Course Plan

- lec. 1: Introduction
- lec. 2: Mathematics for 3D Games
- lec. 3: Scene Graph
- lec. 4: Game 3D Physics +
- lec. 5: Game Particle Systems
- lec. 6: Game 3D Models
- lec. 7: Game Textures
- lec. 9: Game Materials
- lec. 8: Game 3D Animations
- lec. 10: 3D Audio for 3D Games
- lec. 11: Networking for 3D Games
- lec. 12: Artificial Intelligence for 3D Games
- lec. 13: Rendering Techniques for 3D Games

37

Point and vector algebra (summary 6/7)

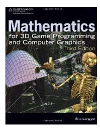
- Dot product (or inner product)
  - Output: a scalar
  - Alternative notations:

$$\vec{v} \cdot \vec{w}$$

$$< \vec{v} , \vec{w} >$$

$$(v^{\mathrm{T}} \vec{w})$$

See...

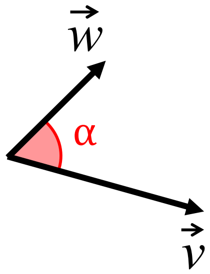


Section 2.2

38

Point and vector algebra (summary 6/7)

- Dot product (or inner product)



$$\vec{v} \cdot \vec{w} \quad = \quad \|\vec{v}\| \cdot \|\vec{w}\| \cdot \cos(\alpha)$$

39

Point and vector algebra (summary 6/7)

- Dot product, useful to:
  - dot is zero: vectors are orthogonal (or, either vector is degenerate)
  - positive dot: acute angle
  - negative dot: obtuse angle

  } valid with both vectors & versors

  - versor dot vector: extension of vector along direction
  - versor dot versor: cosine of angle
  - versor dot versor: also, a similarity measure (in -1 +1)
  - any vector dot itself: its squared length

40

Point and vector algebra (summary 7/7)

- Cross product:

$$\vec{v} \times \vec{w} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \times \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix} = \begin{pmatrix} v_y\ w_z - v_z\ w_y \\ v_z\ w_x - v_x\ w_z \\ v_x\ w_y - v_y\ w_x \end{pmatrix}$$

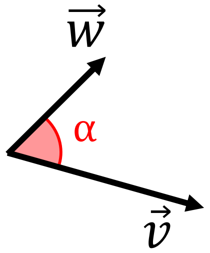
41

Point and vector algebra (summary 7/7)

- Cross product, useful to:
  - find a vector orthogonal to two given vectors
  - therefore: construct orthonormal basis
  - collinearity test (if colinear, then result is (0,0,0))
  - find (double) area of a triangle (floating anywhere in 3D)
  - find normal of a triangle in 3D (remember to renormalize it)
  - norm of (versor cross versor): sin of angle
  - 2D versor × 2D versor: sin of angle

42

Products and angles



$$\vec{v} \cdot \vec{w} = \|\vec{v}\| \, \|\vec{w}\| \, \cos(\alpha)$$

$$\|\vec{v} \times \vec{w}\| = \|\vec{v}\| \, \|\vec{w}\| \, \sin(\alpha)$$

43

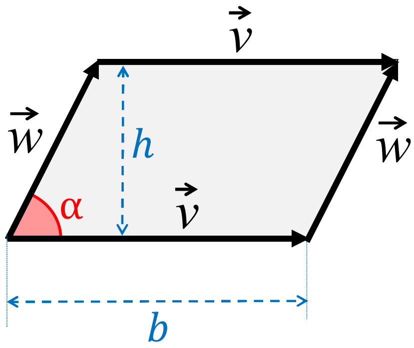
Note: Generalization to N - Dimensions

- Everything seen in this lecture generalizes in 2D (for 2D games), or even in N>3 dimensions
- Exception: the cross product is only defined in 3D
  - But in 2D, the problem of finding a vector/versor orthogonal to one (just one!) given vector/versor is easy: *“swap coordinates, flip one* sign”*
    (x,y) orthogonal to (-y,x), and also to (y,-x)

    *: which coordinate you flip determines if you rotate 90° clockwise or counterclockwise: try!

