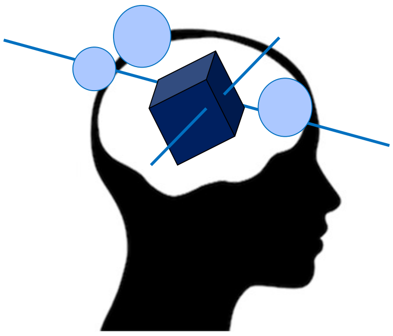
3D VideoGames - UniMi

Points, Vectors, Versors (recap)

Marco Tarini



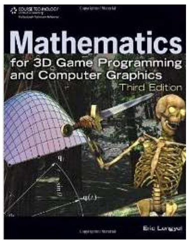
3

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: 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

4

Suggested reading



Mathematics for 3D Game Progr. and C.G. (3rd ed)
Eric Lengyel
Chapters 2, 3, 4

5

Point, Vectors, Versors and Spatial Transformation

They are the basic data-type of 3D Games

- In the computation, for all modules
  - rendering engine
  - physics engine
  - AI
  - 3D sound
  - …
- In the data structures of all 3D Assets
  - Meshes, animations, etc

6

Point, Vectors, Versors

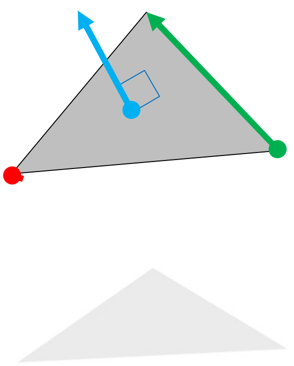
	represents:	example:	imagine it as...
Point	A position A location	Where a character is The center of a sphere	a small floating dot :-D
Vector	A displacement The difference between 2 points. The vector that connects them.	The velocity of a thrown knife The gravity acceleration How to reach the head of a character from its neck	a small arrow :-D *(length is relevant)*
Versor aka unit vector (as length = 1) aka normal aka direction aka normalized vector	A direction A facing	The view direction of a character The facing of a plane in 3D (i.e. its "normal") The direction of a line, or a ray A rotation axis	the same :-D *(its length is irrelevant)*

7

Points, Vectors, Versors
...on a 3D floating tirangle

Examples of...

- point:
  - one vertex of the triangle
- vector:
  - one side of the triangle
- versor:
  - the «normal» of the triangle

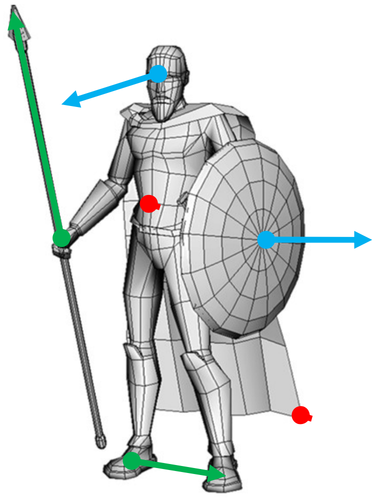


8

Points, Vectors, Versors
...in a character

Examples of...

- points
- vectors
- versors

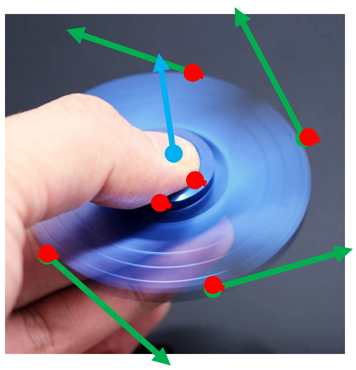


10

Points, Vectors, Versors
...in a spinner

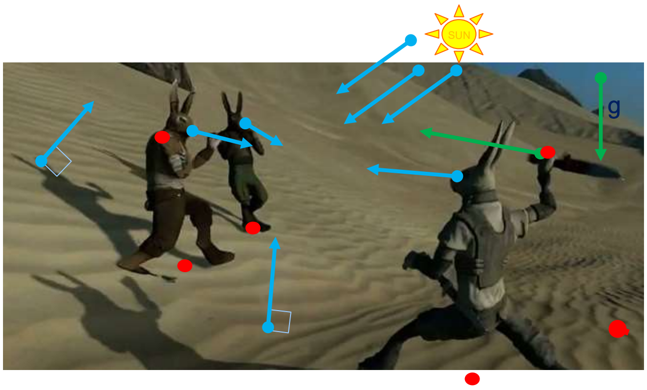
Examples of...

