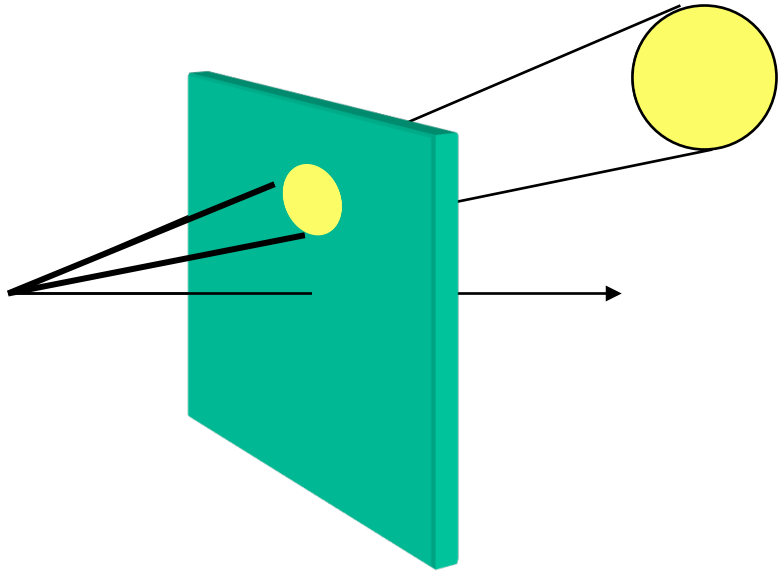
Perspective

- How to draw perspective images
- Difference between perspective projection and transformation
- Inverse perspective transform

Puzzle

- Is the projection of the moon on a window glass a disk?
- Justify your answer.



Lecture Objectives

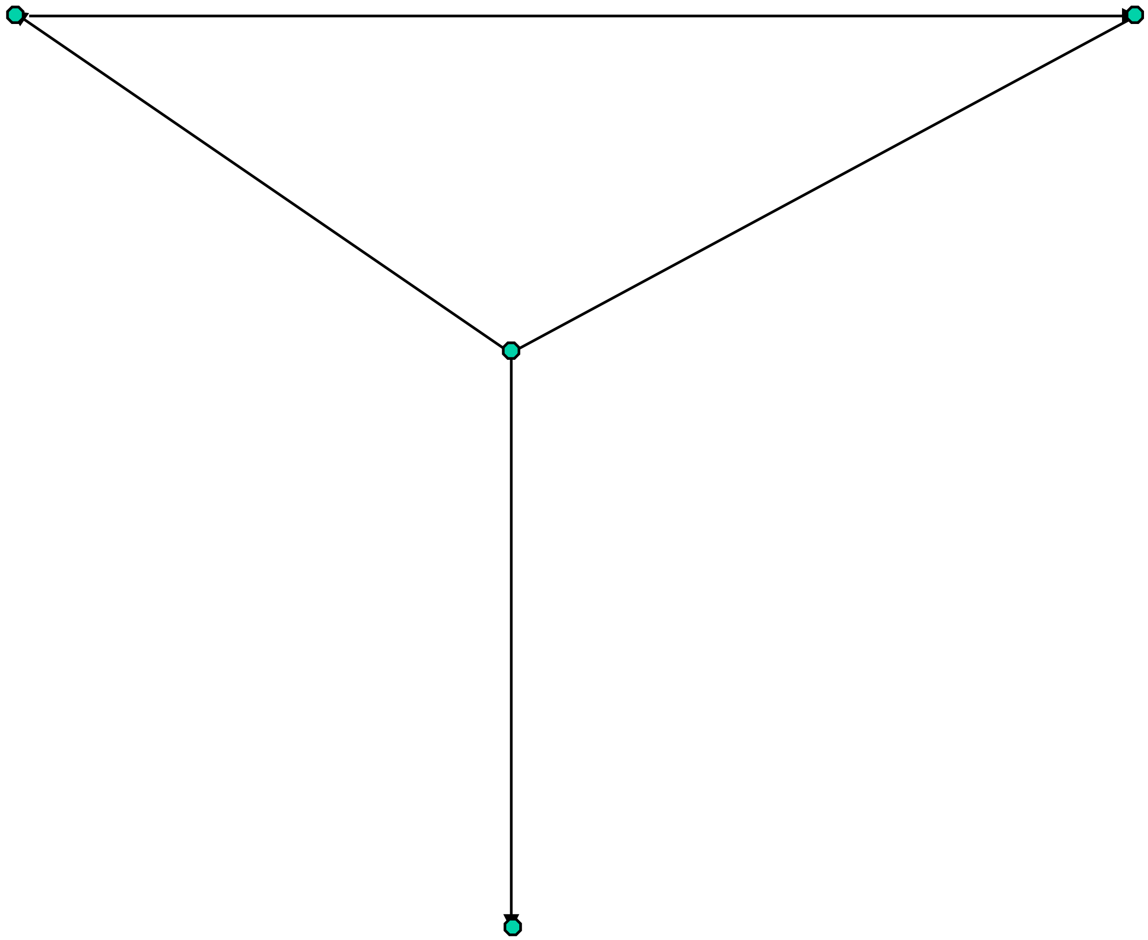
- Perspective projection equation
- Why is the perspective projection of a triangle a triangle
- How to draw a 3-point perspective view of a box
- Why you can't use perspective projection for scan-conversion
- Conversion of points and vectors to screen coordinates
- Perspective transformation and its equations
- Geometric construction of the perspective transform
- Incremental z computation during scan-conversion
- What does perspective transform map $\{0<z\}$ to
- Definition and properties of the horizon plane
- What does perspective transform map $\{n<z<f\}$ to

Additional Objectives

- Proof that perspective transform maps triangles onto triangles
- Why $(x,y,z) \rightarrow (dx/(d+z), dy/(d+y), z)$ is not an option
- Computing the correct eye position for looking at a picture of a block
- The inverse of a perspective transform
- The pre-image of $\{z=k\}$
- The distribution of z-buffer round-off error as a function of z

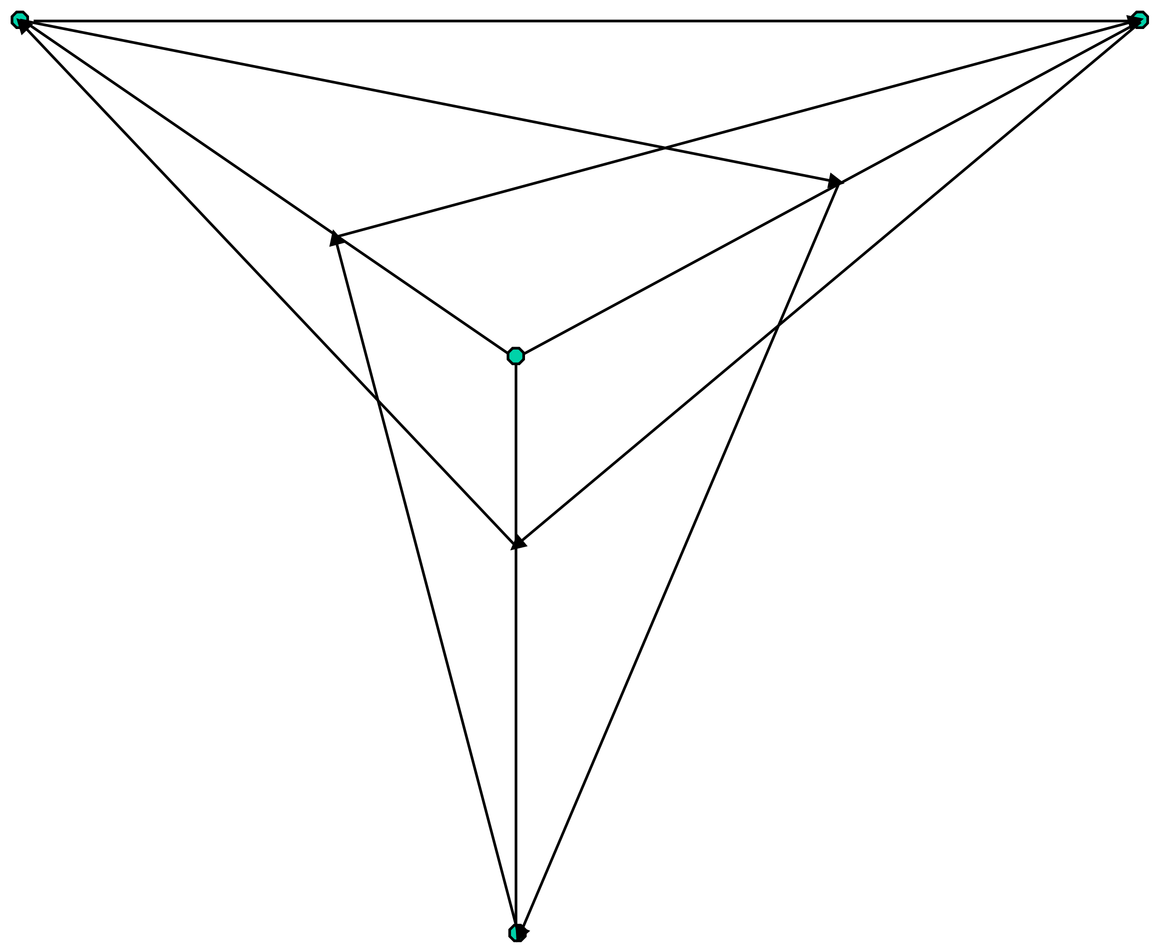
Drawing a 3-point perspective of a box (A)

