Physics based animation

Lecture 04 - dynamics

Grégory Leplâtre

g.leplatre@napier.ac.uk, room D32
School of Computing
Edinburgh Napier University

Semester 1 - 2016/2017

Objectives

- Newtonian physics
- Application to particles
- Numerical integration

Outline

1 Physics
2 Forces
3 Integration
4 Summary

Particle kinematics

particle position $\boldsymbol{r}(t)$ defined by:

- $\boldsymbol{r} = \boldsymbol{r}(t)$
- $\boldsymbol{v} = \dfrac{d\boldsymbol{r}}{dt}$
- $\boldsymbol{a} = \dfrac{d\boldsymbol{v}}{dt} = \dfrac{d^2\boldsymbol{r}}{dt}$

Particle kinematics

particle position $\boldsymbol{r}(t)$ defined by:

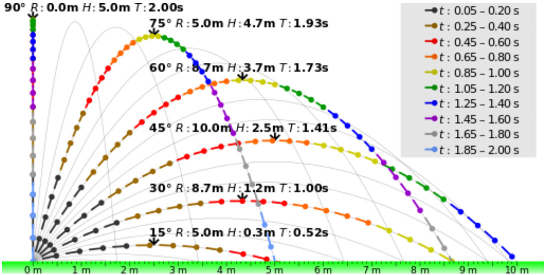
- $\boldsymbol{r} = \boldsymbol{r}(t)$
- $\boldsymbol{v} = \dfrac{d\boldsymbol{r}}{dt}$
- $\boldsymbol{a} = \dfrac{d\boldsymbol{v}}{dt} = \dfrac{d^2\boldsymbol{r}}{dt}$

For constant acceleration:

- $\boldsymbol{a} = \boldsymbol{a_0}$
- $\boldsymbol{v} = \displaystyle\int \boldsymbol{a}dt = \boldsymbol{v_0} + \boldsymbol{a_0}t$
- $\boldsymbol{r} = \displaystyle\int \boldsymbol{v}dt = \boldsymbol{r_0} + \boldsymbol{v_0}t + \frac{1}{2}\boldsymbol{a_0}t^2$

Particle kinematics

- Parabolic trajectory (in x, y and z)
- Fully defined by $\boldsymbol{r}_0$, $\boldsymbol{v}_0$ and $\boldsymbol{a}_0$



Particle Physics

Let's introduce a mass m for each particle.

- **momentum** defined as:

$$\boldsymbol{p} = m\boldsymbol{v}$$

Particle Physics

Let's introduce a mass m for each particle.

- **momentum** defined as:

$$\boldsymbol{p} = m\boldsymbol{v}$$

- **force** is defined as the rate of change of momentum:

$$\boldsymbol{f} = \frac{d\boldsymbol{p}}{dt}$$

$$\boldsymbol{f} = m\boldsymbol{a}$$

Particle Physics

Reminder: Newton's Laws.

1. An object remains at rest or continues to travel with constant velocity unless acted upon by a force.

Particle Physics

Reminder: Newton's Laws.

1. An object remains at rest or continues to travel with constant velocity unless acted upon by a force.
2. $\boldsymbol{f} = m\boldsymbol{a}$

Particle Physics

Reminder: Newton's Laws.

1. An object remains at rest or continues to travel with constant velocity unless acted upon by a force.
2. $\boldsymbol{f} = m\boldsymbol{a}$
3. $\boldsymbol{f}_{AB} = -\boldsymbol{f}_{BA}$

Implication:

- (2) and (3) $\Rightarrow$ conservation of momentum

Particle simulation

1. Compute all **forces** acting within the system.

Particle simulation

1. Compute all **forces** acting within the system.
2. Compute **acceleration** for each particle

Particle simulation

1. Compute all **forces** acting within the system.
2. Compute **acceleration** for each particle
3. **Integrate** to calculate the **position** of each particle

