Gernot Hoffmann

Planar Projections

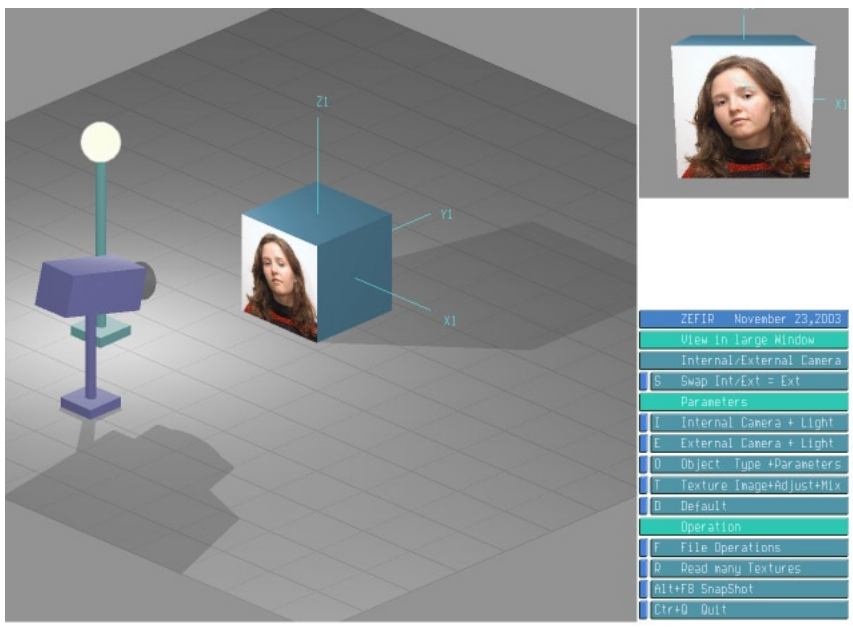


Table of Contents

1. Introduction

A projection from a 3D coordinate space onto a 2D view plane can be generated by a real camera as in photogrammetry, a fictitious camera as in computer graphics or by an ’abstract projection’, which is merely a set of functions.

A camera is defined by its position, three angles and a camera constant which is the distance between the center of projection and the view point along the view line. For a real camera the view line is the optical axis and the camera constant (*Kammerkonstante*) is the distance between the center of projection and the film (here the principle point, *Bildhauptpunkt*, the intersection of the optical axis and the film). A focal length is nowhere required.

The view plane is defined by its orientation and the view point. Mostly it is assumed that the view plane is orthogonal to the view line. But there are some exceptions: error models for real cameras and special projections like cavalier (cabinet) projection, military projection or a projection with rectified verticals. For real cameras: studio cameras are equipped with tiltable film planes.

The abstract projection is not based on a camera model. In this document both methods are combined. We call this here ’general projection’. The relations between the two models are missing in standard text books.

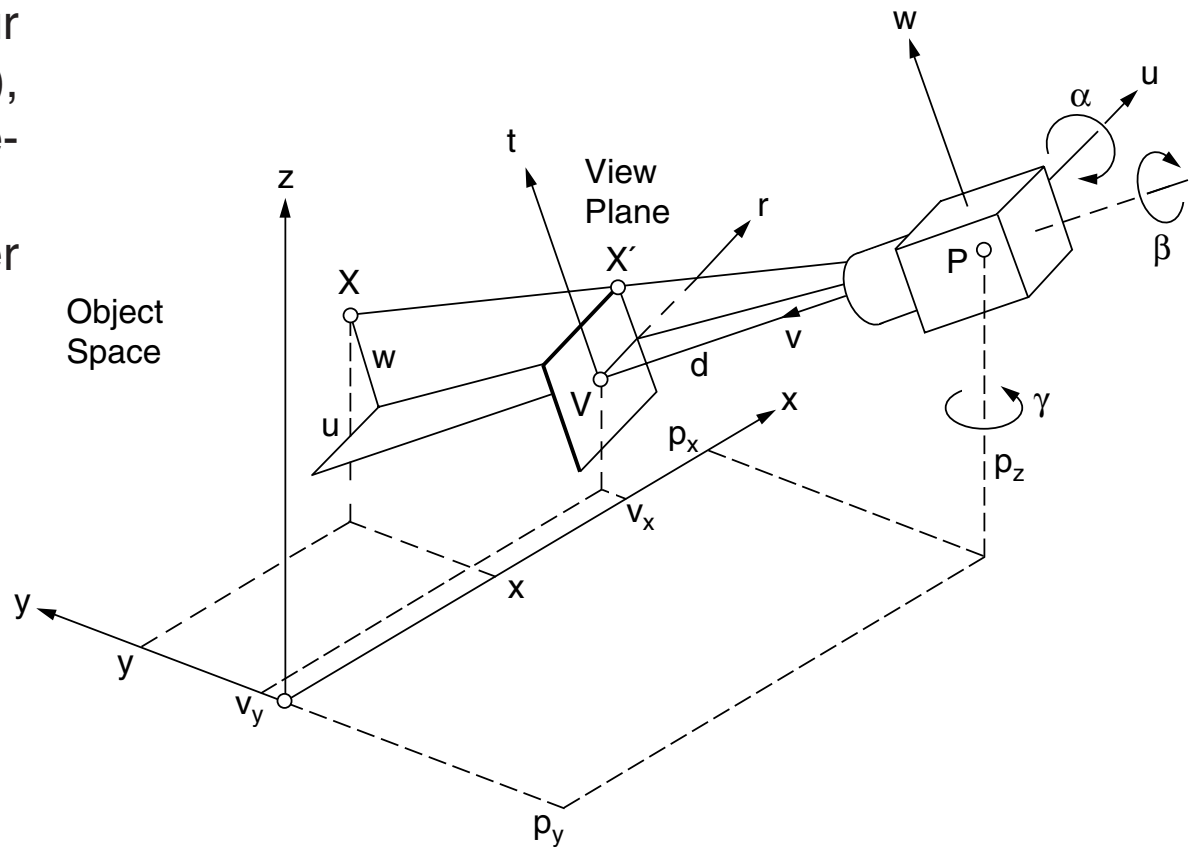
The view plane is a part of the object space. Object points on the view plane are mapped as they are onto the plane, but the coordinates are view plane coordinates. The view plane is infinite.
In a strict sense the view plane does not contain an image or a picture. The image is generated by mapping the content of a rectangular area of the view plane to a device.
This area was originally called a ’window’ (PHIGS [6]). In order to avoid confusion, the author would like to call the selected area a ’frame’.The frame is put on the view plane and the content is mapped to a ’viewport’ on the device. The frame can be defined by a frustum (view pyramide), but this is by no means necessary. For example, the frame can be excentric with respect to the center of the view plane, the view point.
This mapping has been called ’workstation transformation’. Not really convincing. The process should be called ’device transformation’.

All examples were programmed by PostScript. Powerful and very simple tools perform the device transformation. The device is a piece of paper. Results are defined with floating point accuracy and finally printed with the actual printer resolution or shown on a monitor.
Actually one may call this a sequence of device transformations. But the first step – mapping onto a piece of paper – is the essential one.

2.1 Camera Projection / Orthogonal View Plane

All objects including the camera are defined in the object spaxe $\mathbf{x}=\mathbf{x}_1$. P is the camera position, the center of projection. The camera points to the view point V. The view plane is centered in the view point and orthogonal to the view line. The camera coordinate system is $\mathbf{u}=\mathbf{x}_4$.
The camera is rotated from $\mathbf{x}_1$ to $\mathbf{x}_4$ by the Euler sequence γ, α, β. The angles γ and α are found by the view line direction, but β has to be defined explicitly. $\mathbf{x}_4$ is aligned with $\mathbf{x}_1$ for zero angles. This is the standard for ground based photogrammetry, also in the author's laboratory.

The angles are calculated using a four quadrant arctan function atan2(y,x), where y is the numerator and x the denominator. The signs are relevant.
d is the distance between the center of projection and the view point.



$$\tan(\gamma) = \frac{+(p_x - v_x)}{-(p_y - v_y)}$$

$$\tan(\alpha) = \frac{-(p_z - v_z)}{\sqrt{(p_x - v_x)^2 + (p_y - v_y)^2}}$$

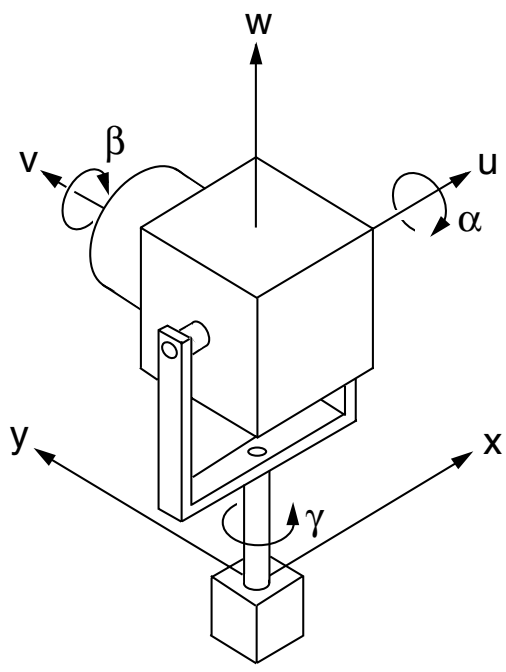
$$d = \sqrt{(p_x - v_x)^2 + (p_y - v_y)^2 + (p_z - v_z)^2}$$

The calculation for the coordinate rotation matrix is based on matrices for single axis rotati-

$$\mathbf{X}_{32} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

$$\mathbf{Y}_{43} = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

$$\mathbf{Z}_{21} = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\mathbf{x}_4 = \mathbf{C}_{41}\mathbf{x}_1 = \mathbf{Y}_{43}\mathbf{X}_{32}\mathbf{Z}_{21}\mathbf{x}_1$$

$$\mathbf{C}_{41} = \begin{bmatrix} \cos(\beta)\cos(\gamma) - \sin(\alpha)\sin(\beta)\sin(\gamma) & \cos(\beta)\sin(\gamma) + \sin(\alpha)\sin(\beta)\cos(\gamma) & -\cos(\alpha)\sin(\beta) \\ -\cos(\alpha)\sin(\gamma) & \cos(\alpha)\cos(\gamma) & \sin(\alpha) \\ \sin(\beta)\cos(\gamma) + \sin(\alpha)\cos(\beta)\sin(\gamma) & \sin(\beta)\sin(\gamma) - \sin(\alpha)\cos(\beta)\cos(\gamma) & \cos(\alpha)\cos(\beta) \end{bmatrix}$$

The coordinates r and t in the view plane are found by simple proportions.

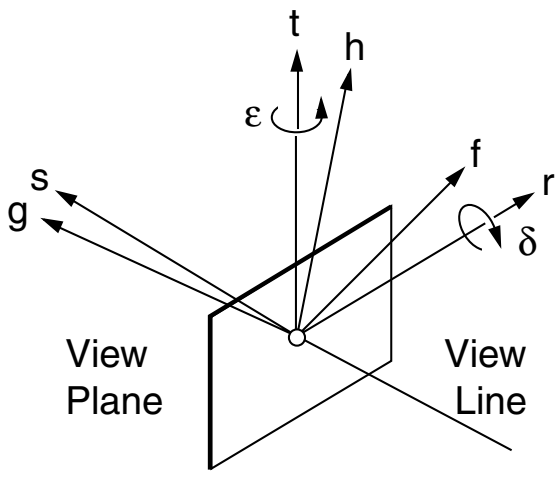
$$\mathbf{u} = (u,v,w)^T = \mathbf{C}_{41}(\mathbf{x} - \mathbf{p})$$

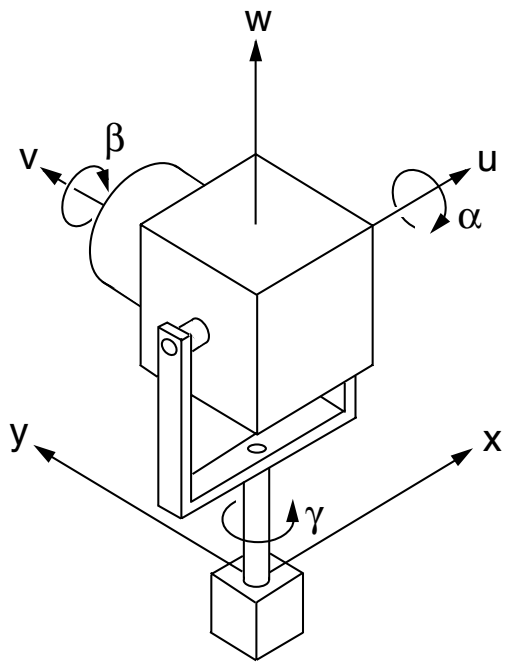
$$r = \frac{u}{v}d$$

$$t = \frac{w}{v}d$$

2.2 Camera Projection / Tilted View Plane

This chapter is included for historical reasons. The transition between the orthogonal view plane r,t and the tilted view plane f,h was introduced by a rotation using two angles δ and ε. First δ about the r-axis, then ε about the new t-axis. This sequence is defined for a reversed order, compared to the camera gimbal angles. The matrix T_{41} describes the transformation from the orthogonal view plane r,s,t to the tilted view plane f,g,h.
Tilted view planes (film planes) are used for studio cameras like *Sinar*.





$$\mathbf{T}_{41} = \begin{bmatrix} \cos(\varepsilon) & \cos(\delta)\sin(\varepsilon) & \sin(\delta)\sin(\varepsilon) \\ -\sin(\varepsilon) & \cos(\delta)\cos(\varepsilon) & \sin(\delta)\cos(\varepsilon) \\ 0 & -\sin(\delta) & \cos(\delta) \end{bmatrix}$$

$$D = 1 - (r/d)\frac{\tan(\varepsilon)}{\cos(\delta)} + (t/d)\tan(\delta)$$

$$f = \frac{r/\cos(\varepsilon)}{D}$$

$$h = \frac{t/\cos(\delta) - r\tan(\varepsilon)\tan(\delta)}{D}$$

Standard applications:

Vertical rectification	$\delta = -\alpha$	$\varepsilon = 0$
Undistorted front face	$\delta = -\alpha$	$\varepsilon = -\gamma$

3. Abstract Projection

The abstract projection is defined by a set of functions, so far without a camera. The view plane coordinates r,t are replaced by f,h. This indicates that the view plane is not necessarily orthogonal to the view line. These formulas can be found e.g. in [4] but also in other books.

$$f = \frac{a_0 + \mathbf{a}^T\mathbf{x}}{1 + \mathbf{c}^T\mathbf{x}}$$

$$h = \frac{b_0 + \mathbf{b}^T\mathbf{x}}{1 + \mathbf{c}^T\mathbf{x}}$$

The parameters for the abstract projection are determined by mapping six points $\mathbf{x}_i$ from the object space to known six points points f_i, h_i in the view plane.
This results in twelve linear equations for the eleven unknowns **q**. The Gauss Normal Equation (multiplication by the transposed system matrix $\mathbf{M}^T$) shrinks the system to eleven equations. The new matrix $\mathbf{M}^T\mathbf{M}$ is symmetric. A special equation solver like *Cholesky* can be used, but for only eleven equations a generic *Gaussian* solver with pivot element optimization is suited as well.

