

Physics for Game Programmers (1) [GDC2007]

Jim Van Verth, Squirrel Eiserloh, Christer Ericson, Gino
van den Bergen, Erin Catto, Marq singer

정리:김성익(noerror@hitel.net)

Spirit of Flame
3D RealTime Graphics Programming Study
Kasa

튜터리얼 대상

- 물리 엔진을 사용하려는 사람
- 간단한 엔진을 작성하는 사람
- 하루 코스 / 나머지는 샘플과 참고자료를
참고

요구되는 배경지식

- 벡터/행렬 연산
- 기본적인 미분
- 뉴턴 물리학 (힘, 가속도, 속도)

참고

- <http://www.essentialmath.com>
- Essential Mathematics for Games and Interactive Applications
- Collision Detection in Interactive 3D Environments
- Real-Time Collision Detection
- Game Programming Gems 4, 5, 6
- Eberly, David H. Game Physics, Morgan Kaufmann, San Francisco, 2003.
- Witken, Andrew, David Baraff, Michael Kass, SIGGRAPH Course Notes, Physically Based Modelling, SIGGRAPH 2004.
- Burden, Richard L. and J. Douglas Faires, Numerical Analysis, PWS Publishing Company, Boston, MA, 1993.
- Halliday, David, Robert Resnik, Jearl Walker. Fundamentals of Physics, Wiley, 2001.
- Eric Weisstein's Mathworld, <http://mathworld.wolfram.com/>
- Eric Weisstein's World of Science, <http://scienceworld.wolfram.com>

Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

게임 물리 프로그래밍의 문제들

- 속여야 할 때 지식
- 단순화
- 물체의 모양을 주는 것
- 물체를 이동시키는 것
- Simulation Baggage
- 충돌 판별
- 지속적인 상호반응
- 불가능을 다루는 것
- 빠르게 만들기

속여야 할 때

- 이산 물리 시뮬은 난처하게도 사실감이 부족함
- 진짜 물리는 엄청나게 비쌌
- 그래서 속임
- 실시간으로 실행가능한 만큼 속일 필요가 있음
- 벼락거리거나, 회복 불가능한 방법으로 망가질 정도로 속이지 않는다
- 많은 시도는 언제, 어떻게 속여야 하는 지 알게 해준다

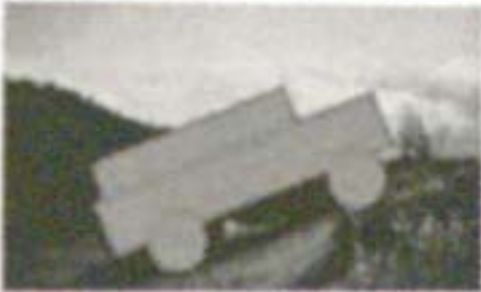
속여야 할 때

- 질문”
- 플레이어가 알아챌것인가 ?
- 플레이어가 신경쓰는가 ?
- 예측 가능한 결과인가 ?
- 적어도 일관된 방법으로 속이는가?
- 시뮬레이션이 망가질까 ?

- Some Crimes are greater than others

단순화

- 단순화
- 더 단순화
- 볼록한 형태



단순화

- Homogeneous bodies
- Rigid bodies
- Indestructible bodies



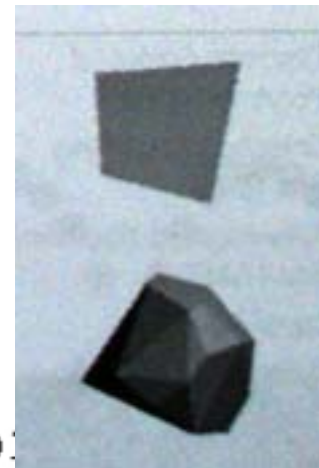
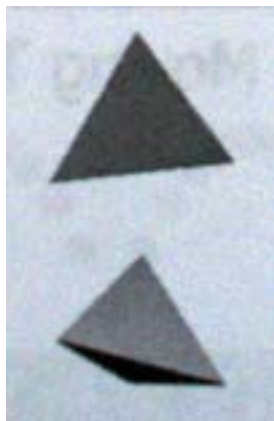
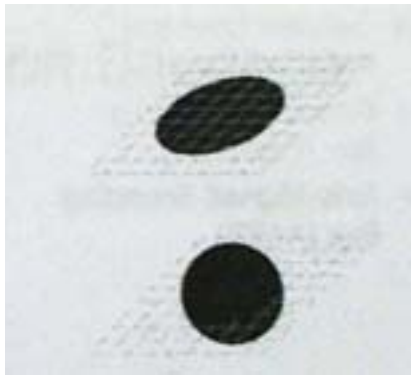
Sp

단순화

- 너무 단순화 한다고 생각되더라도 종종 이
동시 공기저항을 무시되기도 한다
- 충돌 후 반응은 당구공처럼 완전한 탄성을
가지기도 한다.

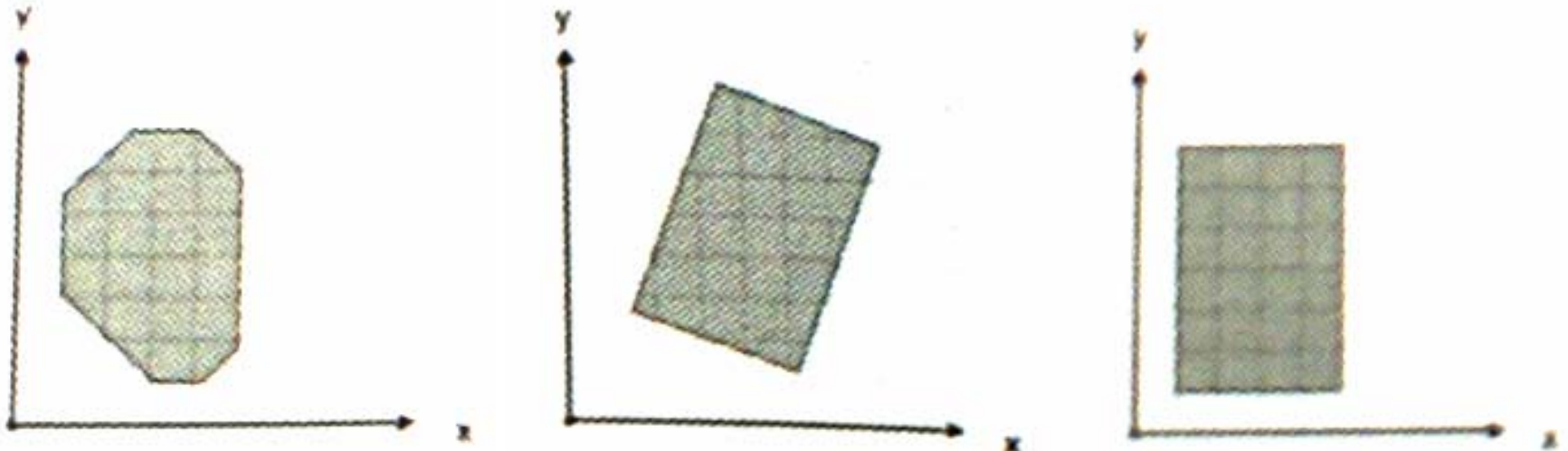
물체의 모양

- 2d:disc, 3d:구
- 2d:삼각형, 3d:사면체
- 2d:볼록다각형, 3d:볼록다면체 ConvexHull, Brush(quake)



