Review: Vector, Projection, Cross Product, A*

Monday, January 9, 2023 2:03 PM

Vectors

vector $\vec{v} = \langle v_1, v_2, \dots v_n \rangle$

$$\vec{a} + \vec{b} = \langle a_1 + b_1,\ a_2 + b_2,\ \dots,\ a_n + b_n \rangle$$

$$\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + \dots + a_n b_n = |\vec{a}||\vec{b}| \cos(\theta)$$

Note:

$$\vec{a} + \vec{b} = \vec{b} + \vec{a}$$

$$\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}$$

$$\vec{a} \cdot (\vec{b} + \vec{c}) = \vec{a} \cdot \vec{b} + \vec{a} \cdot \vec{c}$$

$$k\vec{a} \cdot \vec{b} = \vec{a} \cdot k\vec{b} = k(\vec{a} \cdot \vec{b})$$

Projection

$$\vec{b} \to \vec{a} = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|^2} \vec{a}$$

Cross Product

$$\vec{a} \times \vec{b} = |\vec{a}||\vec{b}| \sin(\theta) = \begin{vmatrix} x & y & z \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} = \begin{bmatrix} a_y b_z - b_y a_z \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}}_{A^*} \cdot \vec{b}$$

Note:

$$\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$$

$$\vec{x} \times \vec{y} = \vec{z}, \quad \vec{y} \times \vec{z} = \vec{x}, \quad \vec{z} \times \vec{x} = \vec{y}$$

$$\vec{a} \times \vec{a} = 0$$

$$\vec{a} \times (\vec{b} + \vec{c}) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c}$$

$$\vec{a} \times (k\vec{b}) = k(\vec{a} \times \vec{b})$$

Review: Coordinate Frame

Wednesday, January 11, 2023 12:16 AM

Idea Each object may have its own coordinate frame

Def A coordinate frame is a set of 3 vectors $\vec{u}, \vec{v}, \vec{w}$ such:

$$|\vec{u}| = |\vec{v}| = |\vec{w}| = 1$$

$$\vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{w} = \vec{w} \cdot \vec{u} = 0$$

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

Any vector $\vec{p}$ can be writen $\vec{p} = (\vec{p} \cdot \vec{u})\vec{u} + (\vec{p} \cdot \vec{v})\vec{v} + (\vec{p} \cdot \vec{w})\vec{w}$

Note Given two vectors $\vec{a}, \vec{b}$, we can define

$$\vec{w} = \frac{\vec{a}}{|\vec{a}|} \quad u = \frac{\vec{b} \times \vec{w}}{|\vec{b} \times \vec{w}|} \quad \vec{v} = \vec{w} \times \vec{u}$$

Review: Matrix, Transpose, Identity, Inverse

Wednesday, January 11, 2023 12:28 AM

<u>Def</u> A matrix $A_{m \times n}$ has m rows, n cols

$A_{m\times n} + B_{m\times n}$, $k \cdot A_{m\times n}$ are performed elementwise

$$A_{m\times n} B_{n\times k} = C_{m\times k} \text{ where } C_{i,j} = A_{\text{row } i} \cdot B_{\text{col } j}$$

Note: $A(B+C) = AB + AC$

$(A+B)C = AC + BC$

<u>Idea</u> We can describe transformations as Matrix • vector operations

<u>Def</u> A^T maps row $\longleftrightarrow$ col

Note: $(AB)^T = B^T A^T$

<u>Def</u> I is the <u>identity matrix</u>:

$A \cdot A^{-1} = I$

$(AB)^{-1} = B^{-1} A^{-1}$

2D Transformations: Scale, Shear, Rotate

Wednesday, January 11, 2023 2:23 PM

Scaling: $\text{Scale}(S_x, S_y) = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$ $\quad \text{Scale}^{-1} = \begin{bmatrix} \frac{1}{S_x} & 0 \\ 0 & \frac{1}{S_y} \end{bmatrix}$

Shear: $\text{Shear}(a) = \begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix}$ where a,b correspond to horizontal, vertical shears.