$$f(1 + \mathbf{c}^T\mathbf{x}) = a_0 + \mathbf{a}^T\mathbf{x}$$
$$h(1 + \mathbf{c}^T\mathbf{x}) = b_0 + \mathbf{b}^T\mathbf{x}$$

$$a_0 + \mathbf{a}^T\mathbf{x} + 0 + \mathbf{0}^T\mathbf{x} - f\,\mathbf{c}^T\mathbf{x} = f$$
$$0 + \mathbf{0}^T\mathbf{x} + b_0 + \mathbf{b}^T\mathbf{x} - h\,\mathbf{c}^T\mathbf{x} = h$$

$$\mathbf{q} = (a_0, a_x, a_y, a_z, b_0, b_x, b_y, b_z, c_x, c_y, c_z)^T$$

$$\mathbf{M} = \begin{bmatrix}
1 & x_1 & y_1 & z_1 & 0 & 0 & 0 & 0 & -f_1 x_1 & -f_1 y_1 & -f_1 z_1 \\
1 & x_2 & y_2 & z_2 & 0 & 0 & 0 & 0 & -f_2 x_2 & -f_2 y_2 & -f_2 z_2 \\
1 & x_3 & y_3 & z_3 & 0 & 0 & 0 & 0 & -f_3 x_3 & -f_3 y_3 & -f_3 z_3 \\
1 & x_4 & y_4 & z_4 & 0 & 0 & 0 & 0 & -f_4 x_4 & -f_4 y_4 & -f_4 z_4 \\
1 & x_5 & y_5 & z_5 & 0 & 0 & 0 & 0 & -f_5 x_5 & -f_5 y_5 & -f_5 z_5 \\
1 & x_6 & y_6 & z_6 & 0 & 0 & 0 & 0 & -f_6 x_6 & -f_6 y_6 & -f_6 z_6 \\
0 & 0 & 0 & 0 & 1 & x_1 & y_1 & z_1 & -h_1 x_1 & -h_1 y_1 & -h_1 z_1 \\
0 & 0 & 0 & 0 & 1 & x_2 & y_2 & z_2 & -h_2 x_2 & -h_2 y_2 & -h_2 z_2 \\
0 & 0 & 0 & 0 & 1 & x_3 & y_3 & z_3 & -h_3 x_3 & -h_3 y_3 & -h_3 z_3 \\
0 & 0 & 0 & 0 & 1 & x_4 & y_4 & z_4 & -h_4 x_4 & -h_4 y_4 & -h_4 z_4 \\
0 & 0 & 0 & 0 & 1 & x_5 & y_5 & z_5 & -h_5 x_5 & -h_5 y_5 & -h_5 z_5 \\
0 & 0 & 0 & 0 & 1 & x_6 & y_6 & z_6 & -h_6 x_6 & -h_6 y_6 & -h_6 z_6
\end{bmatrix} \qquad \mathbf{f} = \begin{bmatrix}
f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6
\end{bmatrix}$$

$$\mathbf{M}\mathbf{q} = \mathbf{f}$$

$$\mathbf{M}^T\mathbf{M}\,\mathbf{q} = \mathbf{M}^T\mathbf{f}$$

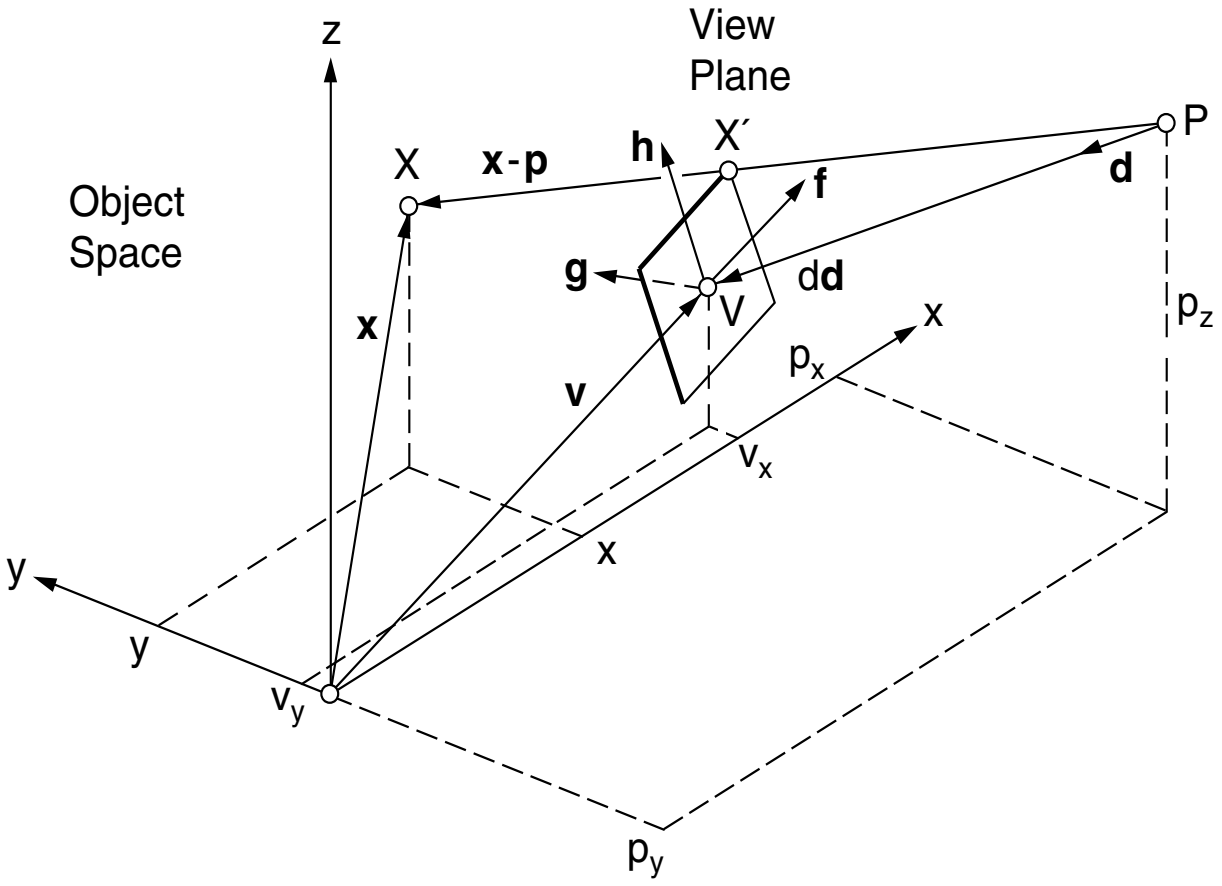
The six points in the object space are mostly chosen on the corners of a unit cube. Common parallel projections can be defined easily.

The formulas above can be used for perspective 2D to 2D mapping as well [10].
Then **x**, **a**, **b**, **c** are column matrices with two components. The system has eight unknowns, only four points are necessary and the Gauss Normal Equation is obsolete.

4.1.1 General Projection / Perspective

All objects including the camera are defined in the object space **x**. The camera position is the center of projection P. The camera points to the viewpoint V. The distance vector is (d**d**) = **v** - **p**. **d** is a unit vector which defines the direction of the view line.
The view plane is centered in the view point **v** and not generally orthogonal to the view line. The view plane is defined by two unit base vectors **f** and **h**. The normal vector **g** = -(**f** x **h**) is not necessarily a unit vector but always orthogonal to the view plane.
The essential idea is: the view plane can be fixed though the camera is moving. The equivalent abstract projection has to be found.

The ray through the arbitrary object space point X and the center of projection P is defined by a parametric equation.
The point X' in the view plane is defined by two parameters f and h as the intersection of the ray and the plane.
Both vectors $\mathbf{x}_r$ and $\mathbf{x}_p$ refer to X'.
Cramer's rule delivers the parameters in a notation with dot products and cross products.



$$\mathbf{x}_r = \mathbf{p} + \gamma(\mathbf{x}-\mathbf{p})$$
$$\mathbf{x}_p = \mathbf{v} + f\,\mathbf{f} + h\,\mathbf{h}$$

$$\mathbf{x}_r = \mathbf{x}_p$$
$$f\,\mathbf{f} + h\,\mathbf{h} - \gamma(\mathbf{x}-\mathbf{p}) = \mathbf{p}-\mathbf{v} = -d\,\mathbf{d}$$

$$D = \begin{vmatrix} f_x & h_x & -(x-p_x) \\ f_y & h_y & -(y-p_y) \\ f_z & h_z & -(z-p_z) \end{vmatrix} = -(\mathbf{f}\times\mathbf{h})^T(\mathbf{x}-\mathbf{p}) = \mathbf{g}^T(\mathbf{x}-\mathbf{p})$$

$$D_f = \begin{vmatrix} -d\cdot d_x & h_x & -(x-p_x) \\ -d\cdot d_y & h_y & -(y-p_y) \\ -d\cdot d_z & h_z & -(z-p_z) \end{vmatrix} = d(\mathbf{d}\times\mathbf{h})^T(\mathbf{x}-\mathbf{p})$$

$$D_h = \begin{vmatrix} f_x & -d\cdot d_x & -(x-p_x) \\ f_y & -d\cdot d_y & -(y-p_y) \\ f_z & -d\cdot d_z & -(z-p_z) \end{vmatrix} = d(\mathbf{f}\times\mathbf{d})^T(\mathbf{x}-\mathbf{p})$$

$$f = \frac{d(\mathbf{d}\times\mathbf{h})^T(\mathbf{x}-\mathbf{p})}{\mathbf{g}^T(\mathbf{x}-\mathbf{p})}$$

$$h = \frac{d(\mathbf{f}\times\mathbf{d})^T(\mathbf{x}-\mathbf{p})}{\mathbf{g}^T(\mathbf{x}-\mathbf{p})}$$

4.1.2 General Projection / Perspective

Now the result of the previous calculations is matched with the abstract projection. Here we encounter an odd problem: $\mathbf{g}^T\mathbf{p} = 0$. For the moment it is assumed that this dot product is not zero.

$$f = \frac{d(\mathbf{d}\times\mathbf{h})^T(\mathbf{x}-\mathbf{p})}{\mathbf{g}^T(\mathbf{x}-\mathbf{p})} = \frac{-d(\mathbf{d}\times\mathbf{h})^T\mathbf{p}+d(\mathbf{d}\times\mathbf{h})^T\mathbf{x}}{-\mathbf{g}^T\mathbf{p}+\mathbf{g}^T\mathbf{x}}$$

$$h = \frac{d(\mathbf{f}\times\mathbf{d})^T(\mathbf{x}-\mathbf{p})}{\mathbf{g}^T(\mathbf{x}-\mathbf{p})} = \frac{-d(\mathbf{f}\times\mathbf{d})^T\mathbf{p}+d(\mathbf{f}\times\mathbf{d})^T\mathbf{x}}{-\mathbf{g}^T\mathbf{p}+\mathbf{g}^T\mathbf{x}}$$

$$f = \frac{\frac{d(\mathbf{d}\times\mathbf{h})^T\mathbf{p}}{\mathbf{g}^T\mathbf{p}} - \frac{d(\mathbf{d}\times\mathbf{h})^T\mathbf{x}}{\mathbf{g}^T\mathbf{p}}}{1 - \frac{\mathbf{g}^T\mathbf{x}}{\mathbf{g}^T\mathbf{p}}} = \frac{a_0+\mathbf{a}^T\mathbf{x}}{1+\mathbf{c}^T\mathbf{x}}$$

$$h = \frac{\frac{d(\mathbf{f}\times\mathbf{d})^T\mathbf{p}}{\mathbf{g}^T\mathbf{p}} - \frac{d(\mathbf{f}\times\mathbf{d})^T\mathbf{x}}{\mathbf{g}^T\mathbf{p}}}{1 - \frac{\mathbf{g}^T\mathbf{x}}{\mathbf{g}^T\mathbf{p}}} = \frac{b_0+\mathbf{b}^T\mathbf{x}}{1+\mathbf{c}^T\mathbf{x}}$$

$$\mathbf{a} = -\frac{d(\mathbf{d}\times\mathbf{h})}{\mathbf{g}^T\mathbf{p}} \qquad a_0 = \frac{d(\mathbf{d}\times\mathbf{h})^T\mathbf{p}}{\mathbf{g}^T\mathbf{p}}$$

$$\mathbf{b} = -\frac{d(\mathbf{f}\times\mathbf{d})}{\mathbf{g}^T\mathbf{p}} \qquad b_0 = \frac{d(\mathbf{f}\times\mathbf{d})^T\mathbf{p}}{\mathbf{g}^T\mathbf{p}}$$

$$\mathbf{c} = -\frac{\mathbf{g}}{\mathbf{g}^T\mathbf{p}}$$

The case $\mathbf{g}^T\mathbf{p} = 0$ happens if the camera position is in the origin of the object space and for arbitrary positions if the camera position vector is orthogonal to the plane normal.
Correction for previous version:
This can easily happen. E.g. the camera is at $\mathbf{p}=(1,0,0)^T$, the view point at $\mathbf{v}=(1,1,0)^T$ and the plane normal is $\mathbf{g}=(0,1,0)^T$. The camera views in y-direction, the view plane is parallel to the xz-plane.

For an arbitrarily moving camera the case $\mathbf{p}=\mathbf{0}$ cannot be excluded. A simple solution would be the introduction of an intermediate translation $\mathbf{x}=\mathbf{x}+\mathbf{x}_0$.

A famous example for a center of projection in the origin is the CIE color transformation from XYZ to xyz. This is explained in chapter 9.

$$x = X/(X+Y+Z)$$
$$y = Y/(X+Y+Z)$$
$$z = Z/(X+Y+Z)$$

4.1.3 General Projection / Perspective

Obviously there is no need for a normalization by $\mathbf{g}^T\mathbf{p}$. Therefore we use a new abstract projection with c_0 in the denominator. Furtheron we have shifted the never vanishing distance d into the denominator.

$$f = \frac{-(\mathbf{d}\times\mathbf{h})^T\mathbf{p}+(\mathbf{d}\times\mathbf{h})^T\mathbf{x}}{-\mathbf{g}^T\mathbf{p}/d+\mathbf{g}^T\mathbf{x}/d} = \frac{a_0+\mathbf{a}^T\mathbf{x}}{c_0+\mathbf{c}^T\mathbf{x}}$$

$$h = \frac{-(\mathbf{f}\times\mathbf{d})^T\mathbf{p}+(\mathbf{f}\times\mathbf{d})^T\mathbf{x}}{-\mathbf{g}^T\mathbf{p}/d+\mathbf{g}^T\mathbf{x}/d} = \frac{b_0+\mathbf{b}^T\mathbf{x}}{c_0+\mathbf{c}^T\mathbf{x}}$$

$$\mathbf{a} = \mathbf{d}\times\mathbf{h} \qquad a_0 = -(\mathbf{d}\times\mathbf{h})^T\mathbf{p}$$

$$\mathbf{b} = \mathbf{f}\times\mathbf{d} \qquad b_0 = -(\mathbf{f}\times\mathbf{d})^T\mathbf{p}$$

$$\mathbf{g} = -(\mathbf{f}\times\mathbf{h})$$

$$\mathbf{c} = \mathbf{g}/d \qquad c_0 = -\mathbf{g}^T\mathbf{p}/d$$

These formulas are used in the examples.

