Mathematics for 3D Graphics

Topics

Points, Vectors, Vertices, Coordinates

Dot Products, Cross Products

Lines, Planes, Intercepts

References

Many texts cover the linear algebra used for 3D graphics . . .
. . . the texts below are good references, Akenine-Möller is more relevant to the class.

J. Ström, K. Åström, and T. Akenine-Möller, "immersive linear algebra," v.0.73, `http://immersivemath.com/ila/index.html`.

Appendix A and Chapter 4 in T. Akenine-Möller, E. Haines, N. Hoffman, "Real-Time Rendering," Third Edition, A. K. Peters Ltd.

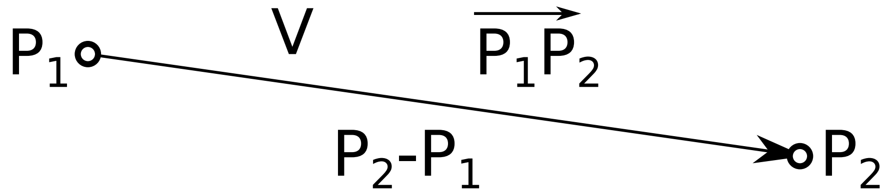
Appendix A in Foley, van Dam, Feiner, Huges, "Computer Graphics: Principles and Practice," Second Edition, Addison Wesley.

Points and Vectors

math-2

Point:

Indivisible location in space.



$$E.g.,\ P_1 = \begin{bmatrix}1\\2\\3\end{bmatrix},\ P_2 = \begin{bmatrix}4\\5\\6\end{bmatrix}$$

Vector:

Difference between two points.

$$E.g.,\ V = P_2 - P_1 = \overrightarrow{P_1P_2} = \begin{bmatrix}4-1\\5-2\\6-3\end{bmatrix} = \begin{bmatrix}3\\3\\3\end{bmatrix}.$$

Equivalently: $P_2 = P_1 + V$.

Don't confuse points and vectors!

Point-Related Terminology

Will define several terms related to points.

At times they may be used interchangeably.

Point:
A location in space.

Coordinate:
A representation of location.

Vertex:
Term may mean point, coordinate, or part of graphical object.

As used in class, vertex is a less formal term.

It might refer to a point, its coordinate, and other info like color.

Coordinate:
A representation of where a point is located.

Familiar representations:

3D Cartesian $P = (x, y, z)$.

2D Polar $P = (r, \theta)$.

In class we will use 3D *homogeneous coordinates*.

Homogeneous Coordinates

math-5

Homogeneous Coordinate:

A coordinate representation for points in 3D space consisting of four components...

... each component is a real number...

... and the last component is non-zero.

Representation: $P = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$, where $w \neq 0$.

P refers to same point as Cartesian coordinate $(x/w, y/w, z/w)$.

To save paper sometimes written as (x, y, z, w).

Homogeneous Coordinates

math-6

Each point can be described by many homogeneous coordinates . . .

. . . for example, $(10, 20, 30) = \begin{bmatrix} 10 \\ 20 \\ 30 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 10 \\ 15 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 20 \\ 40 \\ 60 \\ 2 \end{bmatrix} = \begin{bmatrix} 10w \\ 20w \\ 30w \\ w \end{bmatrix} = \ldots$

. . . these are all equivalent so long as $w \neq 0$.

Column matrix $\begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$ could not be a homogeneous coordinate . . .

. . . but it could be a vector.

Homogeneous Coordinates

Why not just Cartesian coordinates like (x, y, z)?

The w simplifies certain computations.

Confused?

Then for a little while pretend that $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ is just (x, y, z).

Homogenized Homogeneous Coordinate

A homogeneous coordinate is *homogenized* by dividing each element by the last.

For example, the homogenization of $\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$ is $\begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix}$

Homogenization is also known as *perspective divide*.

Vector Arithmetic

Points just sit there, it's vectors that do all the work.

In other words, most operations defined on vectors.

Point/Vector Sum

The result of adding a point to a vector is a point.

Consider point with homogenized coordinate $P = (x, y, z, 1)$ and vector $V = (i, j, k)$.

The sum $P + V$ is the point with coordinate $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} + \begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} x+i \\ y+j \\ z+k \\ 1 \end{bmatrix}$

This follows directly from the vector definition.

Scalar/Vector Multiplication

The result of multiplying scalar a with a vector is a vector...
... that is a times longer but points in the same or opposite direction...
... if $a \neq 0$.

Let a denote a scalar real number and V a vector.

The *scalar vector product* is $$aV = a\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az \end{bmatrix}.$$

Vector/Vector Addition

The result of adding two vectors is another vector.

Let $V_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$ and $V_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$ denote two vectors.

The *vector sum*, denoted $U + V$, is $\begin{bmatrix} x_1 + x_2 \\ y_1 + y_2 \\ z_1 + z_2 \end{bmatrix}$

Vector subtraction could be defined similarly. . .

. . . but doesn't need to be because we can use scalar/vector multiplication: $V_1 - V_2 = V_1 + (-1 \times V_2)$.

Vector Addition Properties

Vector addition is associative:

$$U + (V + W) = (U + V) + W.$$

Vector addition is commutative:

$$U + V = V + U.$$

Vector Magnitude, Normalization

math-13

Vector Magnitude

The magnitude of a vector is its length, a scalar.

