Game Physics

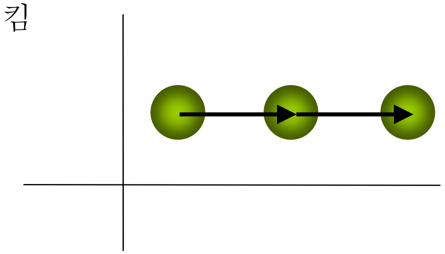
305890
2008년 봄학기
6/11/2008
박경신

Game Physics

- 운동 (Motion)
- 위치 (Position), 속도 (Velocity), 가속도 (Acceleration)
- 힘 (Force), 중력 (Gravity)
- 부력(Buoyancy), 저항 (Drag)
- 마찰 (Friction)
  - 운동마찰 (Kinetic friction)
  - 정지마찰 (Static friction)
- 스프링 (Spring)

Motion

- 운동 (motion)은 시간에 따라 물체의 위치가 변하는 현상
- 운동을 나타내기 위해서 속력(speed), 속도 (velocity), 가속도 (acceleration)라는 물리량을 사용
- 속도 (velocity)는 벡터임
  - 벡터의 방향 (direction)은 움직임의 방향을 나타냄
  - 벡터의 길이 (magnitude)는 움직임의 속력(speed)를 나타냄
- 속도 벡터는 객체가 시간이 경과에 따라 얼마나 움직였나를 가리킴



Basic Motion

- 변위 (displacement) = 속도 (velocity) * 시간 (time)
- 만약 물체가 시작점 P_0에서 출발하여 일정한 속도 (constant velocity) v만큼 움직인다면, t 단위 시간이 경과한 후의 **x**(t) 위치(position):

$$\mathbf{x}(t) = \mathbf{x}_0 + v\, t$$

- NOTE: 단위(unit)는 거리의 경우 meters, 시간의 경우 seconds, 속도의 경우 meters/second를 의미함

Varying Velocity

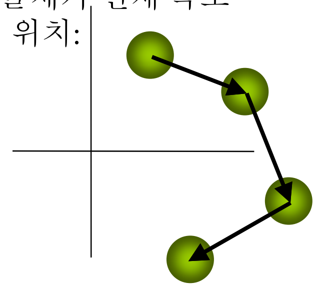
- 앞의 공식은 물체가 일정한 속도로 움직일 때만 적용가능
  - 보다 일반적인 경우, 물체의 속도량은 시간이 경과에 따라 바뀜
- 속도는 단위 시간당 움직임 위치의 변화량
  - 시간에 따른 위치의 변화율
  - 속도는 위치를 시간에 대해 한번 미분한 값: $v(t) = \frac{d}{dt}\mathbf{x}(t)$
  - 속도가 일정한 경우: $v(t) = v_0$
  - 속도가 일정한 가속으로 변하는 경우: $v(t) = v_0 + a\,t$
- 이때, 변위(displacement)는 속도의 적분 값 (integral)

$$displacement = \int_0^t velocity\ dt$$

Euler Method (or Euler Integration)

- 오일러 방법 (Euler method, 혹은 Euler Integration)
  - 테일러 급수에서 유도된 방법으로 가장 기본적인 적분 방법
  - 그러나, 비교적 오차가 크게 남
  - 구간 [a, b]를 N개의 구간으로 나누었을 때 각각의 점을
    $t_i = a + i*h,\ i=0, 1, 2, .., N$ and $h = (b-a)/N$
  - 오일러 방법은 $\omega_0=\alpha,\ \omega_{i+1} = \omega_i + h*f(t_i, \omega_i),\ i=0, 1, 2, .., N-1$
- 오일러 방법을 적용하여 단위 시간별 물체가 현재 속도 (current velocity) 로 직선으로 움직인 위치:

$$dt = t_1 - t_0$$
$$\mathbf{x}(t_1) = \mathbf{x}(t_0) + v\ dt$$



Acceleration

- 가속도는 단위 시간당 속도의 변화량
  - 가속도는 속도를 시간에 대해 한번 미분한 값

$$a(t) = \frac{d}{dt}v(t) = \frac{d^2}{dt^2}\mathbf{x}(t)$$

- 속도는 가속도의 적분 값 (integral)

$$velocity = \int_0^t acceleration\ dt$$

- 오일러 방법을 적용하여 위치 계산:

