Computer Graphics

- Camera Transformations -

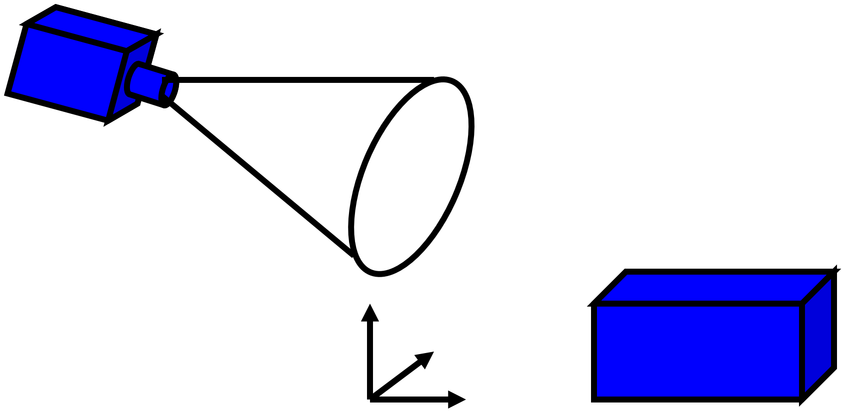
Philipp Slusallek

Overview

- **Last lecture:**
  - Subdivision Surfaces
- **Today:**
  - Generating 2D image from 3D world
    - Coordinate Spaces
    - Camera Specification
    - Perspective transformation
    - Normalized screen coordinates

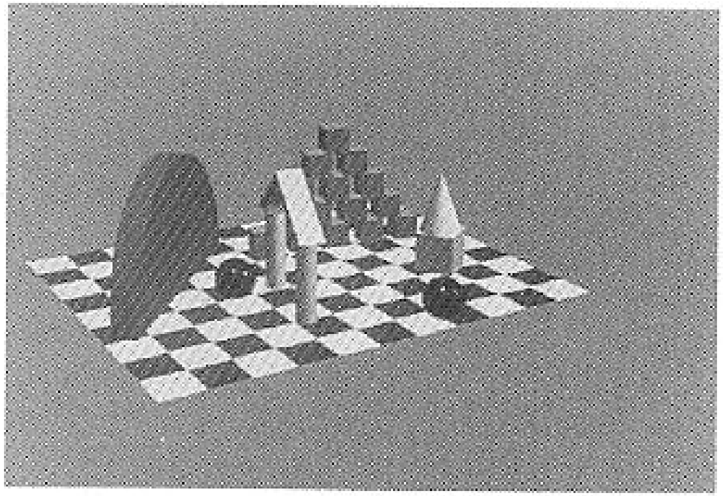
Camera Transformations

- **Goal**
  - Compute the transformation between points in 3D and pixels on the screen
  - Required for rasterization algorithms (OpenGL)
    - They project all primitives from 3D to 2D
    - Rasterization happens in 2D (actually 2-1/2D)
- **Given**
  - Camera description
  - Pixel raster description

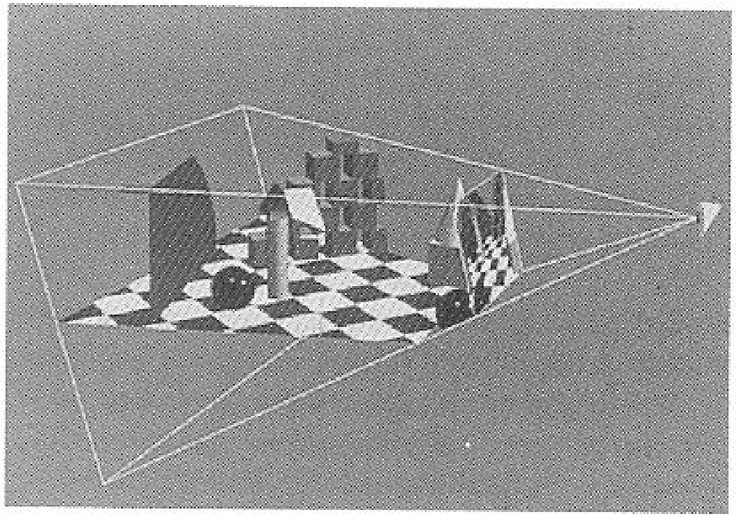


Camera Transformations

- **Model transformation**
  - Object space to world space



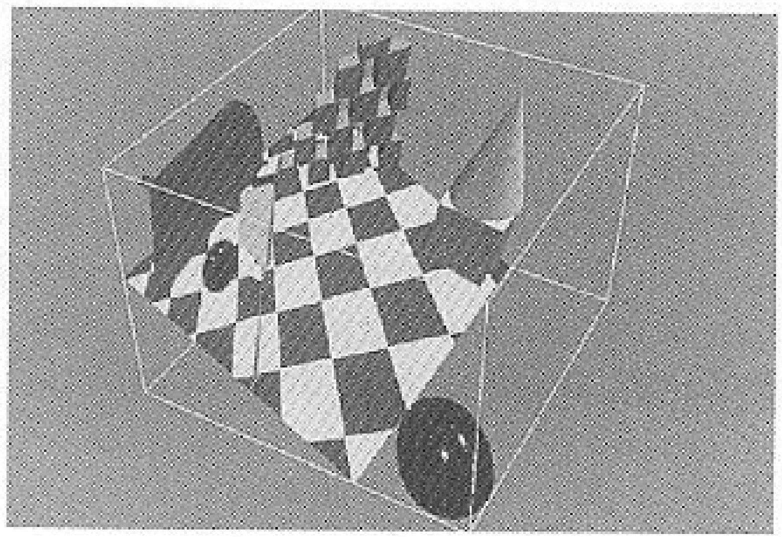
- **View transformation**
  - World space to eye space



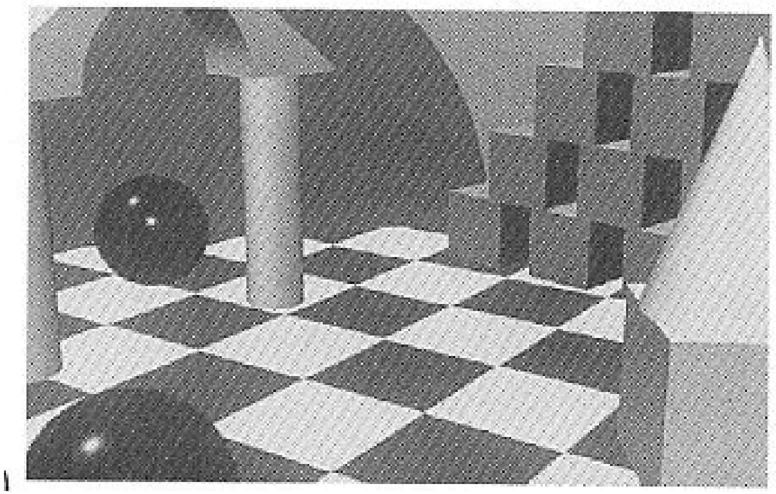
- **Combination: Modelview transformation**
  - Used by OpenGL