The *magnitude* of $V = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ denoted $\|V\|$, is $\sqrt{x^2 + y^2 + z^2}$.

The magnitude is also called the *length* and the *norm*.

Vector V is called a *unit vector* if $\|V\| = 1$.

A vector is *normalized* by dividing each of its components by its length.

The notation $\hat{V}$ indicates $V/\|V\|$, the normalized version of V.

The Vector Dot Product

The dot product of two vectors is a scalar.

Roughly, it indicates how much they point in the same direction.

Consider vectors $V_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$ and $V_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$.

The *dot product* of V_1 and V_2, denoted $V_1 \cdot V_2$, is $x_1 x_2 + y_1 y_2 + z_1 z_2$.

What a Dot Product Does

Let

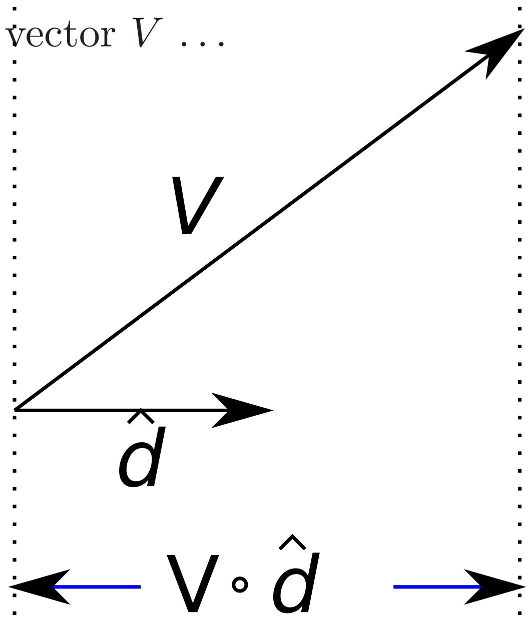
V be some arbitrary vector and

$\hat{d}$ be a unit vector.

Then $V \cdot \hat{d}$...

... measures the length of the vector V ...

... in the direction of $\hat{d}$.



EE 4702-1 Lecture Transparency. Formatted 17:25, 1 September 2017 from set-1-math.

Dot Product Properties

math-16

Let U, V, and W be vectors.

Let a be a scalar.

Miscellaneous Dot Product Properties

$(U + V) \cdot W = U \cdot W + V \cdot W$

$(aU) \cdot V = a(U \cdot V)$

$U \cdot V = V \cdot U$

$\text{abs}(U \cdot U) = \|U\|^2$

Dot Product Properties

math-17

Orthogonality

The more casual term is perpendicular.

Vectors U and V are called *orthogonal* iff $U \cdot V = 0$.

This is an important property for finding intercepts.

math-17

Dot Product Properties

math-18

Angle

Let U and V be two vectors.

Then $U \cdot V = \|U\|\|V\| \cos \phi$...
... where ϕ is the smallest angle between the two vectors.

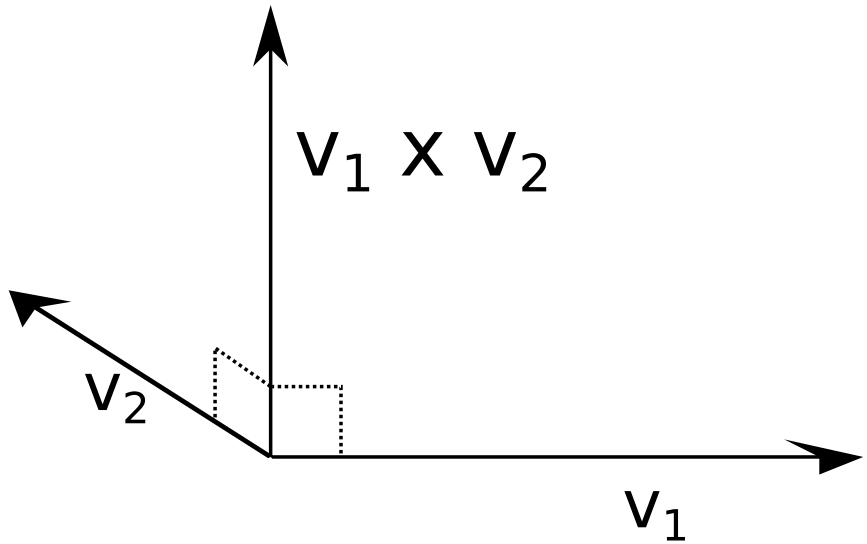
math-18

Cross Product

The cross product of two vectors results in a vector orthogonal to both.

The *cross product* of vectors V_1 and V_2, denoted $V_1 \times V_2$, is

$$V_1 \times V_2 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \times \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} y_1 z_2 - z_1 y_2 \\ z_1 x_2 - x_1 z_2 \\ x_1 y_2 - y_1 x_2 \end{bmatrix}.$$



Cross Product Properties

math-20

Cross Product Properties

Let U and V be two vectors and let $W = U \times V$.

Then both U and V are orthogonal to W.

$\|U \times V\| = \|U\|\|V\| \sin\phi$.

$U \times V = -V \times U$.

$(aU + bV) \times W = a(U \times W) + b(V \times W)$.

$U \times (V \times W) = (U \cdot W)V - (U \cdot V)W$.

If U and V define a parallelogram, its area is $\|U \times V\|$...
... if they define a triangle its area is $\frac{1}{2}\|U \times V\|$.

