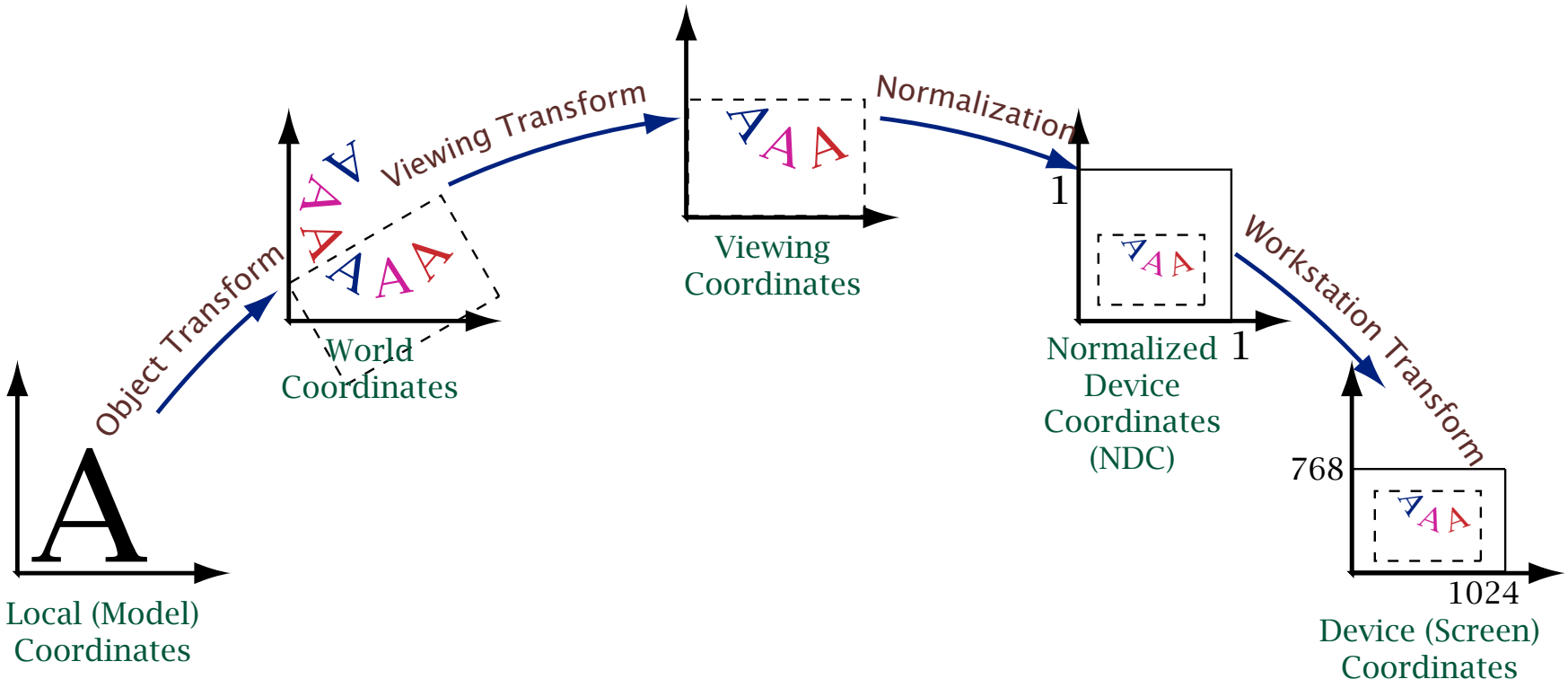
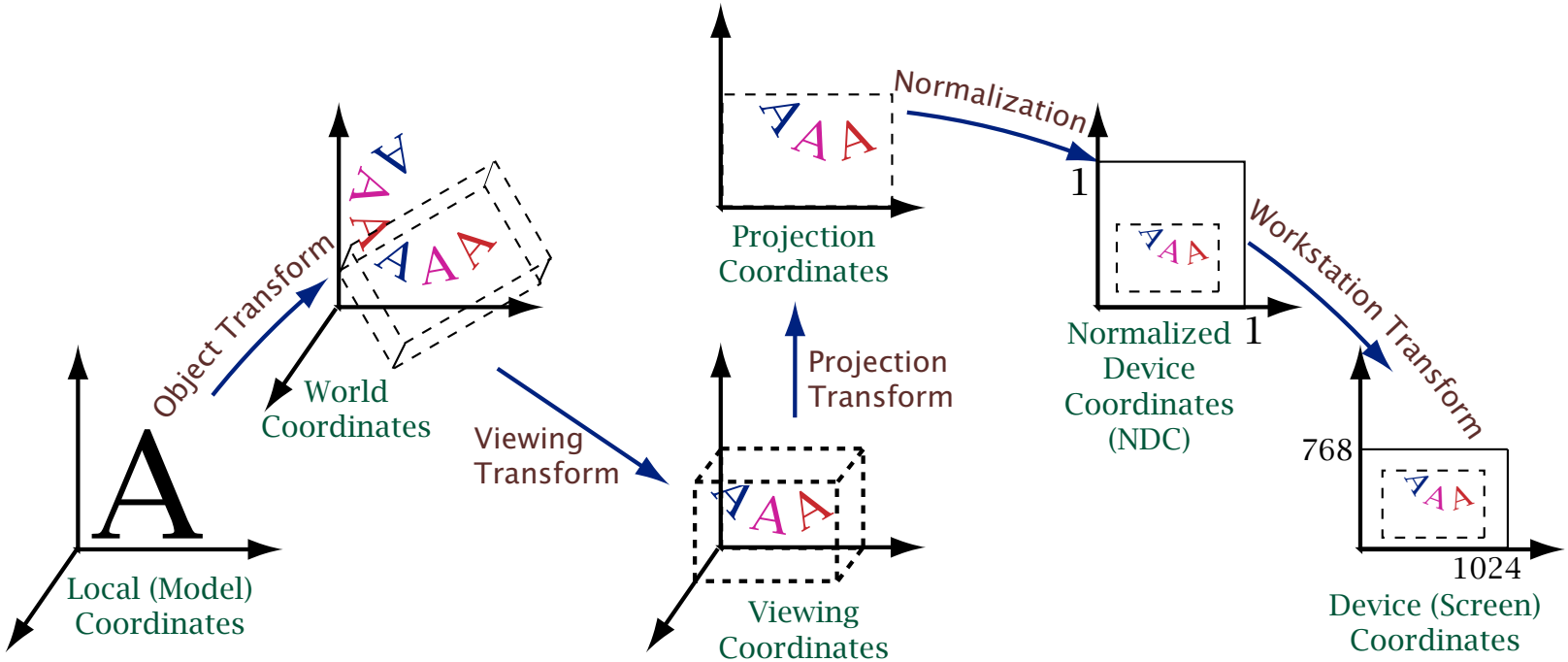
Viewing (Review of 2D)