- Pick corner and 3 vanishing points



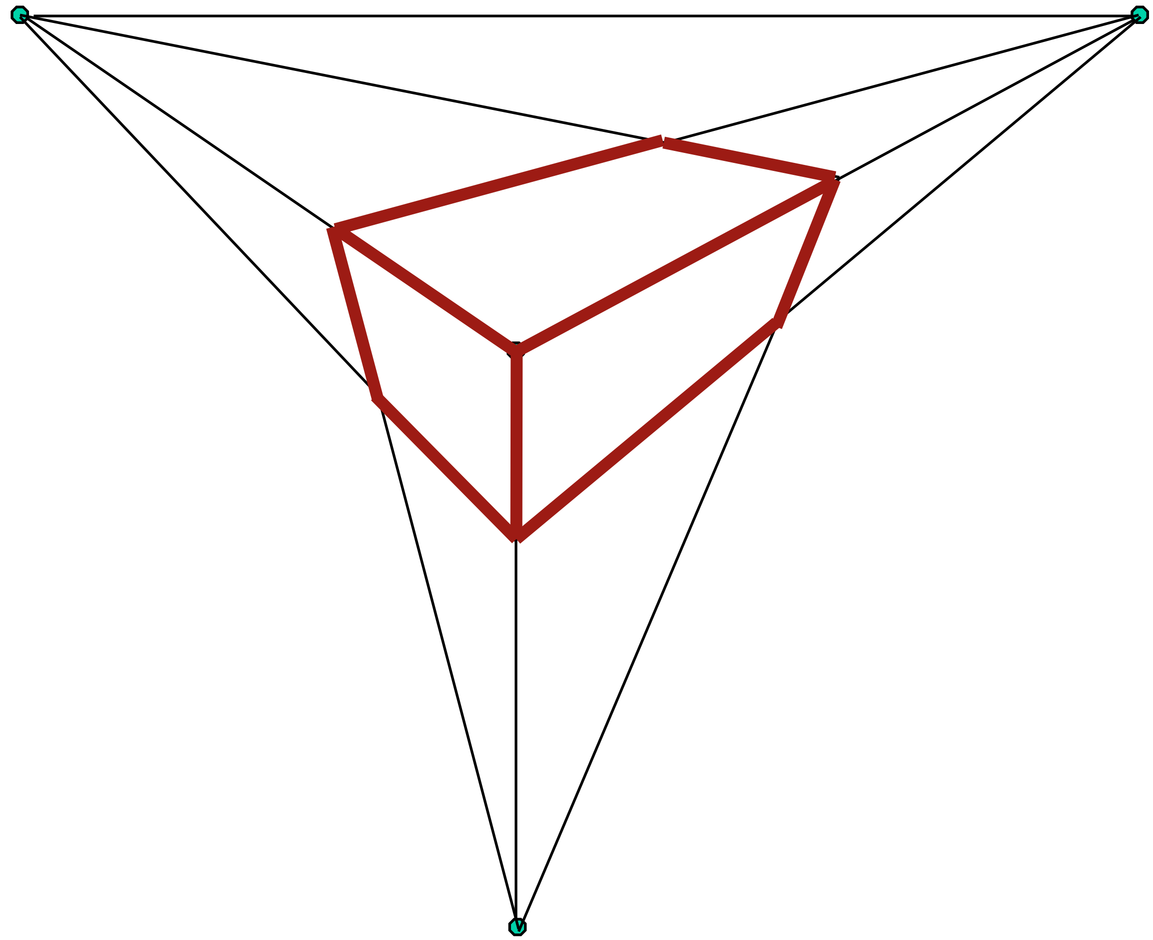
Drawing a 3-point perspective of a box (B)

- Join vanishing points to ray-points



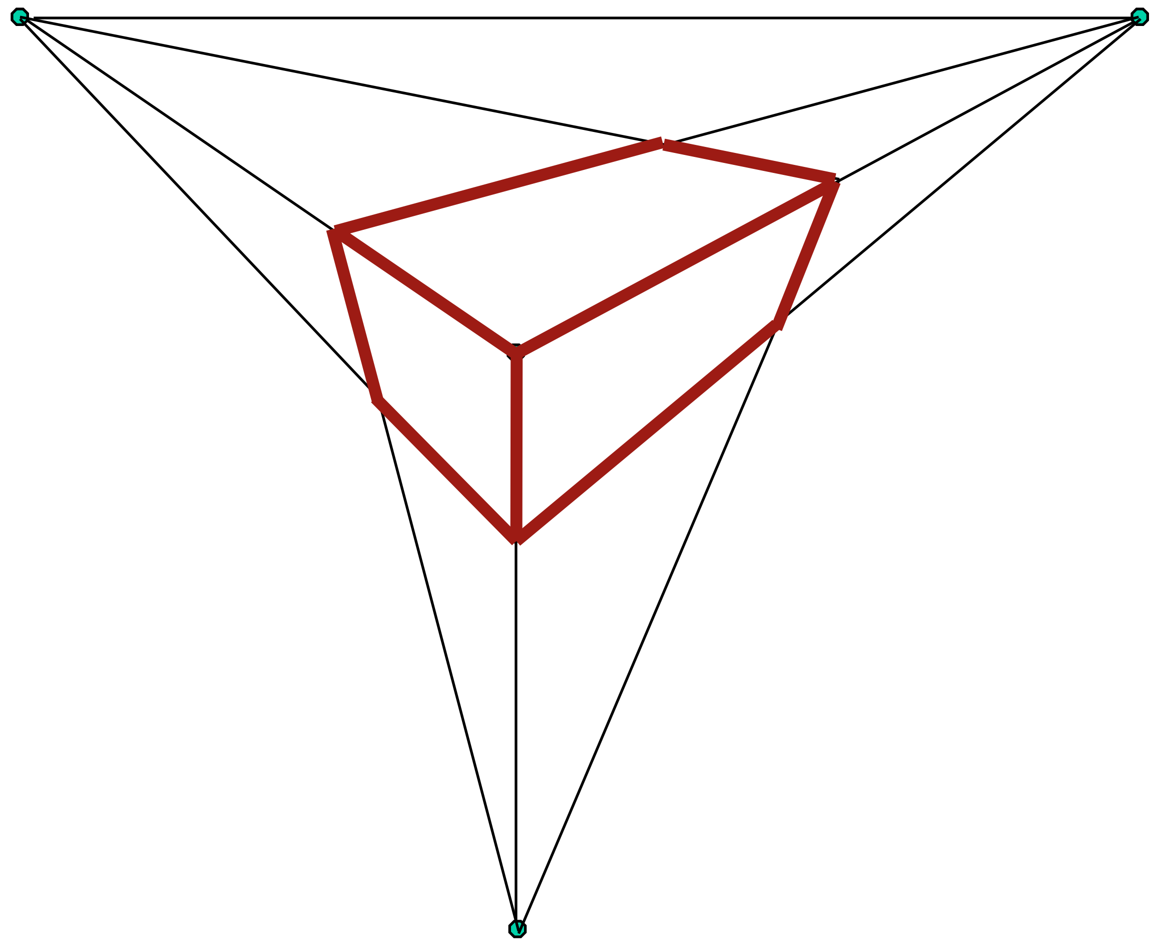
Drawing a 3-point perspective of a box (C)