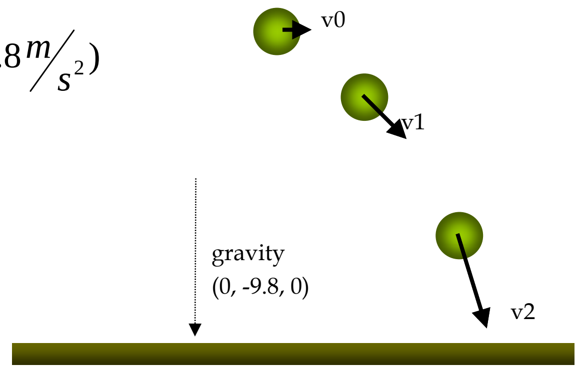
```
dt = t1 – t0;
Acc = computeAcceleration();
Vel = Vel * Acc * dt;
Pos = Pos * Vel * dt;
```

$$\mathbf{x}(t) = \int_0^t (v_0 + at)dt = \mathbf{x}_0 + v_0 t + \frac{1}{2}a\ t^2$$

Gravity

- 중력 (gravity)는 지구의 표면에서 일정한 가속도인 9.8 meter/seconds
- 중력장 내의 물체는 지구 중심 쪽으로 F=mg 힘을 받음. M은 질량(mass)임

$$F = mg\ \ (g = -9.8\ {}^{m}/_{s^2})$$



Force

- 힘 (force)는 운동(motion)에 변화를 일으키는 원인
- 뉴튼 역학의 제 2법칙 (가속도의 법칙)

$$F = ma$$

$$\Rightarrow a = F / m$$

- 만약 질량(mass) M인 물체에 힘 (force) F가 가해졌을 때, 오일러 방법을 적용하여 운동 (motion) 계산:

```
Acc = F/M;
Vel += Acc * dt;
Pos += Vel * dt;
```

뉴튼 역학의 3법칙

- 관성의 법칙: 모든 물체는 다른 물체의 움직임의 영향을 받지 않는다고 할 때, 정지해 있었다면 계속 정지해 있을 것이고, 움직이고 있었다면 일정한 속도로 계속 운동할 것이다.
- 가속도의 법칙: 물체의 운동량의 변화율은, 크기와 방향에서, 그 물체에 작용하는 힘에 따른다.
- 작용, 반작용의 법칙: 모든 작용에는 그 반대방향으로 같은 크기의 반작용이 존재한다.

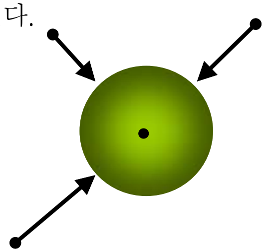
Gravitational Force

- 만유인력/중력 (gravitational force)
  - 이 우주 안에 존재하는 모든 물체들은 다른 물체를 무조건 끌어당기는 성질이 있는데, 그 힘의 크기는 두 물체의 질량의 곱에 비례하고, 둘 사이의 거리의 제곱에 반비례한다.
  - F의 방향은 큰 질량 물체의 무게중심 (center of mass)를 향한다.
- 물리 시뮬레이션
  - 보다 사실적인 시뮬레이션을 위해서 매 프레임마다 모든 물체간의 힘을 계산할 필요가 있다.
  - 여러 개의 힘이 적용될 때, 벡터가 추가된다.

$$F_{gravity} = \frac{GmM}{r^2} \quad (G = 6.673 \times 10^{-11})$$

m, M : 질점의질량(kg)

r : 거리$(meter)$



Projectile Motion

- 시간 t=0에서의 초기 위치가 P_0이고, 초기 속도가 v_0인 발사체 (projectile)의 위치

$$\mathbf{x}(t) = \mathbf{x}_0 + v_0 t + \frac{1}{2} g t^2$$

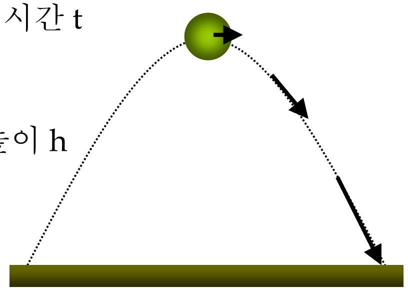
$$x(t) = x_0 + v_x t, \quad y(t) = y_0 + v_y t - \frac{1}{2} g t^2, \quad z(t) = z0 + v_z t$$

