Computer Animation
Algorithms and Techniques

Physically Based Animation
Physics Review:
force, mass, acceleration
velocity, position

\[ f = ma \]
\[ a = \frac{f}{m} \]
\[ v' = v + a \cdot \Delta t \]
\[ p' = p + \frac{(v + v') \Delta t}{2} = p + v \Delta t + \frac{1}{2} a \Delta t^2 \]
$F = \frac{Gm_1m_2}{d^2}$

$F = \frac{Gm_{\text{earth}}m_2}{d_{\text{earth-radius}}^2}$

$a = \frac{F}{m_2} = 9.8 \text{ m/s}^2$
Physics Review: Other forces

\[ f_{st} = k_{st} f_N \quad \text{Static friction} \]

\[ f_k = k_k f_N \quad \text{Kinetic friction} \]

\[ f_{vis} = -K_{vis} \eta v \quad \text{Viscosity for small objects} \]

\[ K_{vis} = 6\pi r \quad \text{No turbulence for sphere} \]
Physics Review: Spring-damper

\[ F = k_s (L_{current} - L_{rest}) \]

Hooke’s Law

\[ F = k_s (L_{current} - L_{rest}) - k_d V_{spring} \]
Physics Review: Momentum

\[ P = mv \]

conservation of momentum (mv)
In a closed system, momentum is conserved

\[ P = \sum mv = \sum m'v' \]

After collision has same momentum as before collision
Elastic Collisions

No energy lost (e.g. deformations, heat)

$$\Sigma m v = \Sigma m' v'$$

$$F = ma = \frac{d(mv)}{dt} = \frac{dP}{dt}$$
Elastic Collisions & Kinetic Energy

As in all collisions: momentum is conserved

\[ P = \Sigma m v = \Sigma m' v' \]

In elastic collisions, kinetic energy is also conserved

\[ P = \Sigma \frac{1}{2} m v^2 = \Sigma \frac{1}{2} m' v'^2 \]

Solve for velocities
Inelastic Collisions

Kinetic energy is NOT conserved
Inelastic Collisions

Kinetic energy lost to deformation and heat

Momentum is conserved

Coefficient of restitution
ratio of velocities before
and after collision

Rick Parent

Computer Animation
Center of Mass

\[ C_{\text{mass}} = \frac{1}{M} \int (r) dm = \frac{1}{M} \int (\rho(r)r) dV = \frac{\int (\rho(r)r) d\rho V}{\int \rho(r) dV} \]

\[ C_{\text{mass}} = \frac{\sum m_i r_i}{\sum m_i} \]
# Physics Review

## Linear v. angular terms

<table>
<thead>
<tr>
<th>Linear</th>
<th>Angular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>Rotation, orientation</td>
</tr>
<tr>
<td>Velocity</td>
<td>Rotational (angular) velocity</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Rotational (angular) acceleration</td>
</tr>
<tr>
<td>Mass</td>
<td>Inertia tensor</td>
</tr>
<tr>
<td>Force</td>
<td>Torque</td>
</tr>
<tr>
<td>Momentum</td>
<td>Angular momentum</td>
</tr>
</tbody>
</table>
Change in point due to rotation

\( \omega(t) \) - Angular velocity

On-axis or off axis rotation
Angular velocity is the same

\[
p(t) = x(t) + r(t)
\]

\[
\dot{r}(t) = \omega(t) \times r(t)
\]

\[
|\dot{r}(t)| = |\omega(t)||r(t)|\sin \theta
\]
Delta-orientation due to rotation

\[ R(t) = \begin{bmatrix} R_1(t) & R_2(t) & R_3(t) \end{bmatrix} \]

\[ \dot{R}(t) = \begin{bmatrix} \omega(t) \times R_1(t) & \omega(t) \times R_2(t) & \omega(t) \times R_3(t) \end{bmatrix} \]

\[ A \times B = \begin{bmatrix} A_y B_z - A_z B_y \\ A_z B_x - A_x B_z \\ A_x B_y - A_y B_x \end{bmatrix} = \begin{bmatrix} 0 & -A_z & A_y \\ A_z & 0 & -A_x \\ -A_y & A_x & 0 \end{bmatrix} \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = \dot{A}^* B \]

\[ \dot{R}(t) = \omega(t)^* R(t) \]
Change to a point on rotating object

\[ q(t) = R(t)q + x(t) \]

\[ R(t)q = q(t) - x(t) \]

\[ \dot{q}(t) = \omega(t) \times R(t)q + v(t) \]

\[ \dot{q}(t) = \omega(t) \times (q(t) - x(t)) + v(t) \]
Physics Review: Angular Stuff

\[ f = ma \]
\[ P = mv \]
\[ f = \frac{dP}{dt} \]
\[ \Sigma P_i = c \]

\[ \tau = r \times F \]
\[ L = r \times p \]
\[ \tau = \frac{dL}{dt} \]
\[ \Sigma L_i = c \]
\[ \tau = I \alpha \]
Angular Momentum

\[
L(t) = \sum((q(t) - x(t)) \times m_i (\dot{q}(t) - v(t)))
= \sum(R(t)q \times m_i (\omega(t) \times (q(t) - x(t))))
= \sum(m_i (R(t)q \times (\omega(t) \times R(t)q)))
\]
Inertia Tensor

