Computación Gráfica

Ing. Gabriel Ávila, MSc.

Proyecciones

Espacio homogéneo

Las coordenadas en 2D se han utilizado con 3 datos: $[x, y, w]^T$. Hasta ahora $w = 1$.

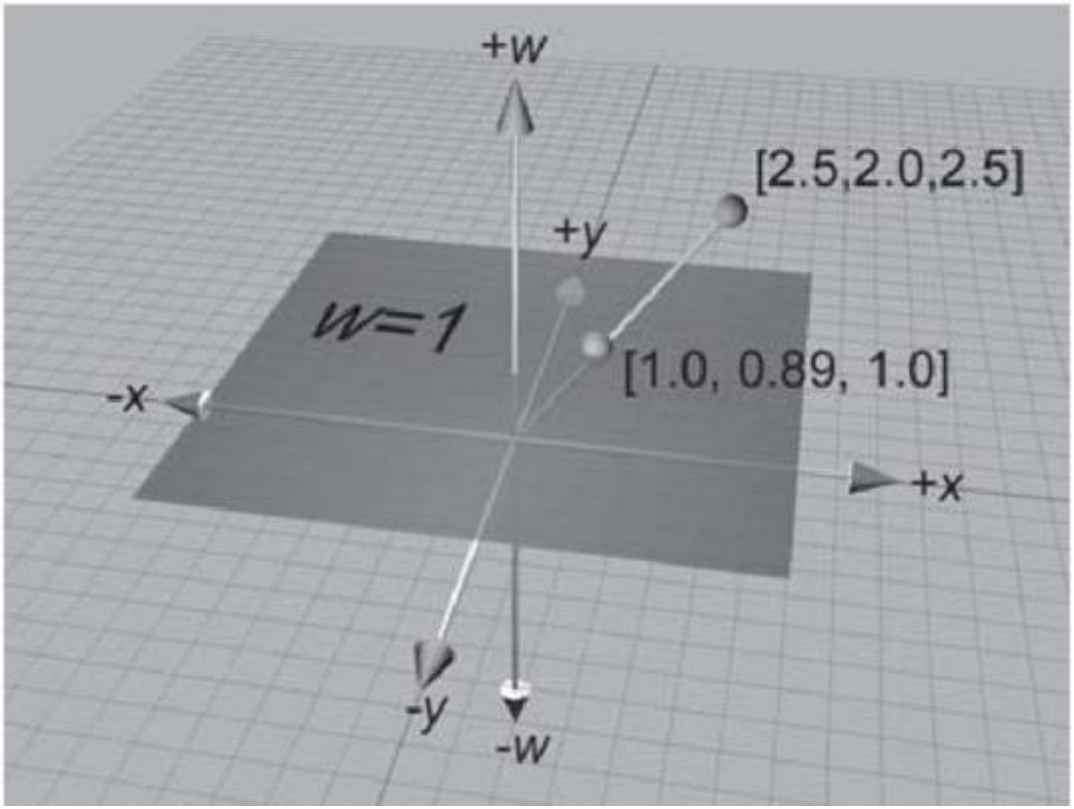
Esto equivale a trabajar en un espacio 3D, para el cual, la coordenada z=1.

La coordenada w se conoce como coordenada homogénea. Cuando su valor es diferente de 1, se puede obtener el punto correspondiente en 2D, dividiendo entre w.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \text{->} \left(\frac{x}{w}, \frac{y}{w} \right) \text{->} \begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}$$

Proyección en 2D

Dado un punto en el espacio homogéneo (x, y, w) con $w \neq 1$, es posible proyectarlo sobre el plano $w = 1$para hallar su correspondencia en 2D.



Espacio homogéneo en 4D

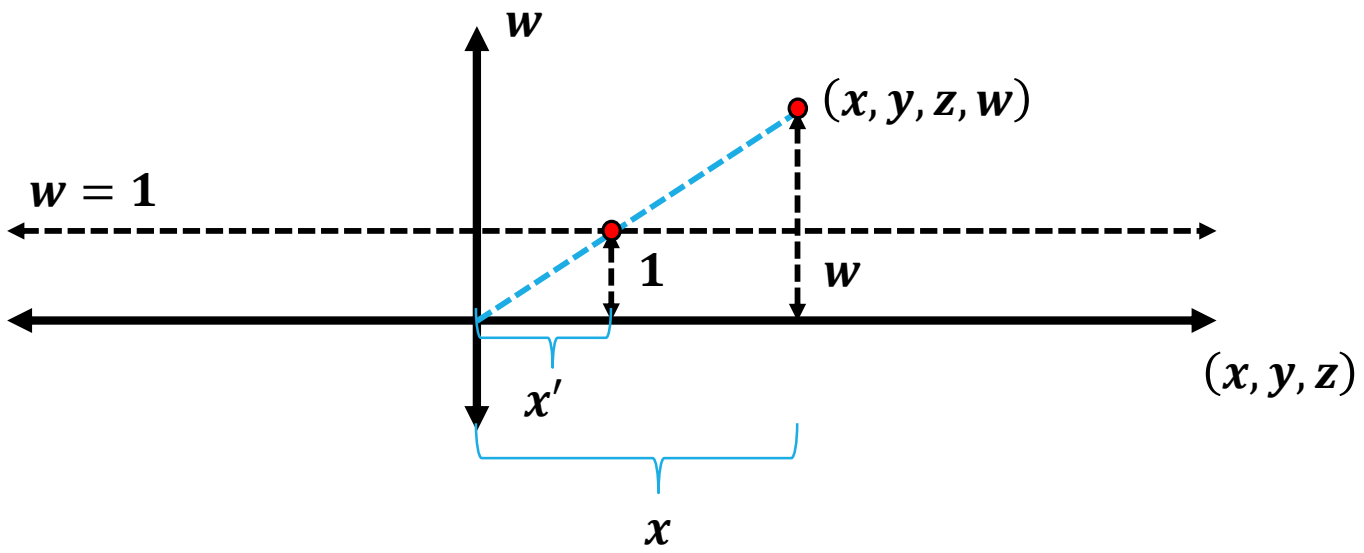
Para un espacio de coordenadas 3D es posible hacer lo mismo. ***El uso de matrices homogéneas 4D facilita las operaciones en 3D.***

Hasta ahora, normalmente a=0, b=0, c=0 y d=1, lo cual hacía que w siempre fuera 1.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

Espacio homogéneo en 4D

Al igual que en 2D, es posible proyectar un punto (x, y, z, w) sobre el "plano" en w=1. El punto resultante en 3D será (x/w, y/w, z/w).



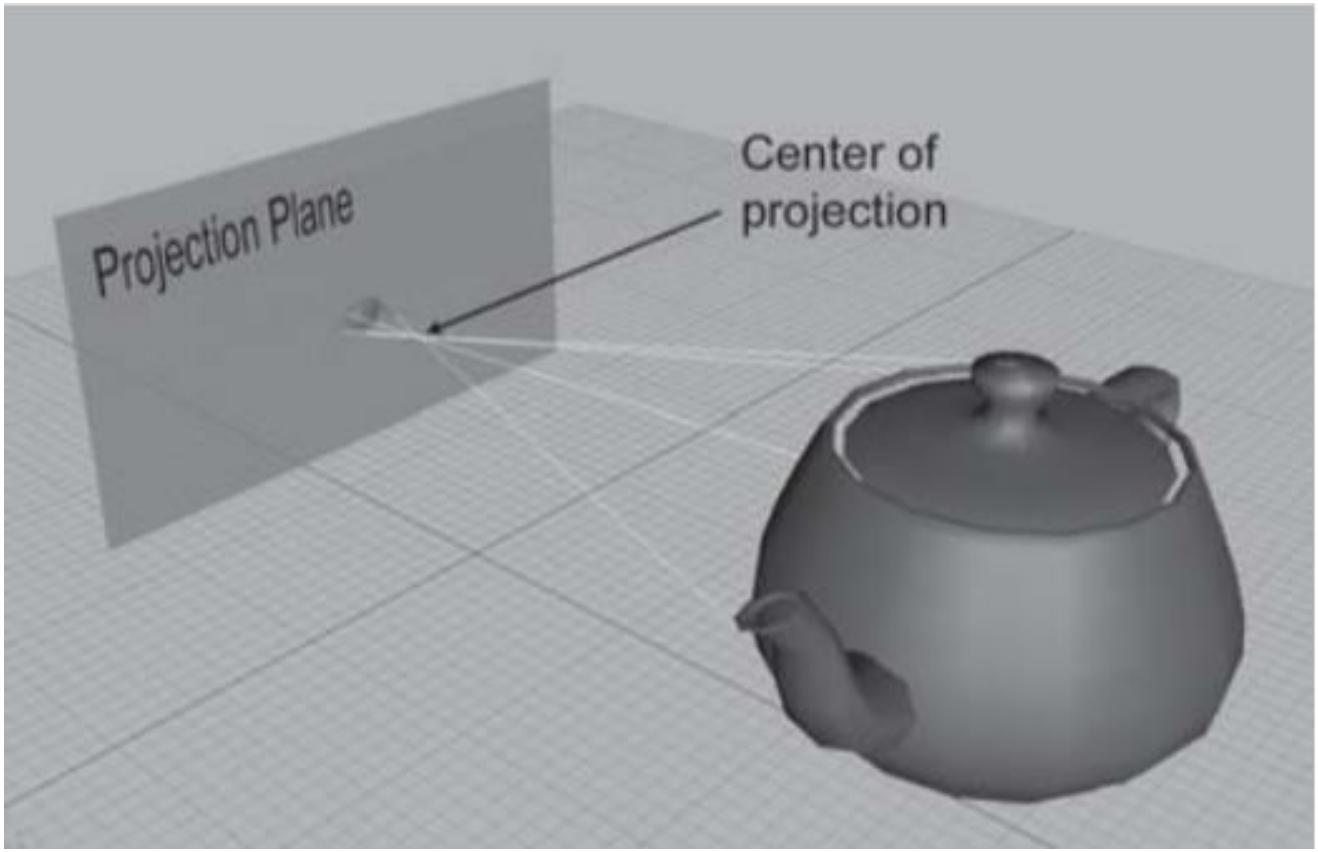
Por similitud de triángulos: $x' = \frac{x}{w}, \; y' = \frac{y}{w}, \quad z' = \frac{z}{w}$