- Highlight the edges of the box



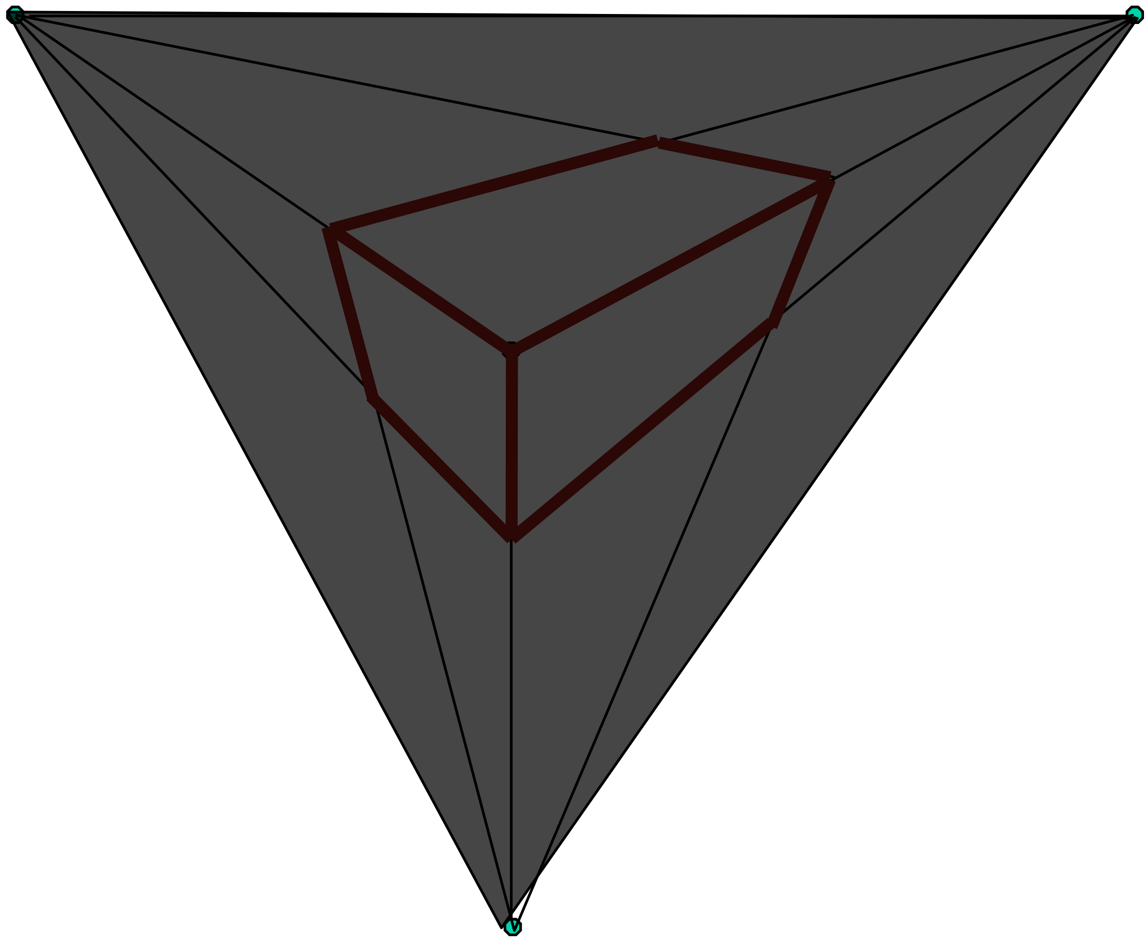
Looking at a 3-point perspective of a box (A)

- Where should the eye be?



Looking at a 3-point perspective of a box (E)

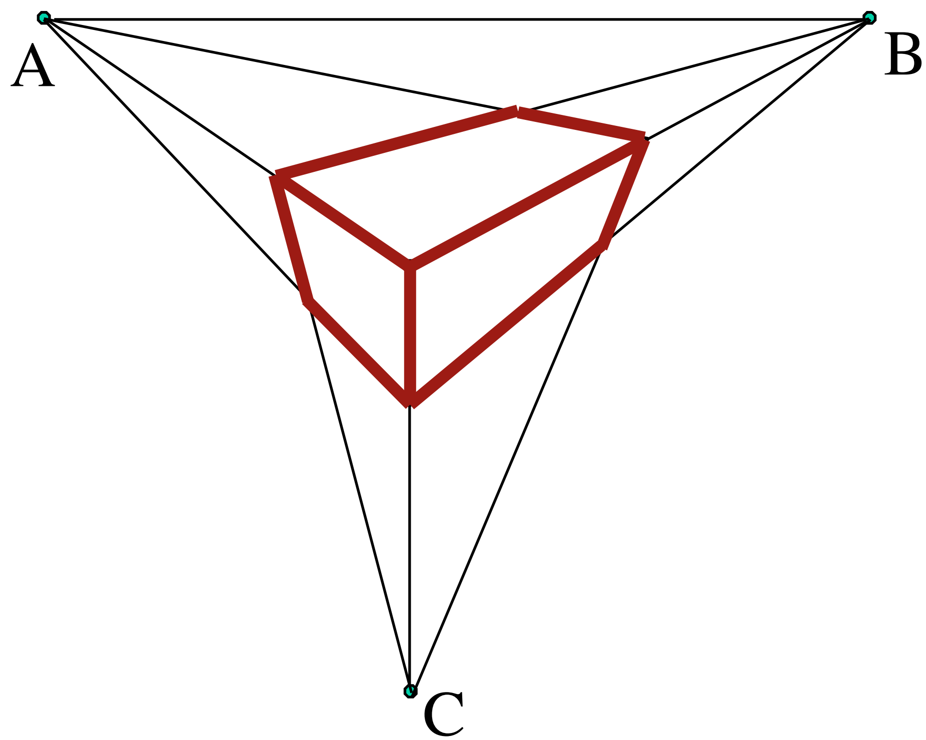
- At the corner of a box positioned to fit in this hole.