물체의 모양

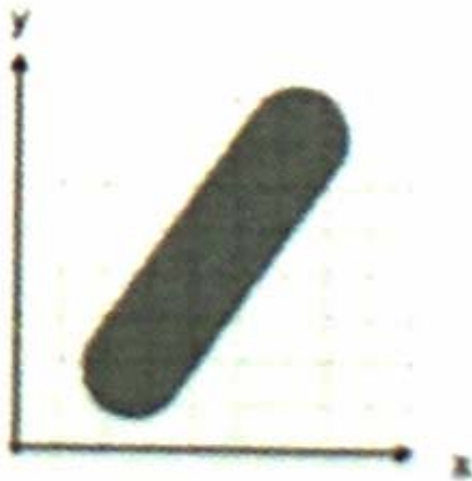
- Discrete Oriented Polytope (DOP)
- Oriented Bounding Box (OBB)
- Axis-Aligned Bounding Box (AABB)



Kasa

물체의 모양

- 캡슐
- 실린더



Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

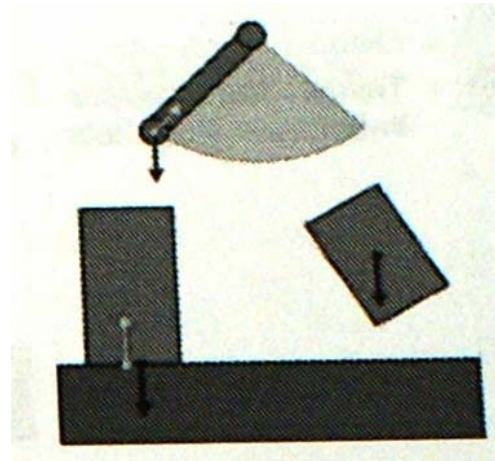
물체의 이동

- 운동학에 의해
위치, 속도, 운동량, 가속도
describes motion



$$X_t = X_0 + Vt + \frac{1}{2}at^2$$

- 동역학에 의해
힘, 임펄스
explains motion

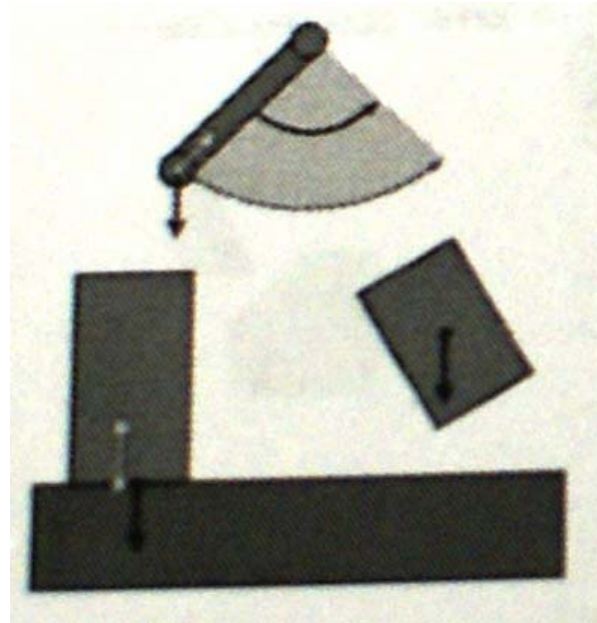


Bit of Flame
RealTime Graphics Programming Study

Kasa

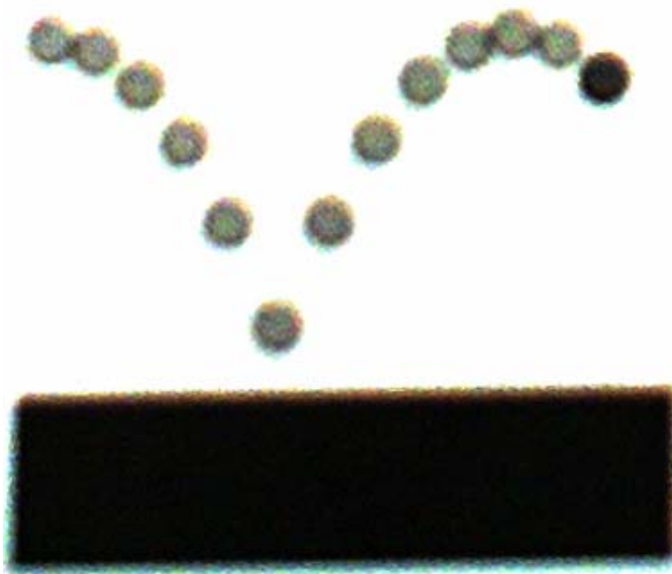
물체의 이동

- 회전
토크(torque)
각운동량
관성 모멘트



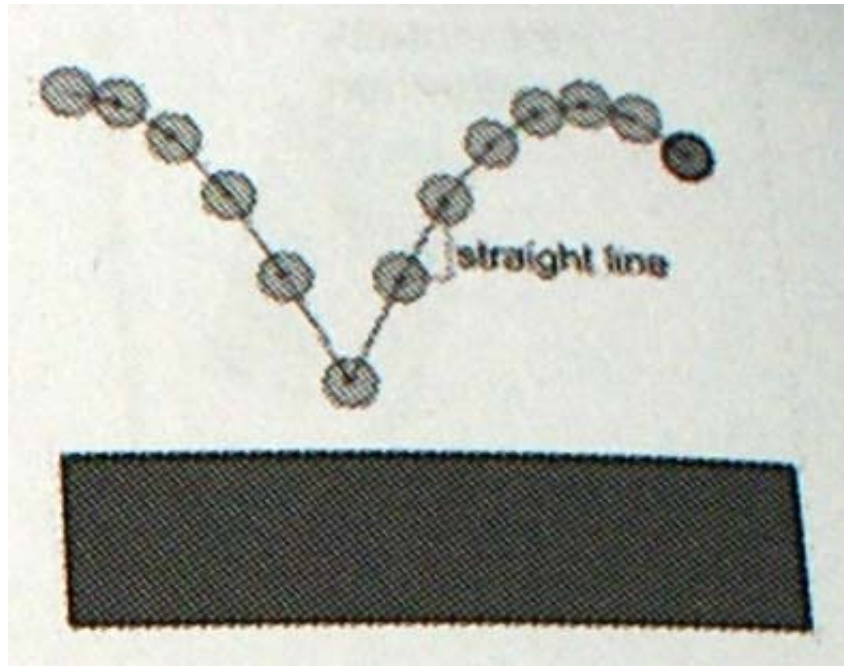
Simulation Baggage

- 연속그림
- 각 프레임 사이에는 무슨 일이 일어나나
- 대개 무언가 일어난다



Simulation Baggage

- 각 궤도는 선형으로 간주할 수 있다

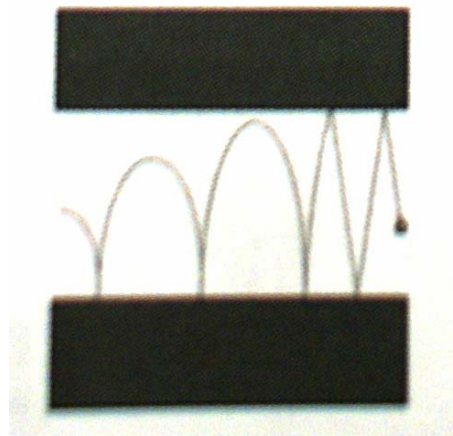