Uso del espacio homogéneo

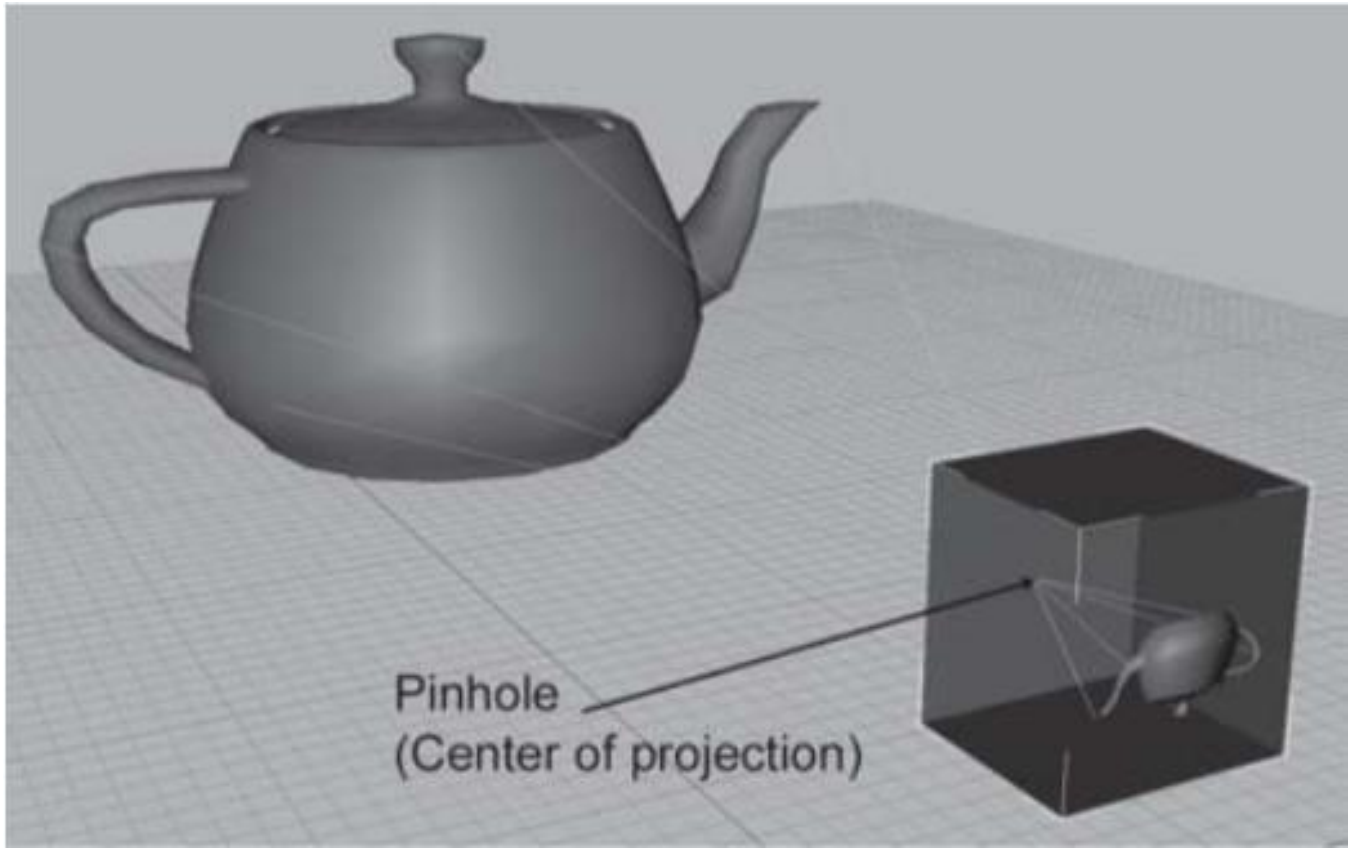
¿Cómo representar una traslación en 3D, utilizando vectores de 3 dimensiones?

Además, el uso de la 4ta dimensión permitirá hacer transformaciones de tipo proyectivo, útiles para el manejo de cámaras.