Where exactly should the viewer stand

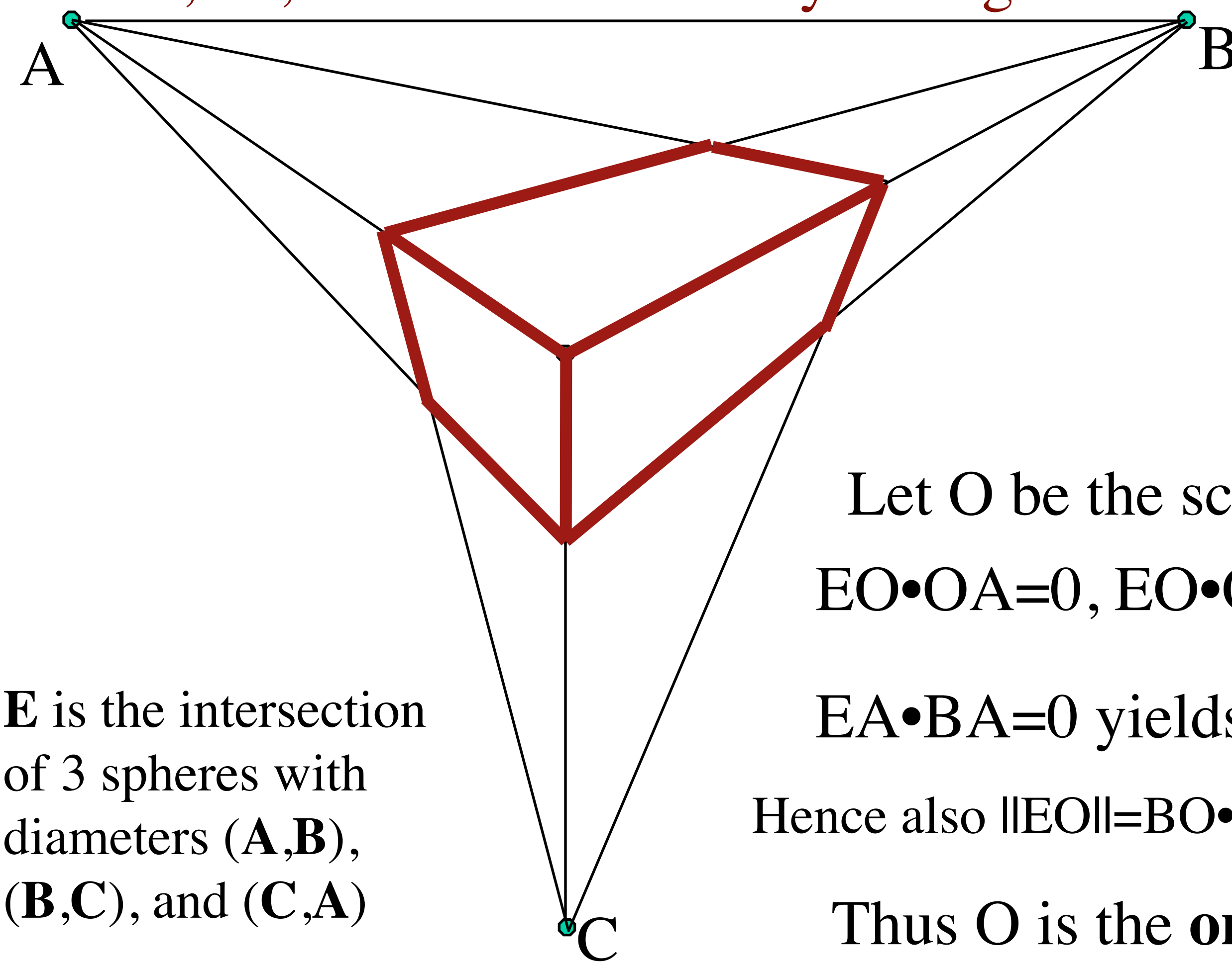
when looking at the image of a block?

- Let A, B, C be the 3 vanishing points
- Compute the viewpoint E



Computing E

- EA,EB,EC must be mutually orthogonal



EA•EB=0
EB•EC=0
EC•EA=0

Solve for E

E is the intersection of 3 spheres with diameters (**A**,**B**), (**B**,**C**), and (**C**,**A**)

Let O be the screen projection of E

EO•OA=0, EO•OB=0, EO•OC=0

EA•BA=0 yields ||EO||=AO•OB

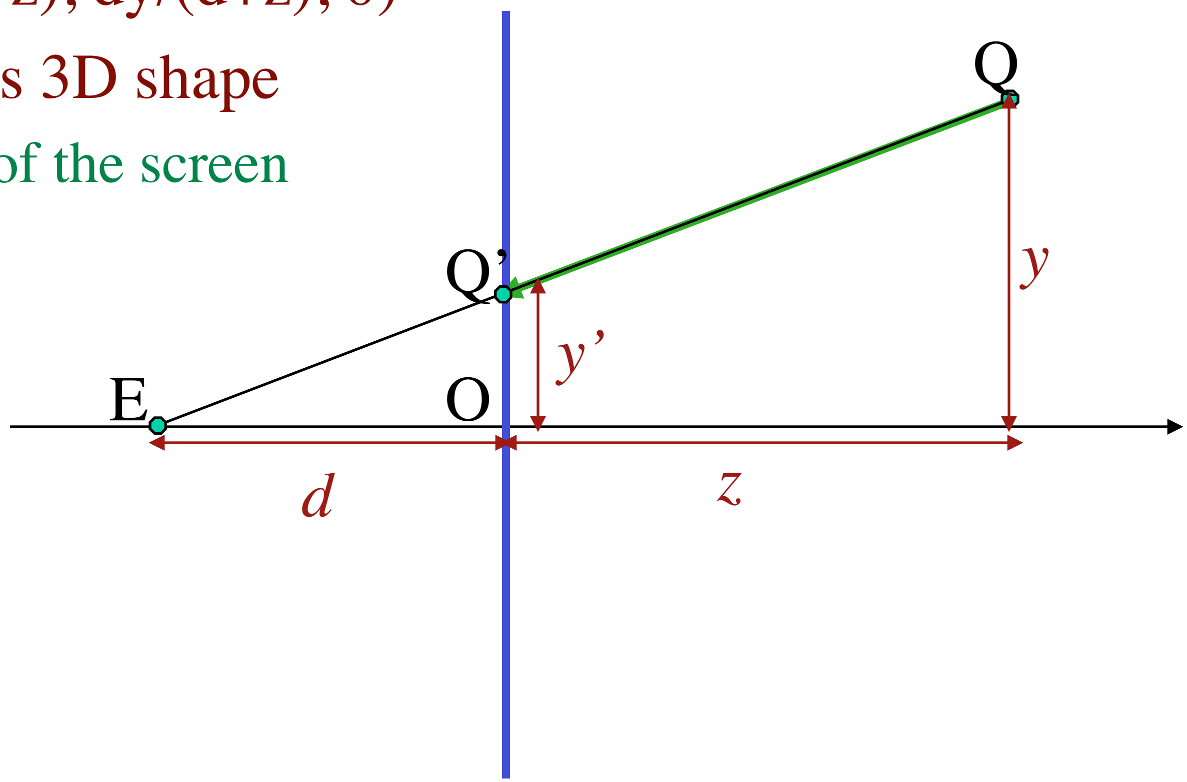
Hence also ||EO||=BO•OC and ||EO||=CO•OA

Thus O is the **orthocenter** of ABC

Intersection of the altitudes

Perspective projection equation

- **M is a projection if M(M(x))=M(x)**
- $y'/d = y/(z+d) \rightarrow y' = dy/(d+z)$
- $(x,y,z) \rightarrow (dx/(d+z), dy/(d+z), 0)$
- Projection flattens 3D shape
  - into a 2D area of the screen

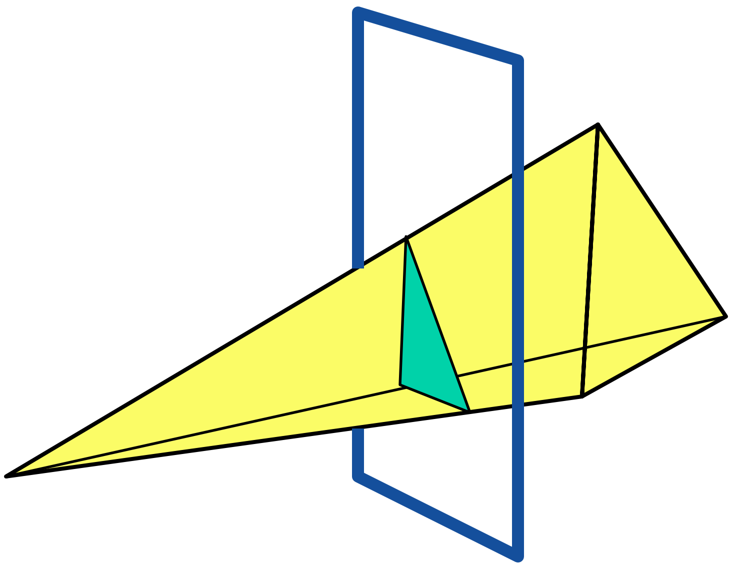


The perspective projection of a triangle is…

- Is it a triangle? If so, why?