Line Definition

A line will be defined in terms of a point and a non-zero vector.

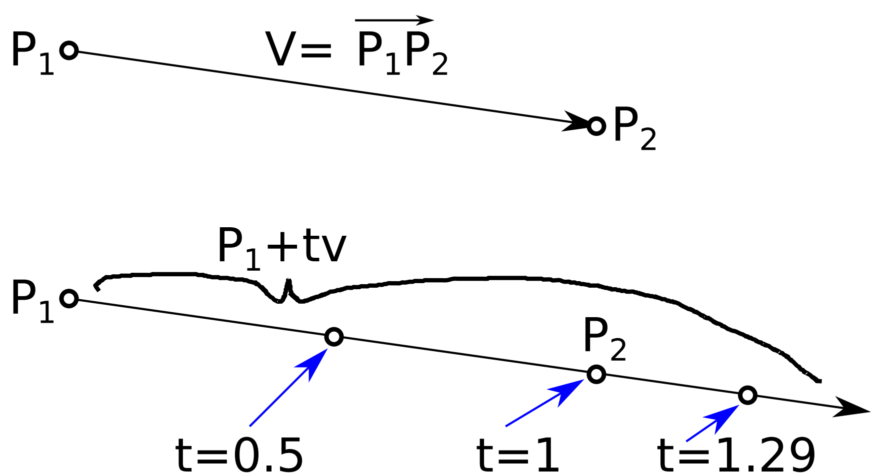
Line:

A set of points generated from a given point, P_1, and vector, v: $\{S \mid P_1 + tv,\ \forall t \in \Re\}$.

Parametric Description of Line

$$P(t) = P_1 + tv.$$

Illustration of defining a line in terms of two points:



Plane Definition

Point P and vector $\overrightarrow{n}$ define a *plane* in which a point S is on the plane iff $\overrightarrow{PS} \cdot \overrightarrow{n} = 0$.

The vector $\overrightarrow{n}$ if referred to as a *normal*.

Plane/Line Intercept

Problem: Given line $L + t\overrightarrow{v}$ and a plain defined by point P and vector $\overrightarrow{n}$, find a point, S, that is both on the line and on the plane.

Since S is on the line, $S = L + t\overrightarrow{v}$.

Since S is on the plane, $\overrightarrow{SP} \cdot \overrightarrow{n} = 0$

Find a t for which both are true by substituting for S and solving for t:

$$\begin{aligned}
\overrightarrow{(L + t\overrightarrow{v})P} \cdot \overrightarrow{n} &= 0 \\
(P - L - t\overrightarrow{v}) \cdot \overrightarrow{n} &= 0 \\
(\overrightarrow{LP} - t\overrightarrow{v}) \cdot \overrightarrow{n} &= 0 \\
t &= \frac{\overrightarrow{LP} \cdot \overrightarrow{n}}{\overrightarrow{v} \cdot \overrightarrow{n}}
\end{aligned}$$

Use this expression for t to find S

$$S = L + \frac{\overrightarrow{LP} \cdot \overrightarrow{n}}{\overrightarrow{v} \cdot \overrightarrow{n}} \overrightarrow{v}$$

Sample Problem

math-24

Problem: *A light model specifies that in a scene with a light of brightness* b *(scalar) at location* L *(coordinate), and a point* P *on a surface with normal* $\hat{n}$*, the* lighted color*,* c*, of* P *(a scalar) will be the dot product of the surface normal with the direction to the light divided by the distance to the light.*

Restate this as a formula.

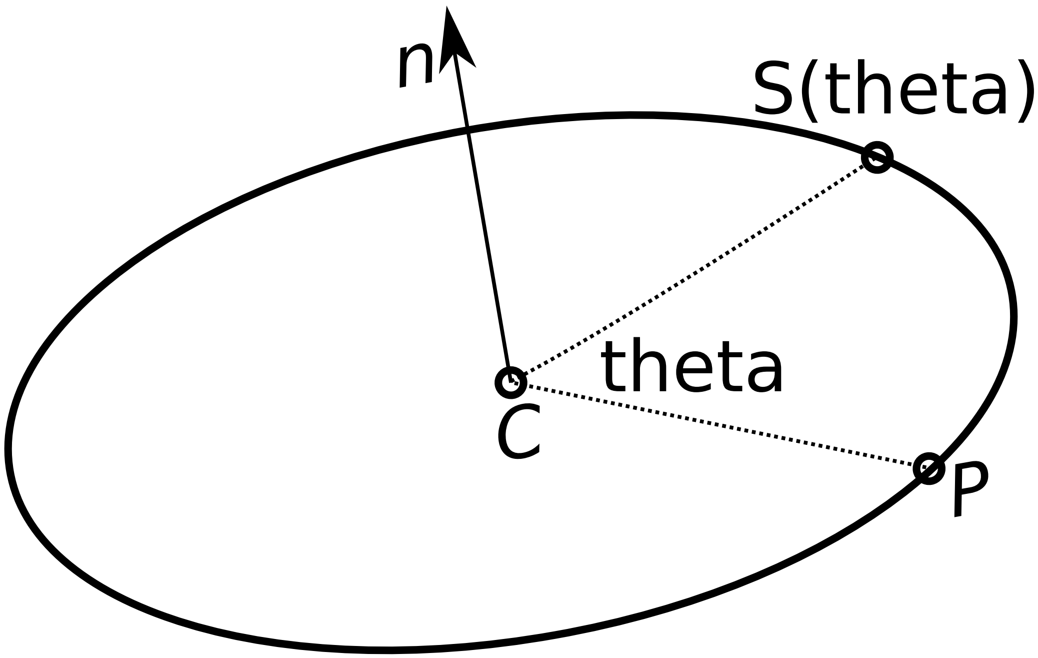
Estimate the number of floating point operations in a streamlined computation.