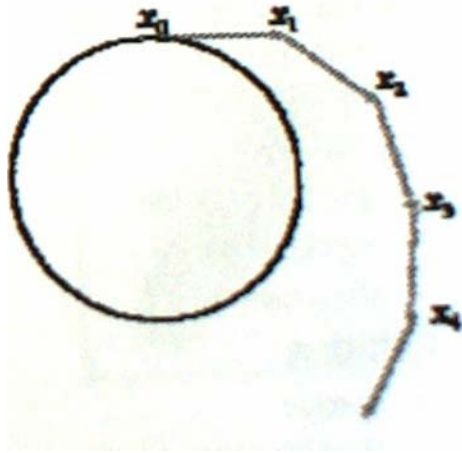


Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

Simulation Baggage

- 오차 누적
- 에너지는 항상 보존되지 않는다
손실을 바람직하지 못하나 증가는 시뮬레이션이 폭발할 수 있다

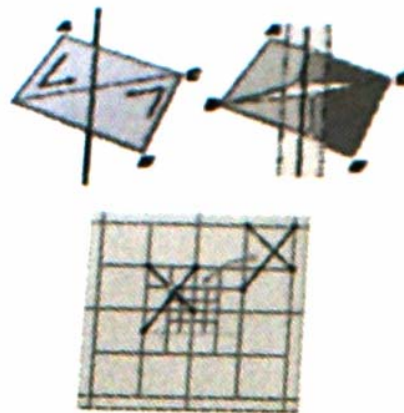
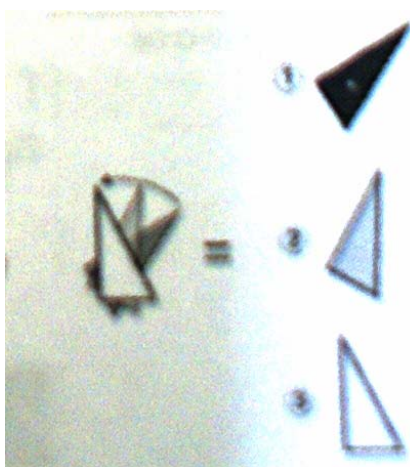


Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

Simulation Baggage

- 회전은 프레임간에 순간적으로 이루어진다.
- 수 처리의 한계

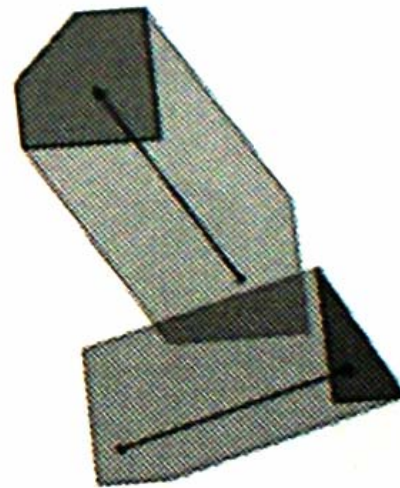


Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

충돌처리

- A와 B가 교차하는 지 검사해야 한다
- 더 나쁜 것은, 움직이는 물체의 것도 검사해야 한다

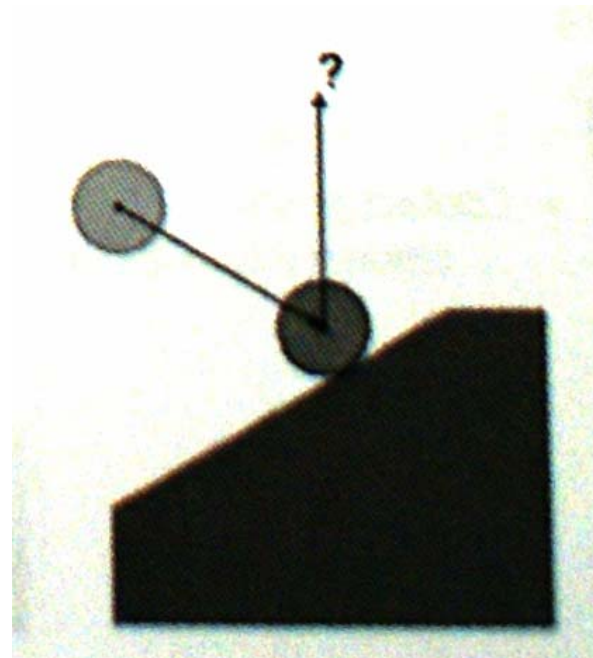
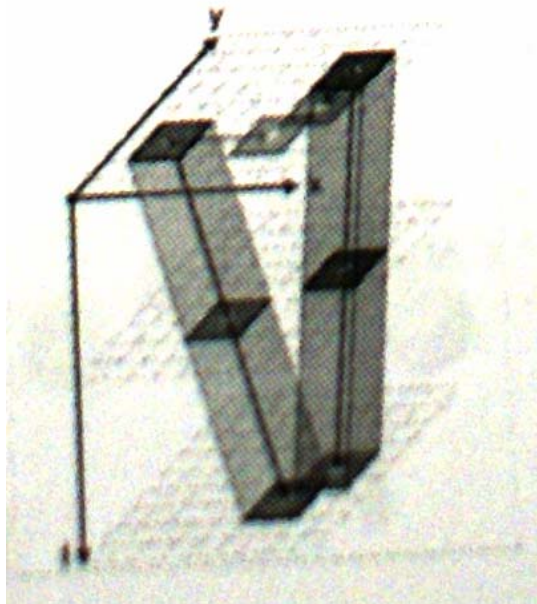


Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

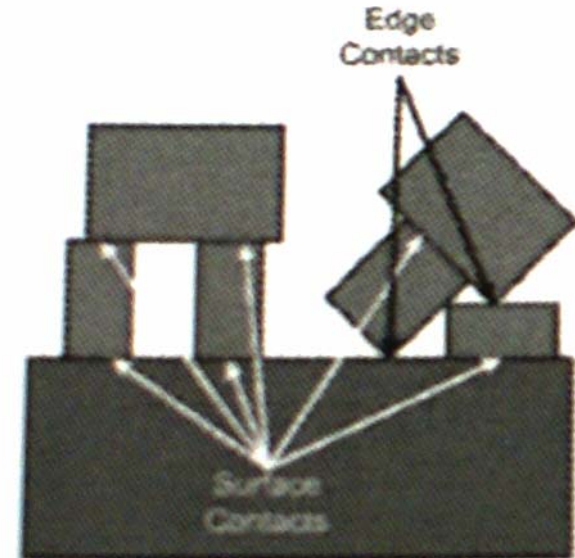
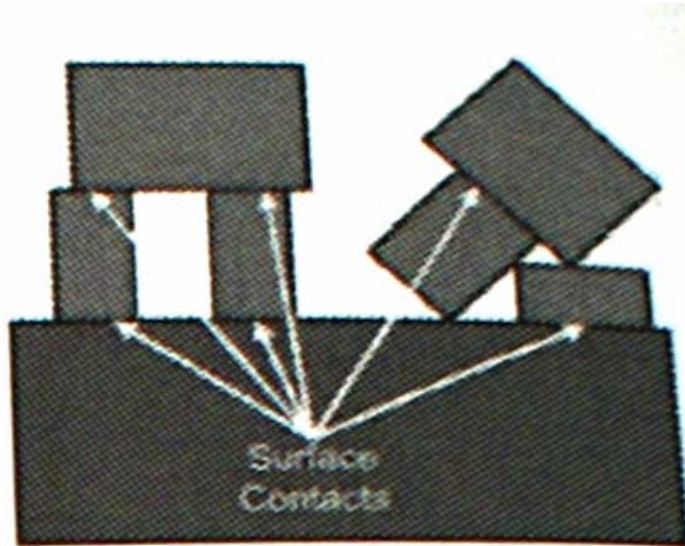
충돌처리

