Axis Aligned

$$R_z = \begin{bmatrix} \cos\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\theta & -s\theta \\ 0 & s\theta & c\theta \end{bmatrix}$$

$$R_y = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}$$

$$\hat{R} = \begin{bmatrix} A & 0 \\ & 0 \\ & 0 \\ 0\,0\,0 & 1 \end{bmatrix}$$

3D Transformations

- Translation $\hat{A} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Scale $\hat{A} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Shear $\hat{A} = \begin{bmatrix} 1 & h_{xy} & h_{xz} & 0 \\ h_{yx} & 1 & h_{yz} & 0 \\ h_{zx} & h_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Rotation - amount + axis of rot.
  - Still orthonormal.
  - Do not commute.
  - Euler angles
    - Gimbal lock → lose 1 dof.

      $R = R_z \cdot R_y \cdot R_z$
- Exponential maps.
  - Directly represent some arbitrary rotation

$$(\hat{r}x) = \begin{bmatrix} 0 & -\hat{r}_z & \hat{r}_y \\ \hat{r}_z & 0 & -\hat{r}_x \\ -\hat{r}_y & \hat{r}_x & 0 \end{bmatrix}$$

  - Singularities at 2π
  - No gimbal lock.

$$x' = e^{\hat{r}_x \theta} x$$

$$x' = (I + (\hat{r}x)\sin\theta + (\hat{r}x)^2(1-\cos\theta))\,x$$

- Quaternions
  - Über-complex numbers

$$q = (z_1, z_2, z_3, s) = (\underset{\sim}{z}, s)$$

$$q = iz_1 + jz_2 + kz_3 + s$$

$$i^2 = j^2 = k^2 = 1 \qquad ij = k \quad ji = -k \qquad jk = i \quad kj = -i \qquad ki = j \quad ik = -j$$

$$q \cdot p = (\underset{\sim}{z}_q s_p + \underset{\sim}{z}_p s_q + \underset{\sim}{z}_p \times \underset{\sim}{z}_q,\ s_p s_q - \underset{\sim}{z}_p \cdot \underset{\sim}{z}_q)$$

$$q^* = (-\underset{\sim}{z}, s)$$

$$\|q\| = \underset{\sim}{z} \cdot \underset{\sim}{z} + s^2 = q \cdot q^*$$

- Vectors $= (\underset{\sim}{v}, 0)$
- Rotation $= (\hat{r}\sin\theta/2, \cos\theta/2)$

$$x' = r \cdot x \cdot r^*$$

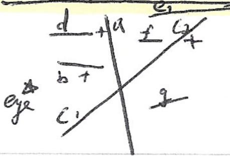
$$r = r_1 \cdot r_2 \leftarrow \text{compose rot.}$$

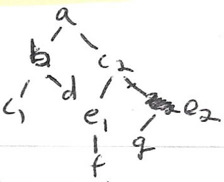
  - No tumbling
  - No gimbal lock
  - $\|r\| = 1$

Scene Hierarchy

- Group by objects big to small
- Draw scene w/ pre + post order traversal
  - Apply node, draw children, undo node if applicable.
- Node can really do anything
  - Geometry, transforms, groups, color, switch, etc
- Requires a stack to implement

Binary Space Partition Trees





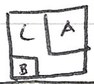
- Visibility traversal
  - Child one, root, child 2

Pre order $c_1, b, d, a, f, e_1, c_2, g, e_2$

Post order $e_2, g, c_2, e_1, f, a, d, b, c_1$

Axis-Aligned Bounding Box

- x, y, z box defined by min/max x, y, z
- Recompute when transforming



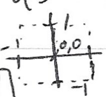
- Can bound multiple boxes.

