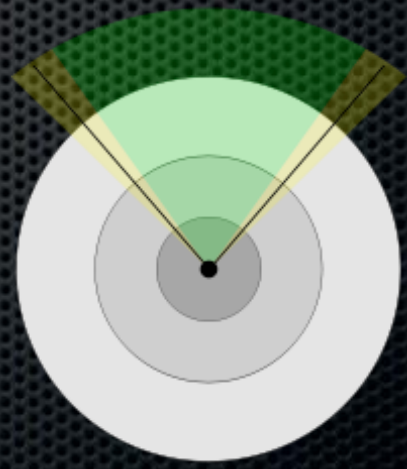
Other Culling Opportunities

Other Culling Opportunities

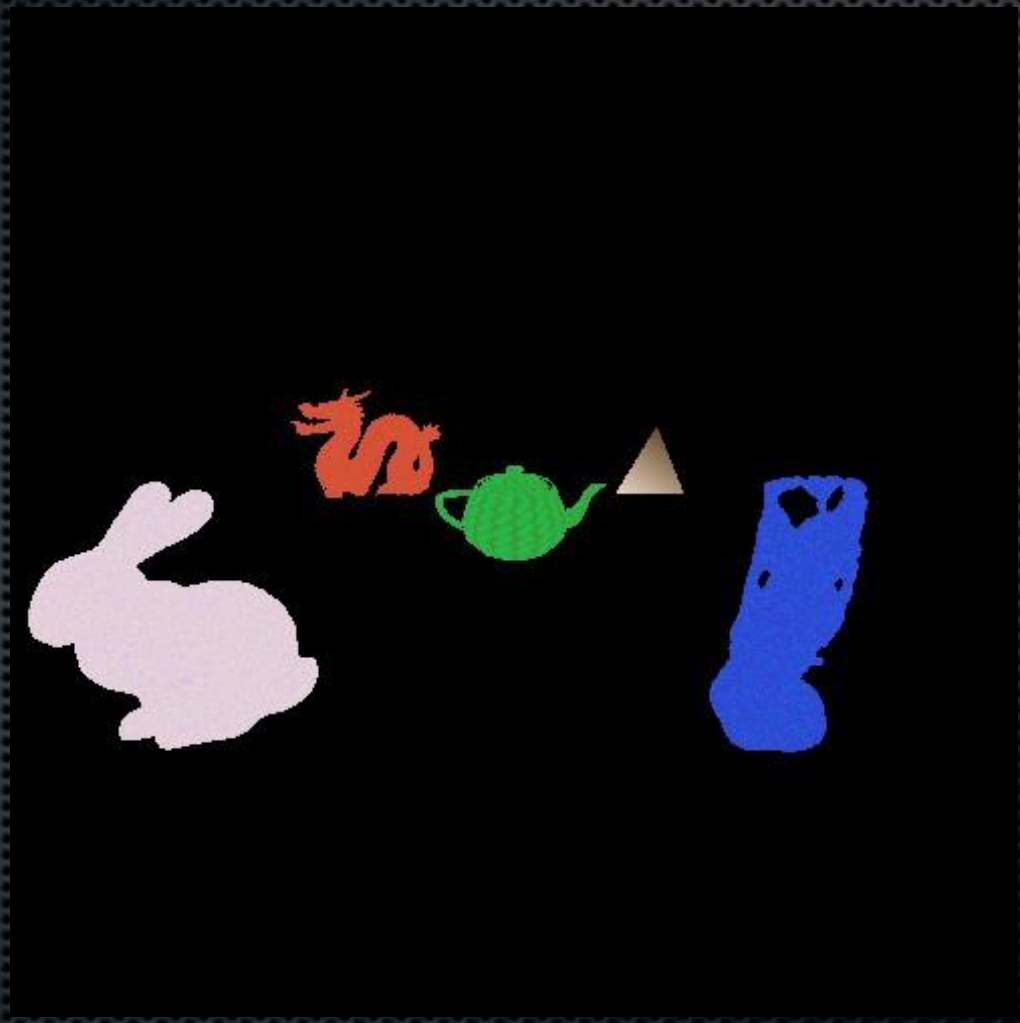
Other Culling Opportunities

Level of Detail (LOD)



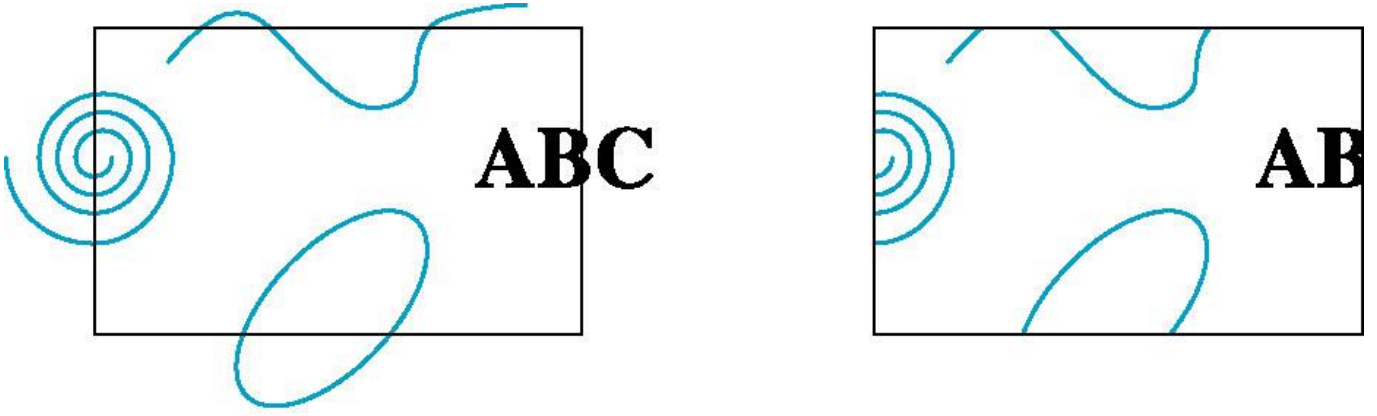


Clipping



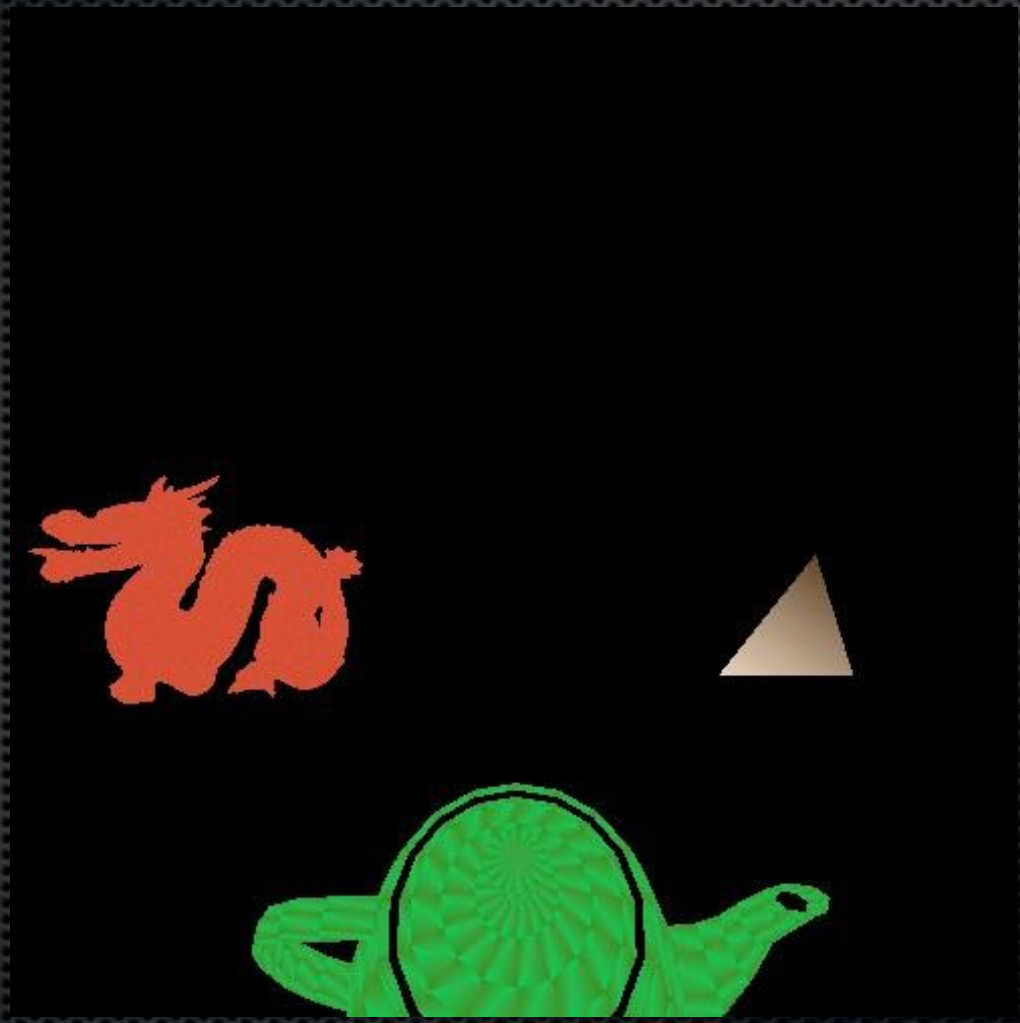
Clipping

- **Clipping:** Remove primitives (lines, polygons, text, curves) outside view frustum (canonical view volume)



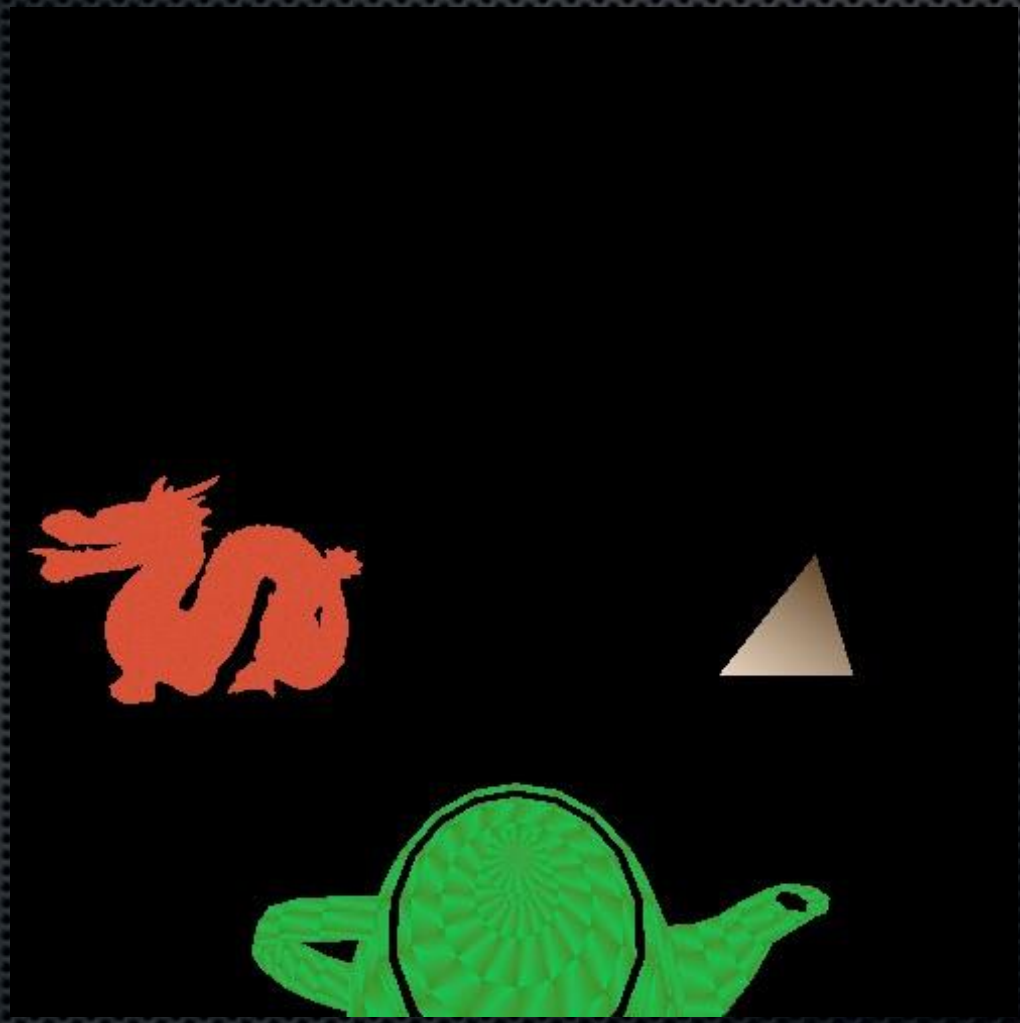
Clipping

Clipping



Clipping

Need to draw bunny?