- 언제 충돌되는 지도 알아야 한다
- 충돌 후 어떻게 되는 지도 알아야 한다



지속적인 상호반응

- 접촉표면 (surface contact)
- 가장 자리 접촉 (edge contact)

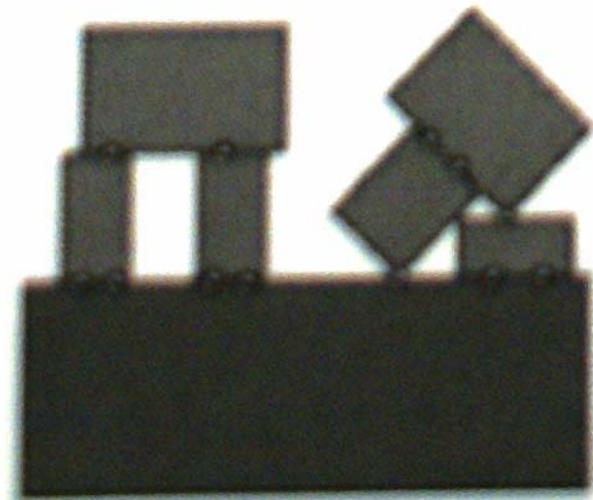
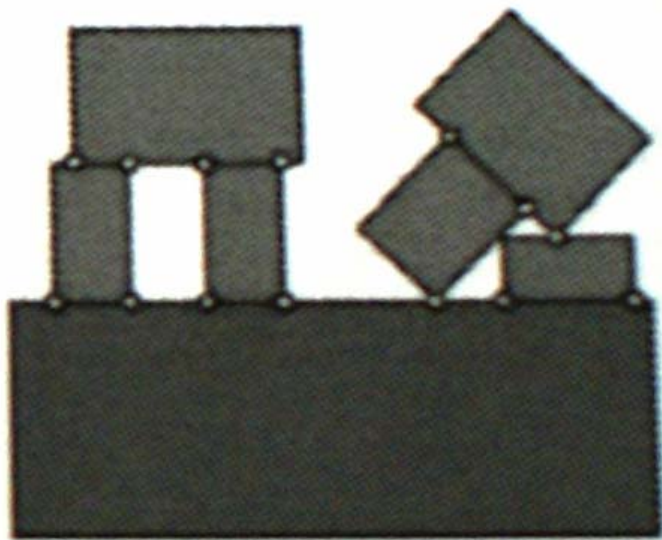


udy

nasa

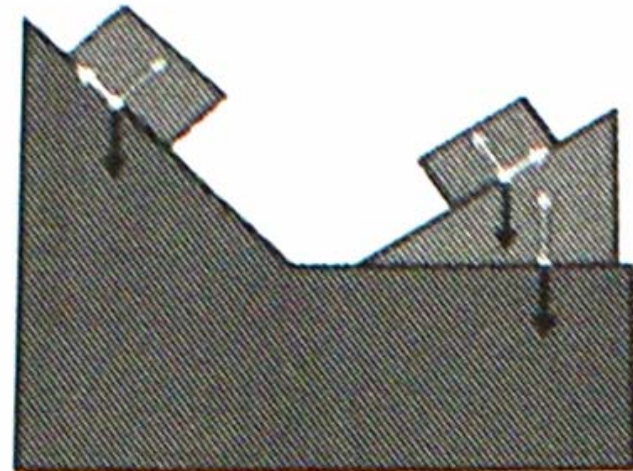
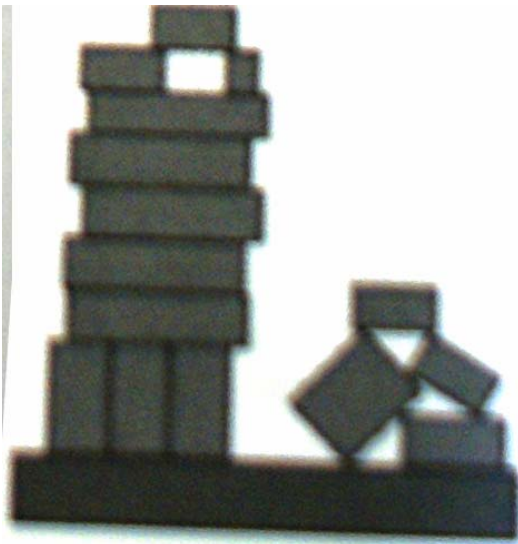
지속적인 상호반응

- 접촉 지점 (contact points)



지속적인 상호반응

- 쌓기 (stacking)
- 마찰 (friction)

