Computer Graphics Lecture Notes

Haoyu Zhen

July 28, 2022

Contents

Acknowledgement

These Notes contain material developed and copyright by CS148: Introduction to Computer Graphics and Imaging, Stanford University. To get more information, you could visit the course website: `https://web.stanford.edu/class/cs148/index.html`.

1 Geometry

Loop Subdivision

- Subdivide each triangle into 4 sub-triangles
- Move both the old/new vertices
- Repeat (if desired)
- C2 continuity almost everywhere (except at some extraordinary vertices where its only C1)

Cardinal Cubic Splines

4 control points $\{p_{i-1}, p_i, p_{i+1}, p_{i+2}\}$. What we want is $f(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$ such that

$$f(0) = p_i, \quad f(1) = p_{i+1}, \quad f'(0) = s(p_{i+1} - p_{i-1}) \quad \text{and} \quad f'(1) = s(p_{i+2} - p_i).$$

1.1 Transformation

Rotation

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}, \quad R_y = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \quad \text{and} \quad R_z = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Scaling

$$S = \begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_1 & 0 \\ 0 & 0 & s_3 \end{pmatrix}.$$

1.2 Homogeneous Coordinates

Definition 1.1 (Homogeneous Coordinates). **Goal**: to represent translation with matrices. In general, the homogeneous coordinates of a point in 3D are:

$$\vec{p}_H = \begin{pmatrix} wx & wy & wz & w \end{pmatrix}^T$$

where $w \neq 0$.

Let the forth component be 1, we have:

$$\vec{p}_H = \begin{pmatrix} x & y & z & 1 \end{pmatrix}^T.$$

Finally,

$$\begin{pmatrix} \boldsymbol{M} & \boldsymbol{t} \\ \boldsymbol{O} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}\boldsymbol{x} + \boldsymbol{t} \\ 1 \end{pmatrix}$$

where $\boldsymbol{M} \in \mathbb{R}^{3\times 3}$ and $\boldsymbol{O} \in \mathbb{R}^{1\times 3}$.

2 Color and Image

2.1 Rasterization

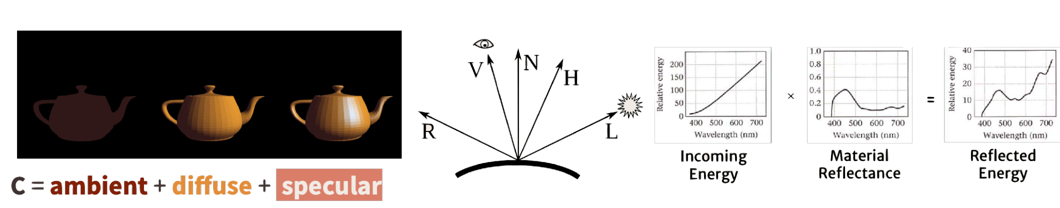
For each pixel, if the center of the pixel is inside the triangle, consider it part of the triangle (and color it with the triangles color.)

2.2 Phong Reflection Model

$$\begin{aligned} C &= \text{ambient} + \text{diffuse} + \text{specular} \\ &= k_a + k_d \max(0, \hat{N} \cdot \hat{L}) + k_s \Big(\max(0, \hat{V} \cdot \hat{R})\Big)^\alpha \end{aligned}$$

where $\hat{N}, \hat{L}, \hat{V}$ and $\hat{R}$ are defined as follows.

- **Ambient**: "Base color". Light bounces arounf the environment.
- **Diffuse**: "Rough marterial". Given the same light source, objects look brighter when hit "perpendicularly".
- **Specular**: "Shiny material". Bright highlights when light reflects into our eyes.