Solution:

Formula: $c = b\widehat{PL} \cdot \hat{n} \frac{1}{\|\overrightarrow{PL}\|}$.

math-24

Problem: *Find a parametric description $S(\theta)$ of a circle that passes through point P, with its center at C, and facing direction* $\hat{n}$.*




```
for ( float theta = 0; theta < 2 * M_PI; theta += delta_theta )
  {
    pCoor pos = S(theta);   // Need to find S(theta).
    // Do something with pos..
  }
```

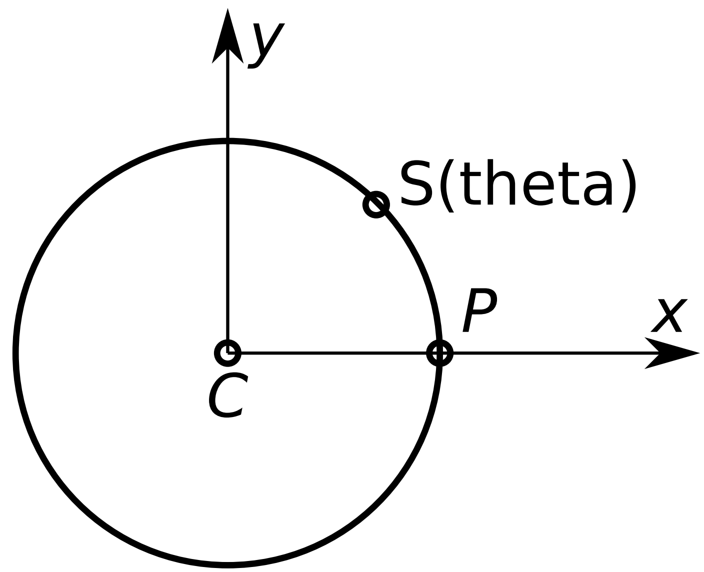
* The quantity $\hat{n}$ is not necessarily orthogonal to $\overrightarrow{CP}$.

Sample problem, continued.

First, lets solve the easy version of the problem: 2D, circle at origin.

To make it easy:

$$C = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, P = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix} \text{ and } \hat{n} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



Parametric formula:

$$S(\theta) = \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ 0 \end{bmatrix}$$

Use of parametric formula in code:

```
for ( float theta = 0; theta < 2 * M_PI; theta += delta_theta )
  {
    pCoor point_S( r * cos(theta), r * sin(theta), 0 );
    // Do something with point_S..
  }
```

math-27

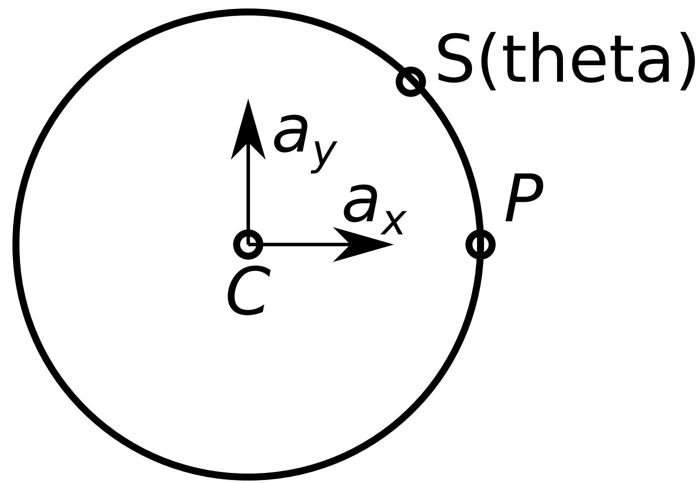
Re-write formula as C plus two vectors:

$$S(\theta) = C + \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ 0 \end{bmatrix}$$

$$= C + r\cos\theta \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + r\sin\theta \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$= C + r\cos(\theta)\,\hat{a}_x + r\sin(\theta)\,\hat{a}_y,$$

where $\hat{a}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ and $\hat{a}_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$.



math-27

The converted formula again:

$$S(\theta) = C + r\cos(\theta)\,\hat{a}_x + r\sin(\theta)\,\hat{a}_y$$

Suppose instead $\hat{a}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ and $\hat{a}_y = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Then circle would be on xz plain instead of the xy plain.

We know that $\hat{a}_y$ points along the z axis, ...
... but the parametric formula thinks its the y axis.

Key Observation:

A circle can be drawn in any orientation by choosing $\hat{a}_x$ and $\hat{a}_y$ appropriately.