- 발사체가 최대 높이에 도달하는 시간 t

$$y(t) = v_y t - g t^2 = 0 \Rightarrow t = \frac{v_y}{g}$$

- 발사체가 도달할 수 있는 최대 높이 h

$$h = y_0 + \frac{v_y^2}{2g}$$



Projectile Motion

- 발사체가 원래의 높이로 내려올 때까지 날아간 수평 거리

$$y(t) = y_0 + v_y t - \frac{1}{2} g t^2 = y_0 \Rightarrow t = 0 \text{또는 } t = \frac{2 v_y}{g}$$

$$x(t) = x_0 + v_x t \text{에 } t \text{를 대입} \Rightarrow r = \frac{2 v_x v_y}{g}$$

- 발사될 때의 초기 속력 s가 주어졌을 때, 발사체를 최대한 높이 올릴 수 있는 발사각도

$$h = y_0 + \frac{v_z^2}{2g} \Rightarrow h = y_0 + \frac{(s \sin\alpha)^2}{2g} \Rightarrow \alpha = \sin^{-1}\left(\frac{1}{s}\sqrt{2g(h - y_0)}\right)$$

- 원하는 도달 거리 r를 가기 위한 발사 각도

$$r = \frac{2 v_x v_y}{g} \Rightarrow r = \frac{2(s\cos\alpha)(s\sin\alpha)}{g} = \frac{s^2}{g}\sin 2\alpha \Rightarrow \alpha = \frac{1}{2}\sin^{-1}\frac{rg}{s^2}$$

Buoyancy

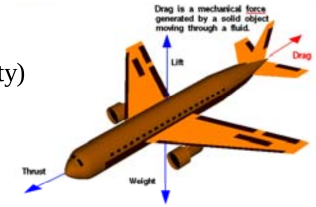
- 공기나 물과 같은 유체 속에 돌과 같은 고체가 존재하는 경우, 중력 외에 부력과 저항력 두 가지 힘이 작용.
- 부력 (buoyancy)
  - 밀도의 차이에 의하여 위로 저절로 상승하여 올라가는 부양력
  - 아르키메데스(Archimedes)의 원리: 물체에 작용하는 부력의 크기는 물체가 밀어낸 유체의 무게와 같다.
  - 부력 = 유체의 밀도 x 중력가속도 (9.81 m/s^2) x 부피 (물체의 바닥 넓이 x 물체의 높이)

$$F_{buoyancy} = -\rho_{liquid} gV$$

Drag

- 저항력 (drag)
  - 물체가 움직임으로서 그를 방해하는 유체의 힘. 저항력은 물체가 물이나 공기 속에서 움직일 때에만 발생.
  - Drag at low velocity (Stoke's drag): 저항력은 물체가 클수록 (r), 점성이 클수록 (η), 속도가 빠를 수록 (v)세다.

$$F_d = -bv$$
$$b = 6\pi\eta r$$
$$r: \text{입자의반지름},\ \eta: \text{유체의 점성(viscosity)}$$



  - Drag at high velocity:

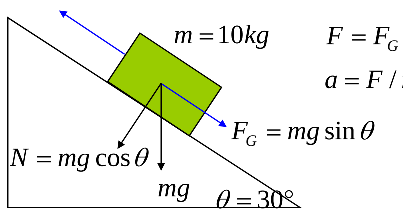
$$F_d = \frac{1}{2}\rho v^2 A C_d \hat{v}$$
$$\rho: \text{유체의 밀도}(density),\ v: \text{유체에서물체의속력}$$
$$A: \text{영역},\ C_d: drag\ coefficient,\ \hat{v}: \text{속도의방향}$$

Kinetic Friction

- 운동마찰 (kinetic friction)
  - 서로 상대적으로 움직이는 두 표면 사이에서 발생하는 힘. 각자의 운동에 대한 저항으로 작용함.
  - 운동마찰력

$$F_K = -\mu_K N$$
$$N: \text{물체가 표면에대해가하는힘의법선}$$
$$\mu_K: \text{운동마찰계수. 물체들이맞닿는 표면의재질에따라 다름}$$

$$F_K = -\mu_K N = -\mu_K mg\cos\theta$$



$$F = F_G(\text{평행면으로의중력}) + F_K(\text{운동마찰})$$
$$a = F/m = (F_G + F_K)/m = g\sin\theta - \mu_K g\cos\theta$$

Static Friction

- 정지마찰 (static friction)
  - 한 표면이 그 위에 놓여 있는 정지된 물체를 움직이지 못하도록 붙잡고 있는 힘. 물체에 가해지는 tangential force의 반대방향으로 작용하여 물체의 운동을 방해함.
  - 정지마찰력

$$F_S = -\mu_S N$$
$$N: \text{법선방향의힘}$$
$$\mu_S: \text{정지마찰계수. 물체들이맞닿는 표면의재질에따라 다름}$$

