Computer Graphics

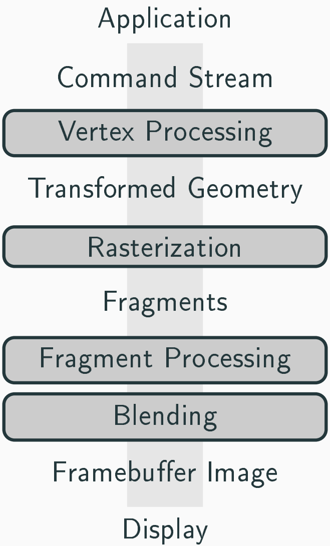
Dr. rer. nat. Martin Möddel

April 6, 2021

Institut für Biomedizinische Bildgebung

Object Ordered Rendering

The stages of a graphics pipeline



Vertex Processor

- Processes vertices
- Arrangement of geometric objects and camera
- Transform vertices to pixel space

Object space

Modeling transformation

World space

Camera transformation

Camera space

Projection transformation

Canonical view volume

Viewport transformation

Screen space

Viewing

- In OOR a series of quite simple matrix transformations are used to project points onto the 2D screen space
    1. Modeling transformation
    2. Camera transformation
    3. Projection transformation
    4. Viewport transformation
- Specific APIs may have slightly different conventions

Viewport Transformation

Canonical Viewing Volume
The canonical viewing volume is the subset $[-1, 1]^3 \subset \mathbb{R}^3$.

Screen Space
The screen space is the subset
$[-0.5, n_x - 0.5] \times [-0.5, n_y - 0.5] \subset \mathbb{R}^2$.

- Project the canonical viewing volume onto the screen space
  - $x = -1$ and $x = +1$ are projected to the left and right side of the screen respectively
  - $y = -1$ and $y = +1$ are projected to the bottom and top side of the screen respectively
- Pixels have integer coordinates (i, j), $i = 0, 1, \ldots, n_x - 1$ and $j = 0, 1, \ldots, n_y - 1$
- For now assume that all geometric objects are completely inside the viewing volume

Viewport Transformation

1. Transformation with the **viewport matrix**

$$\begin{pmatrix} x_{\text{screen}} \\ y_{\text{screen}} \\ z_{\text{canonical}} \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathrm{M}_{vp}} \begin{pmatrix} x_{\text{canonical}} \\ y_{\text{canonical}} \\ z_{\text{canonical}} \\ 1 \end{pmatrix} \tag{1}$$

2. Projection onto the screen coordinates $(x_{\text{screen}}, y_{\text{screen}})$

Remarks

- If drawing routines require pixel coordinates the screen coordinates may be rounded to the nearest pixel coordinate
- $z_{\text{canonical}}$ is kept in the transformation process for later use, e.g. **z-buffering**

Orthographic Projection

Orthographic View Volume

The orthographic view volume is $[l, r] \times [b, t] \times [f, n] \subset \mathbb{R}^3$

- l and r determine the left and right plane respectively
- b and t determine the bottom and top plane respectively
- n and f determine the near and far plane respectively

- To be able to render geometry in some region other than the canonical viewing volume the projection and camera transformation are required
- For now special case of orthographic view is handled (perspective view later)
- The orthographic projection allows us to view the **orthographic view volume**

Orthographic Projection

- Assume a camera with gazing direction along its $-z$-direction and upwards direction along y
- The orthographic projection allows us to view the **orthographic view volume**
- Due to the viewing direction being $-z$ we have $n > f$ which might seem counter intuitive
- The matrix transformation transforming from camera space to the canonical view volume is

$$\begin{pmatrix} x_{\text{canonical}} \\ y_{\text{canonical}} \\ z_{\text{canonical}} \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathrm{M}_{orth}} \begin{pmatrix} x_{\text{camera}} \\ y_{\text{camera}} \\ z_{\text{camera}} \\ 1 \end{pmatrix} \tag{2}$$

Camera Transformation

- Assume a camera with gazing direction along its $-z$-direction and upwards direction along y
- A convention to specify the camera position and orientation in **world coordinates**
  - the eye position e where the observer views from
  - the gaze direction g where the observer is looking towards
  - the view-up vector t which lies in the plane, which bisects the viewers head into left and right

Camera Transformation

- The three directions can be used to construct an orthonormal basis $(\mathrm{x}_{\mathrm{camera}}, \mathrm{y}_{\mathrm{camera}}, \mathrm{z}_{\mathrm{camera}})$ for the camera

$$\mathrm{z}_{\mathrm{camera}} = -\frac{\mathrm{g}}{\|\mathrm{g}\|_2} \tag{3}$$

$$\mathrm{x}_{\mathrm{camera}} = \frac{\mathrm{t} \times \mathrm{z}_{\mathrm{camera}}}{\|\mathrm{t} \times \mathrm{z}_{\mathrm{camera}}\|_2} \tag{4}$$

$$\mathrm{y}_{\mathrm{camera}} = \mathrm{z}_{\mathrm{camera}} \times \mathrm{x}_{\mathrm{camera}} \tag{5}$$

