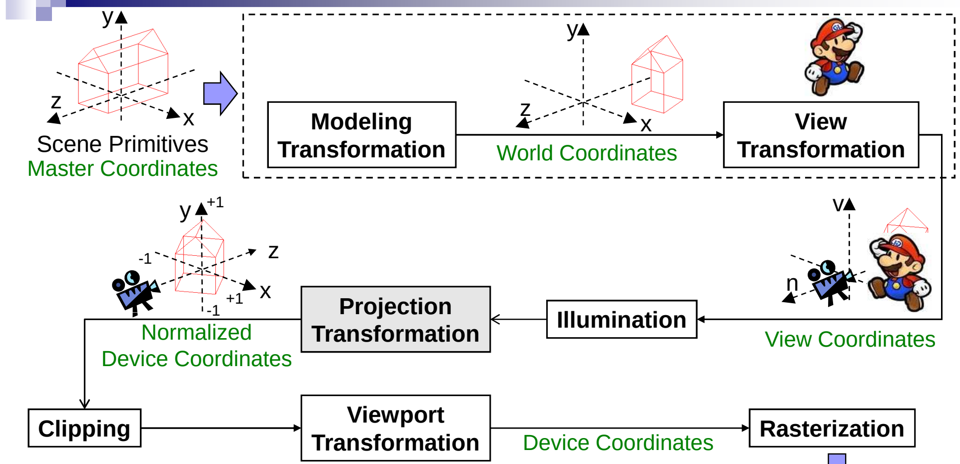
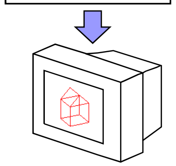
# Computer Graphics: Projection Transformations

Part 2 – Lecture 2





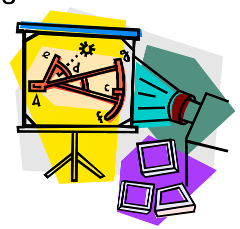

# PROJECTION TRANSFORMATIONS





## Principles of Geometric Projections

- **Projection:** a mapping of coordinate values from a higher dimension to lower dimension, usually N ⇨ N-1, e.g. 3D ⇨ 2D
- **Requirements:**
  - **Projection surface**: plane (linear projection) or surface such as a sphere or conic section (non-linear projection)
  - **Projection rays**, or ***projectors***: lines from object projected towards projection surface
  - **Direction of projection**: orientation of each projector
    - Orthographic (parallel) projection: all projectors parallel to a common direction of projection.
    - Perspective projection: all projectors pass through a center of projection (3D point), but have different directions



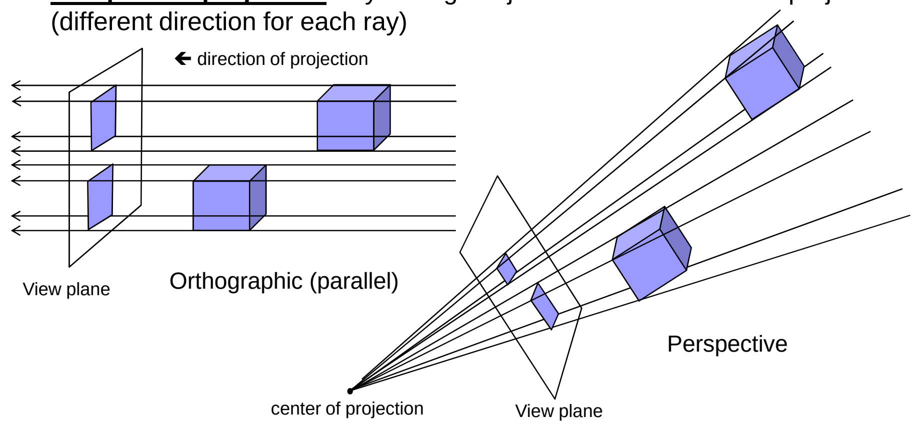
- **How to project:** Intersect projection ray through object vertex with the projection surface