The original problem: *Find a parametric description $S(\theta)$ of a circle that passes through point P, with its center at C, and facing in direction $\hat{n}$.*

The formula:

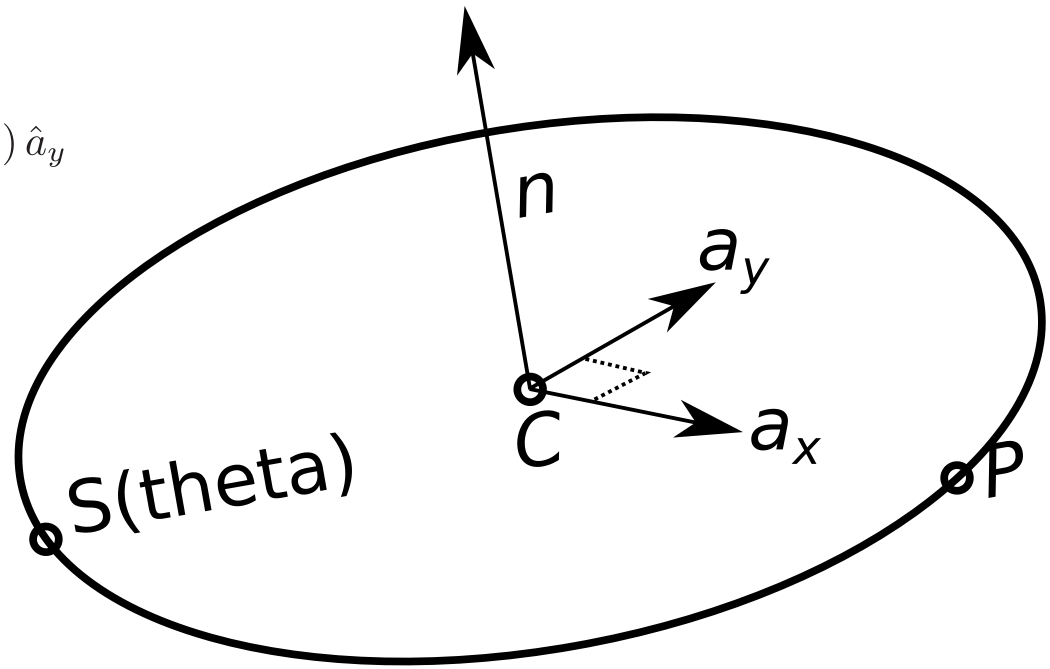
$$S(\theta) = C + r\cos(\theta)\,\hat{a}_x + r\sin(\theta)\,\hat{a}_y$$

Need to find $\hat{a}_x$, $\hat{a}_y$, and r:

Clearly, $r = \|\overrightarrow{CP}\|$

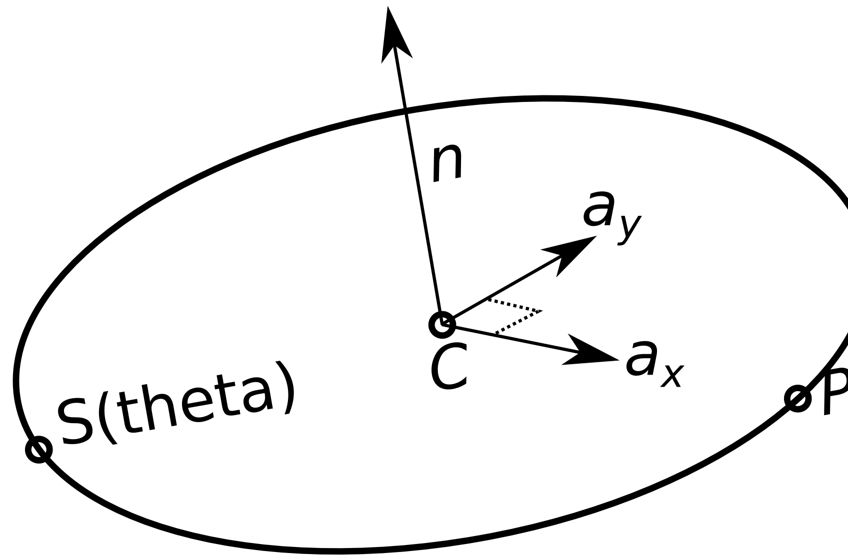
We can set $\hat{a}_x = \frac{1}{r}\overrightarrow{CP}$.

And then $\hat{a}_y = \hat{n} \times \hat{a}_x$.



Recall:

$$r = \|\overrightarrow{CP}\|, \quad \hat{a}_x = \frac{1}{r}\overrightarrow{CP}, \quad \hat{a}_y = \hat{a}_x \times \hat{n}.$$



Code for circle:

```
// Given:
pNorm n(1,2,3);
pCoor C(4,5,6);
pCoor P(7,8,9);

// Compute:
pNorm ax(C,P);            // ax is a unit vector from C to P.
pNorm ay = cross(n,ax);   // Normalize in case n is not orthogonal to CP.
float r = ax.magnitude;

// Construct points on circle:
for ( float theta = 0; theta < 2 * M_PI; theta += delta_theta )
  {
    pCoor pos = C + r * cos(theta) * ax + r * sin(theta) * ay;
    // Do something with pos..
  }
```

Vectors $\hat{a}_x$, $\hat{a}_y$, and $\hat{n}$ form an *orthonormal basis*.

Transforms

math-32

Transformation:

A mapping (conversion) from one coordinate set to another (*e.g.*, from feet to meters) or to a new location in an existing coordinate set.

Particular Transformations to be Covered

Translation: Moving things around.

Scale: Change size.

Rotation: Rotate around some axis.

Projection: Moving to a surface.

Transform by multiplying 4×4 matrix with coordinate.

$P_{\text{new}} = M_{\text{transform}} P_{\text{old}}$.