- points
- vectors
- versors



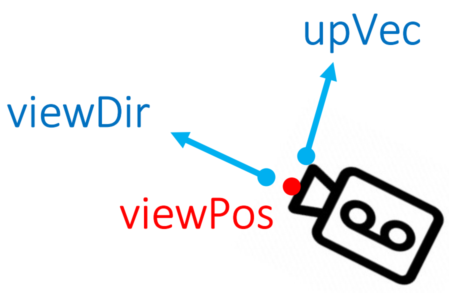
13

Points, Vectors, Versors
...in this screenshot



14

Stuff = Points + Vectors + Versors



Description of a camera
(its so called «extrinsic parameteres»)

15

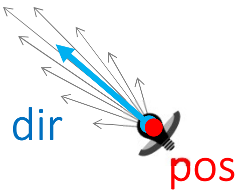
Stuff = Points + Vectors + Versors

description of
a (directional) sound emitter

description of
a (directional) microphone

16

Stuff = Points + Vectors + Versors

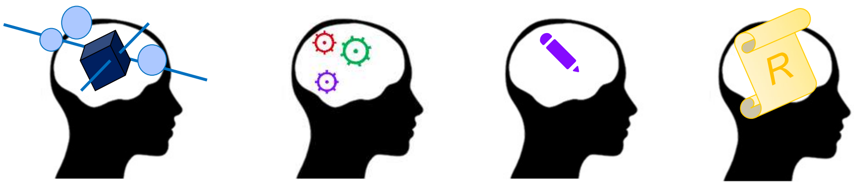


description of a spotlight

17

Point, vector, versor *algebra*

make sure to understand
each operation in 3 different ways



intuitive
know how
to imagine it
spatially,
in 3D

operational
know how to
compute
from data

syntactic
know how to
write down,
(in paper
or code)

plus rules:
know how
to manipulate
equations

18

The *algebra* of points, vectors, versors (and scalars)

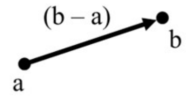
- also, familiarize with the way the operations behave, i.e. rules such as

(1) commutativity? associativity? (of each operation)
(2) distributivity? (between pairs of operations)
(3) inverse operation? identity element? absorbing element?

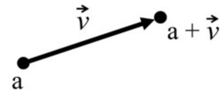
20

Point and vector algebra (summary 1/7)

- Difference:
  point – point = vector



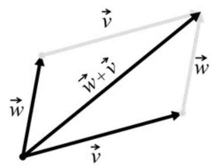
- Addition:
  point + vector = point

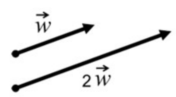


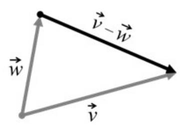
22

Point and vector algebra (summary 2/7)

- Linear operations for vectors
  - addition (vector + vector = vector)
  - product with a scalar (scaling) (vector * scalar = vector)
  - therefore: interpolation
    $\text{mix}(\overrightarrow{v_0}, \overrightarrow{v_1}, t) = (1-t)\,\overrightarrow{v_0} + t\,\overrightarrow{v_1}$
  - therefore: opposite (flip verse) (how to: multiply by – 1)
  - therefore: difference (vector – vector = vector)

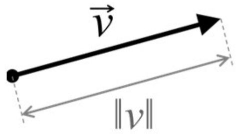






23

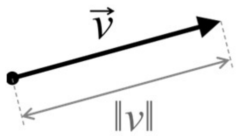
Point and vector algebra (summary 3/7)



- Norm (for vectors)
  - aka length / magnitude / Euclidean norm / 2-norm
  - distance between points:
    length of vector (a – b) = distance between a and b
  - Rules: triangle inequality:

24

Point and vector algebra (summary 4/7)



- Normalization
  - Input: a vector. Result: a versor
  - how to: scale the vector by (1.0 / length)

25

Point and vector algebra (summary 5/7)

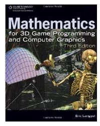
- Dot product (or inner product)
  - Output: a scalar
  - Alternative notations:

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

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

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

See...

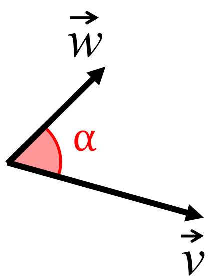


Section 2.2

26

Point and vector algebra (summary 5/7)

- Dot product (or inner product)



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

27

Point and vector algebra (summary 5/7)

- 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

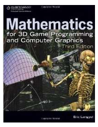
  - versor dot vector: extension of vector along direction
  - versor dot versor: cosine of angle
  - versor dot versor: also, a similarity measure (in -1 +1)
  - any vector dot itself: its squared length

28

Point and vector algebra Products: additional reading

To be continued!

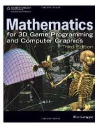
Products between vectors and/or versors



Section 2.2

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

NEXT LECTURE!



