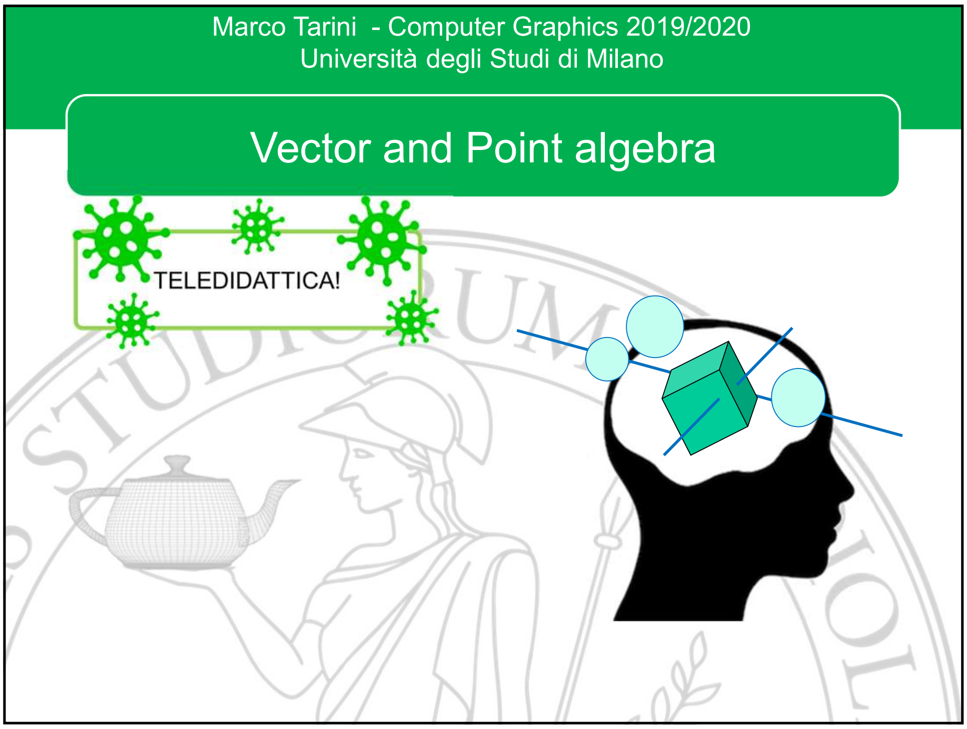
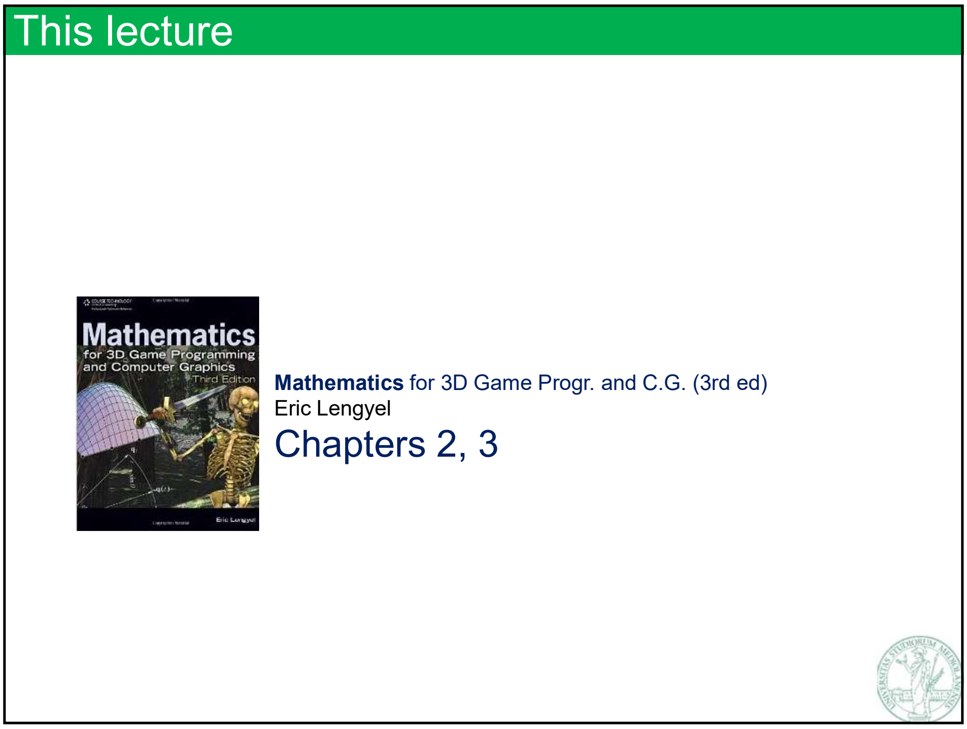


