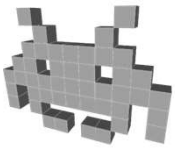
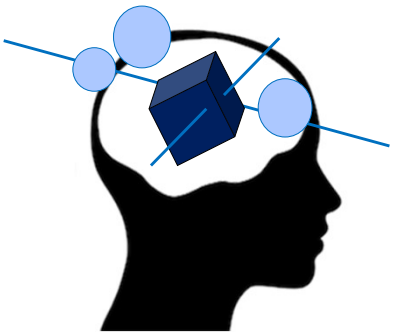
3D VideoGames - UniMi

Points, Vectors, Versors (recap)

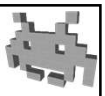




Marco Tarini

3

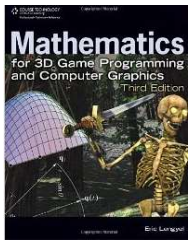
Course Plan



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

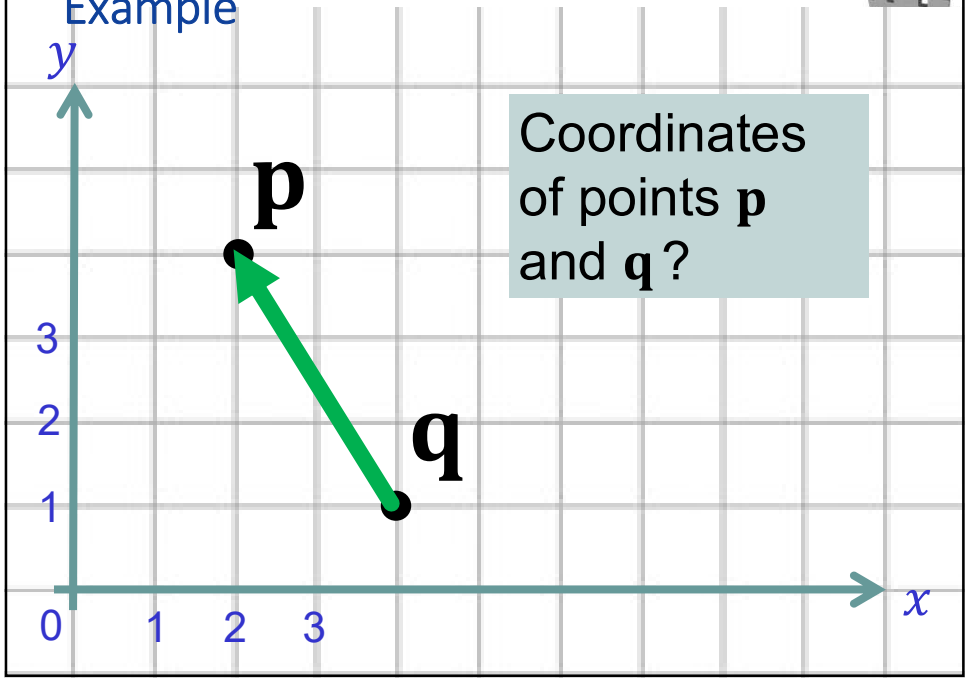
6

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)*

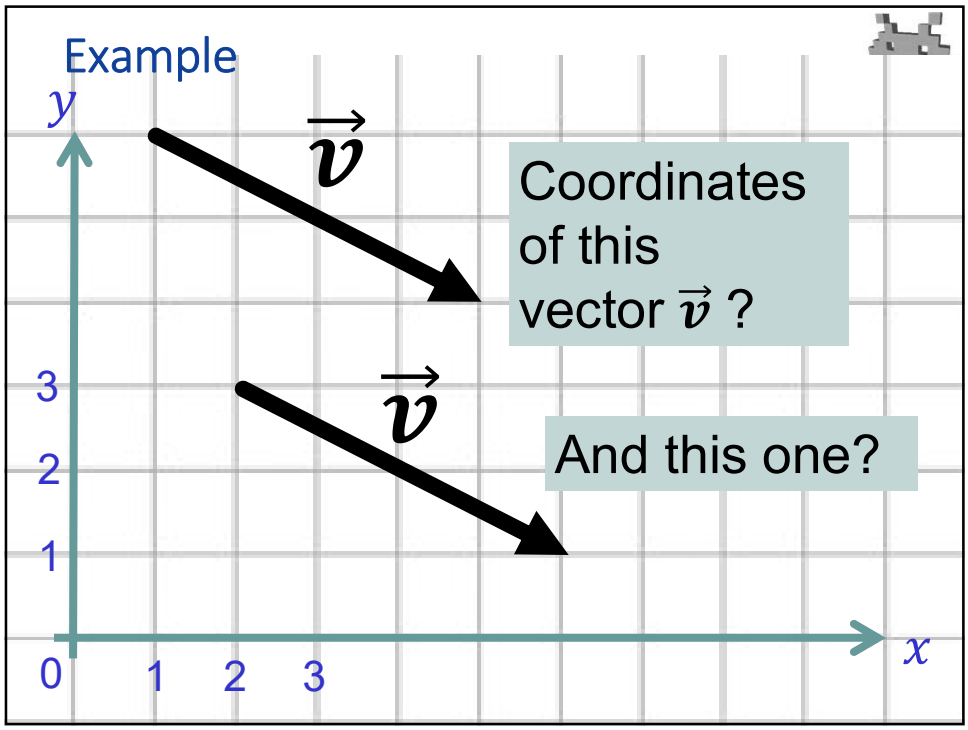
7

Example



Coordinates of points **p** and **q**?

10

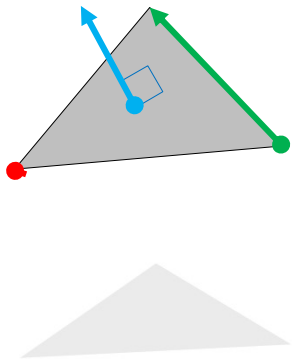


12

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

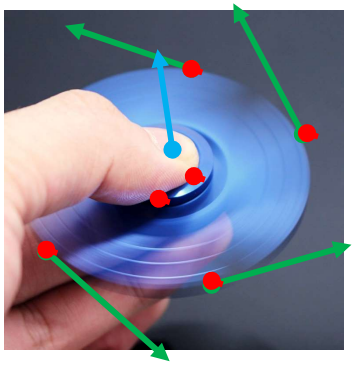


13

Points, Vectors, Versors ...in a spinner

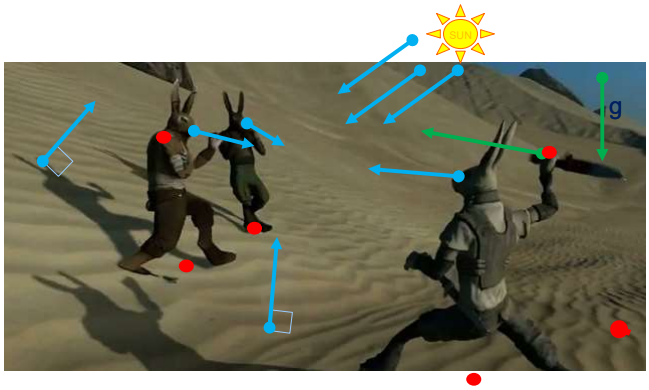
Examples of...

- points:
  - points of contact between finger-spinner
- vectors:
  - linear velocities of these four points
- versors:
  - rotation axis (direction of)



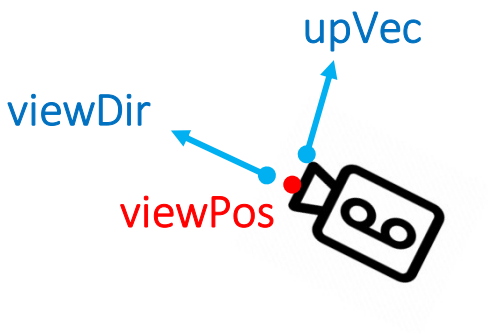
17

Points, Vectors, Versors ...in this screenshot



19

Stuff = Points + Vectors + Versors



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

20

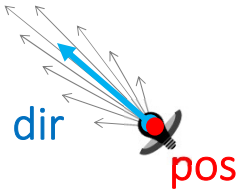
Stuff = Points + Vectors + Versors

description of
a (directional) sound emitter

description of
a (directional) microphone

21

Stuff = Points + Vectors + Versors



description of a spotlight

22

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

- make sure you understand each of operation in 3 different ways:

- intuitive / spatial: what does it do conceptually / visually in 3D

- operational: how to compute the result, starting from
  (a) the coordinates of the operand(s)
  (b) occasionally, also
  the angle between the operands, their lengths, etc

- syntactic: how to write them down
  (a) on paper (mathematical notation)
  (b) in a programming language (Unity C# lib, Unreal C++ lib, GLSL…)

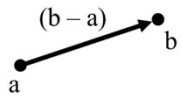
- also, familiarize how to manipulate equations, i.e. rules such as

(a) commutativity? associativity? (of each operation)
(b) distributivity? (between pairs of operations)
(c) inverse operation? identity element? absorbing element?