Section 2.3

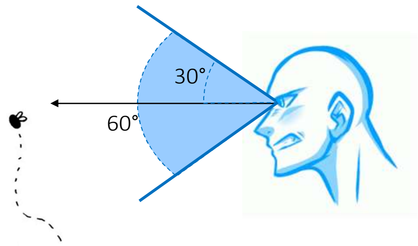
- Cross product (or vector product)
  - Output: a vector (note: not a versor)

29

3D videogames

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

Marco Tarini



59

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

60

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*/
```

62

Ray-Plane intersection Ver0

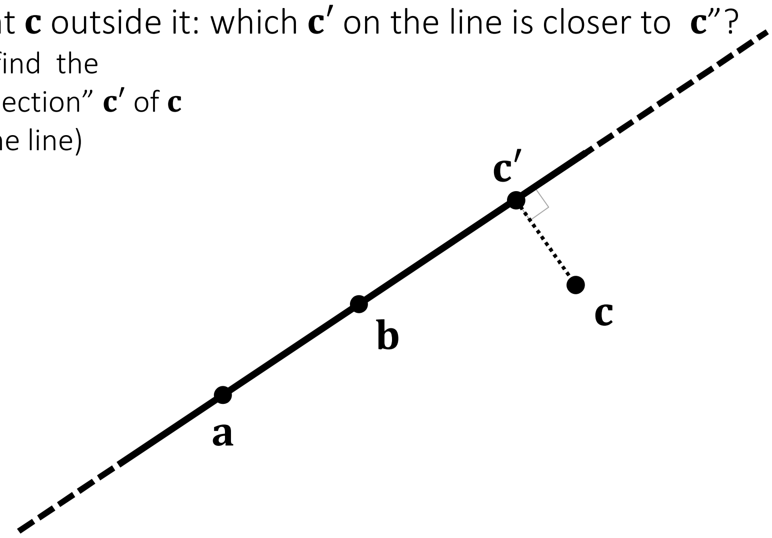
"I shoot a laser from p in direction $\hat{\mathrm{d}}$ toward a plane which contains points q and has normal $\hat{\mathrm{n}}$.
Which point q do I hit?"

- 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

63

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)



64

Sphere-sphere intersection (trivial)

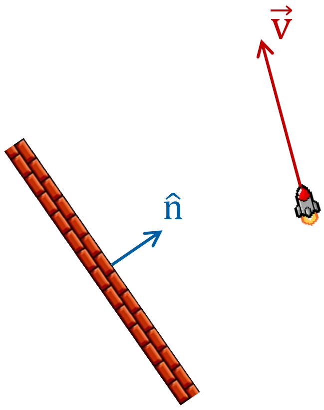
"Given two spheres with center in $\mathbf{c}_0$ and $\mathbf{c}_1$ and radii r_0 and r_1: do they intersect? Do they touch?"

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

65

The missile and the wall (trivial)

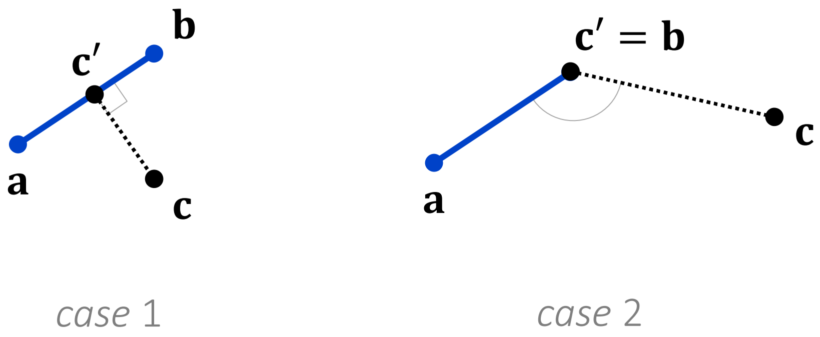
"A missile is moving at constant velocity $\vec{v}$ (meter per sec), in the general proximity of a large (infinite) wall with normal $\hat{n}$. After some time t (sec), how much closer to (or farther from) the wall is it?"



66

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 $\mathbf{c}$"?



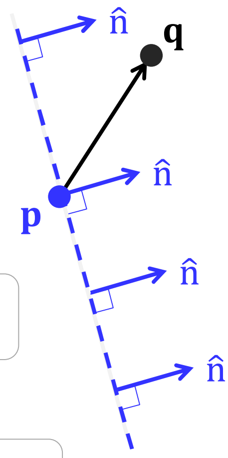
67

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

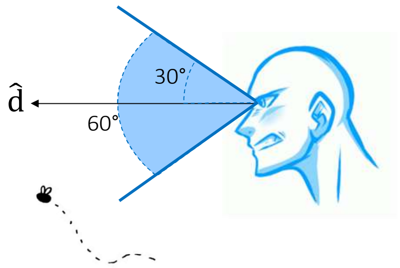


68

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 > $\cos(60°/2)$

69