Rotation: $\text{Rotate}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$ $\quad \text{Rotate}^{-1} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$

Rotations

Wednesday, January 11, 2023 2:37 PM

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R^{-1} = R^T$$

$$R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

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

<u>In General</u>:

for some rotation along an axis a:

$$R_a = I_{3\times3}\cos(\theta) + aa^T(1-\cos\theta) + A^*\sin\theta$$

gluLookAt -: Given eye, center, up:

$$a = eye - center \qquad b = up$$

$$w = \frac{a}{\|a\|} \qquad u = \frac{b\times w}{\|b\times w\|} \qquad v = w\times u$$

gluLookAt

$$\begin{bmatrix} x_u & y_u & z_u & -x_u e_x - y_u e_y - z_u e_z \\ x_v & y_v & z_v & -x_v e_x - y_v e_y - z_v e_z \\ x_w & y_w & z_w & -x_w e_x - y_w e_y - z_w e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Homogenous Basis, Translation

Wednesday, January 11, 2023 8:34 PM

Idea: Add a 4^{th} Homogenous coordinate $(w=1)$ to facilitate translation, viewing, and rotation.

Ex:
$$\begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x+5 \\ y \\ z \\ w \end{bmatrix}$$

Def Given a 4-vector $= \langle x, y, z, w \rangle$ then the corresponding 3-vector $\langle \frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1 \rangle$ represents the vector in 3D space.

Def The translation matrix $T = \begin{bmatrix} I_3 & T \\ 0 & 1 \end{bmatrix}$

$$T^{-1} = \begin{bmatrix} I_3 & -T \\ 0 & 1 \end{bmatrix}$$

Combining Translation and Rotation, Transforming Normals

Wednesday, January 11, 2023 8:54 PM

In general: perform rotation then translation

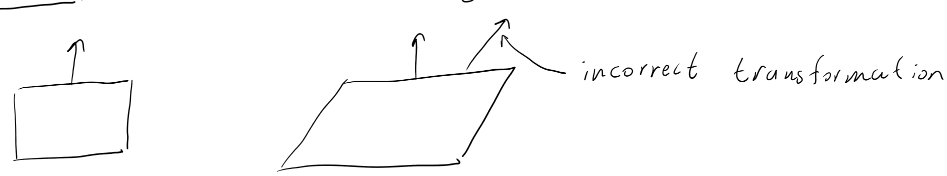
Def: performing a rotation then translation can be written:

$$P' = (TR)P = \begin{bmatrix} I_3 & T \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} R_{3\times3} & 0 \\ 0 & 1 \end{bmatrix} \times P = \begin{bmatrix} R_{3\times3} & T \\ 0 & 1 \end{bmatrix} \times P$$

Def: Performing a translation then rotation can be written:

$$P' = (RT)P = \begin{bmatrix} R_{3\times3} & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} I_3 & T \\ 0 & 1 \end{bmatrix} \times P = \begin{bmatrix} R_{3\times3} & R_{3\times3}T \\ 0 & 1 \end{bmatrix}$$

However, surface normals may not be transformed like usual:



Def Given a normal $\vec{n}$ on a surface with points $\vec{t}$ then:

$t \rightarrow Mt$ and $n \rightarrow Qn$ where $Q = (M^{-1})^T$

note that $(M^{-1})^T$ for homogenous matrix applies only to Upper 3×3

aka there is no effect to translations

Perspective Transformation

Wednesday, January 11, 2023 10:09 PM

<u>Def</u> <u>Orthographic</u> Drop one coordinate

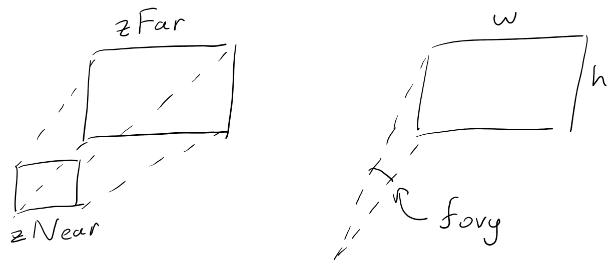
$(x, y, z) \to (x, y)$

$$M = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad glOrtho = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \boxed{\frac{-2}{f-n}} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

aligns some cube to the unit cube at the origin.
Given cube = (left, right, top, bottom, far, near)

<u>Def</u> <u>Perspective</u> Given the values fovy, aspect, zNear = n, zFar = f :

the viewing Frustrum:



the Transformation Matrix is:

$$\begin{bmatrix} \frac{d}{aspect} & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

where $A = -\frac{f+n}{f-n}$, $B = -\frac{2fn}{f-n}$, $d = \cot\left(\frac{fovy}{2}\right)$, $aspect = \frac{w}{h}$