The rays from E that hit a triangle T form a tetrahedron

If the screen separates E and T, the intersection of that tetrahedron with the screen is a triangle.



Why not use perspective **projection**?

- You lose the depth information
  - Can’t compute the z for each pixel
  - Can’t establish visibility using z-buffer

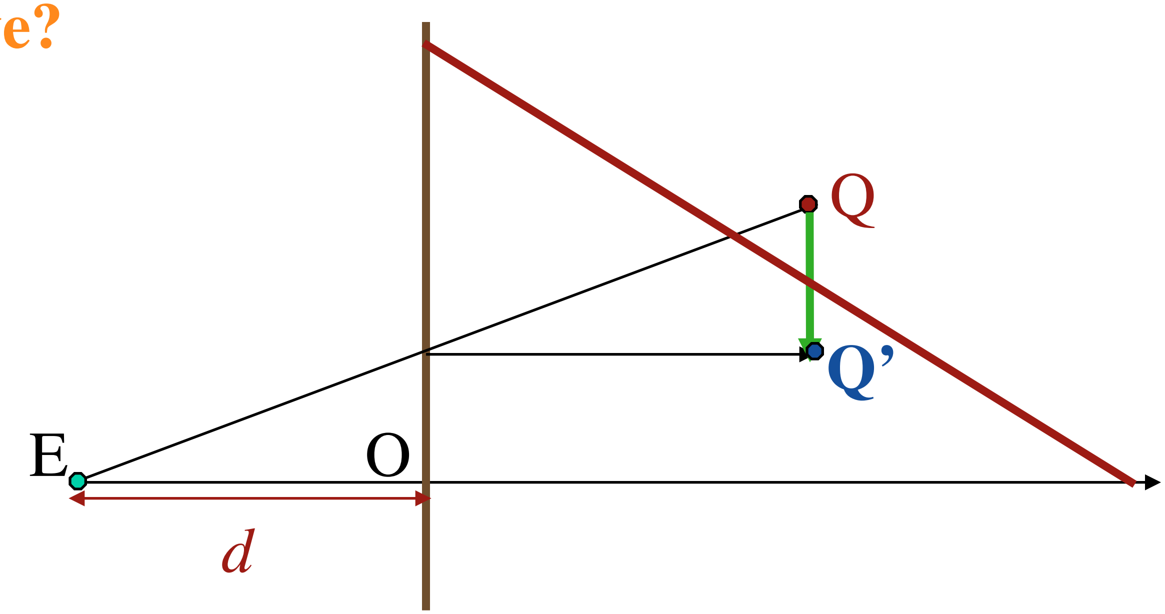
Can we leave z alone?

Can we use (x,y,z) → (dx/(d+z), dy/(d+z), z) for scan-conversion?

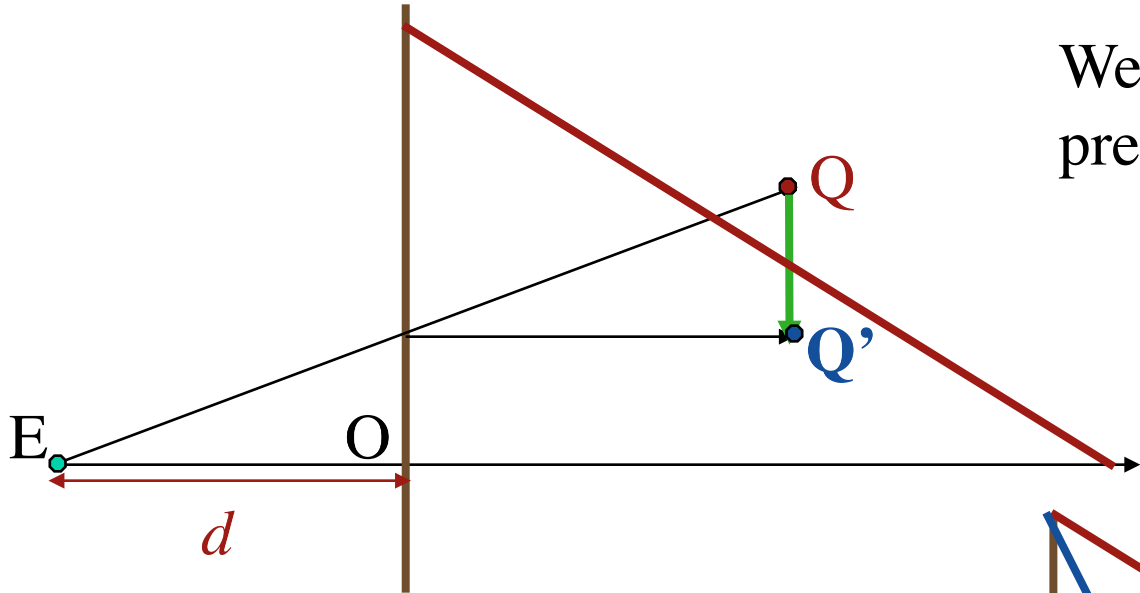
Sure: If a point Q is behind a point P, then Qz>Pz. So, this transformation preserves depth order.

So why don't we?

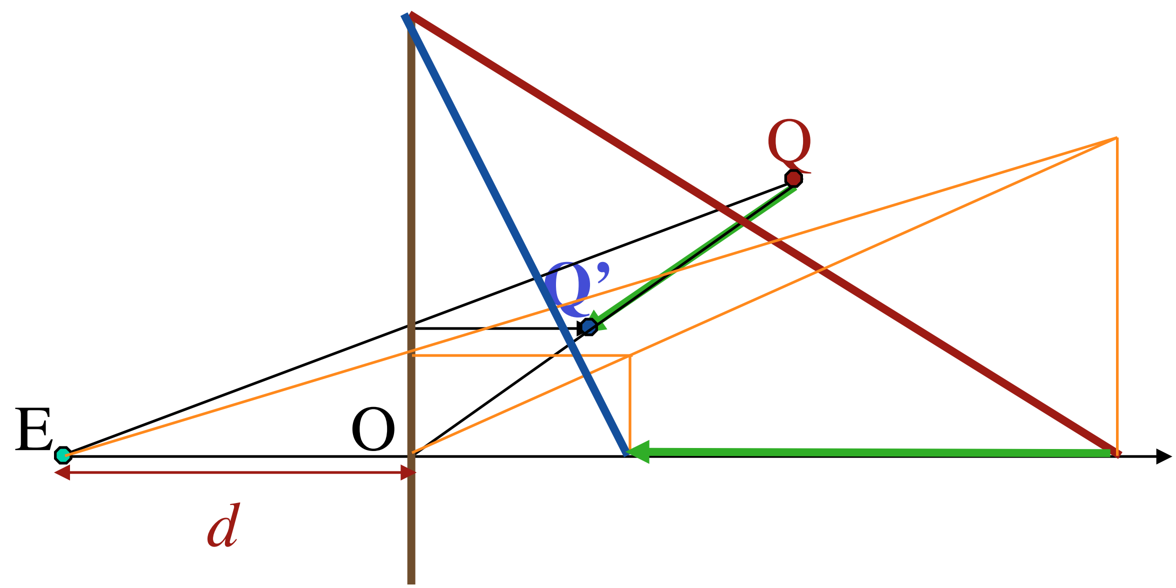
We would not always preserve depth order… when combined with linear depth interpolation



Why we can't leave z unchanged?