Perspective

- Viewport (window) - sub-region of screen
- Screen space not always a screen!
  - Img file
  - Printer
  - etc

$$v = \frac{(j+0.5)}{n_y} \qquad u = \frac{(i+0.5)}{n_x}$$

- Screen coordinates
- Canonical view region

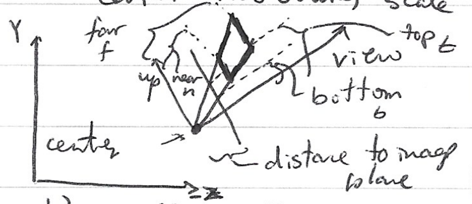


- To make Canonical
- Arbitrary window is then scaled/transformed

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\ 0 & -\frac{n_y}{2} & \frac{n_y-1}{2} \\ 0 & 0 & 0 \end{bmatrix}$$

remove for upside up.

- Projection
  - 3D → 2D → orthographic, Perspective } Linear
  - Linear → planar surface
    - Converge at ∞ or a point
- Canonical view region 3D: [-1, -1, -1] to [1, 1, 1]
  - +Z towards you
  - Translate center to origin, rotate up to +Y, view to -Z, center view volume, scale



1. Translate center to 0, 0
2. Rotate view to -Z, up to +Y
3. Shear center line to -Z
4. Perspective mat.

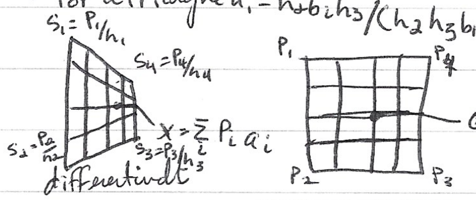
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & f \\ 0 & 0 & \frac{-1}{n} & 0 \end{bmatrix}$$

5. center view
6. scale to canonical

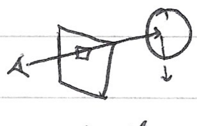
$\left.\begin{matrix}1\\2\end{matrix}\right\} M_v \quad \left.\begin{matrix}3\\4\end{matrix}\right\} M_p \quad \left.\begin{matrix}5\\6\end{matrix}\right\} M_o$

$$M = M_o \cdot M_p \cdot M_v$$

- Vanishing point $\lim_{t\to\infty} = \begin{bmatrix} dx \\ dy \end{bmatrix}$
- Ray Picking - pick object by picking object on screen.
  - ray from pixel coordinates
- Depth distortion
  - For a line $a_1 = h_2 b_1 / (b_1 h_2 + h_1 b_2)$
  - For a triangle $a_1 = h_2 b_1 h_3 / (h_2 h_3 b_1 + h_1 h_3 b_2 + h_1 h_2 b_3)$



Raytracing



- 3D → image
- Geometric reasoning about light rays.
- Build a ray, figure out what it hits, compute shading.
- Image plane built from 4 corner points
- Ray Equation

$$R(t) = E + t(P - E) \quad t \in [1, \infty)$$

through Eye @ t=0, at Pixel at t=1

- Shadow rays.

$$R(t) = S + t(L - S) \quad t \in [\varepsilon, \infty)$$

surface, Light dir, prevents self-shadowing

  - No occluder → Phong
  - Yes occluder → only ambient.

Implicit eq. for sphere $|x - c|^2 - r^2 = 0$

Sphere + Ray ⇒ $|A + tD - c|^2 - r^2$

↳ $t^2(D \cdot D) + 2t(D \cdot (A - C)) + (A - C)(A - C) - r^2 = 0$

- Reflection Rays $R(t) = S + tB,\ t \in [\varepsilon, \infty)$
  - Bounce off object
  - recursive bouncing } truncate at fixed # of bounces
  - Add to original point.
- Refracted Rays $k_t(\theta_i) = k_0 + (1 - k_0)(1 - \cos\theta_i)^5$, $k_0 = \left(\frac{n_t - 1}{n_t + 1}\right)^2$
- Anti-aliasing
  - Distributed - want integral over each pixel
    - Fire lots of points (rays)
      - send out many extra rays
      - Quasi random direction
      - multiple rays for refraction and reflect
- Ray vs Sphere

$$|R(t) - c|^2 - r^2 = 0 \Rightarrow |A + tD - c|^2 - r^2 = 0$$

- Ray vs. Triangle

$$V_1 + \beta(V_2 - V_1) + \gamma(V_3 - V_1) = A + tD$$

3 eqs. 3 unknowns. Beware of /0! divide by zero