Lecture 17

N-body Simulation

- model
- how to represent
  a domain-specific
  problem in code

① Steps of motion

$$\overbrace{v'}^{\text{after}} = \overbrace{v + at}^{\text{before}}$$

$$\underset{\substack{\uparrow\\\text{position}\\\text{after}\\\text{time}}}{s'} = \underset{\substack{\uparrow\\\text{initial}\\\text{pos}}}{s} + \underset{\substack{\uparrow\\\text{velocity}}}{v}\overset{\substack{\text{time}\\\swarrow}}{t} + \frac{1}{2}\underset{\substack{\uparrow\\\text{acceleration}}}{a}t^2$$

$$a' =$$

Newtons 2nd law:

$$F = m a$$

force

mass

accel.

$$a = F/m$$

accel of that planet

force on a planet

mass of that planet

Newtons Law of Gravitation

$$F_i = \sum_{j \neq i} F_{ij}$$

Forces are additive

force on body i

some constant

mass of i

mass of j

$$F_{ij} = \frac{G m_i m_j}{r_{ij}^2}$$

force on body i due to body j

distance between them (squared)

body = mass * position * velocity

from those compute accelerations

new mass/pos/velocity

$$a_i = \frac{F_i}{m_i}$$

↳ accel on body i

$$= \frac{\sum_j F_{ij}}{m_i}$$

$$= \sum_j a_{ij}$$

↳ accel on i due to j

$$= \sum_j \frac{\frac{G m_i m_j}{(r_{ij})^2}}{m_i}$$

$$= \sum_j \frac{G m_j}{(r_{ij})^2}$$

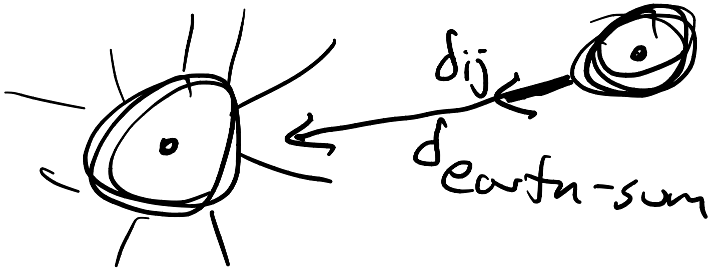
magnitude of acceleration

↳ acceleration vector (also direction)

$$\vec{a}_i = \sum_j \overset{\boxed{\text{direction}}}{\hat{d}_{ij}} * \overset{\text{as well as magnitude of accel}}{\left(\frac{G m_j}{|d_{ij}|^2} \right)}$$