44

Geometric interpretation: cross product is the parallelogram area



$$\|\vec{v} \times \vec{w}\| = \underbrace{\|\vec{v}\|}_{b} \cdot \underbrace{\|\vec{w}\| \cdot \sin(\alpha)}_{h}$$

45

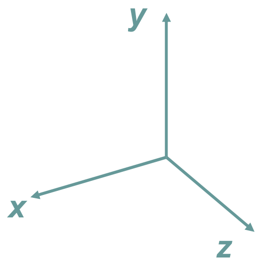
Cross product: full geometric interpretation

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

- Length of $\vec{u}$ = $\|\vec{v}\| \cdot \|\vec{w}\| \cdot \sin(\alpha)$
- Direction of $\vec{u}$ = orthogonal to both $\vec{v}$ and $\vec{w}$
- Verse of $\vec{u}$ = use the «right-hand rule» or the «left-hand rule»
  - whicever hand you are using to imagine your vector space! (and reference frame)

46

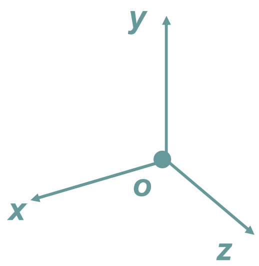
recap: Vector Base



- Axes: set of *n* lin. ind. vectors (***x***,***y***,***z***)
- Any vector ***v*** can be expressed in exactly 1 way as a linar combination of these vectors
- The weights are the coord of ***v*** in that base

47

recap: Reference Frame (or Space)

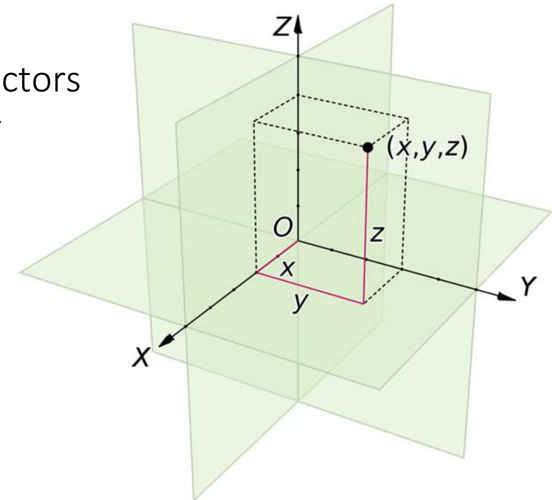


- *n* axes (vectors) (vector base)
  +
  1 origin (point)
- Any vector ***v*** : one linear comb. of the axes
- Any point ***p*** : origin + one linear comb. of axes

48

Recap: Orthonormal Frames Or Cartesian Frame

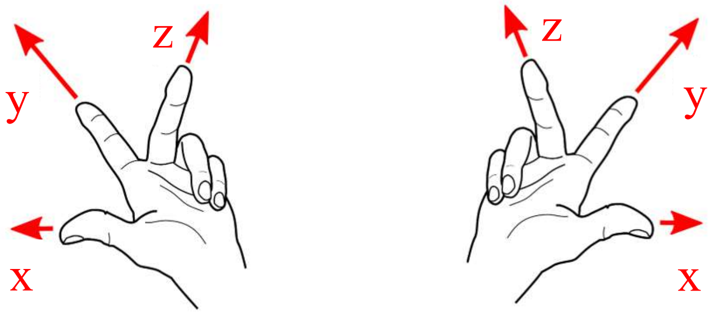
- Axes are unit vectors and reciprocally orthogonal



49

Recap: Handed-ness of a (Cartesian) frame

- They can be right- or left-handed



$x \times y = z$

$x \times y = z$
regardless!