Recall the 2-D viewing pipeline:



In 3-D we must clip with respect to a 3-D window, and must apply a projection to a 2-D device.

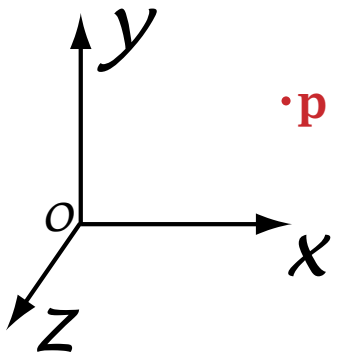
Viewing in 3D



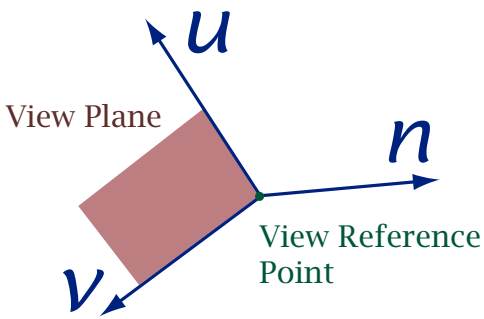
From World to projection coordinates entails:

1. Define a 3-D window, or view volume.
2. Transform to the viewing coordinates.
3. Clip graphical objects with respect to the view volume (e.g., 3D versions of Cohen-Sutherland, Liang-Barsky, Sutherland-Hodgeman).
4. Apply a projection transformation to obtain the projection coordinates.