Camera Transformation

- **Projection transformation**
  - Eye space to normalized device space
  - Parallel or perspective projection



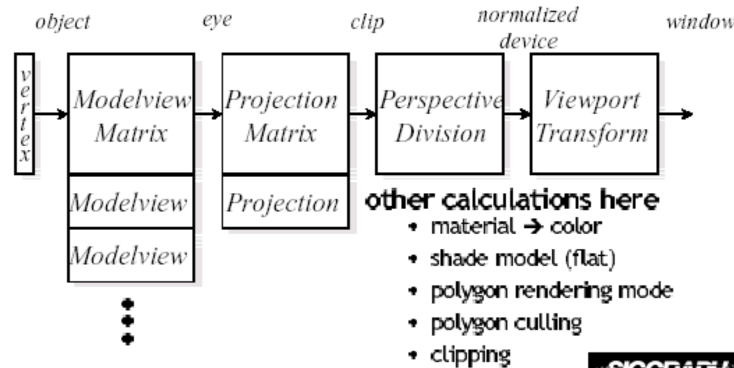
- **Viewport transformation**
  - Normalized device space to window (raster) coordinates

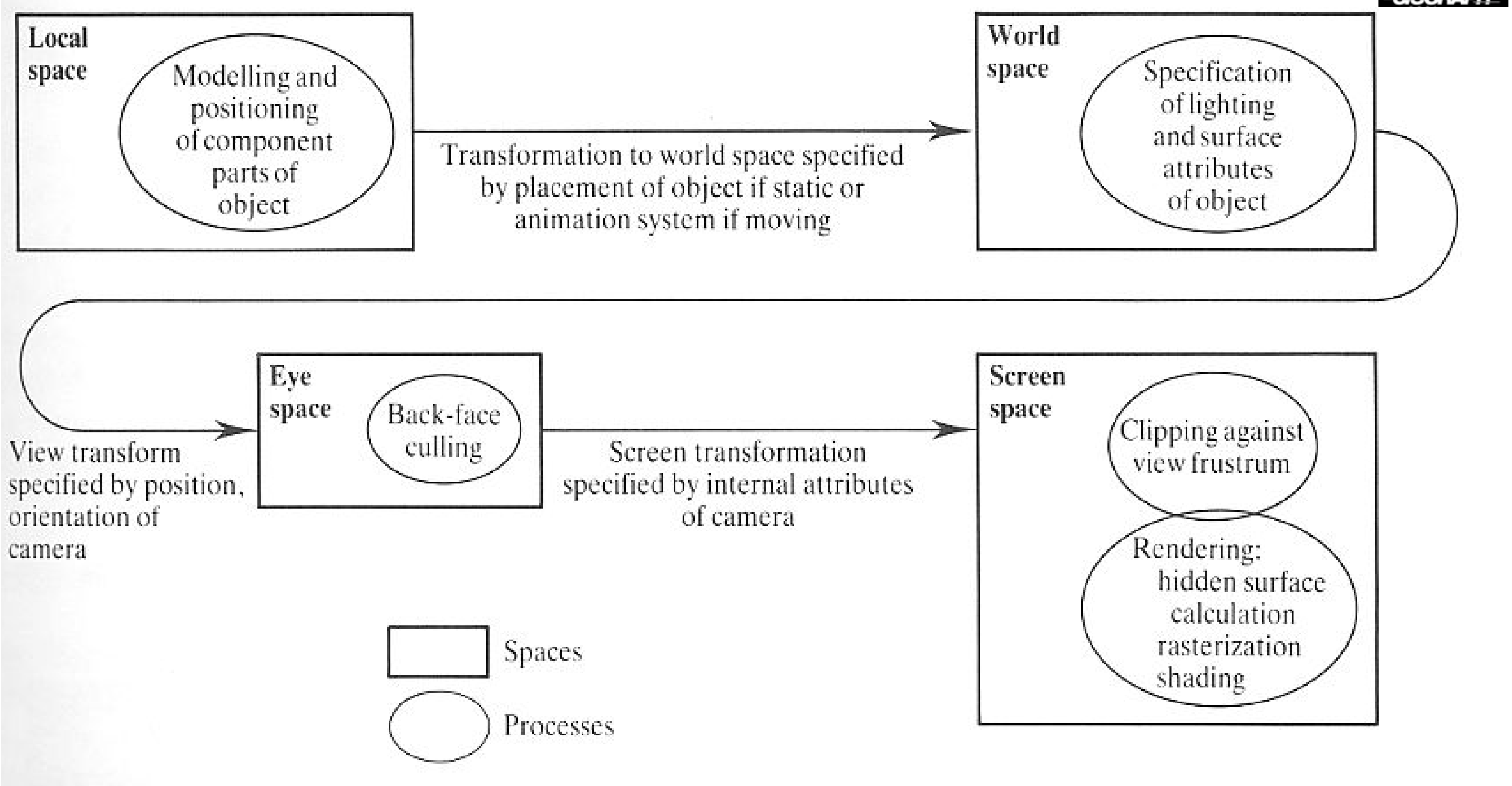


Coordinate Transformations

- **Local (object) coordinate system (3D)**
  - Object vertex positions
- **World (global) coordinate system (3D)**
  - Scene composition and object placement
    - Rigid objects: constant translation, rotation per object
    - Animated objects: time-varying transformation in world-space
  - Illumination
- **Camera/View/Eye coordinate system (3D)**
  - Camera position & direction specified in world coordinates
  - Illumination & shading can also be computed here
- **Normalized device coordinate system (2-1/2D)**
  - Normalization to viewing frustum
  - Rasterization
  - Shading is executed here (but computed in world or camera space)
- **Window/Screen (raster) coordinate system (2D)**
  - 3D to 2D transformation: projection

