Graphics 2008/2009, period 1

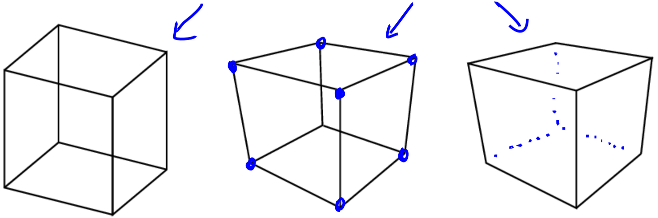
Lecture 6

Perspective projection

Orthographic vs. perspective projection

Goal: Projecting the 3D model to a 2D viewing window

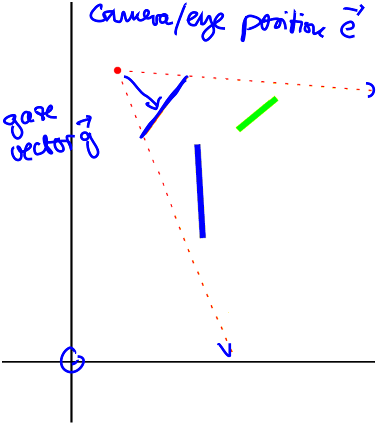
2 approaches: Orthographic vs. perspective projection



hidden
line
removal

From 3D worlds to 2D screens

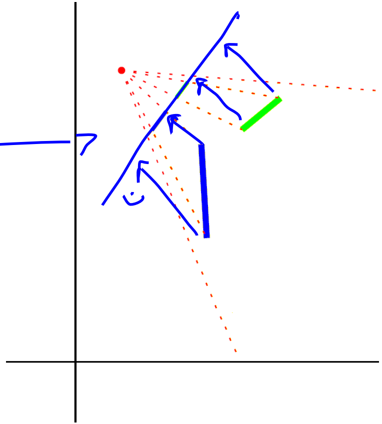
Given an arbitrary camera position, we want to display the objects in the model in an image



From 3D worlds to 2D screens

The projection should be perspective projection.

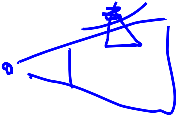
≠ orthographic projection



From 3D worlds to 2D screens

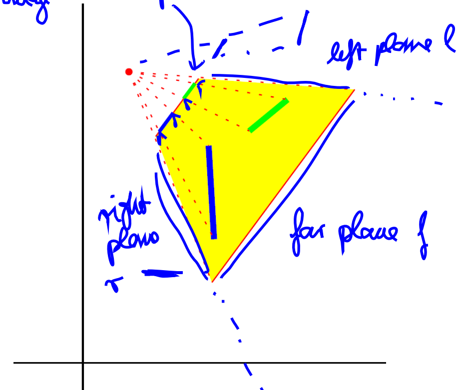
First, we define a view frustum that contains everything we want to project onto the image.

Assume: all objects are within the view frustum



Clipping → later

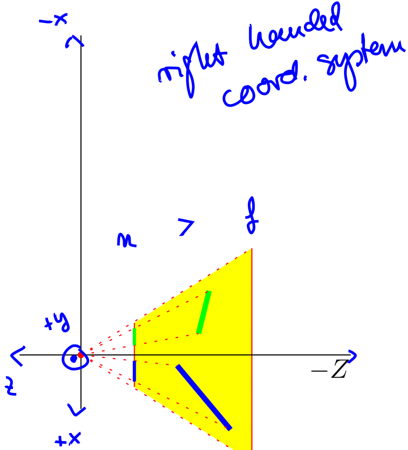
image ≙ near plane n



3d: top + bottom plane t, b

Camera transformation

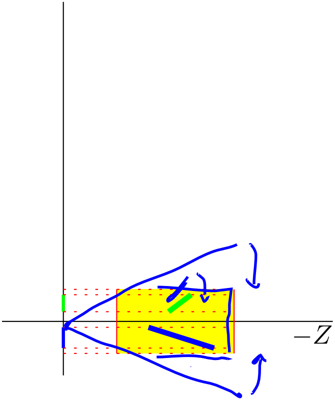
We simplify by moving the camera viewpoint to the origin, such that we look into the direction of the negative z-axis.