OpenGL Reference

Thursday, January 26, 2023 9:32 AM

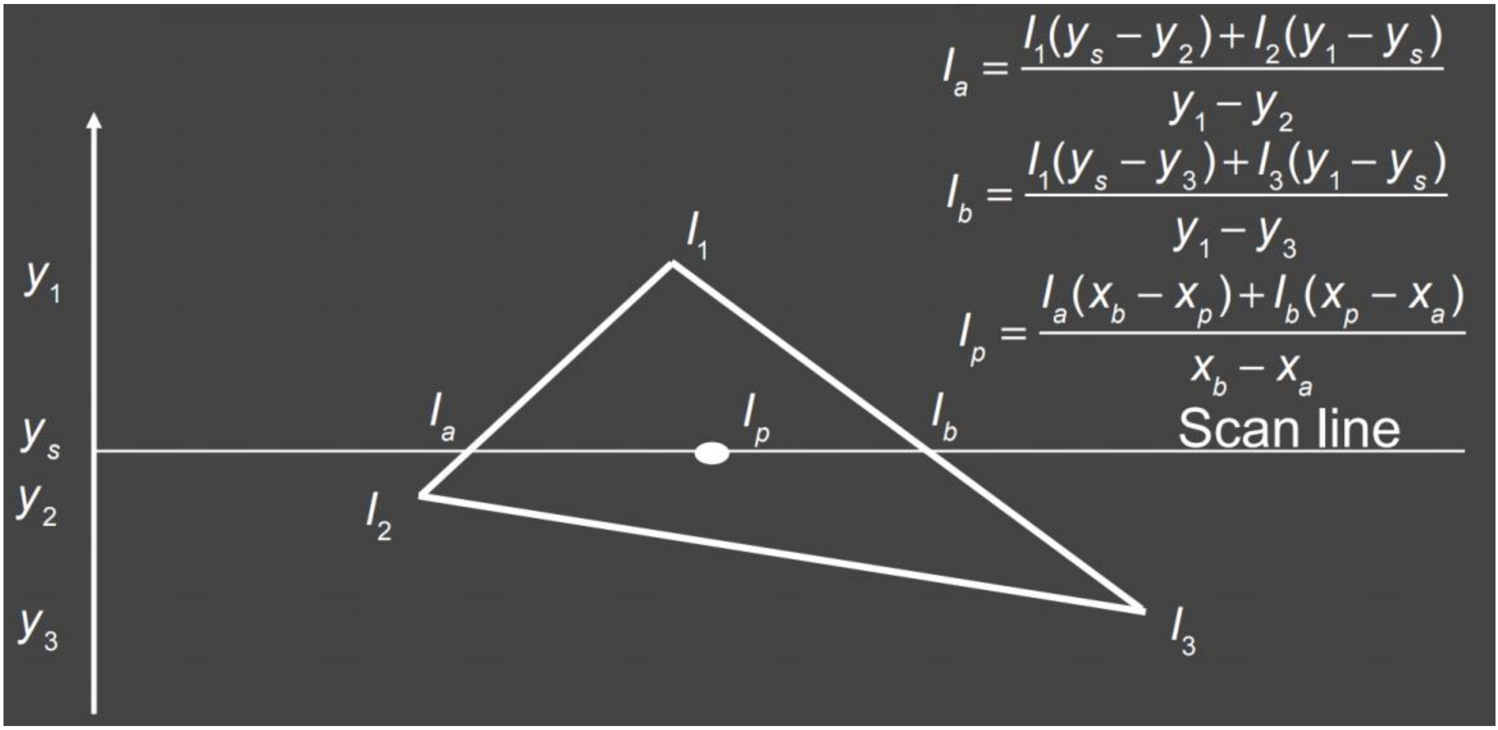
Buffers:
- Front - currently shown frame
- Back - next buffer to be shown, modify this
- Z - z values of objects, determines which object is drawn over others

Viewing:
- modelview: model object transformation and view point transformation
- projection: perspective projection transformation matrix

Transformations:
- Use a stack to store transformations: stack <mat4> modelview; modelview.push(mat4(1.0));

Gouraud Shading

Tuesday, January 31, 2023 9:52 AM



Phong Shading & Light Sources

Tuesday, January 31, 2023 10:13 AM

Types of light sources: point and directional

Point

- Position, Color
- Attenuation (quadratic model) $atten = \dfrac{1}{k_c + k_l d + k_q d^2}$

Attenuation

- Usually assume no attenuation (not physically correct)
- Quadratic inverse square falloff for point sources
- Linear falloff for line sources (tube lights). Why?
- No falloff for distant (directional) sources. Why?