Outline

1 Physics

2 Forces

3 Integration

4 Summary

Gravity

Uniform gravity field:

$$\boldsymbol{f}_{gravity} = m\boldsymbol{g}_0$$

$$\boldsymbol{g}_0 = [0, -9.8, 0] \; m.s^{-2}$$

Gravity

Gravitational force:

$$\mathbf{f}_{gravity} = \frac{Gm_1 m_2}{d^2}\mathbf{e}$$

Where:

- $G = 6.673 \times 10^{-11}\ m^3/kg.s^2$
- $\mathbf{e} = \frac{\mathbf{r_1} - \mathbf{r_2}}{\|\mathbf{r_1} - \mathbf{r_2}\|}$
- d is the distance between the objects, $\mathbf{r}_i$ and m_i, their position and mass, respectively.

Gravity

Gravitational force:

$$\boldsymbol{f}_{gravity} = \frac{Gm_1 m_2}{d^2}\boldsymbol{e}$$

Where:

- $G = 6.673 \times 10^{-11}\ m^3/kg.s^2$
- $\boldsymbol{e} = \dfrac{\boldsymbol{r_1} - \boldsymbol{r_2}}{\|\boldsymbol{r_1} - \boldsymbol{r_2}\|}$
- d is the distance between the objects, $\boldsymbol{r}_i$ and m_i, their position and mass, respectively.

Note: Gravitational forces computation for all N^2 particles in a system: N^2 calculations.

Aerodynamic drag

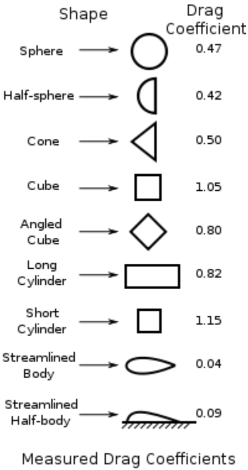
Common approximation of aerodynamic drag:

$$\boldsymbol{f}_{aero} = \frac{1}{2}\rho\|\boldsymbol{v}\|^2 c_d a \boldsymbol{e}$$

Where:

- ρ is the density of the medium (air, water, etc)
- c_d is the coefficient of drag of the object
- a is the cross sectional area of the object.
- $\boldsymbol{e} = -\frac{\vec{v}}{\|\vec{v}\|}$

Aerodynamic drag



- **Coefficient of drag** isn't constant
- but can be **approximated** by a constant (see tables)

Aerodynamic drag

Liquid Units	Viscosity [Pa·s]	Viscosity [cP=mPa·s]
liquid nitrogen @ 77K	1.58×10^{-4}	0.158
acetone	3.06×10^{-4}	0.306
methanol	5.44×10^{-4}	0.544
benzene	6.04×10^{-4}	0.604
water	8.94×10^{-4}	0.894
ethanol	1.074×10^{-3}	1.074
mercury	1.526×10^{-3}	1.526
nitrobenzene	1.863×10^{-3}	1.863
propanal	1.945×10^{-3}	1.945
pitch	2.3×108	2.3×1011
ethylene glycol	1.61×10^{-2}	16.1
sulfuric acid	2.42×10^{-2}	24.2
motor oil SAE 10 (20 °C)	0.065	65
olive oil	0.081	81
motor oil SAE 40 (20 °C)	0.319	319
castor oil	0.985	985
glycerol (at 20 °C)	1.2	1200
corn syrup	1.3806	1380.6
HFO-380	2.022	2022

- idem for **viscosity**

Aerodynamic drag

- By extension, the velocity of the air (wind) can be simulated using aerodynamic drag
- Using this approach, one can define useful turbulence fields, vortices, and other flow patterns

Springs

Simple spring force:

$$\boldsymbol{f}_{spring} = -k_s \boldsymbol{x}$$

Where:

- k_s is a constant describing the **stiffness** of the spring
- $\boldsymbol{x}$ represents the displacement

Dampers

Damping force between particles:

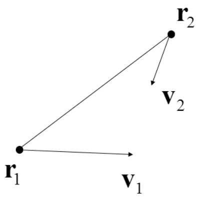
$$\boldsymbol{f}_{damp} = -k_d \boldsymbol{v}$$

- The damping forces are equal and opposite, so they conserve momentum, but they will remove energy from the system
- In real dampers, kinetic energy of motion is converted into complex fluid motion within the damper and then diffused into random molecular motion causing an increase in temperature. The kinetic energy is effectively lost.
- Dampers can be combined with springs as **spring-damper** objects.