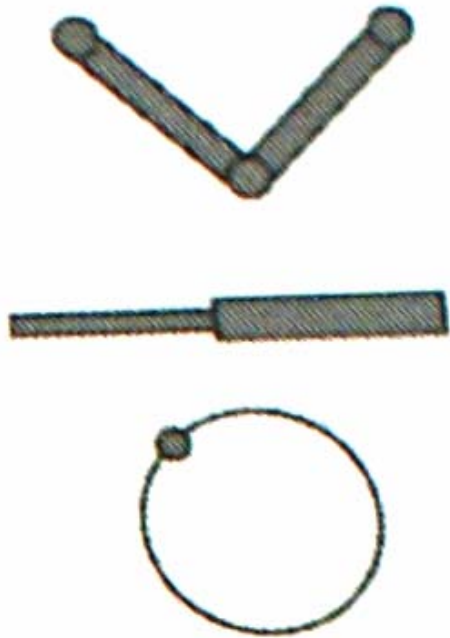


Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

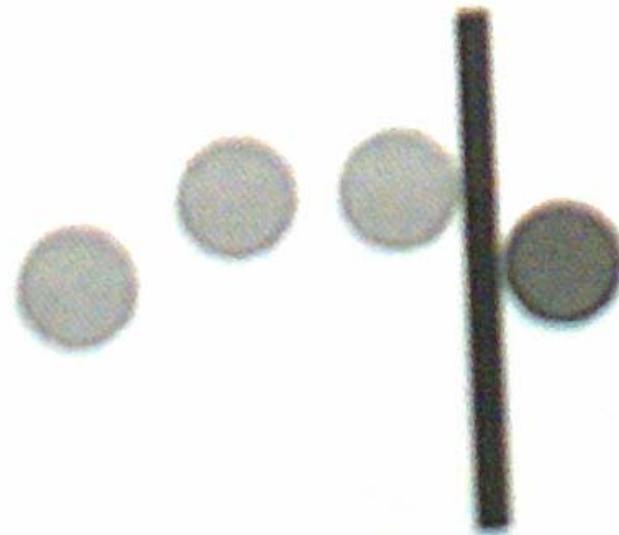
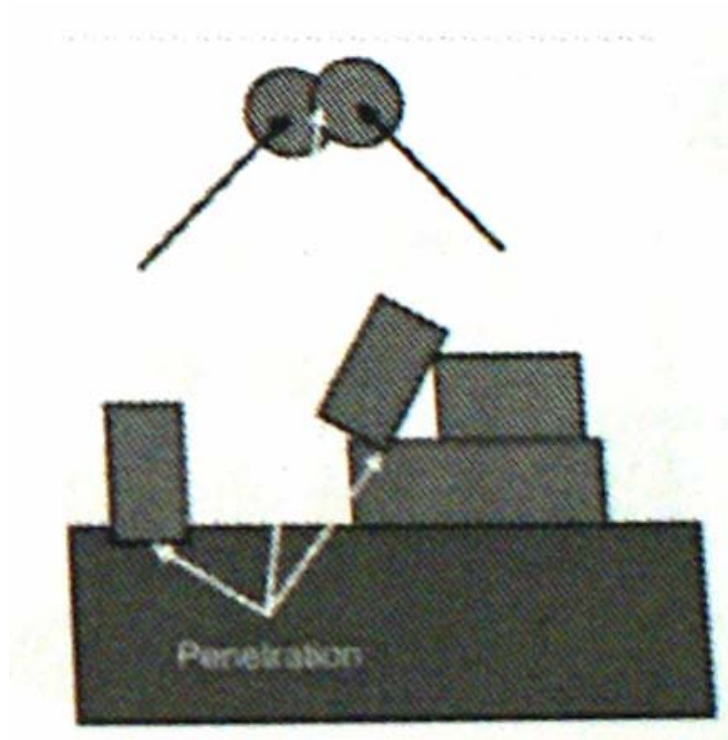
지속적인 상호반응

- 구속과 조인트 (constraints & joints)



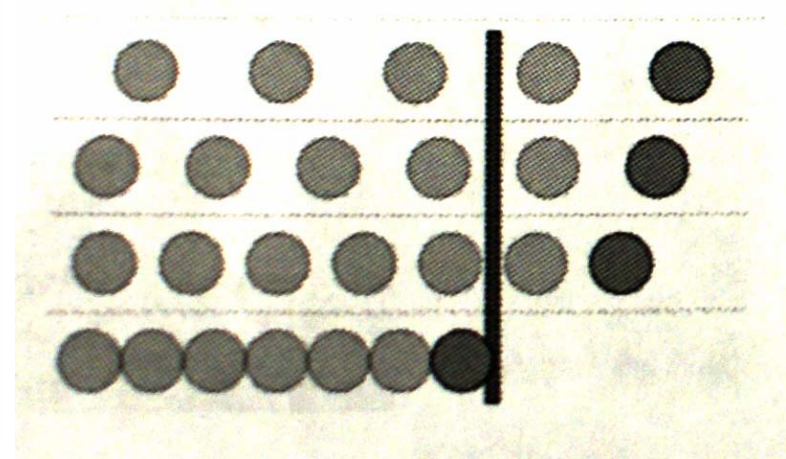
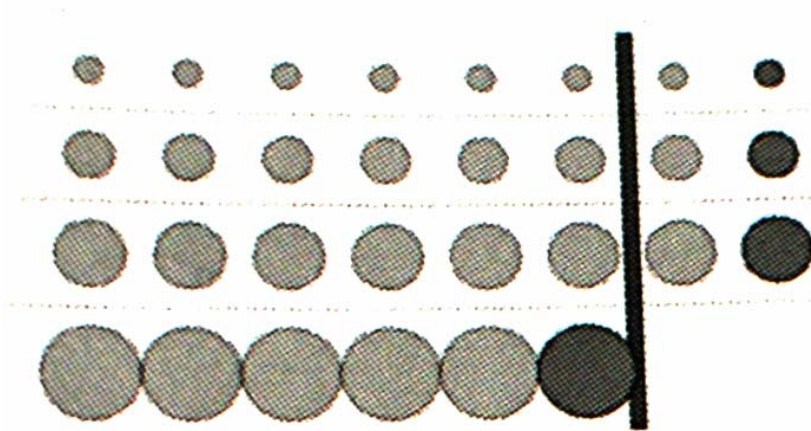
불가능 다루기

- 상호침투(interpenetration)
- 통과 (tunneling)



통과

- 작은 오브젝트가 더 잘 통과한다
최소크기
- 빠른 오브젝트 더 잘 통과한다
속도를 제한, 시뮬레이션 간격을 더 작게

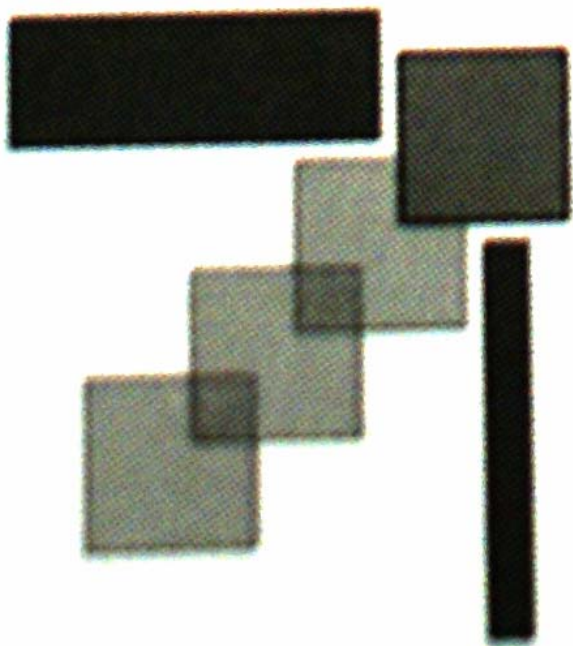


Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

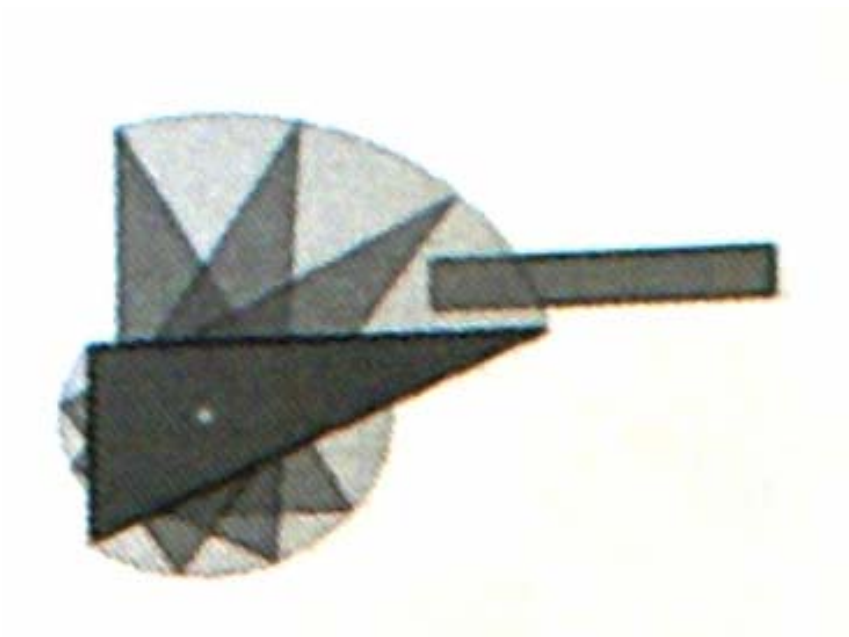
통과

- 최소크기를 정하고, 속도를 제한하고, timestep을 작게 해도 여전히 통과할 수 있다
- Tunneling is very, very bad



