물리기반 모델링 - *Part 1*

물리기반 모델링의 기초

동명대학교 게임공학과
강영민

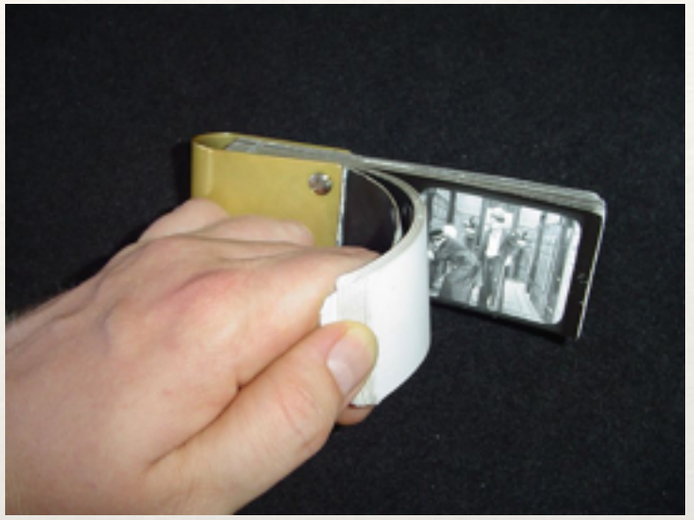
물리기반 모델링

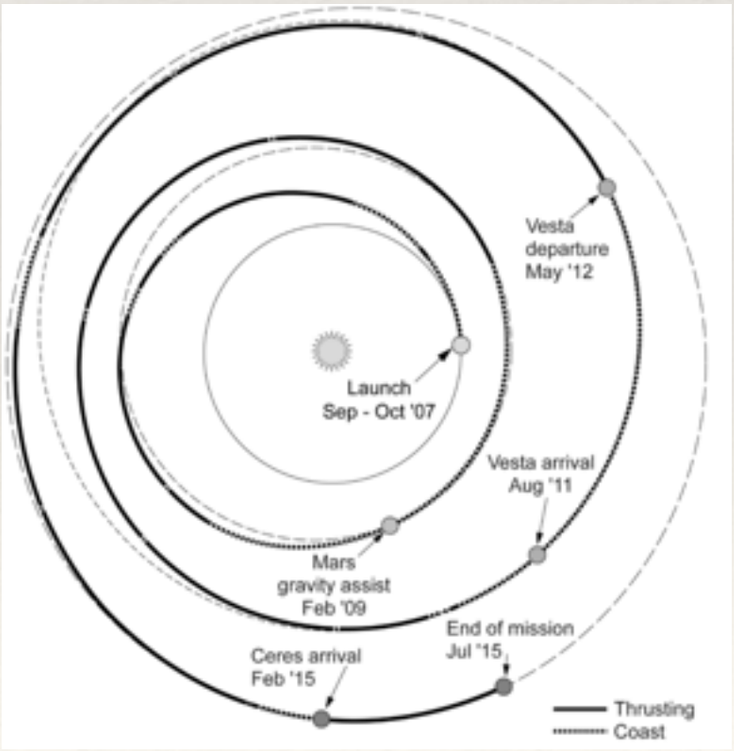
1.1 질량, 힘, 시간

동명대학교 게임공학과
강영민

물리기반 모델링은 왜 하나?

- 애니메이션/시뮬레이션
  - 사실적인 움직임이 필요
- 애니메이션
  - 시간에 따른 상태의 변화
  - 상태 - 위치, 속도...
    - 물리의 중요한 연구 대상
- 사실적인 움직임
  - 물리에 의해 결정됨





운동에 대한 연구

- 뉴턴
  - 고전 물리에서 운동에 대한 이론을 정립
  - 프린키피아 (*Philosophiae Naturalis Principia Mathematica*)
- 뉴턴의 운동법칙
  - 1법칙: 일정한 운동을 하는 객체는 외부의 힘이 가해지기 전에는 그 운동을 유지하려고 한다. (관성)
  - 2법칙: $\mathbf{F}=m\mathbf{a}$.
  - 3법칙: 모든 작용에는 같은 크기의 반대방향으로 반작용이 있다.

