Center for Computer Graphics
and Virtual Reality, Ewha

e-Entertainment를 위한 실시간 물리 시뮬레이션 기술

이화여자대학교 김영준

kimy@ewha.ac.kr

Slides

- http://graphics.ewha.ac.kr/HCI2010.pdf

실시간 물리 시뮬레이션

개념: 객체들 간의 상호작용이 현실 세계의 물리 법칙을 따르도록 실시간에 시각적으로 시뮬레이션 하는 기술

강체

관절체 (ragdoll)

Particle

차량

물리 엔진 응용 게임

마비노기 영웅전, Nexon

Forza Motorsport 3, Microsoft

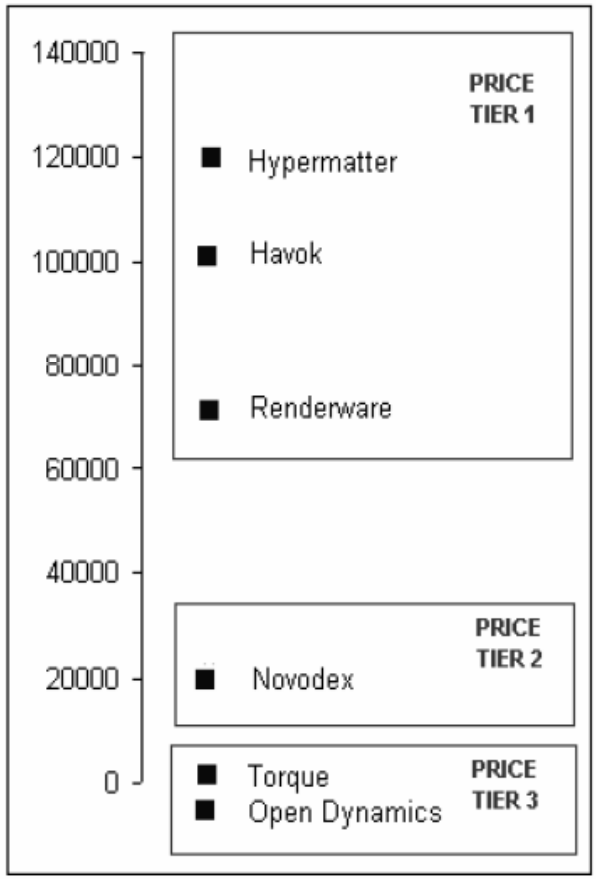
허슬 당구, Dream Arrow

Fight Night Round 4, EA

기술 동향 – 대표적 물리 SW

개발회사명	상품명	형태	국가명	역사
HAVOK	HAVOK	SW ($100K)	아일랜드	1998년에 **Trinity대학의 연구 그룹에** 의해서 시작된 대표적인 실시간 물리 시뮬레이션 개발 회사로, 영국과 미국에 분소가 있으며, **2007년에 Intel에 인수됨**
AGEIA	PhysX	SW/HW ($20K)	스위스	1995년에 **스위스의 ETH공대의 연구 그룹에** 의해서 시작된 회사로, 초기에는 NovodeX라는 프로덕트로 알려져 있었음. 2005년에 Meqon 물리엔진 인수. **2008년 nVIDIA에 인수됨**
Open Dynamics Engine	Open Dynamics Engine	SW (무료)	미국	2000년에 Russel Smith에 의해 시작된 개인 프로젝트. **강체/관절체 시뮬레이션등 제한적 기능**.
BULLET	BULLET	SW (무료)	미국	Sony Playstation Research의 Erwin Coumans에 의해서 유지되는 물리 시뮬레이션 소프트웨어. **강체/관절체 시뮬레이션등 제한적 기능**.

기술 동향 – SW 가격



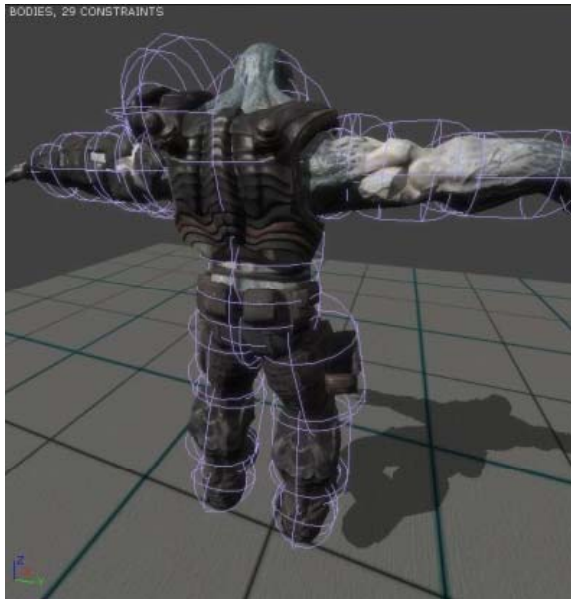
- 가격대
  1. 9천만 원~1억5천만 원
  2. 2천만 원~3천만 원
  3. 10만 원 이하

[Saral and Schmieder 2004]

차세대 물리엔진개발의 필요성

- 흥행 게임 타이틀 필수 요소
- 렌더링 퀄러티와 인터랙션 퀄러티의 격차

고품질 렌더링 결과
(Unreal3 엔진)

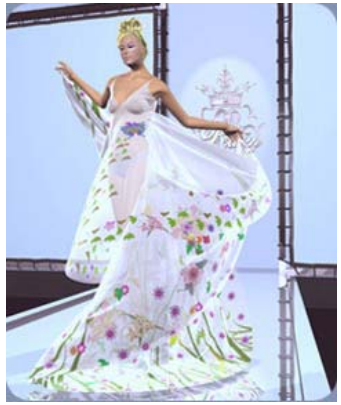


단순화된 물리 시뮬레이션

차세대 물리엔진개발의 필요성

- 비 실시간 위주의 국내 기존 연구

유체, ETRI



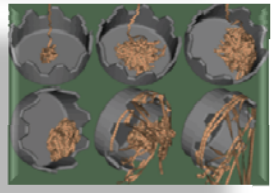
Qualoth, FxGear

- 멀티 코어 CPU, GPU, PPU등의 차세대 하드웨어 플랫폼의 등장

실시간 (30 FPS) VS 비실시간 물리 엔진

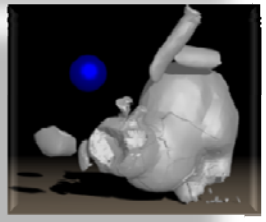
관절체: 3초 / frame

100X 속도 향상 필요



Cloth (리본): 3~30 초 / frame

1000X 속도 향상 필요



깨진 토끼: 1 분/ frame

2000X 속도 향상 필요

유체: 10일

☺

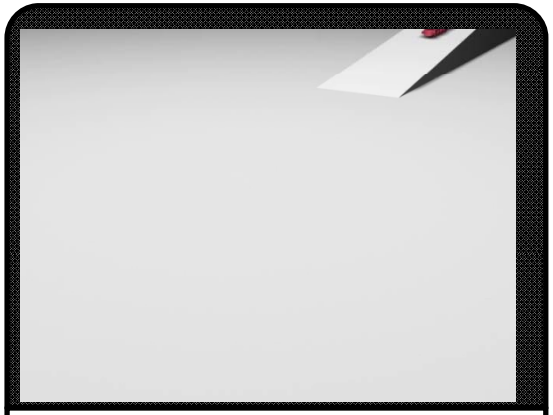
차세대 물리엔진개발의 필요성

- 한국적 상황에 맞는 물리 엔진 제작 필요성
  - 모바일, light-weight, customization

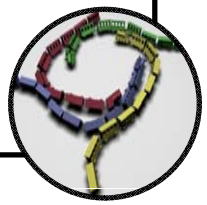
- 모든 영역에 최적화된 물리 엔진은 존재안함
  - Havok, PhysX가 cover하지 못하는 영역이 존재
  - 예: Rockstar games의 GTA4

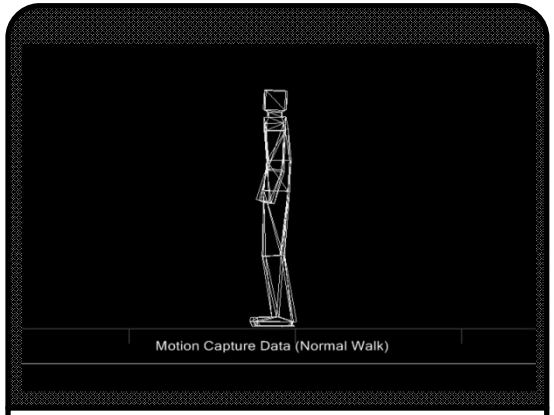
웍샵 발표 내용

차세대 물리 시뮬레이션 기술 개발



동역학 시뮬레이션





데이터 기반
물리 시뮬레이션

변형 시뮬레이션

발표 순서

1:30 pm – 2:40 pm	웍샵 소개
1:40 pm – 2:15 pm	실시간 강체 및 관절체 충돌검사 (김영준, 이화여대)
2:15 pm – 2:50 pm	실시간 강체 동역학 시뮬레이션 (김진욱, KIST)
2:50 pm – 3:00 pm	휴식
3:00 pm – 3:35 pm	물리기반 캐릭터 시뮬레이션 (이윤상/이제희, 서울대)
3:35 pm – 4:10 pm	대용량 변형 모델 실시간 충돌검사 (윤성의, KAIST)
4:10 pm – 4:20 pm	휴식
4:20 pm – 4:55 pm	실시간 유연체 시뮬레이션 (최민규, 광운대)
4:55 pm – 5:15 pm	Space Foosball: 콘텐츠 제작 사례 (방현우, 서울대)
5:15 pm – 5:25 pm	Q&A

Real-time Collision Detection for Rigid and Articulated-Body Dynamics

이화여자대학교　김영준

http://graphics.ewha.ac.kr

Why Proximity Calculation?