4.2 General Projection / Parallel

For a parallel projection the center of projection moves to infinity. We can handle this case by replacing **p** by **v** and (d**d)** with $d \to \infty$.

$$\mathbf{p} = \mathbf{v} - d\mathbf{d}$$

$$f = \frac{d(\mathbf{d}\times\mathbf{h})^T(\mathbf{x}-\mathbf{v}+d\mathbf{d})}{\mathbf{g}^T(\mathbf{x}-\mathbf{v}+d\mathbf{d})}$$

$$(\mathbf{d}\times\mathbf{h})^T\mathbf{d} = 0$$
$$d \to \infty$$

$$f = \frac{d(\mathbf{d}\times\mathbf{h})^T(\mathbf{x}-\mathbf{v})}{\mathbf{g}^T d\mathbf{d}} = \frac{(\mathbf{d}\times\mathbf{h})^T(\mathbf{x}-\mathbf{v})}{\mathbf{g}^T\mathbf{d}}$$

$$f = -\frac{(\mathbf{d}\times\mathbf{h})^T\mathbf{v}}{\mathbf{g}^T\mathbf{d}} + \frac{(\mathbf{d}\times\mathbf{h})^T\mathbf{x}}{\mathbf{g}^T\mathbf{d}}$$

$$h = -\frac{(\mathbf{f}\times\mathbf{d})^T\mathbf{v}}{\mathbf{g}^T\mathbf{d}} + \frac{(\mathbf{f}\times\mathbf{d})^T\mathbf{x}}{\mathbf{g}^T\mathbf{d}}$$

For an orthogonal parallel projection the equations can be further simplified.

$$\mathbf{d}\times\mathbf{h} = \mathbf{r}$$
$$\mathbf{f}\times\mathbf{d} = \mathbf{t}$$
$$\mathbf{g}^T\mathbf{d} = 1$$

$$r = -\mathbf{r}^T\mathbf{v} + \mathbf{r}^T\mathbf{x}$$
$$t = -\mathbf{t}^T\mathbf{v} + \mathbf{t}^T\mathbf{x}$$

The dot products perform projections of the respective vectors onto the base vectors **r** and **t** .

4.3.1 General Projection / Camera Parameters

Sometimes one wants to determine the camera parameters for a given abstract perspective projection. It is assumed that the abstract projection was meant for a camera projection. Only for this case the camera matrix is a pure rotation matrix and therefore orthonormal.
The camera matrix rows are the base vectors of the camera system **u** in object space coordinates **x**.

$$\mathbf{u} = \mathbf{C}(\mathbf{x}-\mathbf{p}) = \begin{bmatrix}\mathbf{c}_1^T\\ \mathbf{c}_2^T\\ \mathbf{c}_3^T\end{bmatrix}(\mathbf{x}-\mathbf{p})$$

$$u = \mathbf{c}_1^T(\mathbf{x}-\mathbf{p})$$

$$v = \mathbf{c}_2^T(\mathbf{x}-\mathbf{p})$$

$$w = \mathbf{c}_3^T(\mathbf{x}-\mathbf{p})$$

$$r = \frac{\mathbf{c}_1^T(\mathbf{x}-\mathbf{p})}{\mathbf{c}_2^T(\mathbf{x}-\mathbf{p})} = \frac{a_0+\mathbf{a}^T\mathbf{x}}{1+\mathbf{c}^T\mathbf{x}}$$

$$t = \frac{\mathbf{c}_3^T(\mathbf{x}-\mathbf{p})}{\mathbf{c}_2^T(\mathbf{x}-\mathbf{p})} = \frac{b_0+\mathbf{b}^T\mathbf{x}}{1+\mathbf{c}^T\mathbf{x}}$$

Equation system for **p**

$$\mathbf{a}^T\mathbf{p} = -a_0$$

$$\mathbf{b}^T\mathbf{p} = -b_0$$

$$\mathbf{c}^T\mathbf{p} = -1$$

Rows of camera matrix, written as columns

$$\mathbf{c}_1 = -\mathbf{a}(\mathbf{c}_2^T\mathbf{p}) = \mathbf{a}/|\mathbf{a}|$$

$$\mathbf{c}_3 = -\mathbf{b}(\mathbf{c}_2^T\mathbf{p}) = \mathbf{b}/|\mathbf{b}|$$

$$\mathbf{c}_2 = -\mathbf{c}(\mathbf{c}_2^T\mathbf{p}) = \mathbf{c}_3\times\mathbf{c}_1$$

Camera constant

$$d = |\mathbf{a}|/|\mathbf{c}| = |\mathbf{b}|/|\mathbf{c}|$$

The signs for the calculation of the camera matrix are based on the assumption $\mathbf{c}_2^T\mathbf{p} < 0$.

The equations for the camera position **p** are valid for any projection, but for a parallel projection the position of the camera is at infinity. For hidden surface suppression we need a reference camera position somewhere on the view line in front of the object.
In the equations for **p** the third is obsolete. We can choose an arbitrary value e.g. $p_z = 10$ (not zero) and solve the remaining two equations for p_x and p_y .
A better method for the parallel projection is this, using e.g. $d = 10$:

$$\mathbf{d} = \mathbf{b}\times\mathbf{a}$$

$$\mathbf{d} = \mathbf{d}/|\mathbf{d}|$$

$$\mathbf{p} = \mathbf{v} - d\mathbf{d}$$

4.3.2 General Projection / Camera Parameters

The angles for the camera can be extracted from the rotation matrix $\mathbf{C}_{41} = (c_{ik})$ in chapter

$$\mathbf{C} = \begin{bmatrix} \mathbf{c}_1^T \\ \mathbf{c}_2^T \\ \mathbf{c}_3^T \end{bmatrix}$$

$$\tan(\alpha) = \frac{c_{23}}{\sqrt{c_{21}^2 + c_{22}^2}}$$

$$\tan(\beta) = \frac{-c_{13}}{c_{33}}$$

$$\tan(\gamma) = \frac{-c_{21}}{c_{22}}$$

This requires a four-quadrant arcustangent like $\gamma = \text{atan2}(-c_{21}, c_{22})$.

5.1 Homogeneous Coordinates

Homogeneous coordinates are introduced here as recipies how to handle 3D transformations by 4D matrices. The matrix **A** means rotation, scaling and shear. **t** is a translation. So far it is a general affine transformation. The intermediate values **u** are divided by the fourth coordinate q, which delivers the result **r**. Altogether and using k = 1/d as a placeholder we have a planar projective transformation. Homogeneous matrices can be multiplied,
In text books [6] the fourth coordinate is mostly called w instead of q, and **u** and **r** are not distinguished.

$$\begin{bmatrix} u \\ v \\ w \\ q \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & k & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{aligned} \mathbf{u} &= \mathbf{A}\mathbf{x} + \mathbf{t} \\ \mathbf{r} &= \mathbf{u}/q \\ r &= u/(ky) \\ s &= v/(ky) \\ t &= w/(ky) \end{aligned}$$

For the camera projection we need first a translation, then a rotation and finally the division. This is done by concatenated matrices. See chapter 2.1.

$$\begin{bmatrix} u \\ v \\ w \\ q \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1/d & 0 & 0 \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} & c_{13} & 0 \\ c_{21} & c_{22} & c_{23} & 0 \\ c_{31} & c_{32} & c_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{aligned} \mathbf{u} &= \mathbf{C}(\mathbf{x} - \mathbf{p}) \\ \mathbf{r} &= \mathbf{u}/q \\ r &= (u/v)d \\ s &= (v/v)d \\ t &= (w/v)d \end{aligned}$$

The abstract projection is handled straightforward by two matrices:

$$\begin{bmatrix} u \\ v \\ w \\ q \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z & a_0 \\ c_x & c_y & c_z & c_0 \\ b_x & b_y & b_z & b_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{aligned} \mathbf{u} &= \mathbf{A}\mathbf{x} + \mathbf{a}_0 \\ \mathbf{f} &= \mathbf{u}/q \\ q &= (c_0 + \mathbf{c}^T\mathbf{x}) \\ f &= (a_0 + \mathbf{a}^T\mathbf{x})/q \\ g &= 1 \\ h &= (b_0 + \mathbf{b}^T\mathbf{x})/q \end{aligned}$$

Once the respective matrices are filled and multiplied then a larger set of points **x** can be mapped onto the view plane by one formalistic homogeneous matrix multiplication which includes the division.
The PostScript examples were programmed without homogeneous coordinates.

5.2 Homogeneous Coordinates

On this occasion we can check the number of informations for each projection.

The camera projection uses either **p**, three angles and d, or **p**, **v** and the roll angle about the view line. These are seven informations. The base vectors **r**, **t** are unit vectors, orthogonal to each other and orthogonal to the view line.
Mapping four points (or three and a half) from 3D to 2D is sufficient for the identification of the parameters.

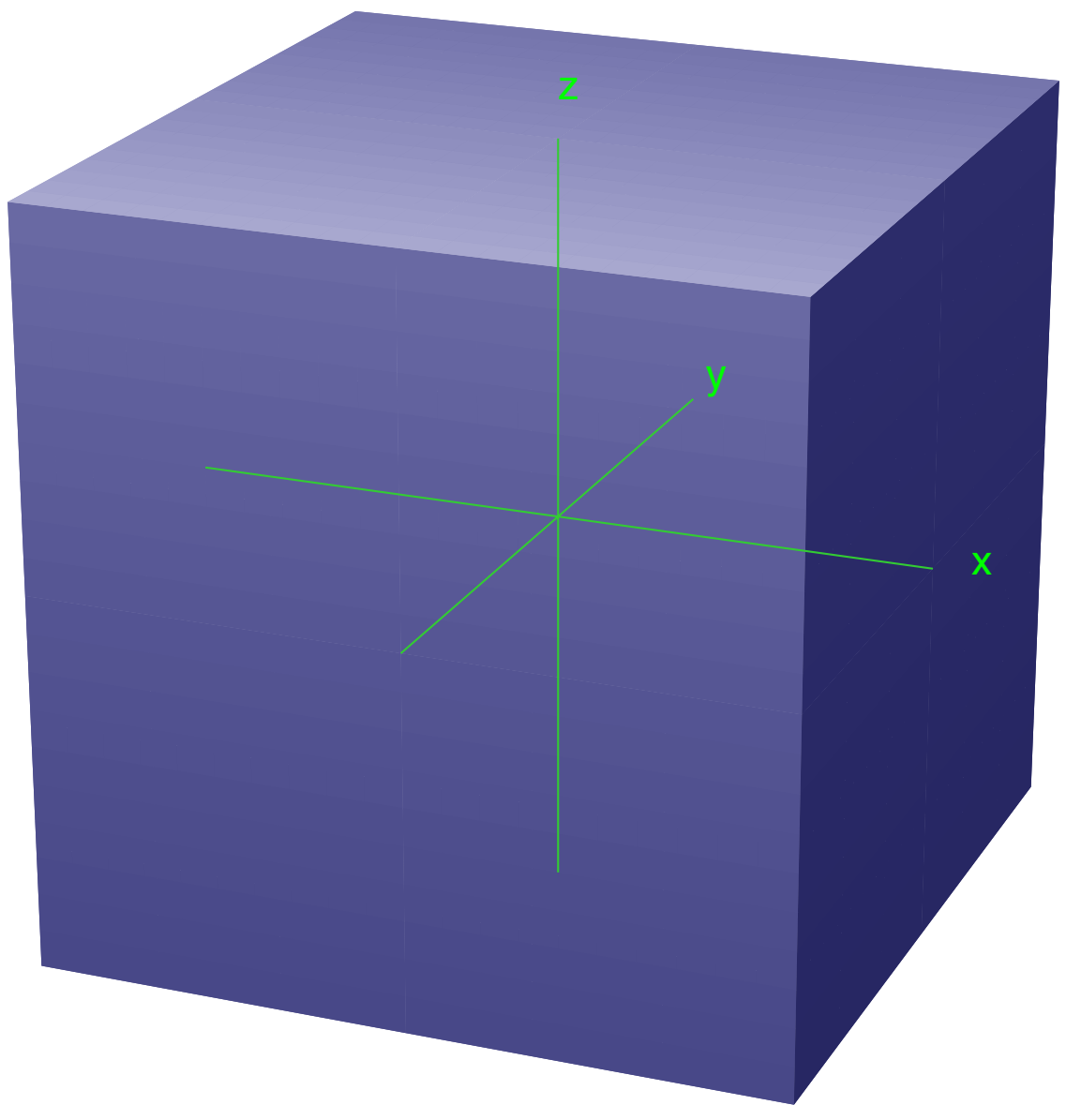
The abstract projection uses a_0, **a**, b_0, **b**, and **c**. c_0 doesn´t count because we can always divide by one non-zero parameter. These are eleven parameters.
Mapping six points (or five and a half) from 3D to 2D is sufficient for the identification of the parameters.

The general projection uses **p**, **v** and **f**, **h**. The base vectors are unit vectors but neither orthogonal to each other nor orthogonal to the view line. These are ten informations.
Mapping five points from 3D to 2D should be sufficient, but it isn´t, as we can easily see by drawing a unit cube and its vanishing points (example 9). Where is the missing eleventh information ?

6.1 Examples

The examples were programmed by PostScript.
In addition to the nomenclature in the previous chapters we use **L** for the light position.
This is a perspective camera projection.

1. Camera Projection Perspective



Px	4.000000
Py	-10.000000
Pz	4.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000
Lx	2.000000
Ly	-3.000000
Lz	4.000000

c11	0.928477
c12	0.371391
c13	0.000000
c21	-0.348155
c22	0.870388
c23	-0.348155
c31	-0.129302
c32	0.323254
c33	0.937437

6.2 Examples

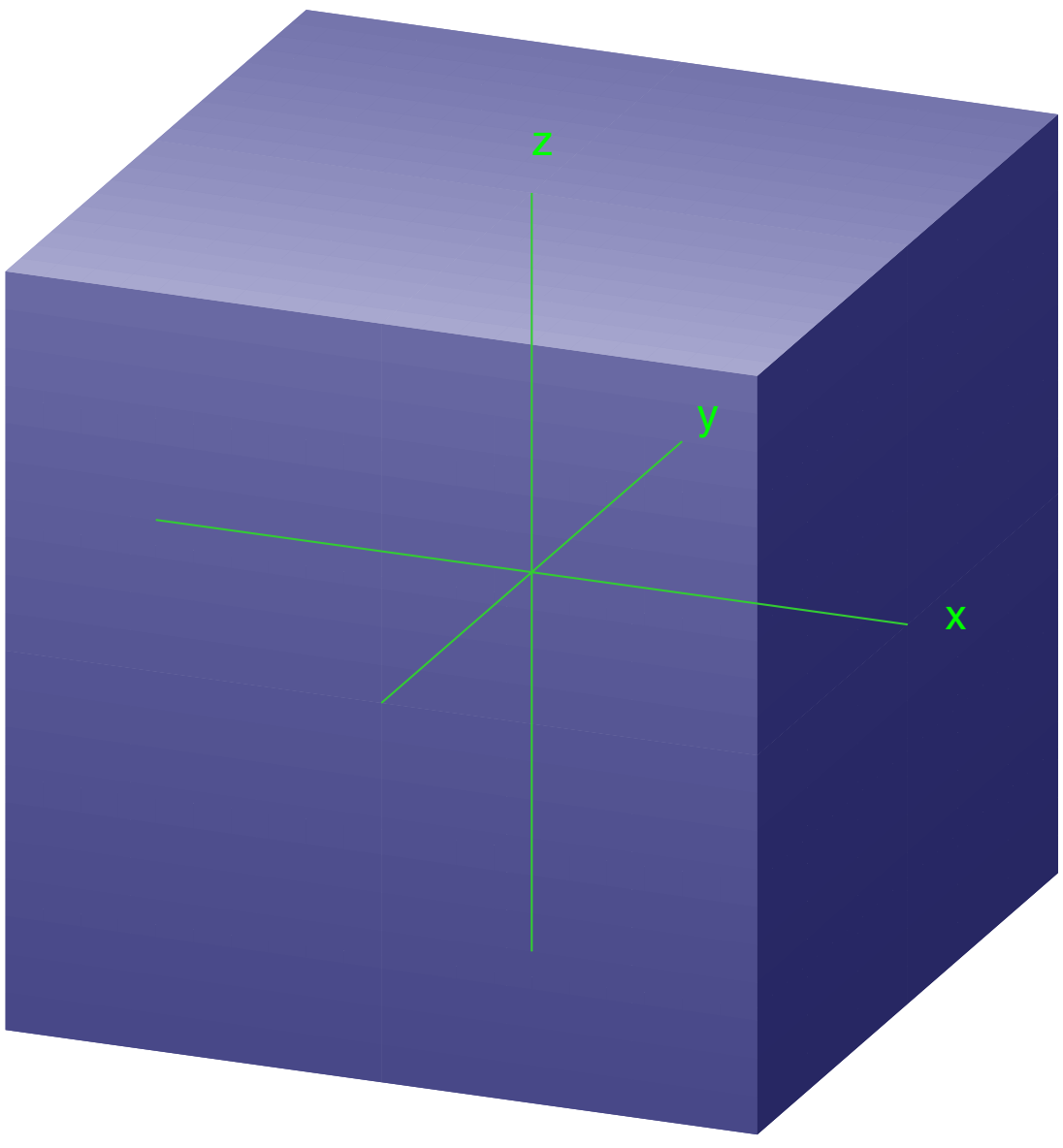
A parallel camera projection.