3 Camera

The mathematical form of the projection:

$$p_{\text{img}} = M_{\text{obj2img}} \cdot p_{\text{obj}} = M_{\text{cam2img}} M_{\text{world2cam}} M_{\text{obj2world}} \cdot p_{\text{obj}}$$

3.1 Camera to Film

3.1.1 Modeling the Pinhole Camera

$$P(x, y, z) = \begin{pmatrix} hx/z & hy/z & h \end{pmatrix}.$$

Condsider homogeneous coordinates:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{0} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} h & 0 & 0 & a \\ 0 & h & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{0} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

3.1.2 Viewing Frustrum

- Idea: transform viewing frustrum to an orthographic bounding box
- Instead of projecting everything to the plane at fixed depth, project to a cube first
- Clip anything thats not within $-1 \to 1$

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} 1/r_x & 0 & 0 & 0 \\ 0 & 1/r_y & 0 & 0 \\ 0 & 0 & \dfrac{d_0 + d_1}{d_0 - d_1} & \dfrac{2 d_0 d_1}{d_0 - d_1} \\ 0 & 0 & \mathbf{-1} & \mathbf{0} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

where we have r_x, r_y corresponding to film boundaries, d_0/d_1 being the clipping planes.

3.1.3 *z*-buffer Algorithm

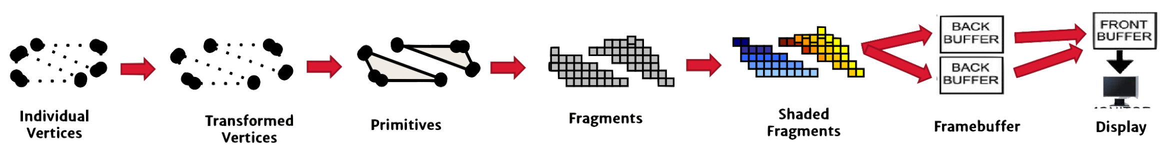
Then we introduce the ***z*-buffer algorithm**:

Algorithm 1 z-buffer algorithm (output: $\mathcal{C}$ and $\mathcal{Z}$)

```
For all x and y, Z(x, t) ← −∞
for Each (x, y, z) in each fragment with corresponding color c do
    if Z(x, y) < z then
        C(x, y) ← c and Z(x, y) ← z
    end if
end for
```

3.2 The rasterization pipeline

1. Vertex shader outputs vertex attributes and orthogonal clipping space positions
2. Vertex data assembled into triangles, data outside the viewing frustrum is discarded
3. For each triangle, rasterize vertex attributes (**barycentric interpolation**)
4. Fragment shader takes interpolated attributes and computes color
5. Use the z-buffer to help assemble pixel colors into the full image



4 Optics

Definition 4.1. **Radiant Intensity** of a light source: $I(\omega) = \mathrm{d}\Phi/\mathrm{d}\omega$

- Total light power (exiting a light) per unit solid angle
- Measure of how strong a (point) light source is

Definition 4.2. **Irradiance** on a surface: $E = \mathrm{d}\Phi/\mathrm{d}A$

- Total light power (hitting a surface) per unit surface area.
- Measure of how much light is hitting a surface.
- Varies based on distance from the light and the tilting angle of the surface.

Some engineering approximations are as follows.

- BRDF (Bidirectional Reflectance Distribution Function): models how much light is reflected.
- BTDF (Bidirectional Transmittance Distribution Function): models how much light is transmitted.
- BSSRDF (Bidirectional Surface Scattering Reflectance Distribution Function): combined reflection/transmission model.

Now we define the **lighting equation**:

$$L_o(\omega_0) = \sum_{i\in\text{in}} L_{o\text{ due to }i}(\omega_i, \omega_o)$$

where the BRDF gives each of $L_{o\text{ due to }i}(\omega_i, \omega_o)$. Then we have

$$L_o(\omega_0) = \int_{i\in\text{in}} \mathrm{BRDF}(\omega_i, \omega_0)\,\mathrm{d}E_i(\omega_i) = \int_{i\in\text{in}} \mathrm{BRDF}(\omega_i, \omega_0) L_i \cos\theta_i \,\mathrm{d}\omega_i\,.$$

Diffuse Materials: a surface reflects light equally in all directions. I.e., $\mathrm{BRDF} = \mathrm{Const.}$

5 Ray Tracing

5.1 Constructing Rays

Shoot a ray for each pixel:

$$R(t) = A + (P - A)t$$

where A is the aperture, P is the pixel center and t is defined $t \in [0, +\infty)$.

5.2 Acceleration

Parallelization: each ray is independent of any other ray

ad