- According to the lecture on transformations we can use this basis to transform from world into camera coordinates by

$$\begin{pmatrix} x_{\mathrm{camera}} \\ y_{\mathrm{camera}} \\ z_{\mathrm{camera}} \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} \mathrm{x}_{\mathrm{camera}} & \mathrm{y}_{\mathrm{camera}} & \mathrm{z}_{\mathrm{camera}} & \mathrm{e} \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1}}_{\mathrm{M}_{cam}} \begin{pmatrix} x_{\mathrm{world}} \\ y_{\mathrm{world}} \\ z_{\mathrm{world}} \\ 1 \end{pmatrix} \tag{6}$$

Modeling Transformation

- Each object has its own object space attached
- Transform objects from object space to world space

$$\begin{pmatrix} x_{\text{world}} \\ y_{\text{world}} \\ z_{\text{world}} \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} \mathrm{A} & \mathrm{t} \\ 0^T & 1 \end{pmatrix}}_{\mathrm{M}_{mod}} \begin{pmatrix} x_{\text{object}} \\ y_{\text{object}} \\ z_{\text{object}} \\ 1 \end{pmatrix} \tag{7}$$

- A and t depend on application
- Might be as simple as a translation
- Might be more sophisticated

Complete Transformation

- From the individual building blocks a single transformation can be build
- It transforms coordinates from object space directly to screen space

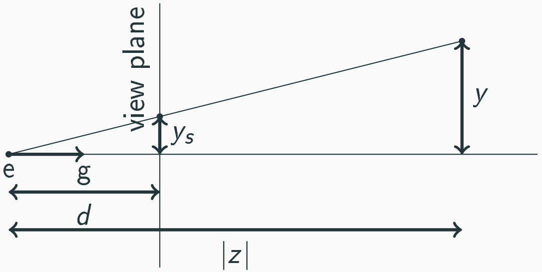
$$\begin{pmatrix} x_{\text{screen}} \\ y_{\text{screen}} \\ z_{\text{canonical}} \\ 1 \end{pmatrix} = \underbrace{\mathsf{M}_{\text{vp}} \mathsf{M}_{\text{orth}} \mathsf{M}_{\text{cam}} \mathsf{M}_{\text{mod}}}_{\mathsf{M}} \begin{pmatrix} x_{\text{object}} \\ y_{\text{object}} \\ z_{\text{object}} \\ 1 \end{pmatrix} \tag{8}$$

- For the overall performance it is most often beneficial to recompute M as early as possible

Projective Projection

- In the camera coordinate system the viewing direction is along the negative z-axis
- The image is projected onto a plane at distance d in front of the eye
- The object size on the screen depends on d and its distance to the eye $|z|$

$$y_s = \frac{d}{|z|} y \tag{9}$$



Projective Projection - Projective Transformations

- The transformation $\frac{d}{|z|}y$ is not affine so we need an extension of our homogeneous space transformations
- To do so we allow for the transformations to change the extra coordinate which remains 1 otherwise (it is not required to transform normal vectors to this point)
- Moreover we consider all points $\mathrm{x}, \mathrm{y} \in \mathbb{R}^4$ equivalent for which there exists an $\alpha \in \mathbb{R} \setminus \{0\}$

$$\mathrm{y} = \alpha \mathrm{x} \tag{10}$$

- The 3D coordinates (x', y', z') are obtained not from any 4D vector $(\tilde{x}, \tilde{y}, \tilde{z}, \tilde{w})$, $w \neq 0$ in the equivalence class, but from $(\tilde{x}/\tilde{w}, \tilde{y}/\tilde{w}, \tilde{z}/\tilde{w}, 1)$

Projective Projection - Projective Transformations

- By doing so linear rational functions

$$x' = \frac{a_x x + b_x y + c_x z + d_x}{ex + fy + gz + h}$$
$$y' = \frac{a_y x + b_y y + c_y z + d_y}{ex + fy + gz + h}$$
$$z' = \frac{a_z x + b_z y + c_z z + d_z}{ex + fy + gz + h}$$

  can be implemented by a homogeneous matrix transform and subsequent projection onto the 3D space

Projective Projection - Projective Transformations

- First the matrix transformation

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \\ e & f & g & h \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{11}$$