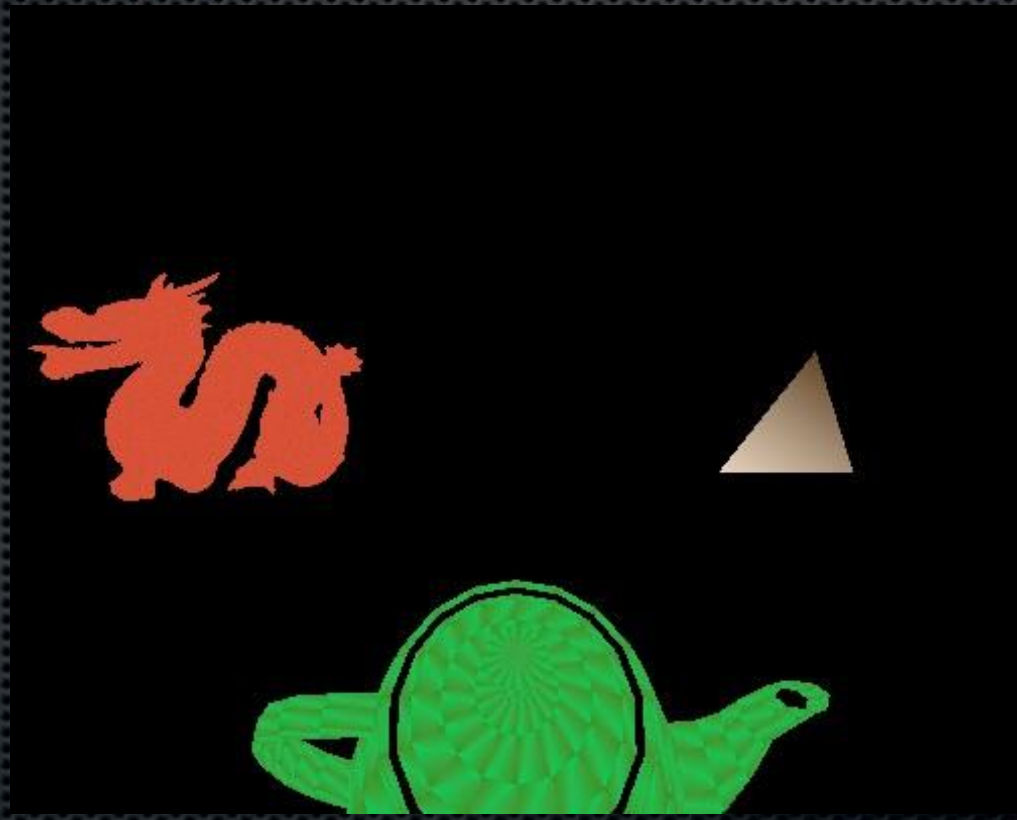


Need to draw budda?

Think-Pair-Share

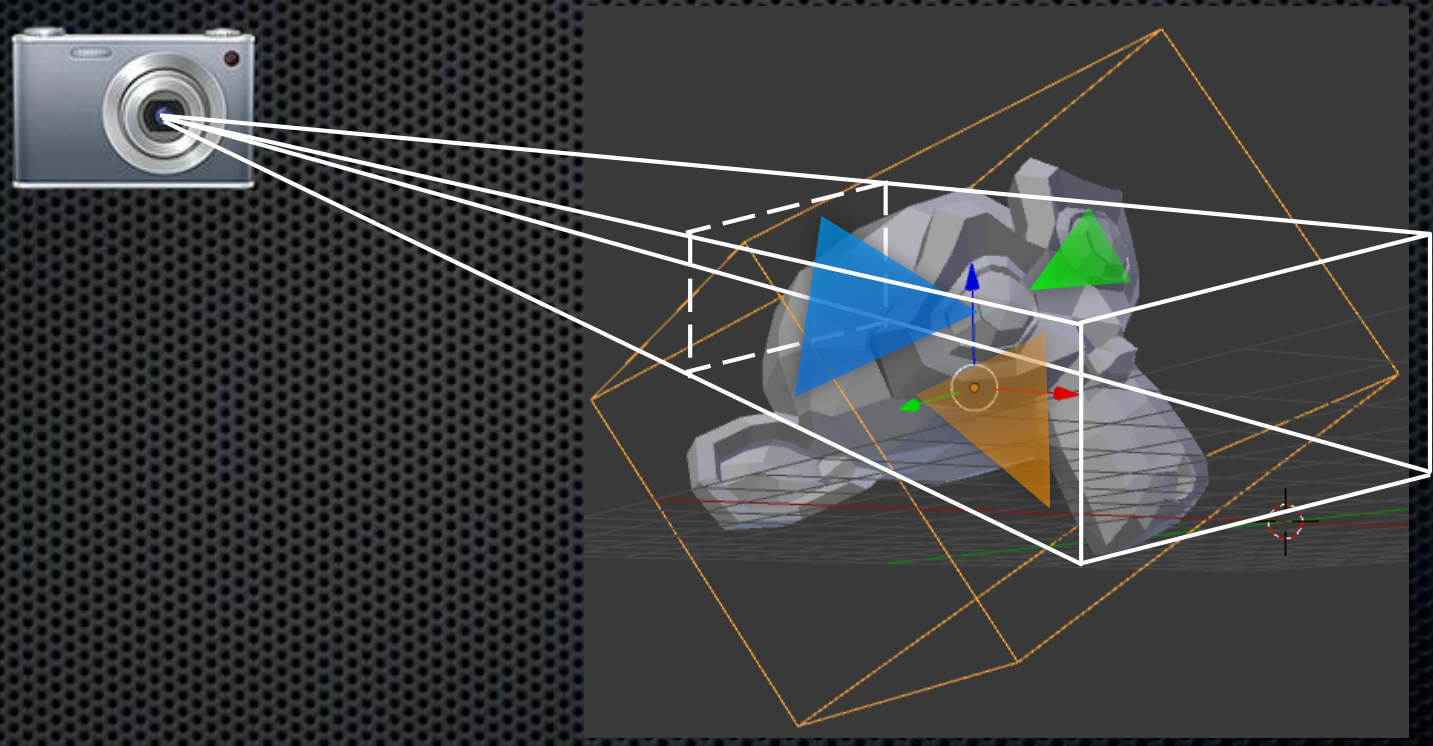
How do we determine if an object is off-screen?

Need to draw bunny?

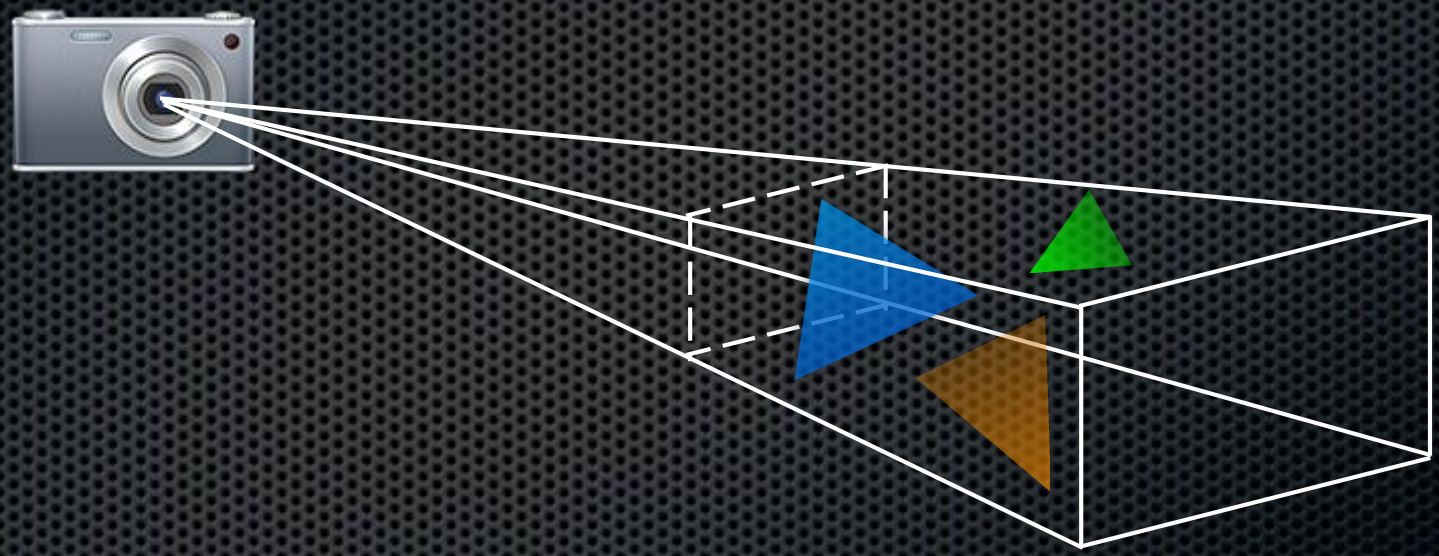


Need to draw budda?