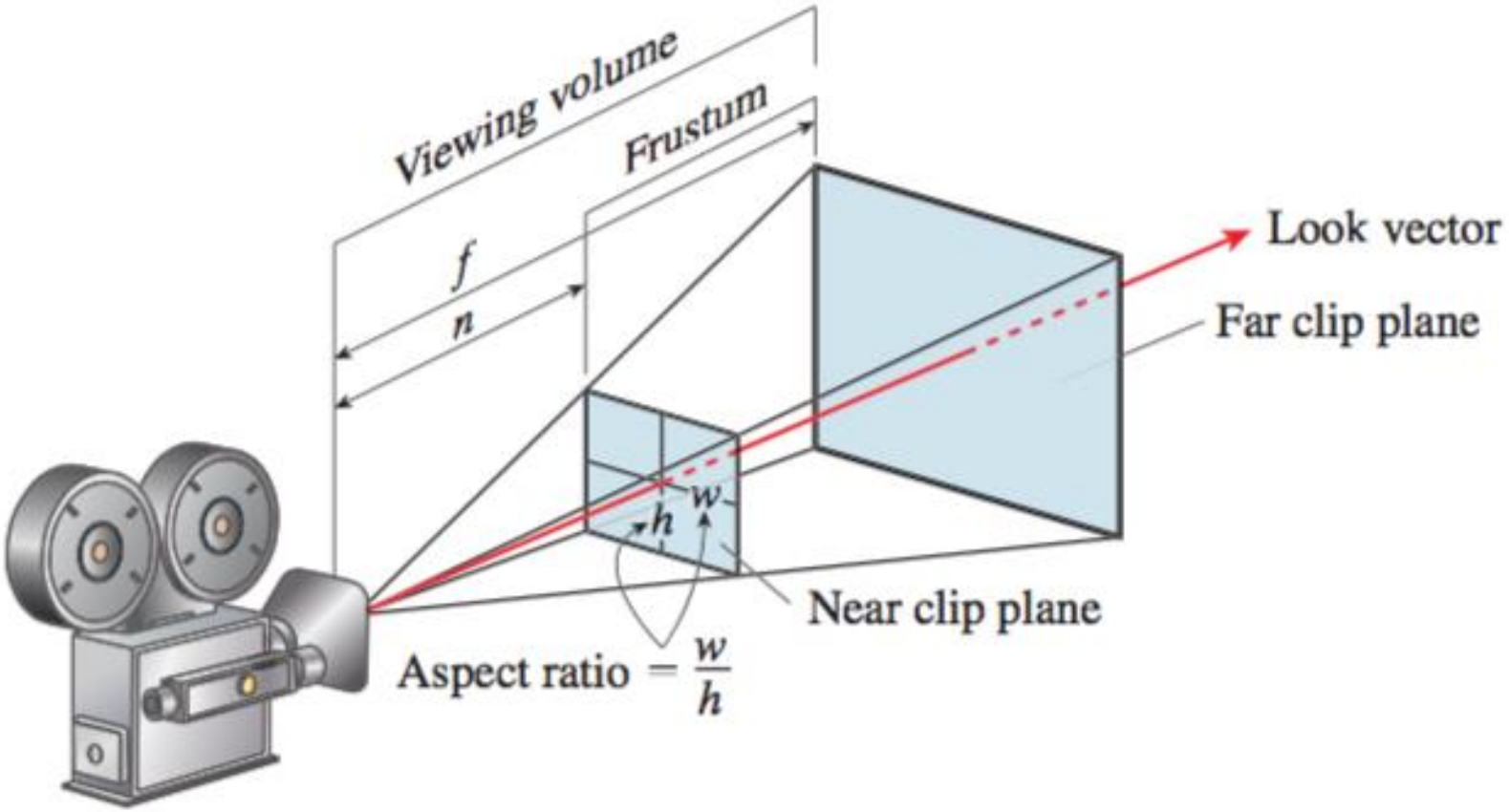
Proyección en perspectiva



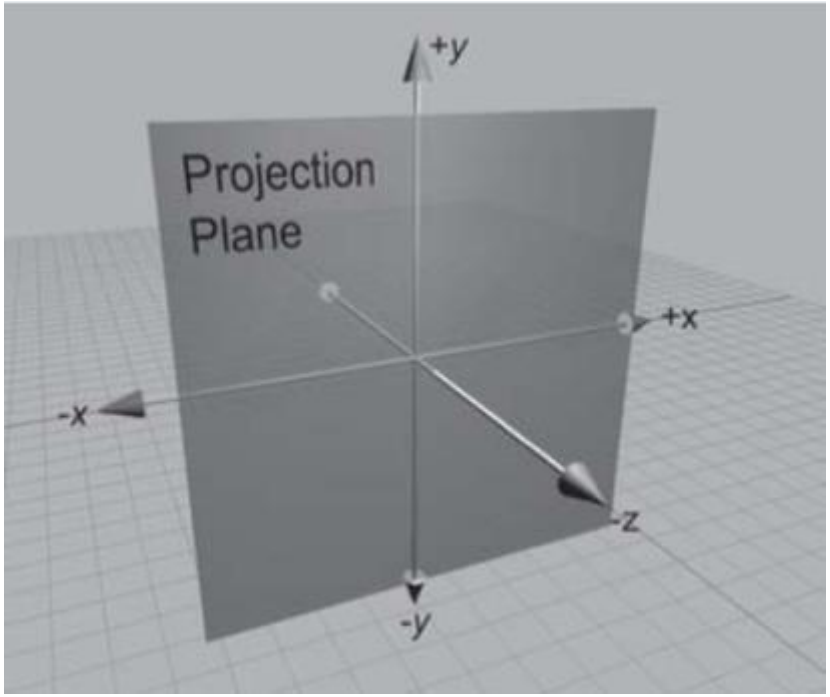
Modelo de cámara

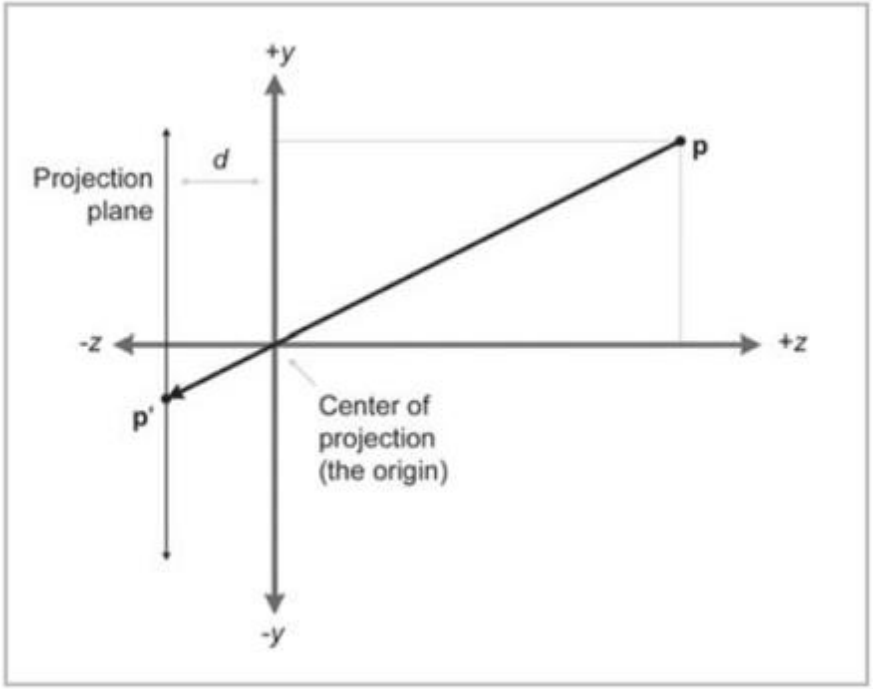


Modelo de cámara en perspectiva

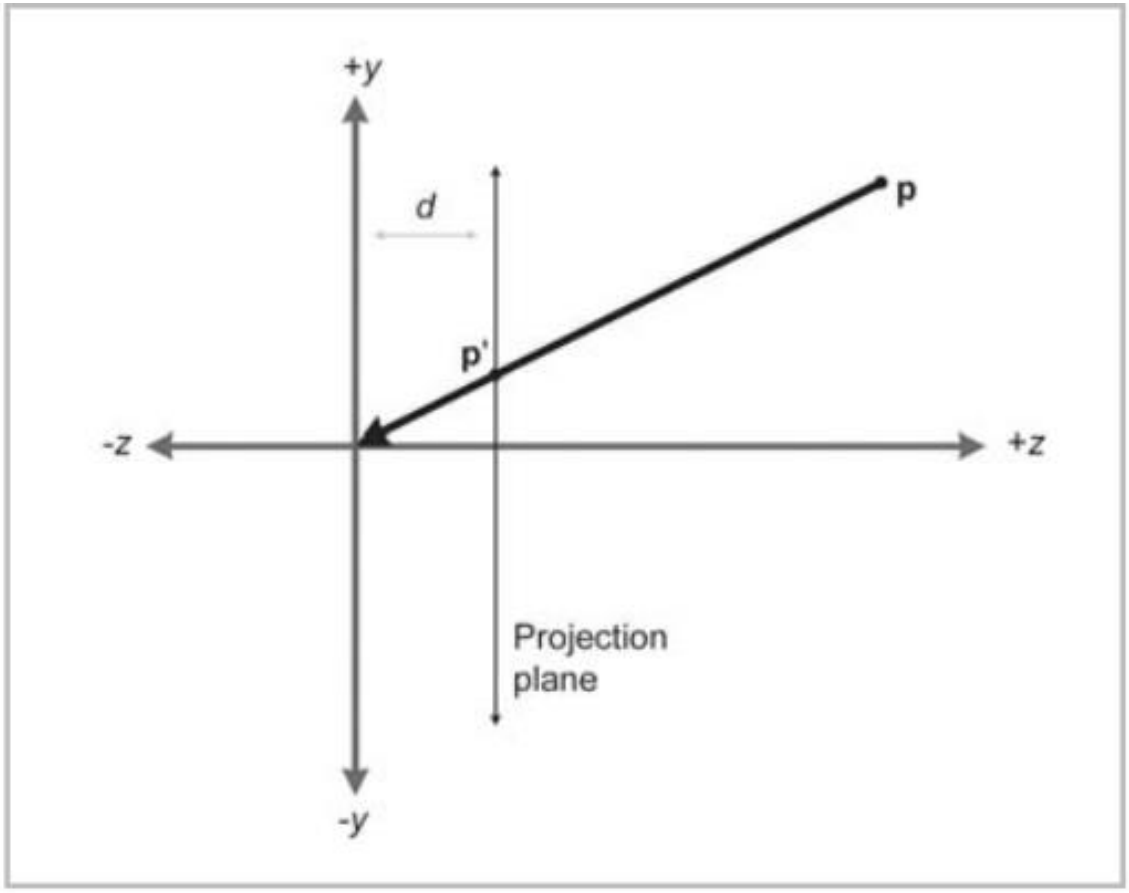


Plano de proyección

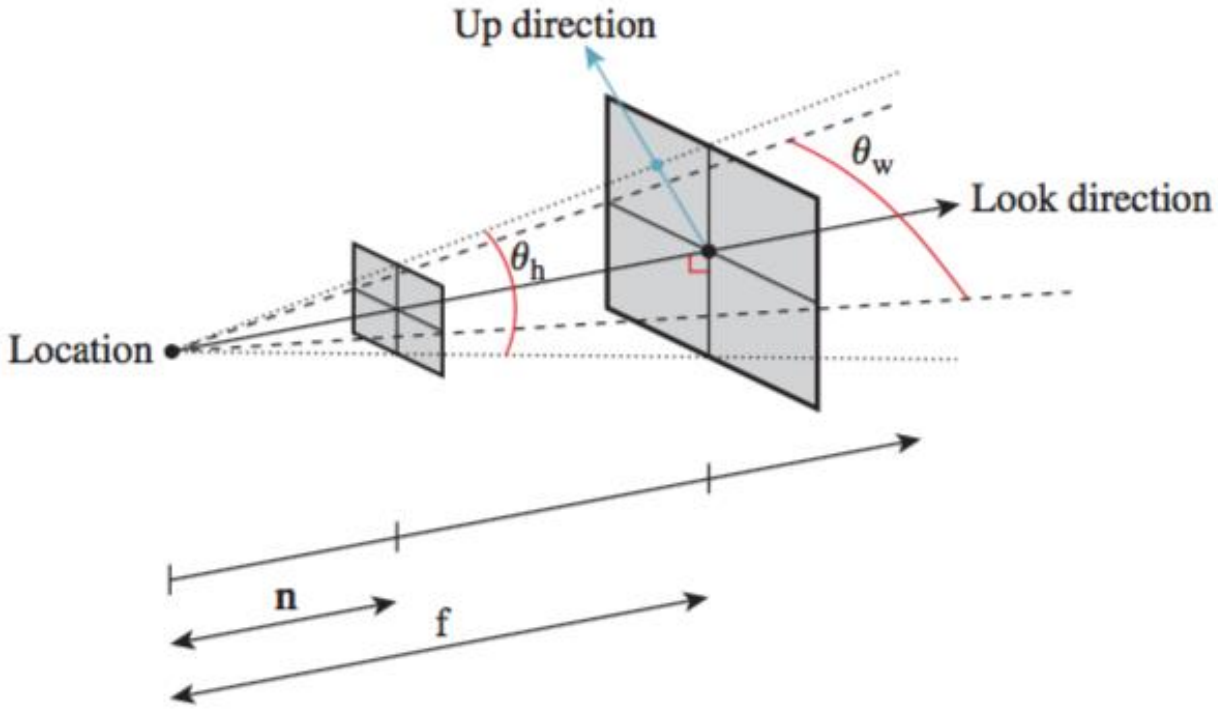