2. Camera Projection Parallel

Px 4.000000
Py -10.000000
Pz 4.000000

Vx 0.000000
Vy 0.000000
Vz 0.000000

Lx 2.000000
Ly -3.000000
Lz 4.000000



c11 0.928477
c12 0.371391
c13 0.000000

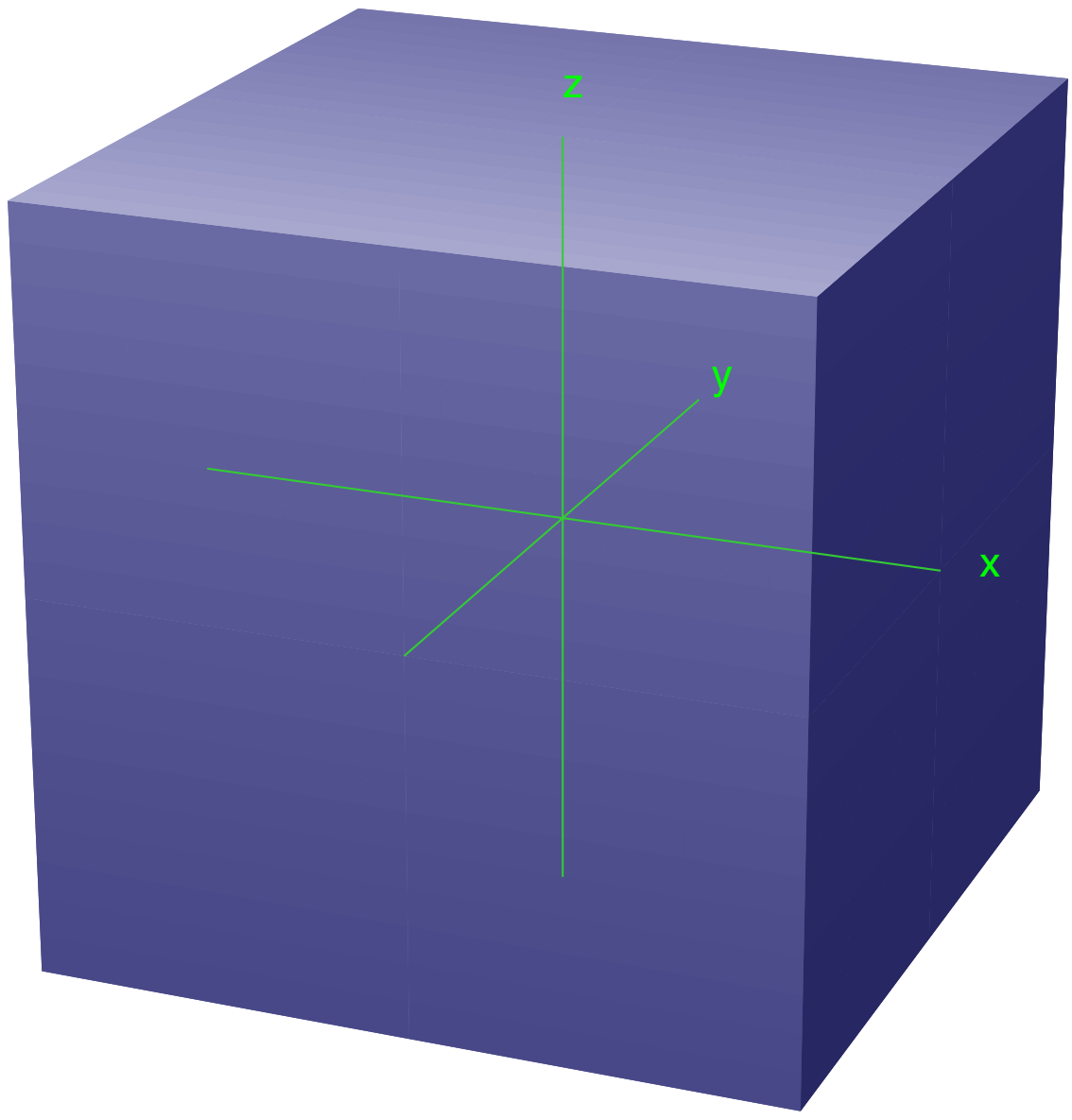
c21 -0.348155
c22 0.870388
c23 -0.348155

c31 -0.129302
c32 0.323254
c33 0.937437

6.3 Examples

A general projection, using the first and the third row of the camera matrix of the first example as base vectors **f** and **h**. The result is the same.

3. General Projection by Camera Matrix



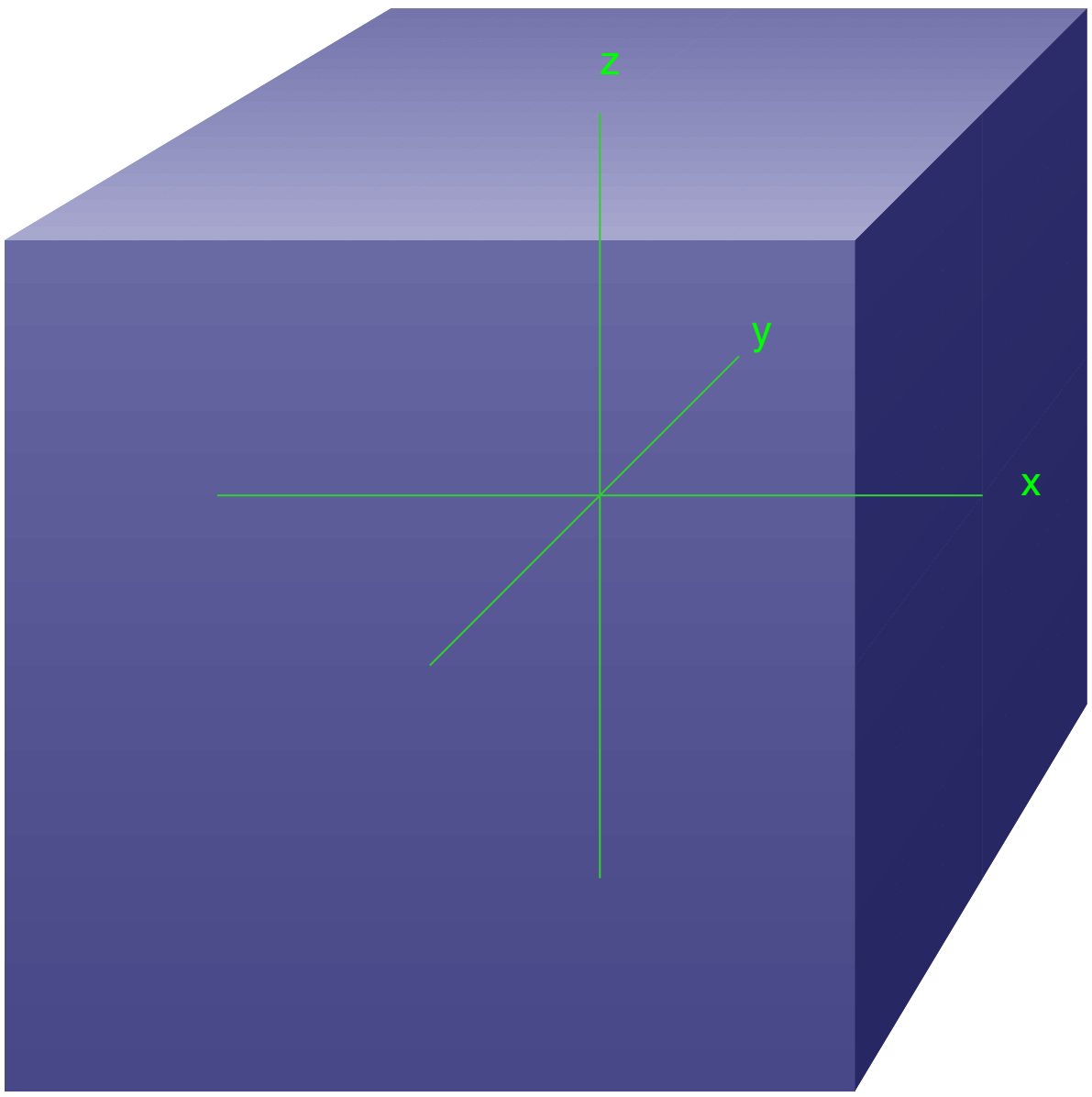
Px	4.000000
Py	-10.000000
Pz	4.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000
Lx	2.000000
Ly	-3.000000
Lz	4.000000
A0	0.000000
Ax	0.928477
Ay	0.371391
Az	-0.000000
B0	0.000000
Bx	-0.129302
By	0.323254
Bz	0.937437
C0	1.000000
Cx	-0.030303
Cy	0.075758
Cz	-0.030303
Dx	-0.348155
Dy	0.870388
Dz	-0.348155
Fx	0.928477
Fy	0.371391
Fz	0.000000
Gx	-0.348155
Gy	0.870388
Gz	-0.348155
Hx	-0.129302
Hy	0.323254
Hz	0.937437

c11	0.928477
c12	0.371391
c13	0.000000
c21	-0.348155
c22	0.870388
c23	-0.348155
c31	-0.129302
c32	0.323254
c33	0.937437

6.4 Examples

A general projection, using base vectors **f** and **h** for an undistorted front face.

4. General Projection by Base Vectors

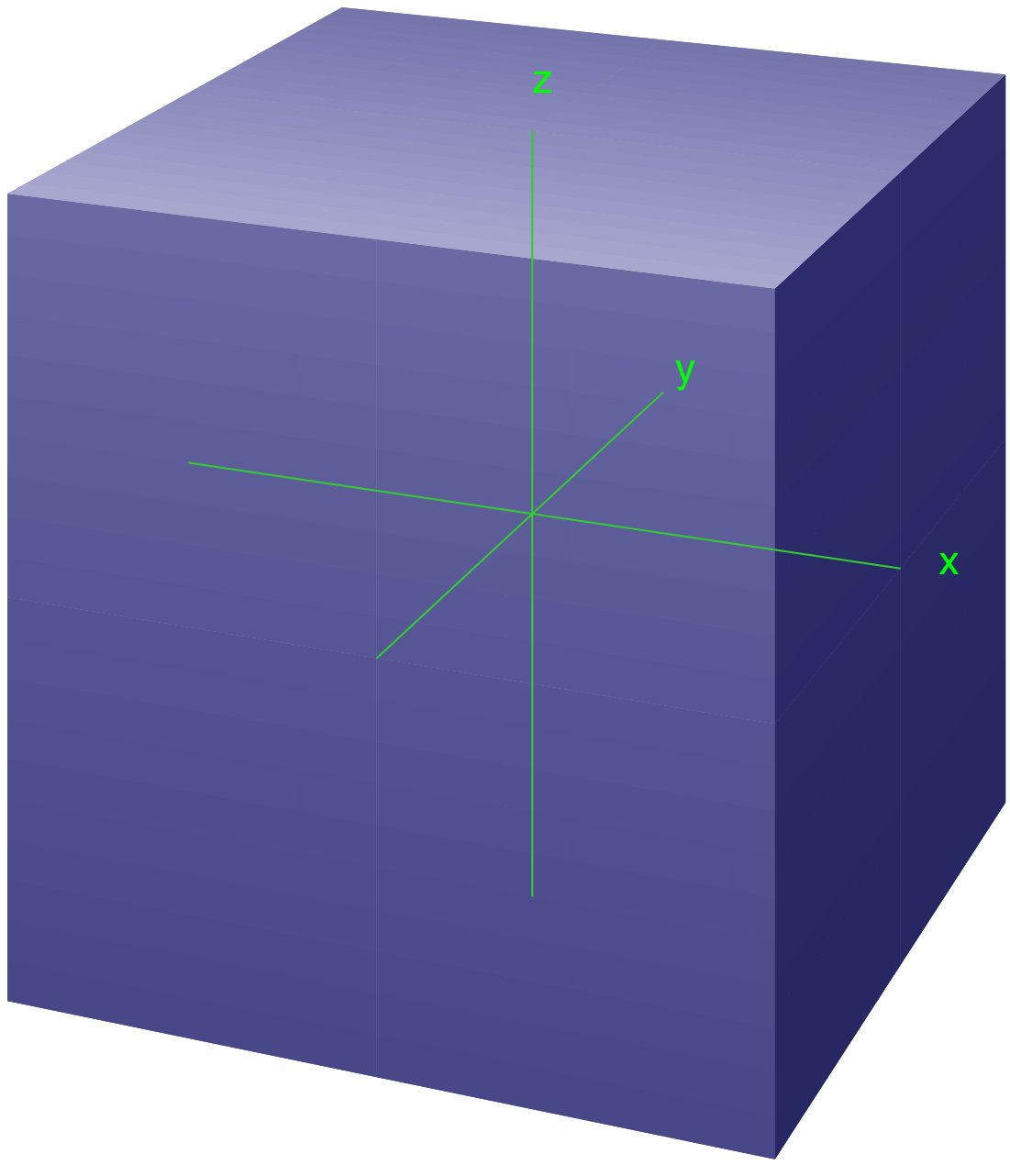


Px	4.000000
Py	-10.000000
Pz	4.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000
Lx	2.000000
Ly	-3.000000
Lz	4.000000
A0	0.000000
Ax	0.870388
Ay	0.348155
Az	0.000000
B0	0.000000
Bx	0.000000
By	0.348155
Bz	0.870388
C0	0.870388
Cx	0.000000
Cy	0.087039
Cz	0.000000
Dx	-0.348155
Dy	0.870388
Dz	-0.348155
Fx	1.000000
Fy	0.000000
Fz	0.000000
Gx	0.000000
Gy	1.000000
Gz	0.000000
Hx	0.000000
Hy	0.000000
Hz	1.000000

6.5 Examples

A general projection, using the first row of the camera matrix for **f** and a new base vector **h** for rectified verticals.