Per-Vertex Transformations





Viewing Transformation

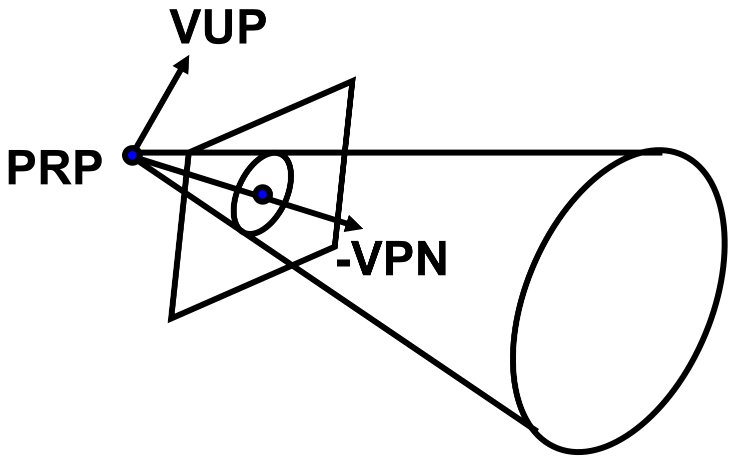
- **Camera position and orientation in world coordinates**
  - Center of projection, projection reference point (PRP)
  - Optical axis, view plane normal (VPN)
  - View up vector (VUP) (not necessarily perpendicular to VPN)

  ⇒ *External (extrinsic) camera parameters*
- **Transformation**

  1.) Translation of all vertex positions by projection center

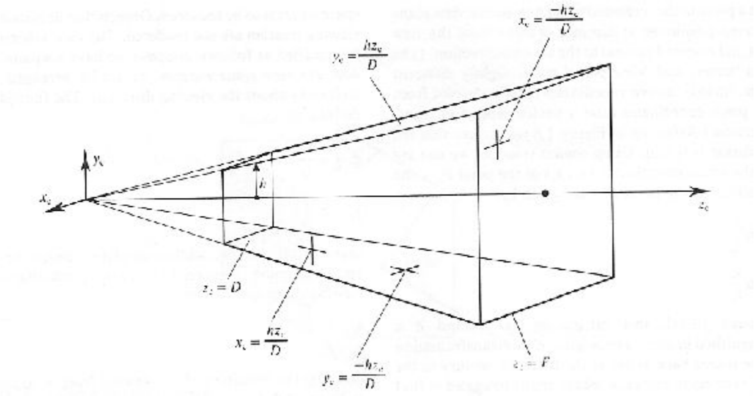
  2.) Rotation of all vertex position by camera orientation

  convention: view direction along Z axis



Perspective Transformation

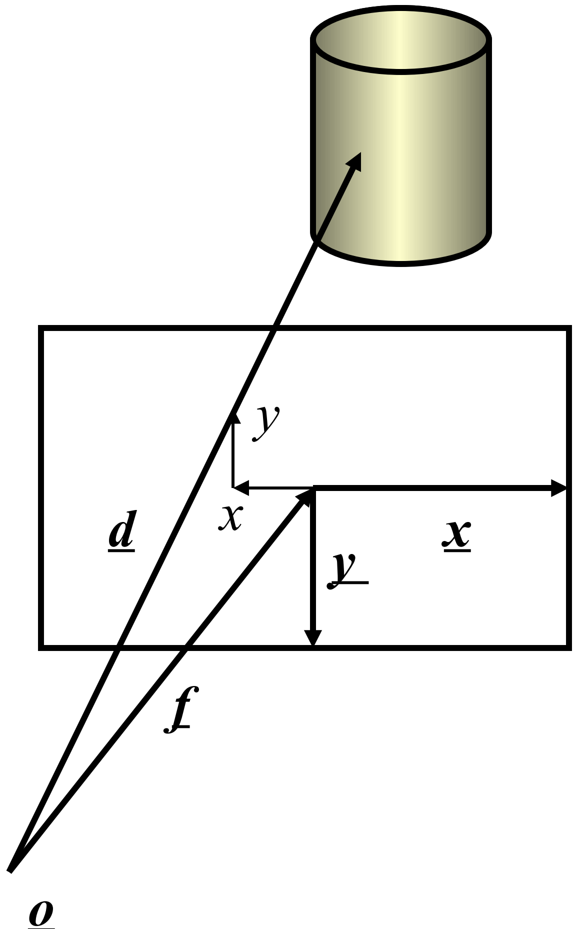
- **Camera coordinates to screen coordinate system**
  - ⇒ *Internal (intrinsic) camera parameters*
  - – Field of view (fov)
    - Distance of image plane from origin (focal length) or field of view (angle)
  - – Screen window
    - Window size on image plane
    - Also determines viewing direction (relative to view plane normal)
  - – Near and far clipping planes
    - Avoids singularity at origin (near clipping plane)
    - Restriction of dynamic depth range (near&far clipping plane)
    - Together define „View Frustum“
  - – Projection (perspective or orthographic)
  - – Mapping to raster coordinates
    - Resolution
    - Adjustment of aspect ratio