Dampers

- To compute the damping force, we need to know the *closing speed* i.e., the speed at which the particles are approaching each other.

Dampers

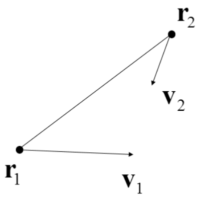


- **Analytical approach**:

$$\boldsymbol{v} = \boldsymbol{v}_1 \bullet \boldsymbol{e} - \boldsymbol{v}_2 \bullet \boldsymbol{e}$$

Where:
$$\boldsymbol{e} = \frac{\boldsymbol{r}_1 - \boldsymbol{r}_2}{\|\boldsymbol{r}_1 - \boldsymbol{r}_2\|}$$

Dampers



- **Numerical approach**: compute the closing velocity by comparing the distance between two particles to their distance a frame earlier.

$$\boldsymbol{v} = \frac{\|\boldsymbol{r}_1 - \boldsymbol{r}_2\| - d_0}{\Delta t}$$

Dampers

which method is better?

- The analytical method is better:
  - Doesn't required extra storage
  - Easier to 'start' the simulation (data from previous frame not required)
  - The result is **not an approximation**, it is **exact**.
- The analytical method isn't computationally expensive.

Force fields

- Any force field can be defined as any arbitrary force applied to a particle in function of its position within the field:

$$\boldsymbol{f}_{field} = \boldsymbol{f}(\boldsymbol{r})$$

Collisions, Impulses and elasticity

- A collision is assumed to be **instantaneous**
- However, for a force to change an object's momentum, it must operate over some time interval
- Therefore, we can't use actual forces to do collisions

Collisions, Impulses and elasticity

- A collision is assumed to be **instantaneous**
- However, for a force to change an object's momentum, it must operate over some time interval
- Therefore, we can't use actual forces to do collisions
- Instead, we introduce the concept of an **impulse**, which can be though of as a large force acting over a small time

Impulse

- An impulse can be thought of as the integral of a force over some time range, which results in a finite change in momentum:

$$\boldsymbol{j} = \int \boldsymbol{f}\,dt = \Delta\boldsymbol{p}$$

- An impulse behaves a lot like a force, except instead of affecting an object's acceleration, it directly affects the velocity
- Impulses also obey Newton's Third Law, and so objects can exchange equal and opposite impulses
- Also, like forces, we can compute a total impulse as the sum of several individual impulses

Particle simulation - revised

1. Compute all **forces** acting within the system.

Particle simulation - revised

1. Compute all **forces and impulses** acting within the system.

Particle simulation - revised

1. Compute all **forces and impulses** acting within the system.

$$\boldsymbol{f} = \sum_i \boldsymbol{f}_i$$

$$\boldsymbol{j} = \sum_i \boldsymbol{j}_i$$

Particle simulation - revised

1. Compute all **forces and impulses** acting within the system.

$$\mathbf{f} = \sum_i \mathbf{f}_i$$

$$\mathbf{j} = \sum_i \mathbf{j}_i$$

2. Compute **acceleration** for each particle

Particle simulation - revised

1. Compute all **forces and impulses** acting within the system.

$$\mathbf{f} = \sum_i \mathbf{f}_i$$

$$\mathbf{j} = \sum_i \mathbf{j}_i$$

2. Compute **acceleration** for each particle
3. **Integrate** to calculate the **position** of each particle

Particle simulation - revised

1. Compute all **forces and impulses** acting within the system.

$$\boldsymbol{f} = \sum_i \boldsymbol{f}_i$$

$$\boldsymbol{j} = \sum_i \boldsymbol{j}_i$$

2. Compute **acceleration** for each particle
3. **Integrate** to calculate the **velocity and position** of each particle