5. General Projection with Vertical Rectification



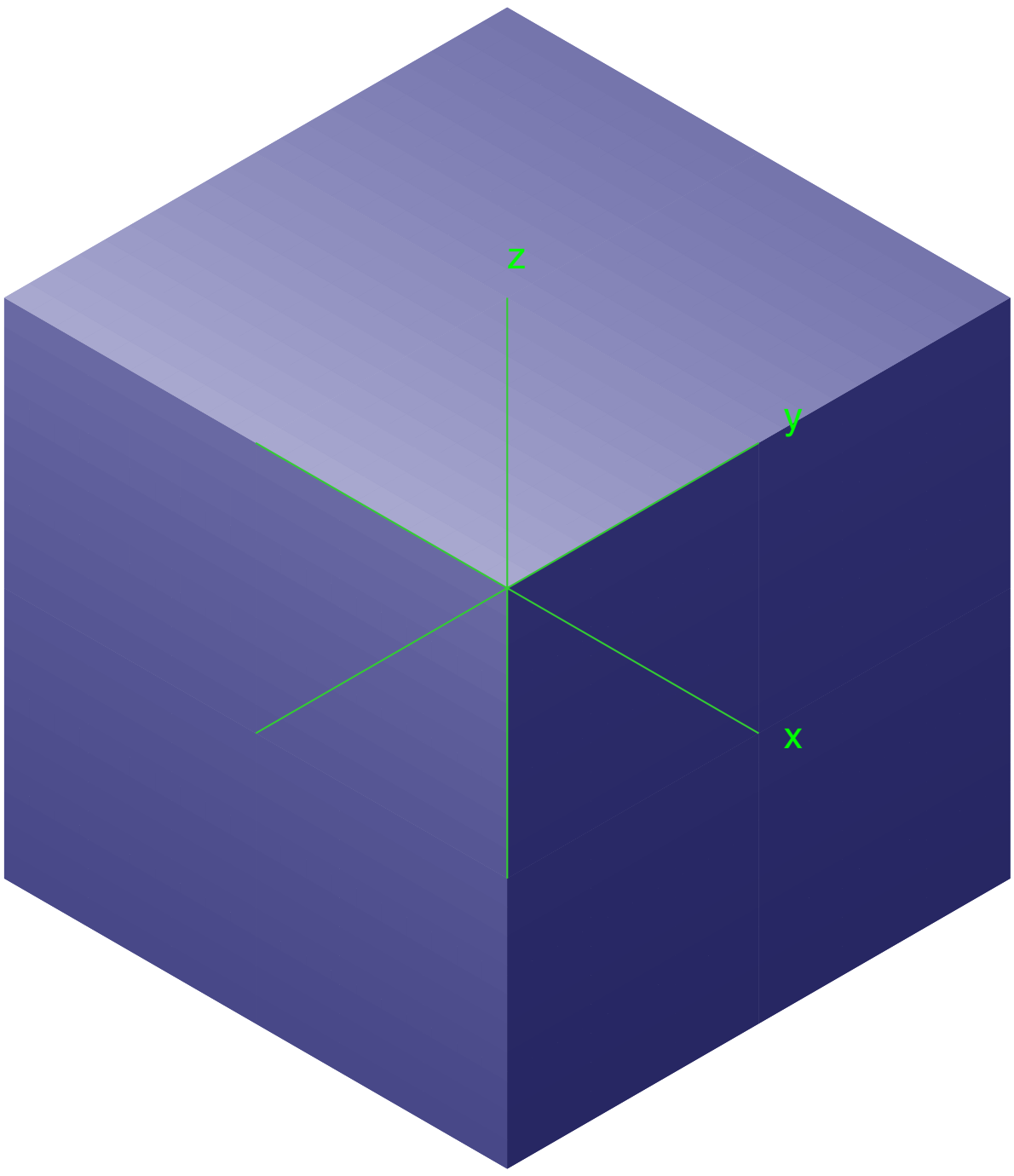
Px	4.000000
Py	-10.000000
Pz	4.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000
Lx	2.000000
Ly	-3.000000
Lz	4.000000
A0	0.000000
Ax	0.870388
Ay	0.348155
Az	0.000000
B0	0.000000
Bx	-0.129302
By	0.323254
Bz	0.937437
C0	0.937437
Cx	-0.032325
Cy	0.080814
Cz	0.000000
Dx	-0.348155
Dy	0.870388
Dz	-0.348155
Fx	0.928477
Fy	0.371391
Fz	0.000000
Gx	-0.371391
Gy	0.928477
Gz	0.000000
Hx	0.000000
Hy	0.000000
Hz	1.000000

c11	0.928477
c12	0.371391
c13	0.000000
c21	-0.348155
c22	0.870388
c23	-0.348155
c31	-0.129302
c32	0.323254
c33	0.937437

6.6 Examples

An isometric parallel projection, using a camera position with equal absolute values for p_x, p_y, p_z and the first and the third row of the camera matrix for base vectors **f** and **h**. The camera position defines merely the view line.

6. General Isometric Projection



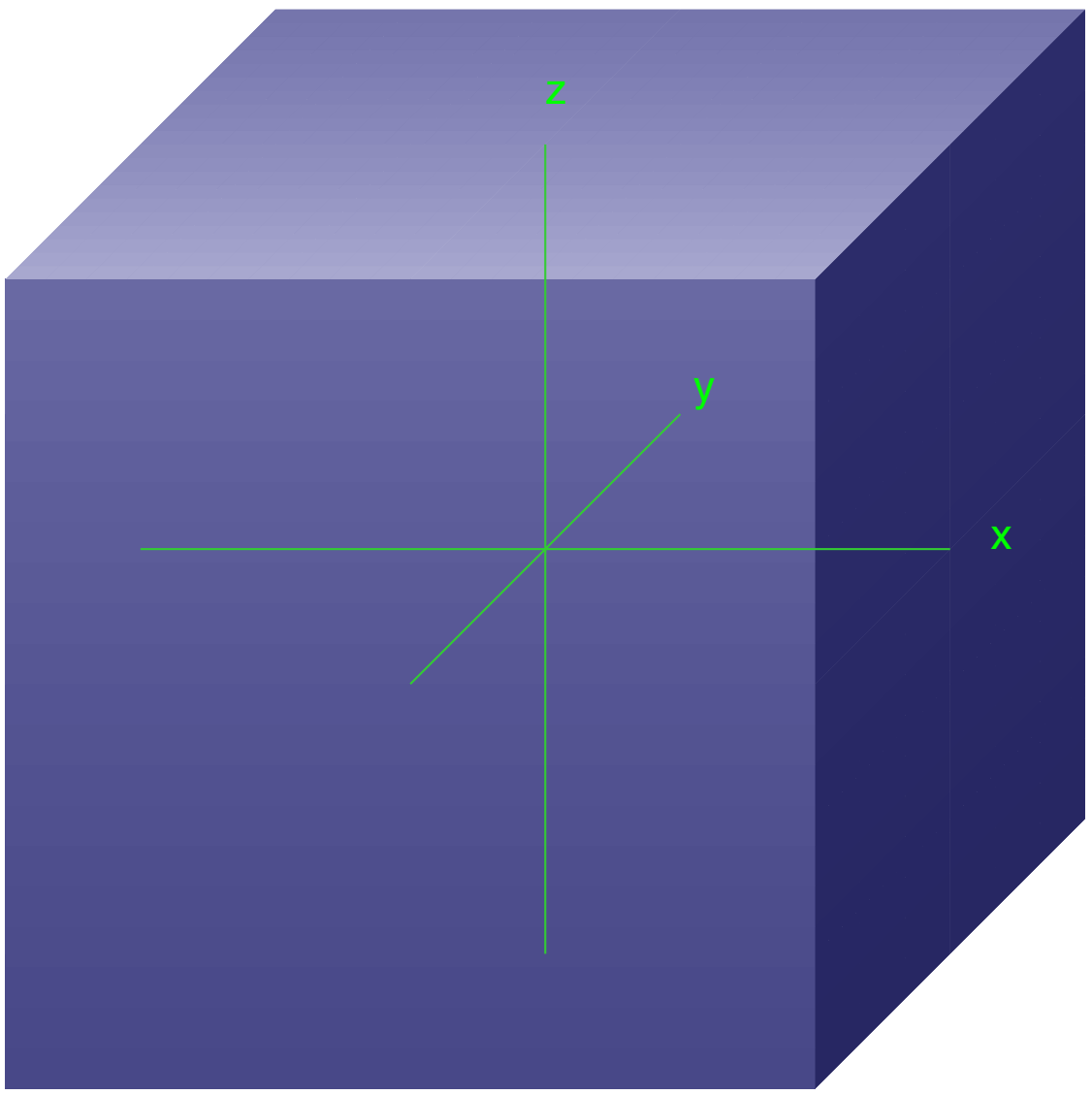
Px	1.000000
Py	-1.000000
Pz	1.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000
Lx	2.000000
Ly	-3.000000
Lz	4.000000
A0	0.000000
Ax	0.707107
Ay	0.707107
Az	0.000000
B0	0.000000
Bx	-0.408248
By	0.408248
Bz	0.816497
C0	1.000000
Cx	0.000000
Cy	0.000000
Cz	0.000000
Dx	-0.577350
Dy	0.577350
Dz	-0.577350
Fx	0.707107
Fy	0.707107
Fz	0.000000
Gx	-0.577350
Gy	0.577350
Gz	-0.577351
Hx	-0.408248
Hy	0.408248
Hz	0.816496

c11	0.707107
c12	0.707107
c13	0.000000
c21	-0.577350
c22	0.577350
c23	-0.577350
c31	-0.408248
c32	0.408248
c33	0.816496

6.7 Examples

A cavalier projection, using simple base vectors **f** and **h** and a specific camera position, which defines merely the direction of the view line.

7. General Cavalier Projection

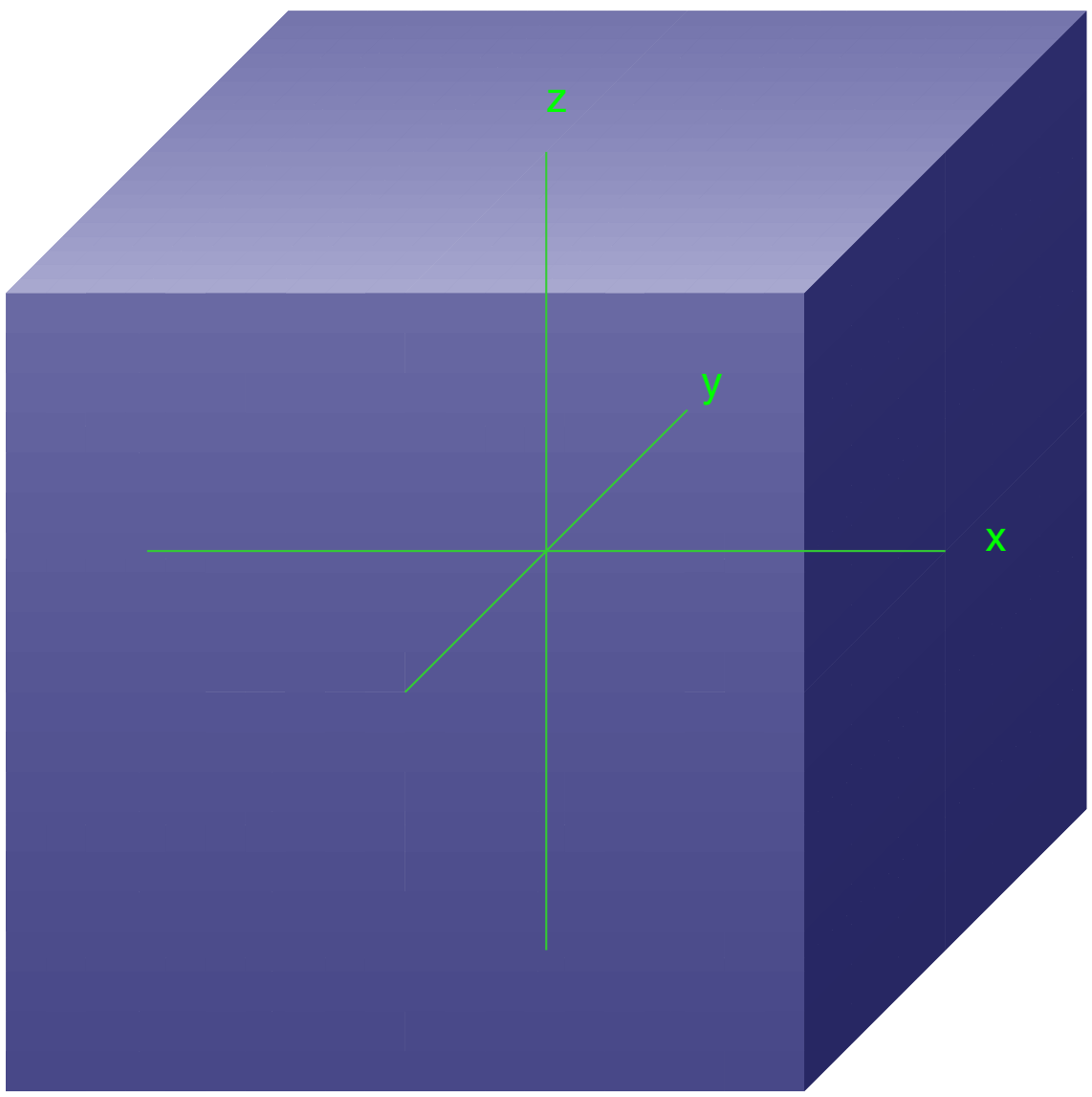


Px	1.000000
Py	-3.000000
Pz	1.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000
Lx	2.000000
Ly	-3.000000
Lz	4.000000
A0	0.000000
Ax	1.000000
Ay	0.333333
Az	0.000000
B0	-0.000000
Bx	0.000000
By	0.333333
Bz	1.000000
C0	1.000000
Cx	0.000000
Cy	0.000000
Cz	0.000000
Dx	-0.301511
Dy	0.904534
Dz	-0.301511
Fx	1.000000
Fy	0.000000
Fz	0.000000
Gx	0.000000
Gy	1.000000
Gz	0.000000
Hx	0.000000
Hy	0.000000
Hz	1.000000

6.8 Examples

A cavalier projection, using directly the abstract projection. The parameters a_0, **a**, b_0, **b**, c_0, **c** were calculated according to chapter 3. For a parallel projection we have c_0=1 and **c**=**0**.

8. Cavalier Projection by ABC



Px	3.162306
Py	-8.944253
Pz	3.162303
Vx	0.000000
Vy	0.000000
Vz	0.000000
Lx	2.000000
Ly	-3.000000
Lz	4.000000
A0	0.000001
Ax	0.999986
Ay	0.353552
Az	-0.000002
B0	0.000001
Bx	-0.000002
By	0.353552
Bz	0.999987
C0	1.000000
Cx	-0.000012
Cy	0.000001
Cz	-0.000012
Dx	-0.316231
Dy	0.894425
Dz	-0.316230

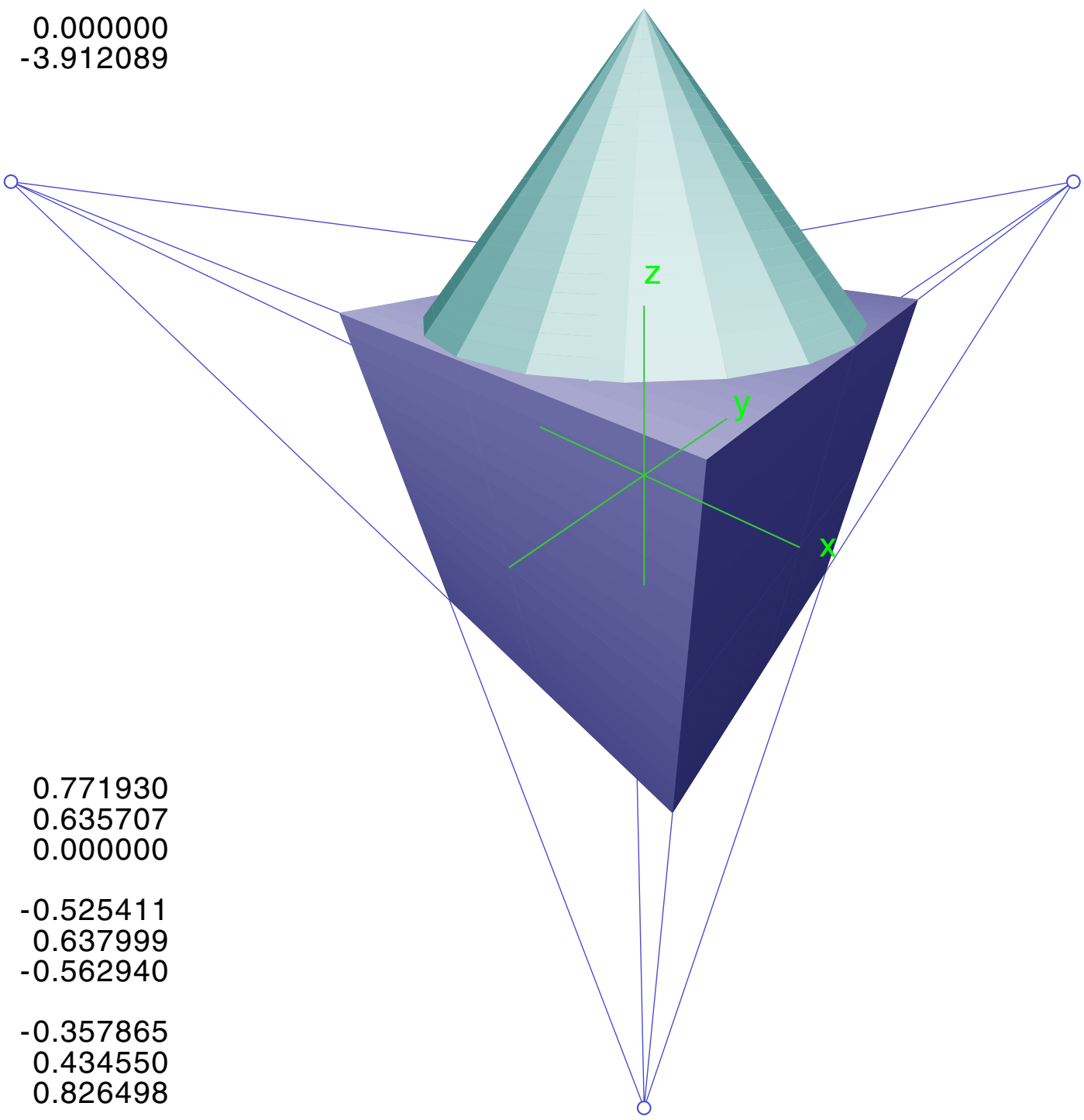
6.9 Examples

A camera projection with vanishing points. The vanishing points belong always to an object. They are easily found for the axis aligned cube:

x-direction $f_x = a_x/c_x$ $h_x = b_x/c_x$
y-direction $f_y = a_y/c_y$ $h_y = b_y/c_y$
z-direction $f_z = a_z/c_z$ $h_z = b_z/c_z$