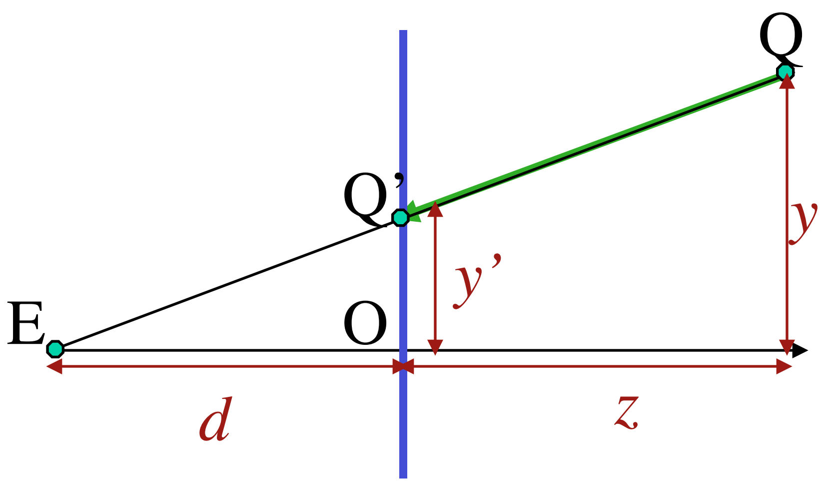
We would not always preserve depth order!



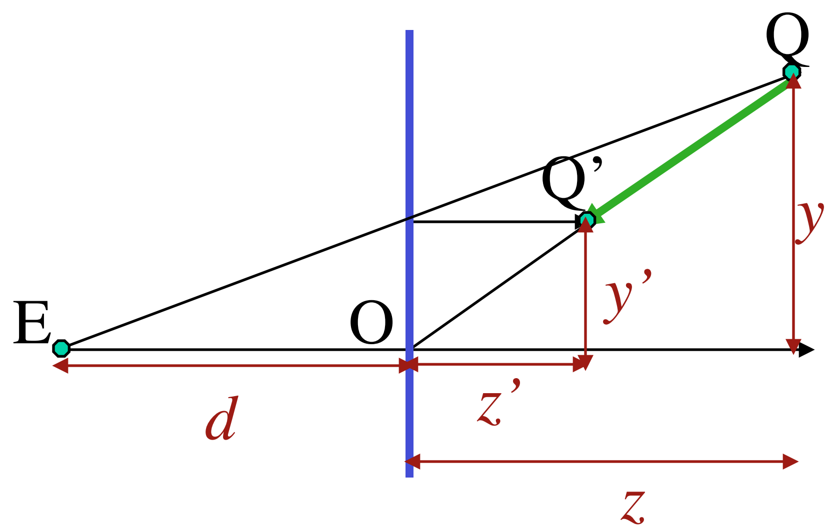
Perspective transformation

- A perspective transform maps a 3D shape into a 3D shape
- **Homogeneous** xform $(x,y,z) \rightarrow (dx/(d+z), dy/(d+z), dz/(d+z))$
- Really a **scaling**: $(x',y',z') = (d/(d+z))\,(x, y, z)$

Perspective projection



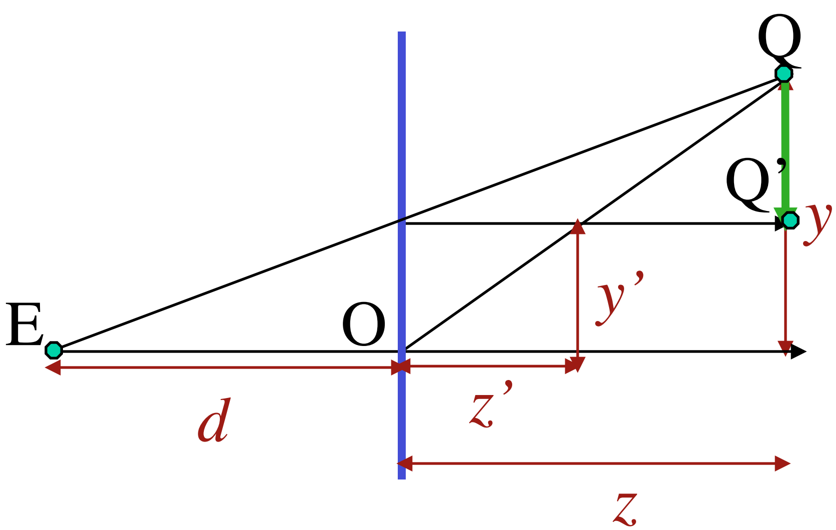
Perspective transform



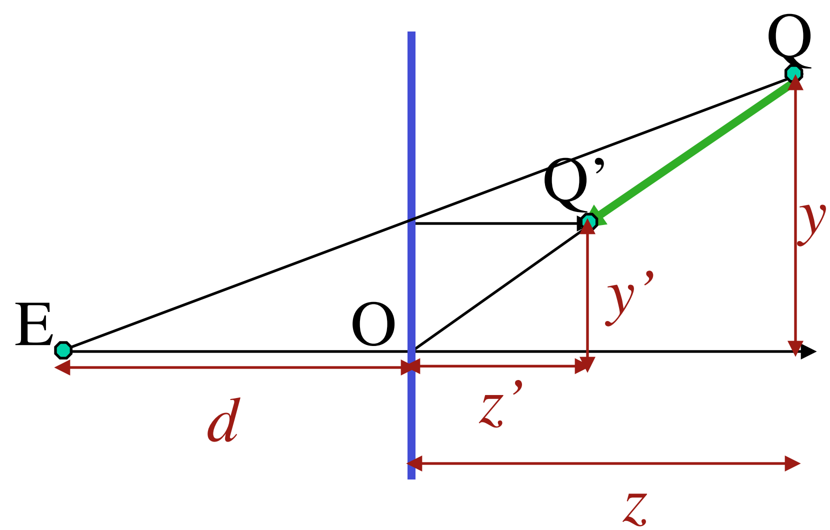
Comparison

(x,y,z) → (dx/(d+z), dy/(d+y), z) is not an option …when combined with a rasterizer that linearly interpolates transformed vertices. Hence, we need the homogeneous transform that maps triangles into triangles