d_{ij} is the vector from i to j

$\hat{v}$ is the unit vector from i to j



2 dimensions

→ make a library of functions for working with 2D vectors

$$\vec{a_i} = \sum_{j} \hat{d_{ij}} * \left(\frac{G m_j}{|d_{ij}|^2} \right)$$

① Summing of vectors

++

② unit vector

unitVec

③ Scaling a vector

**

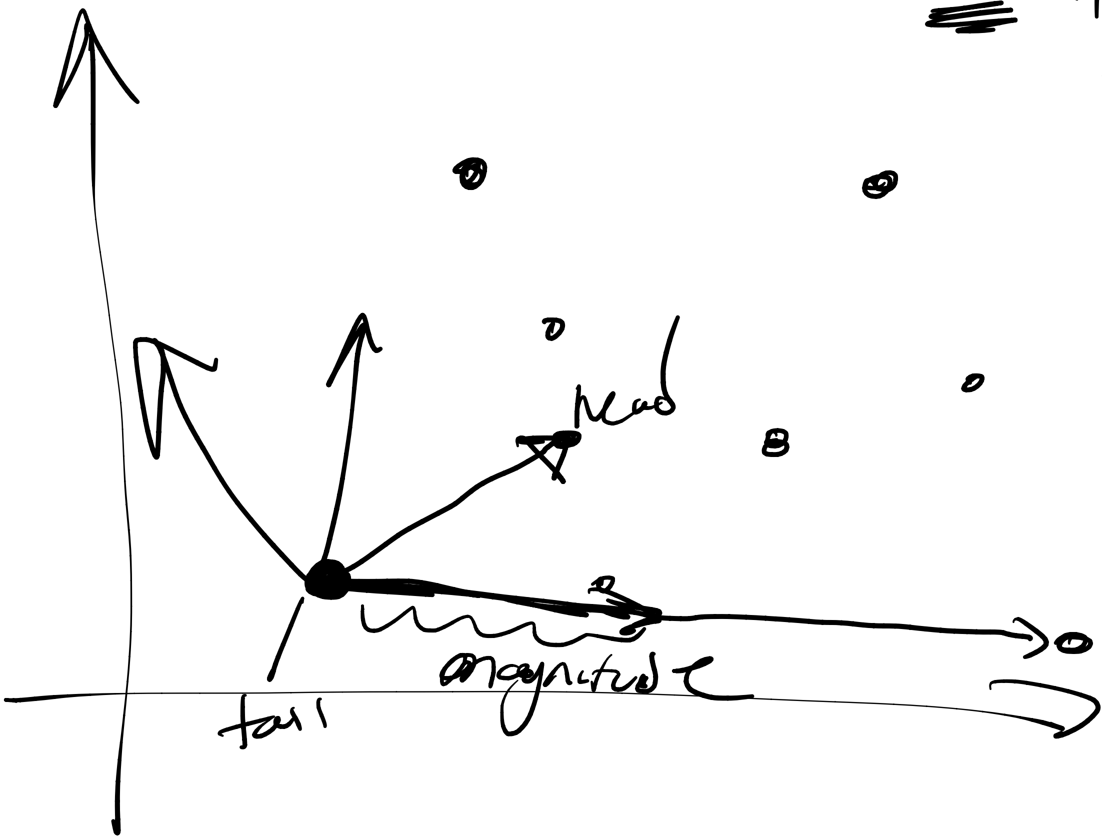
④ magnitude of a vector

mag

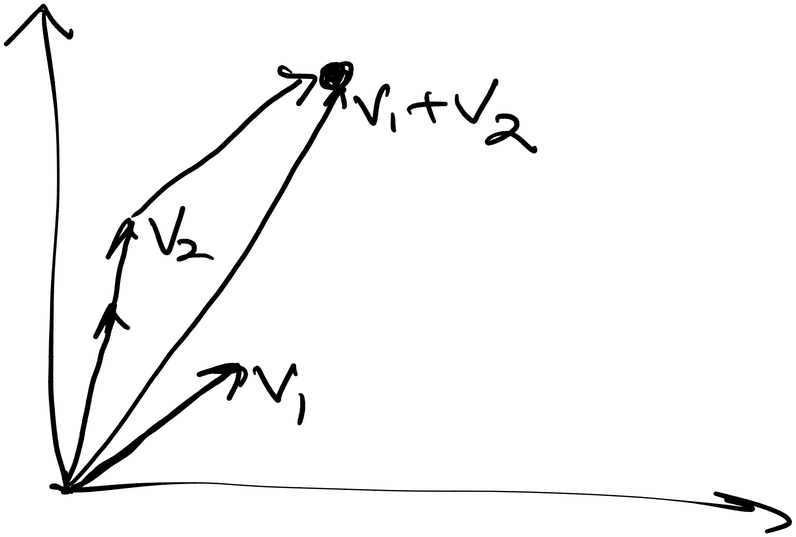
⑤ vector between 2 points

positions

```
type point = real * real   (* x and y
                              coords *)
type vec = real * real (* location of
                          the head
                          if tail is
                          at (0,0)
                          *)
```



Adding vectors

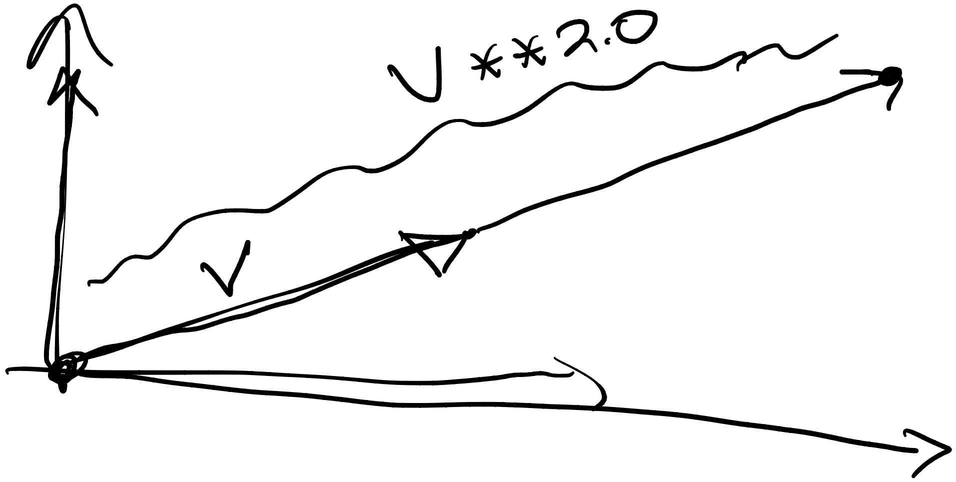


```
[infix ++]
fun((x1,y1):vec)++((x2,y2):vec) : vec = (x1+x2 , y1+y2)
```