9. General Projection by Camera Matrix / Vanishing Points

Fvx	-3.914790
Hvx	1.814887
Fvy	2.655013
Hvy	1.814887
Fvz	0.000000
Hvz	-3.912089



Px	1.400000
Py	-1.700000
Pz	1.500000
Vx	0.000000
Vy	0.000000
Vz	0.000000
Lx	2.000000
Ly	-3.000000
Lz	4.000000
A0	0.000000
Ax	0.771930
Ay	0.635707
Az	-0.000000
B0	0.000000
Bx	-0.357865
By	0.434550
Bz	0.826498
C0	1.000000
Cx	-0.197183
Cy	0.239437
Cz	-0.211268
Dx	-0.525411
Dy	0.637999
Dz	-0.562940

c11	0.771930
c12	0.635707
c13	0.000000
c21	-0.525411
c22	0.637999
c23	-0.562940
c31	-0.357865
c32	0.434550
c33	0.826498

Fx	0.771930
Fy	0.635707
Fz	0.000000
Gx	-0.525411
Gy	0.637999
Gz	-0.562940
Hx	-0.357865
Hy	0.434550
Hz	0.826498

7.1 PostScript Code

The complete PostScript program is shown here for the author´s bookkeeping. Some parts may help the reader for a better understanding of the document.

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox:  0 0 808 808
%%Creator:      Gernot Hoffmann
%%Title:        ProjectM1
%%CreationDate: March 27 2004

% Disable setpagedevice
 /setpagedevice {pop} bind def

% A convex object in parametric representation x(p1,p2),y(p1,p2),z(p1,p2)
% is defined for p1,p2=-1..+1 in the expected volume x,y,z=-1..+1
% General projections
% Visibility by backface culling
% Lambert/Phong light model
% Wireframe or rendering

/mm {2.834646 mul} def
/sx   60 mm def             % Length -sx..+sx
/sy   60 mm def
/bbx 280 mm def             % Bounding box
/bby 280 mm def

/x0  140 mm def             % Center
/y0  140 mm def

/np   20 def                % 2np divisions for p1,p2=-1..+1

/Rnd   2 def                % 1 Wireframe on black background, np=20
                            % 2 Rendered  on white background, np=50..100
/Phg   1 def                % 1   Lambert
                            % 2,3 Phong
/Env 0.3 def                % Environmental light
/H 240 360 div def    % HSB color order system
/S 1.0 def
/B 1.0 def

% For camera projection only
/Mod (Cam) def              % Cam Camera projection
                            % Gen General projection
/Map (Per) def              % Par Parallel projection
                            % Per Perspective projection
/Bet 0 def                  % Roll angle, default 0

/nn np neg def
/n2 np 2 mul 1 add def
/dp 1 np div def

/xa1 n2 array def
/xa2 n2 array def
/ya1 n2 array def
/ya2 n2 array def
/za1 n2 array def
/za2 n2 array def

/lw  0 def
/x   0 def /y   0 def /z   0 def
/xf  0 def /yf  0 def /zf  0 def
/xc  0 def /yc  0 def /zc  0 def
/dx  0 def /dy  0 def /dz  0 def /dr 0 def
/ca  0 def /cb  0 def /cg  0 def
/sa  0 def /sb  0 def /sg  0 def
/c11 0 def /c12 0 def /c13 0 def
/c21 0 def /c22 0 def /c23 0 def
/c31 0 def /c32 0 def /c33 0 def
/DD  0 def /sel 0 def
/p1  0 def /p2  0 def
/Alf 0 def /Gam 0 def
/cp1 0 def /sp1 0 def /cp2 0 def /sp2 0 def
/uc  0 def /vc  0 def /wc  0 def
/dlx 0 def /dly 0 def /dlz 0 def
/dnx 0 def /dny 0 def /dnz 0 def
```

7.2 PostScript Code

```
/x11 0 def /y11 0 def /z11 0 def /x12 0 def /y12 0 def /z12 0 def
/x21 0 def /y21 0 def /z21 0 def /x22 0 def /y22 0 def /z22 0 def
/ax  0 def /ay  0 def /az  0 def /bx  0 def /by  0 def /bz  0 def
/cal 0 def /cav 0 def /nrm 0 def /vfr 0 def /eps 0 def
/Nf  0 def /Nh  0 def /De  0 def
/fd  0 def /hd  0 def
/cax 0 def /cay 0 def /caz 0 def
/cbx 0 def /cby 0 def /cbz 0 def
/ccx 0 def /ccy 0 def /ccz 0 def
/ccd 0 def /DD  0 def /GD  0 def
/txt 0 def

%-Procedures

/MatCam % Camera rotation matrix
{/eps 1E-4 def
/dx Px Vx sub def
/dy Py Vy sub neg def
/dz Pz Vz sub neg def
/dr dx dup mul dy dup mul add sqrt def
  dx abs eps gt dy abs eps gt or
{/Gam dx dy atan def }
{/Gam 0 def          } ifelse
  dz abs eps gt dr abs eps gt or
{/Alf dz dr atan def }
{/Alf 0 def          } ifelse
/ca Alf cos def /sa Alf sin def
/cb Bet cos def /sb Bet sin def
/cg Gam cos def /sg Gam sin def
/c11 cb cg mul sa sb mul sg mul sub def
/c12 cb sg mul sa sb mul cg mul add def
/c13 ca sb mul neg def
/c21 ca sg mul neg def
/c22 ca cg mul def
/c23 sa def
/c31 sb cg mul sa cb mul sg mul add def
/c32 sb sg mul sa cb mul cg mul sub def
/c33 ca cb mul def
/DD  dx dup mul dy dup mul add dz dup mul add sqrt def
} bind def

/Func
{ sel 1 eq % Sphere
  {/cp1 p1 180 mul cos def   % Longitude
   /sp1 p1 180 mul sin def
   /cp2 p2 89.99 mul cos def % Latitude
   /sp2 p2 89.99 mul sin def
   /xf cp2 cp1 mul def
   /yf cp2 sp1 mul def
   /zf sp2 def
  } if
  sel 2 eq % Cone
{/cp1 p1 180  mul cos def  % Longitude
 /sp1 p1 180  mul sin def
 /xf cp1 1.001 p2 abs sub mul 0.7 mul def
 /yf sp1 1.001 p2 abs sub mul 0.7 mul def
 /zf p2 1.5 mul def
} if
sel 11 eq % Cube1 xz front
{/xf p1 def
 /yf -1 def
 /zf p2 def
} if
sel 12 eq % Cube2 xz back
{/xf p2 def
 /yf  1 def
 /zf p1 def
} if
sel 13 eq % Cube3 yz left
{/xf -1 def
 /yf p2 def
 /zf p1 def
} if
```

7.3 PostScript Code

```
  sel 14 eq % Cube4 yz right
  {/xf  1 def
   /yf p1 def
   /zf p2 def
  } if
  sel 15 eq % Cube5 xy bottom
  {/xf p2 def
   /yf p1 def
   /zf -1 def
  } if
  sel 16 eq % Cube6 xy top
  {/xf p1 def
   /yf p2 def
   /zf  1 def
  } if
 } bind def

/Transf
{% Mod=Cam Camera projection
 %   Map=Par Parallel
 %   Map=Per Perspective
 % Mod=Gen General projection
 % xf yf zf on the stack expected
   Mod (Cam) eq
  { /zc exch Pz sub def
    /yc exch Py sub def
    /xc exch Px sub def
    /uc c11 xc mul c12 yc mul add c13 zc mul add def
    /wc c31 xc mul c32 yc mul add c33 zc mul add def
    Map (Par) eq
   { /fd uc def /hd wc def } if
    Map (Per) eq
   {/vc c21 xc mul c22 yc mul add c23 zc mul add def
    /fd uc vc div DD mul def /hd wc vc div DD mul def } if
  } if
   Mod (Gen) eq
  {/zc exch def
   /yc exch def
   /xc exch def
   /Nf A0 Ax xc mul add Ay yc mul add Az zc mul add def
   /Nh B0 Bx xc mul add By yc mul add Bz zc mul add def
   /De C0 Cx xc mul add Cy yc mul add Cz zc mul add def
   /fd Nf De div def
   /hd Nh De div def
  } if
} bind def

/Light
% Lambert/Phong light model
% cal=cosine of angle between normal vector and light direction
{ cal 0 lt {/cal 0 def } if
  Phg 1 sub { /cal cal dup mul def } repeat
 /cal cal 1 Env sub mul Env add 0.95 mul def
  H S 1 cal sub mul cal sethsbcolor
} bind def

/RenderF
% Draw 3D view with backface culling, parameters p1,p2: -1..+1
{ mark
  0.15 mm sx div setlinewidth
 /p1 -1 def
 /p2 -1 def
 /i 0 def
  nn 1 np
 { pop
   Func
   xa1 i xf put
   ya1 i yf put
   za1 i zf put
  /i  i 1 add def
  /p1 p1 dp add def
 } for
 /p2 -1 dp add def
  nn 1 np 1 sub
   {/p1 -1 def
   /i 0 def
    nn 1 np
```

7.4 PostScript Code

```
  { pop
    Func
    xa2 i xf put
    ya2 i yf put
    za2 i zf put
   /p1 p1 dp add def
   /i i 1 add def
  } for
   mark
  /p1 -1 def
  /i 0 def
   nn 1 np 1 sub
{ pop
 /x11 xa1 i get def
 /y11 ya1 i get def
 /z11 za1 i get def
 /x12 xa1 i 1 add get def
 /y12 ya1 i 1 add get def
 /z12 za1 i 1 add get def
 /x21 xa2 i get def
 /y21 ya2 i get def
 /z21 za2 i get def
 /x22 xa2 i 1 add get def
 /y22 ya2 i 1 add get def
 /z22 za2 i 1 add get def
 %Normal vector
 /ax x12 x11 sub def
 /ay y12 y11 sub def
 /az z12 z11 sub def
 /bx x21 x11 sub def
 /by y21 y11 sub def
 /bz z21 z11 sub def
 /dnx ay bz mul by az mul sub def
 /dny az bx mul ax bz mul sub def
 /dnz ax by mul ay bx mul sub def
 /nrm 1 dnx dup mul dny dup mul add dnz dup mul add sqrt div def
 /dnx dnx nrm mul def
 /dny dny nrm mul def
 /dnz dnz nrm mul def
 %View vector
 /dvx Px x11 sub def
 /dvy Py y11 sub def
 /dvz Pz z11 sub def
 /nrm 1 dvx dup mul dvy dup mul add dvz dup mul add sqrt div def
 /dvx dvx nrm mul def
 /dvy dvy nrm mul def
 /dvz dvz nrm mul def
 %Cosine
 /cav dvx dnx mul dvy dny mul add dvz dnz mul add def
  cav 0 ge Rnd 1 eq or
{ newpath
  x11 y11 z11 Transf fd hd moveto
   x12 y12 z12 Transf fd hd lineto
   x22 y22 z22 Transf fd hd lineto
   x21 y21 z21 Transf fd hd lineto
   closepath
  /dlx Lx xf sub def
  /dly Ly yf sub def
  /dlz Lz zf sub def
  /cal dnx dlx mul dny dly mul add dnz dlz mul add def
  /nrm dlx dup mul dly dup mul add dlz dup mul add sqrt def
  /cal cal nrm div def
   Light
   Rnd 1 eq {stroke} if
   Rnd 2 eq {fill  } if
  } if
  /i i 1 add def
  /p1 p1 dp add def
  } for
  cleartomark
 /p2 p2 dp add def
  xa2 xa1 copy
  ya2 ya1 copy
  za2 za1 copy
 } for
 cleartomark
} bind def
```

7.5 PostScript Code

```
/Box % absolute
{ 0 setgray
 /lw 0.2 mm def lw setlinewidth
  newpath
  lw lw moveto
  bbx lw sub 0 rlineto
  0 bby lw sub rlineto
  bbx neg lw add 0 rlineto
  closepath
  Rnd 1 eq {fill   } if
  Rnd 2 eq {stroke } if
} bind def

/Axes % relative
{  Rnd 1 eq {0.5 setgray} if
   Rnd 2 eq {0.0 setgray} if
   0.3 mm sx div setlinewidth
   0.2 0.8 0.2 setrgbcolor
   newpath
   -1  0  0 Transf fd hd moveto  1 0 0  Transf fd hd lineto
    0 -1  0 Transf fd hd moveto  0 1 0  Transf fd hd lineto
    0  0 -1 Transf fd hd moveto  0 0 1  Transf fd hd lineto
   stroke
   0 1 0 setrgbcolor
  /fh 18 sx div def
  /Helvetica findfont fh scalefont setfont
   1.1 0   0   Transf fd hd moveto (x) show
   0   1.1 0   Transf fd hd moveto (y) show
   0   0   1.1 Transf fd hd moveto (z) show
 } bind def

/Cros
% C=AxB  Matrix [ax ay az] x Matrix [bx by bz]—>Matrix [cx cy cz]
{/cbz exch def /cby exch def /cbx exch def
 /caz exch def /cay exch def /cax exch def
 /ccx cay cbz mul caz cby mul sub def
 /ccy caz cbx mul cax cbz mul sub def
 /ccz cax cby mul cay cbx mul sub def
  ccz ccy ccx
} bind def

/Dotp
% c=A*B  Matrix [ax ay az]*Matrix [bx by bz]—>c
{/cbz exch def /cby exch def /cbx exch def
 /caz exch def /cay exch def /cax exch def
 /ccd cax cbx mul cay cby mul add caz cbz mul add def
  ccd
} bind def

/Norm
% A=A/Length(A)
% Matrix [ax ay az]—>Matrix [ax ay az] and Length on top of stack
{/caz exch def /cay exch def /cax exch def
 /ccd cax dup mul cay dup mul add caz dup mul add sqrt def
  caz ccd div cay ccd div cax ccd div ccd
} bind def