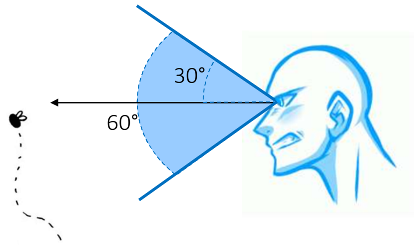
Use the same hand to *imagine* a cross product

50

3D videogames

Points, Vectors, Versors: mini task and exercises

Marco Tarini



63

Points, Vectors, Versors: mini problems

- The following are examples of spatial problem problems that need to be solved in 3D games
  - They can be solved simply using point/vector/versor algebra
  - Many game engines libraries implement functions for many of them
- General schema for finding a solution:
  - identify input and output (and their types)
  - maybe draw a schema
  - write the equations driven by the drawing, (using your spatial understanding of the operations): e.g. what is orthogonal to what?
  - identify the unknowns
  - manipulate the equations according to the rules to extract extract the unknowns
  - if coding: everything is ready to code it!

For some of them, the solution will be given in full here. In other, only a trace of the solution is given

64

Point to point distance (trivial)

"When the player in position p is closer than k to a powerup in pos q, then the powerup is collected"

- Data: p, q points, k scalar
- Test: $\|\mathrm{p}-\mathrm{q}\| < k$
- Optimize vers: $\|\mathrm{p}-\mathrm{q}\|^2 < k^2$
- Pseudo-code example:

```
vec3 p,q;
scalar k;
if ( dot(p-q,p-q) < k*k ) then /*collect*/
```

66

Ray-Plane intersection Ver0

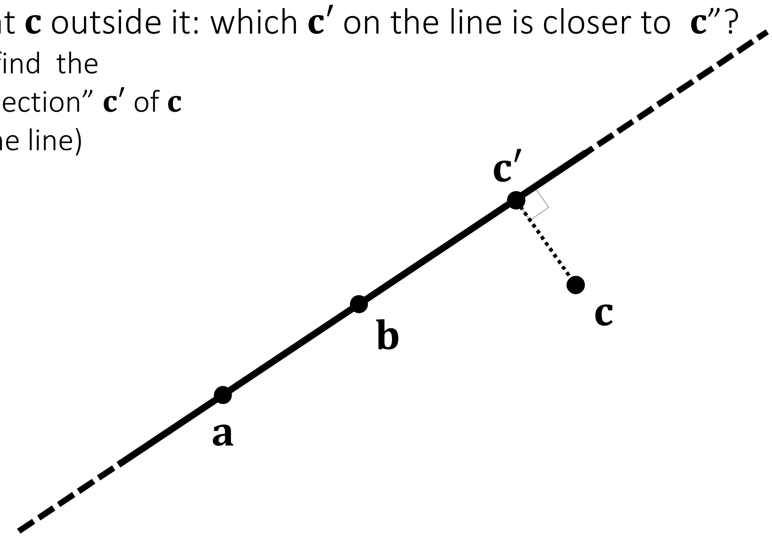
"I shoot a laser from p in direction $\hat{\mathrm{d}}$ toward a plane which contains points q and has normal $\hat{\mathrm{n}}$.
Which point q do I hit?"

- Trace:
  - Define q as a point on the laser (see Ray-Sphere inters.)
  - Define q as a point on the plane
    (hint: the vector connecting it to any other point on the plane is orthogonal to $\vec{\mathrm{n}}$)
  - Combine the two equations into one
  - Extract the only incognita

67

Projection of a point on a line

"Consider a line passing through points $\mathbf{a}$ and $\mathbf{b}$ and a point $\mathbf{c}$ outside it: which $\mathbf{c}'$ on the line is closer to $\mathbf{c}$"?
(i.e. find the "projection" $\mathbf{c}'$ of $\mathbf{c}$ on the line)



68

Sphere-sphere intersection (trivial)

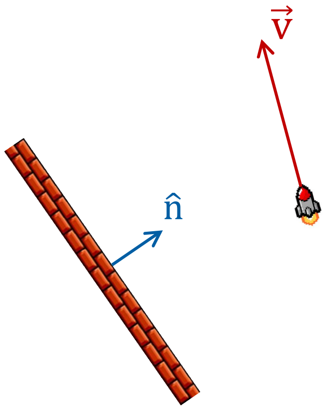
"Given two spheres with center in $\mathbf{c}_0$ and $\mathbf{c}_1$ and radii r_0 and r_1: do they intersect? Do they touch?"