Scaling

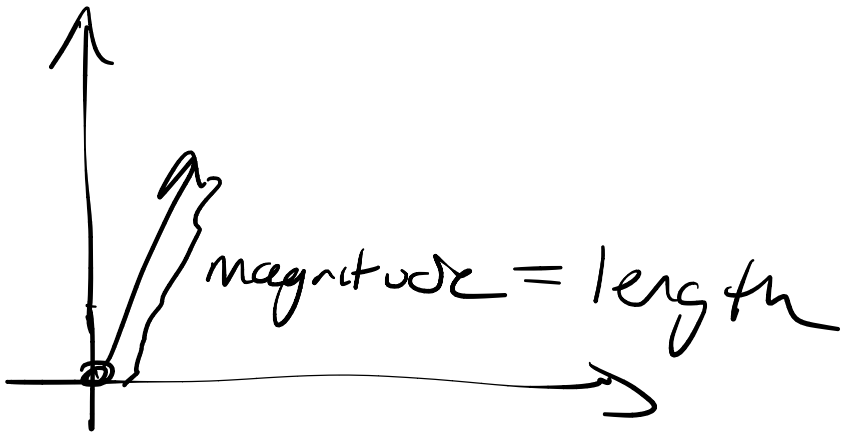
vector

"scalar" ⟶ real



```
[infix **]
fun ((x,y):vec) ** (s:real):vec =
        (s*x, s*y)
```

Magnitude

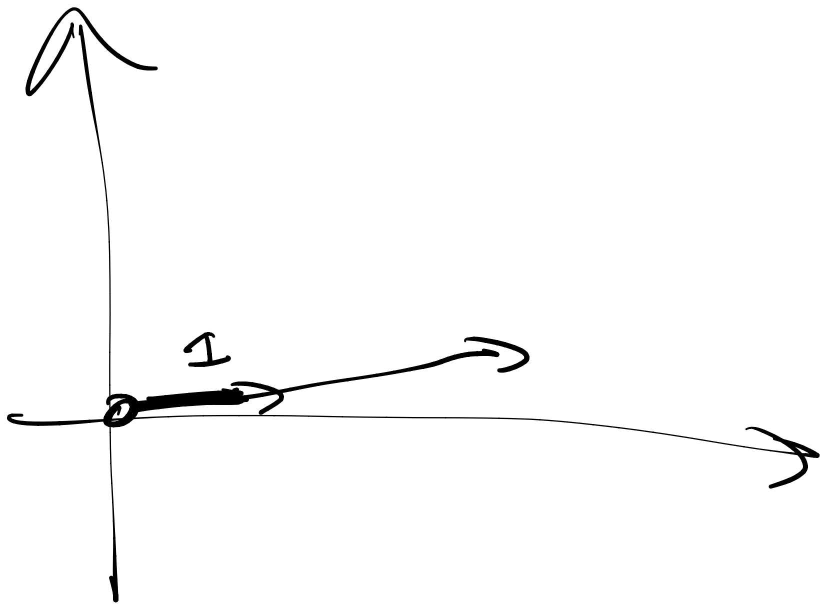


```
fun  mag( (x,y): vec): real =
     Math.sqrt( x*x + y*y)
```