Marco Tarini - Computer Graphics 2019/2020
Università degli Studi di Milano

Vector and Point algebra

TELEDIDATTICA!

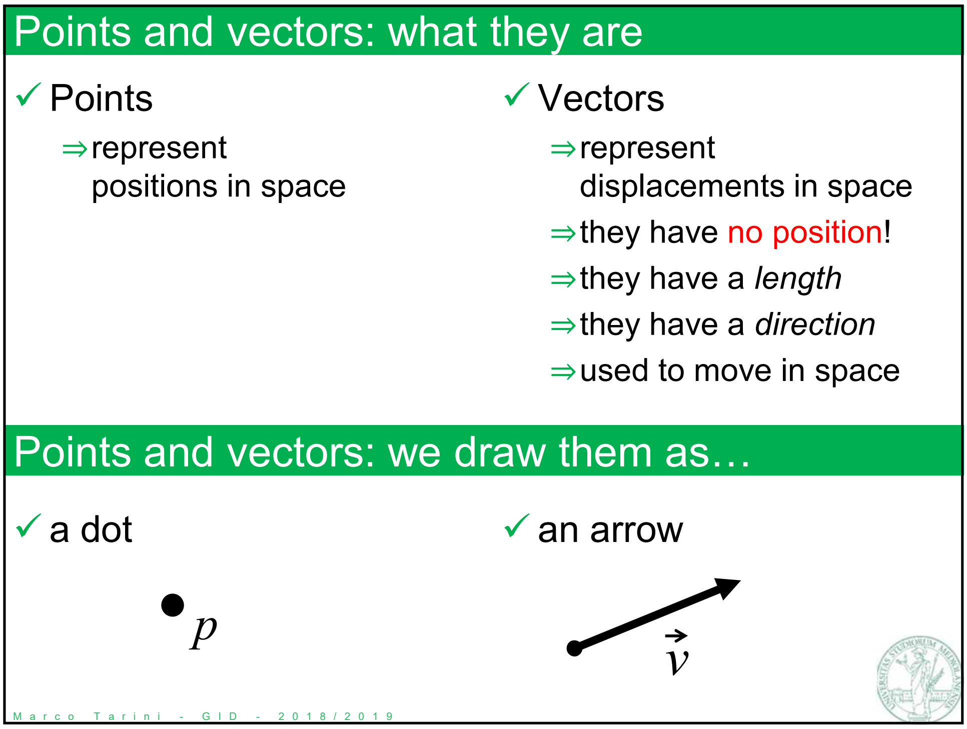
1



This lecture

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

2



Points and vectors: what they are

- ✓ Points
  - ⇒ represent positions in space
- ✓ Vectors
  - ⇒ represent displacements in space
  - ⇒ they have no position!
  - ⇒ they have a *length*
  - ⇒ they have a *direction*
  - ⇒ used to move in space