/Leng
% c=Leng(A) Matrix [ax ay az]—>c
{/caz exch def /cay exch def /cax exch def
 /ccd cax dup mul cay dup mul add caz dup mul add sqrt def
  ccd
} bind def

% Linear equation solver

/N  11 def
/M  12 def
/n1 N 1 sub def
/m1 M 1 sub def
/AMN M N mul array def
/ANM N M mul array def
/FM  M array def
/ANN N N mul array def
/XN  N array def
/YN  N array def
```

7.6 PostScript Code

```
/i 0 def /j 0 def /k 0 def
/dA 0 def /max 0 def /s 0 def /h 0 def /q 0 def /aik 0 def /bik 0 def /bkk 0 def
/pa 0 def /ca 0 def

/m
% Memory map; i,k on stack; top=N*i+k
{ exch
  N mul add
} bind def

/HoGaussP
{% Solve  ANN*XN=YN for XN and det(ANN)
 % Overwrite all arrays
/pa N array def
/ca N array def
/n1 N 1 sub def
/dA 1 def
0 1 n1 1 sub
{/k exch def
 /max 0 def
  pa k 0 put
 k 1 n1
 {/i exch def
  /s 0 def
   k 1 n1
   {/j exch def
    /s s ANN i j m get abs add def
   } for %j
    /bik ANN i k m get abs def
    /q bik s div def
     q max gt {/max q def pa k i put } if
} for %i
   pa k get k ne
   {/dA dA neg def
     0 1 n1
     {/j exch def
      /h ANN k j m get def
       ANN k j m ANN pa k get j m get put
       ANN pa k get j m h put
     } for %j
   } if
   /dA dA ANN k k m get mul def
   k 1 add 1 n1
   {/i exch def
    /aik ANN i k m get def
    /akk ANN k k m get def
    /bkk akk abs def
     bkk 1 lt
     {/bik aik abs def
     } if
     ANN i k m aik akk div put
      k 1 add 1 n1
      {/j exch def
        ANN i j m ANN i j m get ANN i k m get ANN k j m get mul sub put
      } for %j
     } for %i
} for %k
/dA dA ANN n1 n1 m get mul
0 1 n1 1 sub
{/k exch def
  pa k get k ne
 {/h YN k get def
   YN k YN pa k get get put
   YN pa k get h put
 } if
} for %k
0 1 n1
{/i exch def
  ca i YN i get put
  0 1 i 1 sub
  {/j exch def
    ca i ca i get ANN i j m get ca j get mul sub put
  } for %j
} for %i
n1 -1 0
{/i exch def
```

7.7 PostScript Code

```
 /s ca i get def
  i 1 add 1 n1
  {/k exch def
   /s s ANN i k m get YN k get mul sub def
  } for %k
 /bik ANN i i m get abs def
  YN i s ANN i i m get div put
} for %i
0 1 n1
{/i exch def
  XN i YN i get put
} for %i
} def % HoGaussP

/MatFill
{ % Fill AMN and FM
0 1 N M mul 1 sub
{/i exch def
  AMN i 0 put
} for
/k 0 def
0 1 5
{/i exch def
 /j i 3 mul def
 /i i 2 mul def
 /Fi FF i get def
 /Xi XX j get def /Yi XX j 1 add get def /Zi XX j 2 add get def
  AMN k 1 put AMN k 1 add Xi put AMN k 2 add Yi put AMN k 3 add Zi put
  AMN k 8  add Fi Xi mul neg put
  AMN k 9  add Fi Yi mul neg put
  AMN k 10 add Fi Zi mul neg put
 /k k 11 add def
} for
/k 70 def
0 1 5
{/i exch def
 /j i 3 mul def
 /i i 2 mul def
 /Hi FF i 1 add get def
 /Xi XX j get def /Yi XX j 1 add get def /Zi XX j 2 add get def
  AMN k 1 put AMN k 1 add Xi put AMN k 2 add Yi put AMN k 3 add Zi put
  AMN k 4 add Hi Xi mul neg put
  AMN k 5 add Hi Yi mul neg put
  AMN k 6 add Hi Zi mul neg put
 /k k 11 add def
} for
/k 0 def
0 1 5
{/i exch 2 mul def
  FM k       FF i get put
  FM k 6 add FF i 1 add get put
 /k k 1 add def
} for
} bind def

/MTrans
{%ANM = AMN¦
 0 1 n1
 {/i exch def
   0 1 m1
   {/k exch def
     ANM i M mul k add AMN k N mul i add get put
   } for
 } for
} bind def
```

7.8 PostScript Code

```
/MatMul
{% ANN = ANM*AMN; YN=ANM*FM
0 1 n1
{/i exch def
  0 1 n1
  {/k exch def
   /sum 0 def
    0 1 m1
     {/j exch def
      /sum sum ANM i M mul j add get AMN j N mul k add get mul add def
     } for
     ANN i N mul k add sum put
     } for
} for
0 1 n1
{/i exch def
 /sum 0 def
  0 1 m1
  {/j exch def
   /sum sum ANM i M mul j add get FM j get mul add def
  } for
  YN i sum put
} for
} bind def

/General
{ Map (Par) eq % Parallel
 {Vx Px sub Vy Py sub Vz Pz sub Norm /DD exch def
 /Dx exch def /Dy exch def /Dz exch def
  Fx Fy Fz Norm pop
 /Fx exch def /Fy exch def /Fz exch def
  Hx Hy Hz Norm pop
 /Hx exch def /Hy exch def /Hz exch def  Fx Fy Fz Hx Hy Hz Cros
 /Gx exch neg def /Gy exch neg def /Gz exch neg def
 /GD Gx Gy Gz Dx Dy Dz Dotp def
  Dx Dy Dz Hx Hy Hz Cros
 /Ax exch def /Ay exch def /Az exch def
 /A0 Ax Ay Az Vx Vy Vz Dotp GD div neg def
 /Ax Ax GD div def
 /Ay Ay GD div def
 /Az Az GD div def
  Fx Fy Fz Dx Dy Dz Cros
 /Bx exch def /By exch def /Bz exch def
 /B0 Bx By Bz Px Py Pz Dotp GD div def
 /Bx Bx GD div def
 /By By GD div def
 /Bz Bz GD div def
 /Cx 0 def
 /Cy 0 def
 /Cz 0 def
 } if
  Map (Per) eq % Perspective
 {Vx Px sub Vy Py sub Vz Pz sub Norm /DD exch def
 /Dx exch def /Dy exch def /Dz exch def
  Fx Fy Fz Norm pop
 /Fx exch def /Fy exch def /Fz exch def
  Hx Hy Hz Norm pop
 /Hx exch def /Hy exch def /Hz exch def  Fx Fy Fz Hx Hy Hz Cros
 /Gx exch neg def /Gy exch neg def /Gz exch neg def
 /GP Gx Gy Gz Px Py Pz Dotp def
  Dx Dy Dz Hx Hy Hz Cros
 /Ax exch def /Ay exch def /Az exch def
 /A0 Ax Ay Az Px Py Pz Dotp neg def
  Fx Fy Fz Dx Dy Dz Cros
 /Bx exch def /By exch def /Bz exch def
 /B0 Bx By Bz Px Py Pz Dotp neg def
 /Cx Gx DD div def
 /Cy Gy DD div def
 /Cz Gz DD div def
 /C0 GP DD div neg def
 } if
} bind def
```

7.9 PostScript Code

```
/UseCamPer % Use camera projection Perspective
{/Mod (Cam) def
 /Map (Per) def  % or Par
  MatCam
 /txt (1. Camera Projection Perspective) def
} bind def

/UseCamPar % Use camera projection Parallel
{/Mod (Cam) def
 /Map (Par) def  % or Par
  MatCam
 /txt (2. Camera Projection Parallel) def
} bind def

/UseGenPer % Use general projection by camera matrix
{/Mod (Gen) def
 /Map (Per) def
  MatCam
 /Fx c11 def /Fy c12 def /Fz c13 def
 /Hx c31 def /Hy c32 def /Hz c33 def
  General
 /txt (3. General Projection by Camera Matrix) def
} bind def

/UseBasVec % Use basis vectors
{/Mod (Gen) def
 /Map (Per) def
  General
 /txt (4. General Projection by Base Vectors) def
} bind def

/UseVerRec % Vertical rectification
{/Mod (Gen) def
 /Map (Per) def
  MatCam
 /Fx c11 def /Fy c12 def /Fz c13 def
 /Hx   0 def /Hy   0 def /Hz   1 def
  General
 /txt (5. General Projection with Vertical Rectification) def
} bind def

/UseIsoPar % Isometric
{/Mod (Gen) def
 /Map (Par) def
 /Px 1 def /Py -1 def /Pz 1 def
 /Vx 0 def /Vy  0 def /Vz 0 def
  MatCam
 /Fx c11 def /Fy c12 def /Fz c13 def
 /Hx c31 def /Hy c32 def /Hz c33 def
  General
 /txt (6. General Isometric Projection) def
} bind def

/UseCavPar % Cavalier
{/Mod (Gen) def
 /Map (Par) def
 /Px 1 def /Py -3 def /Pz 1 def
 /Vx 0 def /Vy  0 def /Vz 0 def
  MatCam
 /Fx  1 def /Fy 0 def /Fz 0 def
 /Hx  0 def /Hy 0 def /Hz 1 def
  General
 /txt (7. General Cavalier Projection) def
} bind def

/UseABCPar % Cavalier by ABC
{/Mod (Gen) def
 /Map (Par) def
 /Vx 0 def /Vy  0 def /Vz 0 def
 /txt (8. Cavalier Projection by ABC ) def
 /len 0.5 def              % length of y-edge in image
 /alf 45  def              % angle  of y-edge in image
 /ca alf cos len mul def
 /sa alf sin len mul def
 %Cube
```

7.10 PostScript Code

```
 /XX [ 0 0 0
       1 0 0
       0 1 0
       0 0 1
       1 1 0
       0 1 1 ] def

 % Image of cube
 /FF [ 0          0
       1          0
       ca        sa
       0          1
       ca 1 add  sa
       ca        sa 1 add ] def

 MatFill
 MTrans
 MatMul
 HoGaussP
/A0 XN 0 get def /Ax XN 1 get def /Ay XN 2 get def /Az XN  3 get def
/B0 XN 4 get def /Bx XN 5 get def /By XN 6 get def /Bz XN  7 get def
/C0  1       def /Cx XN 8 get def /Cy XN 9 get def /Cz XN 10 get def
/DD 10 def % arbitrary distance
 Bx By Bz Ax Ay Az Cros
/Dx exch def /Dy exch def /Dz exch def
 Dx Dy Dz Norm pop
/Dx exch def /Dy exch def /Dz exch def
/Px Vx Dx DD mul sub def
/Py Vy Dy DD mul sub def
/Pz Vz Dz DD mul sub def
} bind def


%-Begin

false setstrokeadjust

gsave

Box
x0 y0 translate
sx sy scale
% Default
/Ax  1 def /Ay  0 def /Az  0 def /A0 0 def
/Bx  0 def /By  1 def /Bz  0 def /B0 0 def
/Cx  0 def /Cy  0 def /Cz  0 def /C0 1 def
/Dx  0 def /Dy  1 def /Dz  0 def
/Fx  1 def /Fy  0 def /Fz  0 def
/Gx  0 def /Gy  1 def /Gz  0 def
/Hx  0 def /Hy  0 def /Hz  1 def
/Px  5 def /Py -5 def /Pz  5 def
/Vx  0 def /Vy  0 def /Vz  0 def
/Lx  5 def /Ly -7 def /Lz  4 def

% Settings
/Fx   1 def  % Base Vector
/Fy   0 def
/Fz   0 def
/Hx   0 def  % Base Vector
/Hy   0 def
/Hz   1 def
/Px   4 def  % Camera Position
/Py -10 def
/Pz   4 def
/Vx   0 def  % View Point
/Vy   0 def
/Vz   0 def
/Lx   2 def  % Light Position
/Ly  -3 def
/Lz   4 def
```

7.11 PostScript Code

```
% Choose one
/mod 1 def
 mod 1 eq {UseCamPer} if  % Use camera projection Perspective
 mod 2 eq {UseCamPar} if  % Use camera projection Parallel
 mod 3 eq {UseGenPer} if  % Use general projection by camera matrix
 mod 4 eq {UseBasVec} if  % Use basis vectors
 mod 5 eq {UseVerRec} if  % Use vertical rectification
 mod 6 eq {UseIsoPar} if  % Use isometry
 mod 7 eq {UseCavPar} if  % Use cavalier
 mod 8 eq {UseABCPar} if  % Use cavalier by ABC

/np 3 def
/nn np neg def
/n2 np 2 mul 1 add def
/dp 1 np div def

/sel 11 def RenderF
/sel 12 def RenderF
/sel 13 def RenderF
/sel 14 def RenderF
/sel 15 def RenderF
/sel 16 def RenderF

/np 10 def
/nn np neg def
/n2 np 2 mul 1 add def
/dp 1 np div def

/sel 11 def RenderF
/sel 12 def RenderF
/sel 13 def RenderF
/sel 14 def RenderF
/sel 15 def RenderF
/sel 16 def RenderF

Axes

grestore

% cont. next page
```

7.12 PostScript Code

```
/Shownum
% Draw number by string
%  Global
%  xtxt   Actual position not overwritten
%  ytxt
%  nu     Input number    not overwritten  nu =+-999999
%  fh     Font height     not overwritten
%  tms    Mantissa number of characters    tms=0...6
%  tms=3  Example
%  input  -23.56789  -999.99    0.4567  9999.123456
%  result -23.568    -999.990   0.467   9999.123
%  Postscript number to string is not well defined
%  e.g. 1E-5 instead of 0.00001
%  We use a straightforward BCD conversion.
%  This is always affected by round-off errors
%  because of 32-bit arithmetic
%  The procedure itself does not round
%  Results are different, depending on the interpreter

{ /tx0 xtxt def
  /ty0 ytxt def
  /tfw fh 0.6 mul def                    % character distance
  /tna nu abs 0.5E-6 add def             % abs value

  /tdec 1E5 def
  /tchr 1 string def

   tna 999999.1 lt                       % larger number replaced by #
  /tmm true def                          % sign
  {/tx0 tx0 tfw 6 mul sub def
   /tz 0 def
  1 1 5                                  % first 5 digits, no leading 0
  { pop
   /tk 0 def
   { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
   } loop
    tk 0 ne {/tz tz 1 add def} if
    tz 0 ne
    { tx0 ty0 moveto tk tchr cvs show

    } if
    tz 1 eq nu 0 lt and                  % minus
    { tx0 tfw 0.7 mul sub ty0 moveto (-) show
      /tmm false def
    } if
   /tdec tdec 0.1 mul def
   /tx0 tx0 tfw add def
  } for

   /tk 0 def                             % leading 0
   { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
   } loop
     tmm nu 0 lt and                     % minus
     { tx0 tfw 0.7 mul sub ty0 moveto (-) show
     } if
     tx0 ty0 moveto tk tchr cvs show
    /tdec tdec 0.1 mul def
    /tx0 tx0 tfw add def

   tms 0 gt                              % for float
   { tx0 ty0 moveto (.) show
    /tx0 tx0 tfw 0.5 mul add def
   1 1 tms
   { pop
    /tk 0 def
    { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
    } loop
      tx0 ty0 moveto tk tchr cvs show
    /tdec tdec 0.1 mul def
    /tx0 tx0 tfw add def
   } for
   } if
 }{ tx0 tfw sub ty0 moveto (#) show} ifelse
} bind def
```

7.13 PostScript Code