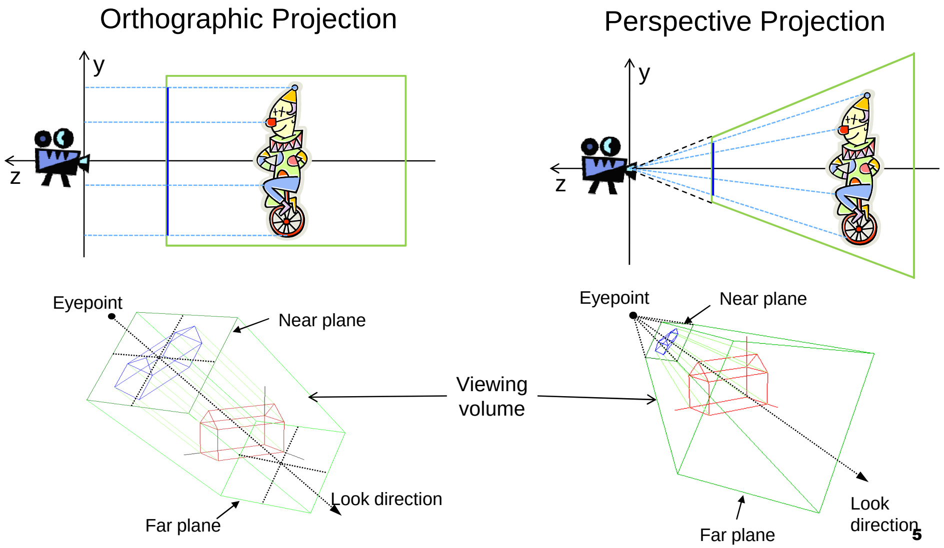


## Orthographic vs. Perspective Projection

- **Orthographic projection**: ray through object vertex in the projection direction (same direction for all rays, orthogonal to projection plane)
- **Perspective projection**: ray through object vertex and center of projection (different direction for each ray)



# Orthographic vs. Perspective Projection



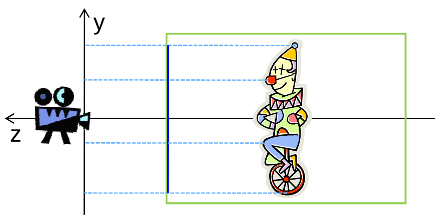



# Ortho / Perspective Cameras in OpenGL

- **Orthographic**
  - ```
    void glOrtho( GLdouble left, GLdouble right,
                  GLdouble bottom, GLdouble top,
                  GLdouble zNear, GLdouble zFar )
    ```
  - View volume boundaries in World Coord units, relative to eyepoint in the look direction. Z is positive distance from eye (along negative Z axis)
  - View volume ***may be* symmetric** about look direction vector (typical)
- **Perspective**
  - ```
    void gluPerspective( GLdouble fovy, GLdouble aspect,
                         GLdouble zNear, GLdouble zFar )
    ```
  - Vertical field of view angle `fovy` specified in degrees.
  - Horizontal fov determined by `aspect` ratio = width/height
    `fovx = aspect * fovy;`
  - View volume (frustum, or truncated pyramid) ***always* symmetric** about eyepoint towards the look direction.