Points and vectors: we draw them as…

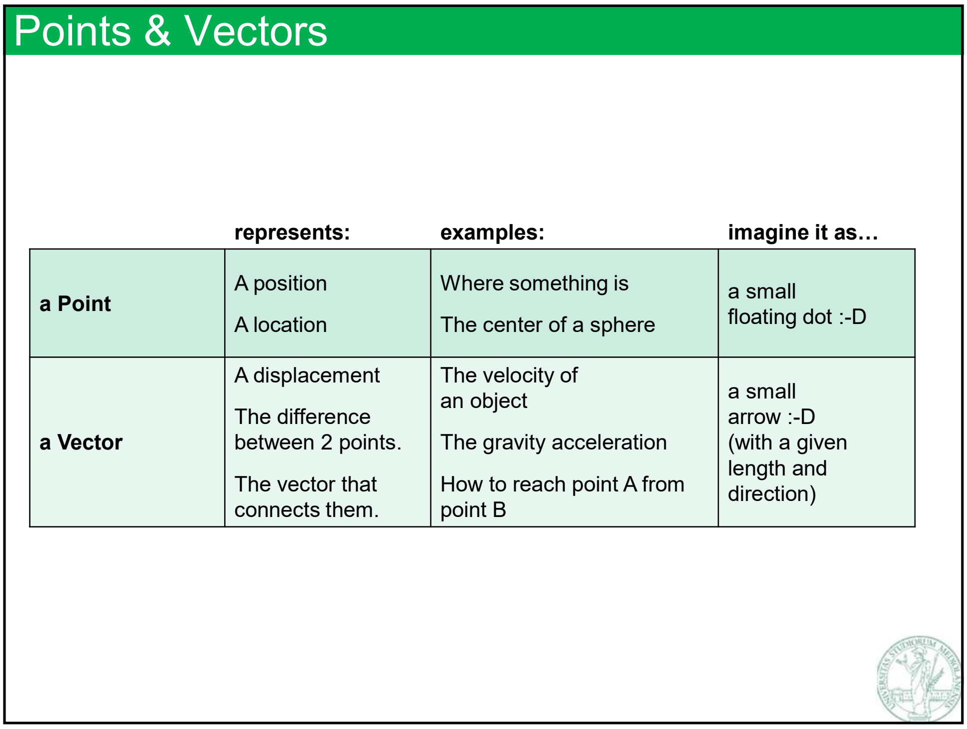
- ✓ a dot
- ✓ an arrow

p

$\vec{v}$

Marco Tarini - GID - 2018/2019

3



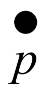
Points & Vectors

	represents:	examples:	imagine it as…
a Point	A position A location	Where something is The center of a sphere	a small floating dot :-D
a Vector	A displacement The difference between 2 points. The vector that connects them.	The velocity of an object The gravity acceleration How to reach point A from point B	a small arrow :-D (with a given length and direction)

4

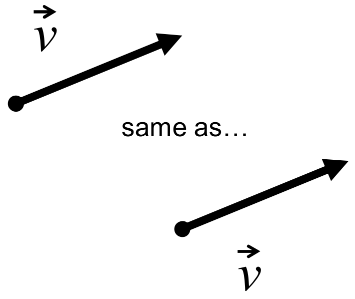
Points and vectors: we draw / write them as…