  - 물체에 가해지는 힘이 F_S의 최대값을 넘는 순간 물체가 움직이기 시작하며, 그때부터는 F_S가 사라지고, F_K가 작용.
  - 정지된 물체를 움직이게 하는 것이 움직이는 물체를 계속 움직이게 하는 것보다 더 힘들다: $F_K < F_S$
  - 평면을 기울일 때 물체가 미끄러지기 시작하는 각도: 정지마찰력 = 평면과 수평인 중력성분

$$\mu_S mg\cos\theta = mg\sin\theta \Rightarrow \theta = \tan^{-1}\mu_S$$

Momentum

- 힘 (force)은 운동량 (momentum) 의 미분 값

$$P = mv$$

$$\Rightarrow \frac{dP}{dt} = m\frac{dv}{dt} = ma = F$$

- 모멘텀을 사용한 방법
  - 힘을 적분하여 모멘텀 계산
  - 모멘텀과 질량으로 속도 계산
  - 속도를 적분하여 위치 계산

```
Force = ComputeTotalForce();
Momentum += Force * dt;
Velocity = Momentum / Mass;
Position += Velocity* dt;
```

Angular Velocity

- 각속도 (angular velocity)는 물체의 회전속도
- 각속도는 시간당 각도의 변화율 (radian/seconds 단위)

$$\omega(t) = \frac{d}{dt}\theta(t)$$

- 각속도는 회전축 A에 평행이고 크기가 ω(t)인 벡터

$$\boldsymbol{\omega}(t) = \omega(t)A$$

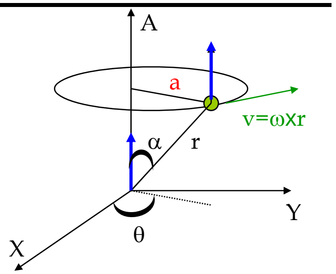
- 회전의 중심으로부터 r만큼 떨어진 곳에서 물체가 속도 v로 운동하고 있을 때
  - 물체의 속도: $v(t) = |\omega(t)r|$
  - 물체의 위치를 r(t)라고 하면 물체의 선속도 (linear velocity):

$$v(t) = \omega(t) \times r(t)$$

Centrifugal Force

- 물체의 선가속도 (linear acceleration)

$$a(t) = \omega'(t) \times r(t) + \omega(t) \times r'(t)$$
$$= \omega'(t) \times r(t) + \omega(t) \times [\omega(t) \times r(t)]$$



- 각속도가 일정한 경우: ω'(t)=0

$$a(t) = \omega(t) \times [\omega(t) \times r(t)]$$

  - 가속도 a는 안쪽 방향임: 끝의 장력 (tension)으로부터 발생
  - 물체에서 장력과 같은 크기의 반대 방향으로 힘이 작용 - 원심력 (centrifugal force):

$$F_c = -m(\omega(t) \times [\omega(t) \times r(t)])$$

  - r(t)와 ω(t)가 수직인 경우, 원심력은 한 scalar로 표현됨

$$F_c = m\omega^2 r = \frac{mv^2}{r}$$

Rigid Motion

- 강체 운동 (rigid motion)
  - 강체란 물체들이 서로에 대해 절대적으로 고정되어 있는 고체 (solid object). Translation과 rotation만 가능함
- Rigid body dynamics
  - 자동차 등 rigid body motion을 다룸
  - Linear와 angular에 대한 위치, 속도, 가속도를 다루어야 함

```
Force = ComputeTotalForce();
Momentum += Force * dt;
Velocity = Momentum / Mass;
Position += Velocity* dt;
Torque = ComputeTotalTorque();
AngMomentum += Torque * dt;
Matrix I = Matrix*RotInertia*Matrix.Inverse(); // tensor
AngVelocity = I.Inverse()*AngMomentum;
Matrix.Rotate(AngVelocity*dt);
```

Integration Method

- Euler method
  - v = v_0 + a*dt, x = x_0 + v*dt
  - 변화율이 상수일 때는 100% 정확함
  - 변화율이 시간에 따라 변할 때는 에러가 존재함

```
float t = 0;      // 현재 시간
float dt = 1;    // 시간 간격 (timestamp)
float velocity = 0;        // 초기 속도
float position = 0;        // 초기 위치
float force = 10;
float mass = 1;
float acceleration = force/mass;
while (t<=10) {
    position += velocity * dt;
    velocity += acceleration * dt;
    t += dt;
}
```