Scale Transform

$$S(s) = \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$S(s,t,u) = \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & t & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$S(s)$ stretches an object s times along each axis.

$S(s,t,u)$ stretches an object s times along the x-axis, t times along the y-axis, and u times along the z-axis.

Scaling centered on the origin.

Example of Scale Transform

Given:

$$S(5) = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad P = \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix}.$$

Compute Q, the result of transforming P by $S(5)$:

$$Q = S(5)P = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} 5a + 0b + 0c + 0 \times 1 \\ 0a + 5b + 0c + 0 \times 1 \\ 0a + 0b + 5c + 0 \times 1 \\ 0a + 0b + 0c + 1 \times 1 \end{bmatrix} = \begin{bmatrix} 5a \\ 5b \\ 5c \\ 1 \end{bmatrix}$$

Code:

```
pMatrix_Scale S(5);  // Construct the scale matrix.
pCoor P(a,b,c);      // Construct the coordinate.
pCoor Q = S * P;
```

Rotation Transformations

$R_x(\theta)$ rotates around x axis by θ; likewise for R_y and R_z.

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Translation Transform

$$T(s,t,u) = \begin{pmatrix} 1 & 0 & 0 & s \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & u \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Moves point s units along x axis, etc.

Example: Show arithmetic for $Q = T(s,t,u)P$ where $P = \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix}$

$$Q = T(s,t,u)P = \begin{pmatrix} 1 & 0 & 0 & s \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & u \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} 1a + 0b + 0c + s \times 1 \\ 0a + 1b + 0c + t \times 1 \\ 0a + 0b + 1c + u \times 1 \\ 0a + 0b + 0c + 1 \times 1 \end{bmatrix} = \begin{bmatrix} a + s \\ b + t \\ c + u \\ 1 \end{bmatrix}$$

Code:

```
pCoor P(a,b,c);
pMatrix_Translate T(s,t,u);
pCoor Q = T * P;
```

Computational Efficiency of Translation Transform

Using Transform:

$$Q = T(s,t,u)P.$$

16 multiplications, 12 additions.

Using Vector Addition:

$$Q = P + \begin{bmatrix} s \\ t \\ u \end{bmatrix}$$

0 multiplications, 3 additions.

Conclusion:

If *all* we want to do is translations, don't use matrix version $(T(s,t,u))$.

Matrix version makes sense if we want to combine transforms.

Composing Transforms

math-38

Often multiple transforms are applied to a point ...

... for example, a rotation, scale, and translation:

$$Q_a = R_x(\theta)\,P, \qquad Q_b = S(1.23)\,Q_a, \qquad Q = T(4,5,6)\,Q_b.$$

Total Computation: $3 \times 4^2 = 48$ multiplies.

Transformations can be combined:

First Compute $M = T(4,5,6)\,S(1.23)\,R_x(\theta)$. 2×4^3 multiplies.

$Q = MP$ 4^2 multiplies

Total Computation: $2 \times 4^3 + 4^2 = 144$ multiplies. Isn't that worse?

Often the same set of transforms applied to multiple points:

$Q_i = MP_i$ for $0 \leq i < n$. Suppose $n = 100$.

Computation using just M: $2 \times 4^3 + n4^2$. $2 \times 4^3 + 100 \times 4^2 = 1728$.

Computation using R, S, and T: $3n4^2$. $3 \times 100 \times 4^2 = 4800$.

math-38

Miscellaneous Matrix Multiplication Math

Let M and N denote arbitrary 4×4 matrices.

Identity Matrix

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$IM = MI = M.$$

Transforms and Matrix Arithmetic

math-40

Matrix Inverse

Matrix A is an inverse of M iff $AM = MA = I$.

Will use M^{-1} to denote inverse.

Not every matrix has an inverse.

Computing inverse of an arbitrary matrix expensive ...
... but inverse of some matrices are easy to compute ...
... for example, $T(x, y, z)^{-1} = T(-x, -y, -z)$.

Matrix Multiplication Rules

Is associative: $(LM)N = L(MN)$.