Plano de proyección

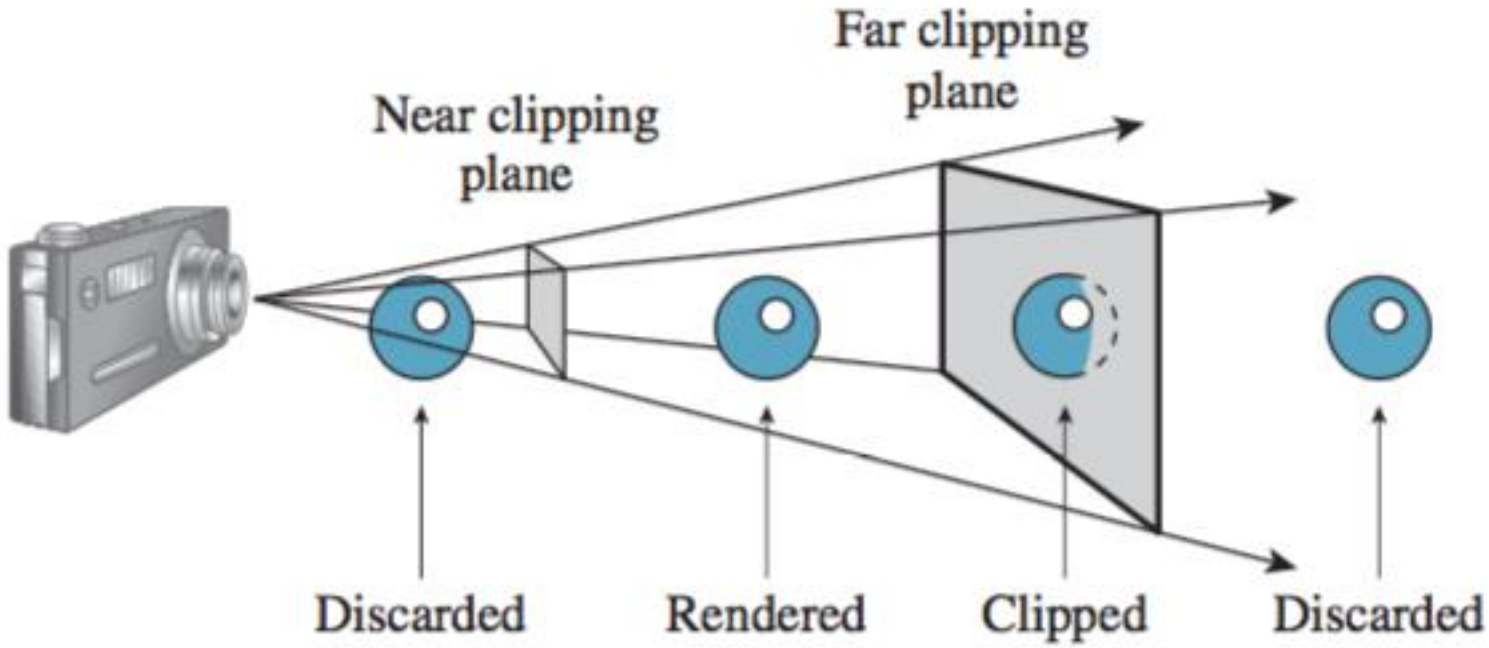


Proyección perspectiva



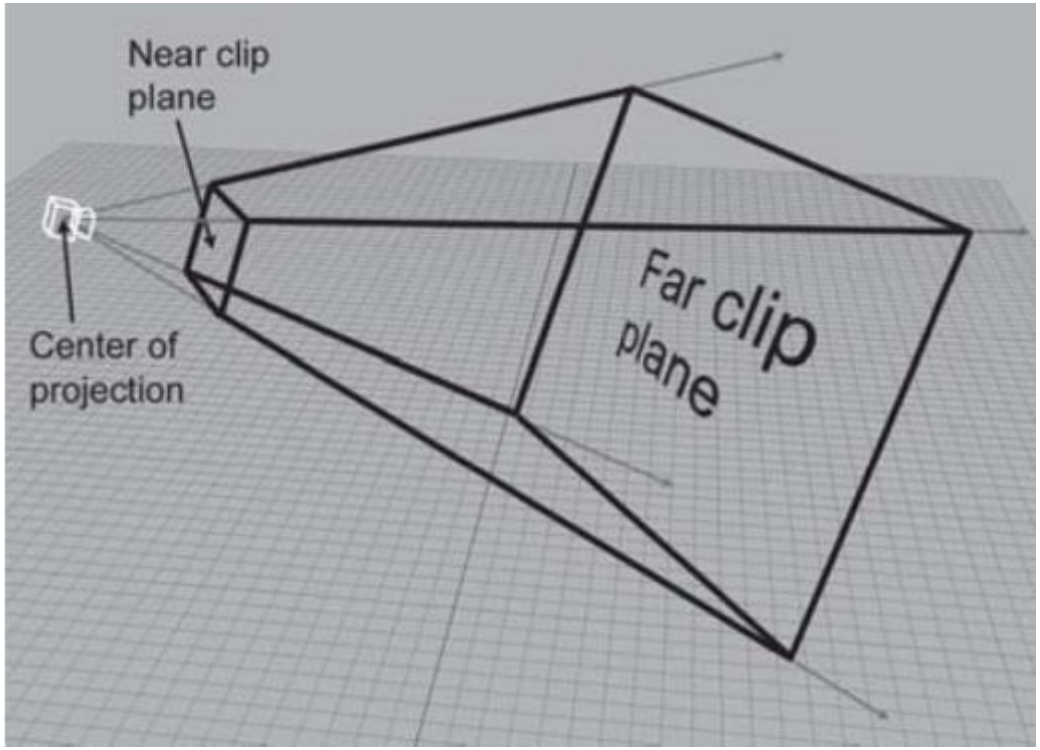
Clipping planes

Pirámide recortada.

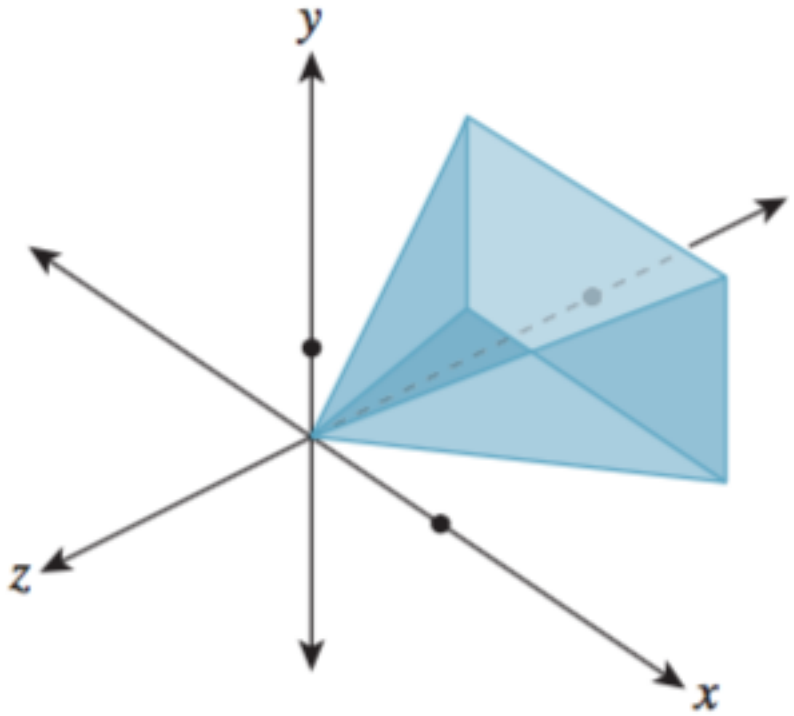


El frustum

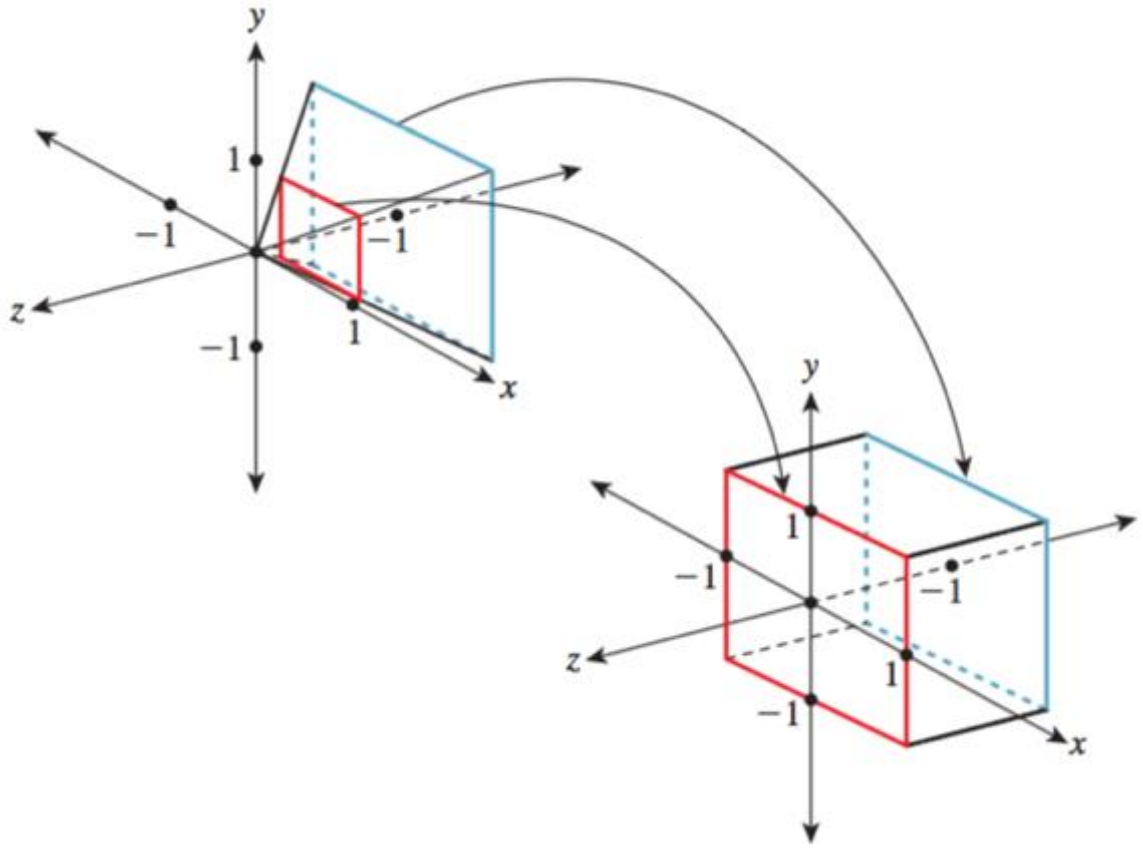
Formado por 6 planos (clipping planes):



