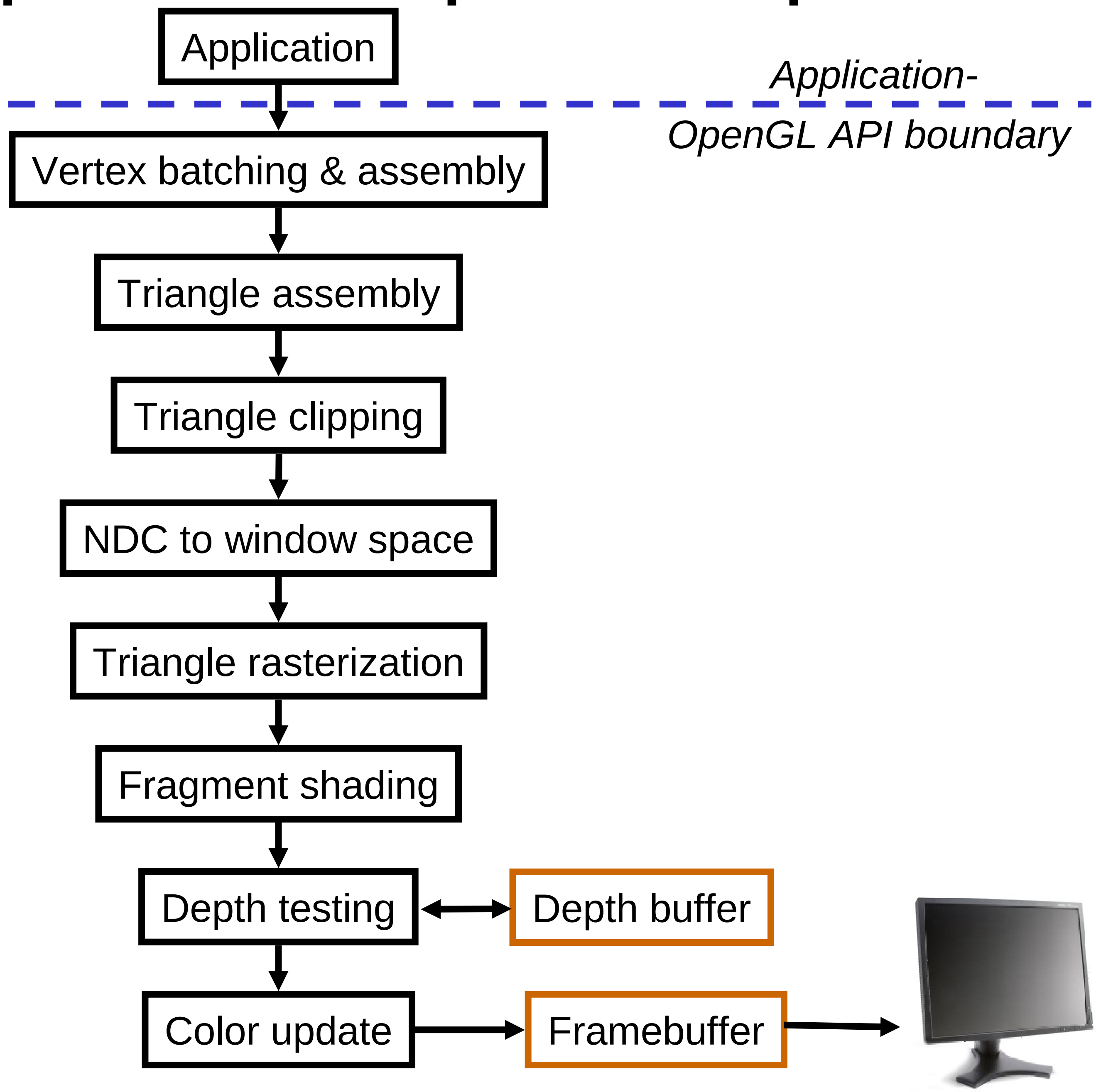
CS 354 Transformation, Clipping, and Culling