- Hint:
  - remember that working with *squared norms* is more efficient (and more accurate) than working with vector norms

69

The missile and the wall (trivial)

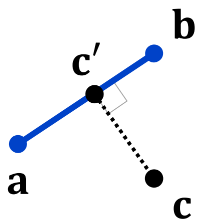
"A missile is moving at constant velocity $\vec{\mathbf{v}}$ (meter per sec), in the general proximity of a large (infinite) wall with normal $\hat{\mathbf{n}}$. After some time t (sec), how much closer to (or farther from) the wall is it?"



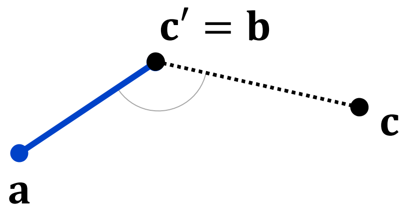
70

Projection of a point on a segment

"Which $\mathbf{c}'$ point on a segment connecting point $\mathbf{a}$ and $\mathbf{b}$ is closer to a third point $\mathbf{c}$"?



case 1



case 2

71

Plane VS Point test

- Input: a point $\mathbf{q}$
  and a plane given by:
  - its normal: $\hat{\mathrm{n}}$
  - a point on it at random: $\mathbf{p}$
- *Q*: on which side of the plane is $\mathbf{q}$?
- *A*: it's the sign of

$\hat{\mathrm{n}} \cdot (\mathbf{q} - \mathbf{p}) =$

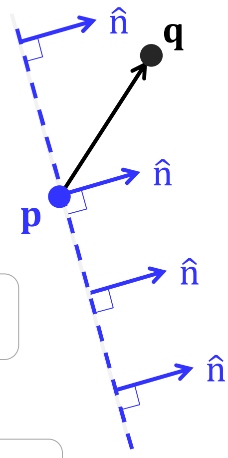
$\hat{\mathrm{n}} \cdot \mathbf{q} - \vec{n} \cdot \mathbf{p} =$

$\hat{\mathrm{n}} \cdot \mathbf{q} + k =$

$(n_x, n_y, n_z, k) \cdot (q_x, q_y, q_z, 1)$

$k = -\vec{n} \cdot \mathbf{p}$
(minus distance of plane from origin)

a 4D vector representing the plane:
a more convenient representation for a plane

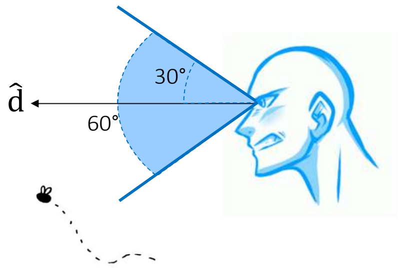


72

Vision cones

"A guard has eyes in position q
and looks in direction $\hat{\mathrm{d}}$.
Does it spot a fly in
position p , if his cone
of vision is 60° wide?"

- Hypotheses: no occlusions



- Trace:
  - For angles α, β in 0..90°: $\alpha < \beta \ \leftrightarrow \cos(\alpha) > \cos(\beta)$
  - Find cosine of angle between view direction and the vector connecting q to p
  - Determine if this cosine is > $\cos(60°/2)$

73

Marco Tarini
Unviersità degli studi di Milano

Ray-sphere intersection

"I shoot a laser from p to direction $\hat{\mathrm{d}}$. Do I hit a sphere in position q of radius r ? Where?"

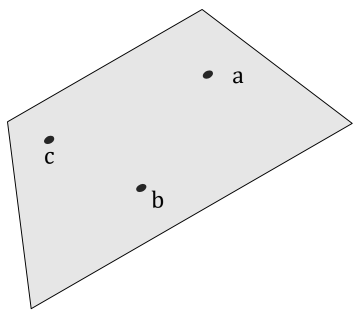
- Data: p, q points, r scalar, $\hat{\mathrm{d}}$ versor
- Trace:
  - Hit-point is s on laser ray:
    $\mathrm{s} = \mathrm{p} + k\,\hat{\mathrm{d}}$, for some unknown scalar $k \geq 0$
  - Hit-point is s on sphere:
    $\|\mathrm{q} - \mathrm{s}\| = r \quad \leftrightarrow \quad (\mathrm{q} - \mathrm{s}) \cdot (\mathrm{q} - \mathrm{s}) = r^2$
  - Combine the two equations (substitute s in second), solve for k (it's a 2nd degree equation), test that k exists and that it is >0)