Camera Model for Projections



World Coordinates



Viewing Coordinates

Transforming from World to Viewing Coordinates

The OpenGL command gluLookAt defines two points, and one vector:

eye= (eyeX, eyeY, eyeZ) denotes the *view reference point* or *eye*: the origin of the viewing coordinate system.

look= (lookX, lookY, lookZ) denotes a point along the axis of the camera lens.

up= (upX, upY, upZ) denotes a vector that will have a vertical projection in the viewport.

From these, the viewing (camera) coorinates are defined by $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{n}$, where

$$\mathbf{n} = \frac{\mathbf{eye} - \mathbf{look}}{\|\mathbf{eye} - \mathbf{look}\|}$$

$$\mathbf{u} = \frac{\mathbf{up} \times \mathbf{n}}{\|\mathbf{up} \times \mathbf{n}\|}$$

$$\mathbf{v} = \mathbf{n} \times \mathbf{u}$$

Note the view direction is really $-\mathbf{n}$.

World to Viewing Coordinate Transformation

Let $\mathbf{p} = (x, y, z)$ denote a point in world coordinates. To find the coordinates of this point in the viewing coordinate system, apply the transformation

$$
\begin{aligned}
V &= R(\mathbf{u}, \mathbf{v}, \mathbf{n}) T(-\mathbf{eye}) \\
&= \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -eyeX \\ 0 & 1 & 0 & -eyeY \\ 0 & 0 & 1 & -eyeZ \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} u_x & u_y & u_z & -\mathbf{u} \cdot \mathbf{eye} \\ v_x & v_y & v_z & -\mathbf{v} \cdot \mathbf{eye} \\ n_x & n_y & n_z & -\mathbf{n} \cdot \mathbf{eye} \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

Projections

Earliest evidence of projection transformations occur in cave paintings from the paleolithic era. These examples from Lascaux, France are approximately 15,000 years old.

Painting portraits of the three-dimensional world on a two-dimensional canvas (or rock wall) is similar to the task of representing a three dimensional model on a two-dimensional graphics device.

Visual Art in Ancient Egypt

Visual Art in the Italian Renaissance

Mathematics of point-line perspective was pioneered during the Italian Renaissance by Filippo Brunelleschi (1377–1446), Leono Battista Alberti (1404–1472), and Piero della Francesca (c.1420–1492).