✓ Point: a dot



✓ Vector: a small arrow

⇒ Note:
the arrow is drawn
in some place,
but they have no position



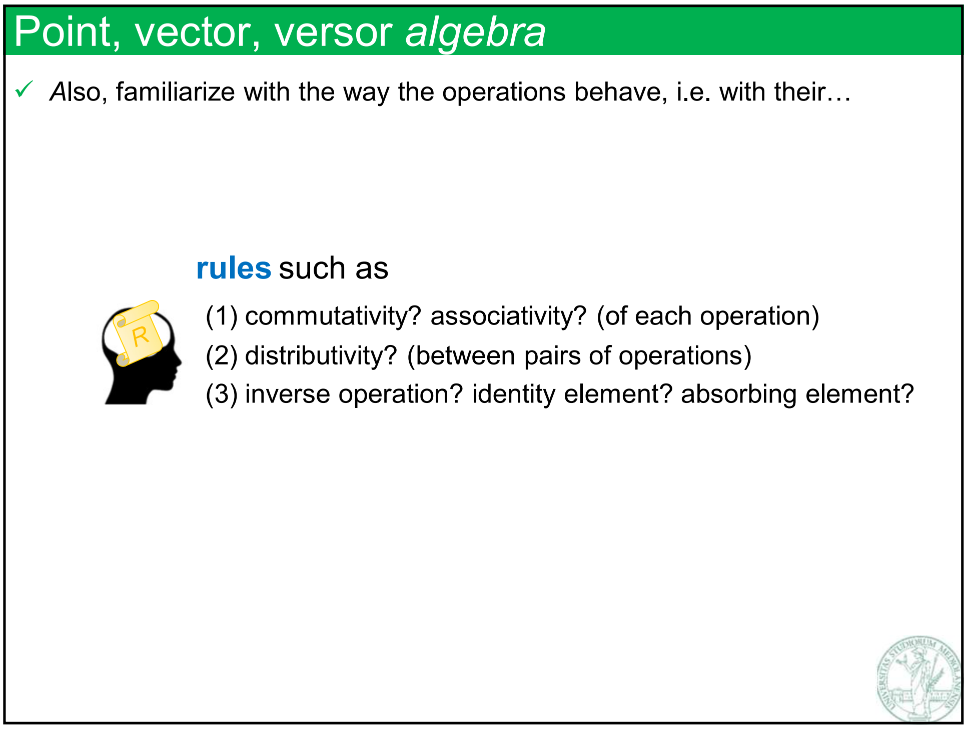
Points & vectors algebra

✓ In the following, make sure to know / understand each operation we will see in 3 ways:

intuitive / spatial: what does it do conceptually

operational: how to compute the result
(1) starting from the coordinates of the operand(s)
(2) (for products only) also, starting from the angles between the two operands, and their lengths

syntactic: how to write them down
(1) on paper (math-notation)
(2) in a programming language (in a C++ library, GLSL…)



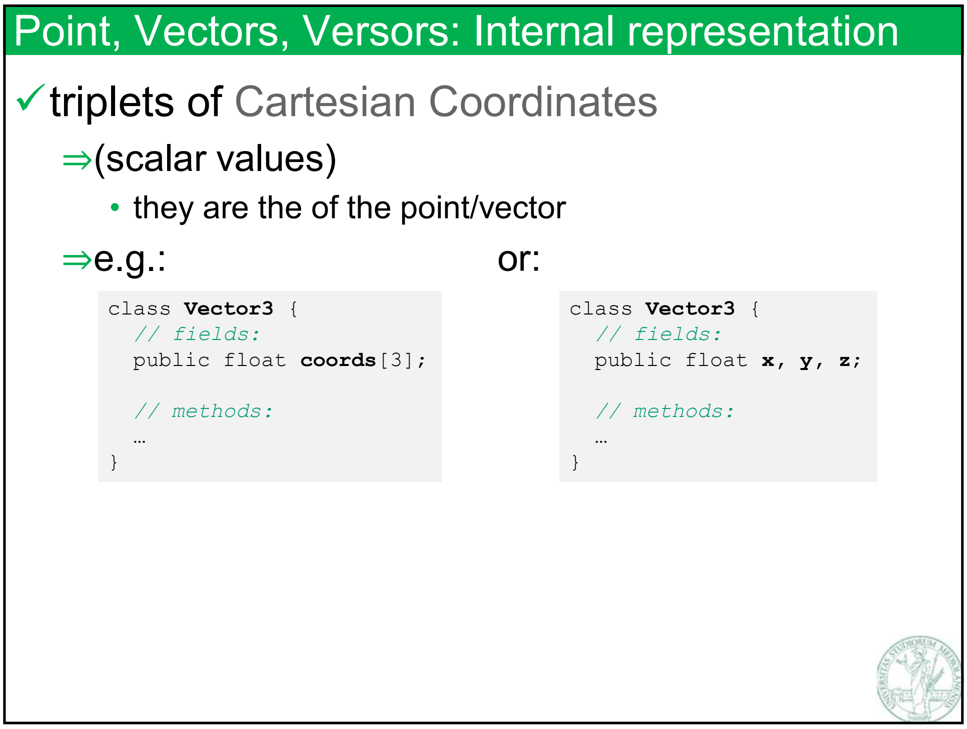
Point, vector, versor *algebra*

✓ *A*lso, familiarize with the way the operations behave, i.e. with their…

rules such as

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

8



Point, Vectors, Versors: Internal representation

✓triplets of Cartesian Coordinates

⇒(scalar values)

- they are the of the point/vector

⇒e.g.:

```
class Vector3 {
  // fields:
  public float coords[3];

  // methods:
  …
}
```