- Contact dynamics
  1. Collision detection
  2. Collision response

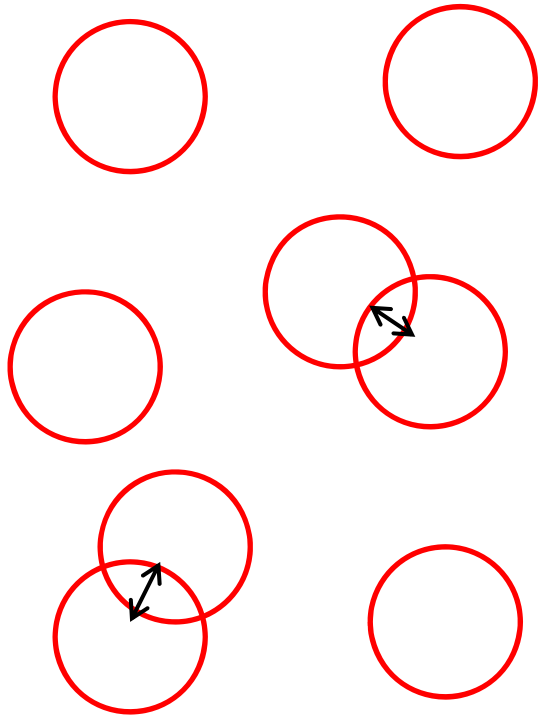
Collision Processing Loop in Contact Dynamics

- Broad phase: n-body collision detection

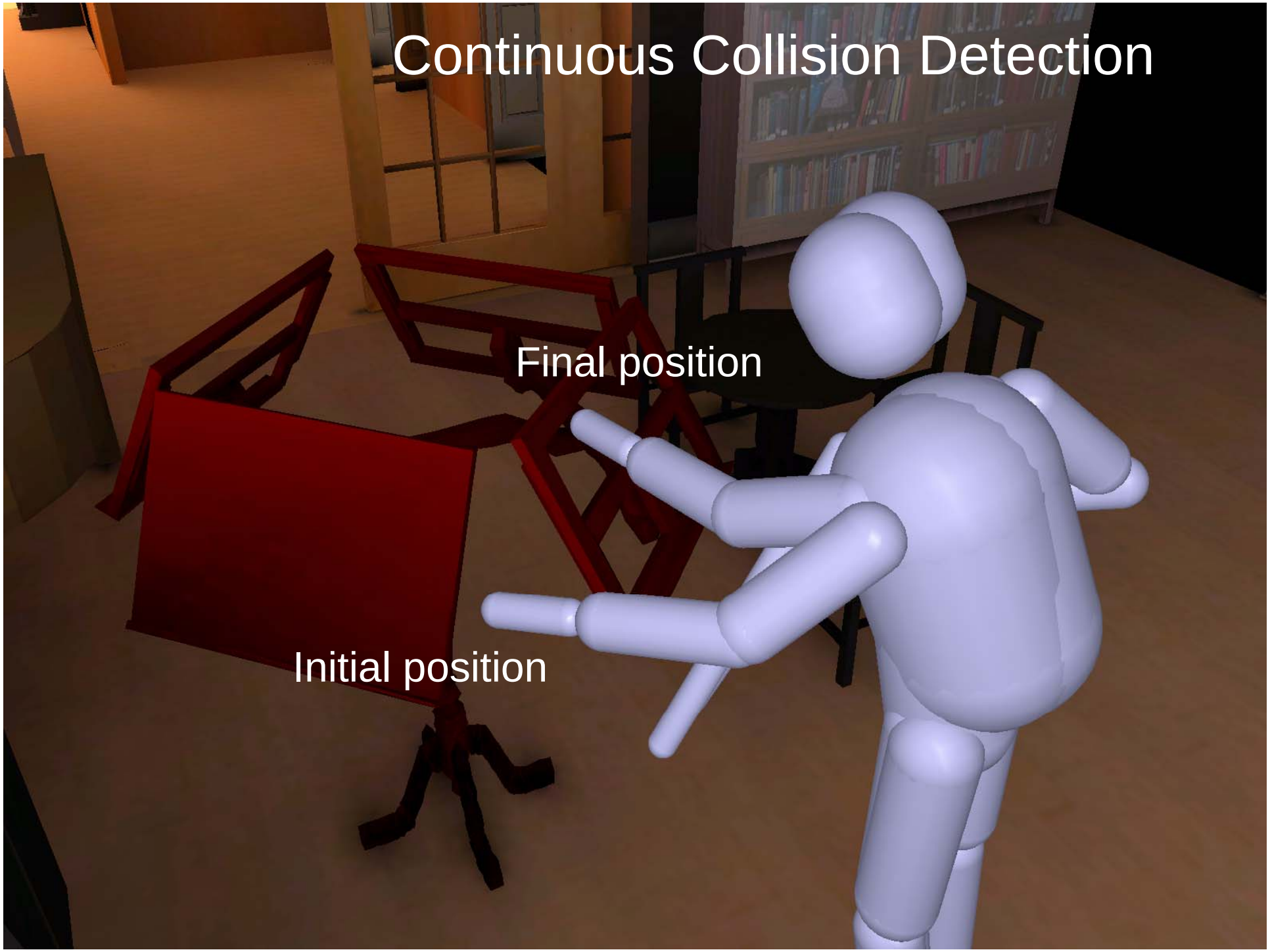
  GPU-based n-body collision detection

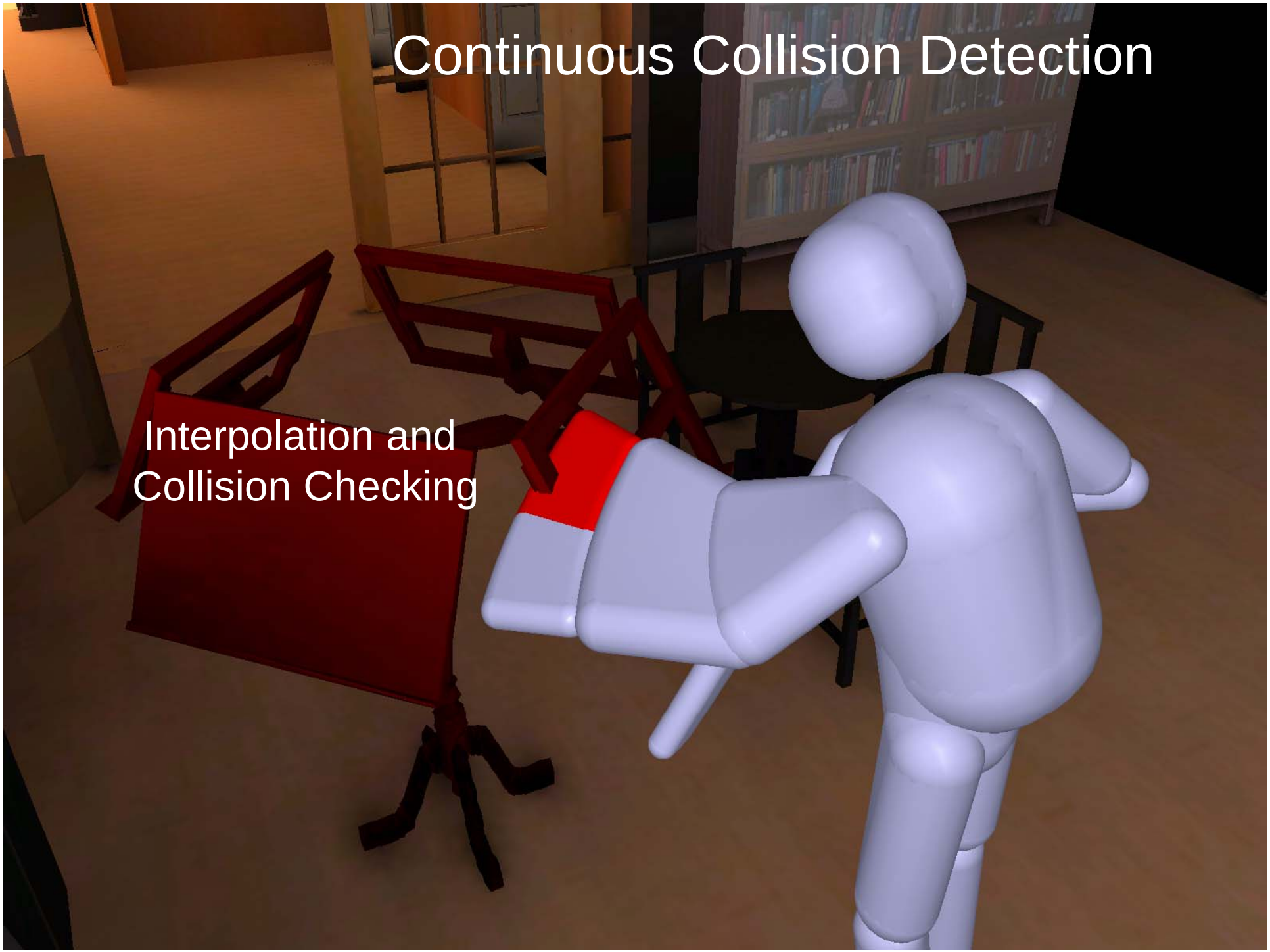
- Narrow phase:
  1. Collision detection
     - Collision point, normal
     - *Penetration depth*
     - *Continuous CD*

