



Course Plan



- lec. 1: **Introduction** ●
- lec. 2: **Mathematics** for 3D Games ●●●●●● (with a red arrow pointing to the second dot)
- 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: **Networking** for 3D Games ●
- lec. 11: **3D Audio** for 3D Games ●
- lec. 12: **Rendering Techniques** for 3D Games ●
- lec. 13: **Artificial Intelligence** for 3D Games ●

30

Point and vector algebra (summary 6/7)



- **Interpolate** between pairs of <something> :
 - $\text{mix}(\text{point}, \text{point}, t) \rightarrow \text{point}$
 - $\text{mix}(\text{vector}, \text{vector}, t) \rightarrow \text{vector}$
 - $\text{mix}(\text{versor}, \text{versor}, t) \rightarrow \text{versor}$
- t is a **scalar «weight»**
 - $t = 0 \rightarrow$ pick the first one
 - $t = 1 \rightarrow$ pick the second one
 - $t \in (0,1) \rightarrow$ get something in between, for example: ← a proper interpolation
 - $t = 0.5 \rightarrow$ just **average** the two
 - $t = 0.1 \rightarrow$ use almost the first, with just a bit of the second in it
 - $t < 0$ or $t > 1 \rightarrow$ **extrapolate**
- Terminology: (in libraries, game engines...)
 - **interpolate** = **mix** = **blend** = **lerp** ← specifically linear

31

Interpolation in general - notes



- Very used in Computer Graphics (e.g., rendering, animation)
- Terminology:
 - $a\mathbf{x} + b\mathbf{y}$: a **linear combination** of \mathbf{x} and \mathbf{y}
 - if $a+b=1$ and $a, b \in [0,1]$: a **(linear) interpolation** of \mathbf{x} and \mathbf{y}
 - if $a+b=1$ but $a, b \notin [0,1]$: a **(linear) extrapolation** of \mathbf{x} and \mathbf{y}
 - a, b : the used **weights**
 - $a + b = 1$: weights are a **partition of unity**
- Generalizes to > 2 objects ($a\mathbf{x} + b\mathbf{y} + c\mathbf{z}$)
- When interpolating 2 objects, we can just give one weight t .
 - The other is given by difference. $a = t, b = 1-t$
 - The interpolation is often written in programming languages as `mix($\mathbf{x}, \mathbf{y}, t$)` (or similar ways). Remember: $t=0 \rightarrow$ pick the first
- It's a general concept! All sorts of objects can be interpolated
 - Intuition: interpolation = a mix between objects
 - Let's analyze case of Points, Vectors, Versors

32

How to interpolate between...



But easily generalizes to > 2

Linear interpolation

- ...two **vectors** \mathbf{v}_0 and \mathbf{v}_1 :

$$(1 - t)\mathbf{v}_0 + (t)\mathbf{v}_1$$

- ...two **points** \mathbf{p}_0 and \mathbf{p}_1 :

$$\mathbf{p}_0 + t(\mathbf{p}_1 - \mathbf{p}_0)$$

which you may also want to write as:

$$(1 - t)\mathbf{p}_0 + (t)\mathbf{p}_1$$

Summing... two points ??

Scaling... a point ??

We usually don't need any such operation.
But it's equivalent, mathematically.

Only legal operations with an easily defined geometric meaning (to-do: check)

33

How to interpolate between...

But easily generalizes to > 2

Linear interpolation

- ...two **vectors** \mathbf{v}_0 and \mathbf{v}_1 :

$$(1 - t) \mathbf{v}_0 + (t) \mathbf{v}_1$$
- ...two **points** \mathbf{p}_0 and \mathbf{p}_1 :

$$\mathbf{p}_0 + t (\mathbf{p}_1 - \mathbf{p}_0)$$
- ...two **versors** \mathbf{d}_0 and \mathbf{d}_1 :

$$(1 - t) \mathbf{d}_0 + (t) \mathbf{d}_1$$

 then renormalize the result (it's no longer unitary).
 Or, use "spherical interpolation" (aka "slerp")...

34

LERP vs SLERP (of versors)

Linear interpolation:

$\mathbf{d} = \text{lerp}(\mathbf{d}_0, \mathbf{d}_1, \frac{2}{3})$

Spherical interpolation:

$\mathbf{d} = \text{slerp}(\mathbf{d}_0, \mathbf{d}_1, \frac{2}{3})$

Then, renormalize:

Not the same result!