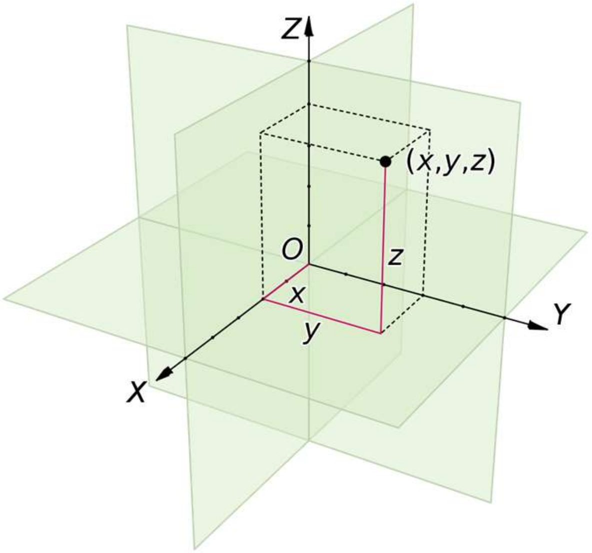
or:

```
class Vector3 {
  // fields:
  public float x, y, z;

  // methods:
  …
}
```

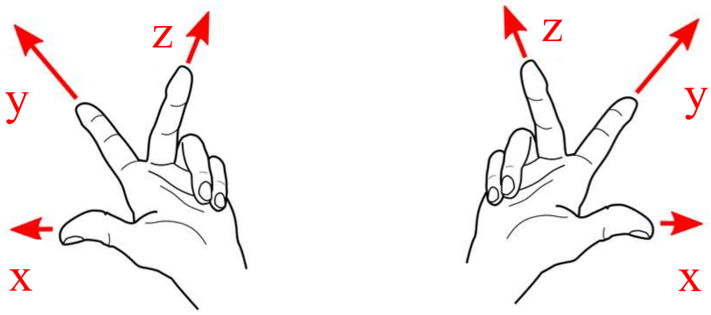
9

Expressed in a ("Cartesian") Coordinate System!



Right-handed or left-handed coord. system?

✓Ambiguity: how are we *imagining* things?



Caveat: one data type, multiple semantics

✓ Even if, often, libraries/languages choose to use the same data type ("vec3d", "Point3D", "vector" etc) for 3D points & 3D vectors (&… 3D versors, colors, etc) they are not the same thing,

⇒that's nothing new!
we use the same scalar data types ("float", "doubles") for widely different things (e.g. weight, or volume, or temperature).

✓ They important thing is to operate on them accordingly.

⇒e.g.: not ok to sum a temperature with a surface

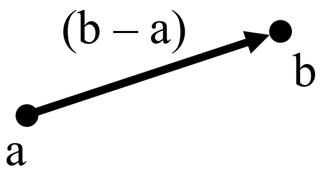
⇒e.g.: ok to divide a weight for a volume (and get a specific weight)

✓ which operations make sense on points, vectors, versors?

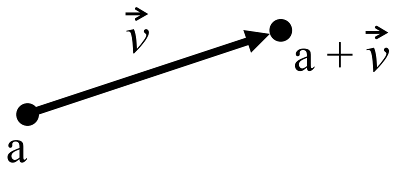
⇒that is, what about their *algebra* ?

Point and vector algebra (summary)

✓ Difference:
point – point = vector

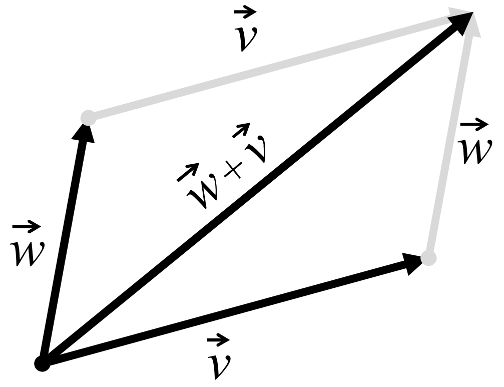


✓ Addition:
point + vector = point



Vector algebra: operation between vectors

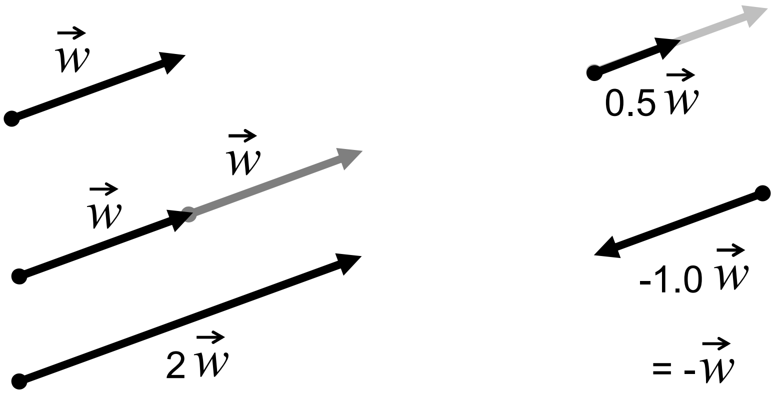
✓ addition (between vectors):
vector + vector = vector