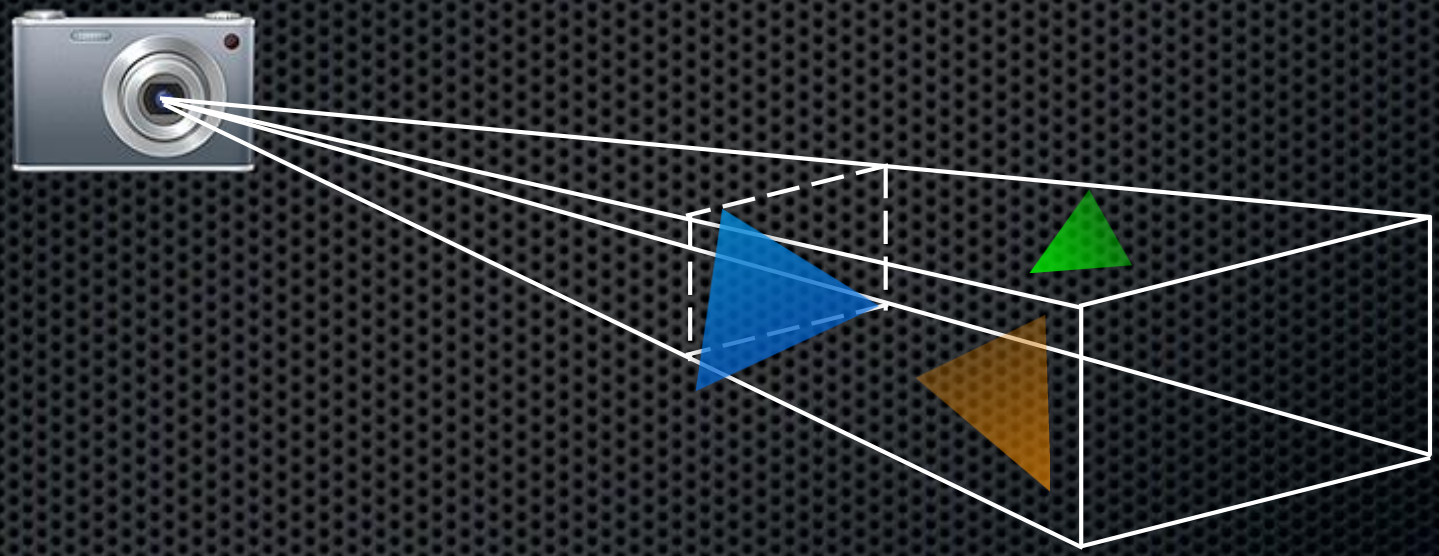
Clipping



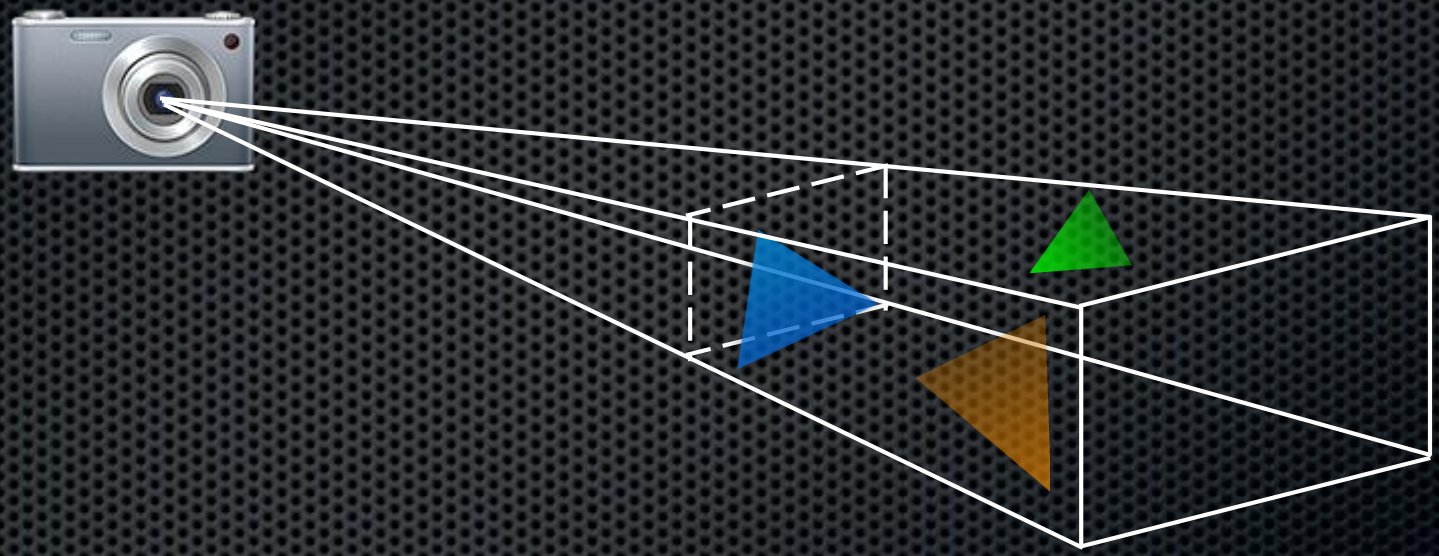
Clipping



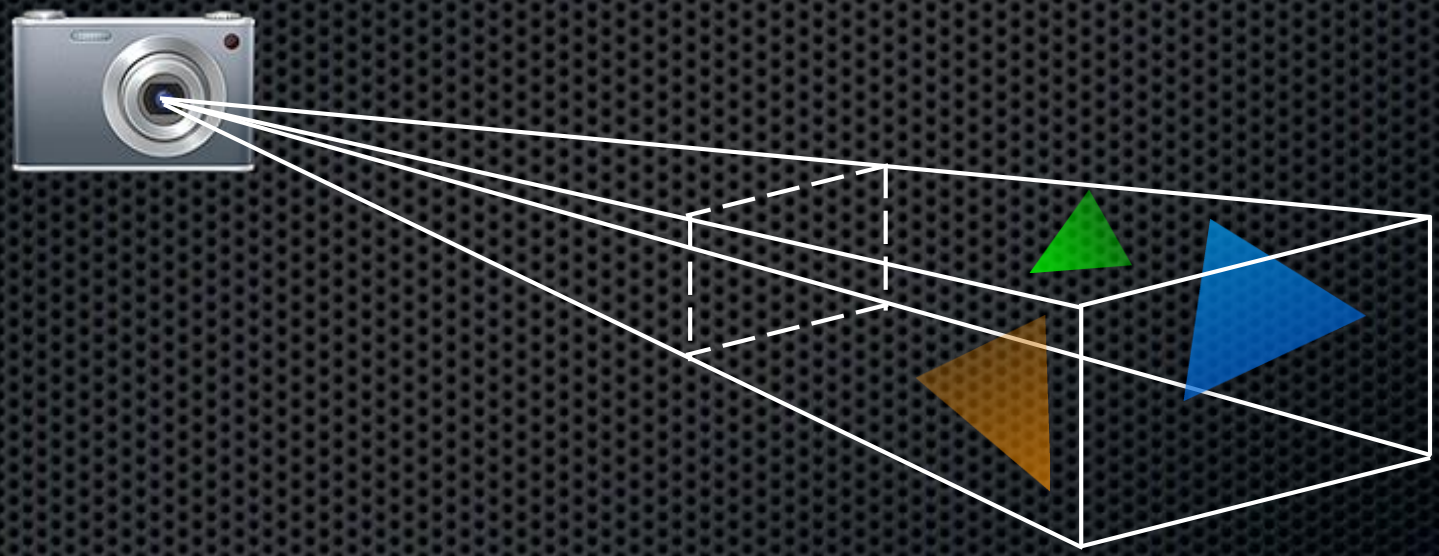
Clipping



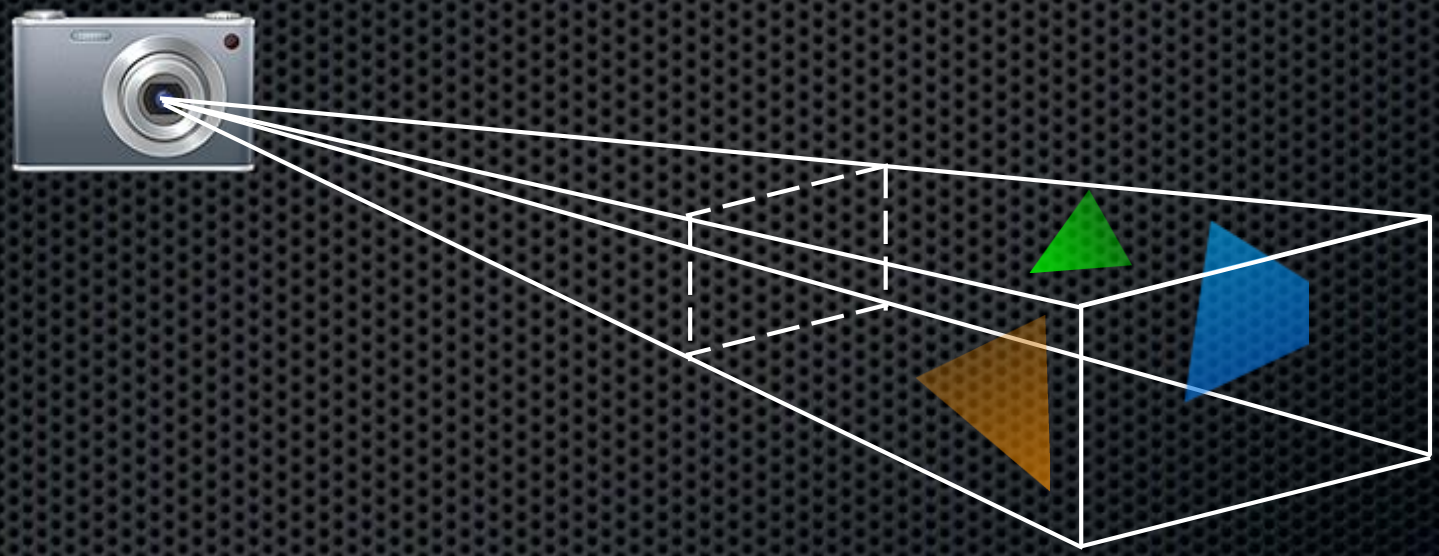
Clipping



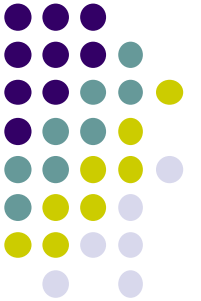
Clipping



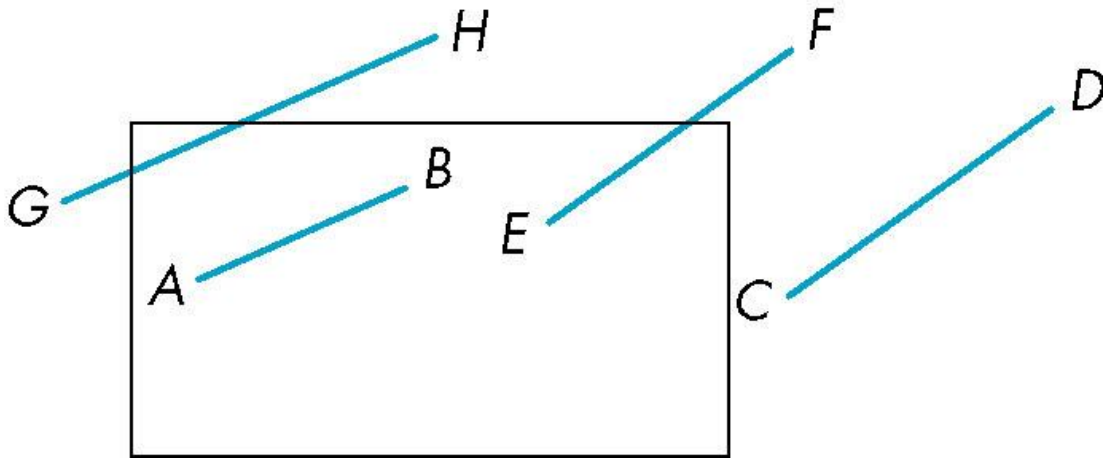
Clipping



Clipping 2D Line Segments

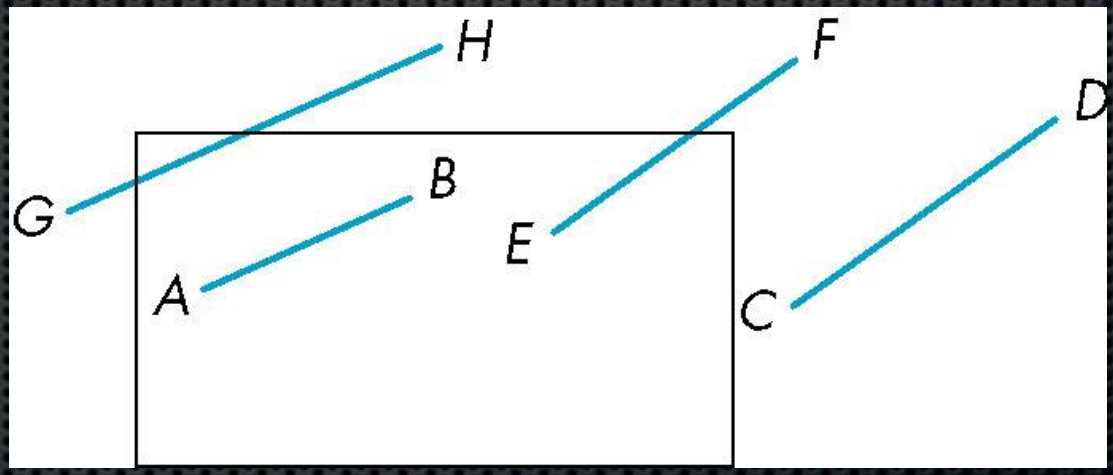


- **Brute force approach:** compute intersections with all sides of clipping window
  - Inefficient: one division per intersection