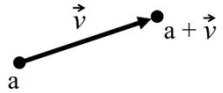
24

Point and vector algebra (summary 1/7)

- Difference:
  point – point = vector



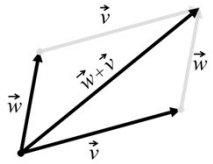
- Addition:
  point + vector = point

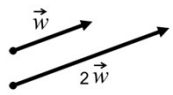


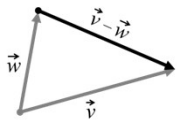
25

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
    $\mathrm{mix}(\overrightarrow{v_0}, \overrightarrow{v_1}, t) = (1-t)\,\overrightarrow{v_0} + t\,\overrightarrow{v_1}$
  - therefore: opposite (flip verse)
    (how to: multiply by – 1)
  - therefore: difference
    (vector – vector = vector)

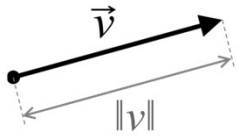






26

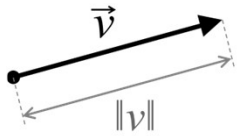
Point and vector algebra (summary 3/7)



- 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:

27

Point and vector algebra (summary 4/7)



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

28

Point and vector algebra (summary 5/7)

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

a proper interpolation

specifically linear

29

Interpolation in general - notes

- Very used in Computer Graphics (e.g., rendering, animation)
- Terminology:
  - $a\,\mathbf{x} + b\,\mathbf{y}$: a linear combination of **x** and **y**
  - if $a+b=1$ and $a,b \in [0,1]$: a (linear) interpolation of **x** and **y**
  - if $a+b=1$ but $a,b \notin [0,1]$: a (linear) extrapolation of **x** and **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( x, y, t )` (or similar ways). Rember: t=0 → 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

30

How to interpolate between…

But easily generalizes to > 2

Linear interpolation

- …two vectors $\mathbf{v}_0$ and $\mathbf{v}_1$:

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

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

- …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.

31

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 $\mathbf{d}_1$:

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

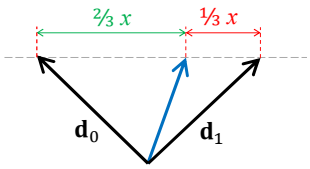
then renormalize the result (it's no longer unitary).

Or, use "spherical interpolation" (aka "slerp")…

32

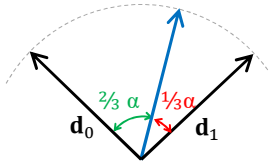
LERP vs SLERP (of versors)

Linear interpolation:



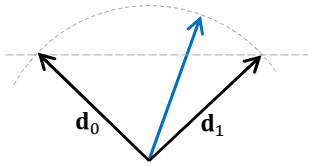
$\mathbf{d} = \mathrm{lerp}(\mathbf{d}_0, \mathbf{d}_1, ⅔)$

Spherical interpolation:



$\mathbf{d} = \mathrm{slerp}(\mathbf{d}_0, \mathbf{d}_1, ⅔)$

Then, renormalize:



Not the same result!

- But, close enough
- Even closer when:
  $\mathbf{d}_0$, $\mathbf{d}_1$ similar OR t close to ½
- Is it worth the extra computation cost? 🤔

33

The formulas

- LERP + normalization:

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

then re-normalize

aka "NLERP"

- or SLERP:

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

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

34

SLERP: notes

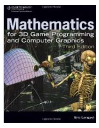
- Applicable to any versor (unit vector) including 2D, 3D, and quaternions (see later)
- SLERP can even be used on general vectors:
  - Compute magnitudes of vectors
  - Compute directions of vectors (divide by magnitude, i.e., normalize)
  - new direction = SLERP of the directions (unit vectors)
  -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

35

Point and vector algebra
Products: additional reading

To be continued!

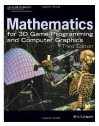
Products between vectors and/or versors



Section 2.2

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

NEXT LECTURE!



Section 2.3

- Cross product (or vector product)
  - Output: a vector

36