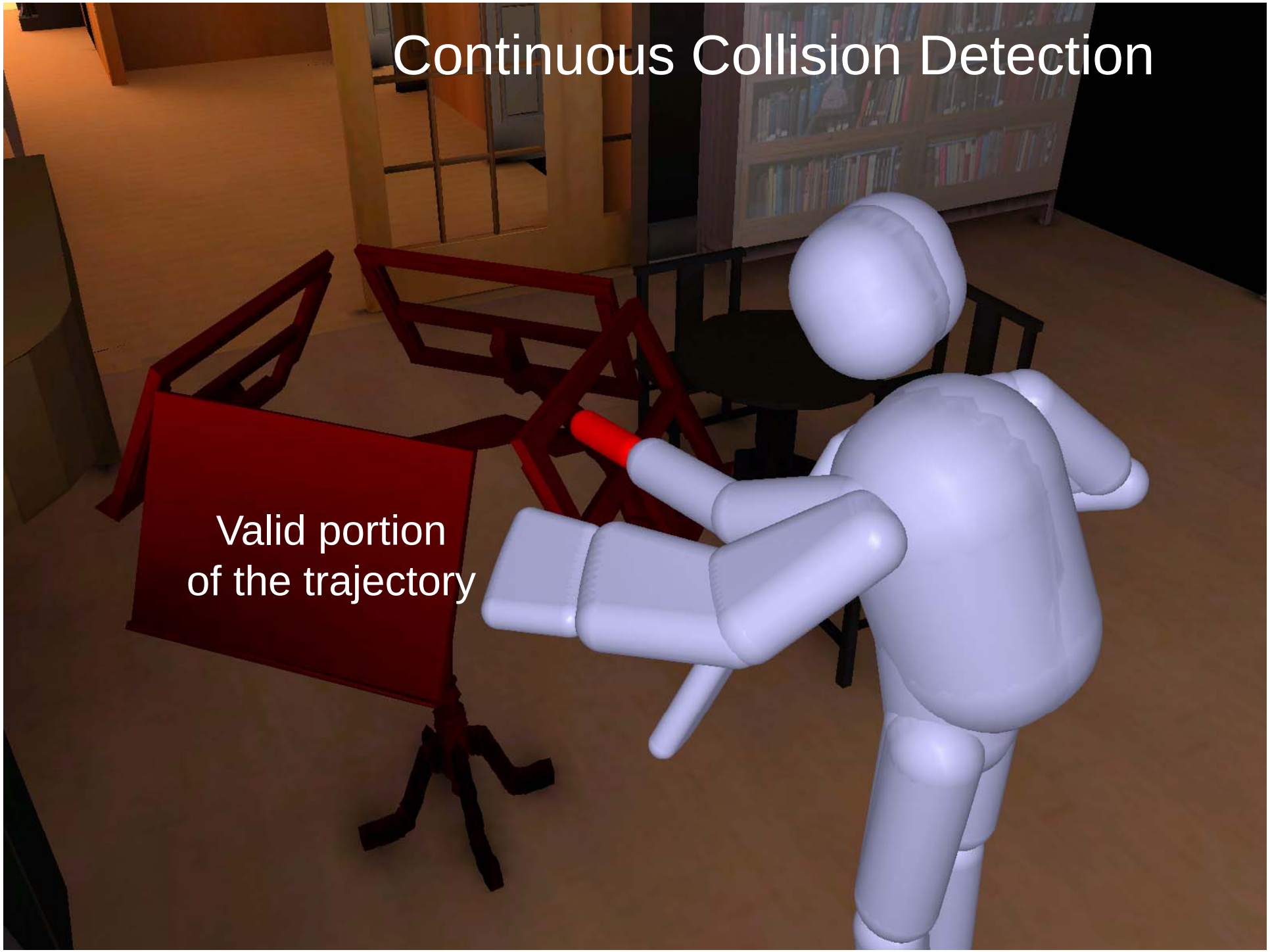
  Real-time penetration depth
  Real-time continuous collision detection
  Parallel collision detection



REAL-TIME CONTINUOUS COLLISION DETECTION

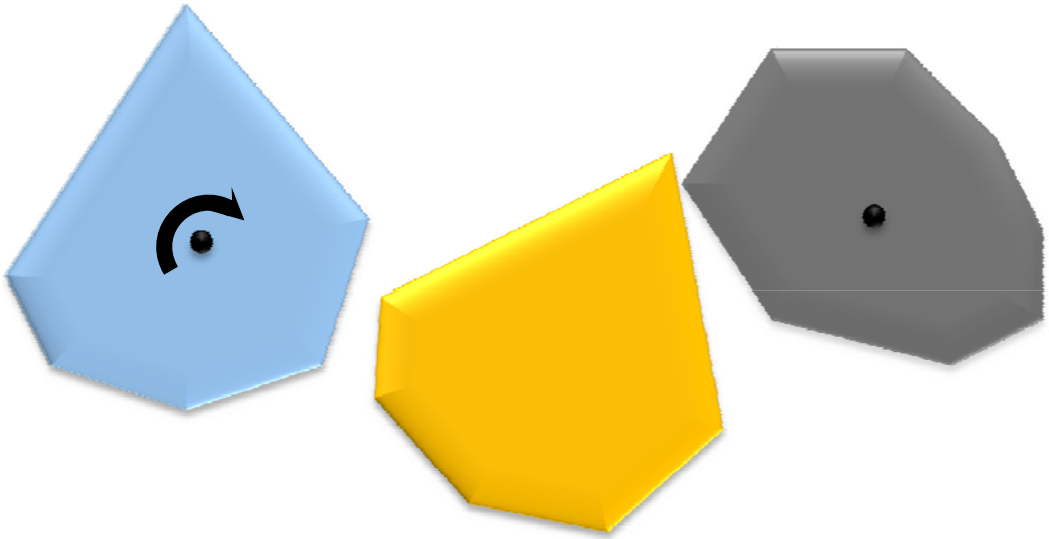

Continuous Collision Detection
Final position
Initial position


Continuous Collision Detection
Interpolation and
Collision Checking


Continuous Collision Detection
Valid portion
of the trajectory

Conservative Advancement (CA)

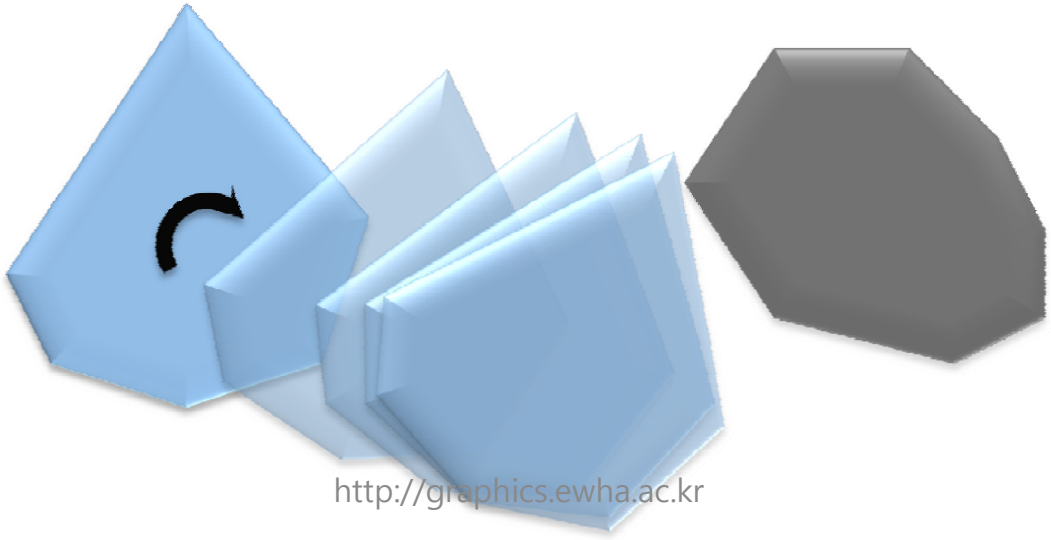
- Assume objects are *convex*
- Find the 1st time of contact (ToC) of a moving object



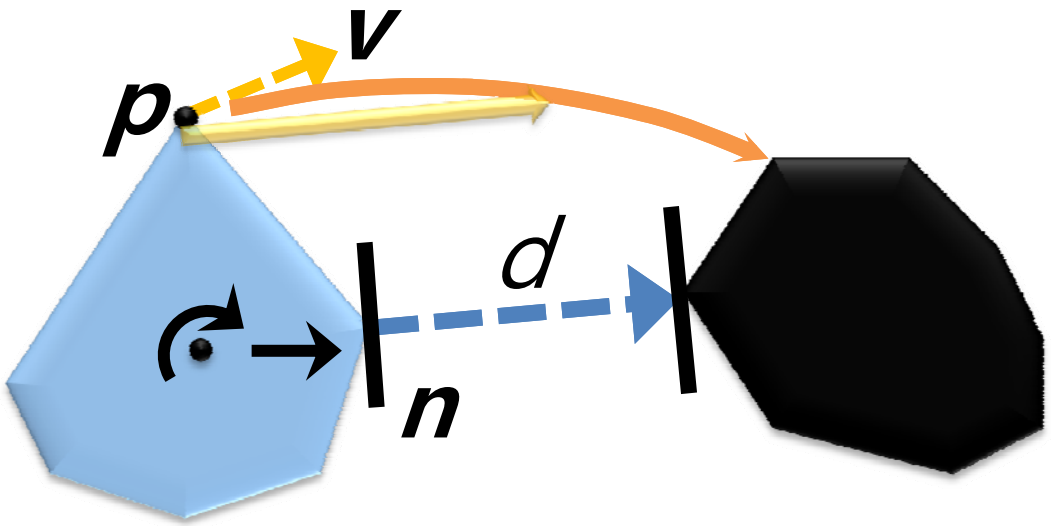
Conservative Advancement (CA)

1. Find a step size Δt_i to conservatively advance the object without collision
2. Repeat until inter-distance < ε

$$ToC = \Delta t_1 + \Delta t_2 + \Delta t_3 + \Delta t_4$$



Calculating Δt in CA



d, ***n***: closest distance, direction vector

v: velocity

$$\int_0^{\Delta t} \left| \mathbf{v}(t) \cdot \mathbf{n}(t) \right| dt \le \max(|\mathbf{v}(t)|) \int_0^{\Delta t} \left| \mathbf{n}(t) \right| dt = \mu$$