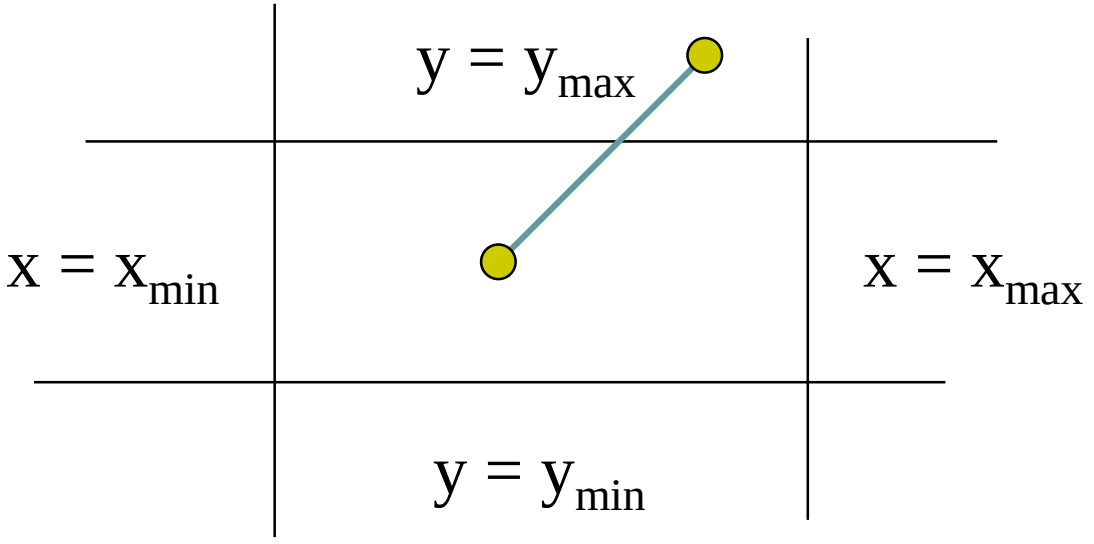
Think-Pair-Share

What's a faster approach?

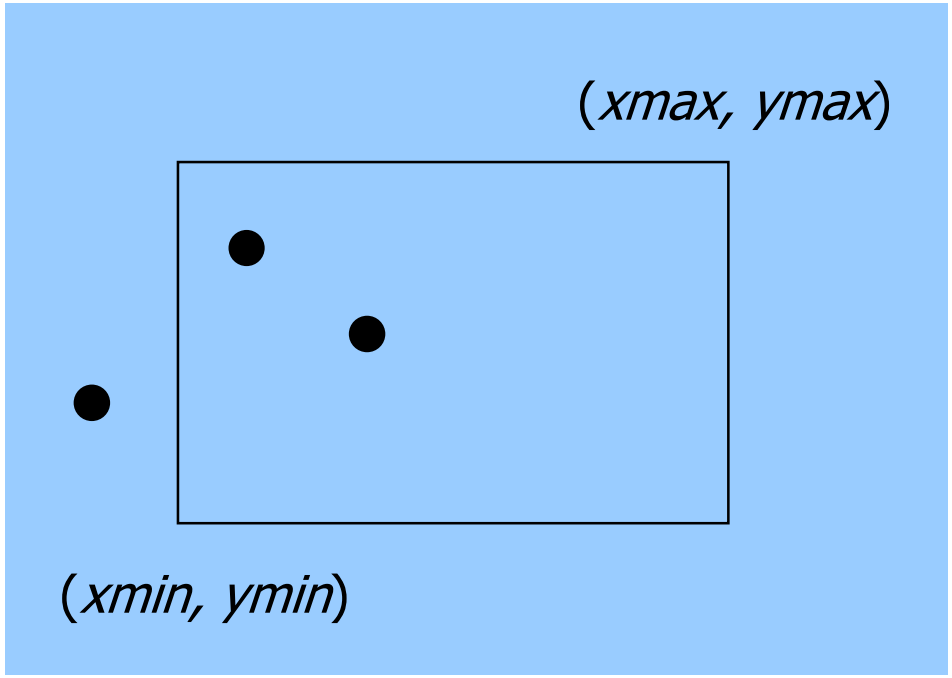


2D Clipping

- **Better Idea:** eliminate as many cases as possible without computing intersections



Clipping Points

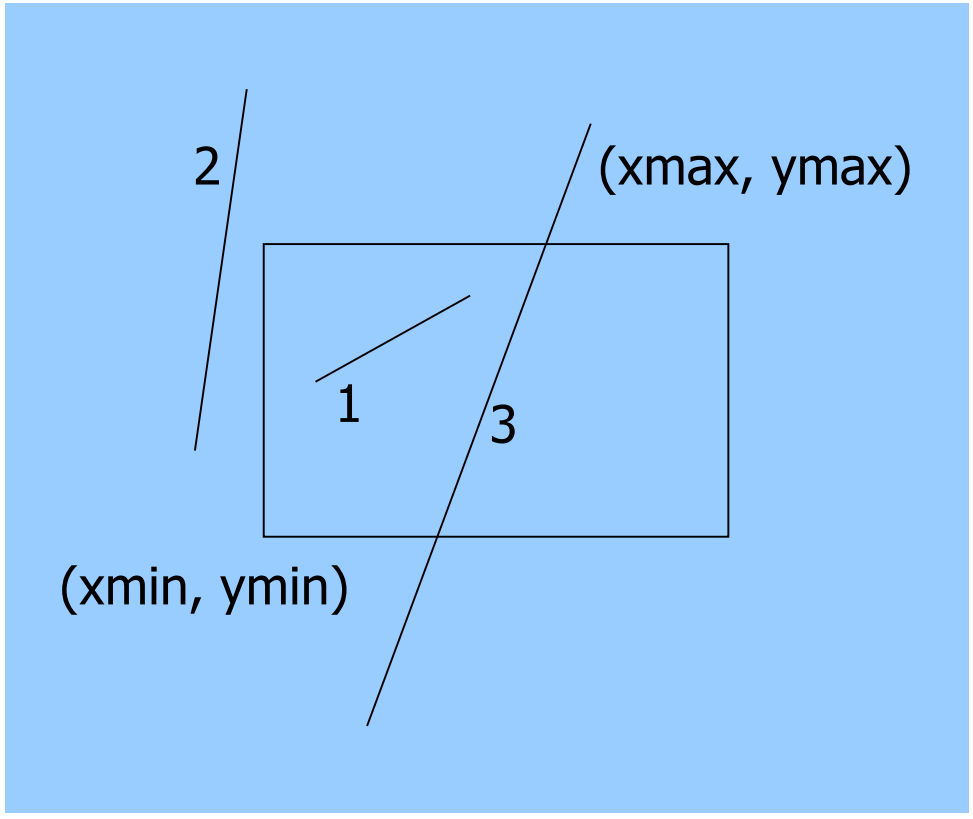


Determine whether a point (x,y) is inside or outside of the world window?

If (xmin <= x <= xmax)
and (ymin <= y <= ymax)

then the point (x,y) is inside
else the point is outside

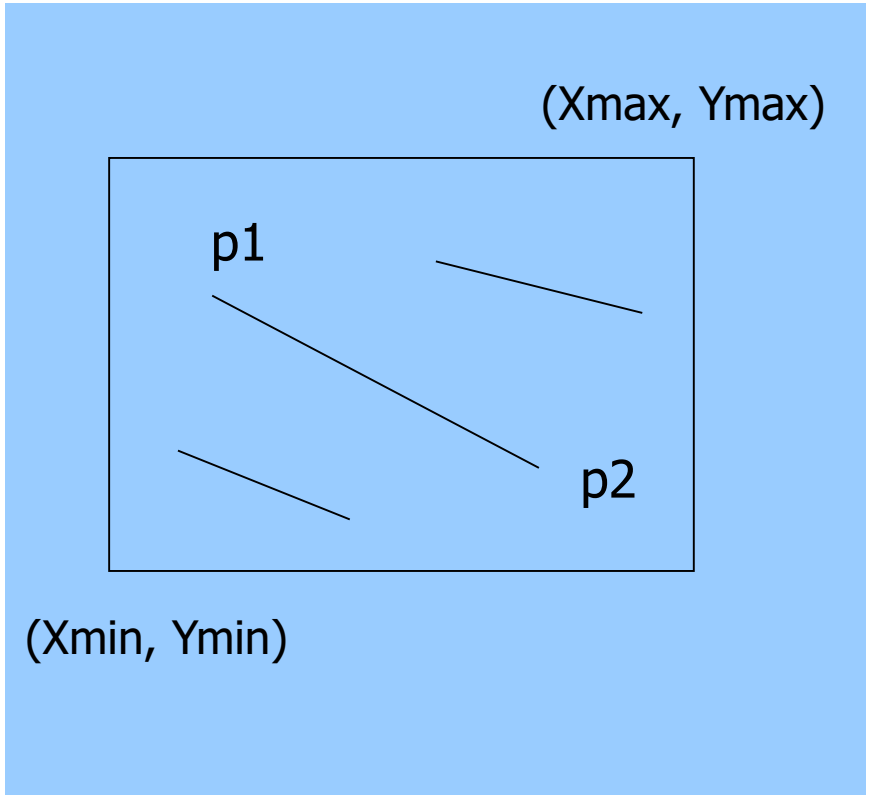
Clipping Lines



3 cases:

- **Case 1:** All of line in
- **Case 2:** All of line out
- **Case 3:** Part in, part out