통과

- 회전시 통과

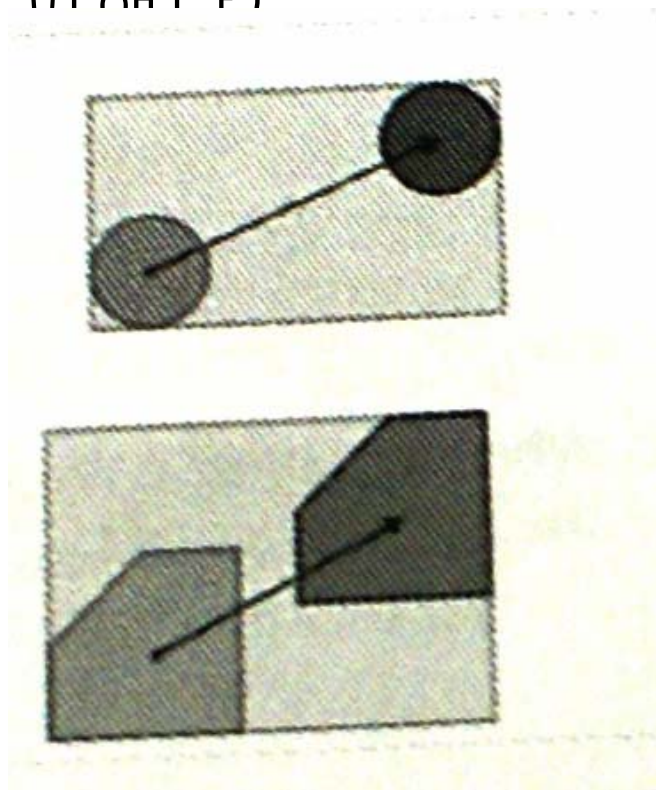


Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

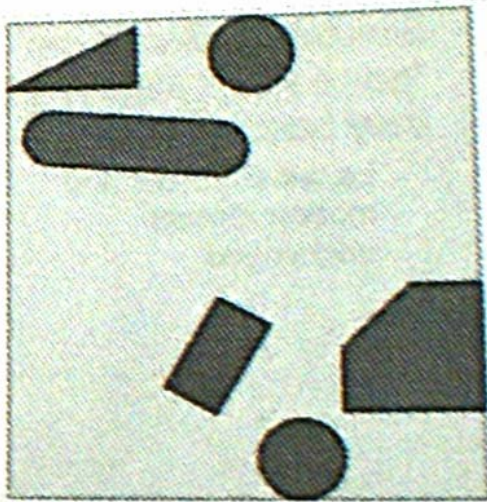
최적화

- 자진해서 모두 계산하지 마라. (불필요한 작업은 피해라)



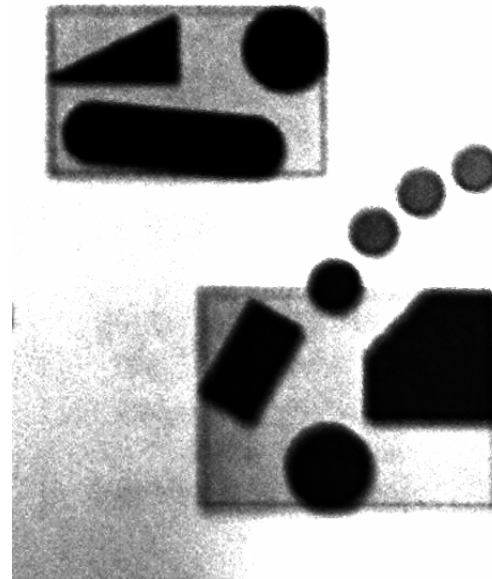
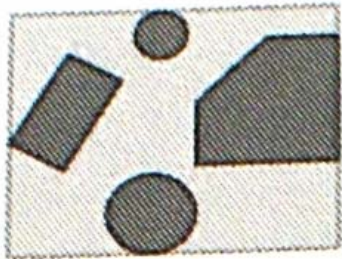
최적화

- 모두 vs 모두는 피해라
- 영역끼리 나눠서 처리한다
1000 오브젝트가 1그룹이면 1000x1000 번 검사
1000 오브젝트가 10그룹이면 10x(100x100) 번 검사



최적화

- 안전화된 그룹은 go to sleep
- 오브젝트가 영역 안에 들어오면 wake up
들어온 오브젝트는 그룹에 포함됨



요약

- How should we choose to represent physical bodies?
- How should we simulate and compute motion?
- How can we prevent energy build-up?
- How do we cope with floating point error ?
- How can we detect collisions – especially when large numbers of objects are involved ?
- How should we resolve penetration ?
- How should we handle contact ?
- How can we prevent tunneling ?
- How do we deal with non-rigid bodies ?

Dynamic 101

Physically Based-Motion

- 게임 오브젝트들이 일괄되게 움직이길 원한다
- 현실의 경험과 비교한다
- 이것은 게임이지만,...
- 너무 비싸도 안된다.

역사:아리스토텔레스

- 오브젝트를 밀면, 멈춘다
돌은 깃털보다 빨리 떨어진다
- 오브젝트는 멈추려 한다.
- 수직운동
- 움직임은 힘을 줄 때 생긴다.
- 무거운 물체가 더 빨리 떨어진다.

- Kinematics

역사II:갈릴레오

- 오브젝트는 멈추려한다
- 포탄은 똑 같이 떨어진다.
- 보이지 않는 힘이 움직임을 느리게 한다. (마찰)
- 움직일 때는 그대로 움직임을 유지하려 한다 (inertia)
- 힘은 위치가 아닌 속도에 영향을 준다
- 속도에 질량은 영향을 주지 않는다
- Dynamics

역사3:뉴턴

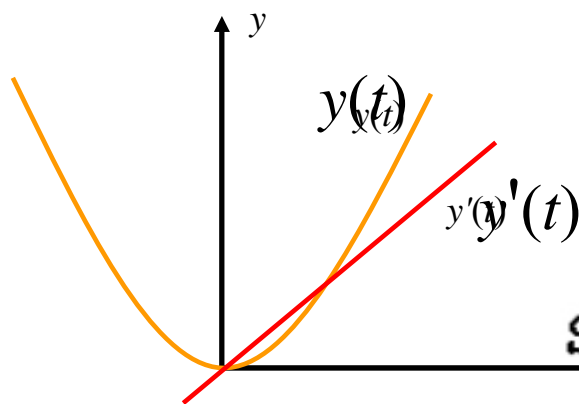
- 행성은 중력을 통해 움직인다
- 행성과 작은 물체는 연결되어있다
- 힘은 무게와 속도와 관련이 있다