```
/List
{/fh 16 def
 /Helvetica findfont fh scalefont setfont
  /tms 6 def
  /xtxt 250 mm def
  /ytxt 255 mm def
  /x xtxt 20 mm sub def
   x ytxt moveto (Px) show /nu Px def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Py) show /nu Py def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Pz) show /nu Pz def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   x ytxt moveto (Vx) show /nu Vx def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Vy) show /nu Vy def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Vz) show /nu Vz def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   x ytxt moveto (Lx) show /nu Lx def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Ly) show /nu Ly def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Lz) show /nu Lz def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   mod 3 eq mod 4 eq or mod 5 eq or mod 6 eq or mod 7 eq or mod 8 eq
   {
   x ytxt moveto (A0) show /nu A0 def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Ax) show /nu Ax def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Ay) show /nu Ay def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Az) show /nu Az def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   x ytxt moveto (B0) show /nu B0 def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Bx) show /nu Bx def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (By) show /nu By def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Bz) show /nu Bz def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   x ytxt moveto (C0) show /nu C0 def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Cx) show /nu Cx def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Cy) show /nu Cy def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Cz) show /nu Cz def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   x ytxt moveto (Dx) show /nu Dx def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Dy) show /nu Dy def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Dz) show /nu Dz def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   mod 8 ne
   {
   x ytxt moveto (Fx) show /nu Fx def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Fy) show /nu Fy def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Fz) show /nu Fz def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   x ytxt moveto (Gx) show /nu Gx def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Gy) show /nu Gy def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Gz) show /nu Gz def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   x ytxt moveto (Hx) show /nu Hx def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Hy) show /nu Hy def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (Hz) show /nu Hz def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   } if
   } if

/xtxt  30 mm def
/ytxt 255 mm fh 31 mul sub def
/x xtxt 20 mm sub def
 mod 1 eq mod 2 eq or mod 3 eq or mod 5 eq or mod 6 eq or
 {
   x ytxt moveto (c11) show /nu c11 def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (c12) show /nu c12 def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (c13) show /nu c13 def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   x ytxt moveto (c21) show /nu c21 def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (c22) show /nu c22 def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (c23) show /nu c23 def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   x ytxt moveto (c31) show /nu c31 def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (c32) show /nu c32 def Shownum /ytxt ytxt fh sub def
   x ytxt moveto (c33) show /nu c33 def Shownum /ytxt ytxt fh sub def /ytxt ytxt fh sub def
   } if
 /fh 16 def
 /Helvetica-Bold findfont fh scalefont setfont
   x 2 mm sub 270 mm moveto txt show
} bind def

List

showpage

%%EndDocument
```

8. Ray-Triangle Test

A related task is the test whether a ray hits a triangle. The triangle is represented by one corner V and two edge vectors **f** and **h**. The ray is cast from P in direction **q** .
The calculation of the parameters f and h follows chapter 4.1.1.

$$\mathbf{x}_r = \mathbf{p} + \gamma\mathbf{q}$$
$$\mathbf{x}_p = \mathbf{v} + f\,\mathbf{f} + h\,\mathbf{h}$$

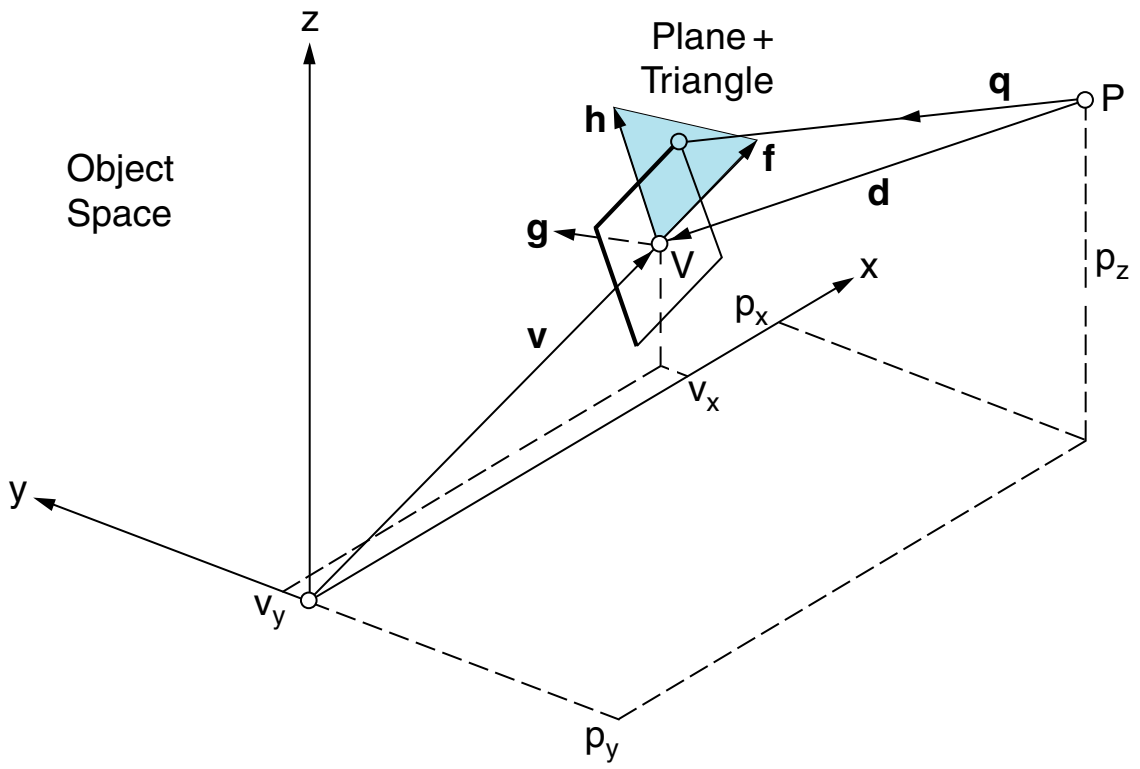
$$\mathbf{x}_r = \mathbf{x}_p$$
$$f\,\mathbf{f} + h\,\mathbf{h} - \gamma\mathbf{q} = \mathbf{p} - \mathbf{v} = -\mathbf{d}$$

$$D = \begin{vmatrix} f_x & h_x & -q_x \\ f_y & h_y & -q_y \\ f_z & h_z & -q_z \end{vmatrix} = -(\mathbf{f}\times\mathbf{h})^T\mathbf{q}$$

$$D_f = \begin{vmatrix} -d_x & h_x & -q_x \\ -d_y & h_y & -q_y \\ -d_z & h_z & -q_z \end{vmatrix} = (\mathbf{d}\times\mathbf{h})^T\mathbf{q}$$

$$D_h = \begin{vmatrix} f_x & -d_x & -q_x \\ f_y & -d_y & -q_y \\ f_z & -d_z & -q_z \end{vmatrix} = (\mathbf{f}\times\mathbf{d})^T\mathbf{q}$$

$$D_\gamma = \begin{vmatrix} f_x & h_x & d_x \\ f_y & h_y & d_y \\ f_z & h_z & d_z \end{vmatrix} = (\mathbf{f}\times\mathbf{h})^T\mathbf{d}$$



The point is inside the triangle if the following three inequalities are true. This can be checked without divisions $f = D_f/D$ etc. The intersection is calculated if the point is inside. Singularities are clearly excluded.

$0 < f$

$0 < h$

$f + h < 1$

Case $D > 0$:

$0 < D_f$

$0 < D_h$

$D_f + D_h < D$

Case $D < 0$:

$0 > D_f$

$0 > D_h$

$D_f + D_h > D$

Intersection

$f = D_f / D$

$h = D_h / D$

$\gamma = D_\gamma / D$

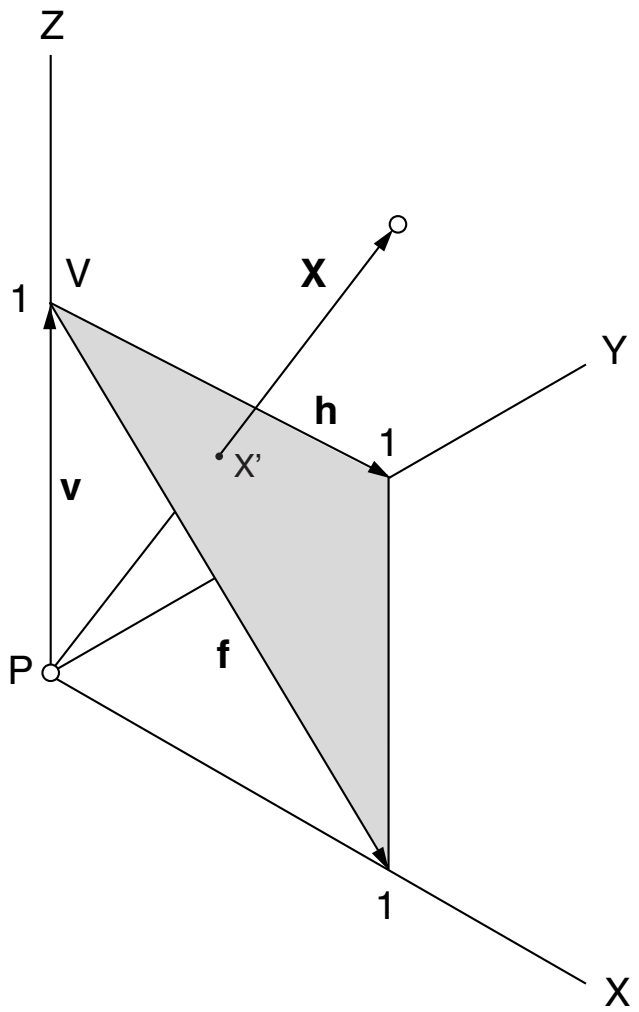
9. CIE Chromaticity Projection

The CIE (1931) color space XYZ contains all visible colors in a part of the first octant. A two-dimensional diagram xy is achieved by a sequence of projections:

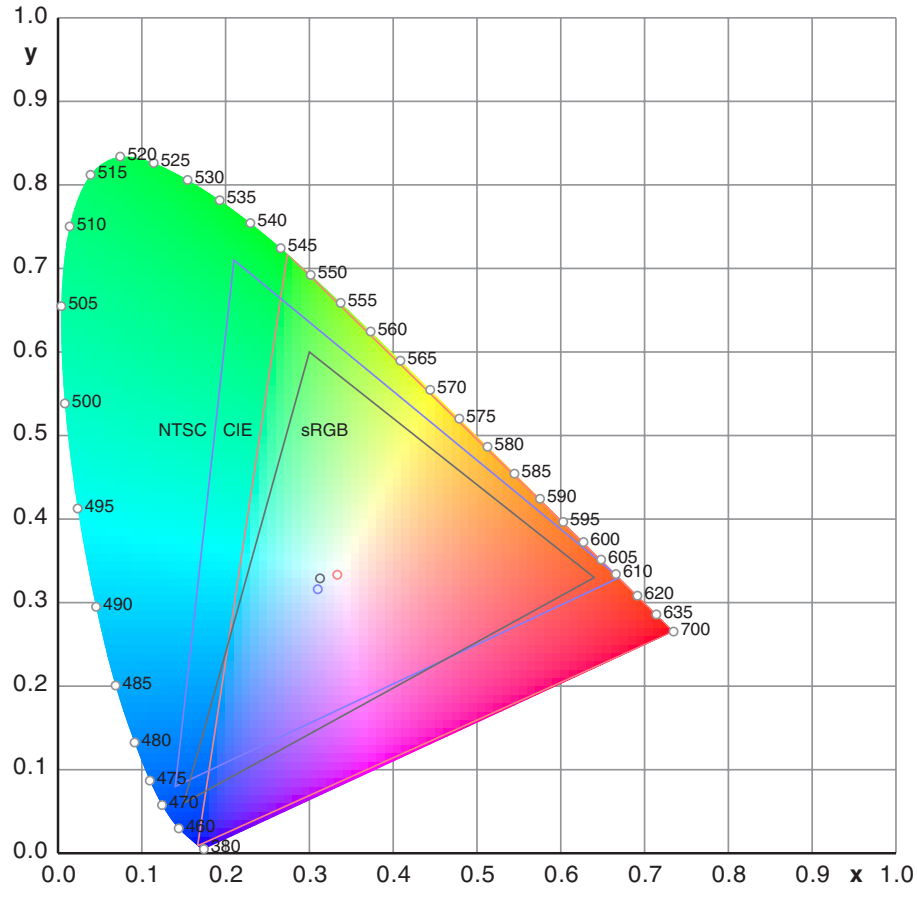
1. a perspective projection onto the plane which is defined in the drawing by the edge vectors **f** and **h** and the view point V.
2. an orthographic projection onto the horizontal plane.

Both projections are combined by defining the camera position in $\mathbf{p}=\mathbf{0}$ and the view point as shown. Instead of using the formulas in chapter 4.1.3, the method is derived once again with actual values. Both vectors $\mathbf{x}_r$ and $\mathbf{x}_p$ refer to X‘.

$$\mathbf{x}_r = \mathbf{p} + \gamma\mathbf{X} = \gamma\mathbf{X}$$
$$\mathbf{x}_p = \mathbf{v} + f\,\mathbf{f} + h\,\mathbf{h}$$
$$\mathbf{x}_p = \mathbf{x}_r$$
$$\mathbf{v} + f\,\mathbf{f} + h\,\mathbf{h} = \gamma\mathbf{X}$$
$$\begin{bmatrix}0\\0\\1\end{bmatrix} + f\begin{bmatrix}+1\\0\\-1\end{bmatrix} + h\begin{bmatrix}0\\+1\\-1\end{bmatrix} = \gamma\begin{bmatrix}X\\Y\\Z\end{bmatrix}$$
$$f = \gamma X$$
$$h = \gamma Y$$
$$1 - f - h = \gamma Z$$
$$\gamma = 1/(X+Y+Z)$$
$$f = \frac{X}{X+Y+Z} = x$$
$$h = \frac{Y}{X+Y+Z} = y$$
$$\frac{Z}{X+Y+Z} = z$$



The parameters f and h are finally renamed as x and y. The coordinate z is found by the plane equation $x+y+z=1$.



The chromaticity diagram CIE xy or CIE xyY.

10. References

[1] J.Encarnação, W.Strasser
Computer Graphics
R.Oldenbourg Verlag München Wien 1986

[2] K.W.Rüger, J.Pietschner, K.Regensburger
Photogrammetrie
VEB Verlag für Bauwesen , Berlin 1987

[3] W.-D.Fellner
Computergrafik
Wissenschaftsverlag Mannheim,Leipzig,Wien,Zürich 1992

[4] J.Hoscheck, D.Lasser
Grundlagen der geometrischen Datenverarbeitung
B.G.Teubner Stuttgart 1992

[5] A.Watt, M.Watt
Advanced Animation and Rendering Techniques
Addison-Wesley Publishing Company, Reading Massachusetts...1992

[6] J.D.Foley, A.v.Dam, S.K.Feiner, J.F.Hughes
Computer Graphics
Addison-Wesley Publishing Company, Reading Massachusetts...1993

[7] C.H.Chen, L.F.Pau, P.S.P.Wang
Handbook of Pattern Recognition & Computer Vision
World Scientific, Singapore, New Jersey, London, Hongkong 1995

[8] E.Lengyel
Mathematics for 3D Game Programming & Computer Graphics
Charles River Media, Inc. Hingham, Massachusetts 2002

[9] G.Hoffmann
Euler angles and projections
http://docs-hoffmann.de/euler26112001.pdf

[10] G.Hoffmann
Ellipse through four points
http://docs-hoffmann.de/ellipse08032004.pdf

[11] G.Hoffmann
Teach-In of a Robot by Showing the Motion / ICIP 1995
http://docs-hoffmann.de/icip01.pdf

This doc
http://docs-hoffmann.de/project18032004.pdf

Gernot Hoffmann

March 18 / 2004 – February 17 / 2013 – October 27 / 2014
Website
Load Browser / Click here