Particle simulation - revised

1. Compute all **forces and impulses** acting within the system.

$$\boldsymbol{f} = \sum_i \boldsymbol{f}_i$$

$$\boldsymbol{j} = \sum_i \boldsymbol{j}_i$$

2. Compute **acceleration** for each particle
3. **Integrate** to calculate the **velocity and position** of each particle

$$\boldsymbol{v}' = \boldsymbol{v}_0 + \frac{1}{m}(\boldsymbol{f}\Delta t + \boldsymbol{j})$$

$$\boldsymbol{r}' = \boldsymbol{r}_0 + \boldsymbol{v}'\Delta t$$

Elasticity

- **Problem**: How to find the collision impulse $\boldsymbol{j}$ applied to a particle during the collision with a **static object**

Elasticity

- **Problem**: How to find the collision impulse $\boldsymbol{j}$ applied to a particle during the collision with a **static object**
- Simple solution: *Elasticity* (ranges from 0 to 1):
  - elasticity of 0 $\Rightarrow$ closing velocity after collision is 0
  - elasticity of 1 $\Rightarrow$ closing velocity after collision is exact opposite of closing velocity before collision.

Elasticity

- **Problem**: How to find the collision impulse $\boldsymbol{j}$ applied to a particle during the collision with a **static object**
- Simple solution: *Elasticity* (ranges from 0 to 1):
  - elasticity of 0 $\Rightarrow$ closing velocity after collision is 0
  - elasticity of 1 $\Rightarrow$ closing velocity after collision is exact opposite of closing velocity before collision.
- For a collision with no friction:

$$\boldsymbol{j} = -(1 + e)mv_{close}\boldsymbol{n}$$

Where:

  - $\boldsymbol{v}_{close} = \boldsymbol{v} \bullet \boldsymbol{n}$
  - e is the elasticity

Outline

1 Physics
2 Forces
3 Integration
4 Summary

Forward (explicit) Euler integration

Problem: solving two Ordinary Differential Equations (ODEs)

- $\mathbf{v} = \frac{d\mathbf{r}}{dt}$
- $\mathbf{a} = \frac{d\mathbf{v}}{dt} = \frac{d^2\mathbf{r}}{dt}$

Forward (explicit) Euler integration

Problem: solving two Ordinary Differential Equations (ODEs)

- $\boldsymbol{v} = \frac{d\boldsymbol{r}}{dt}$
- $\boldsymbol{a} = \frac{d\boldsymbol{v}}{dt} = \frac{d^2\boldsymbol{r}}{dt}$

Let's formulate the general problem as:

$$\frac{dy}{dt} = f(y, t)$$

Forward (explicit) Euler integration

Taylor expansion:

$$y(t+h) = y(t) + hy'(t) + \frac{1}{2!}h^2 y''(t) + \frac{1}{3!}h^3 y^3(t) + \ldots$$

Forward (explicit) Euler integration

Taylor expansion:

$$y(t+h) = y(t) + hy'(t) + \frac{1}{2!}h^2y''(t) + \frac{1}{3!}h^3y^3(t) + \ldots$$

Truncated series:

$$y(t+h) = y(t) + hy'(t) + O(h^2)$$

Forward (explicit) Euler integration

Taylor expansion:

$$y(t+h) = y(t) + hy'(t) + \frac{1}{2!}h^2y''(t) + \frac{1}{3!}h^3y^3(t) + \ldots$$

Truncated series:

$$y(t+h) = y(t) + hy'(t) + O(h^2)$$

Which gives us:

$$y(t+h) \approx y(t) + hf(t)$$

Forward (explicit) Euler integration

Taylor expansion:

$$y(t+h) = y(t) + hy'(t) + \frac{1}{2!}h^2 y''(t) + \frac{1}{3!}h^3 y^3(t) + \ldots$$