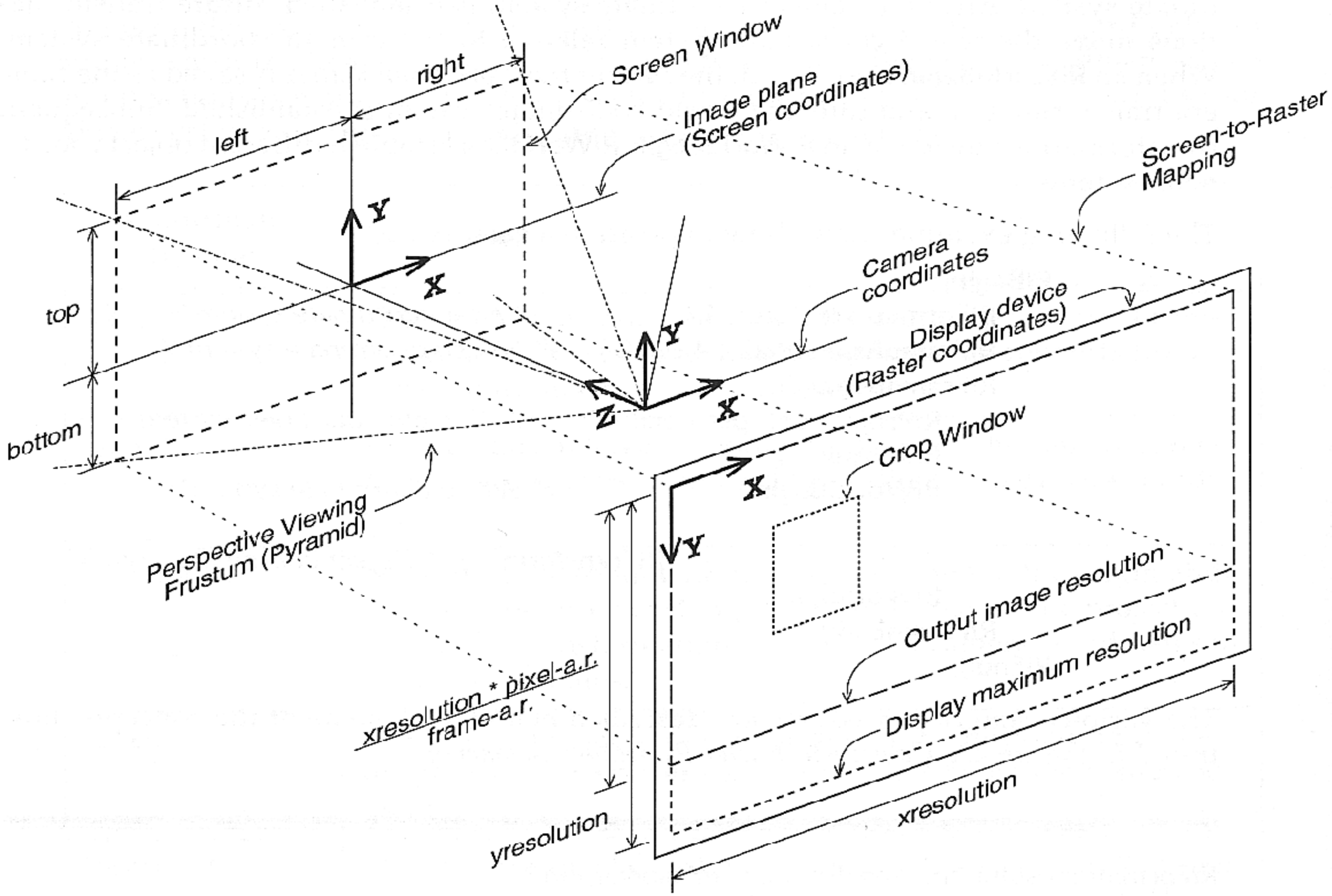
View Frustum

Camera Parameters: Simple

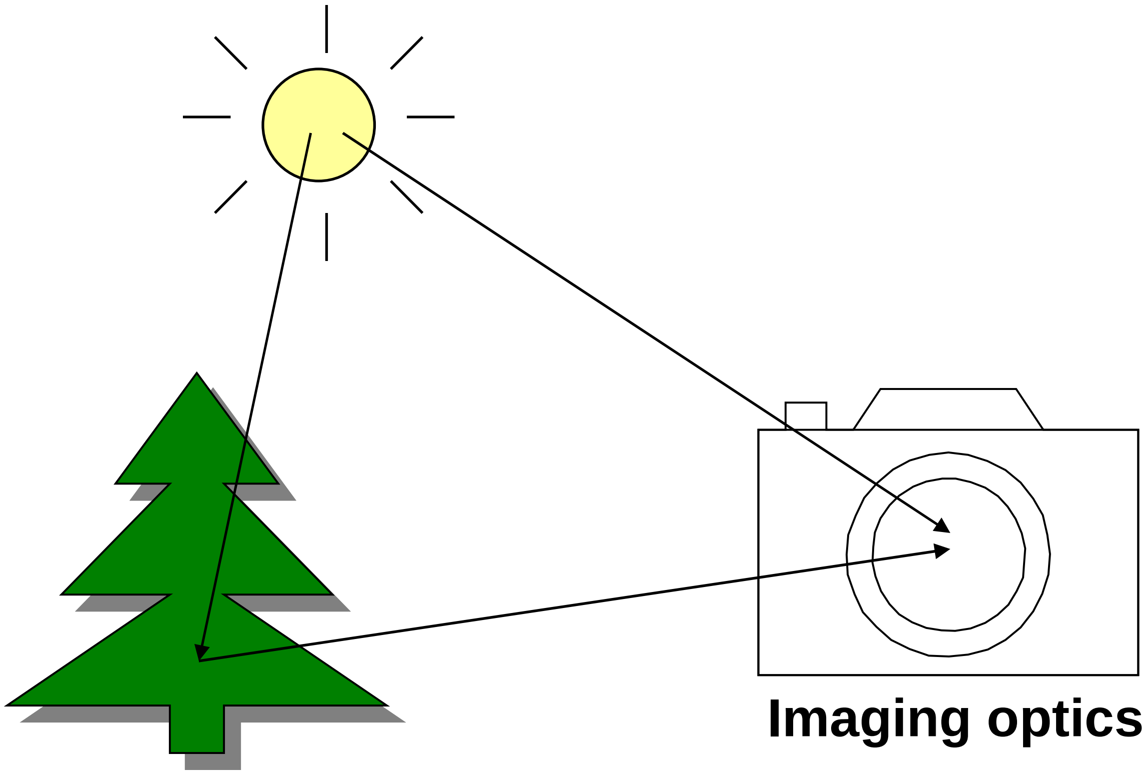
- **Camera definition in ray tracer**
  - $\underline{\boldsymbol{o}}$: center of projection, point of view
  - $\underline{\boldsymbol{f}}$: vector to center of view, optical axis
  - $\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}$: span of half viewing window
  - $xres, yres$: image resolution
  - x, y : screen coordinates