Wrong Perspective <u>transform</u>



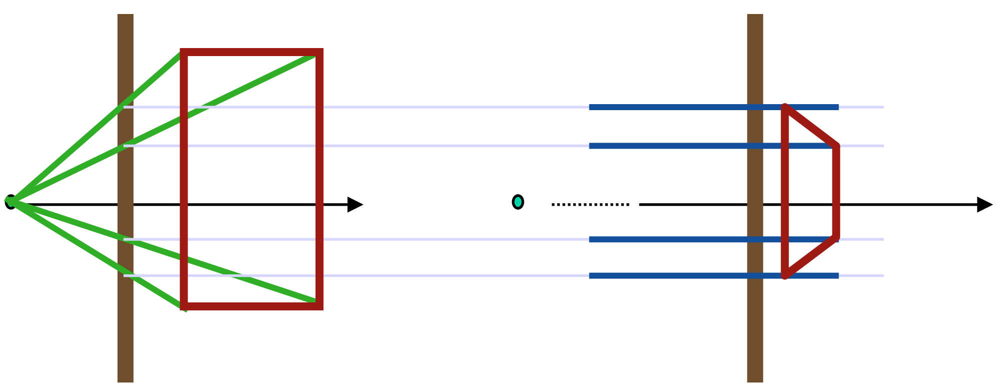
Correct Perspective <u>transform</u>

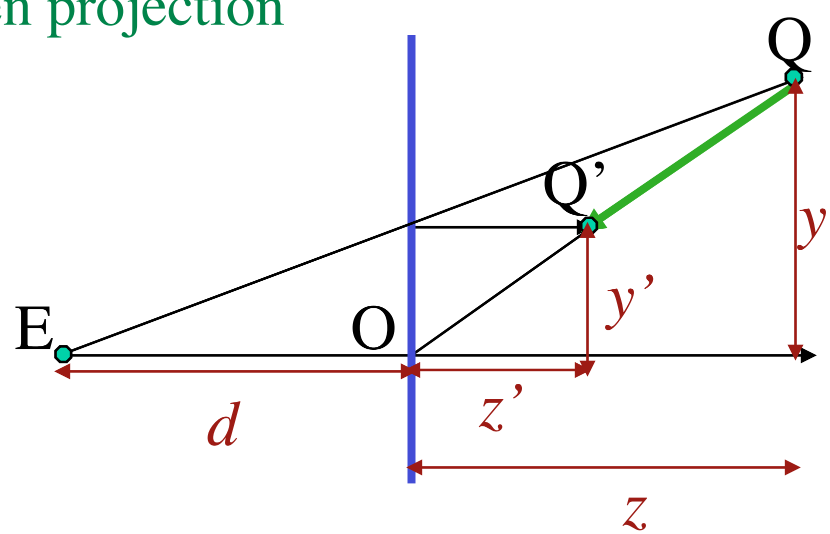


Geometric construction

- Q' = dQ/(d+z): Scaling towards O by d/(d+z)
- Scaling sets the (x,y) coordinates of Q' where the ray from E to Q hits the screen
  - Identifies 2D coordinatges for scan-conversion
- Moves E to (0,0, – ∞)
  - Parallel projection
  - x' and y' are coordinates of screen projection