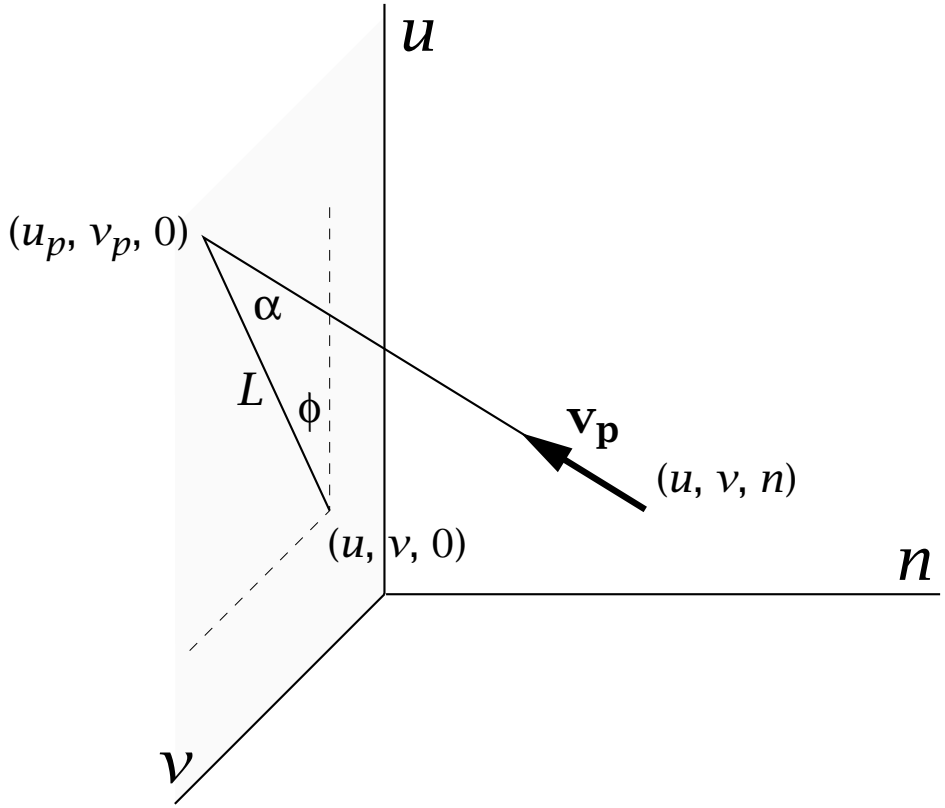
Ideal Piazza, Piero della Francesca

Projections

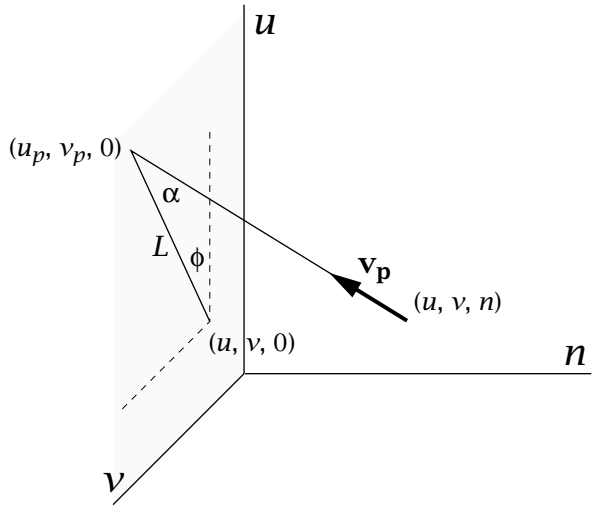
- Parallel
  - Orthographic
    - Top (plan)
    - Front Elevation
    - Side Elevation
    - Axonometric (e.g., Isometric)
  - Oblique
    - Cabinet
    - Cavalier
    - Other
- Perspective
  - One-point
  - Two-point
  - Three-point

Parallel Projection

Given a constant vector v_p, construct a line (parallel to v_p) through every vertex of interest in viewing coordinates. The intersections of these lines with the view plane define the parallel projections.



Parallel Projection (cont.)



Alternatively, the projection can be defined using angles α and ϕ.

$$\begin{aligned} u_p &= u + L\cos\phi \\ v_p &= v + L\sin\phi \end{aligned}$$

where, $L = n/\tan\alpha$. Letting $L_1 = 1/\tan\alpha$, we obtain

$$M_{\text{parallel}} = \begin{pmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Special Parallel Projections