Clipping Lines: Trivial Accept



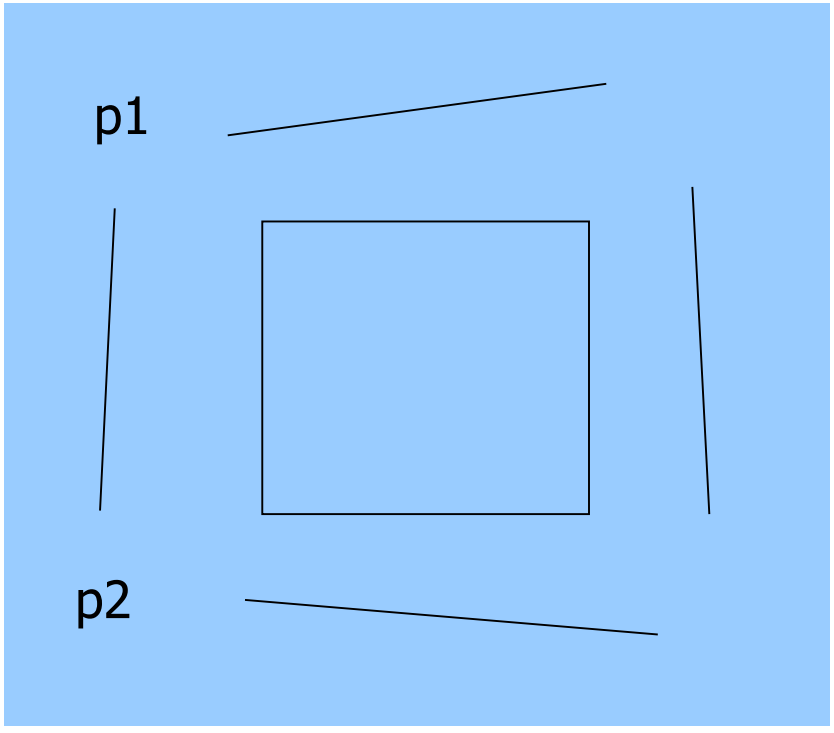
Case 1: All of line in
Test line endpoints:

Xmin <= P1.x, P2.x <= Xmax and
Ymin <= P1.y, P2.y <= Ymax

Note: simply comparing x,y values of endpoints to x,y values of rectangle

Result: trivially accept.
Draw line in completely

Clipping Lines: Trivial Reject



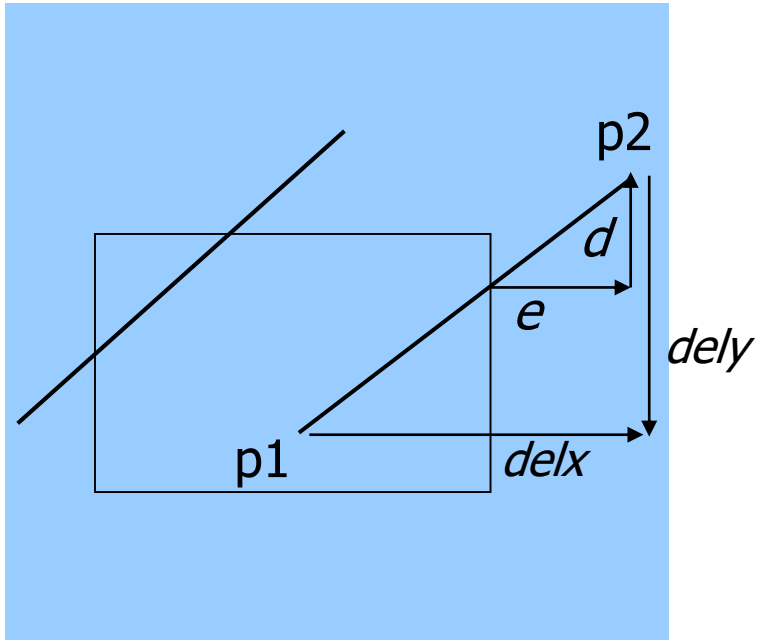
Case 2: All of line out
Test line endpoints:

- ***p1.x, p2.x <= Xmin*** **OR**
- ***p1.x, p2.x >= Xmax*** **OR**
- ***p1.y, p2.y <= ymin*** **OR**
- ***p1.y, p2.y >= ymax***

Note: simply comparing x,y values of endpoints to x,y values of rectangle

Result: trivially reject.
Don't draw line in

Clipping Lines: Non-Trivial Cases



Case 3: Part in, part out

Two variations:

- One point in, other out
- Both points out, but part of line cuts through viewport

