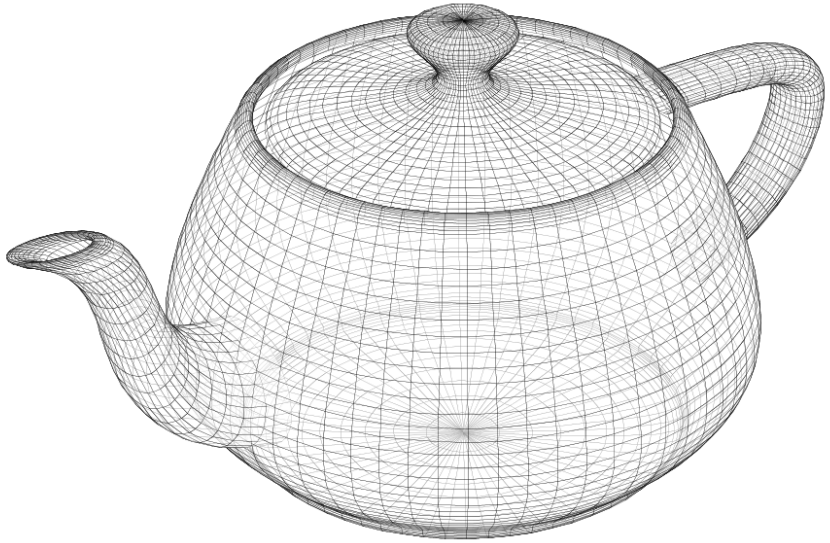


A Simple Physics Engine

Interactive Computer Graphics

Professor Eric Shaffer

Newtonian Physics

- We will animate particles (aka point masses)
- Position is changed by velocity
- Velocity is changed by acceleration
- Forces alter acceleration

- Our physics engine will integrate to compute
  - Position
  - Velocity
- We set the acceleration by applying forces

ILLINOIS

Force and Mass and Acceleration

- How do we update acceleration when force is applied?
- To find the acceleration due to a force we have

$$\ddot{\mathbf{p}} = \frac{1}{m}\mathbf{f}$$

- So we need to know the inverse mass of the particle
  - You can model infinite mass objects by setting this value to 0

- For the MP, you can use a uniform mass of 1
  - Or make the masses different if you want…

ILLINOIS

Force: Gravity

- Law of Universal Gravitation

$$f = G\frac{m_1 m_2}{r^2}$$

- G is a universal constant
- m_i is the mass of an object
- r is the distance between object centers
- if we care only about gravity of the Earth
  - m1 and r are constants
  - r is about 6400 km on Earth
- We simplify to f = mg
  - g is about $10 ms^{-2}$

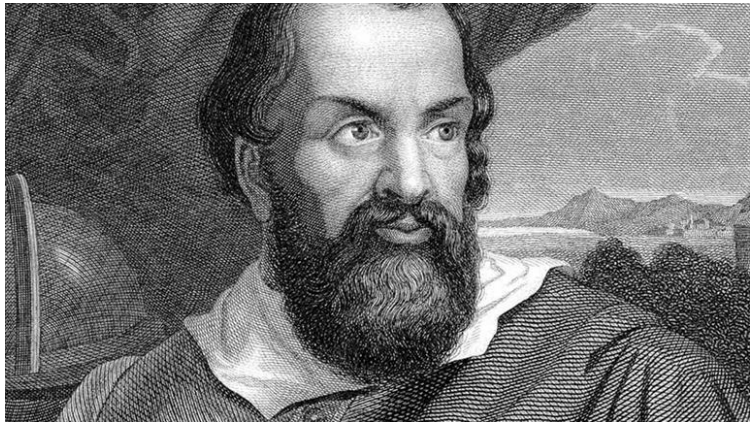
ILLINOIS

Acceleration due to Gravity

- If we consider acceleration due to gravity we have

$$\ddot{p} = \frac{1}{m}(mg) = g$$

- So acceleration due to gravity is independent of mass



Acceleration due to Gravity

- In your MP the magnitude and direction of acceleration would be

$$\mathbf{g} = \langle 0, -g, 0 \rangle$$

- For gaming, $10ms^{-2}$ tends to look boring
  - Shooters often use $15ms^{-2}$
  - Driving games often use $20ms^{-2}$
  - Some tune g object-by-object

ILLINOIS

Force: Drag

- Drag dampens velocity
  - Caused by friction with the medium the object moves through
- Even neglecting drag, you need to dampen velocity
  - Otherwise numerical errors likely drive it higher than it should be
- A velocity update with drag can be implemented as

$$\dot{\mathbf{p}}_{new} = \dot{\mathbf{p}} d^t$$

  - important to incorporate time so drag changes if the frame rate varies
  - for the MP, have all objects have the same drag, calculate once per frame
- What range should ***d*** be in? $(0, 1)$

The Integrator

- The position update can found using Euler's Method:

$$\mathbf{P_{new}} = \mathbf{P_{old}} + \dot{\mathbf{P}}\mathbf{t}$$

- This is a pretty inaccurate approximation of analytical integration
  - formula gets more inaccurate as acceleration gets larger
    - why?
  - In general we can characterize Euler method error as O(t)
  - …good enough for the MP
- The velocity update is computed using Euler integration as well

$$\dot{\mathbf{p}}_{new} = \dot{\mathbf{p}}d^t + \ddot{\mathbf{p}}t$$

ILLINOIS

The Integrator

- You should ideally use actual time for *t*
  - or some scaled version of it
- In JavaScript, Date.now() returns current time in ms
  - so keep a previous time variable
  - each frame find out how much time has elapsesd
- …or you could use some uniform timestep you like