Mark Kilgard
University of Texas

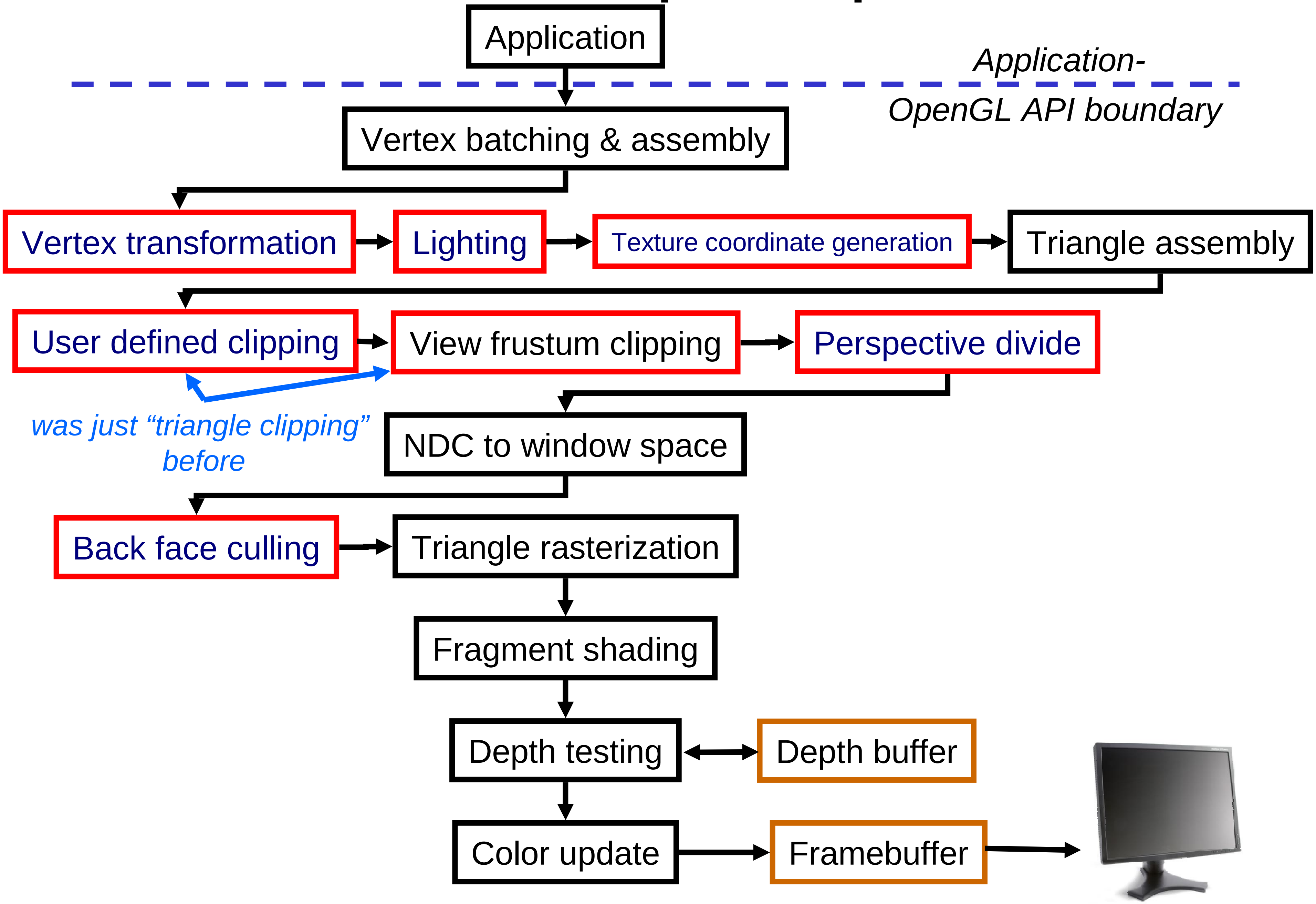
January 31, 2012

A Simplified Graphics Pipeline

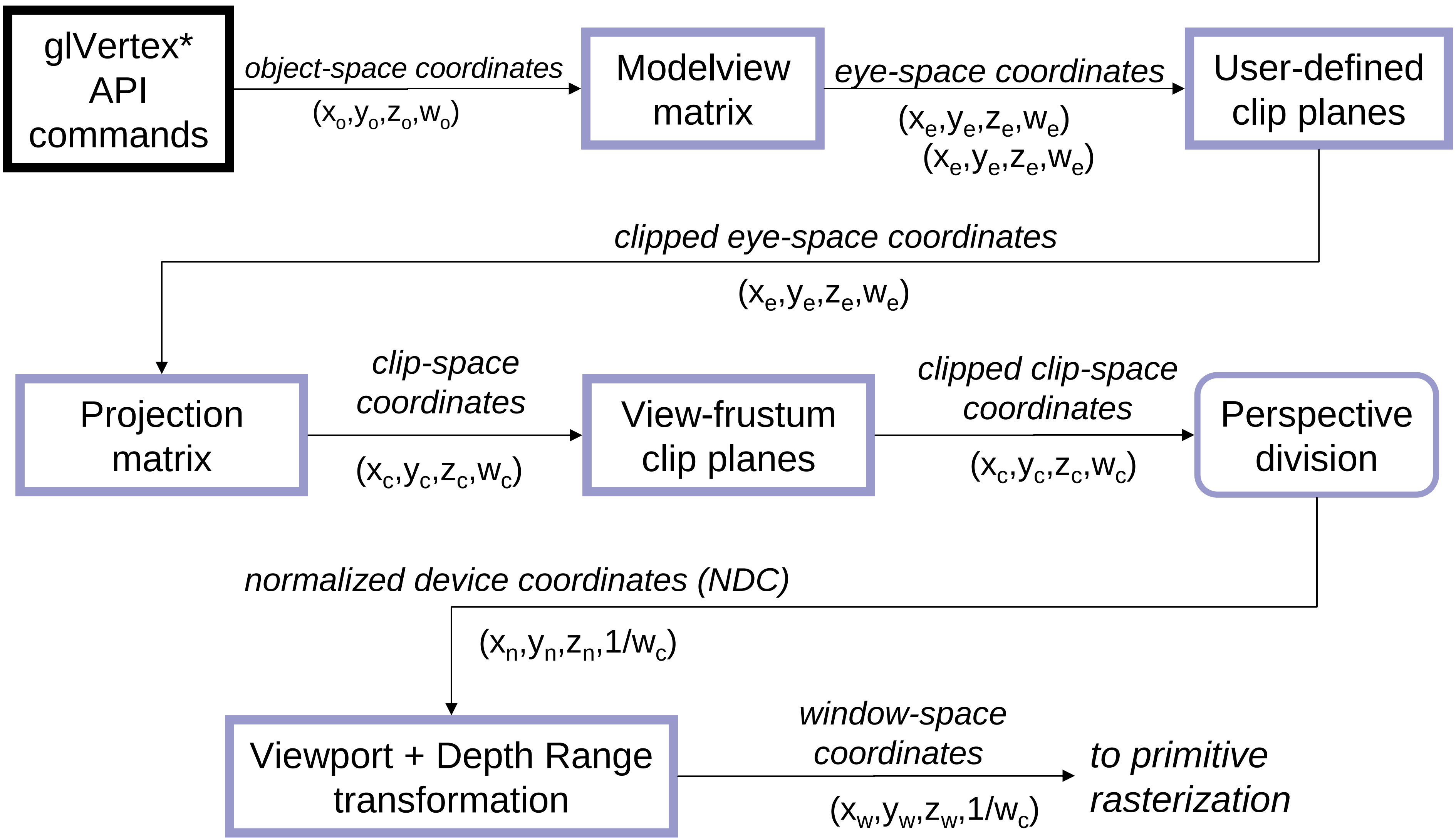
several operations left out for simplicity in explaining the simple_triangle example