Orthographic projection

Orthographic projection is a lot simpler than perspective projection, so we transform the clipped view frustum to an axis-parallel box

orthographic view volume

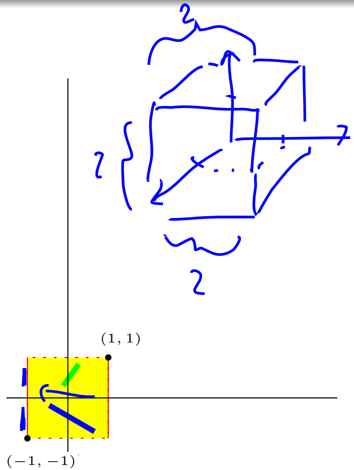


The canonical view volume

To simplify our calculations, we transform to the canonical view volume

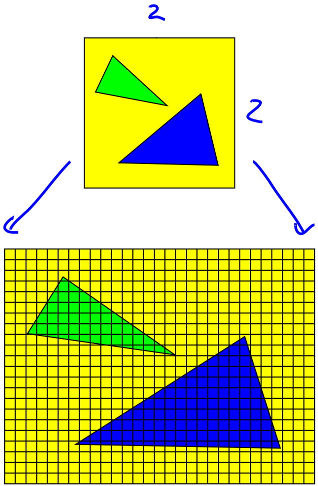
cube with sides of length 2
centered around origin

$-1 \leq x, y, z \leq 1$



Windowing transform

We apply a windowing transform to display the square $[-1,1] \times [-1,1]$ onto an $m \times n$ image.

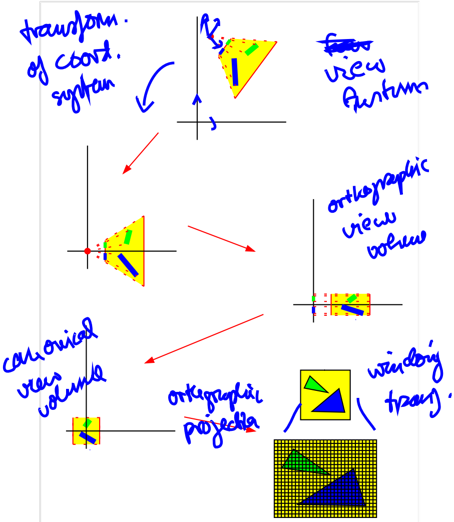


The graphics pipeline (part I)

Every step in the sequence can be represented by a matrix operation, so the whole process can be applied by performing a single matrix operation!

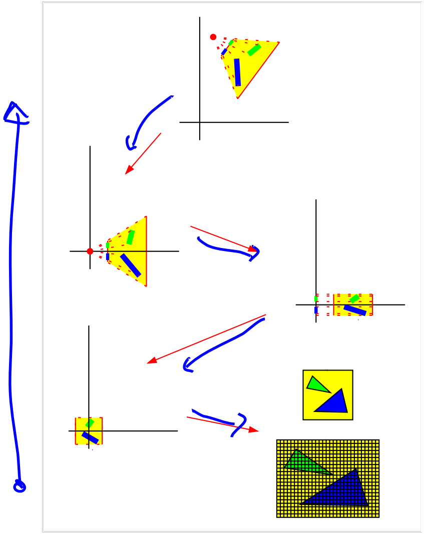
(Well: almost.)

We call this sequence a graphics pipeline.



The graphics pipeline (part I)

Graphics pipeline = a special software or hardware subsystem that efficiently draws 3D primitives in perspective.

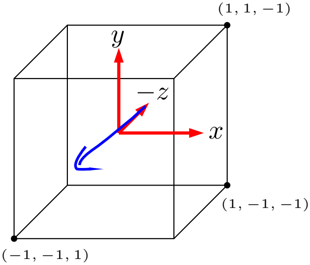


The canonical view volume

The canonical view volume is a $2 \times 2 \times 2$ box, centered at the origin.

The clipped view frustum is transformed to this box (and the objects within the view frustum undergo the same transformation).

Vertices in the canonical view volume are orthographically projected onto an $m \times n$ image.



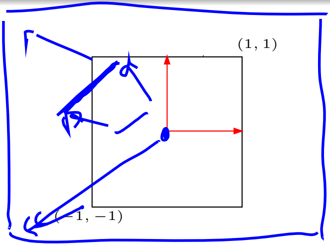
Mapping the canonical view volume

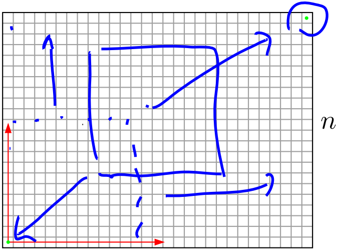
We need to map the square $[-1,1]^2$ onto a rectangle $[-\frac{1}{2}, m-\frac{1}{2}] \times [-\frac{1}{2}, n-\frac{1}{2}]$.

