black art of manipulating numbers

TRIG

fund #1 - triangle angles add up to 180

fund #2 - right triangles are called "special"

sin th = opposite/hypotenuse

cos th = adjacent/hypotenuse

tan th = opposite/adjacent

assume circle with radius 1 (unit circle)

angle at center, then sin angle = y, cos angle = x

sin 90 = 1 = pi/2

cos 90 = 0

sin 0 = 0

cos 0 = 1

sin 180 = 0

cos 180 = -1

application: Projector throw

ratio of distance to image width

ex: T = 2:1 from 2 meters image width is 1

tan th/2 = 1/2t

from throw distance

th = 2*arctan(1/2*throw)

VECTORS

vector == offset

vector notation

-> -> ->

V = (10,5) = 10i + 5j = Vxi+Vyj+Vzk

magnitude = sqrt(x*x+y*y)

Vector arithmetic is component-wise addition is commutative

To visualize = draw vectors running tip to tail

Vector subtraction is not commutative

To visualize = put both tails at zero and draw vector between tips

Vector-scalar math

can't add

multiply is scaling, i.e. (10,5)x2 = (10,5)+(10,5)

Vector multiplication is mathematically undefined
Vector libraries perform component-wise multiplication

Vector multiplication is possible through dot and cross PRODUCT
DOT PRODUCT
Dot = (x,y) . (x0,y0) = mag(v0)*mag(v1)*cos theta (theta is angle between) = (x0x1)+(y0y1)+(z0z1)
dot product is "shadow" of a vector is a scalar

normalizing a vector = make magnitude 1
sqrt(vx*vx+vy*vy)/(vx*vx+vy*vy);

reflection
vector reflected off normal n
take tangent and normal
tangent

CROSS PRODUCT
cross = v0 X v1 = mag(v0)*mag(v1)*sin theta
dot product is shadow (scalar), cross product is vector of rotation, that is what is the vector you rotate v0 around to get v1

getting poly strips from a vector v0 defined by p1 and p2
need to get perpendicular vectors v1 and v2
p2-p1 gives us direction vector (p0)
we need an up vector up = (0,0,1)

p0 X up gives us p1
multiply p1.normalized * desired width(w)/2 = offset vector

p1+offset and p1-offset gives us v1
p2+offset and p2-offset gives us v2

optimization tip for finding contour direction
- put neighboring vectors tail to tail
- take dots
- check cross only if dot meets thresh

finding “hills” in 3d
- cross dot up

MATRICES and XFORMS

Matrices == denote frame of reference for xform
Matrix == xform applied to axes
Matrix*Vector number of columns in matrix must match number of elements in vector
Matrix*Vector = Matrix

Generally use 4x4

identity matrix:
[1 0 0] = x axis
[0 1 0] = y axis
[0 0 1] = z axis

using identity matrix
example camera look
camera o, target t
direction vector dv = (t-o).norm
world up vector wup = (0,0,1)
right vector rv = (d X wup).norm

object up vector ov = (d X r)

look matrix = [rv 0 0 0]
[0 ov 0 0]
[0 0 dv 0]
[ox oy oz 1]

Wow...all this shit finally makes sense, holy crap. Thanks, Memo!