A few more steps expanded

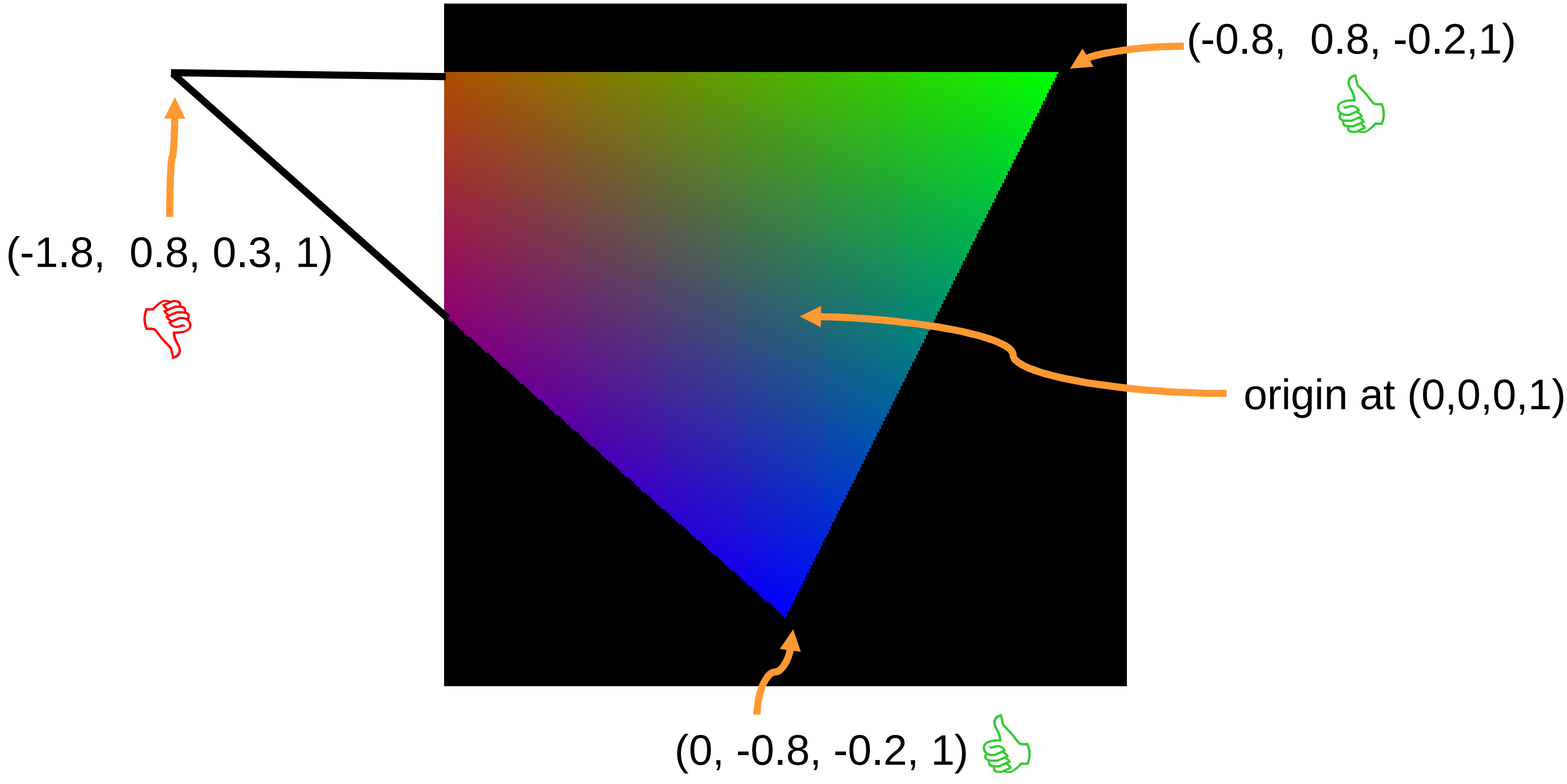


Conceptual Vertex Transformation

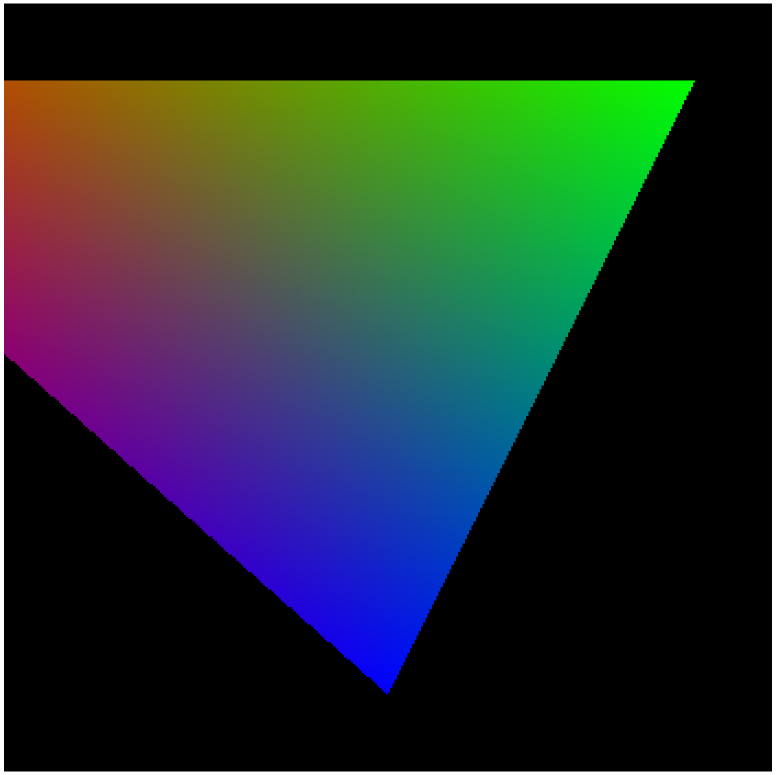