질량

- 질량
  - 힘에 의한 가속에 저항하는 속성
  - 누적된 밀도 (kg/m^3)
    - 질량 = 밀도의 적분

$$m = \int \rho dV$$

질량 중심

- 이론

$$x_c = \frac{\int x_0 dm}{m}$$

$$y_c = \frac{\int y_0 dm}{m}$$

$$z_c = \frac{\int z_0 dm}{m}$$

$$c_g = \frac{\sum (c_{g_i} m_i)}{m}$$

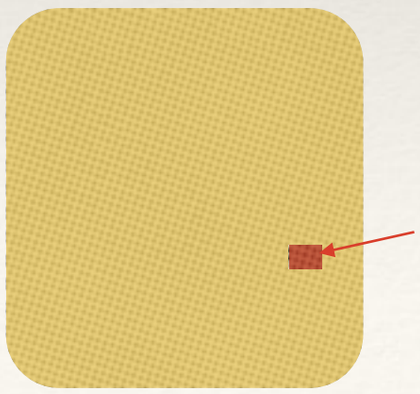
- 계산

$$x_c = \frac{\sum x_i m_i}{\sum m_i}$$

$$y_c = \frac{\sum y_i m_i}{\sum m_i}$$

$$z_c = \frac{\sum z_i m_i}{\sum m_i}$$

V



뉴턴의 운동 제2법칙

- 힘 = 질량 x 가속

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

- 총 힘의 합 $\mathbf{f}_{net}$
- 가해진 모든 힘의 합 $\mathbf{f}_{net} = \sum \mathbf{f}_i$

$$\sum f_x = ma_x$$
$$\sum f_y = ma_y$$
$$\sum f_z = ma_z$$

- 가속은 힘의 총합에 의해 결정된다

$$\mathbf{a} = \frac{\mathbf{f}_{net}}{m}$$

선운동량과 미분

- 선운동량: **G**
  - 질량 x 속도
  - $\mathbf{G} = m\mathbf{v}$
- 선운동량을 시간에 대해 미분하면

$$\frac{d\mathbf{G}}{dt} = \frac{dm\mathbf{v}}{dt} = m\frac{d\mathbf{v}}{dt} = m\mathbf{a}$$

  - 결과는 힘

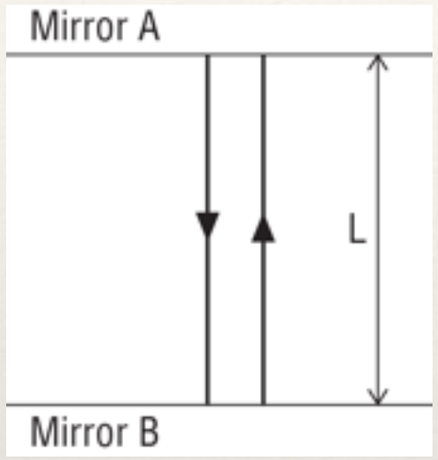
$$\frac{d\mathbf{G}}{dt} = \sum \mathbf{f}$$

시간

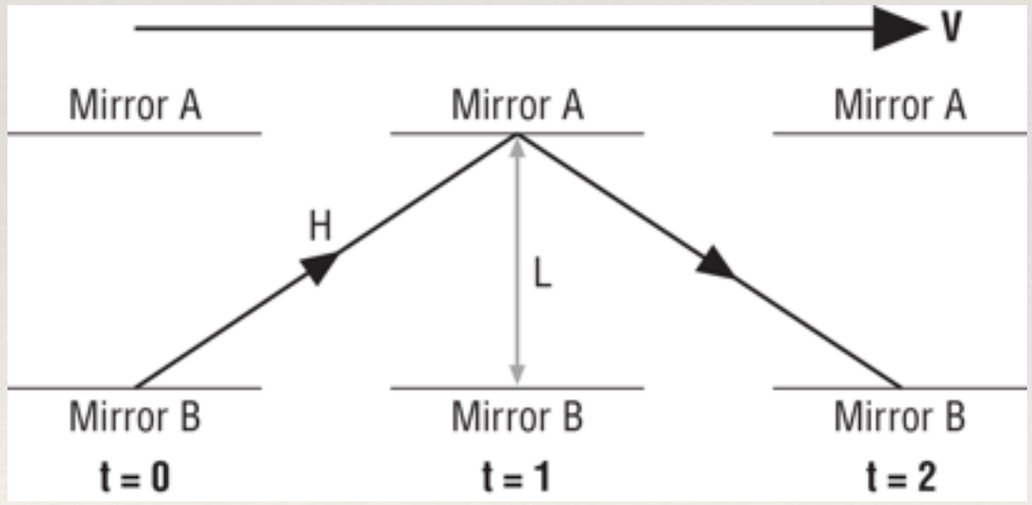
- 고전 물리
  - 시간은 불변
- 현대 물리
  - 시간은 가변적
  - 빛의 속도는 상수: c
    - $c = 299{,}792{,}458\ m/s$

상대론적 시간

수업 시간에 사용하지는 않을 것이지만 재미로 다뤄보는 상대론적 시간



$$c = 2L/\Delta t$$



속도 v로 이동하는 우주선에서 빛의 이동

우주선에서

$$c = 2L/\Delta t_s$$

지구에서

$$c = 2H/\Delta t_e$$

빛의 속도 c 가 상수라면
두 공간의 시간은 서로 달라야!