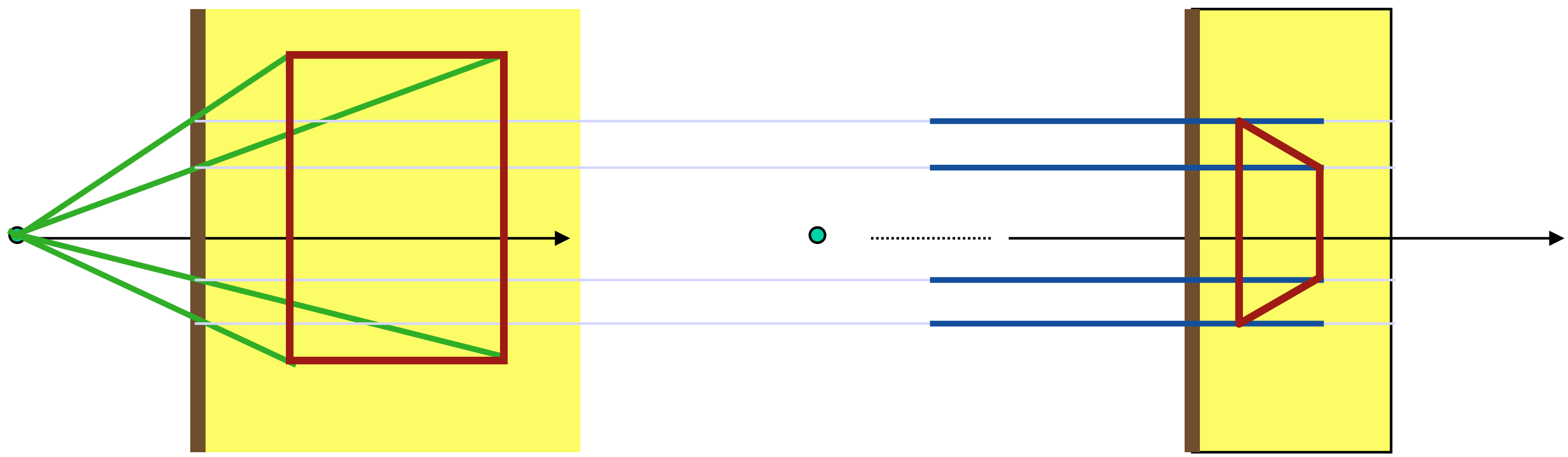
Perspective <u>transform</u>





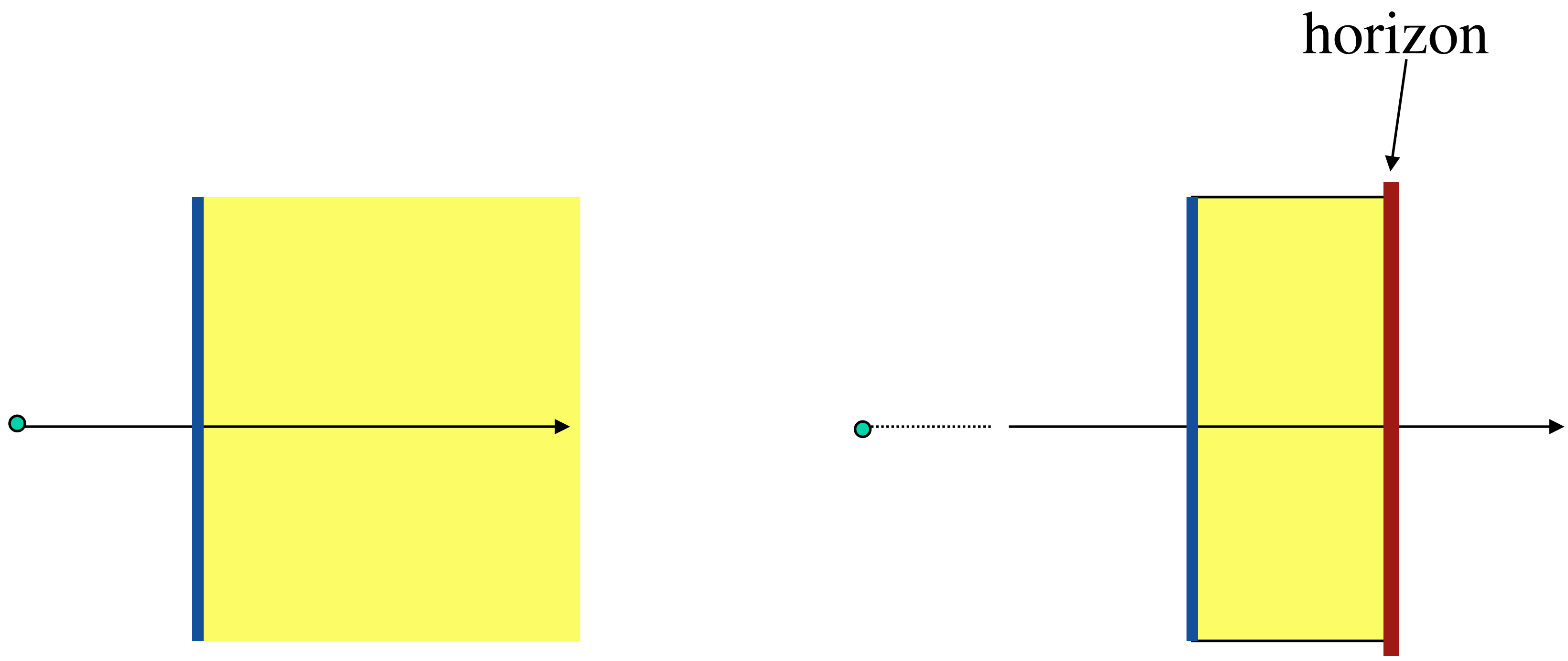
Perspective transform of {0<z}

- The perspective transform maps the whole space behind the screen to? 0<z<d



The horizon plane

- Plane z=d in the transformed space
- That is where all vanishing points live



Perspective transform of {n<z<f}

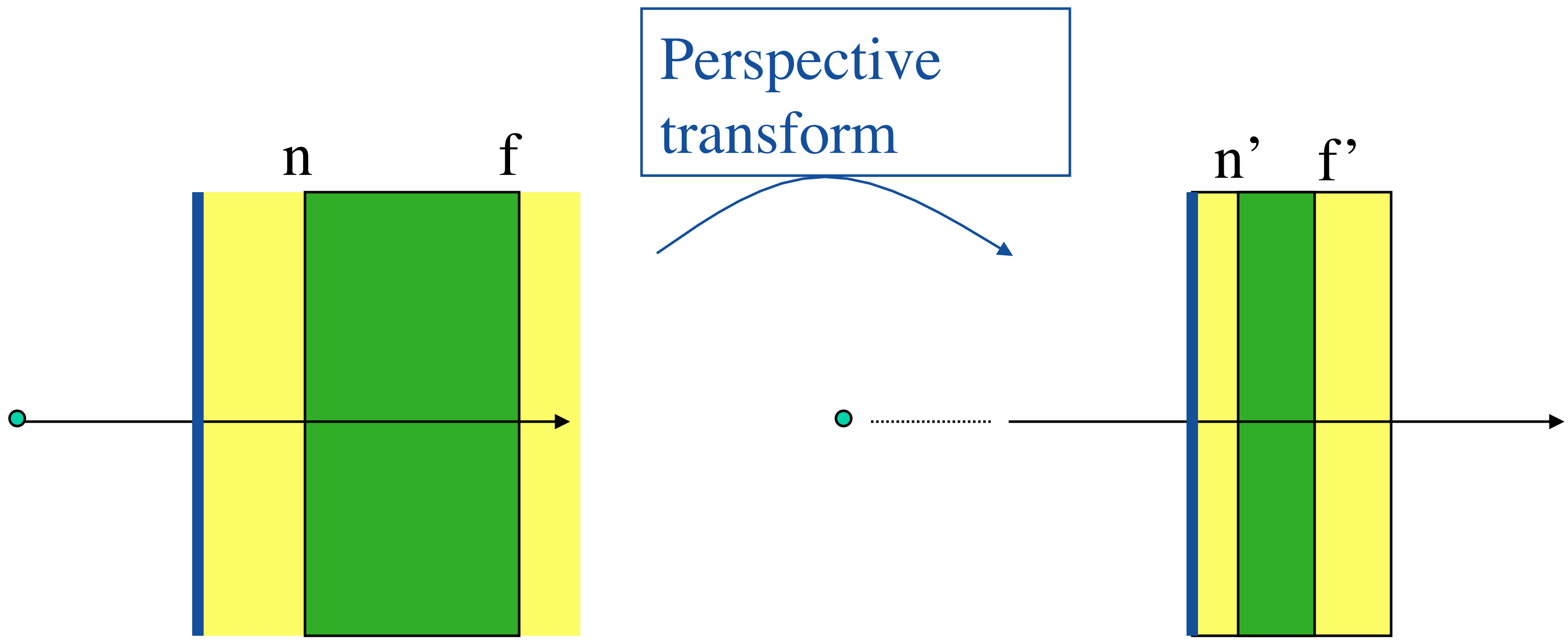
- Select z-buffer values to match near and far clipping planes

n'=dn/(d+n) should be 0

z'=zd/(d+z)

f'=df/(d+f) should be $2^{24} - 1$

Z[x,y]:=(z'–n') 2^{24}/ (f'–n')



Standard matrix form of perspective

- Origin at eye, not screen: change z to z–d

  (x’,y’,z’) = (dx/(d+z), dy/(d+z), dz/(d+z))

  (x’,y’,z’) = (dx/z, dy/z, d(z–d)/z)

- Scale to the near-far interval (transformed) and to window

  n’= d(n–d)/n, f’= d(f–d)/f.

  z”=(z’–n’)/(f’–n’)=(d(z–d)/z–d(n–d)/n)/(d(f–d)/f–d(n–d)/n)

  = f(z–n)/((f–n)z) if 0≤z”≤1 as n≤z≤f

  = (z(f+n)–2fn)/(f–n) if -1≤z”≤1 as n≤z≤f

$$\begin{bmatrix} x'w \\ y'w \\ z'w \\ w \end{bmatrix} = \begin{bmatrix} \frac{2 \cdot \text{near}}{\text{right} - \text{left}} & 0 & \frac{-(\text{right} + \text{left})}{\text{right} - \text{left}} & 0 \\ 0 & \frac{2 \cdot \text{near}}{\text{bottom} - \text{top}} & \frac{-(\text{bottom} + \text{top})}{\text{bottom} - \text{top}} & 0 \\ 0 & 0 & \frac{\text{far} + \text{near}}{\text{far} - \text{near}} & \frac{-2 \cdot \text{near} \cdot \text{far}}{\text{far} - \text{near}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The pre-image of $\{z'=k\}$

- Use the inverse of the perspective transform

$z'=k$

Replace z' by dz/(d+z) in the equation

$dz/(d+z)=k$

Solve for z

$dz=kd+kz$

$z=kd/(d-k)$

For example, $k=d$ yields z=infinity
For example, $k=d/2$ yields $z=d$
For example, $k=d/4$ yields $z=d/3$

The inverse of a perspective transform

- $Q' = dQ/(d+z)$
- $Q = ?$

$$Q = dQ'/(d-z')$$

Maps planes to planes

- Prove that perspective transform maps planes into planes

Define set of points (x',y',z') with constraint ax+by+cz+h=0

Use **Inverse** of perspective transform to rewrite constraint

ax'd/(d-z')+by'd/(d-z')+cz'd/(d-z')+h=0

Multiplying by (d-z')/d yields ax'+by'+cz'+(d-z')h/d=0

Which is a plane: ax'+by'+(c-h/d)z'+h=0

Half-spaces: ax'+by'+cz'+h<0 maps to ax'+by'+(c-h/d)z'+h<0

Map triangles onto triangles

Triangle is the intersection of a plane with 3 linear half-spaces.
Planes form a closed set under perspective transform.

Practice exercises

- Consider an object that spans the depth interval [f,n] in screen coordinates before perspective transformation
  - What interval does it span after perspective transformation?
- Assume that z is stored (without shift or scaling) in a z-buffer as 24-bit integers
  - What is the range of maximum round-off errors between the actual z of a point before perspective transformation and the z that could be recovered from the rounded value stored in the z-buffer?
- How much accuracy improvement does one get by shifting the z-buffer values so that the z of points (x,y,n) is mapped to 0?
- How much additional accuracy improvements does one get by also scaling the z-buffer so that the z points (x,y,f) is mapped to the maximum value stored in the z-buffer?
- Does perspective transform map quadrics to quadrics?

Advanced questions on perspective

- Why not use the perspective projection?
- Why not leave z unchanged?
- How to draw the result of perspective transform?
- Where is the half-space z>0 mapped by perspective transform?
- Where are points at infinity mapped to? (Horizon)
- How is depth represented in the z-buffer?
- How is this encoded as a 4x4 matrix?
- What is the preimage of plane (z'=k)?
- What is the inverse of the perspective transform?
- Proof that perspective transform maps planes to planes?
- Proof that perspective transform maps triangles to triangles?
- Where should the viewer stand (given 3 vanishing points)?
- How to add shadows without knowing where the light is?

Additional practice question

- Show the computation and the geometric construction of the result of transforming point (d,d,d) by a perspective transform for a viewpoint at distance d from the screen

- Explain where the ray from point P with direction T is mapped by such a perspective transform

Adding shadows

You are given the picture of two vertical poles with their shadows on a flat floor.

You add a third pole. How to draw its correct shadow?