View Frustum Clipping Generalizes Cleanly

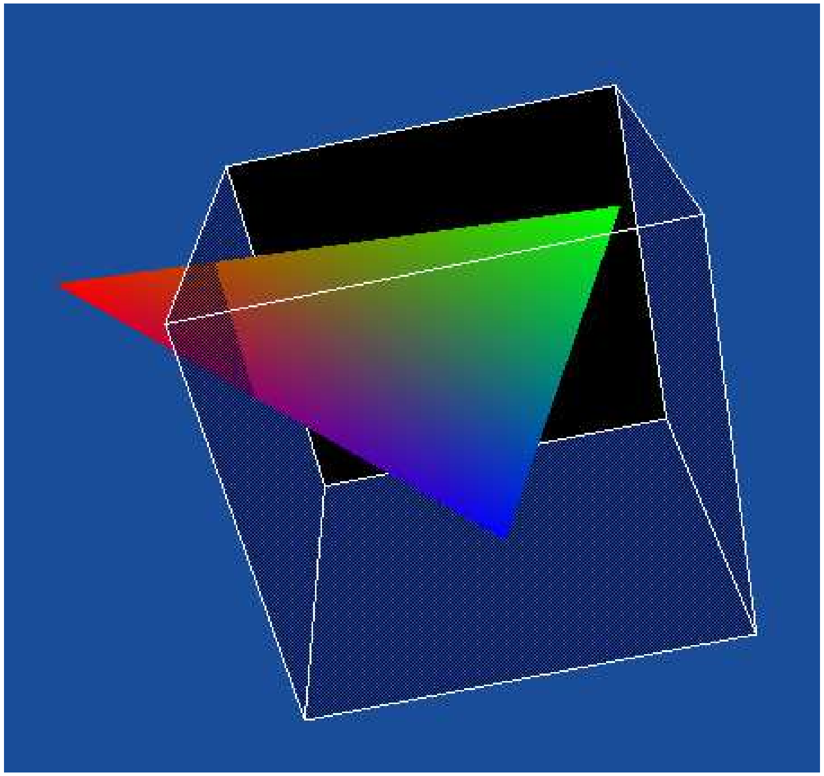
- Recall moving left vertex so it’s X = -1.8
  - Result is a clipped triangle