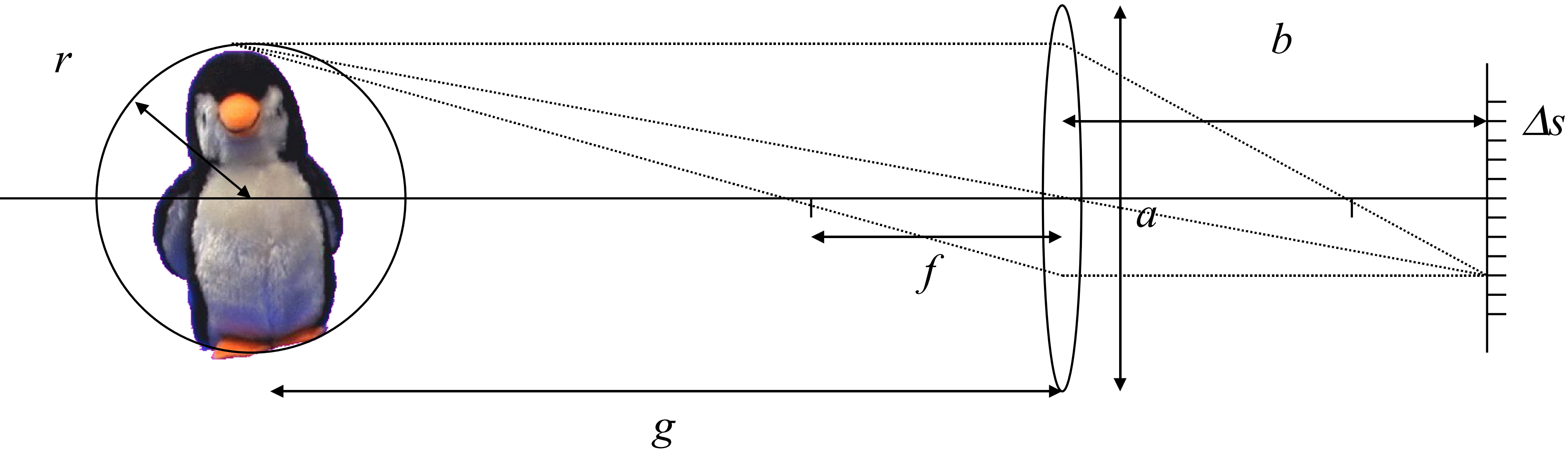
Camera Parameters: RMan



Camera Model



Lens Camera

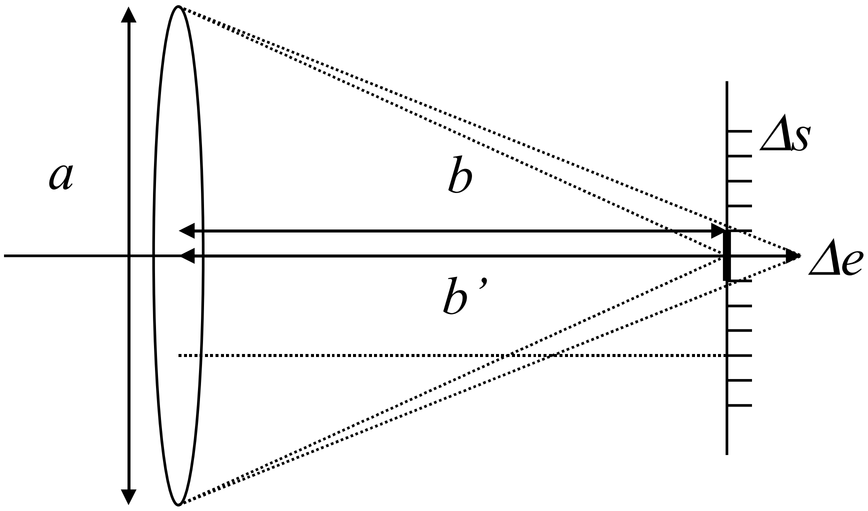


Lens Formula $\frac{1}{f} = \frac{1}{b} + \frac{1}{g}$

Object center in focus $b = \frac{f\,g}{g - f}$

Object front in focus $b' = \frac{f\,(g - r)}{(g - r) - f}$

Lens Camera: Depth of Field



Circle of Confusion $\Delta e = \left| a \left(1 - \frac{b}{b'} \right) \right|$

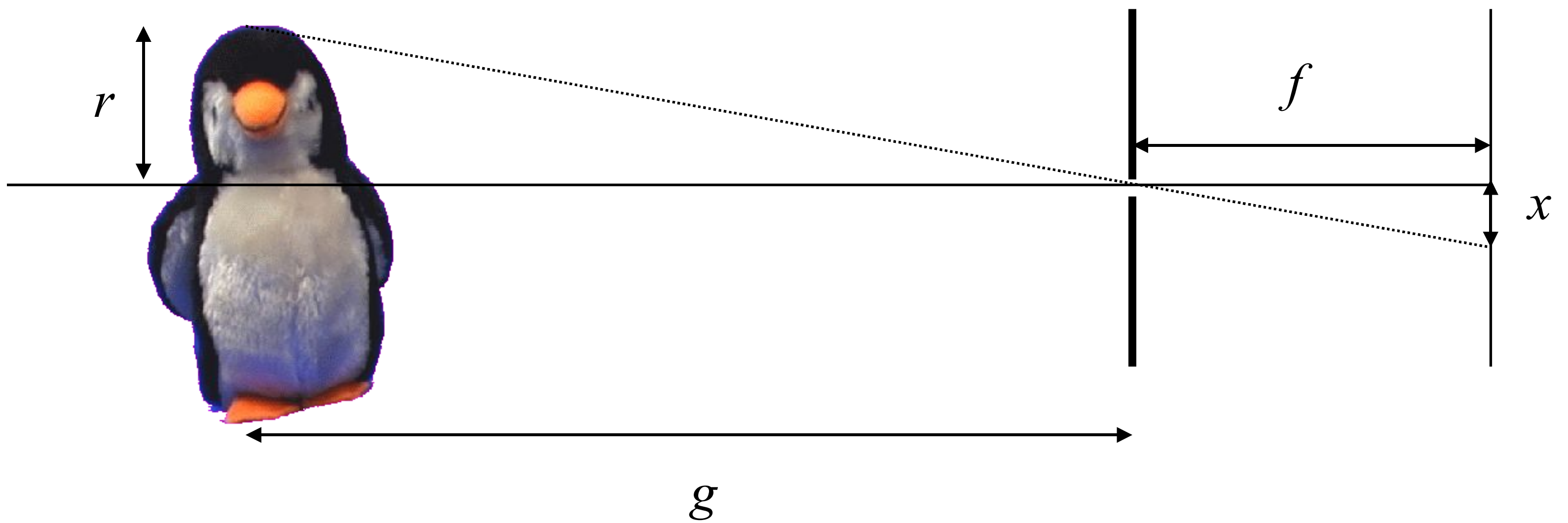
Sharpness Criterion $\Delta s > \Delta e$

Depth of Field (DOF) $r < \frac{g \, \Delta s (g - f)}{af + \Delta s (g - f)}$ $\Rightarrow$ $r \propto \frac{1}{a}$