Truncated series:

$$y(t+h) = y(t) + hy'(t) + O(h^2)$$

Which gives us:

$$y(t+h) \approx y(t) + hf(t)$$

Or:

$$y_{n+1} = y_n + hf(y_n, t_n)$$

Forward Euler integration

$$y_{n+1} = y_n + hf(y_n, t_n)$$

Forward Euler integration

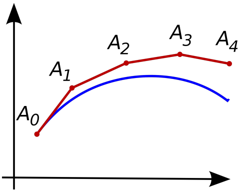
$$y_{n+1} = y_n + hf(y_n, t_n)$$

For our two ODEs, we get:

$$v_{n+1} = v_n + ha_n$$

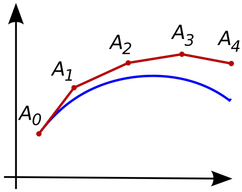
$$r_{n+1} = r_n + hv_n$$

Stability



- Stability is a crucial issue. Errors may lead to energy producing simulations

Stability



- Stability is a crucial issue. Errors may lead to energy producing simulations
- Local Truncation Error (LTE): $O(h^2)$ (first order method)
- ok for non-oscillating systems
- Adds energy!
- Stability requires small steps (h)

Backward (implicit) Euler integration

$$v_{n+1} = v_n + ha_{n+1}$$
$$r_{n+1} = r_n + hv_{n+1}$$

Backward (implicit) Euler integration

$$v_{n+1} = v_n + ha_{n+1}$$
$$r_{n+1} = r_n + hv_{n+1}$$

- Need to ‘predict’ future acceleration. In practice, this requires solving a non-linear equation, or using predictor-corrector solver
- Loses energy!
- more costly, but more stable.

Verlet integration

$$x_{n+1} = 2x_n - x_{n-1} + h^2 a_n$$

Verlet integration

$$x_{n+1} = 2x_n - x_{n-1} + h^2 a_n$$

- Very stable and cheap
- Problem if velocity needs to change (impulse).

Semi-implicit/simplectic euler integration

$$v_{n+1} = v_n + ha_n$$

$$r_{n+1} = r_n + hv_{n+1}$$

Semi-implicit/simplectic euler integration

$$v_{n+1} = v_n + ha_n$$
$$r_{n+1} = r_n + hv_{n+1}$$

- cheap and stable (with oscillating functions)
- But not as accurate as RK4

Runge-Kutta (4th order) integration

$$v_{n+1} = v_n + (k_1 + 2k_2 + 2k_3 + k_4)/6$$

Where:

$k_1 = ha(n, v_n)$
$k_2 = ha(n + h/2, v_n + k_1/2$
$k_3 = ha(n + h/2, v_n + k_2/2$
$k_4 = ha(n + h, v_n + k_3$

Runge-Kutta (4th order) integration

$$v_{n+1} = v_n + (k_1 + 2k_2 + 2k_3 + k_4)/6$$

Where:

$k_1 = ha(n, v_n)$
$k_2 = ha(n + h/2, v_n + k_1/2$
$k_3 = ha(n + h/2, v_n + k_2/2$
$k_4 = ha(n + h, v_n + k_3$

- Very stable and accurate
- Conserves energy well
- But expensive

Choice of integration method

- With simple forces, standard Euler might be okay
- But constraints, springs, etc. require stability
- Recommendation: Symplectic Euler
  - Generally stable
  - Simple to compute
- More complex integrators available

Outline

1 Physics

2 Forces

3 Integration

4 Summary

Summary

- Newtonian physics

Summary

- Newtonian physics
- Physics simulation:
  from forces and impulses $\Rightarrow$ Compute position

Summary

- Newtonian physics
- Physics simulation:
  from forces and impulses $\Rightarrow$ Compute position
- Integration methods to compute velocity and position

Coming up

- Practical: Physics tick and integration methods
- No lecture on Wednesday (only tutorial)
- Next week: more about particles ...