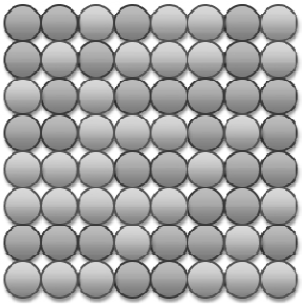


UNIT 12B
N-Body Simulation

Robots vs. Planets

- Differences between the robot simulation and a simulation of the solar system:
  - no preconceived notion of where planets are supposed to move
  - planets change velocity as they move

2-Body System

- Imagine a watermelon dropped from a high point.
- Assuming the melon does not slow down due to wind resistance, after *t* seconds, the melon will fall a distance *d* given by

  $d = 0.5gt^2$ meters

  where *g* is the acceleration due to gravity. ($g = 9.8$ m/sec^2)
- The amount of time *t* it takes to fall a distance *d* meters is:

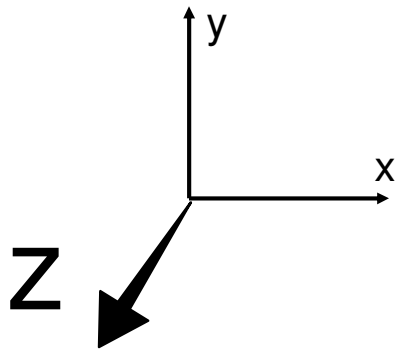
  $t = (2d/g)^{1/2}$ seconds

RubyLabs

```
include SphereLab
b = make_system(:melon)
view_melon(b)                    display world
position_melon(b, 50)
update_melon(b, 0.5)             move melon once
position_melon(b, 100)
drop_melon(b, 0.5)               move until splat
position_melon(b, 100)
drop_melon(b, 0.1)               better precision
position_melon(b, 100)
b[0].velocity.x = 5
drop_melon(b, 0.1)               push and drop
```

Forces

- Objects can move in three dimensions.

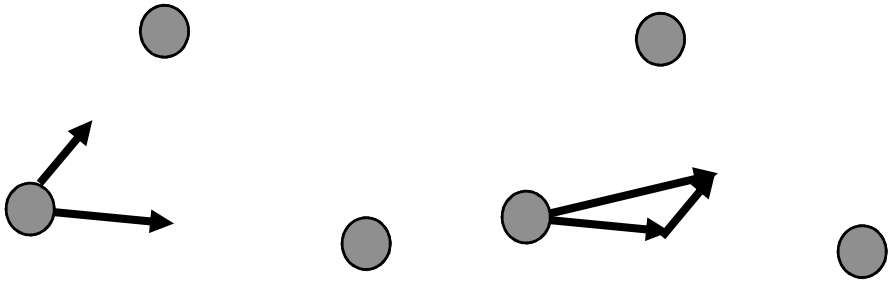


Forces (cont'd)

- Newton's law of universal gravitation:
  $F \propto m_1 m_2 / d^2$

Forces (cont'd)

- Forces that act on a body are additive.



RubyLabs

```
b = make_system(:fdemo)
view_system(b, :pendown => :track)
50.times {
  update_one(b[0], b[1..5], 1.0)
}
```

NOTE: In this simulation, body b[0] is moving but bodies b[1]...b[5] are stationary.

N-body Simulation

- To simulate N-bodies in motion affected by the universal law of gravitation, for each time step we need to compute the effect of every pair of bodies on each other and sum the resulting vectors for each body.
- The force pulling body i to body j is the same as the force pulling body j to body i, so the force between two bodies is only calculated once (not twice).
- How many body interactions need to be computed per time step if there are n bodies?

N-Body Simulation Algorithm

```
def step_system(bodies, dt)
  nb = bodies.length
  for i in 0..(nb-2) do
      for j in (i+1)..(nb-1) do
            Body.interaction(bodies[i],bodies[j])
      end
  end
  for b in bodies do
      b.move(dt); b.clear_force
  end
end
```

RubyLabs

```
b = make_system(:solarsystem)
view_system(b)
view_system(b[0..4], :dash => 1)
365.times {
  update_system(b, 86459)
}
```

86459 = the number of seconds in one day

Simulation

- Simulation is a computational principle that allows us to compute and visualize a complex process that would be difficult or impossible to create in real life.
- Simulation can be time-based or event-based, and it can grid-based or agent-based.