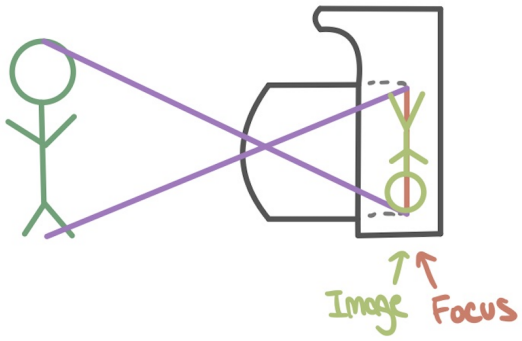
2 Projection

2.1 Camera Models

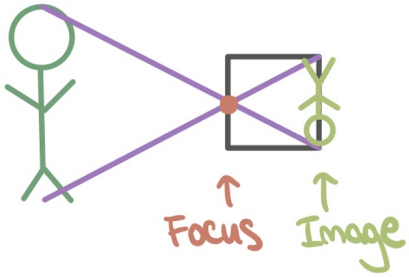
1. *Real Camera*

   A real camera has a finite depth of field that it can focus on, due to physical constraints.



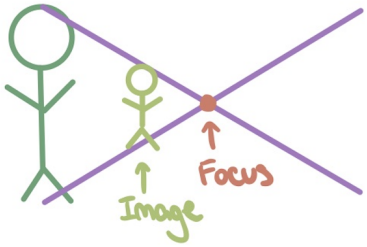
2. *Pinhole Camera*

   A pinhole camera has theoretically an 'infinite' depth of field. However, the image on the image plane is reversed.



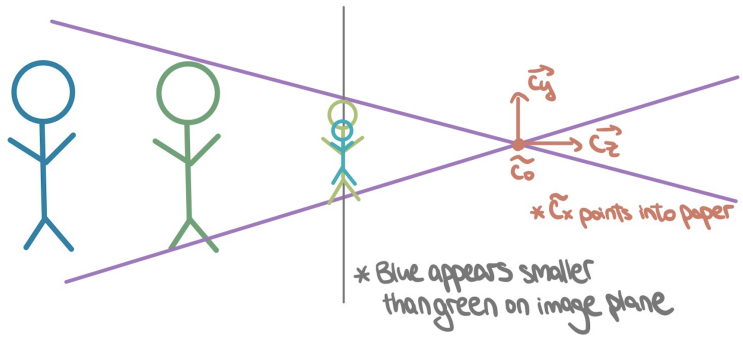
3. *Graphics Camera*

   A graphics camera is like a pinhole camera, but the image plane is placed in front of the focal point. As a result, the image has the same orientation as the original scene.



2.2 The Camera Frame

Foreshortening describes the effect that an object farther away would appear smaller to the human eye, as shown in the diagram below:

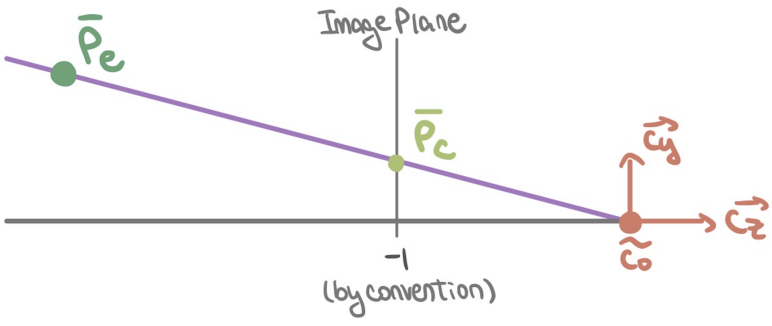


In graphics, to achieve the foreshortening effect so that the scene looks realistic, we can use perspective projection.

Take-home question 7: What is the difference between perspective and orthographic projection? What would the scene above look like in orthographic projection?

2.3 Perspective Projection

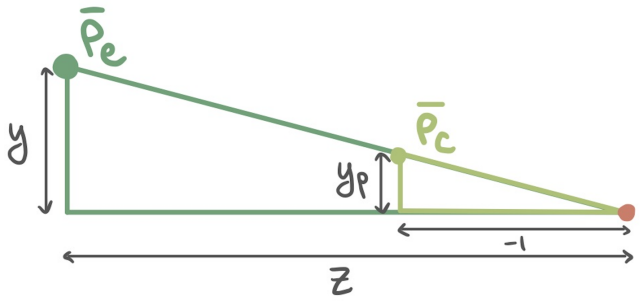
Suppose we have a point $\widetilde{\boldsymbol{p}}$ that is being captured by the graphics camera. Its coordinates in the camera frame are denoted by $\overline{\mathsf{p}}_e$ (also called the *eye coordinates*), and its coordinates after being projected onto the image plane frame are denoted by $\overline{\mathsf{p}}_c$ (also called the *clip coordinates*):



$$\overline{\mathsf{p}}_e = \begin{pmatrix} y \\ z \\ 1 \end{pmatrix} \qquad \overline{\mathsf{p}}_c = \begin{pmatrix} y_p \\ -1 \\ 1 \end{pmatrix}$$

Note: In this example, we only have the y and z axes for simplification. The same principle can be applied to the x axis for perspective projection.