Calculating Δt in CA

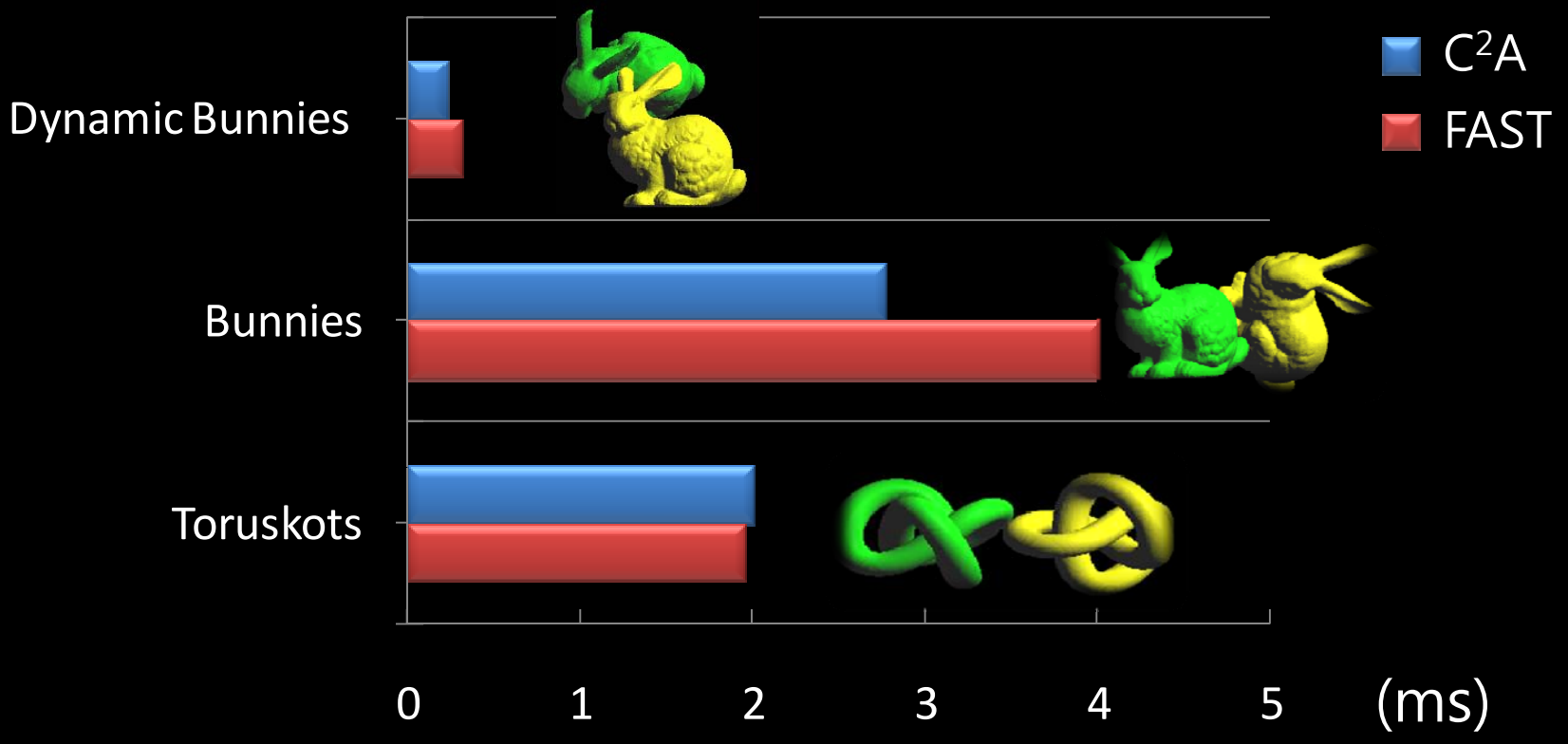
$$\mu \Delta t \le d \qquad \therefore \Delta t \le \frac{d}{\mu}$$

$$\int_0^{\Delta t} |\, v(t) \bullet n(t) \,|\, dt \le \int_0^{\Delta t} \max(|\, v(t) \bullet n(t) \,|)\, dt \le d$$

$= \mu$

FAST: 2-Manifold Models

C²A: Polygon Soup Models



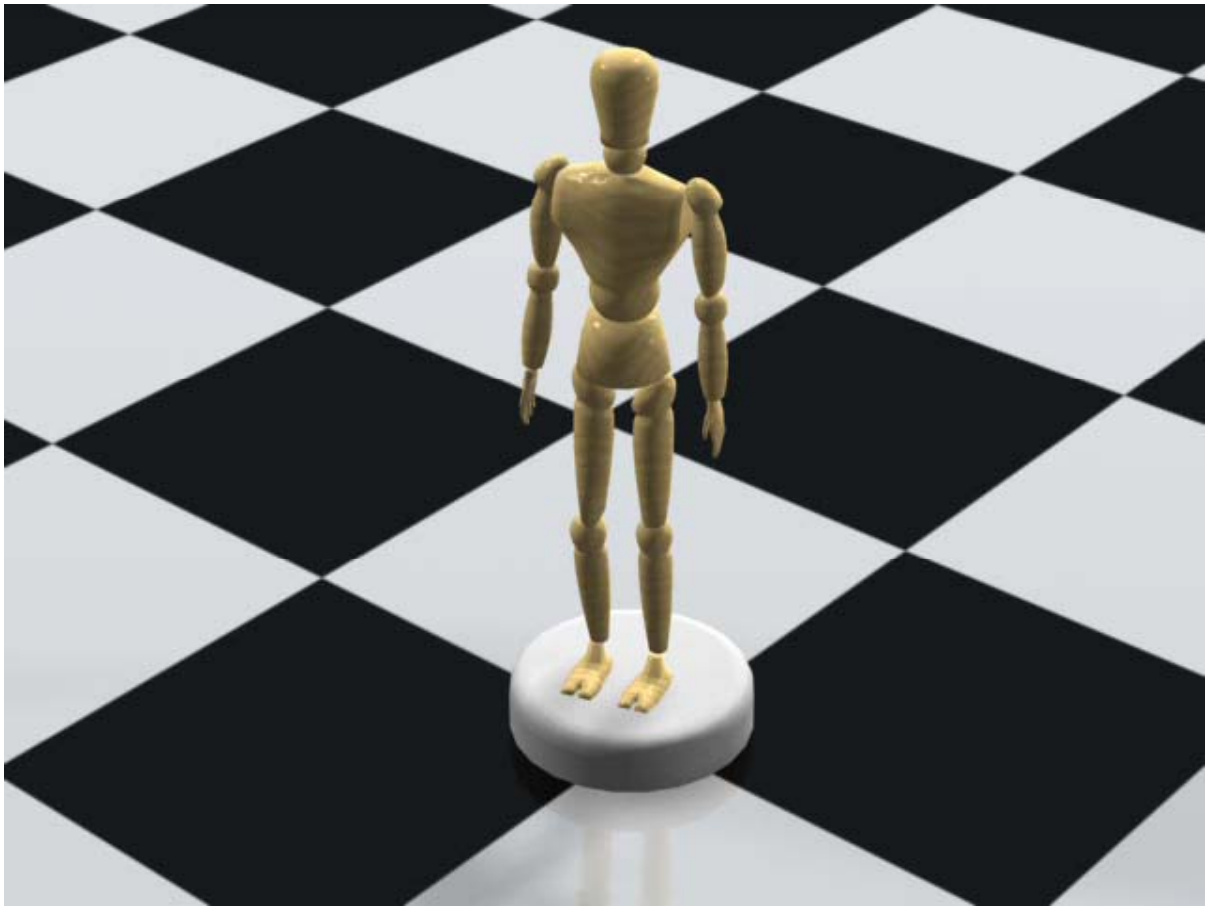
- FAST can handle only manifold surfaces

Articulated Models (Demo)

- **CCD performance**
  - **1.22 msec**
- Mannequin
  - 15 links, 20K tri
- Obstacles
  - 101K tri
- Locomotion SW
  - Footstep™

Articulated Models

- Mannequin
  - 15 links, 20K triangles
- **Self-CCD performance**
  - **0.38 msec**



Articulated Body Dynamics Benchmark

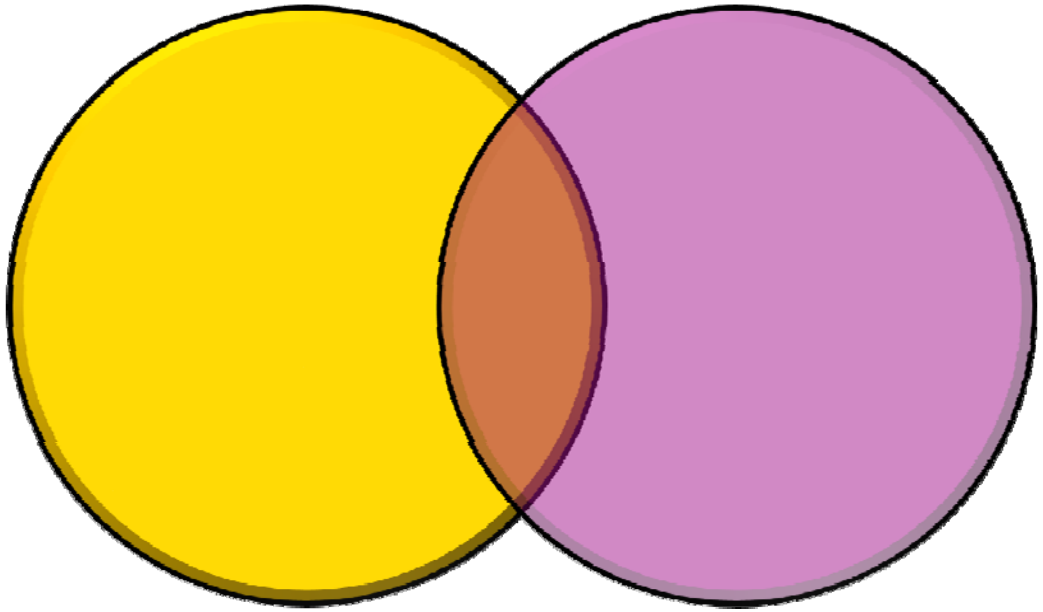
- Four trains
  - 10 links, 23K tri (each)
- **CCD performance**
  - **535 msec**