Clipped Triangle Visualized



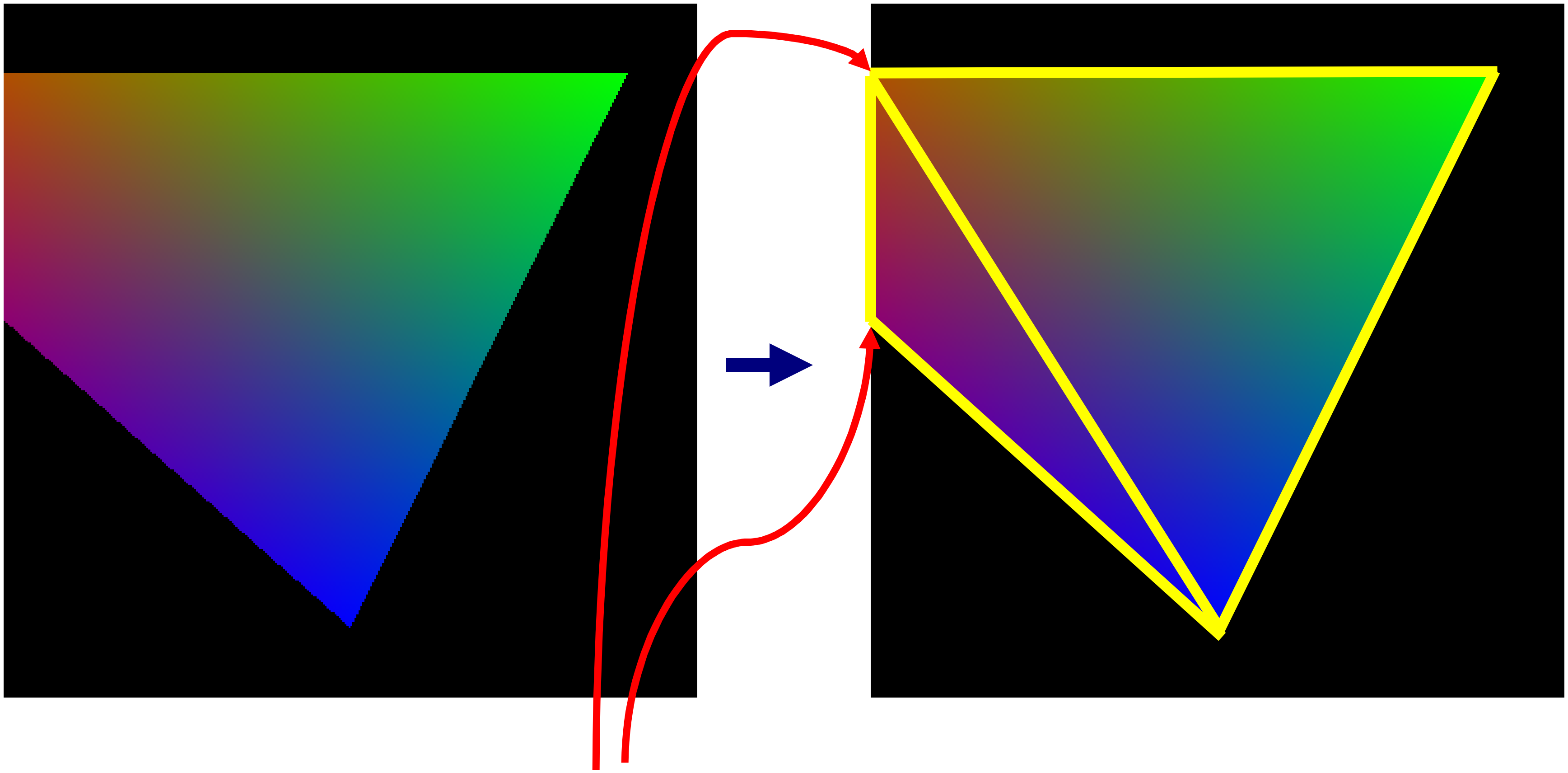
Clipped and Rasterized Normally



Visualization of NDC space