pixel center

The following matrix takes care of that:

$$\begin{pmatrix} \frac{m}{2} & 0 & \frac{m}{2}-\frac{1}{2} \\ 0 & \frac{n}{2} & \frac{n}{2}-\frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{m}{2}x + \frac{m-1}{2} \\ \frac{n}{2}y + \frac{n-1}{2} \end{pmatrix}$$



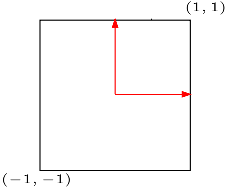


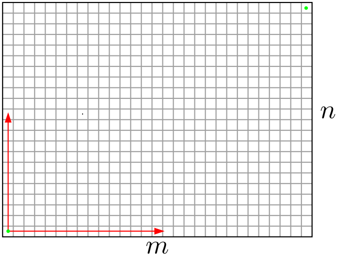
Mapping the canonical view volume

We need to map the square $[-1,1]^2$ onto a rectangle $[-\frac{1}{2}, m-\frac{1}{2}] \times [-\frac{1}{2}, n-\frac{1}{2}]$.

The following matrix takes care of that:

$$\begin{pmatrix} \frac{m}{2} & 0 & \frac{m}{2}-\frac{1}{2} \\ 0 & \frac{n}{2} & \frac{n}{2}-\frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}$$



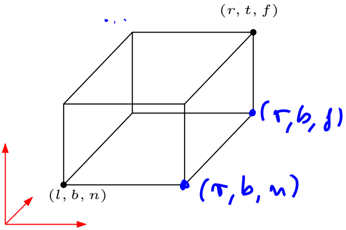


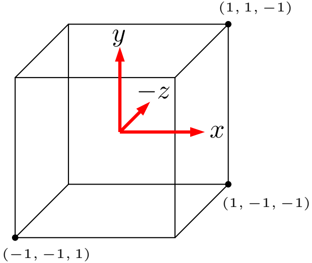
The orthographic view volume

The orthographic view volume is an axis-aligned box $[l, r] \times [b, t] \times [n, f]$.

l = left
r = right
b = bottom
t = top
n = near
f = far

plane



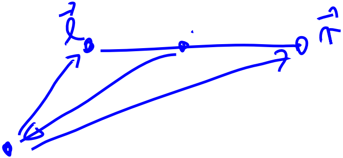


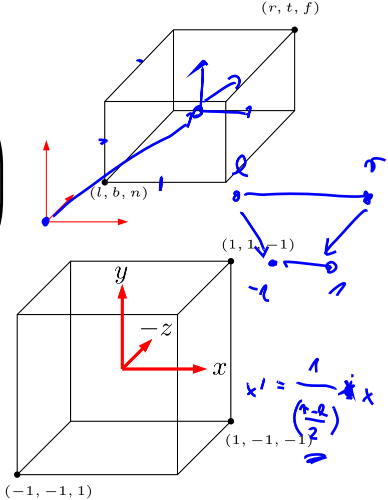
The orthographic view volume

Transformation to canonical view volume is done by

4x4

$$\underbrace{\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{scaling}} \underbrace{\begin{pmatrix} 1 & 0 & 0 & -\frac{l+r}{2} \\ 0 & 1 & 0 & -\frac{b+t}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{translation}}$$





The orthographic projection matrix

wind. tr. orthogr. proj. can. view volume orthographic view vol

We can combine the matrices into one:

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

Given a point p in the orthographic view volume, we map it to a pixel $[i, j]$ as follows:

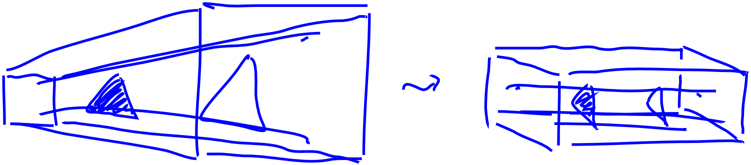
$$\begin{pmatrix} i \\ j \\ z_{canonical} \\ 1 \end{pmatrix} = M_o \begin{pmatrix} x_p \\ y_p \\ z_p \\ 1 \end{pmatrix}$$