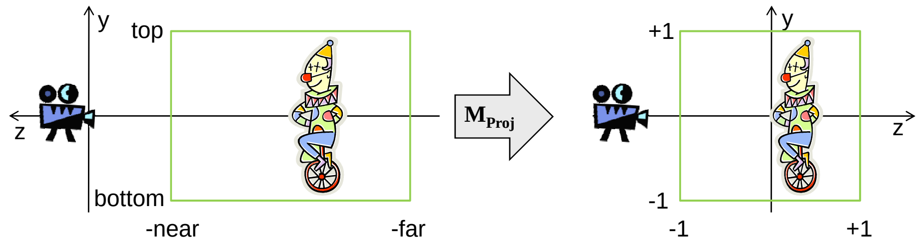



# ORTHOGRAPHIC PROJECTION



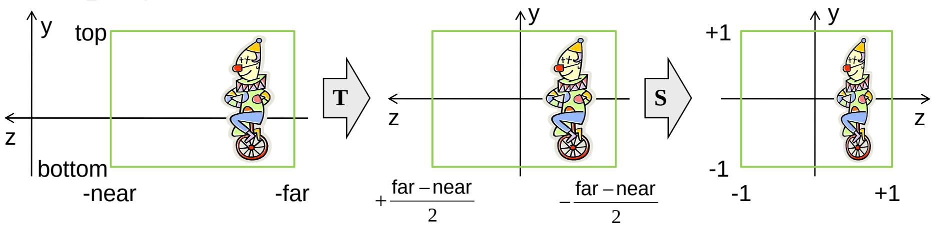
# Projection Transformation Matrix $\mathbf{M}_{\mathbf{proj}}$

- Maps **View Coords.** to **Normalized Device Coords.** (**NDC**)
- View volume boundaries are mapped to (-1,+1) cube in X, Y, Z
- View Coordinates are **RHS** and NDC are **LHS**, so $\mathbf{M}_{\mathbf{proj}}$ inverts Z values
- For orthographic projection:




To get a 2D image: take only x and y components

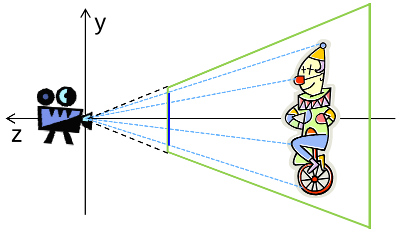
# $\mathbf{M_{proj}}$ for Orthographic Projections




$$\mathbf{M_{proj}} = \mathbf{S}\ \mathbf{T}$$

$$= \begin{pmatrix} \frac{2}{right-left} & 0 & 0 & 0 \\ 0 & \frac{2}{top-bottom} & 0 & 0 \\ 0 & 0 & -\frac{2}{far-near} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -(left+right)/2 \\ 0 & 1 & 0 & -(top+bottom)/2 \\ 0 & 0 & 1 & +(near+far)/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

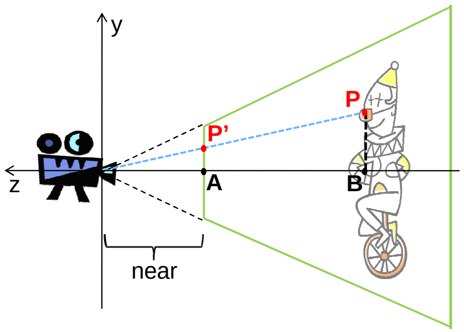





# PERSPECTIVE PROJECTION



# Perspective Projection of a Vertex




- What are the coordinates of **P'** ?
- Camera-**A**-**P'** and Camera-**B**-**P** are similar triangles
- Ratios of similar sides are equal:

$$\frac{P_y'}{near} = \frac{P_y}{-P_z} \Leftrightarrow \boxed{P_y' = \frac{near}{-P_z} P_y}$$

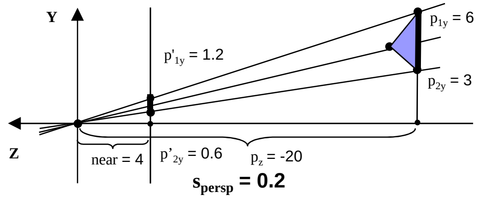
- When looking from the bottom, we get analogous calculations for the x-coordinate of **P'**:

$$\frac{P_x'}{near} = \frac{P_x}{-P_z} \Leftrightarrow \boxed{P_x' = \frac{near}{-P_z} P_x}$$