Need to find inside segments

Use similar triangles to figure out length of inside segments

$$\frac{d}{dely} = \frac{e}{delx}$$

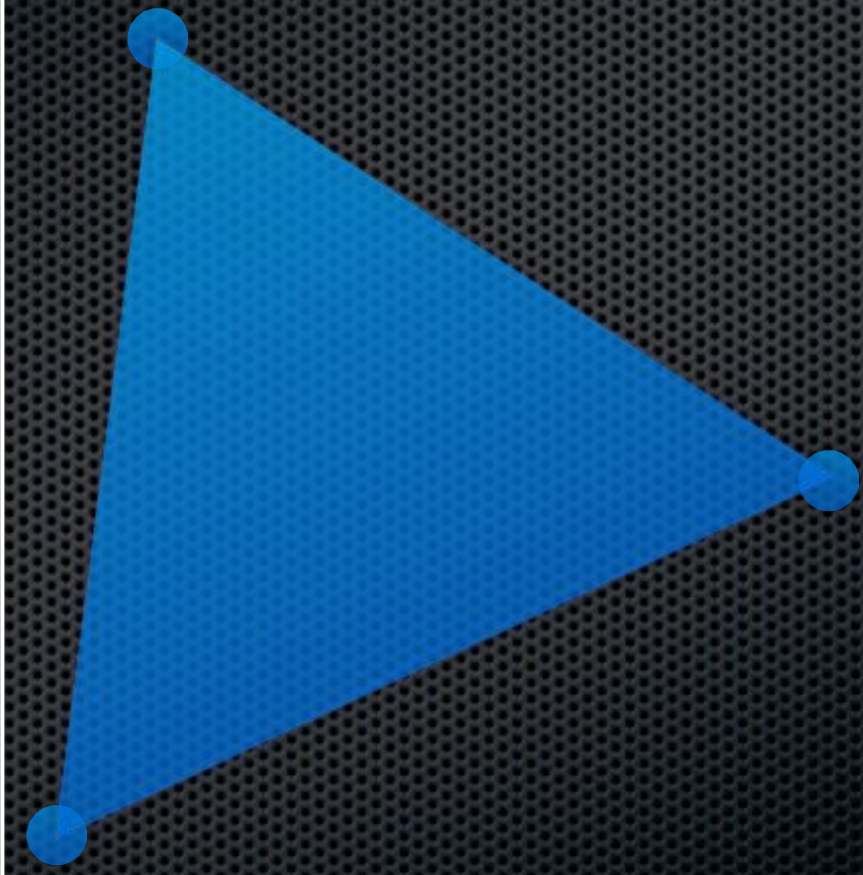
Clipping

Clipping

All 3 Points Inside

Outside

Inside

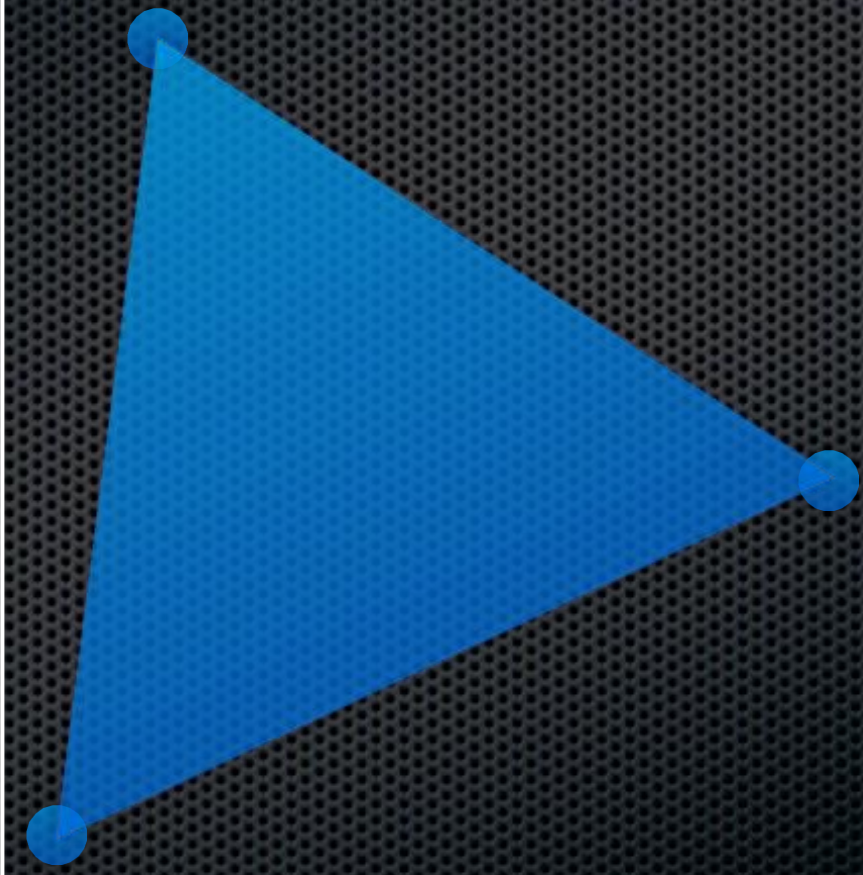


Clipping

All 3 Points Outside

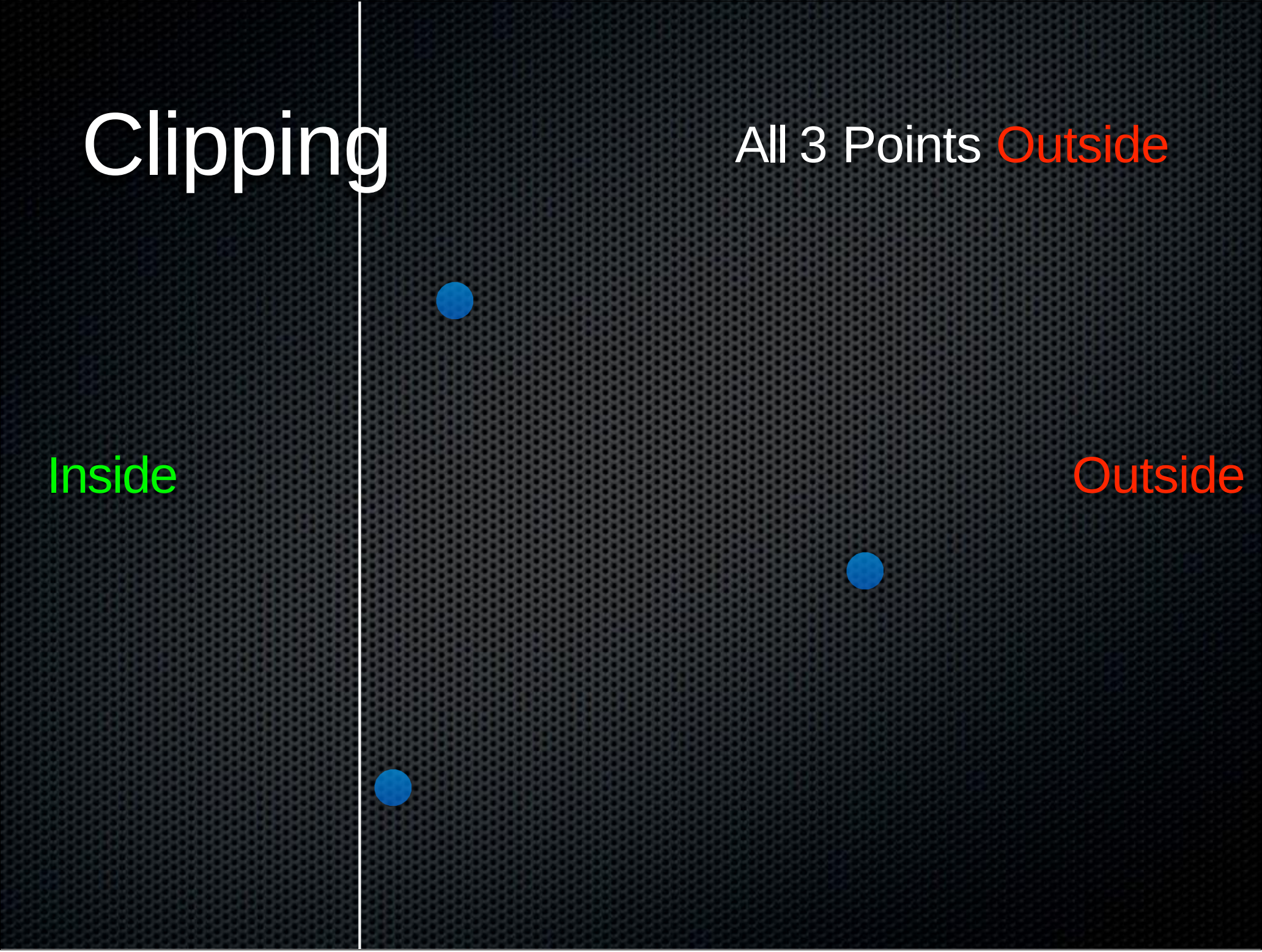
Inside

Outside



Clipping

All 3 Points Outside

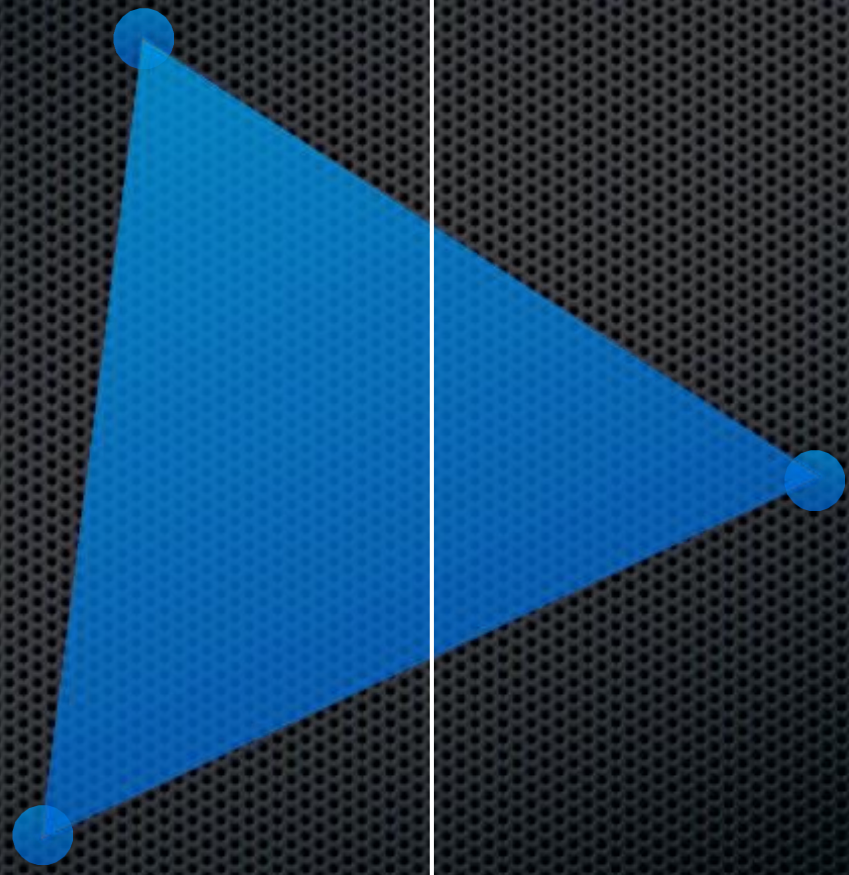


Clipping

1 Point Inside

Outside

Inside



Clipping

1 Point Inside

Outside

Inside

Clipping

1 Point Inside

Outside

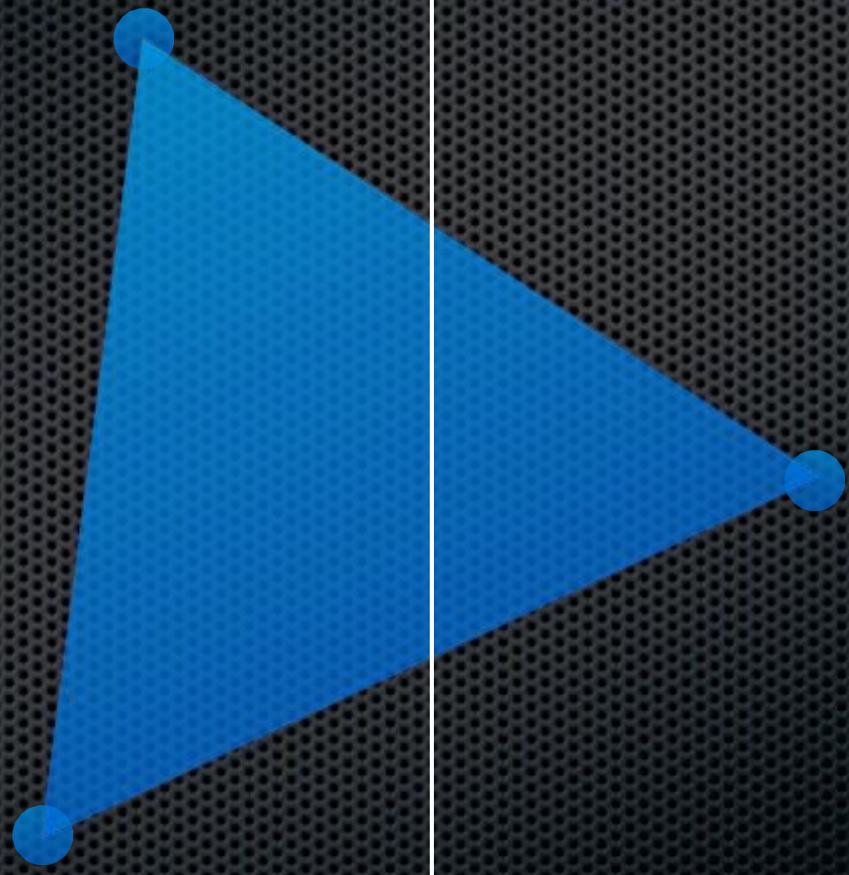
Inside

Clipping

2 Points Inside

Inside

Outside



Clipping

Inside

2 Points Inside

Outside

Clipping

2 Points Inside

Inside

Outside

Clipping

2 Points Inside

Inside

Outside

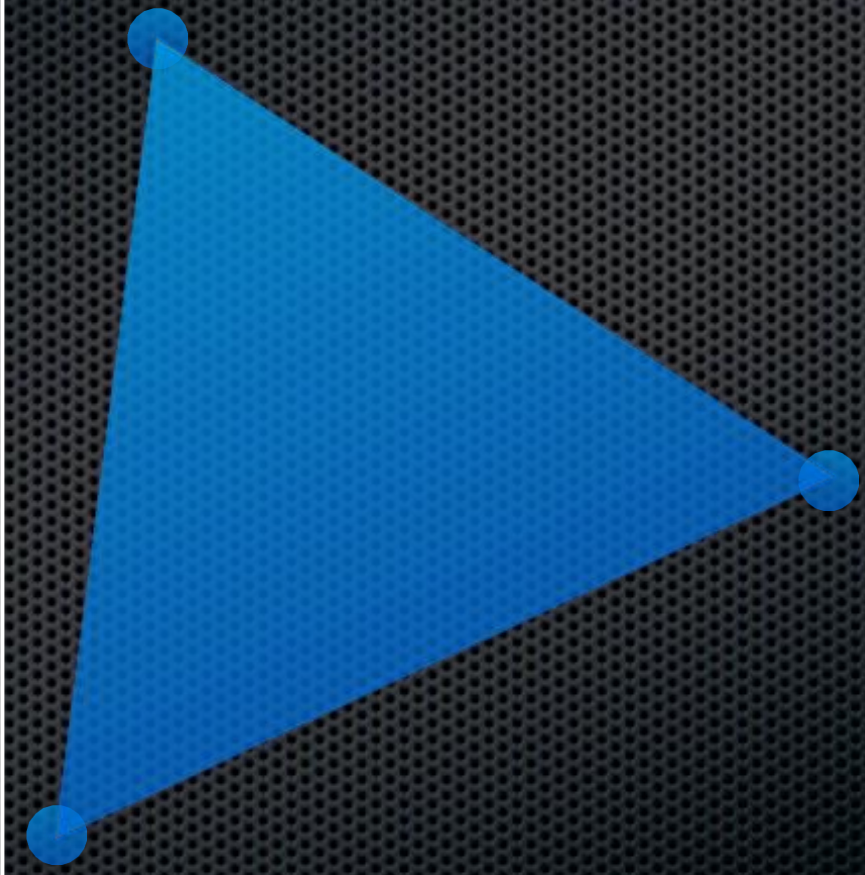
Need 2 Triangles!