74

Problem: surface normal (trivial)

"I have three points on a , b , c on a plane: find the normal $\hat{\mathrm{n}}$ of this plane (a versor)"

- Trace:
  find any two
  different vectors
  on the plane
  …
- Question: what determines the direction of $\hat{\mathrm{n}}$?

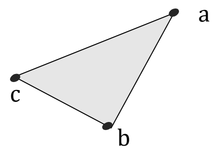


75

Problem: triangle area (trivial)

“I have three points on a , b , c in space.
Find the area of the triangle connecting them”

- Hint:
  it’s half the area
  of a parallelogram



76

Vector orthogonalization

“Find a versor $\hat{\mathrm{u}}'$ that is ortogonal to a given $\hat{\mathrm{n}}$ such that it is as similar as possible to a given versor $\hat{\mathrm{u}}$”

Solution: $\hat{\mathrm{u}}' = \hat{\mathrm{n}} \times \hat{\mathrm{u}} \times \hat{\mathrm{n}}$, then renormalize it.

Coding examples, in different languages:

GLSL
```
vec3 n,u;
u = cross( cross( n , u ) , n );
u = normalize( u );
```

C++, with UE
```
FVector n,u;
u = FVector::CrossProduct( FVector::CrossProduct(n,u),n );
u.Normalize();
```

C#, with Unity
```
Vector3 n,u;
u = Vector3.Cross( Vector3.Cross( n , u ) , n );
u = u.normalized;
```

77

Orthonormal base completion

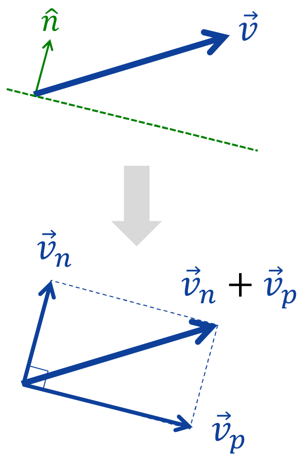
"I have only two axes $\hat{x}$ and $\hat{y}$ of an orthonormal bases, how do I find the third vector $\hat{z}$?"

- Data: $\hat{x}$, $\hat{y}$ versors
- Hypotheses: $\hat{x}$ and $\hat{y}$ are already orthogonal
- Variant: $\hat{y}$ is not exactly orthogonal to $\hat{x}$, but I want to change it the least to make it orthogonal ($\hat{x}$ is to be kept constant)
  (see previous problem)

78

Decompose a vector into components

- Given a vector $\vec{v}$ and a plane normal $\hat{n}$, split $\vec{v}$ in the vector sum $\vec{v} = \vec{v}_n + \vec{v}_p$ with
  - $\vec{v}_n$ orthogonal to the plane (= parallel to $\hat{n}$)
  - $\vec{v}_p$ parallel to the plane (= orthogonal to $\hat{n}$)



Alternative solution

$\vec{v}_n \leftarrow (\hat{n} \cdot \vec{v})\,\hat{n}$

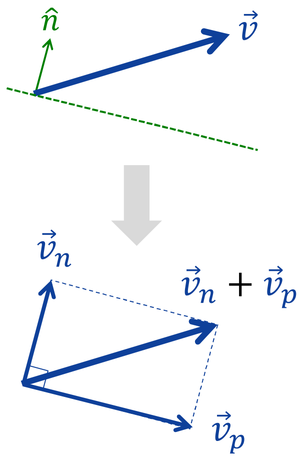
$\vec{v}_p \leftarrow (\hat{n} \times \vec{v}) \times \hat{n}$

try to see why this works

79

Decompose a vector into components

- Given a vector $\vec{v}$ and a plane normal $\hat{n}$, split $\vec{v}$ in the vector sum $\vec{v} = \vec{v}_n + \vec{v}_p$ with
  - $\vec{v}_n$ orthogonal to the plane (= parallel to $\hat{n}$)
  - $\vec{v}_p$ parallel to the plane (= orthogonal to $\hat{n}$)