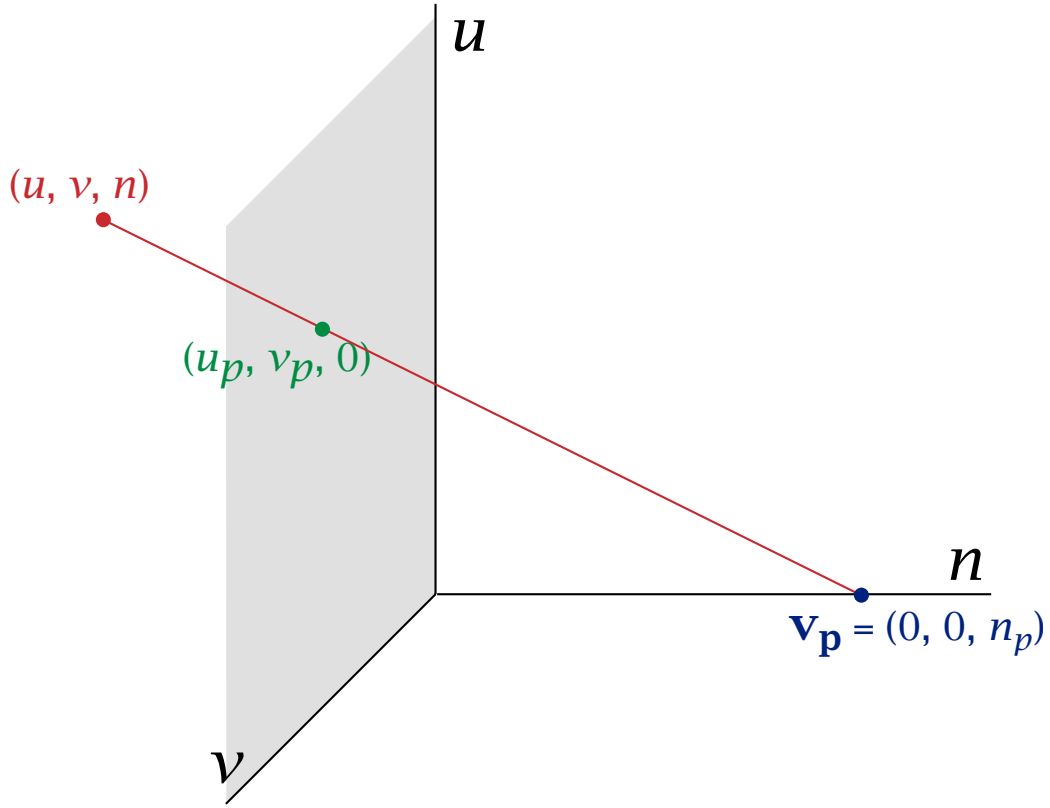
- *Orthographic*: $\alpha = \pi/2$ or $L_1 = 0$. Useful for architecture and CAD: elevations and plans. Axonometric (orthographic) projections reveal more that one side of an object. An isometric (orthographic) projection treats distances along each coordinate axis equally. In general, lengths and angles are accurate. Normal lengths are invisible.
- *Cavalier*: $\alpha = \pi/4$, thus $L_1 = 1$. Lines perpendicular to the projected plane appear with no change in length.
- *Cabinet*: $\alpha = \tan^{-1} 2$, thus $L_1 = 1/2$. More realistic than cavalier: normal lengths are half as long.

Popular values for ϕ are $\pi/4$ and $\pi/6$.

Theorem: The parallel projections of parallel lines are parallel.

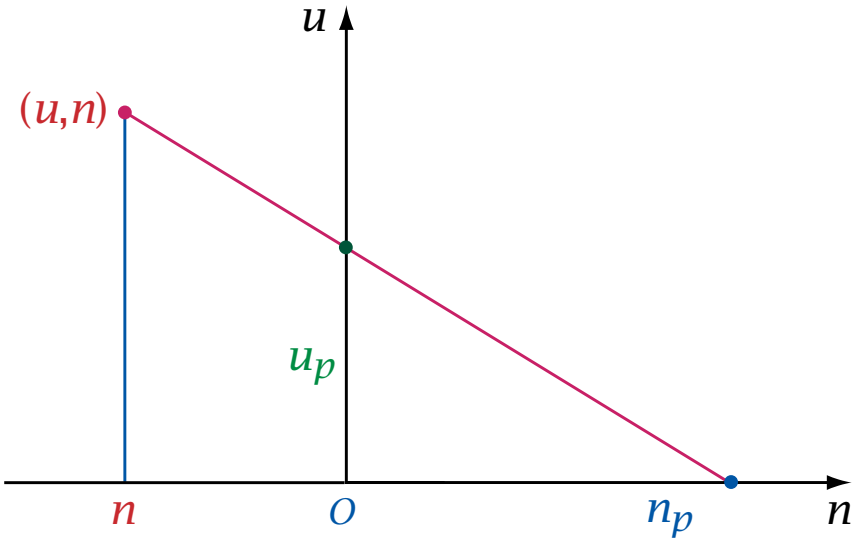
Point-Line Perspective (Phigs)