- Second the homogeneous divide

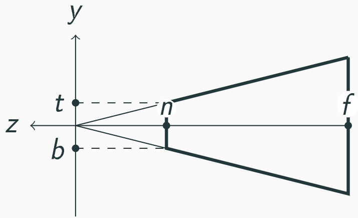
$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \frac{\tilde{x}}{\tilde{w}} \\ \frac{\tilde{y}}{\tilde{w}} \\ \frac{\tilde{z}}{\tilde{w}} \end{pmatrix} \tag{12}$$

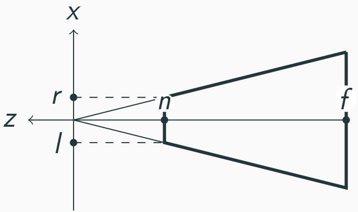
Remark

If we concatenate transformations later on we can choose at which point the homogeneous divide is performed, so we can perform all matrix-vector multiplications in homogeneous space prior to the homogeneous divide.

Projective Projection - The Projective View Volume

- The projective viewing volume is defined by the same values as the orthographic viewing volume
  - l and r determine the left and right plane respectively
  - b and t determine the bottom and top plane respectively
  - $0 > n > f$ determine the near and far plane respectively
- The details work out a little bit differently





Projective Projection - Perspective Projection

- Projection onto the viewing plane is done in a two step procedure
    1. The projective view volume is mapped to the orthographic view volume by

$$\begin{pmatrix} \tilde{x}_{\text{ocamera}} \\ \tilde{y}_{\text{ocamera}} \\ \tilde{z}_{\text{ocamera}} \\ \tilde{w}_{\text{ocamera}} \end{pmatrix} = \underbrace{\begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathrm{P}} \begin{pmatrix} x_{\text{camera}} \\ y_{\text{camera}} \\ z_{\text{camera}} \\ 1 \end{pmatrix} \tag{13}$$

    2. The orthographic view volume is mapped to the canonical view volume by

$$\begin{pmatrix} \tilde{x}_{\text{canonical}} \\ \tilde{y}_{\text{canonical}} \\ \tilde{z}_{\text{canonical}} \\ \tilde{w}_{\text{canonical}} \end{pmatrix} = \mathrm{M}_{orth}(r, l, t, b, n, f) \begin{pmatrix} \tilde{x}_{\text{ocamera}} \\ \tilde{y}_{\text{ocamera}} \\ \tilde{z}_{\text{ocamera}} \\ \tilde{w}_{\text{ocamera}} \end{pmatrix} \tag{14}$$

Projective Projection - Perspective Projection

Remarks

- The actual perspective projection matrix is then simply $\mathrm{M}_{\mathrm{per}} = \mathrm{M}_{\mathrm{orth}}\mathrm{P}$ and

$$\begin{pmatrix} \tilde{x}_{\text{canonical}} \\ \tilde{y}_{\text{canonical}} \\ \tilde{z}_{\text{canonical}} \\ \tilde{w}_{\text{canonical}} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{l+r}{l-r} & 0 \\ 0 & \frac{2n}{t-b} & \frac{b+t}{b-t} & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathrm{M}_{per}} \begin{pmatrix} x_{\text{camera}} \\ y_{\text{camera}} \\ z_{\text{camera}} \\ 1 \end{pmatrix} \tag{15}$$

- The matrix P does not change points on the near plane, but squishes them on all other planes by the correct amount

Projective Projection - Perspective Projection

Remarks

- The transform also preserves the relative z-order of points inside the perspective view volume $n \geq z_{\text{camera}} \geq f$

$$z_{\text{ocamera}} = n + f - \frac{fn}{z_{\text{camera}}} \tag{16}$$

Let $n \geq z_n > z_f \geq f$ then

$$\frac{fn}{z_n} < \frac{fn}{z_f} \tag{17}$$

$$\Rightarrow n + f - \frac{fn}{z_n} > n + f - \frac{fn}{z_f} \tag{18}$$

- Lines are mapped to lines
- Planes are mapped to planes

Projective Projection - Perspective Projection

Remarks

- Sometimes the inverse of P

$$\mathrm{P}^{-1} = \begin{pmatrix} \frac{1}{n} & 0 & 0 & 0 \\ 0 & \frac{1}{n} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{1}{fn} & \frac{n+f}{fn} \end{pmatrix} \tag{19}$$

  is required, e.g. for picking a screen and z-coordinate in world space

Projective Projection - Perspective Projection

Remarks

- We can rescale any 4×4 matrix without changing the underlying 3D mapping

$$P^{-1} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & fn \\ 0 & 0 & -1 & n+f \end{pmatrix} \tag{20}$$

- This is possible due to the equivalence relation we introduced earlier
- Why? Cause math is awesome!