Software Implementations

- Source codes:
  - http://graphics.ewha.ac.kr/FAST (2-manifold)
  - http://graphics.ewha.ac.kr/C2A (polygon soups)
  - http://graphics.ewha.ac.kr/CATCH (articulated models)

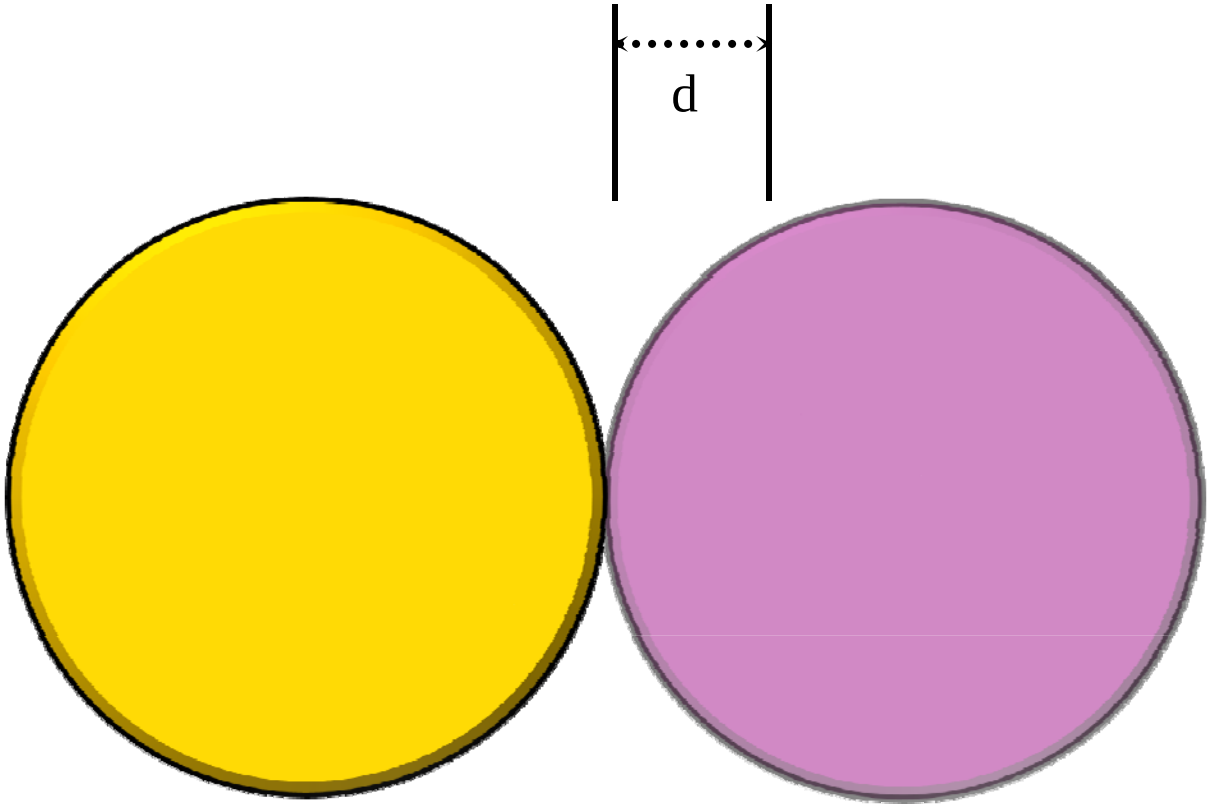
REAL-TIME PENETRATION DEPTH COMPUTATION

Penetration Depth



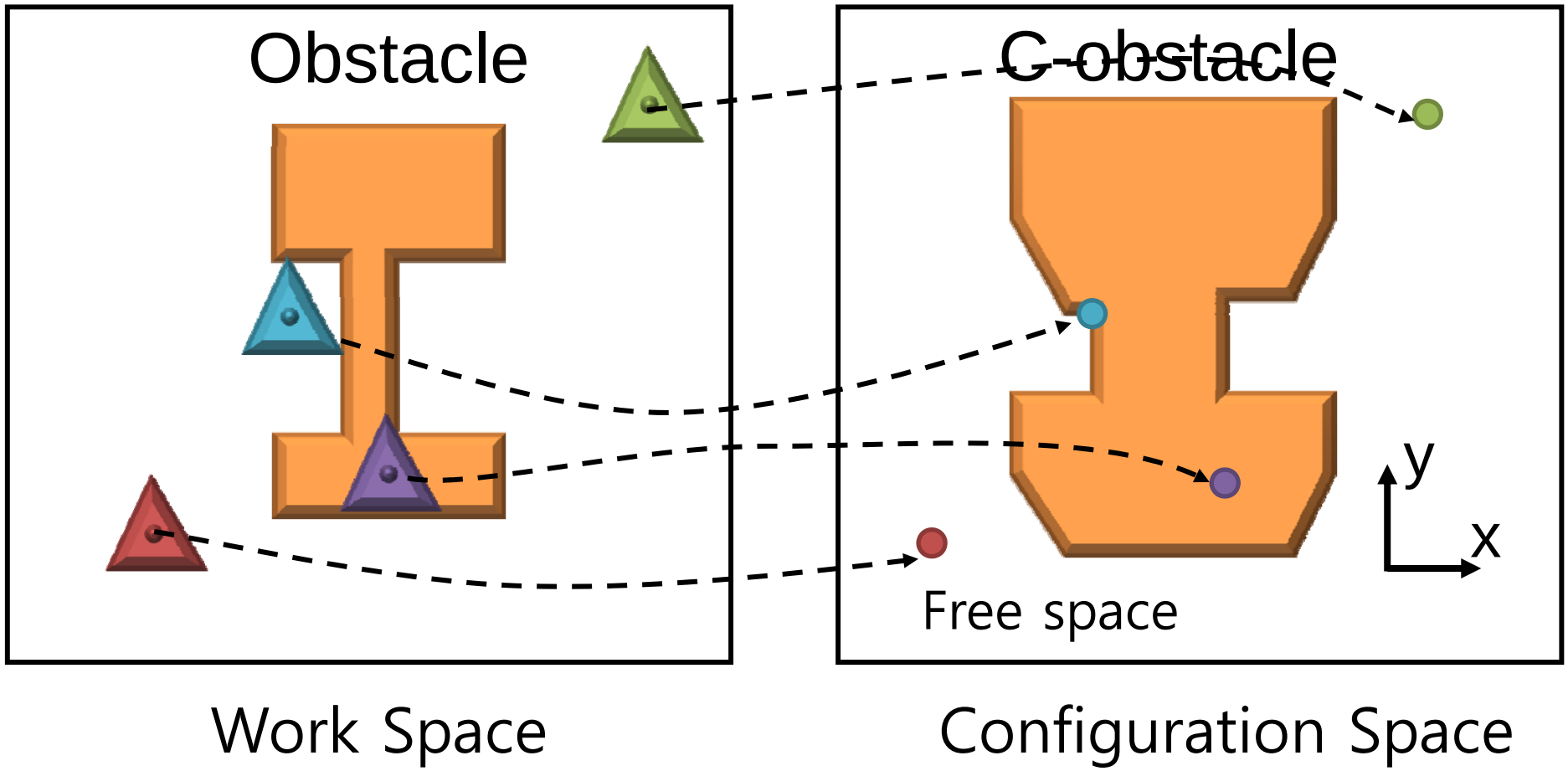
Minimum translational distance needed to separate objects

Penetration Depth



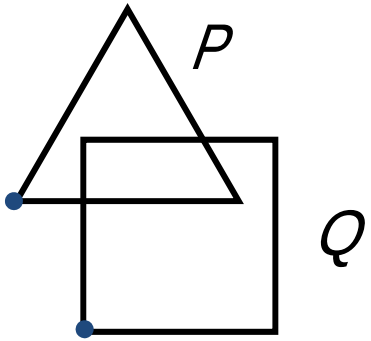
Minimum translational distance needed to separate objects