Transforming the view frustum

(cf. book, fig. 7.12)

view frustum

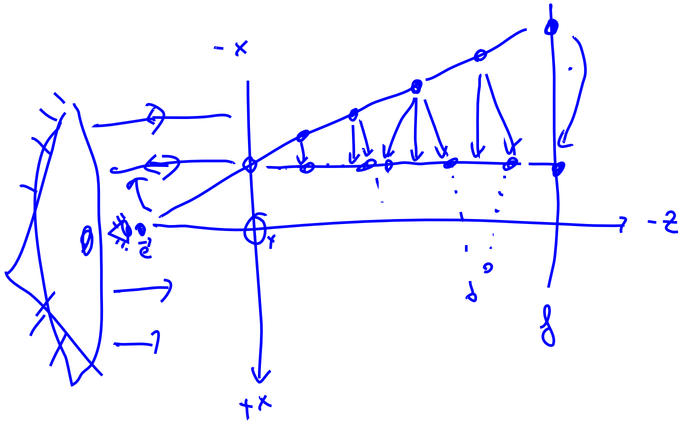
orth. view volume



Transforming the view frustum

(cf. book, fig. 7.10)

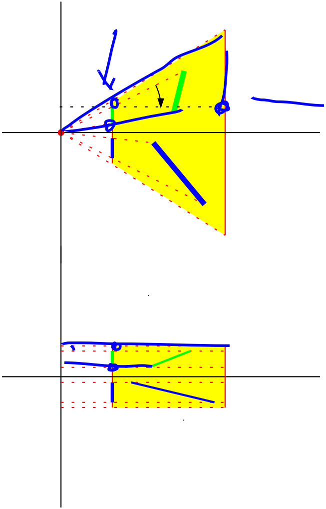
x-values (in x/z-plane)



Transforming the view frustum

We have to transform the view frustum into the orthographic view volume. The transformation needs to

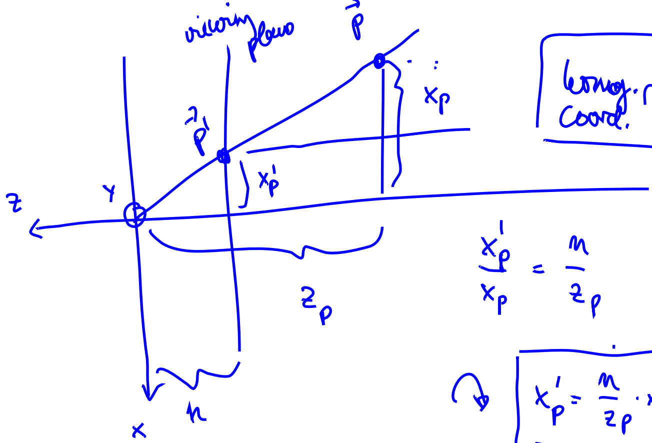
- Map lines through the origin to lines parallel to the z axis
- Map points on the viewing plane to themselves.
- Map points on the far plane to (other) points on the far plane.
- Preserve the near-to-far order of points on a line.



Transforming the view frustum

How do we calculate this? (cf. book, fig. 7.9)

x-coord.



$\boxed{\text{homog. coord.} \sim \frac{n}{z}}$

$$\frac{x'_p}{x_p} = \frac{n}{z_p}$$

$$\boxed{x'_p = \frac{n}{z_p} \cdot x_p}$$

y'_p similar

Transforming the view frustum

So we need a matrix that gives us

- $x\prime = \frac{nx}{z}$
- $y\prime = \frac{ny}{z}$

and a z-value that

- stays the same for all points on the near and fare planes
- does not change the order along the z-axis for all other points

if $z = n$
then $x' = x$
$y' = y$

Problem: we can't do division with matrix multiplication

$z = n$

$x' = \frac{nx}{z}$

Perspective transform matrix

But the following matrix M_p does the trick:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -f \\ 0 & 0 & \frac{1}{n} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z\frac{n+f}{n} - f \\ \frac{z}{n} \end{pmatrix}$$

$$\begin{pmatrix} \frac{n}{z} \cdot x \\ \frac{n}{z} \cdot y \\ z' \\ 1 \end{pmatrix}$$

$$\frac{1}{z/n}$$

Hmmmm. . . is that what we want. . . ?

Homogeneous coordinates

We have seen homogeneous coordinates before, but so far, the fourth coordinate of a 3D point has alway been 1.

In general, however, the homogeneous representation of a point (x, y, z) in 3D is (hx, hy, hz, h). Choosing $h = 1$ has just been a convenience.