시간 확장

- 간단한 기하

$$H^2 = L^2 + \left(\frac{\mathbf{v}\Delta t_e}{2}\right)^2 \Rightarrow \left(\frac{c\Delta t_e}{2}\right)^2 = L^2 + \left(\frac{\mathbf{v}\Delta t_e}{2}\right)^2$$

- 시간 확장의 계산

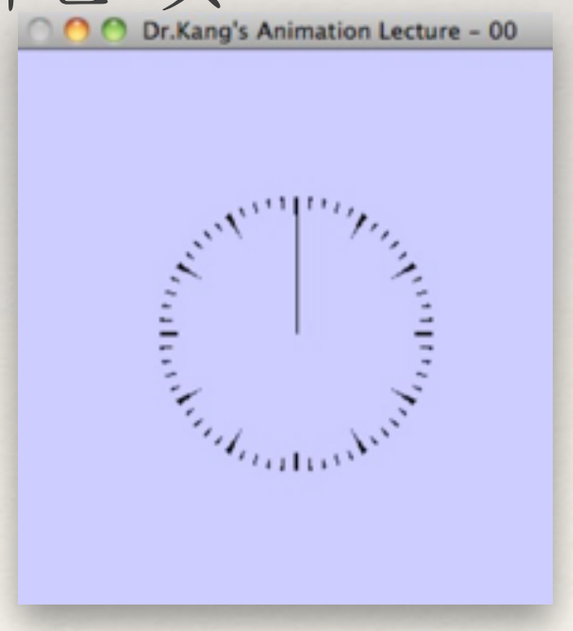
$$\begin{aligned} 4L^2 &= c^2\Delta t_e^2 - \mathbf{v}^2\Delta t_e^2 \\ 4L^2 &= (c^2 - \mathbf{v}^2)\Delta t_e^2 \\ \frac{4L^2}{c^2} &= (1 - \frac{\mathbf{v}^2}{c^2})\Delta t_e^2 \end{aligned}$$

$$\Rightarrow \quad \begin{aligned} \frac{2L}{c} &= \sqrt{1 - \frac{\mathbf{v}^2}{c^2}}\,\Delta t_e \\ \Delta t &= \sqrt{1 - \frac{\mathbf{v}^2}{c^2}}\,\Delta t_e \end{aligned}$$

$$\Delta t_e = \frac{1}{\sqrt{1 - \frac{\mathbf{v}^2}{c^2}}}\Delta t$$

시간과 애니메이션

- 애니메이션
  - 시간에 따른 변화
- 컴퓨터 애니메이션
  - 시간에 대해 적절한 물리적 상태를 계산하는 것
- 시간을 측정해야만 애니메이션이 가능
  - 시간 측정 프로그램이 필요



Stop Watch (header)

```
#ifndef _STOPWATCH_YMKANG_H
#define _STOPWATCH_YMKANG_H

#ifdef WIN32   // Windows system specific
#include <windows.h>
#else          // Unix based system specific
#include <sys/time.h>
#endif

class StopWatch {
#ifdef WIN32
    LARGE_INTEGER frequency;                    // ticks per second
    LARGE_INTEGER startCount;                   //
    LARGE_INTEGER endCount;                     //
#else
    timeval startCount;                         //
    timeval endCount;                           //
#endif
    double startTimeInMicroSec;
    double endTimeInMicroSec;
public:
    StopWatch();
    void start();                // start StopWatch and record time to "startCount"
    void stop();                // stop StopWatch and record time to "endCount"
    double getElapsedTime();    // return the elapsed time at the last stop since the last start (microsec)
};

#endif
```

Stop Watch (implementation)

```
/*
 *  StopWatch.cpp
 *  Young-Min Kang
 *  Tongmyong University
 *
 */

#include "StopWatch.h"

StopWatch::StopWatch() {
#ifdef WIN32
    QueryPerformanceFrequency(&frequency);
    startCount.QuadPart = 0;
    endCount.QuadPart = 0;
#else
    startCount.tv_sec = startCount.tv_usec = 0;
    endCount.tv_sec = endCount.tv_usec = 0;
#endif
    startTimeInMicroSec = endTimeInMicroSec = 0.0;
}

void StopWatch::start() {
#ifdef WIN32
    QueryPerformanceCounter(&startCount);
#else
    gettimeofday(&startCount, NULL);
#endif
}

void StopWatch::stop() {
#ifdef WIN32
    QueryPerformanceCounter(&endCount);
#else
    gettimeofday(&endCount, NULL);
#endif
}

double StopWatch::getElapsedTime(){
#ifdef WIN32
    startTimeInMicroSec = startCount.QuadPart * (1000000.0 / frequency.QuadPart);
    endTimeInMicroSec = endCount.QuadPart * (1000000.0 / frequency.QuadPart);
#else
    startTimeInMicroSec = (startCount.tv_sec * 1000000.0) + startCount.tv_usec;
    endTimeInMicroSec = (endCount.tv_sec * 1000000.0) + endCount.tv_usec;
#endif
    return endTimeInMicroSec - startTimeInMicroSec;
}
```

시간 측정 결과 가시화

- 자신의 시계를 구현해 보라.