- But, close enough
- Even closer when:
 - $\mathbf{d}_0, \mathbf{d}_1$ similar OR t close to $\frac{1}{2}$
- Is it worth the extra computation cost? 😞

35

The formulas

- LERP + normalization:

$$(1 - t) \mathbf{d}_0 + t \mathbf{d}_1 \quad \left. \vphantom{(1 - t) \mathbf{d}_0 + t \mathbf{d}_1} \right\} \text{ aka "NLERP"}$$

then re-normalize

- or SLERP:

$$\frac{\sin((1 - t) \alpha)}{\sin(\alpha)} \mathbf{d}_0 + \frac{\sin(t \alpha)}{\sin(\alpha)} \mathbf{d}_1$$

angle
between
 \mathbf{d}_0 and \mathbf{d}_1

36

SLERP: notes

- Applicable to any versor (unit vector) including 2D, 3D, and quaternions (see later)
- SLERP can even be used on general vectors:
 - Compute magnitudes of vectors
 - Compute directions of vectors (divide by magnitude, i.e., normalize)
 - new direction = SLERP of the directions (unit vectors)
 - new magnitude = LERP of the magnitudes (scalars)
 - multiply new dir with new mag to get the final result

37

Point and vector algebra (summary 7/7)



- Cross product:

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

38

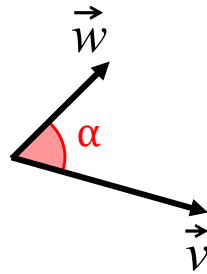
Point and vector algebra (summary 7/7)



- 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 \times 2D versor: sin of angle

39

Products and angles



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

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

40

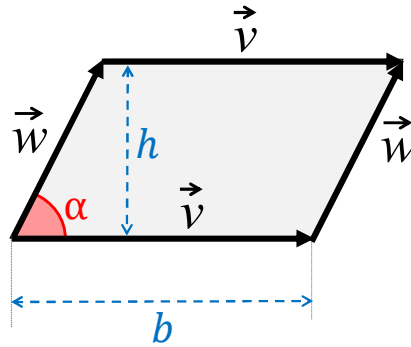
Note: Generalization to N - Dimensions

- 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!

41

Geometric interpretation:
cross product is the parallelogram area



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

42

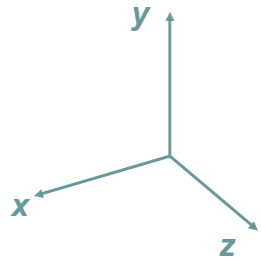
Cross product:
full geometric interpretation

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

- 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»
 - whichever hand you are using to imagine your vector space! (and reference frame)

43

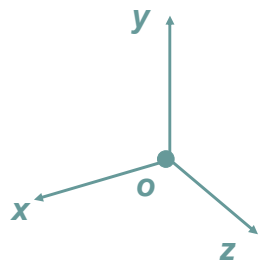
recap: Vector Base



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

44

recap: Reference Frame (or Space)

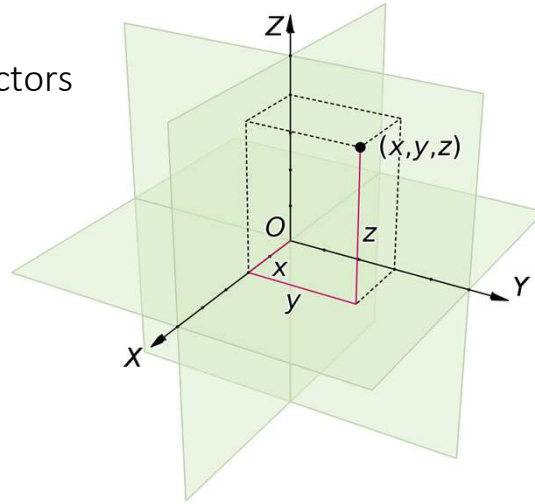


- n axes (vectors) (vector base)
+
1 origin (point)
- Any vector \mathbf{v} :
one linear comb. of the axes
- Any point \mathbf{p} :
origin + one linear comb. of axes

45

Recap: Orthonormal Frames Or Cartesian Frame

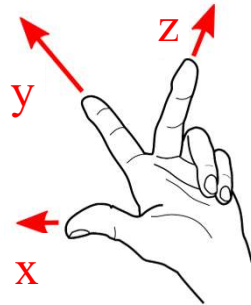
- Axes are unit vectors and reciprocally orthogonal



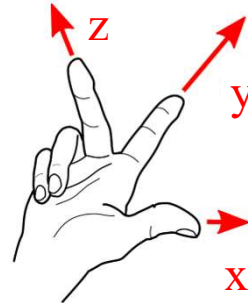
46

Recap: Handed-ness of a (Cartesian) frame

- They can be right- or left-handed



$$x \times y = z$$



$$x \times y = z$$

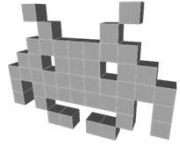
regardless!