Standard perspective view volume

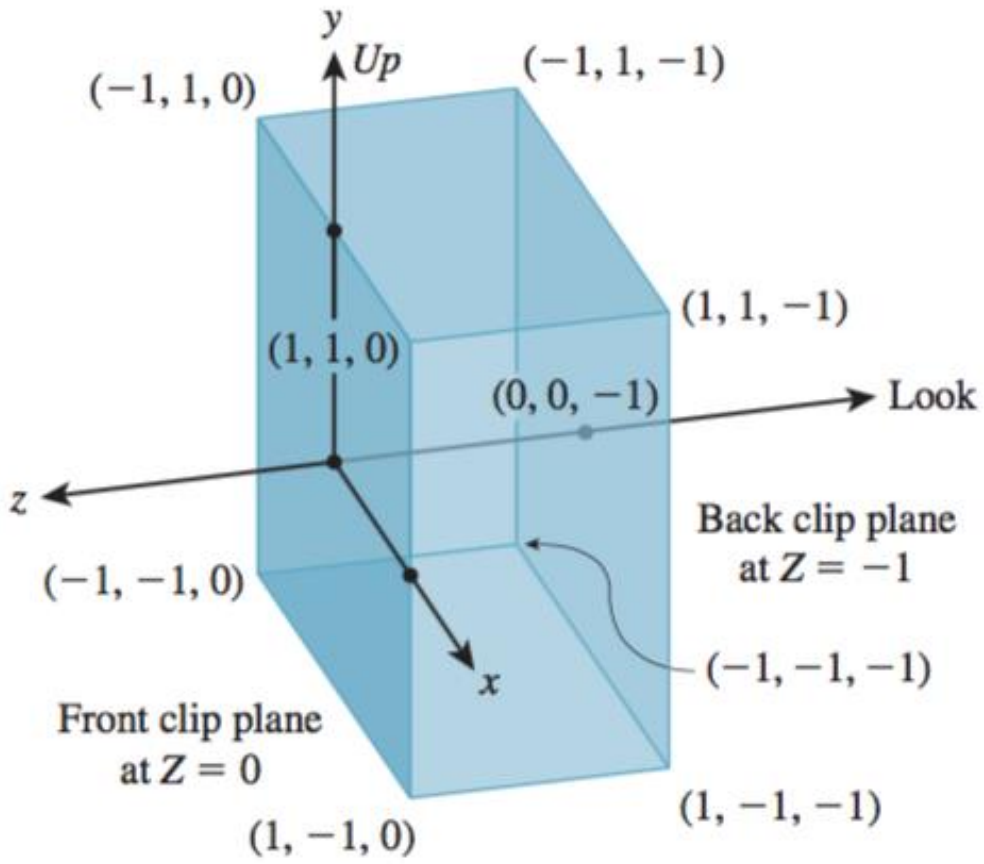


Transformación entre modelos: Matriz de proyección



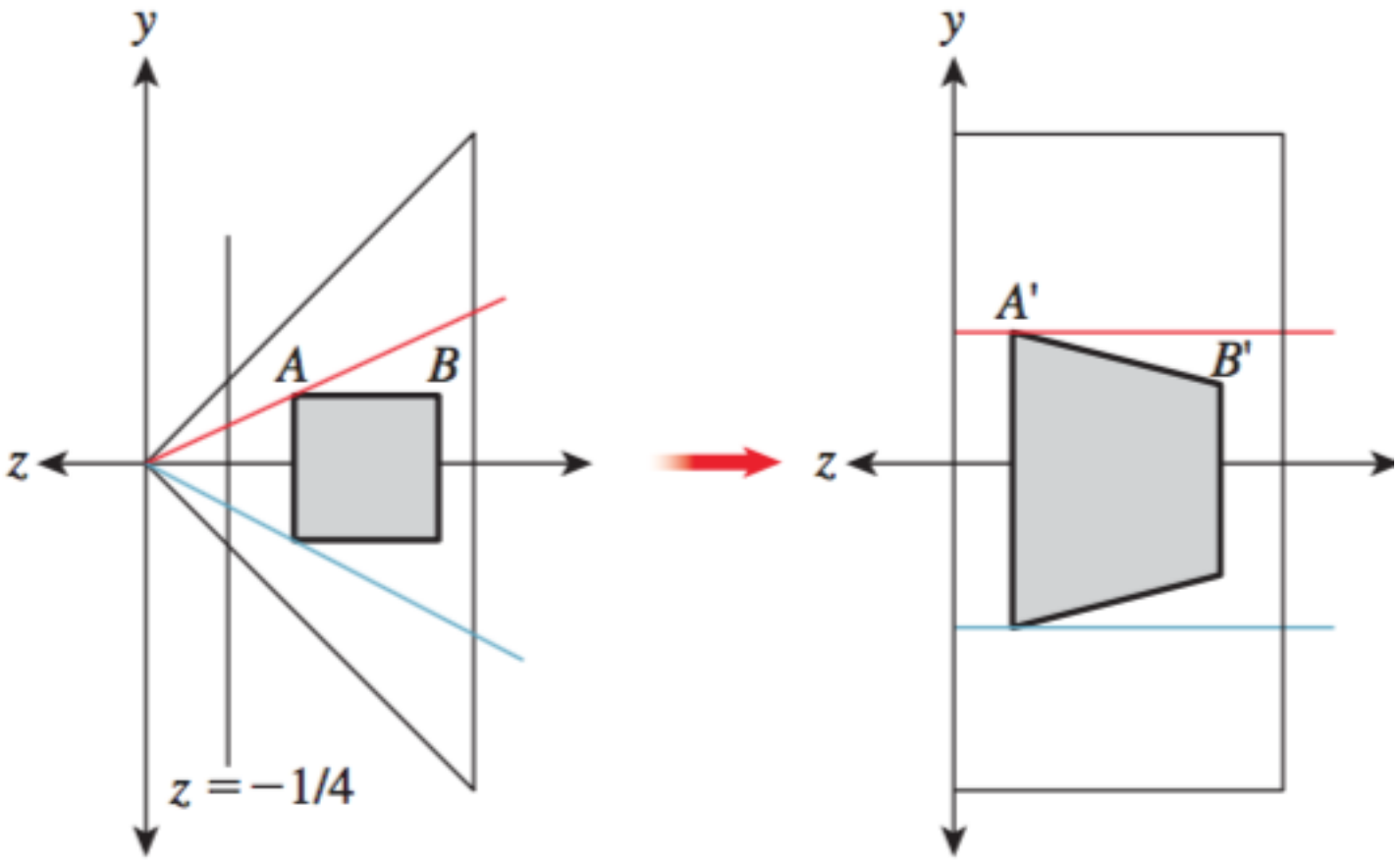
Volumen canónico de visualización (DirectX).

Standard parallel view model

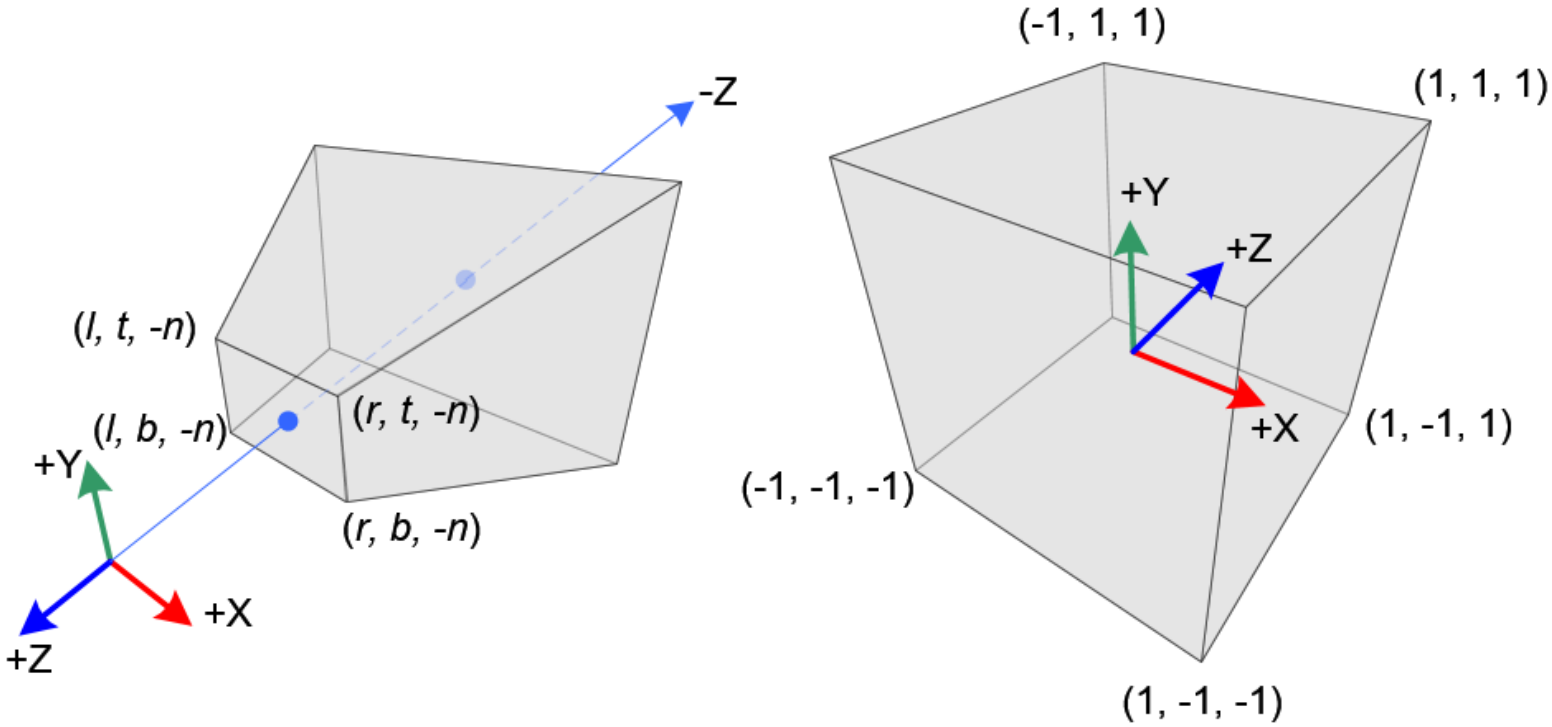


Volumen canónico de visualización (DirectX).