DOF: Defined as length of interval (b') with CoC smaller than Δs

The smaller the aperture, the larger the depth of field

Pinhole Camera Model



$$\frac{r}{g} = \frac{x}{f} \quad \Rightarrow \quad x = \frac{f\,r}{g}$$

Pinhole small

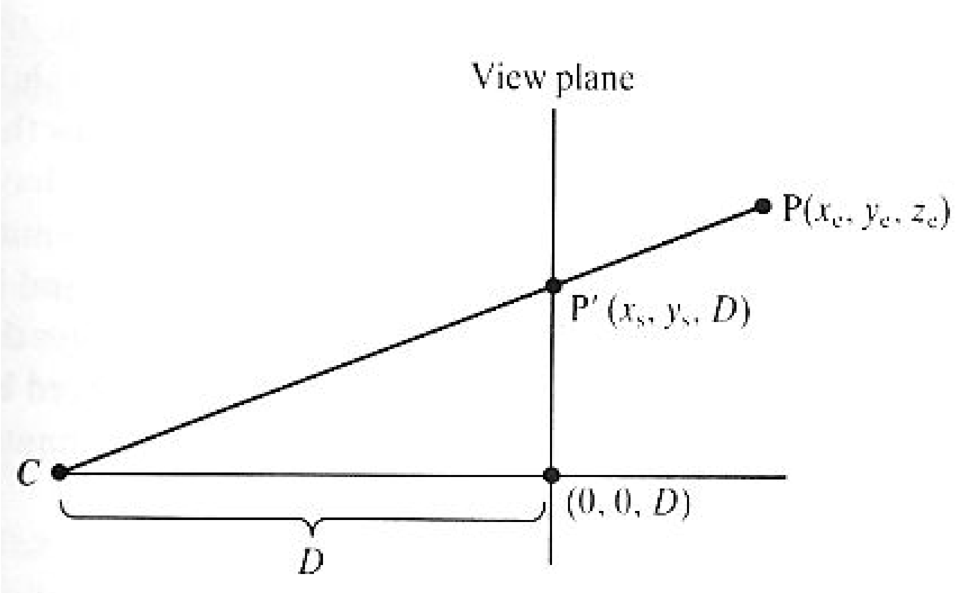
- ⇒ image sharp
- ⇒ infinite depth of field
- ⇒ image dark
- ⇒ diffraction effects

Perspective Transformation

- **3D to 2D projection**
  - Point in eye coordinates: P(x_e, y_e, z_e)
  - Distance: center of projection to image plane: D
  - Image coordinates: (x_s, y_s)

$$x_s = D\frac{x_e}{z_e}$$

$$y_s = D\frac{y_e}{z_e}$$



Transformations

- **Homogeneous coordinates (reminder :-)**

$$R^3 \ni \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in \mathrm{P}(\mathrm{R}^4), \quad \text{and} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} \rightarrow \begin{pmatrix} X/W \\ Y/W \\ Z/W \end{pmatrix}$$

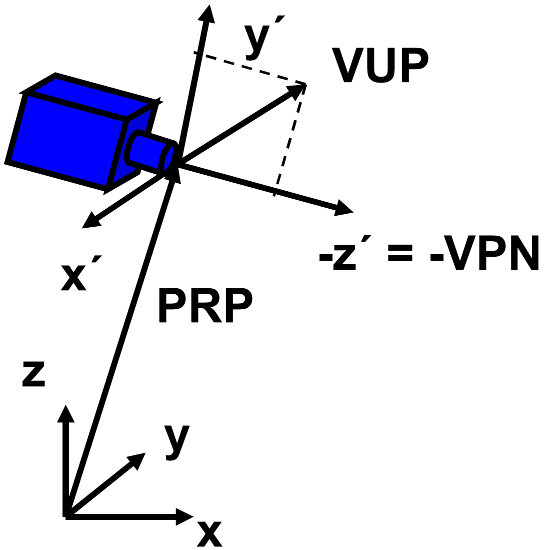
- **Transformations**
  - 4x4 matrices
  - Concatenation of transformations by matrix multiplication

$$T(d_x, d_y, d_z) = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R(\alpha, \beta, \gamma) = \begin{pmatrix} r_{00} & r_{01} & r_{02} & 0 \\ r_{10} & r_{11} & r_{12} & 0 \\ r_{20} & r_{21} & r_{22} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Viewing Transformation

- **Goal:**
  - Camera: at origin, view along –Z, Y upwards (right hand)
  - Translation of PRP to the origin
  - Rotation of VPN to Z-axis
  - Rotation of projection of VUP to Y-axis
- **Rotations**
  - Build orthonormal basis for the camera and form inverse
    - Z´= VPN, X´= normalize(VUP x VPN), Y´= Z´ × X´
- **Viewing transformation**

$$V = RT = \begin{pmatrix} X'_x & Y'_x & Z'_x & 0 \\ X'_y & Y'_y & Z'_y & 0 \\ X'_z & Y'_z & Z'_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^T T(-PRP)$$