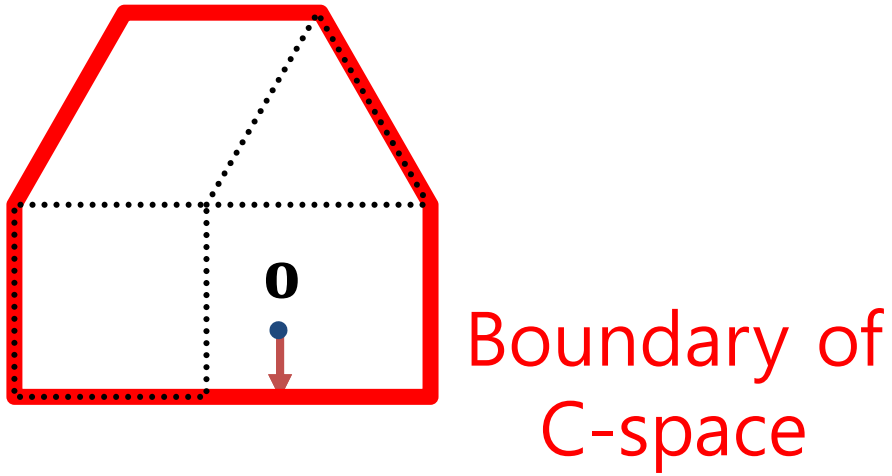
Configuration Space



C-Space and PD

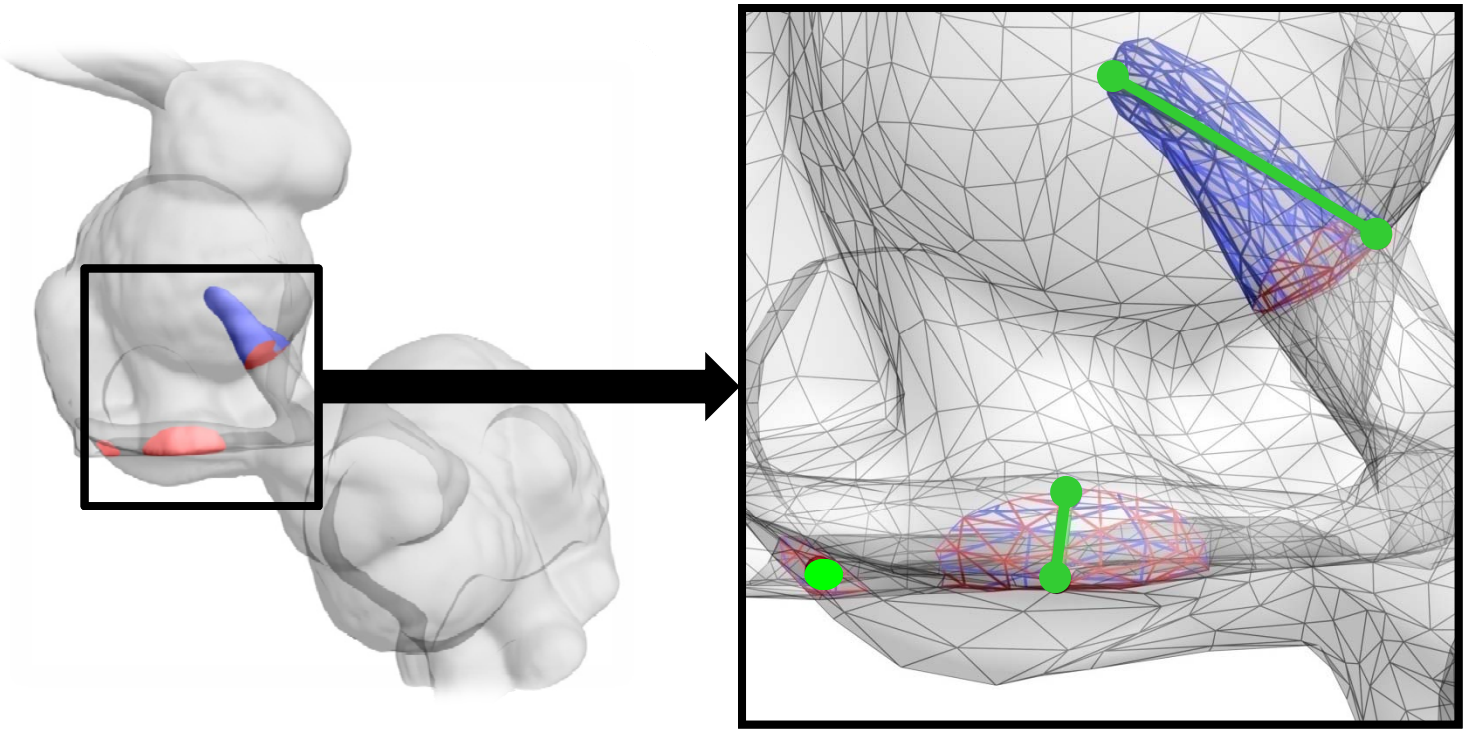
- PD^t = min distance btwn the origin and the boundary of C-space





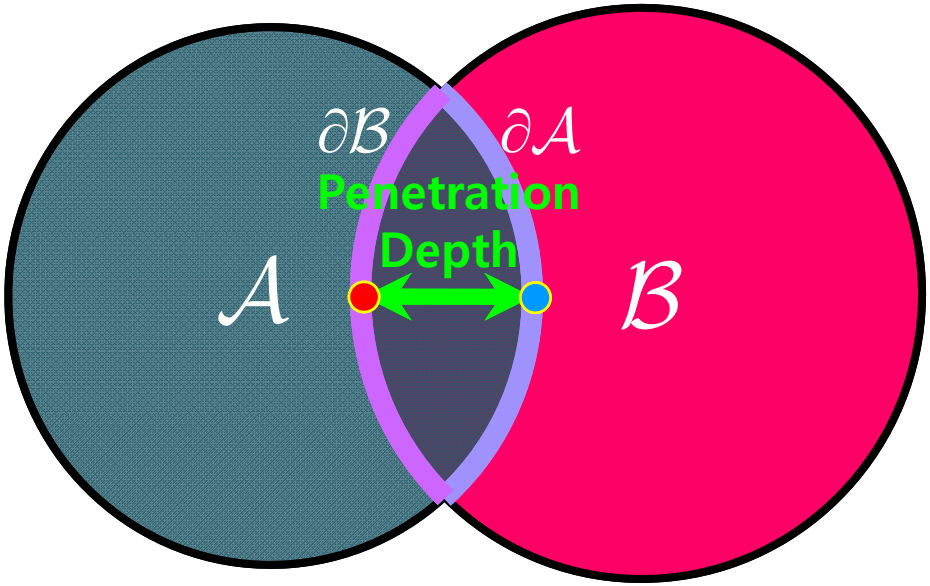
Pointwise Penetration Depth

- Defined as deepest interpenetrating points

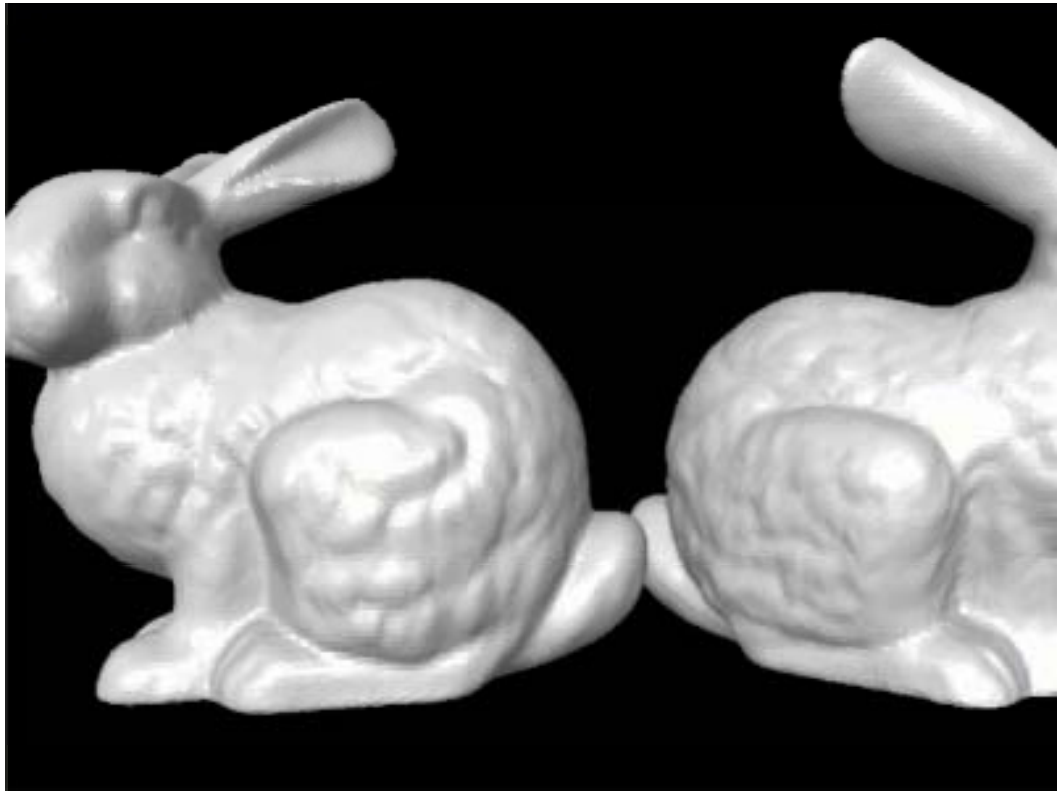


Penetration Depth

1. Find intersection surfaces $\partial\mathcal{A}$ and $\partial\mathcal{B}$
2. Penetration depth = $H(\partial\mathcal{A}, \partial\mathcal{B})$



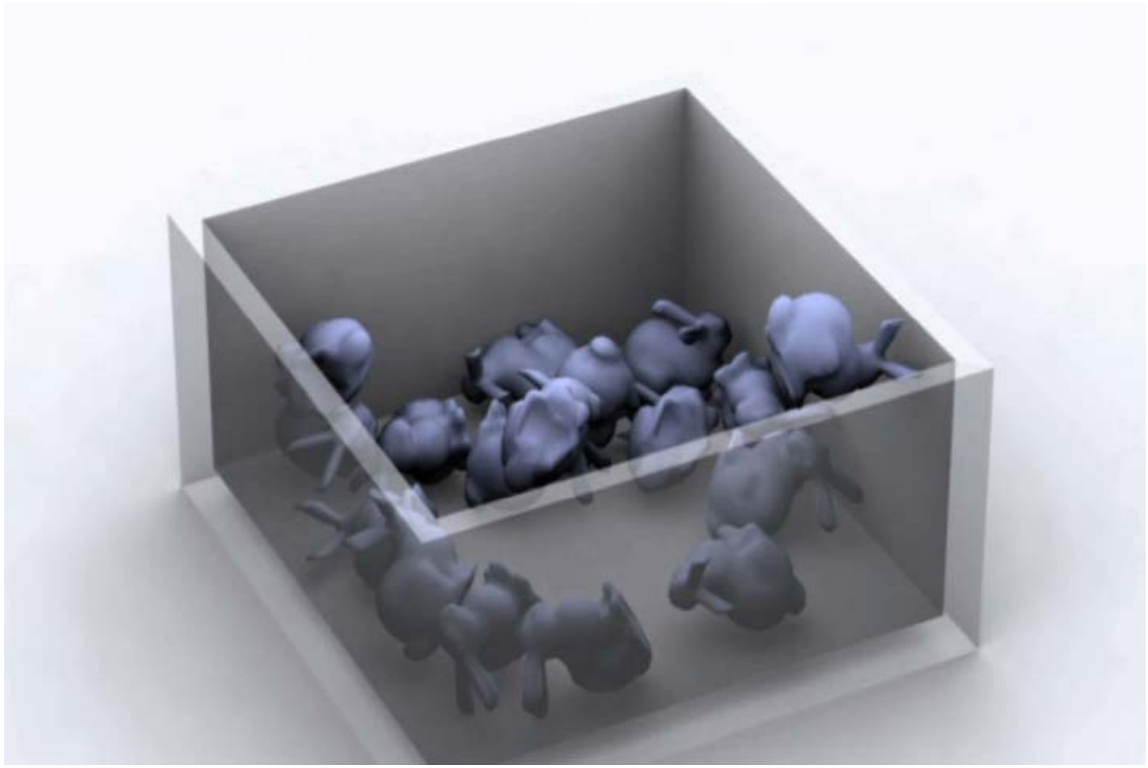
Penetration Depth



Demo (40K Bunny vs 40K Bunny)