So we have:

$$M_p \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z\frac{n+f}{n} - f \\ \frac{z}{n} \end{pmatrix} \xrightarrow{\text{homogenize}} \begin{pmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ n + f - \frac{fn}{z} \\ 1 \end{pmatrix}$$

Homogeneous coordinates and perspective transformation

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -f \\ 0 & 0 & \frac{1}{n} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z\frac{n+f}{n} - f \\ \frac{z}{n} \end{pmatrix} \xrightarrow{\text{homogenize}} \begin{pmatrix} \frac{nx}{z} \checkmark \\ \frac{ny}{z} \checkmark \\ n + f - \frac{fn}{z} \leftarrow ? \\ 1 \checkmark \end{pmatrix}$$

$z = n \curvearrowright n + f - f = n = z \checkmark$

$z = f \curvearrowright f\frac{n+f}{n} - f = f \checkmark$

$z > n \curvearrowright n + f - \frac{fn}{z}$

$> n + f - \frac{fn}{n} = n \checkmark$

$z < f \curvearrowright \ldots.$

$z_1 > z_2$

$n + f - \frac{f \cdot n}{z_1} > n + f - \frac{f \cdot n}{z_2}$

$-\frac{1}{z_1} > -\frac{1}{z_2}$

$z_1 > z_2 \checkmark$

Homogeneous coordinates and perspective transformation

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -f \\ 0 & 0 & \frac{1}{n} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z\frac{n+f}{n} - f \\ \frac{z}{n} \end{pmatrix} \xrightarrow{\text{homogenize}} \begin{pmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ n + f - \frac{fn}{z} \\ 1 \end{pmatrix}$$

Homogeneous coordinates

Note: this fits well in our existing framework

For example: translation

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} hx \\ hy \\ hz \\ h \end{pmatrix} = \begin{pmatrix} hx + ht_x \\ hy + ht_y \\ hz + ht_z \\ h \end{pmatrix} \xrightarrow{\text{homogenize}} \begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{pmatrix}$$

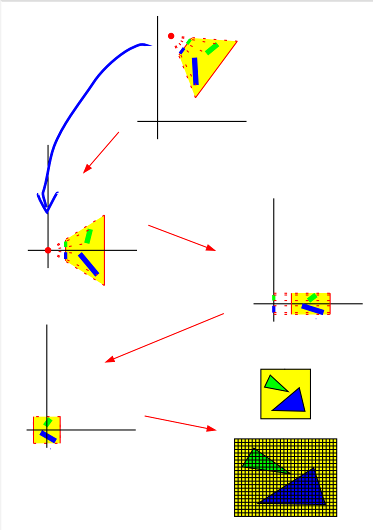
Similar for rotation, scaling, ...

Aligning coordinate systems

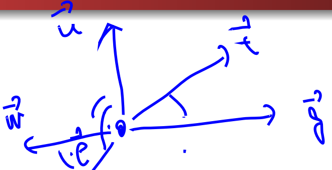
The only thing left:

Given a camera coordinate system with origin e and orthonormal base (u, v, w), we need to tranform objects in world space coordinates into objects in camera coordinates.

This can alternatively be considered as aligning the canonical coordinate system with the camera coordinate system.



Camera transformation



$\vec{t}$ = view up vector

$\vec{v}$

The required transformation is taken care of by

$$M_v = \begin{pmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

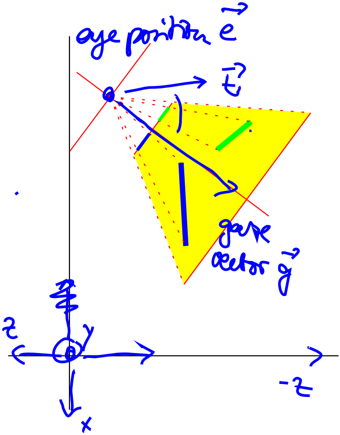
$\vec{g} \times \vec{t} = \vec{u}$

$\vec{u} \times \vec{g} = \vec{v}$

$\vec{w} = -\vec{g}$

Normalis.

$\|\cdot\| = 1$



Wrap-up

If we combine all steps, we get:

compute $\mathbf{M_o}$
compute $\mathbf{M_v}$
compute $\mathbf{M_p}$
$\mathbf{M} = \mathbf{M_o M_p M_v}$
for each line segment $(\mathbf{a_i}, \mathbf{b_i})$ do
  $\mathbf{p} = \mathbf{Ma_i}$
  $\mathbf{q} = \mathbf{Mb_i}$
  drawline$(x_p/h_p, y_p/h_p, x_q/h_q, y_q/h_q)$