Use the same hand to *imagine* a cross product

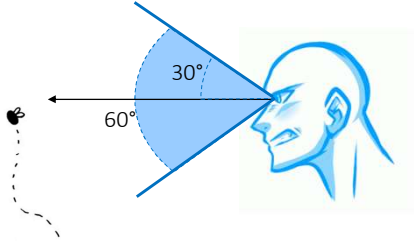
47

3D videogames

Points, Vectors, Versors: mini task and exercises Part II




Marco Tarini



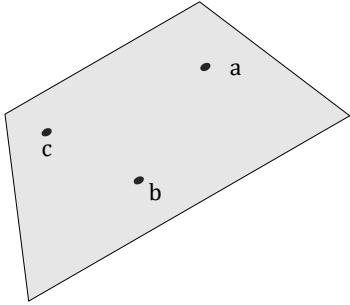
70

Problem: surface normal (trivial)



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

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



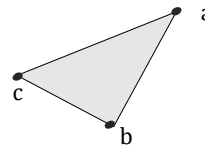
71

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



72

Vector orthogonalization



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

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

Coding examples, in different languages:

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

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

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

73

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} ?”

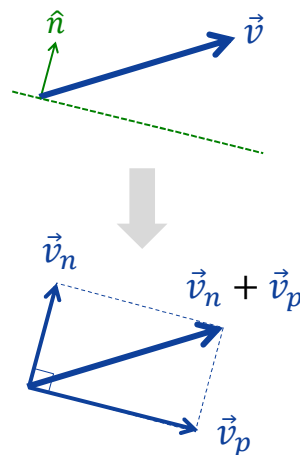
- Data: \hat{x} , \hat{y} versors
- Hypotheses: \hat{x} and \hat{y} are already orthogonal
- Variant: \hat{y} is not exactly orthogonal to \hat{x} , but I want to change it the least to make it orthogonal (\hat{x} is to be kept constant)
(see previous problem)

74

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

75

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:
 - $k \leftarrow \vec{v} \cdot \hat{n}$ k is a (signed) scalar: the extension of \vec{v} along dir \hat{n}
 - $\vec{v}_n \leftarrow k \hat{n}$ \vec{v}_n is the component of \vec{v} along \hat{n}
 - $\vec{v}_p \leftarrow \hat{n} \vec{v} - \vec{v}_n$ the component of \vec{v} orthogonal to \hat{n}

76

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” \mathbf{p}_A and its direction $\hat{\mathbf{d}}_A$
a point on line “B” \mathbf{p}_B and its direction $\hat{\mathbf{d}}_B$
- Output: two points \mathbf{q}_A and \mathbf{q}_B

77

Ray-Plane intersection Ver1



“I shoot a laser from \mathbf{p} in direction $\hat{\mathbf{d}}$ 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 $\vec{\mathbf{n}}$ 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 $\vec{\mathbf{n}}$)
 - Combine the two equations into one
 - Extract the only incognita

78

Plane-plane intersection



“Given two 3D planes, find the line they share”

- Input: a point on plane “A” \mathbf{p}_A and its normal $\hat{\mathbf{n}}_A$
a point on plane “B” \mathbf{p}_B and its normal $\hat{\mathbf{n}}_B$
- Output:
a point on the line \mathbf{q} and the line direction $\hat{\mathbf{d}}$

79

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



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

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

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

80

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



Trace

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

$$\begin{aligned} & \| (\mathbf{p} + t \vec{\mathbf{v}}) - (\mathbf{q} + t \vec{\mathbf{w}}) \|^2 \\ &= \\ & \| (\mathbf{p} - \mathbf{q}) + t (\vec{\mathbf{v}} - \vec{\mathbf{w}}) \|^2 \end{aligned}$$

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

81