1 Display and Color

1.1 Display Devices and terms

Emissive: we have electrodes shooting (CRT, FED, LED, OLED), **non-emissive:** we manipulate light (LCD) Terms: **resolution, viewing angle, pixel density (PPI), brightness, black level, contrast, color gamut, stereopsis (3D), motion parallax, (auto)stereoscopic (1,2), multi-view, (1): circularly polarized glasses, active shutter gl, (2): lenticular lens, parallax barriers, multi-view: 2D, (auto)stereoscopic and how, frame buffer, resolution: #scan lines, #pixels per scan line, bits per pixel; color tables, look up tables (LUT), RGB, CMY, HSV, YIQ, CIE, gamuts, alpha channel, double buffering,**

2 2D and 3D Maths

Represent (x, y) as $(\frac{x}{w}, \frac{y}{w}, w)$ in hom. coordinate rep.,where $w = 1$, 0 for infinite distance.

2.1 Transformation matrixes

Scaling:$\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$ Rotation:$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ Shear in x: $\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}$ Translation: $\begin{pmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{pmatrix}$ Rotation about X-axis: $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, for y and z swap values around such that y and z row and column only have 1 and 0s Reflect in $y = mx+b$: $\frac{1}{1+m^2}\begin{pmatrix} 1-m^2 & 2m & -2mb \\ 2m & m^2-1 & 2b \\ 0 & 0 & 1+m^2 \end{pmatrix}$

Reflection over XY (change for YZ, XZ): $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Always use fixed point rule: translate to origin, scale, rotate, shear, translate back to fixed point

- Rigid: Translation + rotation (distance preserving)
- Similarity: Translation + rotation + uniform scaling (angle preserving)
- Affine: Translation + rotation + scaling + shear (parallelism preserving)

3 Hierarchical Modeling

- Render polygons: triangle strip (preferred) and triangle fan (central vertex)
- Data structure: vertex set (seq memory access, - duplicated vertices), indexed face set (reuse vertices and compact, - random memory access)
- Hierarchical model: hierarchy of objects and its subobjects
- Viewworld transformation: Object space → world space → eye space: $M_{world2eye} \times M_{obj2world \times P_{obj}}$
- Modelling glitches: T-join (not touching due to float computations) break into two triangles, overlapping polygons (flipping colors/z fighting) add triangle or turn off depth test

4 Interactive 3D control

We can use the mouse for interactive rotation, we need to construct a rotation matrix for that:

1. Put the mouse motion vector in XY eye space: (dx, dy, 0)
2. Axis of rotation perpedicular to the motion vecotr (-dy, dx, 0)
3. Angle of rotation relative to motion vector length $\sqrt{dx^2 + dy^2}$

Make sure to do it from a fixed point.

5 Camera, Projection, Clipping

Physical camera models:

1. Pin-hole: (+) easy to simulate, everything in focus, (-) need bright scene, everything in focus
2. Lenscamera: (+) no need for bright scene, not everything in focus, (-) need more programming, not everything in focus

Camera/OpenGL requires objects, viewer and projection Viewing coordinate transformation: $Z_v = \frac{P-L}{|P-L|}, X_v = \frac{X \times Z_v}{|V \times Z_V|}, Y_v = Z_v \times X_v$ $M = \begin{bmatrix} x_v^x & x_v^y & x_v^z & 0 \\ y_v^x & y_v^y & y_v^z & 0 \\ z_v^x & z_v^y & z_v^z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = R \cdot T$

- Perspective projection: all rays converge at COP
- Orthographic projection: all rays are parallel
  - Oblique Projection: DOP not perpendicular to PP
    * Cavalier Projection: DOP makes 45d angle with PP, preserves lengths
    * Cabinet Projection: DOP makes 64.4d angle with PP, foreshorten lines by one-half
  - Orthographic projection: DOP perpedicular to PP
    * Orthogonal Projection: DOP align with an axis, implement by throwing away an axis
    * Axonometric Projection: arb DOP, not parallel to world X, Y, Z-axis. isometric 120d.

$$P_{or} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} P_{pers} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Multiply with eye coordinate

6 Clipping

Goal: eliminate parts outside the viewing frustum Clipping Reasons:

- Avoid degeneracies
- Efficiency
- Rasterization is expensive

Clipping is taking a part out, culling is taking a whole part out. No difference to final image.

For point (x,y,z) clipping: Let's have a plane H (A,B,C,D) such that $Ax+By+Cz+D = 0$, assume $D = 1$ for now. If $d = H \cdot p = \frac{(A,B,C,D)(a,b,c,1)}{\sqrt{A^2+B^2+C^2}} \geq 0$ inside. 'd' is signed distance. Test against each 6 planes, $(0, 0, -1, -near), (0, 0, 1, far), (0, d, h, 0), (0, d, -h, 0), (d, 0, h, 0), (d, 0, -h, 0)$.

For a line segment pq check $H \cdot p, H \cdot q$ clip the part that's outside. Find clipping point by intersecting $L(t) = P_0 + t \cdot (P_1 - P_0)$ with the plane's equation

6.1 Cohnen Sutherland

Outcodes: $\begin{bmatrix} 1010 & 0010 & 0110 \\ 1000 & 0000 & 0100 \\ 1001 & 0001 & 0101 \end{bmatrix}$, we have the algorithm:

1. Outcode end-point pairs are checked for trivial acceptance
2. If it can't be trivially accepted, do region check for trivial rejection
3. If neither, subdivide so that one or both segments can be discard
4. Repeat until we can trivially accept or reject

6.2 Polygon Clipping

Polygon clipping is symmetric, even when convex. When concave is hard. Naive method requires $N \cdot m$ intersections.