14

Vector algebra: operation between vectors

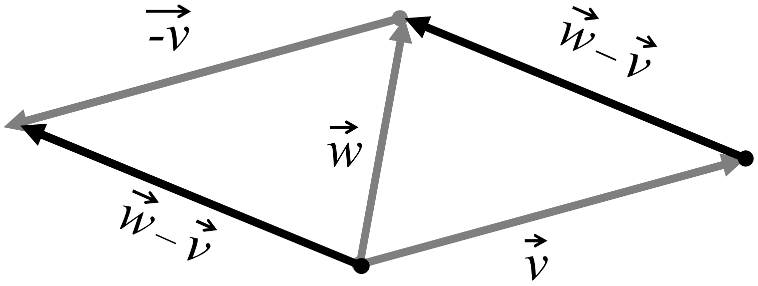
✓ product of a vector with a scalar:
scalar · vector = vector



15

Vector algebra: operation between vectors

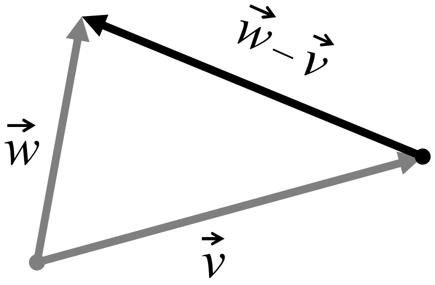
✓ difference (between vectors):
vector - vector = vector



16

Vector algebra: operation between vectors

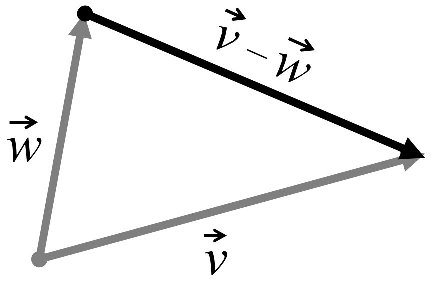
✓ difference (between vectors):
vector - vector = vector



17

Vector algebra: operation between vectors

✓ difference (between vectors):
vector - vector = vector



18

Vector algebra: operation between vectors

Linear operations :

✓ addition (between vectors):
vector + vector = vector

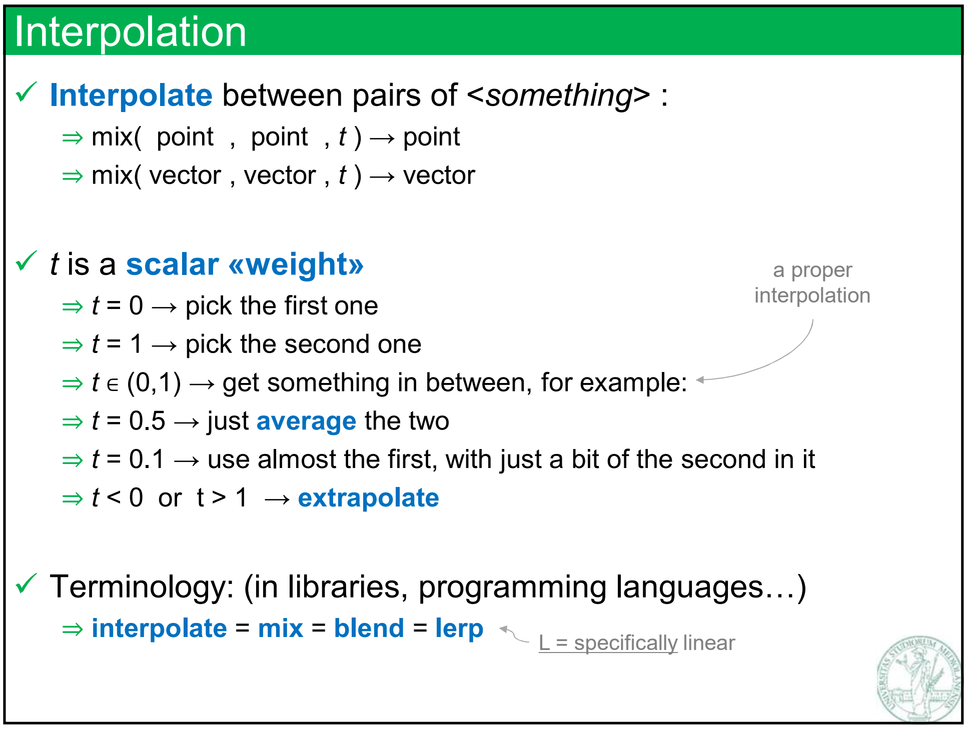
✓ product with a scalar:
scalar · vector = vector