Notice triangle is "poking out" of the cube; this is the reason that should be clipped

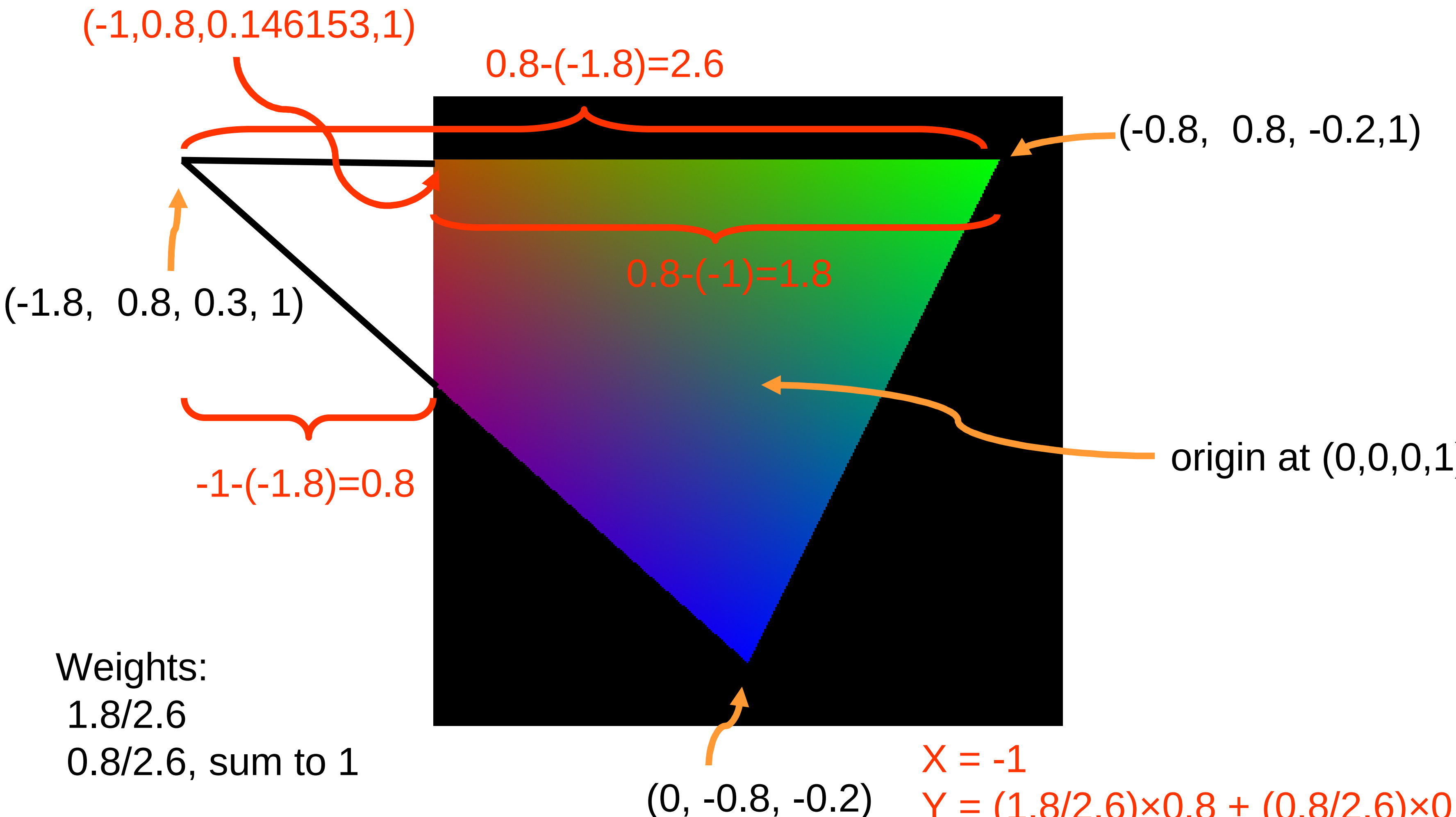
Break Clipped Triangle into Two Triangles