6.3 Weiler-Atherton Clipping

- Walk around polygon/window boundary CCW
- Compute intersection points
- Mark where polygons enter and leave clipping window
- Keep those that are inside the window

7 Rasterization

7.1 Triangles

- Scan line coherence: long horizontal stretches of same fill value
- Edge coherence: easy to use line agorithm to calculate line's end points

7.2 Flood Fill vs Boundary Fill

Both need seed x,y and fill color

- **Boundary Fill:** need boundary color, ignore existing non-boundary colors, bresenhamp stairstep boundary ok (4C), must have no diagonal gaps in boundary (8C)
- **Flood Fill:** color at seed pixel, replace seed color, ignores boundary color, fills stairs only (4C), fills diagonally adjacent (8C)

7.3 Anti-aliasing

Issues with aliasing: jagged profiles, loss of details, disintegrating textures. Two concepts

- Pre-filtering: treat each pixel as an area, compure relative area ratio and mix colours
- Post-filtering: combine multiple samples per pixel

AA methods: SSAA: supersampling (render higher resolution then downsample), MSAA: multisampling (easy but hardware dependent), FXAA: fast approximate (find edges and smooth edges), TXAA: specifically to reduce temporal aliasing.

We can also have aliasing in temporal domain (time), so we can use motion blur to fix this.

8 Hidden Surface Removal

Some parts are not visible to us at all, even after clipping since they could be hidden behind something. Methods: **Object order:** consider each 3D obj once, draw pixels and move on, might draw same pixel, e.g. *Z-Buffer* .
Image order: consider each pixel once, might compute relationships between objects multiple times, e.g. *Ray Casting*
Sort first: find some depth-based ordering and draw from back to front, e.g. *Painter's algorithm*
Sort last: sort implicitly as more information becomes available, e.g. *Z-buffer* **Ray Casting:** for each pixel p_i construct ray from COP through PP at that pixel and into scene, intersect the ray with every object, color the pixel according to the object with closest intersection: $R(t) = (1-t)c + tp_i$
Z-buffer: keep track of depth, if a different pixel is closer replace. **Z-value interpretation:** bilionear interpolation or barycentric coordinates
Back Face Culling: given a solid geometry, we can cull polygons facing away from the viewer. Using the normal vector, a polygon is front facing when $N_p \cdot V > 0$. Works completely for single closed convex object, not for multiple obects, not for concave object, usually only as preprocessing step. **Potentially Visible Set:** divide scenes into cells, precompute objects that may be visible in each cell. (+) speed up, (-) memory, computing time, dynamic scenes (?)

9 Lighting, Material, Shading

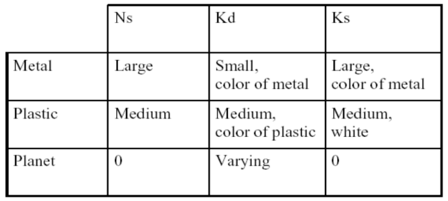
Types of light sources

1. Directional: infinity away, hom. coordinate (X,Y,Z,0), light parallel, easy comp
2. Point: finite distance, hom. coordinate (X,Y,Z,1), all direct, hard comp, attennuation: dist. dimmer
3. Spot: spherical point light, has cutOffAngle, attenuation by deviation from pointing direction
4. Extended: line or area of light, all points point light sources, realistic, hard computation, has shadow: umbra and penumbra
5. Environment: (ass.) infinity away, trace reflected aray at each point, looks like texture mapping

9.1 Phong Illummination Model

$I = k_e + k_a I_a + \sum_i f(d_i) I_{li} (k_d (\mathbf{N} \cdot \mathbf{L_i})_+ + k_s (\mathbf{V} \cdot \mathbf{R}_+^{n_s}))$

1. k_e : intrinsic shade (self emission), one colour
2. $k_a I_a$: ambient reflection coeff, ambient intensity,
3. k_d: diffuse reflection coefficient, I_l intensity of light rouce, **N**: normal at surface point, **L**: direction to light source
4. Light intensity drops off,so we have $f(d) = min(1, \frac{1}{a+bd+cd^2})$
5. **R** : reflection angle, **V**: viewing angle.



	Ns	Kd	Ks
Metal	Large	Small, color of metal	Large, color of metal
Plastic	Medium	Medium, color of plastic	Medium, white
Planet	0	Varying	0

9.2 Interpolative shading

We want to recover the visual appeareance of a curved surface, we can approximate normals by averaging the normals of all polygons sharing that vertex. The shading can be obtained by a bilinear interpolation or barycentric-coordinate-based interpolation of the approriate quantitties from adjacent vertices

9.3 Gouraud Shading

- Calculate intensity at each vetex using a local reflection model
- Determine by bilinearly interpolating the vertices' intensities

Issues:

- Highlight anomalies: if highlight is in the interior, we may fail to shade it. Solution: smaller polygons
- Mach banding: we emphasize intesity changes at a boundary, which creates a banding effect. This can be obvious. Solution: smaller polygons

9.4 Phong Shading

Solves the interior highlight problem, since we interpolate the vertex normal vectors. This is done at **each** interior surface point.

Issue in interpolative shading: screen space is not same as world space.