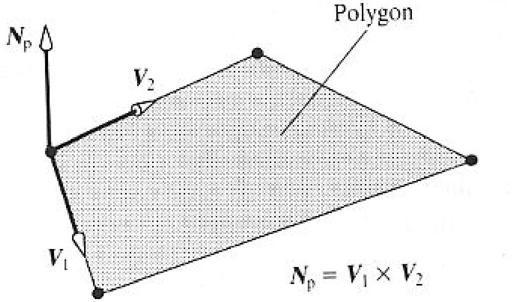


Backface Culling

- **Polygon normal in world coordinates**

$$N_P = V_1 \times V_2$$

Oriented polygon edges V_1, V_2

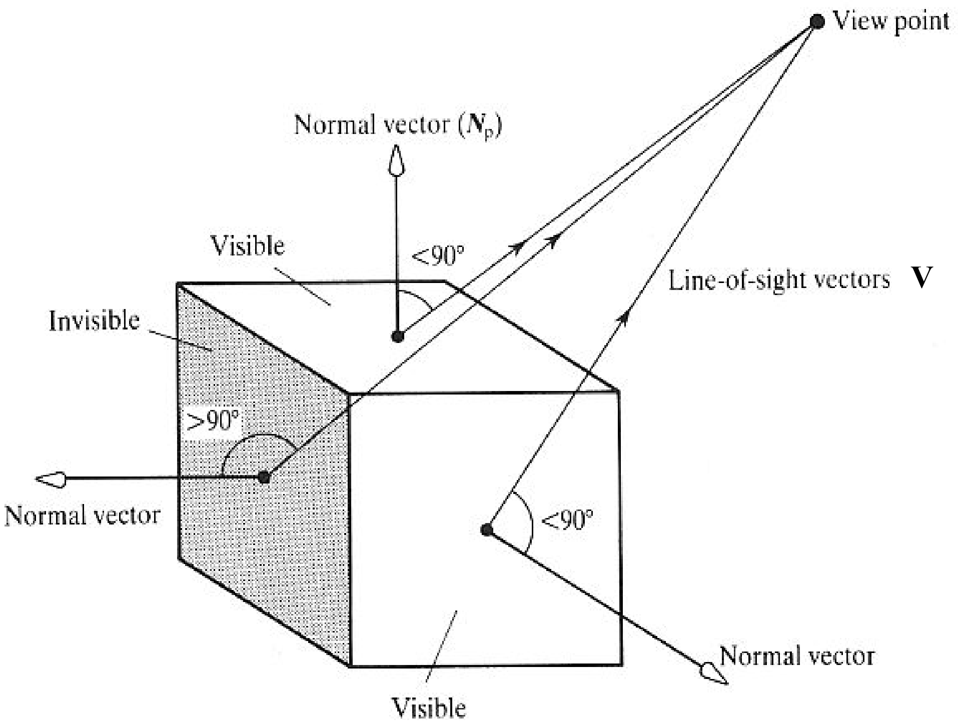


- **Line-of-sight vector V**
  - Dot product

$$N_P \cdot V$$

> 0 : surface visible
< 0 : surface not visible

⇒ Draw only visible surfaces
⇒ Applicable to closed objects only
⇒ Do *not* use shading normal



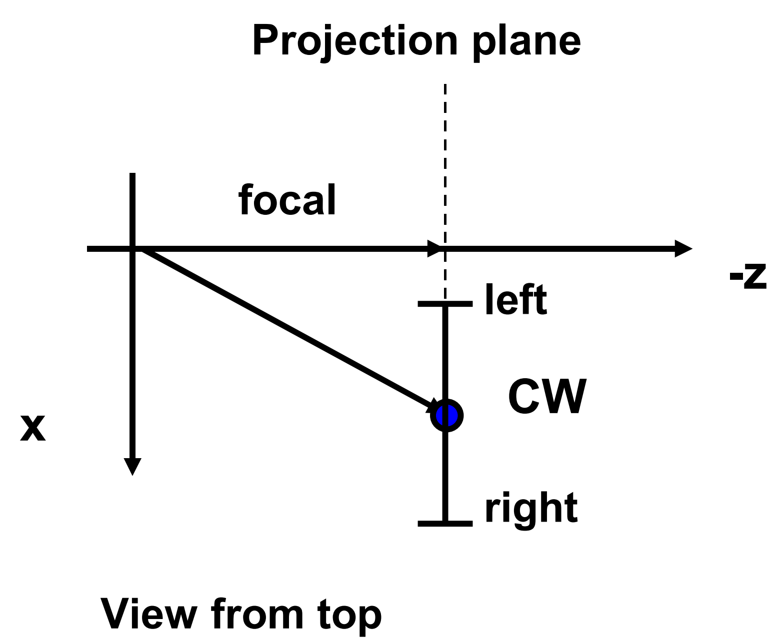
Perspective Transformation

- **Step 1: Optical axis may not go through screen center**
  – Oblique viewing configuration

⇒ **Shear (Scherung)**
  – Shear such that viewing direction is along Z-axis
  – Window center CW (in 3D view coordinates)
    - $CW = ((right+left)/2, (top+bottom)/2, -focal)^T$

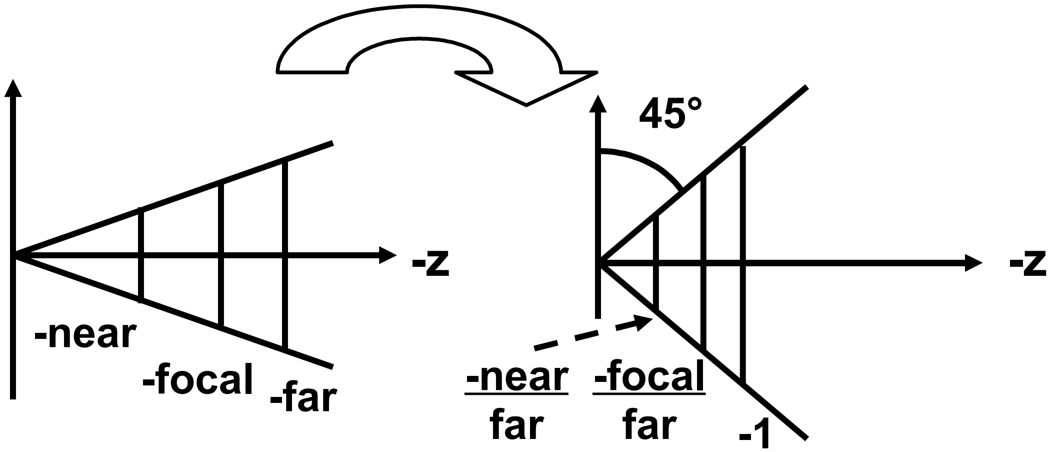
- **Shear matrix**

$$H = \begin{pmatrix} 1 & 0 & -\frac{CW_x}{CW_z} & 0 \\ 0 & 1 & -\frac{CW_y}{CW_z} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Normalizing

- **Step 2: Scaling to canonical viewing frustum**
  - Scale in X and Y such that screen window boundaries open at 45 degree angles
  - Scale in Z such that far clipping plane is at Z= -1