To find y_p, we can use similar triangles:



$$\frac{y}{y_p} = \frac{z}{-1} \longrightarrow y_p = \frac{y}{-z}$$

We want to find a transformation matrix $\underline{\overline{\mathsf{P}}}$ that takes coordinates from the camera frame into the image plane frame. This transformation is linear if we use homogeneous coordinates:

$$\begin{pmatrix} y \\ z \\ 1 \end{pmatrix} \xrightarrow{\underline{\overline{\mathsf{P}}}} \begin{pmatrix} y/-z \\ -1 \\ 1 \end{pmatrix}$$

Assume that we can multiply any coordinate by an arbitrary constant w ($w \neq 0$) and still obtain the same representation of the same coordinate:

$$\begin{pmatrix} y \\ z \\ 1 \end{pmatrix} \equiv \begin{pmatrix} wy \\ wz \\ w \end{pmatrix}$$

Then, we can rewrite the coordinates of the point in the image plane frame in the perspective projection equation by multiplying it with $-z$:

$$\begin{pmatrix} y \\ z \\ 1 \end{pmatrix} \xrightarrow{\underline{\overline{\mathsf{P}}}} \begin{pmatrix} y/-z \\ -1 \\ 1 \end{pmatrix} \equiv \begin{pmatrix} y \\ z \\ -z \end{pmatrix}$$

The transformation matrix of this perspective projection is:

$$\overline{\mathsf{p}}_c = \underline{\overline{\mathsf{P}}}\ \overline{\mathsf{p}}_e$$

$$\begin{pmatrix} y \\ z \\ -z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} y \\ z \\ 1 \end{pmatrix}$$

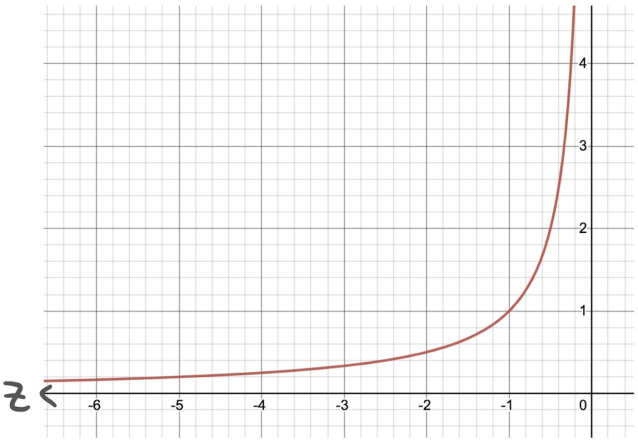
There is still an issue with this transformation: The depth information is lost due to the matrix being singular (the last column has only 0's). To retain all of the 3D information, we can slightly modify $\overline{\underline{P}}$:

$$\begin{pmatrix} y \\ 1 \\ -z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} y \\ z \\ 1 \end{pmatrix}$$

Then, perform **perspective divide** on the clip coordinates with the arbitrary constant $w = -z$ that was used in a previous step to obtain the normalized device coordinates:

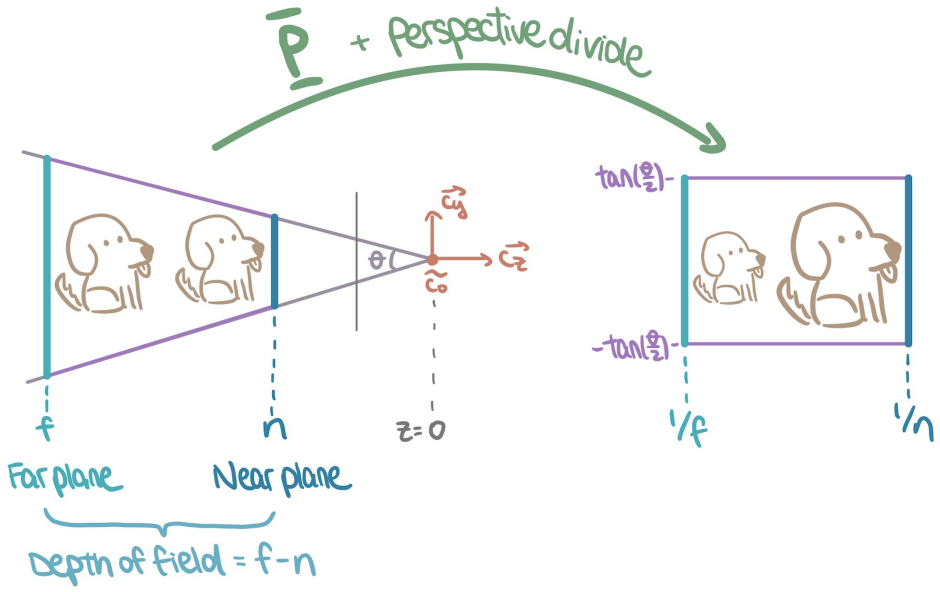
$$\overline{\mathsf{p}}_n = \overline{\mathsf{p}}_c / w = \begin{pmatrix} y/-z \\ 1/-z \\ 1 \end{pmatrix}$$

Now, the second entry of the coordinate contains a *depth-like* value (monotonic function with depth), so depth information is now preserved. We can plot its relationship with depth as:



2.4 The View Frustum

A *frustum* is a truncated pyramid, which is the shape of the region that a perspective camera can capture. From the side, when only viewing the y and z axes, this region has a trapezoid shape bounded by the far plane and the near plane. We only generate images for objects that are found within this region. After performing perspective projection on this region using $\overline{\underline{P}}$ and after performing perspective divide, the view frustum region becomes a parallel-sided box:



2.5 The OpenGL Pipeline

Redrawing the pipeline using the different coordinate systems seen so far, we have: