

3D videogames

# Points, Vectors, Versors, and Spatial Transforms

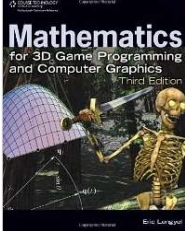



---




Marco Tarini

## This lecture




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

## Point, Vectors, Versors (unitary vectors)



- The base type, used by
  - rendering engine
  - physics engine
  - AI,
  - etc.
- The ubiquitous components in the data structures of 3D Assets

## Point, Vectors, Versors

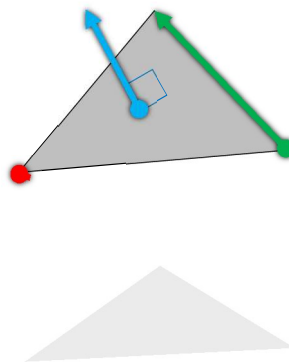


	represents:	example:	imagine it as...
<b>Point</b>	A position A location	Where a character is The center of a sphere	a small floating dot :-D
<b>Vector</b>	A displacement The difference between 2 points. The vector that connects them.	The velocity of a thrown knife The gravity acceleration How to reach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

## Points, Vectors, Versors in a 3D floating triangle

Examples of...

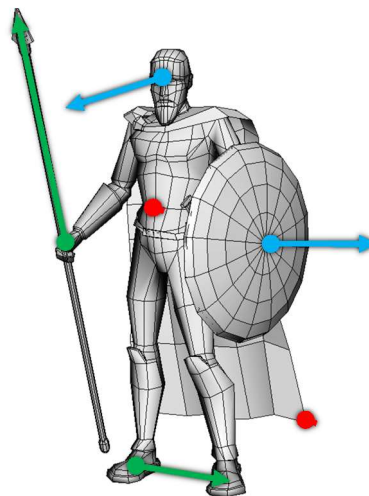
- **point:**
  - one vertex of the triangle
- **vector:**
  - one side of the triangle
- **versor:**
  - the «normal» of the triangle



## Points, Vectors, Versors in a character

Examples of...

- **points:**
  - the pos of the navel
  - the pos of lower-left tip of the hood
- **vectors:**
  - the vector connecting the L to the R foot
  - the vector from the hand to the tip of the lance
- **versors:**
  - the gaze direction
  - the facing of the shield

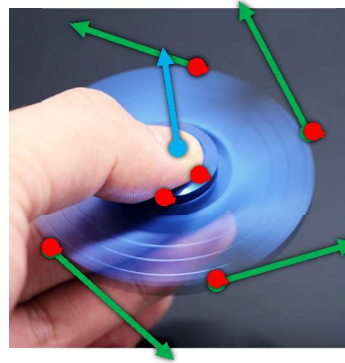


## Points, Vectors, Versors in a spinner

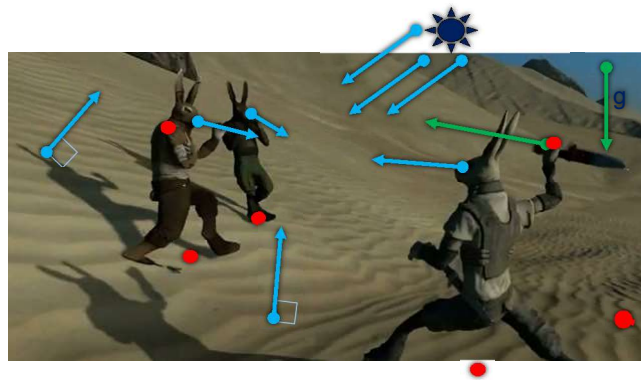


Examples of...

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



## Points, Vectors, Versors ...in this screenshot



## Points, Vectors, Versors: Internal representation



- $n$ -tuple of scalar values ( $n$  is the dimension)
  - with  $n = 3$  (rarely, 2 or 4)
  - they are the **Cartesian coordinates** of the point/vector

- e.g.:

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

  // methods:
  ...
}
```

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

  // methods:
  ...
}
```

- note: the same structure is often used for **points**, **vectors**, and **versors**

## Points, Vectors, Versors: Internal representation



- one class for **points**, **vectors**, and **versors**
- E.g. done by:



class Vector3

<https://docs.unity3d.com/ScriptReference/Vector3.html>



class FVector

<http://api.unrealengine.com/INT/API/Runtime/Core/Math/FVector/>

- (and also by: GLSL, HLSL, Eigen, GLM, ...)

## Caveat: one type, multiple semantics



- Even libraries/engines choose can opt to use the same **data type** for 3D points, 3D vectors, 3D versors, (plus, sometimes: colors, and more) they should be considered the same thing
  - that's nothing new:
    - we normally use the same scalar data types ("float", "doubles") with widely different semantics (e.g. weight, volume, temperature).
    - (alternatively, a library can use different types, e.g. Vector, Point, Versor)
- The important thing is only to *operate* on them accordingly
  - e.g.: not ok to **sum** a *temperature* with a *surface*
  - e.g.: ok to **divide** a *weight* by a *volume* (and get a *specific weight*)
- next: which operations make sense on points, vectors, versors?
  - that is, what about their *algebra* ?