Transformación entre modelos

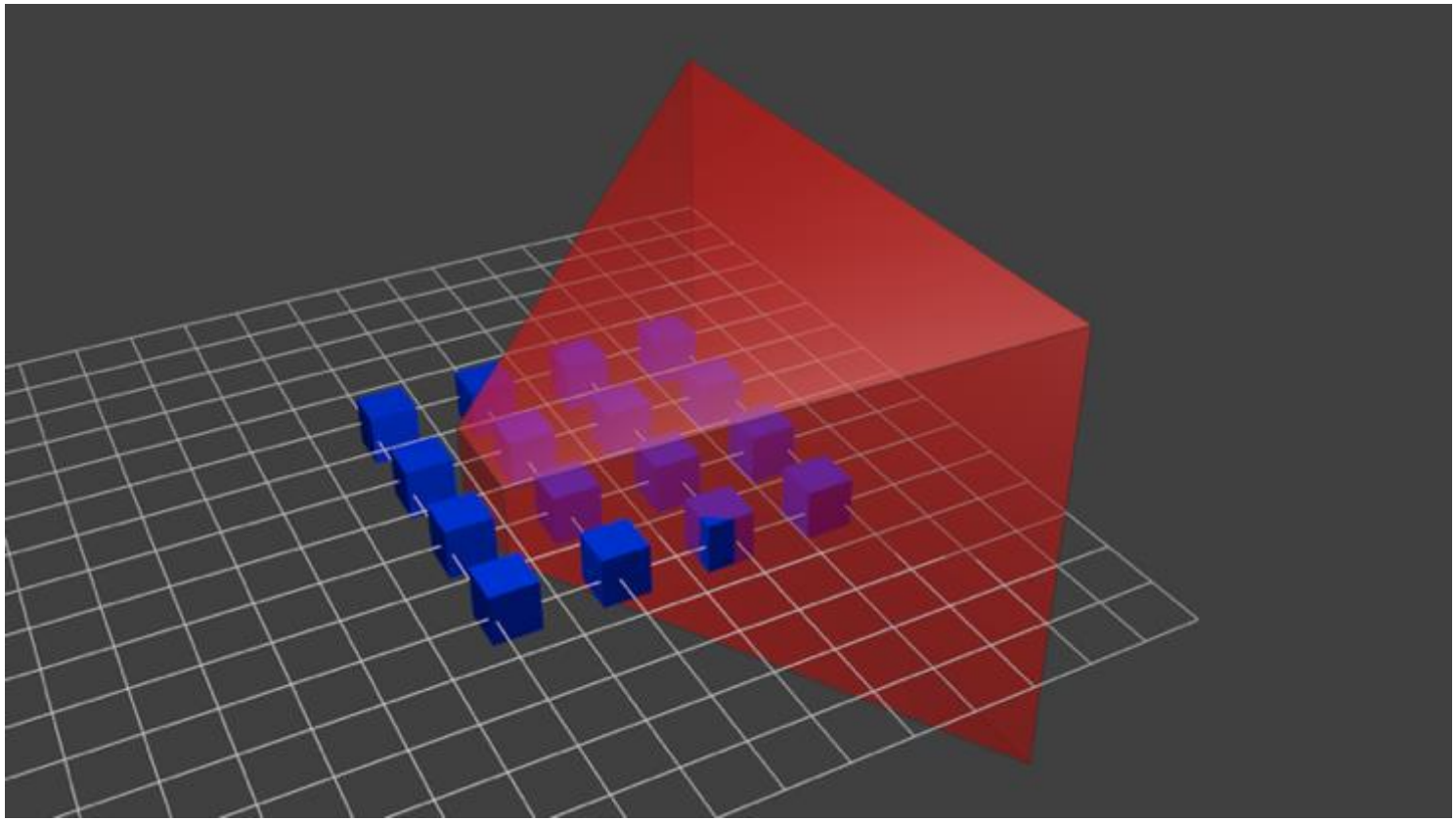


Transformación entre modelos



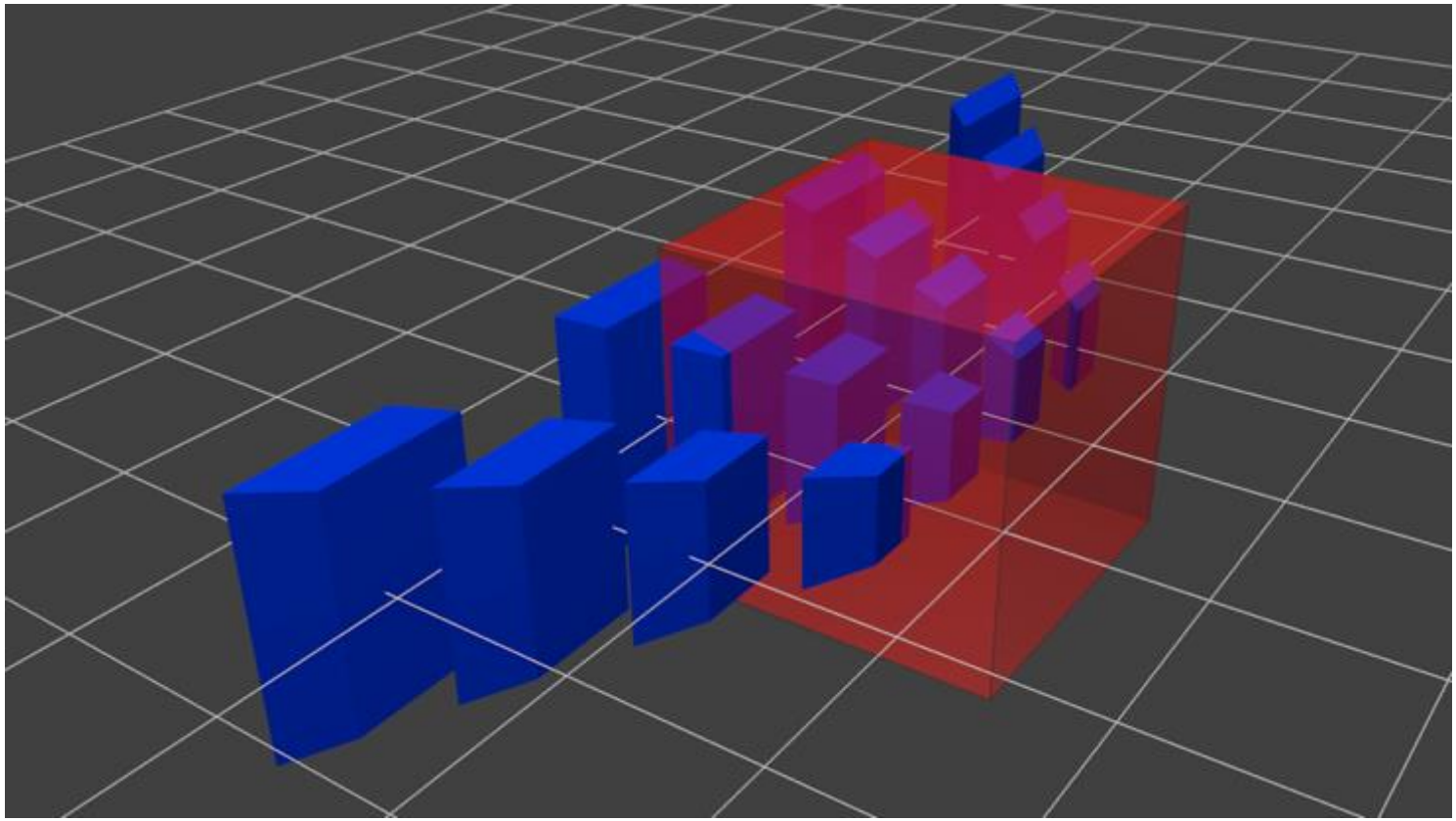
Volumen canónico de visualización (OpenGL).

Transformación entre modelos



Tomado de: http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/

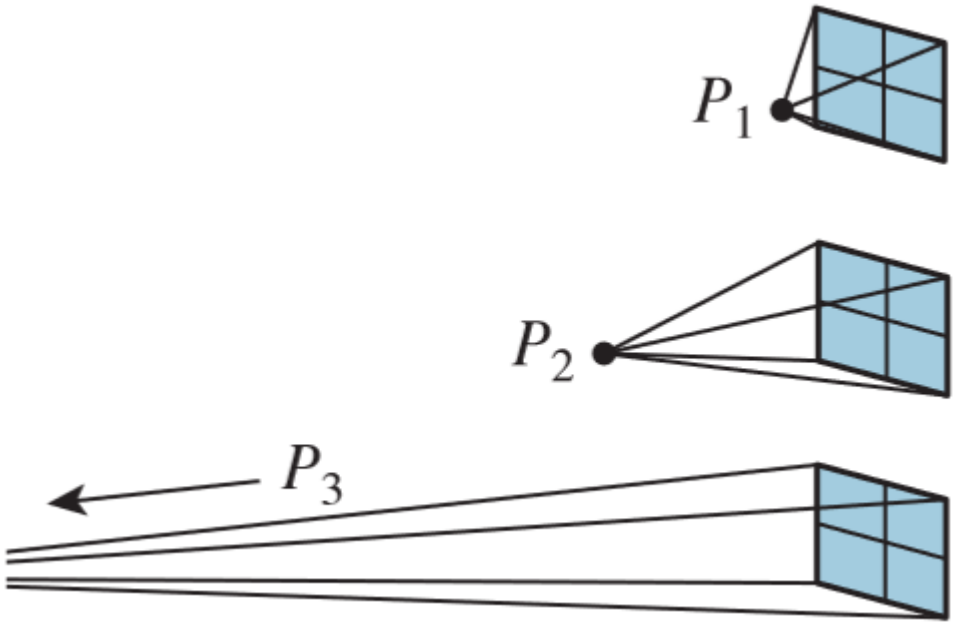
Transformación entre modelos



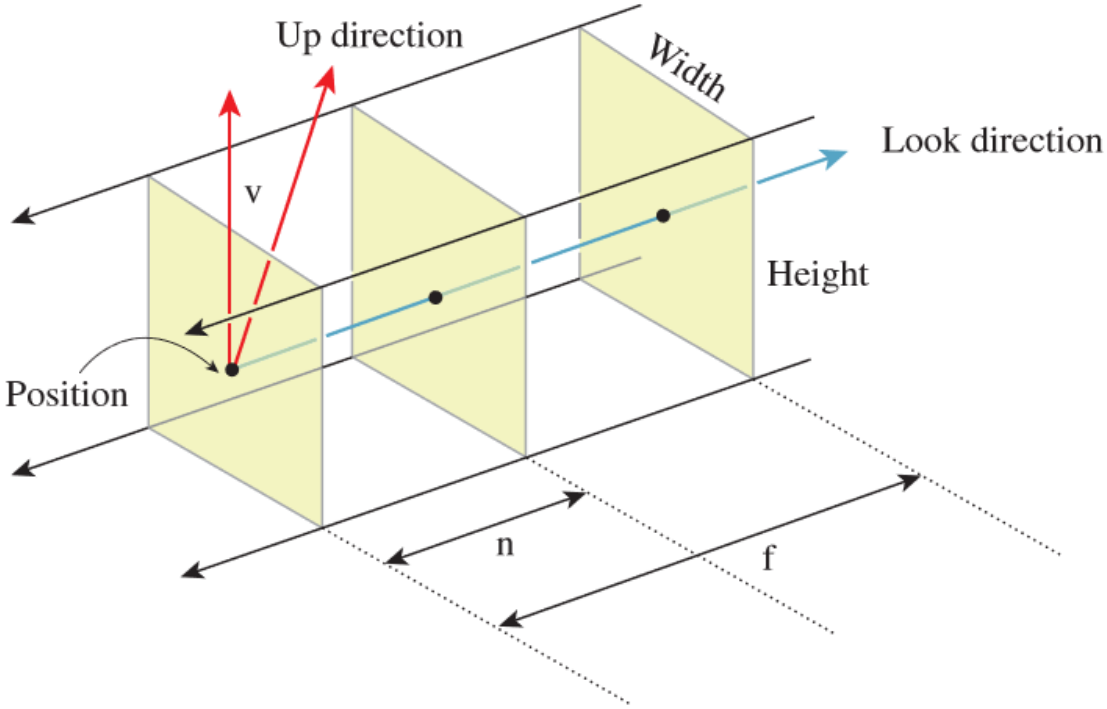
Tomado de: http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/

Cámara ortográfica: Proyección paralela

Todas las líneas llegan al plano de proyección de manera paralela. Las coordenadas del “ojo” se alejan hasta el infinito.