Initial : $y'(t) = f(t, y(t)), y(t_0) = y_0$

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

Integration Method

- Runge-Kutta method
  - RK4는 매우 정확하고 미분에 안정적임

Initial : $y' = f(t, y), y(t_0) = y_0$

RK4 : $y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

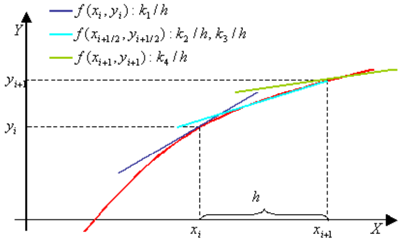
$k_1 = f(t_n, y_n)$

$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$

$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$

$k_4 = f(t_n + h, y_n + hk_3)$

$slope = \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$



Integration Method

```
void RK4Integration(vector3& pos, vector3& vel, float t, float dt) {
    vector3 k1Vel = vel;
    vector3 k1Acc = f(t, pos, vel);
    vector3 k2Vel = vel + 0.5f * dt * k1Acc;
    vector3 k2Acc = f(t + 0.5f * dt, pos + 0.5f * dt * k1Vel, k2Vel);
    vector3 k3Vel = vel + 0.5f * dt * k2Acc;
    vector3 k3Acc = f(t + 0.5f * dt, pos + 0.5f * dt * k2Vel, k3Vel);
    vector3 k4Vel = vel + dt * k3Acc;
    vector3 k4Acc = f(t + dt, pos + dt * k3Vel, k4Vel);
    pos += (dt / 6.0f) * (k1Vel + 2.0f * k2Vel + 2.0f * k3Vel + k4Vel);
    vel += (dt / 6.0f) * (k1Acc + 2.0f * k2Acc + 2.0f * k3Acc + k4Acc);
}
while (t<=10) {
    RK4Integration(position, velocity, t, dt);
    t += dt;
}
```

Springs

- Hooke's Law
  - 스프링의 힘은 스프링의 길이/변위에 비례

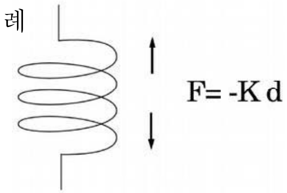
$F = -K_s d$

K_s **(spring constant) :**

스프링강도를 표현하는 상수

d **(displacement from rest length) :**

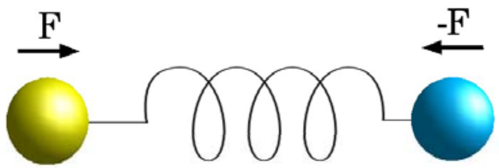
스프링을 이루는 2 **position vectors**의 차



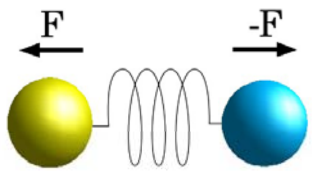
- 스프링은 스프링으로 연결된 2 질점으로 모델링함.
- 동일한 크기의 반대방향의 힘이 양쪽에 적용됨.

Springs

- 스프링의 입자가 멀리 떨어질 수록 스프링의 안정상태 위치로 끌어 당기는 힘이 커짐



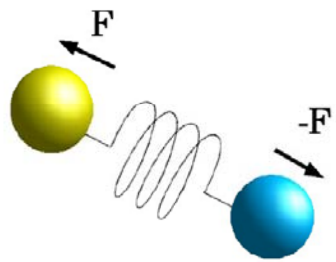
- 스프링의 양끝을 동시에 눌렀을 때 입자의 위치가 스프링의 안정상태에서의 길이만큼 떨어지도록 미는 힘이 작용함



Springs

- 두 점간의 벡터를 사용하여 변위와 힘의 방향을 계산

```
Vector3 v = point1 – point0;
float displacement = v.length() – restLength;
v.normalize();
Vector3 force = springConstant * displacement * v;
```



References

- http://en.wikipedia.org/wiki/Newton%27s_laws_of_motion
- http://en.wikipedia.org/wiki/Equations_of_motion
- http://en.wikipedia.org/wiki/Projectile
- http://en.wikipedia.org/wiki/Trajectory
- http://en.wikipedia.org/wiki/Buoyancy
- http://en.wikipedia.org/wiki/Drag_(physics)
- http://en.wikipedia.org/wiki/Euler_method
- http://en.wikipedia.org/wiki/RK4
- http://www.gaffer.org/game-physics/