Directional (w=0, infinite far away, no attenuation)

Material shading properties:

- Emissive: light emitted by the object, useful for light sources
- Ambient: ambient color of area, useful for simulating multiple bounces of light around a room
- Diffuse: reflection off a rough matte surface
- Specular: reflection off a smooth surface

Blinn-Phong formula: where N is the surface normal and H is the surface half angle
H = |Light + Eye|

$$I = \sum_{i=0}^{n} intensity_{light\ i} * specular_{material} * atten_i * [\max(N \bullet H, 0)]^{shininess}$$

Texturing

Thursday, February 2, 2023 10:47 AM

Idea: map vertices to an index in a texture map

Curves

Monday, February 13, 2023 9:59 PM

Bezier Curves

Tuesday, February 7, 2023 9:33 AM

Problem: How to define the geometry of objects?

Bezier Curve: Interpolates, tangent to end points

Simple example: given 2 control points P_0, P_1:
$F(x) = (1-x)P_0 + (x)P_1$

Recursive deCasteljau algorithm: Given an arbitrary n control points $P_0 \ldots P_{n-1}$
For each adjacent points P_k, P_{k+1} define
$P'_k = (1-x)P_k + (x)P_{k+1}$
Recurse on P' until a single point remains

Explicit Bernstein-Bezier algorithm:
$F(x) = \sum_{k=0}^{n-1} P_k * nCk * (1-u)^{n-k} * (u)^k$

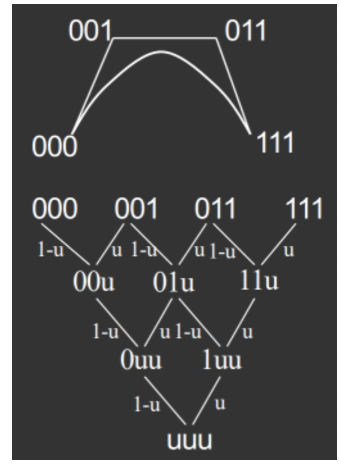
Matrix algorithm for cubics: Given 4 control points $P_0 \ldots P_3$

$$F(x) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} * \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

Polar form labeling (blossoms)

Monday, February 13, 2023 9:59 PM

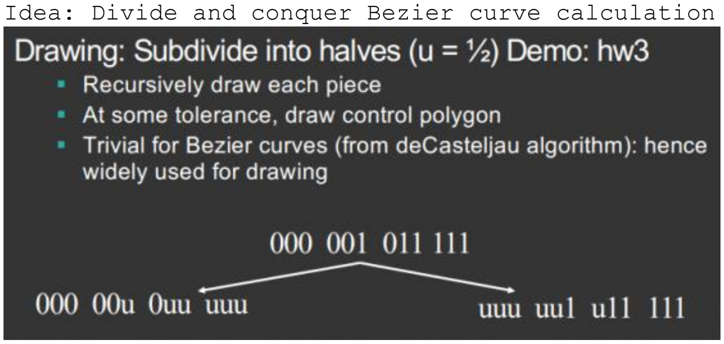
Idea: Labeling trick for control points and intermediate deCasteljau points



Ex: Bezier curve F(x)

- Define auxiliary function $f(x_1,x_2)$ [number of args - degree]
- Points on curve simply have $x_1=x_2$ so that F(x) = f(x,x)
- We can label control points by changing 1 argument at a time
  - Ie (000) -> (001) -> (011) -> (111)
- Only interpolate linearly between points with one arg different
  - f(0,u) = (1-u) f(0,0) + u f(0,1) - interpolate f(0,0) and f(0,1)=f(1,0)

Idea: Divide and conquer Bezier curve calculation



B-spline Curves

Monday, February 13, 2023 10:00 PM

Knot vector for quadratic: -1 0 1 2
Knot vector for cubic: -2 -1 0 1 2 3

Gamma, Color

Tuesday, February 21, 2023 10:45 AM

Problem: intensity of display is not linear to value

$I = a^\gamma$ where I is display intensity, a is pixel value

Def: Gamma correction

$a' = a^{(1/\gamma)}$

Adjusts for nonlinear intensity by increasing the pixel values to compensate

Overlaying Colors

Tuesday, February 21, 2023 10:42 AM

Given C and α for A and B where A overlays B:

$\alpha = \alpha A + (1-\alpha A)\alpha B$
$C = \alpha A*CA + (1-\alpha A)*\alpha B*CB$