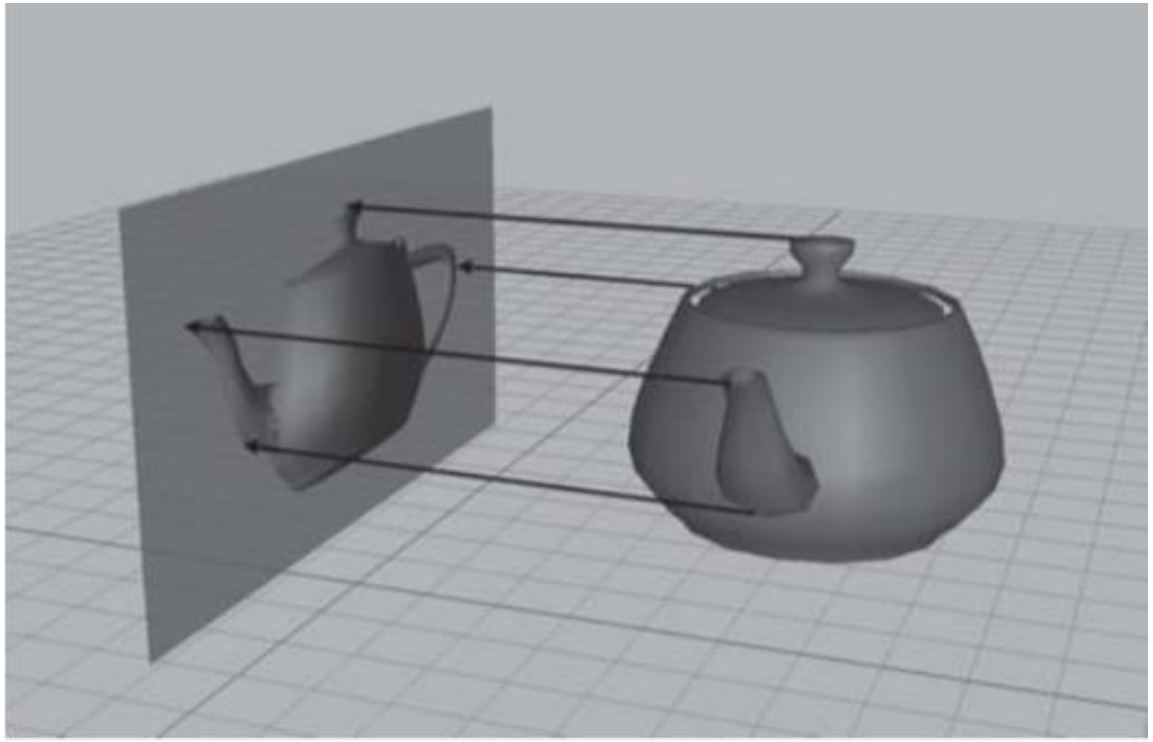


Modelo de cámara ortográfica



No es un tipo de cámara existente en el mundo real.

Proyección ortográfica

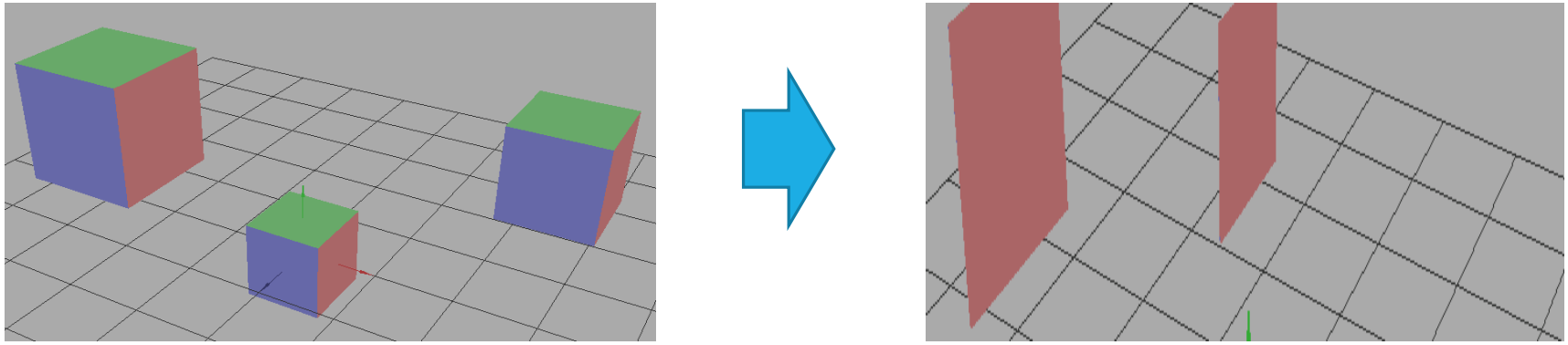


Proyección ortográfica

Corresponde a igualar a cero (o descartar) una de las dimensiones del objeto, básicamente llevando todos los elementos a un plano.

De esta manera el objeto se convierte de 3D a 2D.

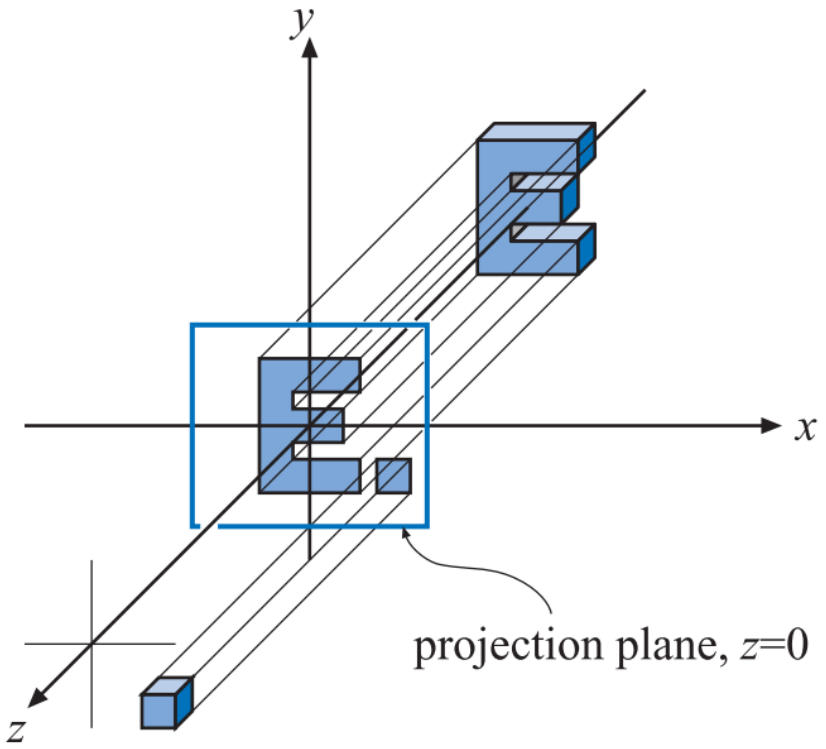
$$P_{xy} = \mathbf{S}([0 \quad 0 \quad 1], 0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



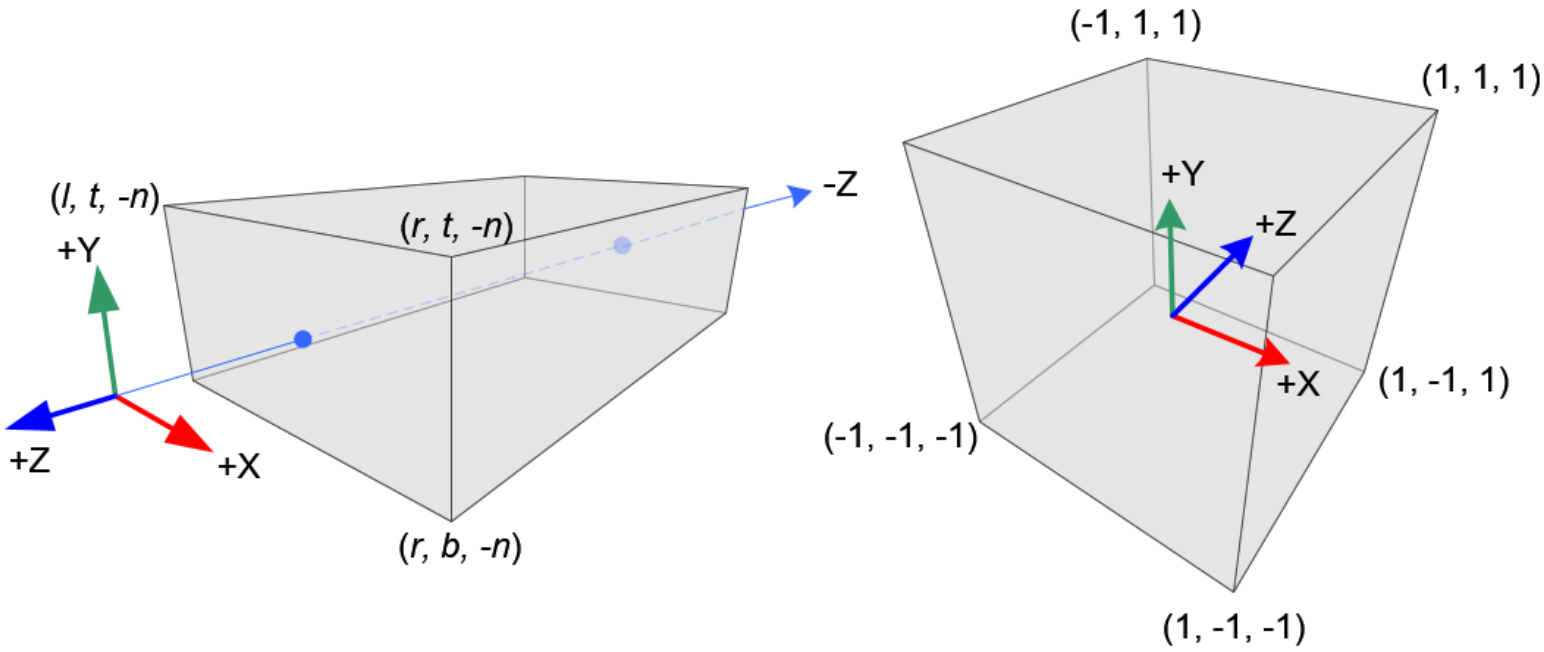
Proyección ortográfica: Matriz de proyección