Aka Angular mass

\[ I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \]

\[ I_{xx} = \int (y^2 + z^2)\,dm \quad I_{xy} = -\int xy\,dm \]
\[ I_{yy} = \int (x^2 + z^2)\,dm \quad I_{xz} = -\int xz\,dm \]
\[ I_{zz} = \int (x^2 + y^2)\,dm \quad I_{yz} = -\int yz\,dm \]
Inertia
Tensor of particles

Discrete version
For particle collection

\[
I = \begin{bmatrix}
I_{xx} & I_{xy} & I_{xz} \\
I_{xy} & I_{yy} & I_{yz} \\
I_{xz} & I_{yz} & I_{zz}
\end{bmatrix}
\]

\[
I_{xx} = \sum m_i (y_i^2 + z_i^2) \quad \quad I_{xy} = -\sum m_i x_i y_i \\
I_{yy} = \sum m_i (x_i^2 + z_i^2) \quad \quad I_{xz} = -\sum m_i x_i z_i \\
I_{zz} = \sum m_i (x_i^2 + y_i^2) \quad \quad I_{yz} = -\sum m_i y_i z_i
\]
Standard Inertia Tensors

Symmetric wrt axes

\[ I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \]

Cuboid

\[ I = \frac{1}{12} \begin{bmatrix} M(b^2 + c^2) & 0 & 0 \\ 0 & M(a^2 + c^2) & 0 \\ 0 & 0 & M(a^2 + b^2) \end{bmatrix} \]

Sphere

\[ I = \frac{2MR^2}{5} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]
Inertia Tensor transformations

\[ I_{\text{translated}} = \begin{bmatrix}
I_{xx} + M(Y^2 + Z^2) & -I_{xy} - MXY & -I_{xz} - MXZ \\
-I_{xy} - MXY & I_{yy} + M(X^2 + Z^2) & -I_{yz} - MYZ \\
-I_{xz} - MXZ & -I_{yz} - MYZ & I_{zz} + M(X^2 + Y^2)
\end{bmatrix} \]

\[ I_{\text{rotated}} = R I_{\text{object}} R^{-1} \]
The Equations

\[ S(t) = \begin{bmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{bmatrix} \]

Object attributes \( M, I_{\text{object}}^{-1} \)

\[ I_{\text{rotated}}^{-1} = R I_{\text{object}}^{-1} R^{-1} \]

\[ v(t) = \frac{P(t)}{M} \]

\[ w(t) = I(t)^{-1} L(t) \]

If using rotation matrix, will need to orthonormalize updated rotation matrix

\[ \frac{d}{dt} S(t) = \frac{d}{dt} \begin{bmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ w(t)^* R(t) \\ F(t) \\ \tau(t) \end{bmatrix} \]

\( \tau = r \times F \)
Springs

Flexible objects
Cloth
Virtual springs
Proportional derivative controllers (PDCs)
Spring-mass-damper system
Spring-mass system
“Virtual” edge springs system
Angular springs

Linear spring between vertices

Dihedral angular spring
Spring mesh

Each vertex is a point mass
Each edge is a spring-damper
Diagonal springs for rigidity
Angular springs connect every other mass point
Global forces: gravity, wind

Rick Parent

Computer Animation
Virtual springs - soft constraints
Proportional (Derivative) Controllers

e.g., particle reacts to other forces while trying to maintain position on curve – virtual spring

\[ d = x(t) - p(t) \]
\[ F = k_s d \]
\[ F = k_s d - k_d v_{relative} \]
Particle systems

Lots of small particles - local rules of behavior
Create ‘emergent’ element

Particles:
- Do collide with the environment
- Do not collide with other particles
- Do not cast shadows on other particles
- Might cast shadows on environment
- Do not reflect light - usually emit it
Particle system

- Collides with environment but not other particles
- Particle’s midlife with modified color and shading
- Particle’s demise, based on constrained and randomized life span
- Particle’s birth: constrained and time with initial color and shading (also randomized)
Particle system implementation

**STEPS**
1. for each particle
   1. if dead, reallocate and assign new attributes
   2. animate particle, modify attributes
2. render particles

Use constrained randomization to keep control of the simulation while adding interest to the visuals