- Perspective scaling factor $s_{persp} = \frac{near}{-P_z}$

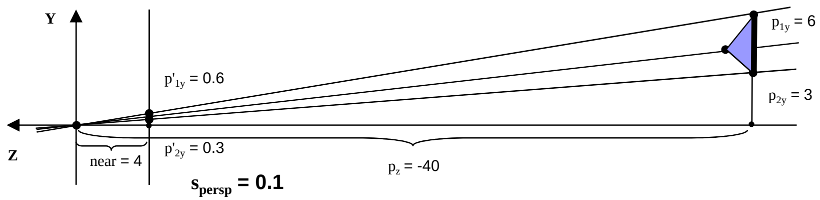


# Perspective Foreshortening




$d_y = (p_{1y} - p_{2y}) = 3$

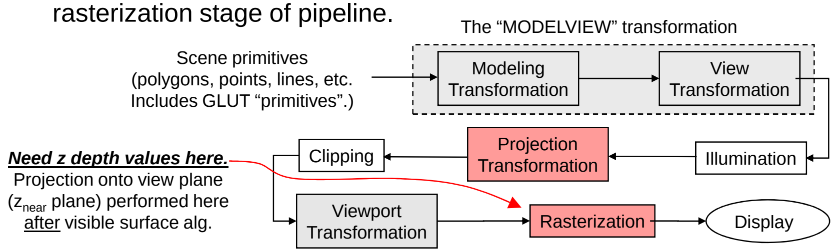
$d_y' = (p'_{1y} - p'_{2y})$
$= s_{persp}\, d_y$
$= (0.2)\ 3 = 0.6$




$d_y = (p_{1y} - p_{2y}) = 3$ $\quad$ $d_y' = (p'_{1y} - p'_{2y}) = \mathbf{s_{persp}}\, d_y = (0.1)\ 3 = 0.3$

# Perspective Transformation

- <u>Perspective projection CANNOT be used in 3D graphics pipeline!</u>
  - Why not? Because it sets all projected z coordinates to <u>same value</u>, $z_{near}$ But, visible surface algorithm (Z buffer alg.) needs <u>z depth values</u> during rasterization stage of pipeline.

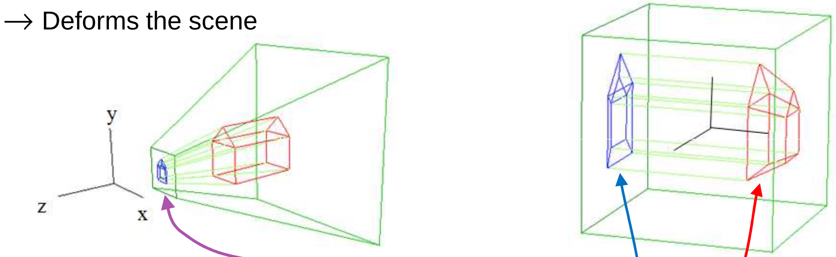



  - Therefore, pipeline uses **perspective *<u>transformation</u>***, not **perspective *<u>projection</u>***
  - Scales x, y, **<u>and z</u>** coordinates by a scale factor dependent upon 1/z
  - Projection is performed during rasterization stage after hidden surface removal




# Perspective Transformation and Projection

- **Perspective transformation** converts 3D coordinates to perspective corrected 3D coordinates
  $\rightarrow$ Deforms the scene




  - We want perspective projection to look like this
  - But, we actually perform it in a 2 step process:
    - Perspective **<u>transformation</u>**: 3D → 3D
    - Orthographic **<u>projection</u>**: 3D → 2D




# Perspective Transformation (cont'd)

- Perspective transformation <u>requirements</u>:
  1. x and y values must be scaled by same factor as derived in perspective projection equations.
  2. z values must maintain depth ordering (monotonic increasing)
  3. z values must map: $\mathbf{-z_{near}} \rightarrow -1$ and $\mathbf{-z_{far}} \rightarrow +1$, view volume → NDC cube.
- In other words, we need a transformation that given a point *P* results in a transformed point *P'* such that $P'_x$ and $P'_y$ meet requirement 1 and $f(p_z)$ meets requirements 2 and 3.

$$P' = \left( \frac{-near}{p_z} p_x, \quad \frac{-near}{p_z} p_y, \quad f(p_z) \right)$$

- **<u>Question:</u>** Is there any matrix, **P**, such that **P** $P = P'$ ?
- **<u>Answer:</u>** **<u>Not possible</u>** because no linear combination of $p_x$, $p_y$, $p_z$, can result in a term with $p_z$ in the denominator!

$$\begin{pmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{-near\ p_x}{\boxed{p_z}} \\ \frac{-near\ p_y}{\boxed{p_z}} \\ f(p_z) \\ 1 \end{pmatrix}$$