Ortográfica con respecto al eje z (Esta operación no es reversible)

$$\mathbf{P}_o = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



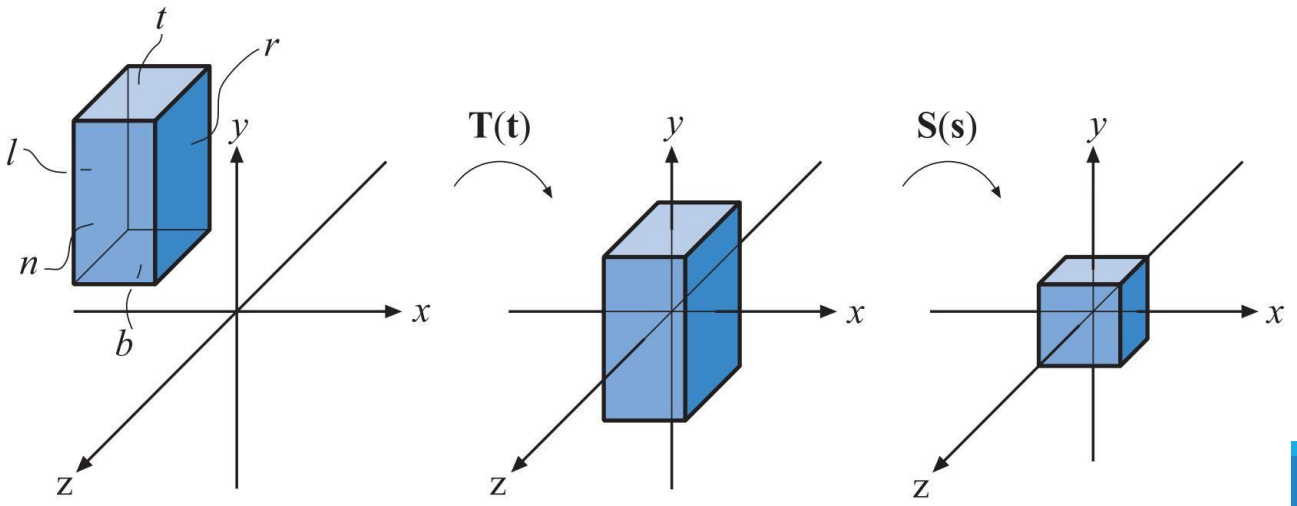
Transformación a Volumen Canónico de Visualización



Volumen canónico de visualización (OpenGL).

Transformación ortográfica
Volumen de visualización canónico

1. Alinear el volumen de visualización a los ejes (AABB – Axis Aligned Bounding Box). Este paso implica rotación en los 3 ejes y se hace previamente.
2. Trasladar la AABB para que su origen coincida con el del origen de coordenadas aplicando **T(s)**
3. Escalar para obtener el tamaño del volumen de visualización canónico aplicando **S(s)**



Transformación ortográfica
Volumen de visualización canónico

Trasladar la **AABB** para que su origen coincida con el del origen de coordenadas aplicando **T(s)**

$$\mathbf{T(s)} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -\frac{l+r}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformación ortográfica
Volumen de visualización canónico

Dado un **AABB** en el origen, escalar para obtener el tamaño del volumen de visualización canónico aplicando **S(s).**

$$\mathbf{S}(\mathbf{s}) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformación ortográfica
Volumen de visualización canónico

Fíjese que se trata de la concatenación entre una traslación y una escala.

$$\mathbf{P}_o = \mathbf{S}(\mathbf{s})\mathbf{T}(\mathbf{s}) = \underbrace{\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{S}(\mathbf{s})} \underbrace{\begin{pmatrix} 1 & 0 & 0 & -\frac{l+r}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{T}(\mathbf{s})}$$

Transformación ortográfica
Volumen de visualización canónico

Para obtener el volumen de visualización ortográfico se aplica:

$$\mathbf{P}_o = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(OpenGL)

Al aplicar esta transformada a un punto, el componente w de ese vector será diferente de 1. ***Se debe homogeneizar dividiendo entre w para obtener la proyección sobre el plano.***

Transformación ortográfica
Volumen de visualización canónico

Para obtener el volumen de visualización ortográfico en DirectX, se debe desplaza y escalar en la coordenada z, se aplica:

$$\mathbf{P}_o = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{1}{f-n} & -\frac{n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(DirectX)

Proyección ortográfica: Matriz de proyección perspectiva

Perspectiva con respecto al eje z (asumiendo que se desea proyectar sobre un plano ubicado en z = -d). (Esta operación no es reversible).

$$\mathbf{T}(\mathbf{s}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{pmatrix}$$

Transformación de perspectiva
Volumen de visualización canónico

Transforma el frustum en un cubo unitario.

$$\mathbf{P}_p = \begin{pmatrix} \frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

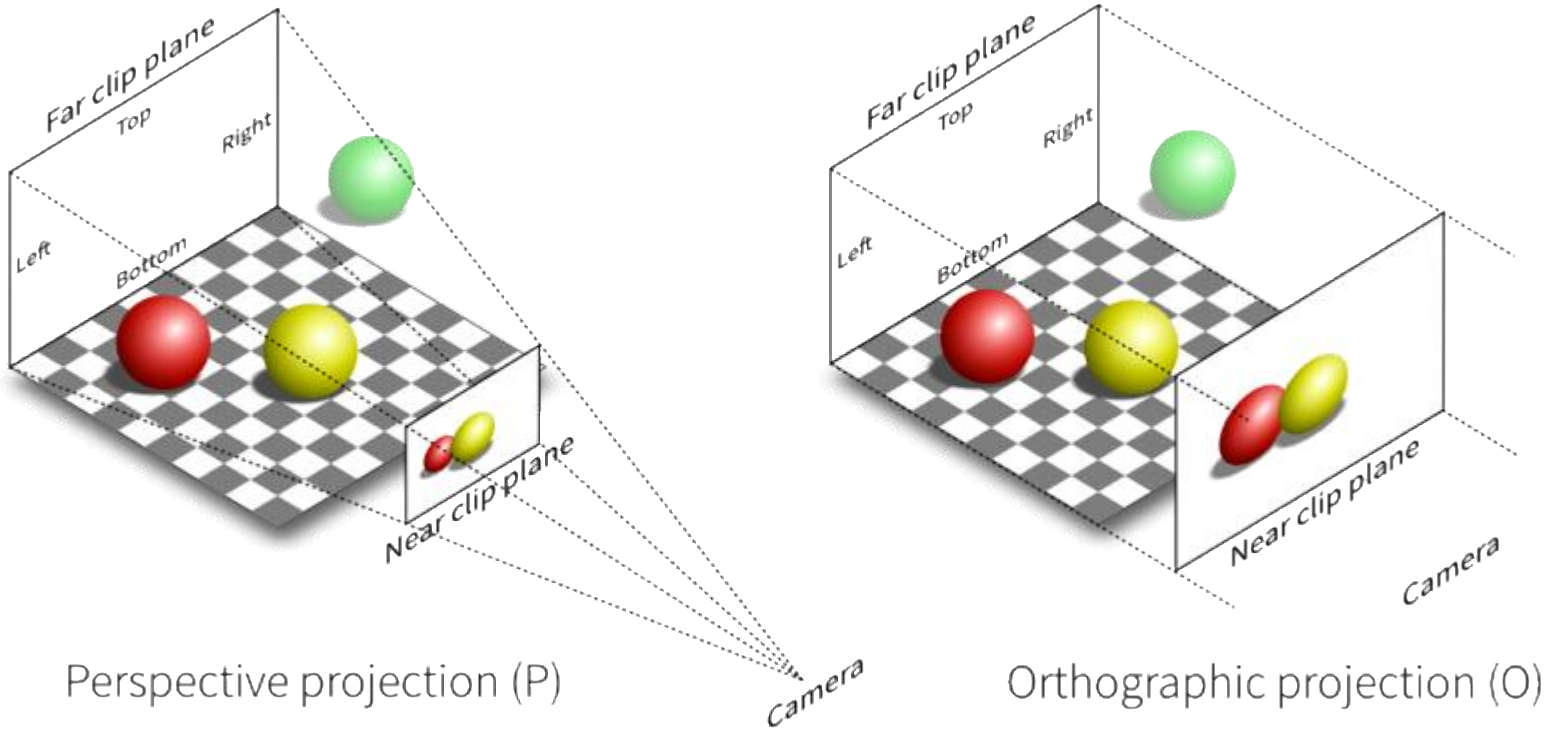
Al aplicar esta transformada a un punto, el componente w de ese vector será diferente de 1. ***Se debe homogeneizar dividiendo entre w para obtener la proyección sobre el plano.***

Transformación de perspectiva Volumen de visualización canónico

Transforma el frustum en un cubo unitario si el plano lejano está en el infinito:

$$\mathbf{P}_p = \begin{pmatrix} \frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & 1 & -2n \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Resumen:
Proyección perspectiva y ortográfica



Tomado de: http://glumpy.readthedocs.io/en/latest/tutorial/cube-ugly.html

Tarea

1. Revisar el ejemplo de cámara ortográfica en Three.js: https://threejs.org/examples/?q=ort#canvas_camera_orthographic
2. Revisar los ejemplos de cámara ortográfica y en perspectiva en el aula virtual.
3. Realizar el código según el grupo asignado.
4. Ver el video "The camera transform"

Bibliografía

Dunn, F. y Parberry, I. (2002). Chapter 15: 3D Math for Graphics en: ***3D Math Primer for Graphics and Game Development***. Wordware Publishing, Inc.

Hughes, J et al. (2014). Chapter 13: Camera Specifications and Transformations en: ***Computer Graphics: Principles and Practice***. 3rd Ed. Addison-Wesley.

Joy, Ken. (2009). ***Lecture 05: The pinhole camera model and associated transform matrix***. UC Davis

Tutorial 3: Matrices. http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/

Song Ho Ahn. (2018). OpenGL Projection Matrix. Tomado de: http://www.songho.ca/opengl/gl_projectionmatrix.html