But how do we find these "new" vertices?

The edge clipping the triangle is the line at X = -1
so we know X = -1 at these points—but what about Y?

Use Ratios to Interpolate Clipped Positions

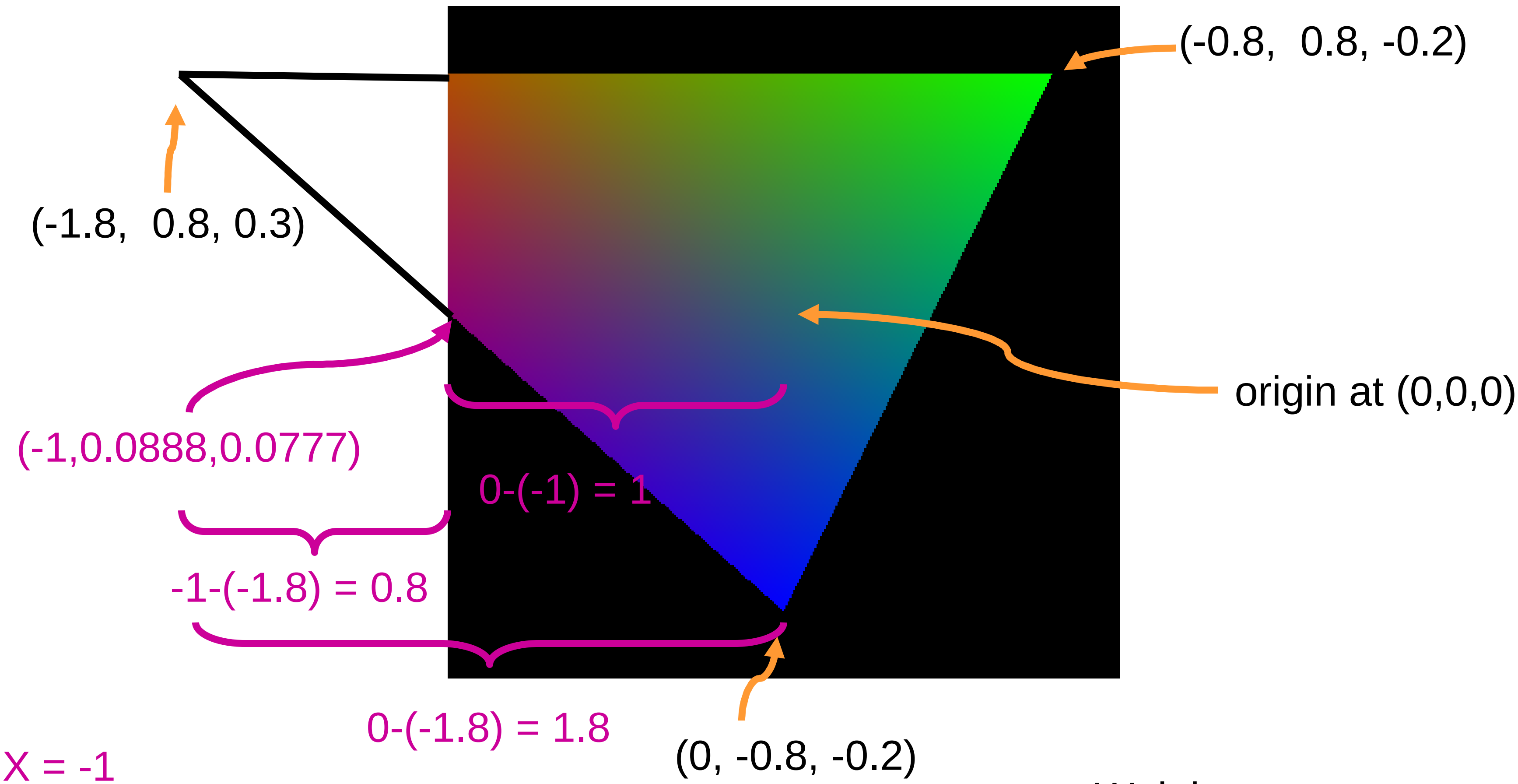


Weights:
1.8/2.6
0.8/2.6, sum to 1

X = -1
Y = (1.8/2.6)×0.8 + (0.8/2.6)×0.8
= 0.8
Z = (1.8/2.6)×0.3 + (0.8/2.6)×-0.2
= 0.1461538
W = (1.8/2.6)×1 + (0.8/2.6)×1 = 1