Culling

- For now we assumed all objects are inside the view volume
- Let us now assume that objects may also be inside the view volumes complement
- Especially in a large scene much of the geometry might be outside the view volume, occluded or facing away from the observer and therefore not contributing to the final image
- The process of identifying and removing such geometry from the pipeline is known as **culling**
- There are three common culling procedures
    - View volume culling
    - Occlusion culling
    - Backface culling

View Volume Culling

- Test if a primitive lies outside the view volume
- Remove the object from the pipeline if so
- Most obvious would be to test all triangles of the object against all faces of the view volume
- A triangle is outside the view volume if it is located at the outside side of at least one of the view volume planes
- Better to perform the test on a bounding volume of the object, e.g. intersection sphere and viewing volume
- Even better to use the spatial data structures we discussed in ray tracing

Occlusion Culling

- Test if primitive is occluded by other objects in scene
- Remove the object from the pipeline if so
- This is a quite complex topic with many approaches
    - Precomputed culling (Smartphone), e.g. precomputed visibility volumes
    - Software occlusion culling, e.g. Masked Software Occlusion Culling
    - GPU assisted culling, e.g. depth buffer reprojection
    - GPU driven culling (high-end GPU), e.g. GPU-driven rendering pipelines

Backface Culling

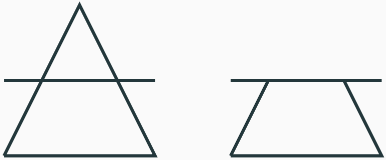
- Consider closed polygonal models bounding a closed space with no holes in the surface
- Common assumption on the surface normal of each primitive is that it faces outwards
- In such a case primitive facing away from the viewer can never be visible
- Removed such primitives

Clipping

- The only case remaining now is where primitives intersect with the view volume boundaries
- This case has to be handled with caution since the perspective projection transformation may map points outside the view volume to nonsensical locations
- More specifically, the problem is that points behind the eye might get mapped to points in front of the eye behind the far plane
- The **clipping** operation removes parts that could extend behind the eye
- More general clipping is an operation, where primitives are cut by some geometric entity (e.g. plane) and the part cut off is discarded, whereas the other part is kept

Clipping

- The simples scenario is where a triangle is clipped against a plane



  - Triangle might be on the outside side of the cutting plane in which case it is removed
  - Triangle might be on the inside side of the cutting plane in which case it is kept
  - Triangle might lie inside the plane in which case it is kept
  - Triangle might be cut in two by plane in which case the part on the outside side is removed
  - If quadrilateral remaining, split it up into two triangles

Clipping

- The two most common approaches for clipping are
  1. In world/camera coordinates before the projection transformation
  2. In 4D homogeneous space before the homogeneous divide
- In both cases we have to calculate the intersection of triangle edges (line segments) and hyperplanes

Line Hyperplane Intersection

- Using one point q on the plane and the normal vector n we can describe the plane implicitly by

$$f(\mathsf{p}) = \mathsf{n} \cdot (\mathsf{p} - \mathsf{q}) = 0 \tag{21}$$

- The line segment between two vertices a or b is given by

$$\mathsf{l}(t) = \mathsf{a} + t(\mathsf{b} - \mathsf{a}) \; 0 \leq t \leq 1 \tag{22}$$

- If (a − b) and n non-orthogonal, intersection occurs at

$$t = \frac{\mathsf{n} \cdot (\mathsf{a} - \mathsf{q})}{\mathsf{n} \cdot (\mathsf{a} - \mathsf{b})} \tag{23}$$

Clipping in Homogeneous Space

Lemma

Let $\mathrm{n} \in \mathbb{R}^3$ be the normal vector and $\mathrm{q} \in \mathbb{R}^3$ a point on a Hyperplane in $\mathbb{R}^3$, such that $d = -\mathrm{n} \cdot \mathrm{q} \neq 0$. The hyperplane in homogeneous space, which maps onto the 3D hyperplane by homogeneous divide, is defined by the implicit equation

$$f\left(\begin{pmatrix} \mathrm{p} \\ w \end{pmatrix}\right) = \begin{pmatrix} \mathrm{n} \\ d \end{pmatrix} \cdot \left(\begin{pmatrix} \mathrm{p} \\ w \end{pmatrix} - \begin{pmatrix} \mathrm{q} \\ 1 \end{pmatrix}\right) = 0. \tag{24}$$

Sketch of the Proof

- Let (p, w), $w \neq 0$ a point on the plane then either
  1. $w = \frac{\mathrm{n} \cdot \mathrm{p}}{\mathrm{n} \cdot \mathrm{q}}$
  2. $w = 1$ and $\mathrm{n} \cdot (\mathrm{p} - \mathrm{q}) = 0$
- In both cases (p, w) is mapped to $w^{-1}\mathrm{p}$ by homogeneous divide and satisfies the implicit equation $\mathrm{n} \cdot (w^{-1}\mathrm{p} - \mathrm{q}) = 0$ for the 3D Hyperplane