# Perspective Transformation (cont'd)

- But, there is a matrix **P** that can produce this result:

$$\begin{pmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} near\ p_x \\ near\ p_y \\ -f(p_z)\ p_z \\ -p_z \end{pmatrix}$$

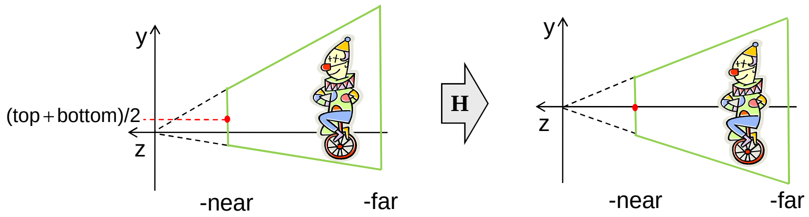
- After conversion to ordinary coordinates: $P' = \left( \frac{near\ p_x}{-p_z}, \ \frac{near\ p_y}{-p_z}, \ f(p_z) \right)$

$$\mathbf{P} = \begin{pmatrix} near & 0 & 0 & 0 \\ 0 & near & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad \text{with} \quad a = -\frac{far + near}{far - near}, \quad b = \frac{-2\ far\ near}{far - near}$$

so that $P' = \left( \frac{near\ p_x}{-p_z}, \ \frac{near\ p_y}{-p_z}, \ \frac{a\ p_z + b}{-p_z} \right)$

- Result: perspective transformation can be done with matrix multiplication in the rendering pipeline (using hardware!)

## $\mathbf{M_{proj}}$ for Perspective Projections 1

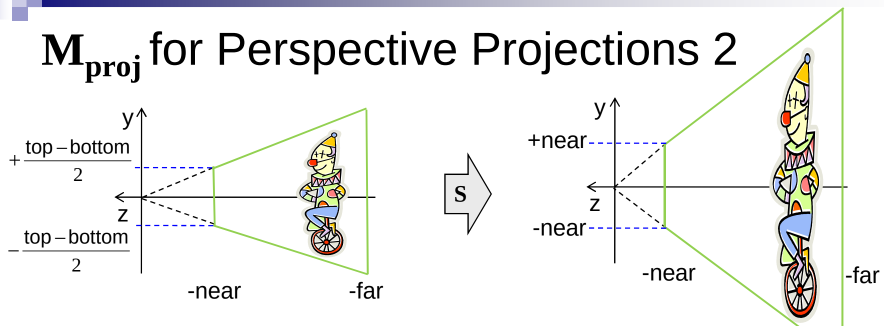



Shear everything (**H**) so that z-axis is in the middle of the frustum

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & \dfrac{-(left+right)/2}{-near} & 0 \\ 0 & 1 & \dfrac{-(top+bottom)/2}{-near} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dfrac{left+right}{2\ near} & 0 \\ 0 & 1 & \dfrac{top+bottom}{2\ near} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



## $\mathbf{M_{proj}}$ for Perspective Projections 2

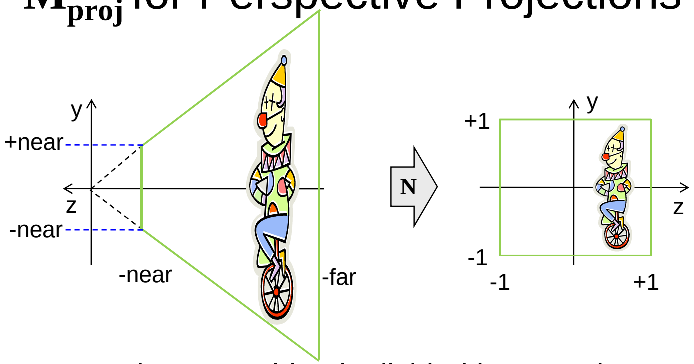



Scale everything (**S**) so that view plane has height and width 2*near

$$\mathbf{S} = \begin{pmatrix} \dfrac{near}{(right-left)/2} & 0 & 0 & 0 \\ 0 & \dfrac{near}{(top-bottom)/2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \dfrac{2\ near}{right-left} & 0 & 0 & 0 \\ 0 & \dfrac{2\ near}{top-bottom} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