Each point (in viewing coordinates) is projected along a line that contains a common *projection reference point* p_{prp}. Intersections of these lines with the view plane defines the projections.



Point-Line Perspective (cont.)

An orthographic view, illustrating two similar triangles:



$$\frac{u_p}{n_p} = \frac{u}{n_p - n} \Longrightarrow u_p = \frac{u}{1 - \frac{n}{n_p}}$$

A similar equation can be obtained for v_p.

N.B. The expression is not linear in n. However, we can exploit the rational nature of homogeneous coordinates to express the above as a linear operation.

Point-Line Perspective

Let,

$$M_{\text{persective}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{n_p} & 1 \end{pmatrix}$$

Then

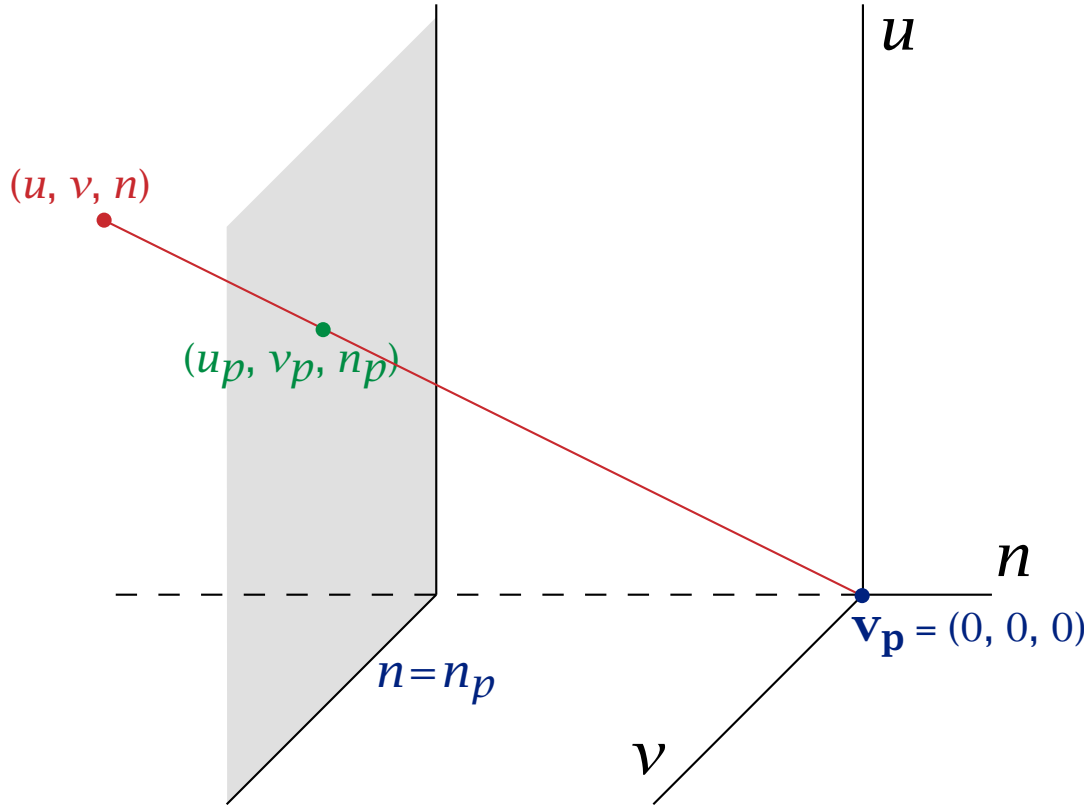
$$\begin{pmatrix} u_h \\ v_h \\ n_h \\ h_h \end{pmatrix} = M_{\text{persective}} \begin{pmatrix} u \\ v \\ n \\ h \end{pmatrix}$$