Benchmark 1

- Run rigid body dynamics and measure the average timing of penetration depth computation



Model complexity

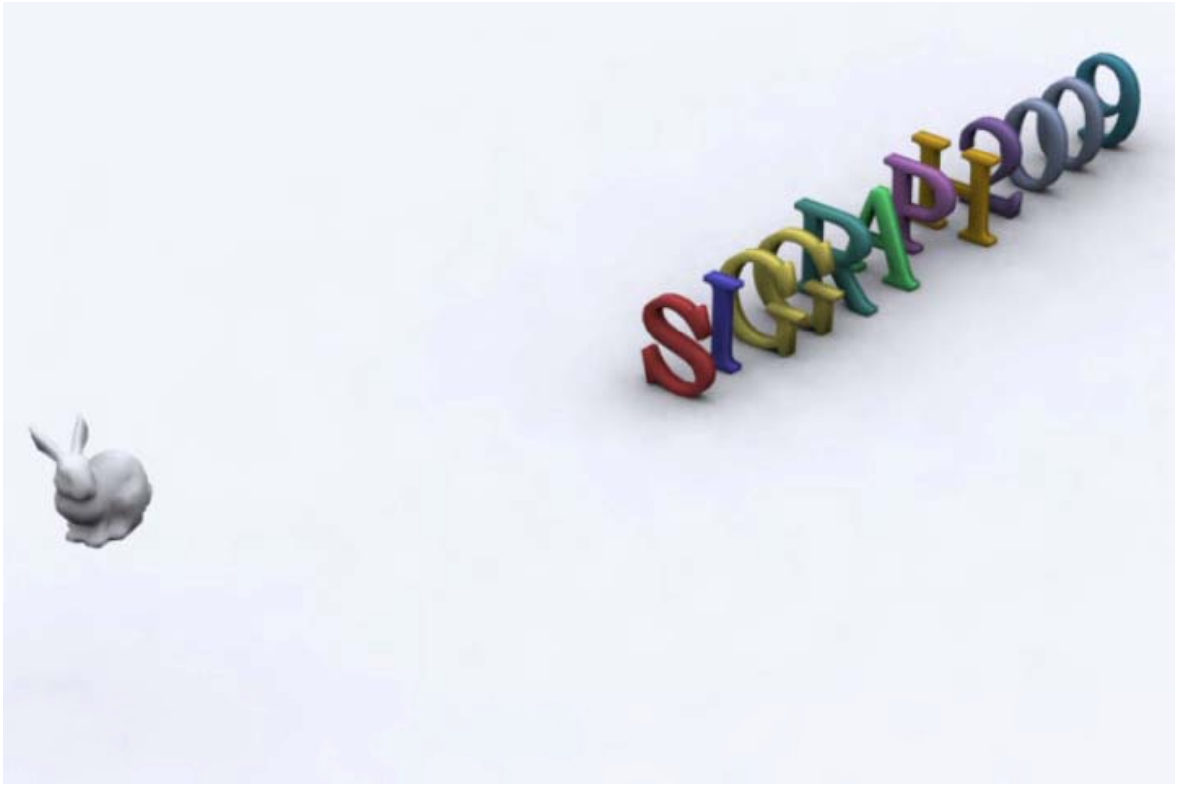
- 50K tri

Avg. Performance

- 3.88ms/pair

Benchmark 2

- Run rigid body dynamics and measure the average timing of penetration depth computation



Model complexity

- 3.5K tri

Avg. performance

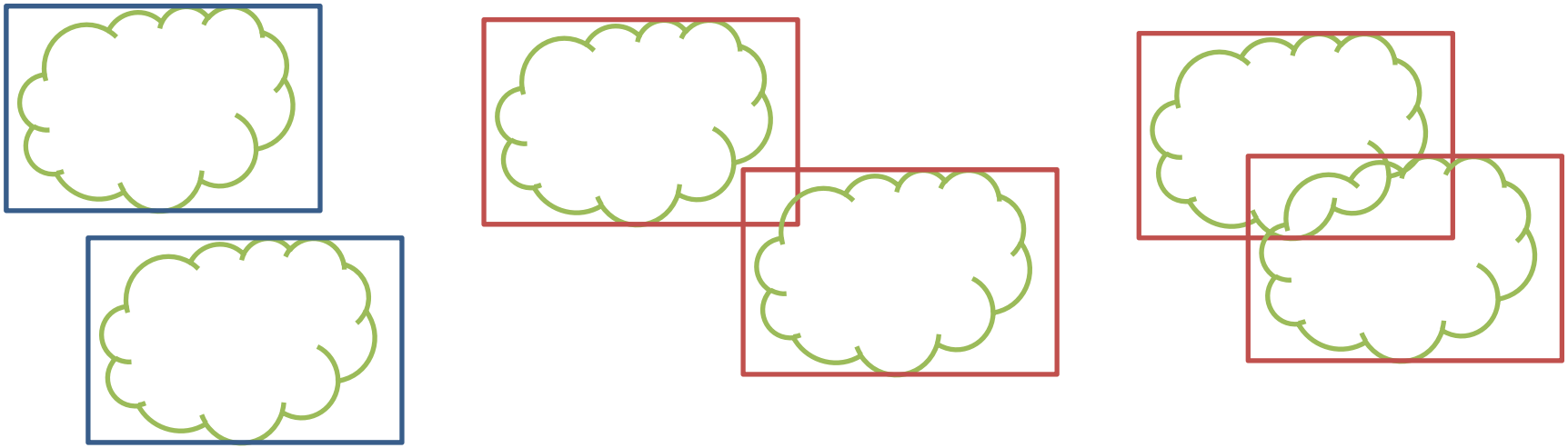
- 0.95ms/pair

MORE ON 1/28, 13:00
휘닉스볼륨 2 GRAPH 세션

PARALLEL COLLISION DETECTION

Bounding Volume

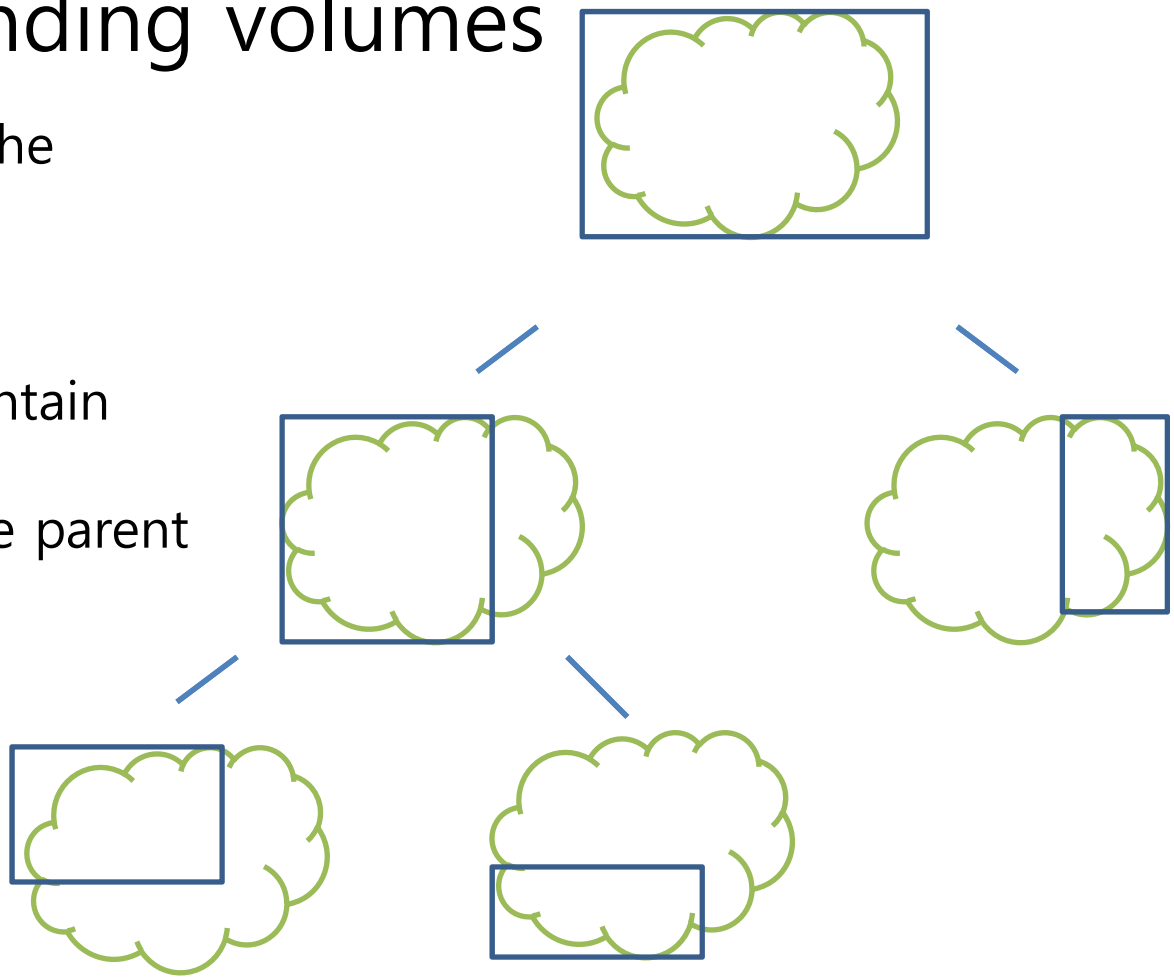
- Simplify surface representation for fast proximity queries