- **Scaling matrix**

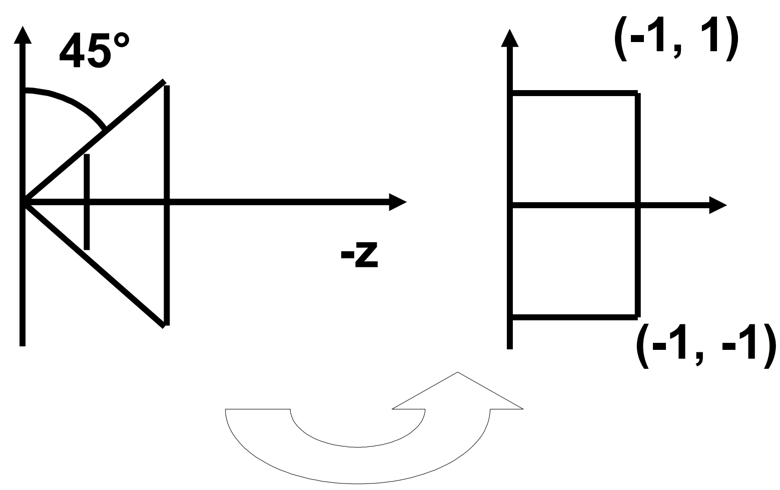
$$S = S_{far} S_{xy} = \begin{pmatrix} 1/far & 0 & 0 & 0 \\ 0 & 1/far & 0 & 0 \\ 0 & 0 & 1/far & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{2\,focal}{right - left} & 0 & 0 & 0 \\ 0 & \frac{2\,focal}{top - bottom} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Perspective Transformation

- **Step 3: Perspective Transformation**
  - From canonical perspective viewing frustum (= cone at origin around -Z-axis) to regular box $[-1 \,..\, 1]^2 \times [0 \,..\, 1]$
- **Mapping of X and Y**
  - Lines through the origin are mapped to lines parallel to the Z-axis
    - $x' = x/-z$ und $y' = y/-z$
- **Perspective Transformation**

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ A & B & C & D \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

unknown



- ***Perspective Projection* =**
  Perspective Transformation + Parallel Projection

Perspective Transformation

- **Computation of the coefficients**
  - No shear w.r.t. X and Y
    - A= B= 0
  - Mapping of two known points
    - Computation of the two remaining parameters C and D
    - n = near/far

$$(0,0,-1,1)^T = P(0,0,-1,1)^T$$

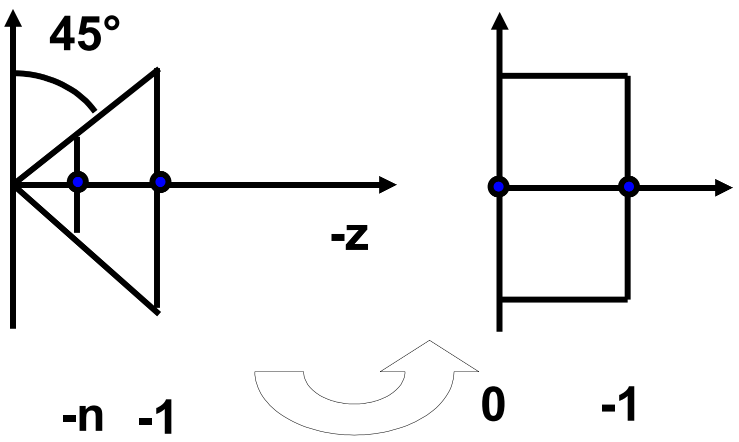
$$(0,0,0,1)^T = P(0,0,-n,1)^T$$

$$z' = -\left(\frac{z+n}{z(1-n)}\right)$$

Nonlinear transformation of z

- **Projective Transformation**

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1-n} & \frac{n}{1-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$



Parallel Projection to 2D

- **Parallel projection to $[-1 .. 1]^2$**
  - Scaling in Z with factor 0
- **Transformation from $[-1 .. 1]^2$ to $[0 .. 1]^2$**
  - Scaling (by 1/2 in X and Y) and translation (by (1/2,1/2))
- **Projection matrix for combined transformation**
  - Delivers normalized device coordinates

$$P_{parallel} = \begin{pmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Viewport Transformation

- **Scaling and translation in 2D**
  - Adjustment of aspect ratio
    - Size of screen/window
    - Size in raster coordinates
    - Scaling matrix S_{raster}
      - May be non-uniform $\rightarrow$ Distortion
  - Positioning on the screen
    - Translation T_{raster}

Orthographic Projection

- **Step 2a: Translation (orthographic)**
  - Bring near clipping plane into the origin
- **Step 2b: Scaling to regular box [-1 .. 1]² x [0 .. -1]**
- **Mapping of X and Y**

$$P_o = S_{xyz} T_{near} = \begin{pmatrix} \frac{2}{l-r} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{1}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & near \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Camera Transformation

- **Complete Transformation**
  - Perspective Projection

$$K = T_{raster} S_{raster} \quad P_{parallel} \quad P_{persp} S_{far} S_{xy} H \quad RT$$

  - Orthographic Projection

$$K = T_{raster} S_{raster} \quad P_{parallel} \quad S_{xyz} T_{near} H \quad RT$$

- **Other representations**
  - Different camera parameters as input
  - Different canonical viewing frustum
  - Different normalized coordinates
    - $[-1 .. 1]^3$ versus $[0 .. 1]^3$ versus ...
  - ...

  ➔ ***Different transformation matrices***

Coordinate Systems

- **Normalized (projection) coordinates**
  - 3D: Normalized $[-1 .. 1]^3$ oder $[-1 .. 1]^2 \times [0 .. -1]$
  - Clipping
  - Parallel projection
- **Normalized 2D device coordinates $[-1 .. 1]^2$**
  - Translation and scaling
- **Normalized 2D device coordinates $[0 .. 1]^2$**
  - Where is the origin?
    - RenderMan, X11: Upper left
    - OpenGL: Lower left
  - Viewport-Transformation
    - Adjustment of aspect ratio
    - Position in raster coordinates
- **Raster Coordinates**
  - 2D: Units in pixels [0 .. xres-1, 0 .. yres-1]

OpenGL

- **ModelView Matrix**
  - Modeling transformations AND viewing transformation
  - No explicit world coordinates
- **Perspective transformation**
  - simple specification
    - glFrustum(left, right, bottom, top, near, far)
    - glOrtho(left, right, bottom, top, near, far)
- **Viewport transformation**
  - glViewport(x, y, width, height)

Limitations

- **Pinhole camera model**
  - Linear in homogeneous coordinates
    - Fast computation
- **Missing features**
  - Depth-of-field
  - Lens distortion, aberrations
  - Vignetting
  - Flare

Wrap-Up

- **World coordinates**
  - Scene composition
- **Camera coordinates**
  - Translation to camera position
  - Rotation to camera view orientation, optical axis along z axis
  - Different camera specifications
- **Normalized coordinates**
  - Scaling to canonical frustum
- **Perspective transformation**
  - Lines through origin → parallel to z axis
- **Parallel projection to 2D**
  - Omit depth
- **Viewport transformation**
  - Aspect ratio adjustment
  - Origin shift in image plane