## $\mathbf{M_{proj}}$ for Perspective Projections 3




Set w so that everything is divided by -z and normalize z to (-1, +1)

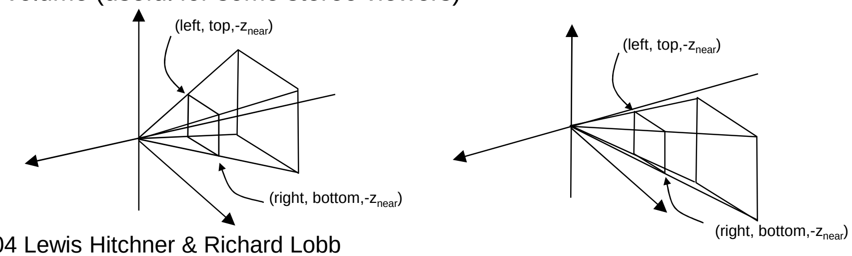
$$\mathbf{N} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix} \qquad a = -\frac{far + near}{far - near}, \quad b = \frac{-2\ far\ near}{far - near}$$

The final result is: $\mathbf{M_{proj}} = \mathbf{N\ S\ H}$



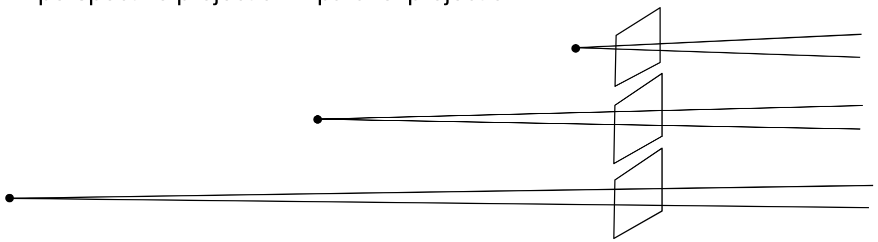
## Perspective Transformation in OpenGL

- View volume given by ***frustum*** (truncated pyramid):
  `glFrustum(left, right, bottom, top, znear, zfar)`
- `gluPerspective` computes these terms from its parameters:
  ```
  top = zNear * tan((π/180)viewAngle/2);
  bottom = -top;
  right = top * aspect;   left = -right;
  ```
- **Note:** with `gluPerspective` the view volume is symmetric about the view direction vector. With `glFrustum` you can specify a non-symmetric view volume (useful for some stereo viewers)



## Principles Geometric Projections (cont'd)

- **Observation about perspective projection:** as center of projection moves farther and farther away, lines of projection become more nearly parallel. In the limit, when center of projection is at an infinite distance, perspective projection ≡ parallel projection.



- Rays of light from a point source shining on an opaque object forming a shadow on a projection plane are similar to perspective projection rays.
- Rays of light from a point source at "infinite distance" (e.g., the Sun 93x$10^6$ miles from the Earth) forming a shadow are similar to parallel projection.





# SUMMARY



## Summary

- Projection transformation matrix $\mathbf{M}_{proj}$:
  maps World Coordinate values in view volume to Normalized Device Coordinates (NDC) in the range (-1, +1)
- Orthographic projection:
  - Objects keep their original size, no matter how far away
  - $\mathbf{M}_{proj} = \mathbf{S}\ \mathbf{T}$ (translate and scale)
- Perspective projection:
  - The further away an object, the smaller it appears
  - $\mathbf{M}_{proj} = \mathbf{N}\ \mathbf{S}\ \mathbf{H}$ (shear, scale, normalize z & set w for division by z)

References:

Perspective Projections: Hill, Chapter 7.4



## Quiz

1. What are normalized device coordinates (NDCs)?
2. What is the difference between orthographic and perspective projection?
3. For given left, right, top, bottom, near and far, derive the S and T in the transformation matrix $\mathbf{M}_{proj}$ = S T for orthographic projections.
4. In the diagram below, how do you calculate P' for a given P and near?