$\sqrt{}$

Unit vector:

vector
in the
same dir
of magnitude 1

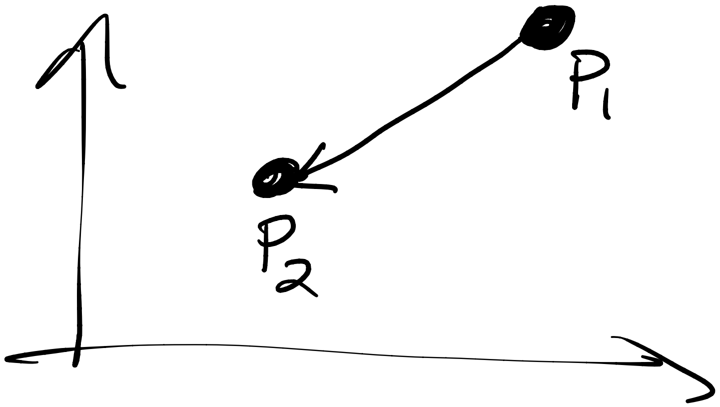


```
fun unitVec(v:vec): vec =
    v ** (1.0/mag v)
```

```
type point
type vec
val ++ : vec * vec -> vec
val ** : vec * real -> vec
val mag : vec -> real
val unitVec : vec -> vec
val --> : point * point -> vec
val equal : point * point -> bool
val zero : vec
```

fun $((x_1, y_1) : \text{point})$ --> $((x_2, y_2) : \text{point}) : \text{vec} =$

$$\left(\underline{\;x_2 - x_1\;} \,,\; \underline{\;y_2 - y_1\;} \right)$$



fun equal ((x_1, y_1):point, (x_2, y_2):point):bool =

Real.==(x_1, x_2)

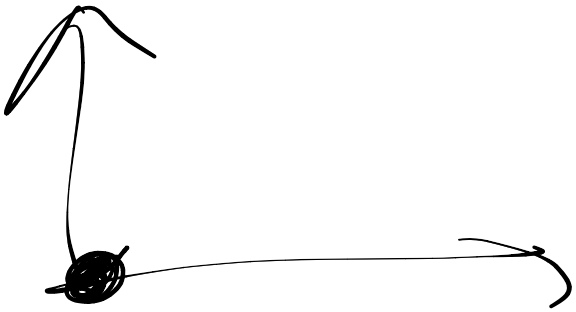
Real.==(y_1, y_2)

$\boxed{\text{No } x_1 = x_2}$

better:

$|x_1 - x_2| \leq \varepsilon$

```
val zero: Vec = (0.0, 0.0)
```



```
type point
type vec
val ++: vec*vec -> vec
val **: vec * real -> vec
val mag: vec -> real
val unitVec: vec -> vec
val --> : point*point -> vec
val equal : point*point -> bool
val zero: vec
```

$$a = \langle \vec{a}_1, \vec{a}_2, \ldots \vec{a}_n \rangle$$

$$\vec{a}_i = \sum_j \vec{a}_{ij}$$

$$\vec{a}_{ij} = \hat{d}_{ij} * \left(\frac{G m_j}{|d_{ij}|^2} \right)$$

```
type body = real * point * vec (mass / pos / velocity)

fun accOn((mi, pi, vi):body, (mj, pj, vj):body) : vec =
case equal(pi, pj) of
   true => zero
 | false => let val dij = (pi --> pj)
            in
              unitVec(dij) ** (G*mj / ((mag dij) * (mag dij)))
            end
```

Given

$\langle b_1, b_2, b_3, b_4 \ldots b_n \rangle$

make

$a = \langle \vec{a}_1, \vec{a}_2, \ldots \vec{a}_n \rangle$

$\vec{a}_i = \sum_j \vec{a_{ij}}$

```
fun accelerations(bodies: body Seq.seq): vec Seq.seq =
  Seq.map(fn body i =>
            Seq.reduce( fn(x,y) => x ++ y,
                        zero
                      )
            Seq.map(fn body j => accOn(body i,
                                       body j), bodies)
          bodies)
```

$\sum_j \vec{a_{ij}} \longrightarrow$

a_{ij}

Math $\langle \sum_j \vec{a_{1j}}, \sum_j \vec{a_{2j}}, \sum_j \vec{a_{3j}} \ldots \rangle$

	input	work	span
each inner map	n	$O(1)$	$O(1)$
reduce	n	$O(n)$	$O(\log n)$
outer map	n	$O(n^2)$	$O(\log n)$ max of all

Barnes-Hut

approximate

octrees