- 힘의 집합이 가속이다
- 가속은 속도를 변화시킨다
- 속도는 위치를 변화시킨다

- 미적분을 고안

미적분

- $y(t)$ 함수가 있다
- 함수 $y'(t)$ 는 t 가 바뀔때 y 의 변화를 나타낸다. (dy/dt 로도 표시)
- $y'(t)$ 는 시간 t 에서의 기울기다 \dot{y}



미적분

- 위치 함수 : $\mathbf{x}(t)$
- 미분 값은 속도:

$$\mathbf{v}(t) = \mathbf{x}'(t) = \frac{d\mathbf{x}}{dt} = \dot{\mathbf{x}}$$

- 속도의 미분은 가속도

$$\mathbf{a}(t) = \mathbf{v}'(t) = \frac{d\mathbf{v}}{dt} = \dot{\mathbf{v}}$$

$$= \mathbf{x}''(t) = \frac{d^2\mathbf{x}}{dt^2} = \ddot{\mathbf{x}}$$

Spirit of Flame

3D RealTime Graphics Programming Study

Kasa

뉴턴 기본 물리학

- 모든 오브젝트는 힘에 영향을 받는다
중력
땅 (밀러온린다.)
서로 다른 오브젝트가 서로 미치는 힘
- 힘은 속도를 결정한다 ($\mathbf{F} = m\mathbf{a}$)
- 가속도는 속도를 바꾼다 $\frac{d\mathbf{v}}{dt} = \mathbf{a}$
- 속도는 위치를 바꾼다 $\frac{d\mathbf{x}}{dt} = \mathbf{v}$

뉴턴 기본 물리학

- 가속도가 상수면

$$\frac{d\mathbf{v}}{dt} = \mathbf{a}$$

$$d\mathbf{v} = \mathbf{a}dt$$

$$\int d\mathbf{v} = \int \mathbf{a}dt$$

$$\mathbf{v} = \mathbf{v}_0 + \mathbf{a}t$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}$$

$$d\mathbf{x} = \mathbf{v}dt$$

$$\int d\mathbf{x} = \int \mathbf{v}dt$$

$$\mathbf{x} = \int (\mathbf{v}_0 + \mathbf{a}t)dt$$

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{v}_0t + \frac{1}{2}\mathbf{a}t^2$$

Spirit of Flame

3D RealTime Graphics Programming Study

Kasa

뉴턴 기본 물리학

- 기본 방정식

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

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_i t + 1/2 \mathbf{a} t^2$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \mathbf{a} t$$

- 힘이 모멘텀 \mathbf{P} 의 미분

$$\mathbf{P} = m\mathbf{v} \qquad \frac{d\mathbf{P}}{dt} = m \frac{d\mathbf{v}}{dt} = m\mathbf{a} = \mathbf{F}$$

뉴턴 기본 물리학

- 오브젝트에 가해진 힘들을 합으로 계산한다.
- 가속도를 계산한다.
- 기존의 위치에서 속도와 가속도로 새로운 위치를 계산한다.
- 기존의 속도와 가속도로 새로운 속도를 계산한다.

뉴턴 물리

- 가속도가 상수면 잘 작동한다
- 가속도가 위치나 속도에 영향을 받으면 좋지 않다
- E.g. spring force: $\mathbf{F}_{\text{spring}} = -k\mathbf{x}$
- E.g. drag force: $\mathbf{F}_{\text{drag}} = -m\rho\mathbf{v}$

해석적인 해법

- 해석적인 해법을 찾는다
함수 x 와 v
- 이 경우 간단하게 끌어낸 식은

$$\int -\rho v dt = v_0 e^{-\rho t}$$

- 하지만 항상 해법은 아니다
- 아니면 다른 다양한 미분 함수를 사용해야 한다

수치적 해법

- Problem : 물리적으로 위치와 속도에 따라 힘이 달라진다
- $\mathbf{x}(0) = \mathbf{x}_0, \mathbf{v}(0) = \mathbf{v}_0$ 로 시작
- 오직 아는 것은
$$\dot{\mathbf{x}} = \mathbf{v}(t)$$
$$\dot{\mathbf{v}} = \mathbf{F}(\mathbf{x}(t), \mathbf{v}(t)) / m$$
- 기본 해법 : 오일러 방식

오일러 방식

- 기본아이디어: 기울기를 알고 있다. \dot{x} \dot{v}
- 다음의 계산법을 이용해서

$$\dot{f} = \lim_{h \rightarrow 0} \frac{f(t+h) - f(t)}{h}$$

- h 가 충분히 작다면

$$\dot{f} \approx \frac{f(t+h) - f(t)}{h}$$

오일러 방식

- 다음 처럼 가정할 수 있다

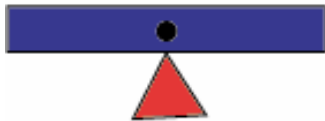
$$f(t+h) \approx f(t) + hf'$$

- 다음의 값은 현재의 값과 현재 미분 값을 알면 알 수 있다.

- 힘은 오브젝트의 무게 중심에 적용된다
- 토크는 오브젝트의 무게 중심과의 위치로부터 적용된다
- 토크도 힘처럼 더한다

무게중심

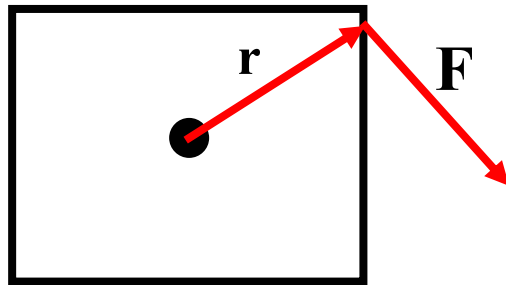
- 하나의 파티클인것 처럼 힘을 줄 수 있는 지점
- 오브젝트의 “균형점 (balance point)”
- 오브젝트의 밀도나 모양은 다양하다



- 무게 중심이 아닌 곳을 밀거나 당기면 토크가 생긴다.

힘 vs 토크

- 토크는 벡터 r (com과 힘을 받은 지점의 벡터)과 힘 벡터 F 의 외적으로 계산한다
- 벡터의 반시계 방향으로 토크가 생긴다



비교

- 힘 F vs 토크
- 운동량 P vs 각 운동량 L
- 속도 v vs 각 속도 ω
- 위치 x vs orientation R
- 무게 m vs 관성 모멘트 I

왜 L ?