- A solution in 3 steps:

(1) $k \leftarrow \vec{v} \cdot \hat{n}$ — k is a (signed) scalar: the extension of $\vec{v}$ along dir $\hat{n}$

(2) $\vec{v}_n \leftarrow k\ \hat{n}$ — $\vec{v}_n$ is the component of $\vec{v}$ along $\hat{n}$

(3) $\vec{v}_p \leftarrow \hat{n}\ \vec{v} - \vec{v}_n$ — $\vec{v}_p$ the component of $\vec{v}$ orthogonal to $\hat{n}$

80

Line-Line "intersection"

"Given two 3D lines, find the two points on both lines that are as close as possible to each other"

(they are the same point, if the lines intersect!)

- Input: a point on line "A" $\mathrm{p_A}$ and its direction $\hat{\mathrm{d}}_\mathrm{A}$
  a point on line "B" $\mathrm{p_B}$ and its direction $\hat{\mathrm{d}}_\mathrm{B}$
- Output: two points $\mathrm{q_A}$ and $\mathrm{q_B}$

81

Ray-Plane intersection Ver1

"I shoot a laser from $\mathbf{p}$ in direction $\hat{\mathrm{d}}$ toward a plane which contains points $\mathbf{a}$ $\mathbf{b}$ $\mathbf{c}$. Which point $\mathbf{q}$ do I hit?"

- Hypotheses: $\mathbf{a}$ $\mathbf{b}$ $\mathbf{c}$ are not colinear (not on a line)
- Trace:
  - Find vector $\vec{\mathrm{n}}$ orthogonal to plane, use cross product (question for later: are magnitude and verse important?)
  - Define $\mathbf{q}$ as a point on the laser (see Ray-Sphere inters.)
  - Define $\mathbf{q}$ as a point on the plane (hint: the vector connecting it to any other point on the plane is orthogonal to $\vec{\mathrm{n}}$)
  - Combine the two equations into one
  - Extract the only incognita

82

Plane-plane intersection

"Given two 3D planes, find the line they share"

- Input: a point on plane "A" $\mathbf{p}_\mathrm{A}$ and its normal $\hat{\mathrm{n}}_\mathrm{A}$
  a point on plane "B" $\mathbf{p}_\mathrm{B}$ and its normal $\hat{\mathrm{n}}_\mathrm{B}$
- Output:
  a point on the line $\mathbf{q}$ and the line direction $\hat{\mathrm{d}}$

83

Shooting a walking target (with a finite speed bullet) 1/2

“I shoot a bullet from p with velocity $\vec{\mathrm{v}}$. At which time the bullet will be the closest to a target currently in position q and moving with velocity $\vec{\mathrm{w}}$? Where will bullet and target be, at that point?”

(useful, e.g., for a sniper AI “*leading*” a target)

- Data: p, q points, $\vec{\mathrm{v}}$ and $\vec{\mathrm{w}}$ vectors
- Hypothesis: nothing accelerates (everything keeps moving at a constant speed)

84

Shooting a walking target (with a finite speed bullet) 2/2

Trace

- Position of bullet at time t : $\mathrm{p} + t\,\vec{\mathrm{v}}$
- Position of target at time t : $\mathrm{q} + t\,\vec{\mathrm{w}}$
- Squared distance between the two at time t :

$$\| (\mathrm{p} + t\,\vec{\mathrm{v}}) - (\mathrm{q} + t\,\vec{\mathrm{w}}) \|^2$$

$$=$$

$$\| (\mathrm{p} - \mathrm{q}) + t\,(\vec{\mathrm{v}} - \vec{\mathrm{w}}) \|^{2}$$

- Work on formulas (remember that $\|\vec{\mathrm{v}}\|^2 = \vec{\mathrm{v}} \cdot \vec{\mathrm{v}}$)
  find derivative for dt,
  equate derivative to 0, extract t

85