물리기반모델링

1.3 역학과 수치적분

동명대학교
강영민

역학

- 역학
  - 힘에 기반하여 운동을 이해
- 중요한 공식 (뉴턴의 제2법칙)
  - $\mathrm{f} = m\mathrm{a}$
- 강체는.... 회전힘 = 회전질량*회전가속

$$\tau = \mathbf{I}\dot{\omega}$$

운동방정식의 적분

- 뉴턴의 운동 제2법칙
  - f = ma
- 다시 말하면...

$$\mathbf{f} = m\frac{d\mathbf{v}}{dt}$$

- dv = ?

$$\frac{\mathbf{f}dt}{m} = d\mathbf{v}$$

$$\int_{t_1}^{t_2} \frac{\mathbf{f}dt}{m} = \int_{\mathbf{v}_1}^{\mathbf{v}_2} d\mathbf{v}$$

힘이 (이 시간 동안) 상수일 경우

$$\frac{\mathbf{f}}{m}(t_2 - t_1) = \mathbf{v}_2 - \mathbf{v}_1$$

$$\frac{\mathbf{f}}{m}\Delta t = \Delta\mathbf{v}$$

초기 조건 문제

- 초기 조건
  - x(t), v(t)
  - 시간 t에서의 위치와 속도
- 문제
  - 조금의 시간 dt가 흐른 뒤를 예측
  - 예측의 대상 위치 x(t+dt)와 속도 v(t+dt)를 구하기
- x(t+dt), v(t+dt)를 초기 조건으로 예측을 반복

속도의 갱신

- 속도의 초기 조건
  - v1 = v(t)
- 찾아야 하는 속도
  - v2 = v(t+dt)

$$\frac{\mathbf{f}}{m}(t_2 - t_1) = \mathbf{v}_2 - \mathbf{v}_1$$

$$\frac{\mathbf{f}}{m}\Delta t = \Delta \mathbf{v}$$

- 다음 프레임(t+dt)에서의 속도

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \frac{\mathbf{f}(t)}{m}\Delta t$$

- 혹은

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \mathbf{a}(t)\Delta t$$

위치의 갱신

- 속도와 위치

$$\mathbf{v} = d\mathbf{s}/dt \qquad\qquad \mathbf{v}dt = d\mathbf{s}$$

- 수치 적분

$$\mathbf{v}(t + \Delta t)\Delta t = \Delta\mathbf{s}$$

- 시간 t에서 가해지는 힘 혹은 가속도를 알 수 있다면
  - 시간 t+dt에서의 위치와 속도를 추정할 수 있다.

실시간 시뮬레이션

- 힘을 계산한다: f
- 가속도를 계산한다: $a = f/m$
- 정해진 시간 간격이 흐른 뒤의 속도를 계산한다.
  - 오일러 적분

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \mathbf{a}\Delta t$$

- 시간 간격이 흐른 뒤의 위치도 계산한다.
  - 오일러 적분

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

시뮬레이션 코드 - main.cpp

```
void keyboardFunction(unsigned char key, int x, int y) {
    if (key == 27) exit(0);
    switch (key) {
        case 's':
            if(!myWatch.bRunning()) { Simulator->start(); myWatch.start(); }
            else { myWatch.stop(); Simulator->stop(); }
            break;
        case 'p':
            myWatch.pause(); Simulator->pause();
            break;
        case 'r':
            myWatch.resume(); Simulator->resume();
        default:
            break;
    }
}
void displayFunction(void) {
    …
    // check DT (in microsecond) from StopWatch and store it to "deltaTime" (in seconds)
    deltaTime = myWatch.checkAndComputeDT() / 1000000.0;
    currentTime = myWatch.getTotalElapsedTime() / 1000000.0;
    Simulator->actions(deltaTime, currentTime);
    glutSwapBuffers();
}
```

시뮬레이션 코드 - Simulator.cpp

```
#include "Simulator.h"
#include <stdlib.h>
#include <stdio.h>
// Constructor
CSimulator::CSimulator(): bRunning(false) {  }
CSimulator::~CSimulator() { }

void CSimulator::actions(double dt, double currentTime) {
    if(bRunning) {
        doBeforeSimulation(dt, currentTime);
        doSimulation(dt, currentTime);
        doAfterSimulation(dt, currentTime);
    }
    visualize();
}
// Control Event Handlers
void CSimulator::start() {
    this->init();
    bRunning = true;
}
void CSimulator::stop() {
    bRunning = false;
    this->clean();
}
void CSimulator::pause() {}
void CSimulator::resume() {}
```

시뮬레이션 코드 - DynamicSimulator.cpp

```
#include "DynamicSimulator.h"

CDynamicSimulator::CDynamicSimulator() : CSimulator() {}

void CDynamicSimulator::init() {
   for(int i=0;i<NUMPARTS;i++)particle[i].randomInit();
}

void CDynamicSimulator::clean() {}

void CDynamicSimulator::doBeforeSimulation(double dt, double currentTime) {}

void CDynamicSimulator::doSimulation(double dt, double currentTime) {
    for(int i=0;i<NUMPARTS;i++)
        particle[i].simulate(dt, currentTime);

}
void CDynamicSimulator::doAfterSimulation(double dt, double currentTime) {}

void CDynamicSimulator::visualize(void) {
    for(int i=0;i<NUMPARTS;i++) {
        particle[i].drawWithGL(POINT_DRAW);
    }
}
```

시뮬레이션 코드 - Particle.cpp

```
void CParticle::simulate(double dt, double et) {
    // Update velocity: Euler Integration of acceleration
    vel = vel + dt*gravity;
    // Update position: Euler Integration of velocity
    loc = loc + dt*vel;

    // collision handling
    if(loc[1]<0) {
        loc.set(loc[0], -0.9*loc[1], loc[2]);
        if(vel[1]<0) vel.set(vel[0], -0.9*vel[1], vel[2]);
    }
    setPosition(loc[0], loc[1], loc[2]);
}
```

결과