- 회전 방정식에서는 각속도를 쉽게 합칠 수가 없다

$$\tau = \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}$$

- 대신, 각 속도는 토크의 합으로 각 운동량을 계산한다
- 운동량(momentum)으로 각속도를 계산한다
- $\mathbf{L} = \mathbf{I}\boldsymbol{\omega}$ 로 각 운동량을 구한다

$$\boldsymbol{\omega} = \mathbf{I}^{-1}\mathbf{L}$$

관성 모멘트

- 관성 모멘트는 m 처럼 하나의 스칼라 값이 아니라 3×3 행렬로 표현된다
- 모양에 따라 회전 요소는 영향을 받는다
- 다양한 축에 어떻게 오브젝트가 회전하는지를 묘사한다
- 계산이 항상 쉬운 것은 아니다
- Orientation 변하듯이 변한다

I 구하기

- 박스나 실린더를 모멘트로 사용할 수 있다
- 구를 사용할 수 있다 (하나의 값 $2mr^2/5$)
- 한 축으로 회전시킨 형태

- 지형 정보로 구할 수도 있다
- 각 정점의 밀도와 무게가 일정하다고 가정
- 속이 꼭찬 모양

- See Mirtich, Eberly for more details

월드 좌표계에서 l 사용하기

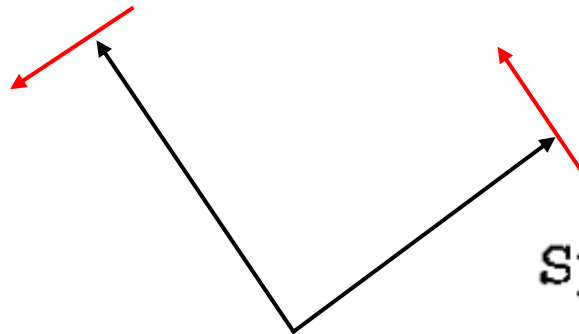
- $\omega = \mathbf{I}^{-1}\mathbf{L}$
- 로컬 좌표계로 계산된 l는 월드좌표계로 트랜스폼해야 한다
- 만약 회전 매트릭스 R을 사용한다면, 다음 식을 사용한다

$$\mathbf{I}_i^{-1}\mathbf{L}_i = \mathbf{R}_i\mathbf{I}_0^{-1}\mathbf{R}_i^T\mathbf{L}_i$$

- 쿼터니언을 사용한다면 매트릭스로 변환한다

새로운 Orientation 구하기

- 매트릭스 $R(\text{Orientation})$ 과 벡터 ω (각속도)
- 어떻게 적용시켜야 하나
- ω 를 R 에 적용할 수 있도록 변환한다
기저벡터의 선형적인 속도로 변환한다
각 기저는 3×3 행렬로 표현
오일러 방법을 이용할 수 있다
- 예



새로운 Orientation 구하기

- $\omega \times r$ 는 r 에서 선형적인 속도
- 각 기저 벡터에 대해서 적용
- 더 좋은 방법
왜대칭 (symmetric skew) 행렬을 외적 구하는 데 사용한다
orientation 매트릭스에 곱한다
- Orientation이 행렬 R 이라면

$$\mathbf{R}_{i+1} = \mathbf{R}_i + h\tilde{\omega}_i \mathbf{R}_i$$

$$\tilde{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}$$

새로운 Orientation 구하기

- Orientation이 쿼터니언 q 라면

$$\mathbf{q}_{i+1} = \mathbf{q}_i + \frac{h}{2} \mathbf{w}_i \mathbf{q}_i$$

$$\mathbf{w}_i = (0, \boldsymbol{\omega}_i)$$

- See Baraff or Eberly for derivation
- $\mathbf{w}\mathbf{q}$ 를 매트릭스 $\mathbf{Q}\boldsymbol{\omega}$ 곱으로 표현한다면

$$\mathbf{Q} = \begin{bmatrix} -x & -y & -z \\ w & z & -y \\ -z & w & x \\ y & -x & w \end{bmatrix}$$

- $\mathbf{q} = (w, x, y, z)$

회전 공식

$$\boldsymbol{\tau} = \sum_k \mathbf{r}_k \times \mathbf{F}_k$$

$$\mathbf{R}_{i+1} = \mathbf{R}_i + h\tilde{\boldsymbol{\omega}}_i \mathbf{R}_i$$

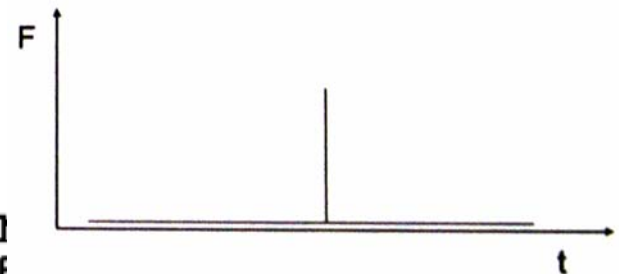
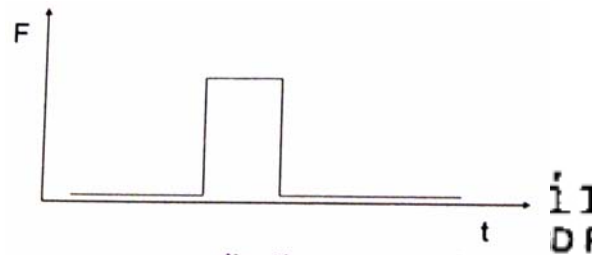
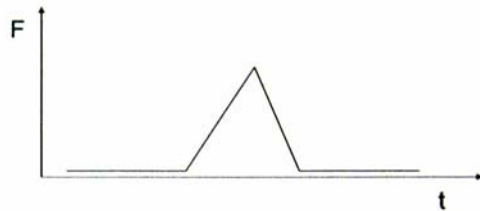
$$\mathbf{L}_{i+1} = \mathbf{L}_i + h\boldsymbol{\tau}$$

$$\mathbf{I}_{i+1}^{-1} = \mathbf{R}_{i+1} \mathbf{I}_0^{-1} \mathbf{R}_{i+1}^T$$

$$\boldsymbol{\omega}_{i+1} = \mathbf{I}_{i+1}^{-1} \mathbf{L}_{i+1}$$

충격 (impulse)

- 힘은 일정시간 동안 움직이게 한다
eg. 책상밀기
- 일정하게 힘을 주더라도 전체에 영향을 준다
- 순간적인 변화라면 ?? (impulse)
충돌, 구속 등에 사용하기 좋다



Kasa

요약

- 기본 뉴턴 역학
위치, 속도, 힘, 모멘텀
- 선형 시뮬레이션
힘 → 가속도 → 속도 → 위치
- 회전 시뮬레이션
코트 → 각 모멘텀 → 각 속도 →
orientation

More Tutorials

- Numeric Integration Methods
오일러 방식으로 인한 문제와 해결
(Runge-Kutta, Implicit Methods, Implicit Methods, Verlet Methods, Symplectic Methods)
- Numerical Robustness
부동 소수점 오차로 인한 문제와 해결

More Tutorials

- Reframing the Problem
충돌, 터널링 등에 대한 문제와 대세(?)
(Minkowski Differences 소개)
- Collisions using separating axis-tests
- Collision Detection
충돌 처리 (GJK알고리즘, AABB 트리,
Sweep 테스트)

More Tutorials

- Modeling and Solving Constraints
조인트, 구속
- Contact Manifolds
접촉점 계산 (SAT, GJK)
- Bending, Breaking and Squishing Stuff
소프트바이 다이내믹스

More Tutorials

- Breaking Stuff
dynamic destruction
- Squishing Stuff
dynamic deformation

질문

Spirit of Flame
3D RealTime Graphics Programming Study

Kasa