Poor Man's Clip

All 3 Points Inside

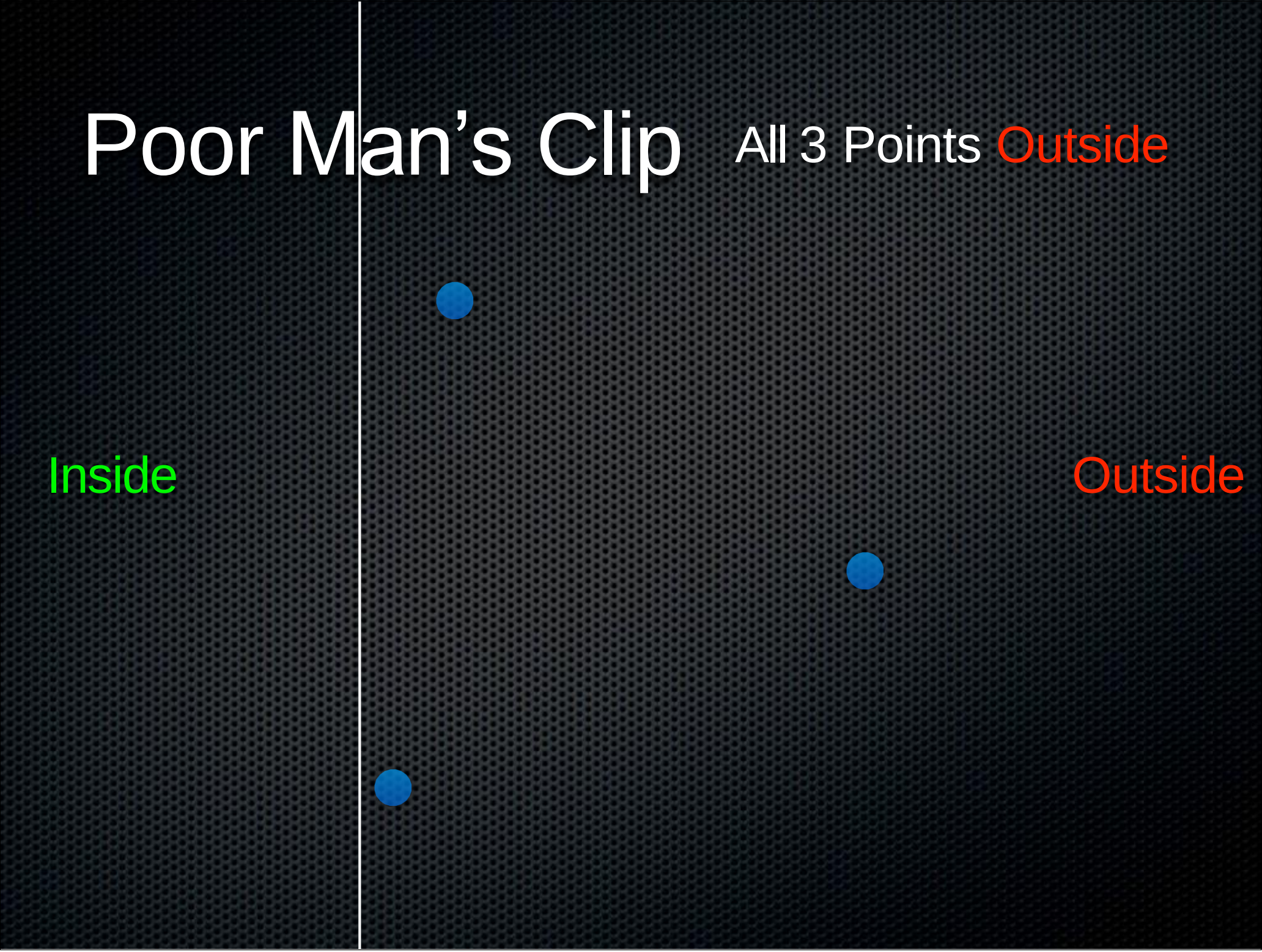
Outside

Inside

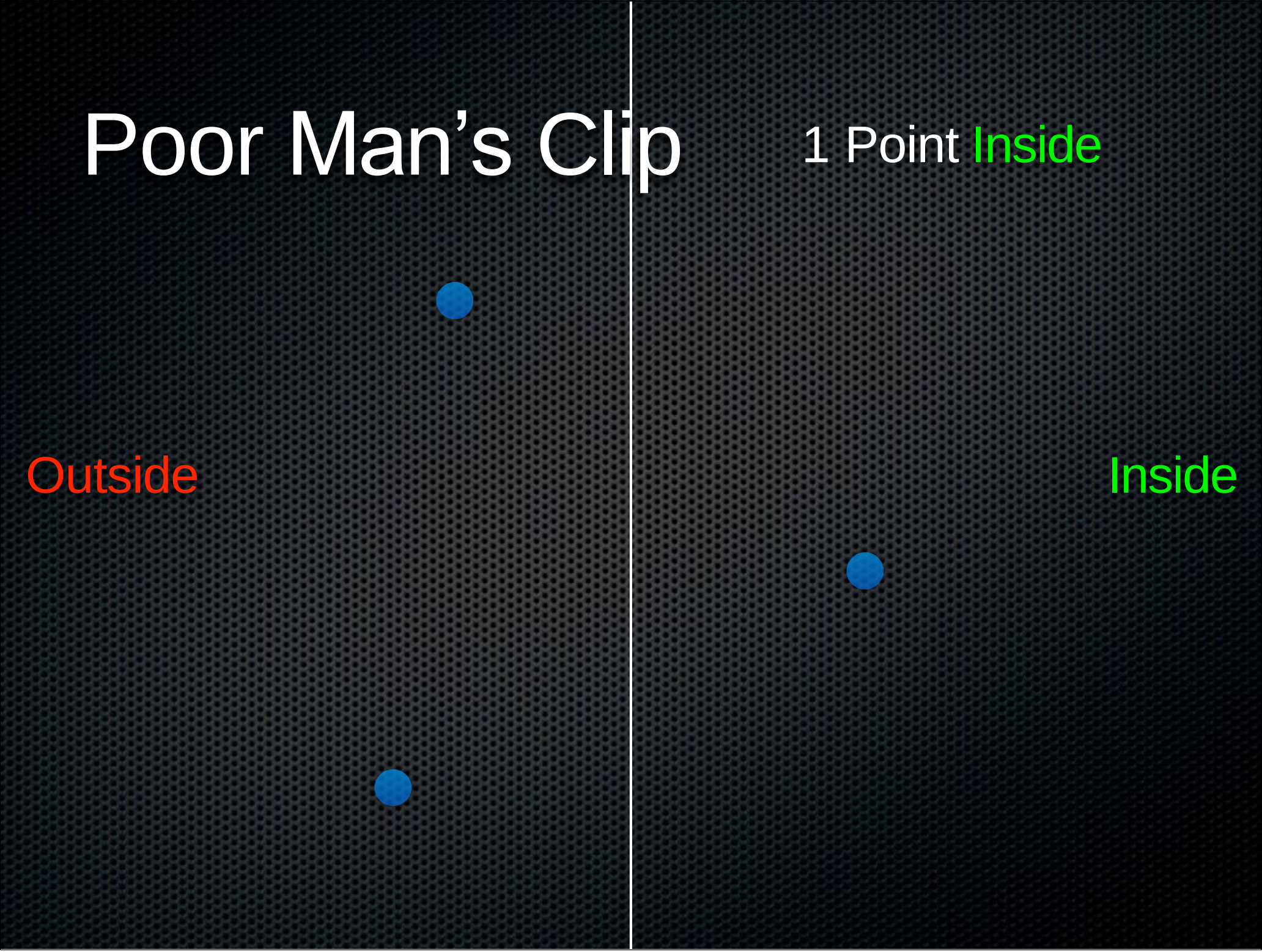


Poor Man's Clip

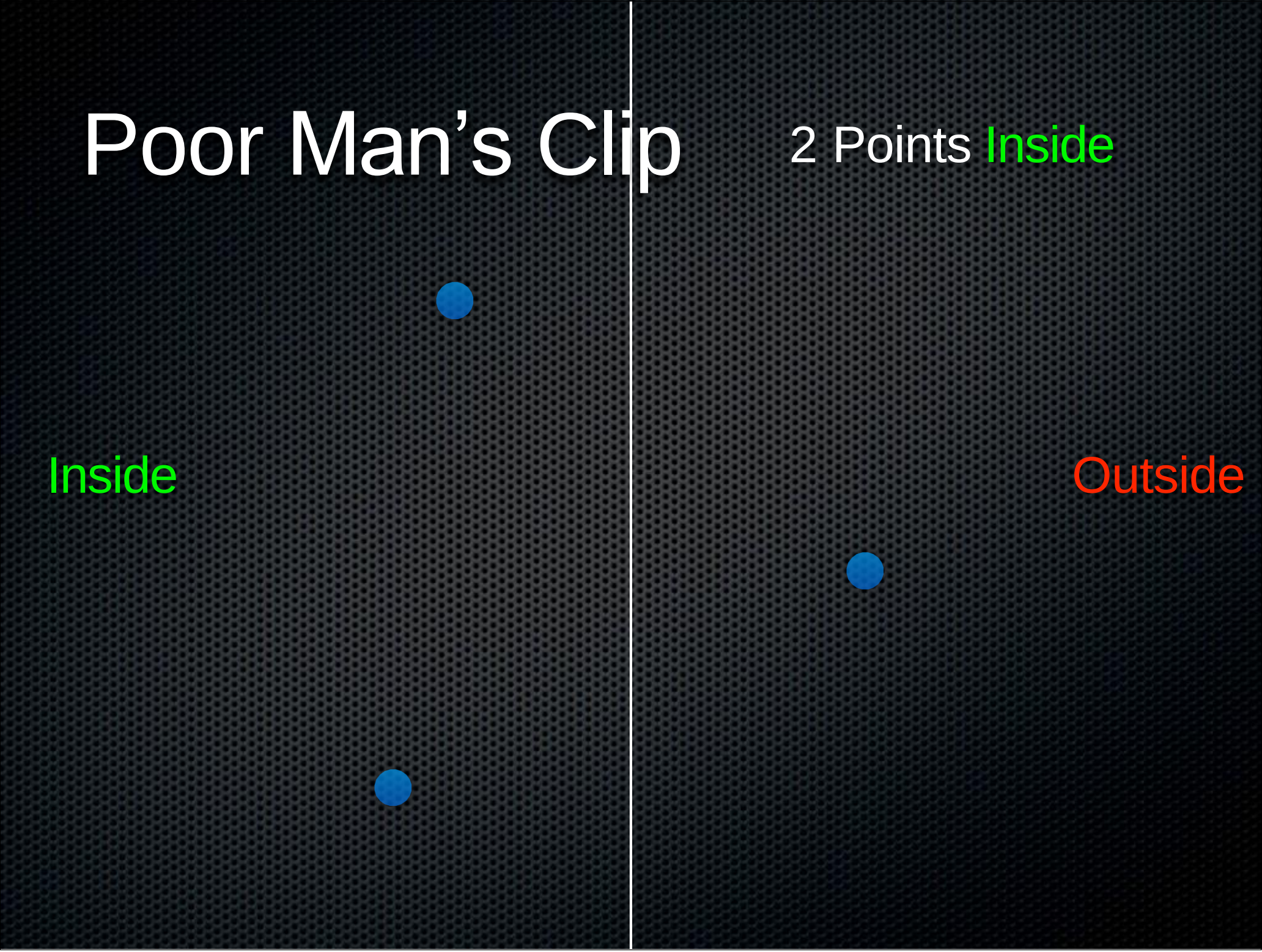
All 3 Points Outside



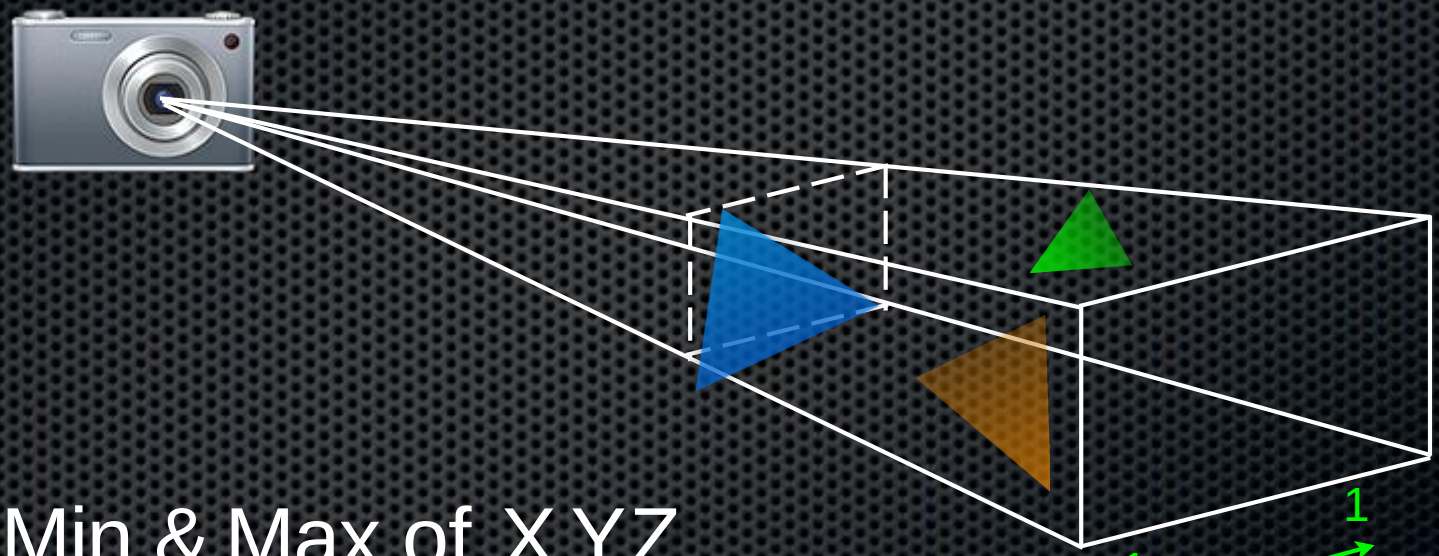
Poor Man's Clip



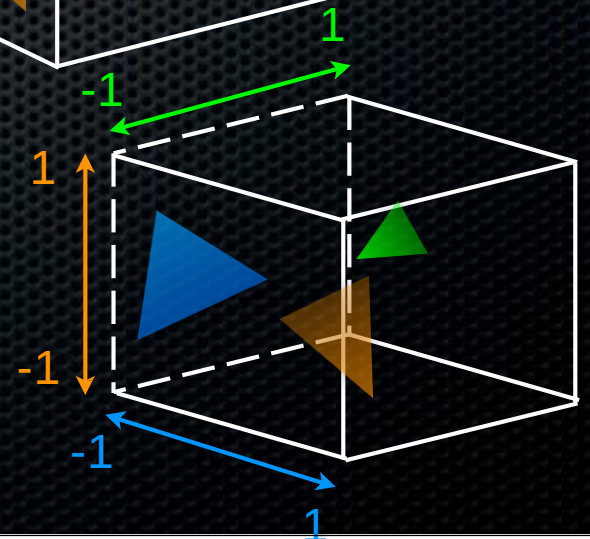
Poor Man's Clip



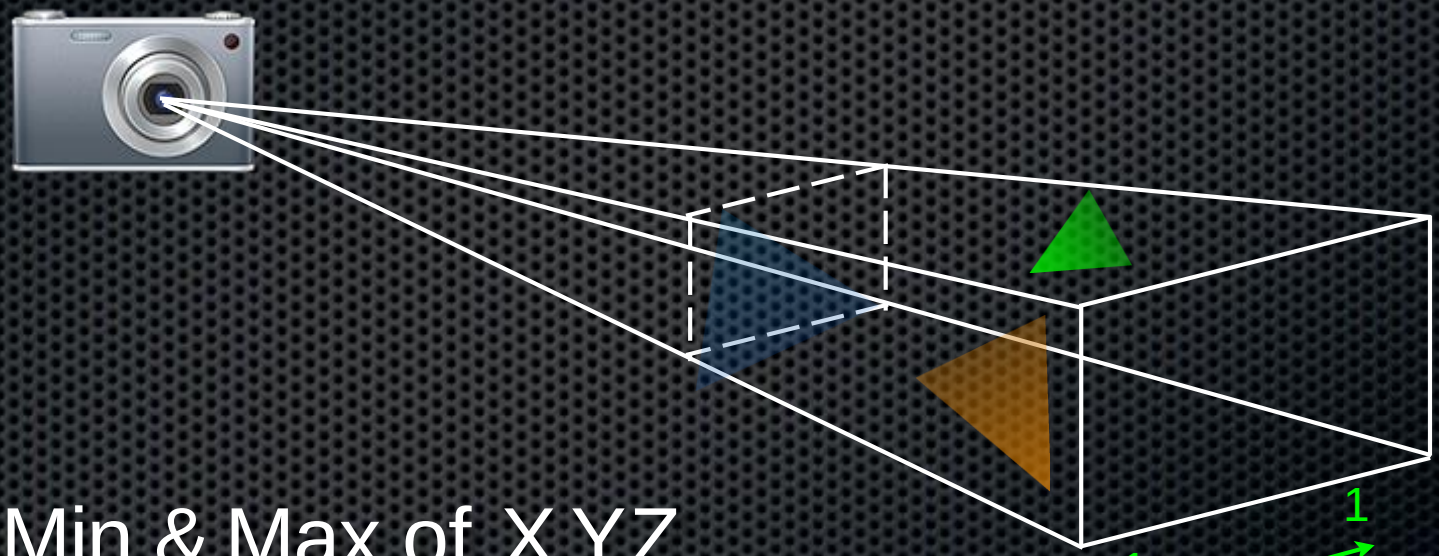
Poor Man's Clip



Check Min & Max of X,Y,Z

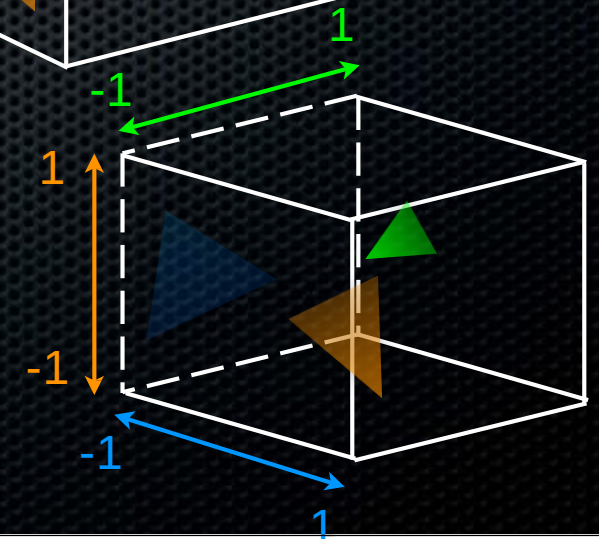