⇒therefore: interpolation (between vectors)

✓ opposite (flip verse):

⇒therefore: difference (between vectors)

19



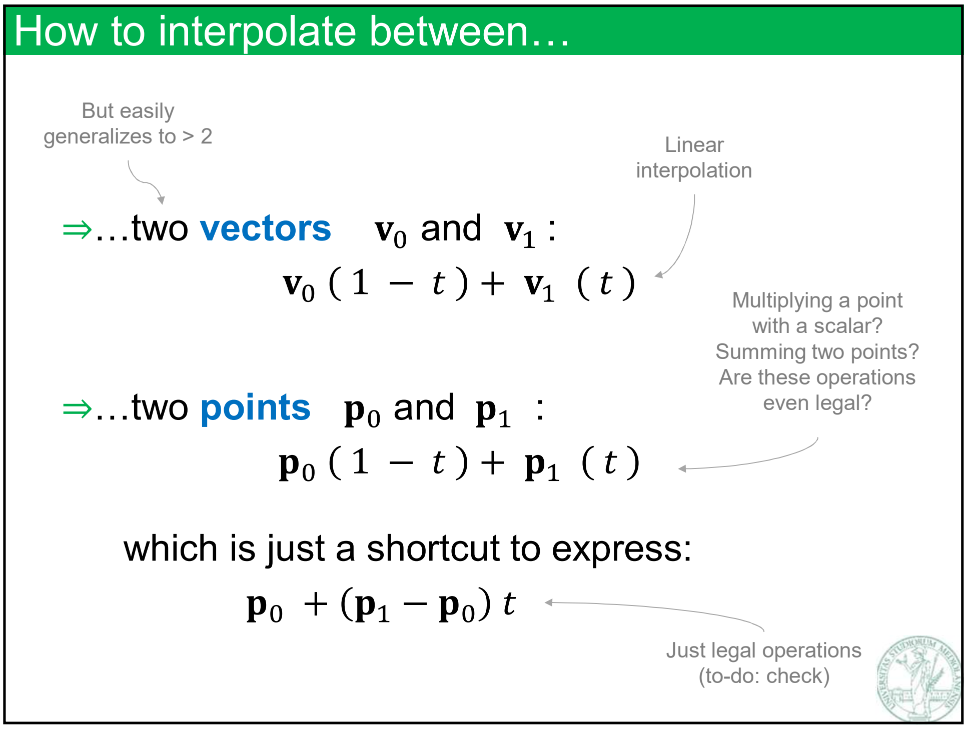
Interpolation

- ✓ **Interpolate** between pairs of <*something*> :
  - ⇒ mix(point , point , *t*) → point
  - ⇒ mix(vector , vector , *t*) → vector
- ✓ *t* is a **scalar «weight»**
  - ⇒ *t* = 0 → pick the first one
  - ⇒ *t* = 1 → pick the second one
  - ⇒ *t* ∈ (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, programming languages…)
  - ⇒ **interpolate** = **mix** = **blend** = **lerp**

a proper interpolation

L = specifically linear

20



How to interpolate between…

But easily generalizes to > 2

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

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

Linear interpolation

⇒…two **points** $\mathbf{p}_0$ and $\mathbf{p}_1$:

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

Multiplying a point with a scalar? Summing two points? Are these operations even legal?

which is just a shortcut to express:

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

Just legal operations (to-do: check)

21