whence,

$$u_p = \frac{u_h}{h_h}, \qquad v_p = \frac{v_h}{h_h}, \qquad n_p = 0$$

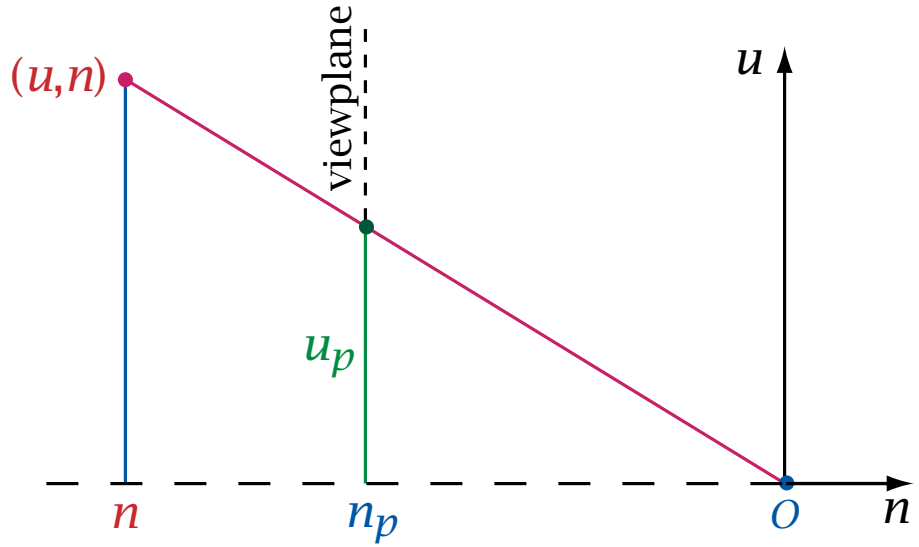
Point-Line Perspective (OpenGL)

The projection reference point $\mathbf{v}_p = (0,0,0)$ is a the origin of the camera (uvn) coordinate system. The view plane is described by the equation $n = n_p$ for some chosen $n_p < 0$. Each model vertex (u,v,n) is projected onto the viewplane (u_p, v_p, n_p) via a line segment that connects the model vertex with the projection reference point.



Point-Line Perspective (cont.)

An orthographic view, illustrating two similar triangles:



$$\frac{u_p}{-n_p} = \frac{u}{-n} \Longrightarrow u_p = \frac{u}{n/n_p}$$

A similar equation can be obtained for v_p.

N.B. The expression is not linear in n. However, we can exploit the rational nature of homogeneous coordinates to express the above as a linear operation.

Point-Line Perspective

Let,

$$M_{\text{persective}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{n_p} & 0 \end{pmatrix}$$

Then

$$\begin{pmatrix} u_h \\ v_h \\ n_h \\ h_h \end{pmatrix} = M_{\text{persective}} \begin{pmatrix} u \\ v \\ n \\ h \end{pmatrix}$$

whence,

$$u_p = \frac{u_h}{h_h} = \frac{u n_p}{n}, \qquad v_p = \frac{v_h}{h_h} = \frac{v n_p}{n}, \qquad n_p = \frac{n_h}{h_h} = n_p$$

Defining a Perspective Projection with OpenGL

```
gluPerspective(GLdouble fovy, GLdouble aspect,
               GLdouble near, GLdouble far);
```

or

```
glFrustum(GLdouble left,   GLdouble right,
          GLdouble bottom, GLdouble top
          GLdouble near,   GLdouble far);
```

Clipping a Perspective Projection

A perspective transformation defines a frustum of a rectangular cone.