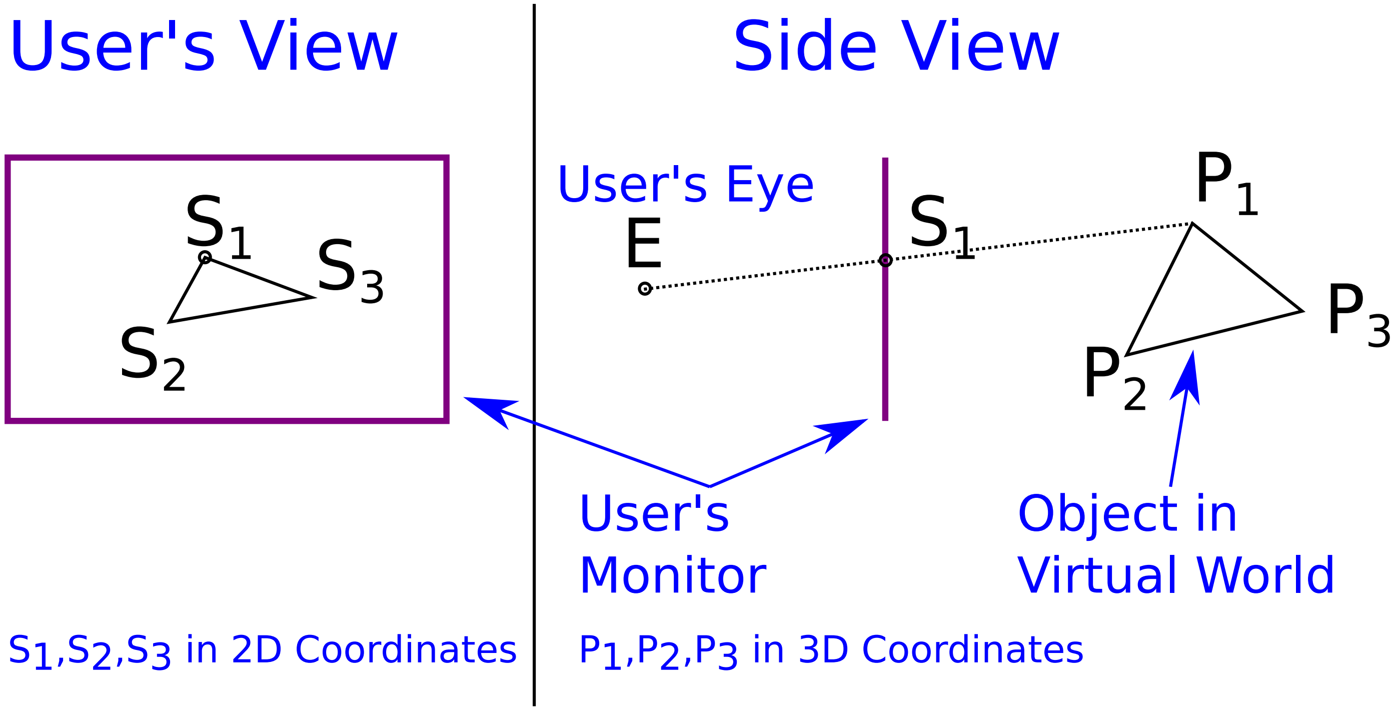
Is not commutative: $MN \neq NM$ for arbitrary M and N.

$(MN)^{-1} = N^{-1}M^{-1}$. (Note change in order.)

Projection Transform:

A transform that maps a coordinate to a space with fewer dimensions.

A projection transform maps a 3D coord. from our virtual world (such as P_1) . . .
. . . to a 2D location on our monitor (such as S_1).



$$S_1 = T_{\text{projection}} P_1$$

Projection Types

Vague definitions on this page.

Perspective Projection

Points appear to be in "correct" location,...
... as though monitor were just a window into the simulated world.

This projection used when realism is important.

Orthographic Projection

A projection without perspective foreshortening.

This projection used when a real ruler will be used to measure distances.

Perspective Projection Derivation

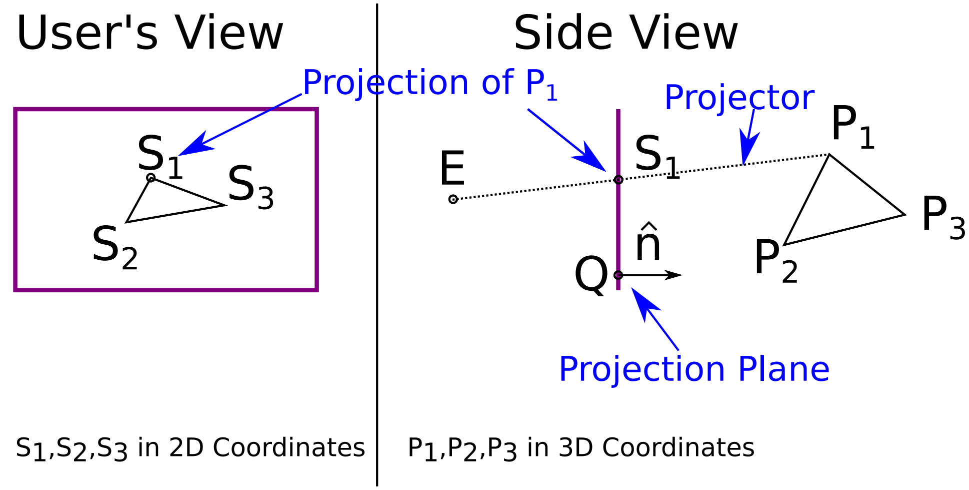
math-43

Lets put user and user's monitor in world coordinate space:

Location of user's eye: E.

A point on the user's monitor: Q.

Normal to user's monitor pointing away from user: $\hat{n}$.



Goal:

Find S_1, point where line from E to P_1 intercepts monitor (plane $Q, \hat{n}$).

Line from E to P called the *projector*.

The user's monitor is in the *projection plane*.

The point S is called the *projection* of point P on the projection plane.

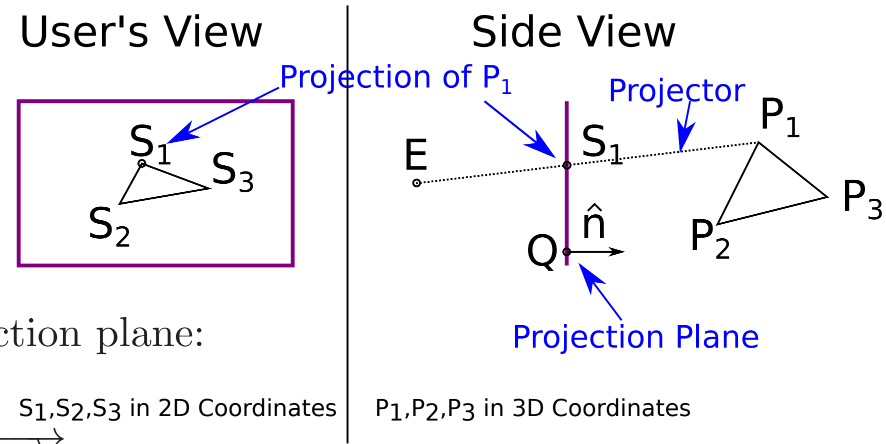
math-43

Solution:

Projector equation: $S = E + t\overrightarrow{EP}$.