Poor Man's Clip

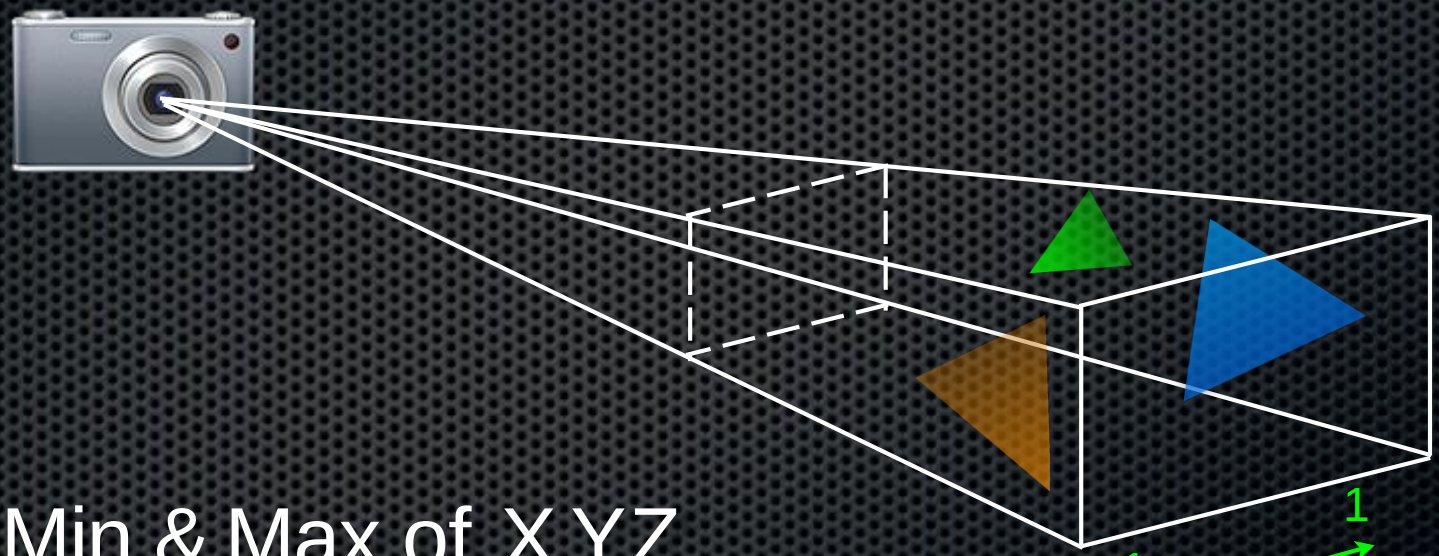


Check Min & Max of X,Y,Z

In Canonical Coordinates:
if Min < -1 or Max > 1
  Ignore Triangle

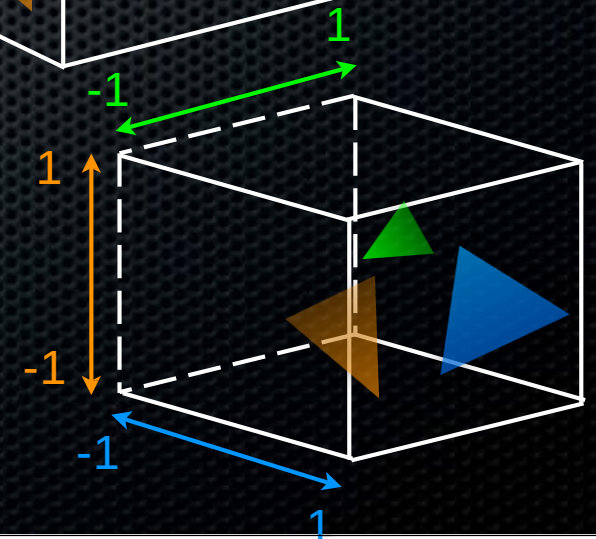


Poor Man's Clip

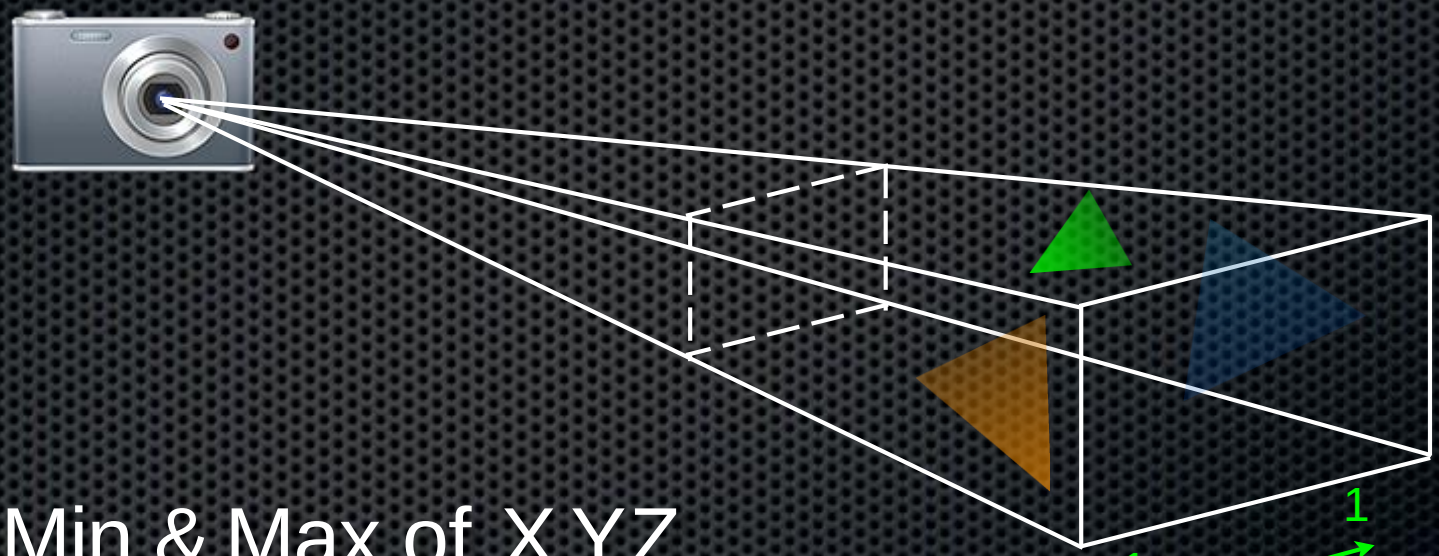


Check Min & Max of X,Y,Z

In Canonical Coordinates:
if Min < -1 or Max > 1
  Ignore Triangle



Poor Man's Clip



Check Min & Max of X,Y,Z

In Canonical Coordinates:
if Min < -1 or Max > 1
  Ignore Triangle