Straightforward because all the edges are orthogonal

Use Ratios to Interpolate Clipped Positions



X = -1

Y = (1/1.8)×0.8 + (0.8/1.8)×-0.8

= 0.08888…

Z = (1/1.8)×0.3 + (0.8/1.8)×-0.2

= 0.07777…

Weights:

1/1.8

0.8/1.8, sum to 1

Generalize to Non-1 W

- Affine clipping plane in example
  - $-1 \leq x_c$
- Generalizes to
  - $1\ x_c + 0\ y_c + 0z_c + 1\ w_c \geq 0$
  - Looks like a plane equation
    - $A\ x_c + B\ y_c + C\ z_c + D\ w_c \geq 0$
    - with coefficients A, B, C, and D

View Frustum Plane Equations

- All six view frustum planes can be described by simple projective plane equations

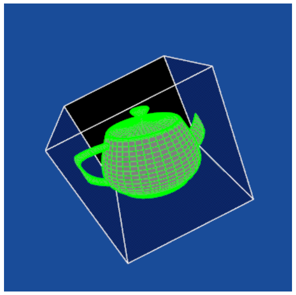
Name	**A**	**B**	**C**	**D**	**Plane equation**
Left	1	0	0	1	$1\, x_c + 1w_c \geq 0$
Right	-1	0	0	1	$-1\, x_c + 1w_c \geq 0$
Bottom	0	1	0	1	$1\, y_c + 1w_c \geq 0$
Top	0	-1	0	1	$-1\, y_c + 1w_c \geq 0$
Near	0	0	1	1	$1\, z_c + 1w_c \geq 0$
Top	0	0	-1	1	$-1\, z_c + 1w_c \geq 0$

Projective Clipping

- Each vertex computes its clip distance w.r.t. a plane
  - Plug vertex's (x,y,z,w) into $Ax+By+Cz+Dw \geq 0$ plane equation… provides a clip distance
- For two vertexes forming a triangle edge
  - Both negative? Discard the edge
  - Both positive? Accept the edge (no clipping)
  - One negative, one positive
    - Clipping is needed
    - Compute t as s / (s + p) where s and p are clip distances
      - s is the "inside" distance; p is the "outside" distance
    - Weight all per-vertex attributes based on t
    - Makes new "clipped" vertex on the clip plane
    - Generate 1 triangle if 1 of 3 vertices is inside; if 2 inside, generate 2 triangles
- Repeat process (recursively) for all clip planes
  - Only slightly more complicated than prior clipping algorithm

Readily Extends to User-defined Clip Planes

- In addition to the six view frustum planes of clip space…
  - OpenGL supports user-defined clip planes
  - Allows slicing into geometry



- Operate in eye space instead of clip space
  - Enabled with glEnable(GL_CLIP_PLANE0+*num*)
  - Plane equation set by glClipPlane
    - Clip planes are transformed current modelview projection matrix
  - Plane equation is $Ax_e+By_e+Cz_e+Wz_e \geq 0$
    - Instead of using (x_c, y_c, z_c, w_c) as view frustum planes do

(Clip) Plane Transformation

- Vertex positions (and direction vectors) are transformed like column vectors

$$\begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix} = \begin{bmatrix} MV_0 & MV_4 & MV_8 & MV_{12} \\ MV_1 & MV_5 & MV_9 & MV_{13} \\ MV_2 & MV_6 & MV_{10} & MV_{14} \\ MV_3 & MV_7 & MV_{11} & MV_{15} \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ z_o \\ w_o \end{bmatrix}$$

glVertex4f *parameters*

- Plane equations are transformed like row vectors

$$\begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix} = \begin{bmatrix} x_{clip} & y_{clip} & z_{clip} & w_{clip} \end{bmatrix} \begin{bmatrix} MV_0 & MV_4 & MV_8 & MV_{12} \\ MV_1 & MV_5 & MV_9 & MV_{13} \\ MV_2 & MV_6 & MV_{10} & MV_{14} \\ MV_3 & MV_7 & MV_{11} & MV_{15} \end{bmatrix}$$

glClipPlane *parameters*

Conceptual Vertex Transformation