Projection plane equation: $\overrightarrow{QS} \cdot n = 0$.

Find point S that's on projector and projection plane:



$$\overrightarrow{Q(E + t\overrightarrow{EP})} \cdot n = 0$$

$$(E + t\overrightarrow{EP} - Q) \cdot n = 0$$

$$\overrightarrow{QE} \cdot n + t\overrightarrow{EP} \cdot n = 0$$

$$t = \frac{\overrightarrow{EQ} \cdot n}{\overrightarrow{EP} \cdot n}$$

$$S = E + \frac{\overrightarrow{EQ} \cdot n}{\overrightarrow{EP} \cdot n}\overrightarrow{EP}$$

Note: $\overrightarrow{EQ} \cdot n$ is distance from user to plane in direction n ...

... and $\overrightarrow{EP} \cdot n$ is distance from user to point in direction n.

Perspective Projection Derivation

math-45

To simplify projection:

Fix $E = (0,0,0)$: Put user at origin.

Fix $n = (0,0,1)$: Make "monitor" parallel to xy plane.

Before: $$S = E + \frac{\overrightarrow{EQ} \cdot n}{\overrightarrow{EP} \cdot n} \overrightarrow{EP}$$

After: $$S = \frac{q_z}{p_z} P,$$

where q_z is the z component of Q, and p_z defined similarly.

The key operation in perspective projection is dividing out by z (given our geometry).

Simple Perspective Projection Transformation

math-46

Simple Projection Transform 1

Eye at origin, projection surface at (x, y, q_z), normal is $(0, 0, 1)$.

$$F_{q_z} = \begin{pmatrix} q_z & 0 & 0 & 0 \\ 0 & q_z & 0 & 0 \\ 0 & 0 & q_z & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Applying the projection to coordinate $(x, y, z, 1)$:

$$F_{q_z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} q_z x \\ q_z y \\ q_z z \\ z \end{bmatrix} = \begin{bmatrix} \frac{q_z}{z} x \\ \frac{q_z}{z} y \\ \frac{q_z z}{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{q_z}{z} x \\ \frac{q_z}{z} y \\ q_z \\ 1 \end{bmatrix}$$

This maps the z coordinate to the constant q_z ...

... *meaning that the position along the z axis has been lost.*

But we'll need the z position to determine visibility of overlapping objects.

math-46

Simple Perspective Projection Transformation

math-47

Simple Projection Transform, *Preserving z*

Eye at origin, projection surface at (x, y, q_z), normal is $(0, 0, 1)$.

$$F_{q_z} = \begin{pmatrix} q_z & 0 & 0 & 0 \\ 0 & q_z & 0 & 0 \\ 0 & 0 & 0 & q_z \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Applying the projection to coordinate $(x, y, z, 1)$:

$$F_{q_z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} q_z x \\ q_z y \\ q_z \\ z \end{bmatrix} = \begin{bmatrix} \frac{q_z}{z} x \\ \frac{q_z}{z} y \\ \frac{q_z}{z} \\ 1 \end{bmatrix}$$

This maps z coordinate to q_z/z, ...

... which though a reciprocal, will still be useful.

math-47

View Volume, Frustum

math-48

View-Volume Related Definitions

View Volume:

Parts of the scene which should be visible to the user.

Frustum:

A shape constructed by slicing off the top of a square-base pyramid with a plane parallel to the base.

EE 4702-1 Lecture Transparency. Formatted 17:25, 1 September 2017 from set-1-math.

math-48

Frustum View Volume Motivation

Consider the simple projection transformation:

Shape of view volume consists of two pyramids . . .
. . . one pyramid in front, the other in back, . . .
. . . and both points on eye.

Some points are behind the user. . .
. . . and we don't want these to be visible (because they would be unnatural).

Some points in view volume are so far from the user. . .
. . . that they would be invisible.

For example, points might form a triangle that covers 1% of a pixel.

These points waste computing power.

Frustum View Volume

View volume in shape of frustum with smaller square on projection plane.

The smaller square of frustum defines a *near plane*.

The larger square defines a *far plane*.

Variables describing a frustum view volume:

n: Distance from eye to near plane.

f: Distance from eye to far plane.

Coordinates of lower-left corner of $(l, b, -n)$.

Coordinates of upper-right corner of $(r, t, -n)$.

Frustum Perspective Transform

Given six values: l, r, t, b, n, f (left, right, top, bottom, near, far).

Eye at origin, projection surface at (x, y, n), normal is $(0, 0, -1)$.

Viewer screen is rectangle from $(l, b, -n)$ to $(r, t, -n)$.

Points with $z > -t$ and $z < -f$ are not of interest.

$$F_{l,r,t,b,n,f} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -2\frac{fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$