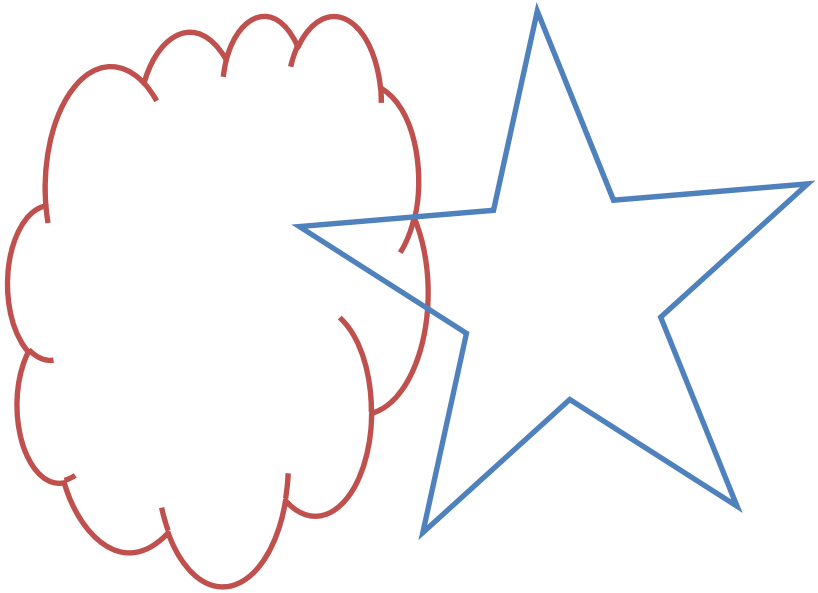
Bounding Volume Hierarchy

- A tree of bounding volumes

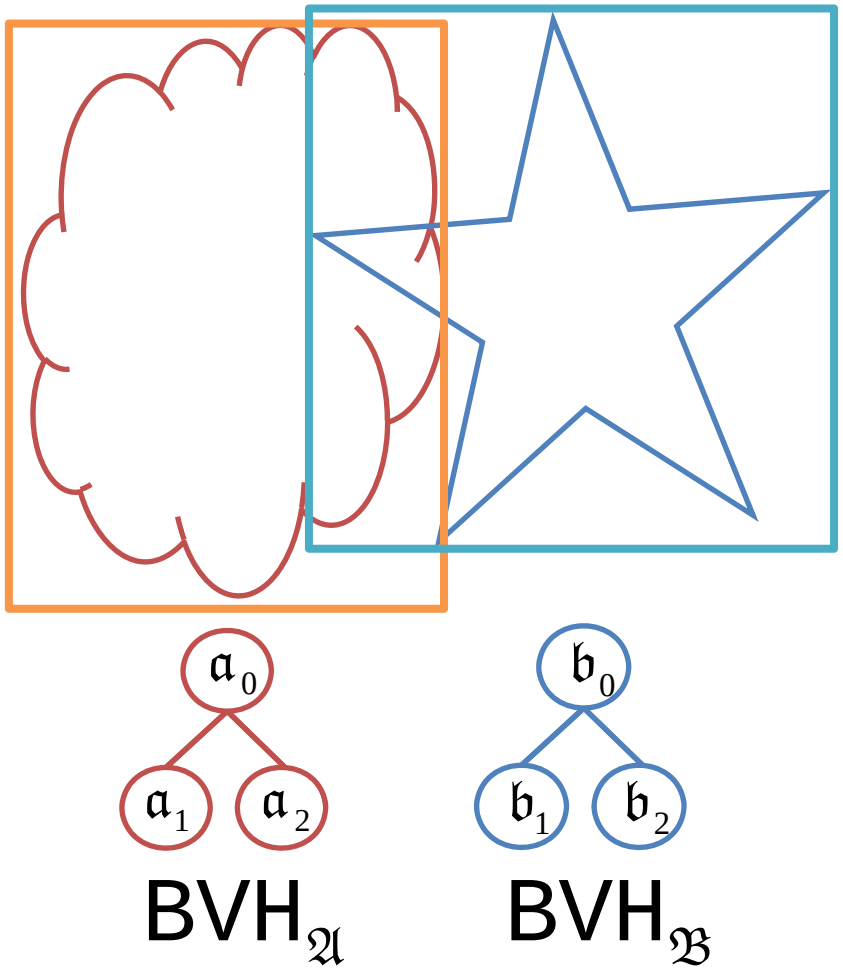
-**Root node** contains all the primitives of a model
-**Leaf nodes** contain one primitive
-**Children nodes** each contain separate partitions of the primitives enclosed by the parent



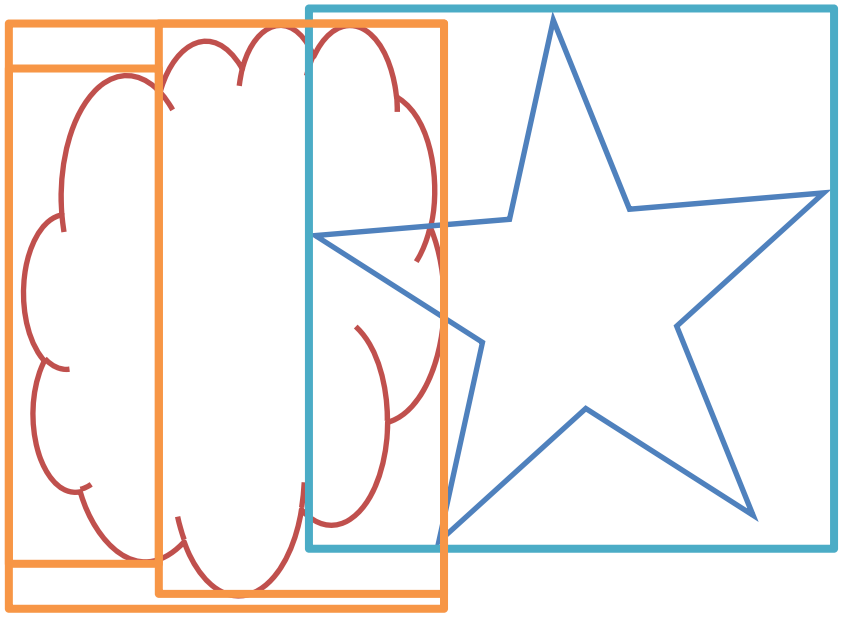
BVH Collision Test



BVH Collision Test

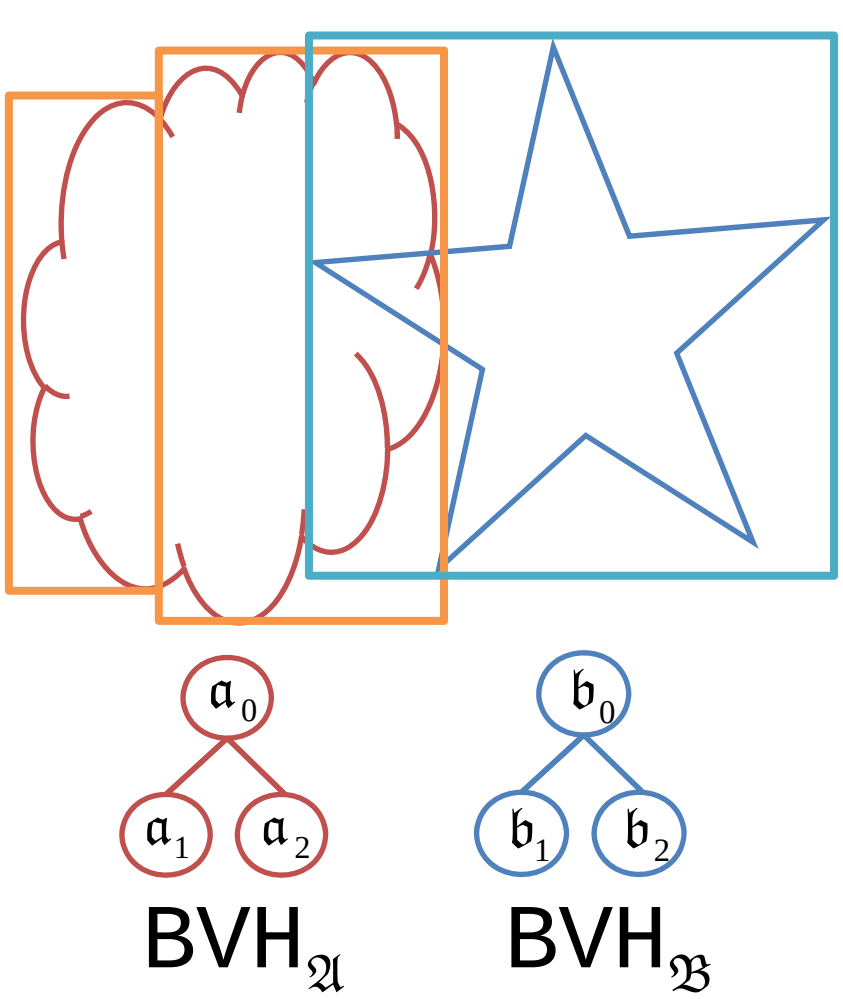


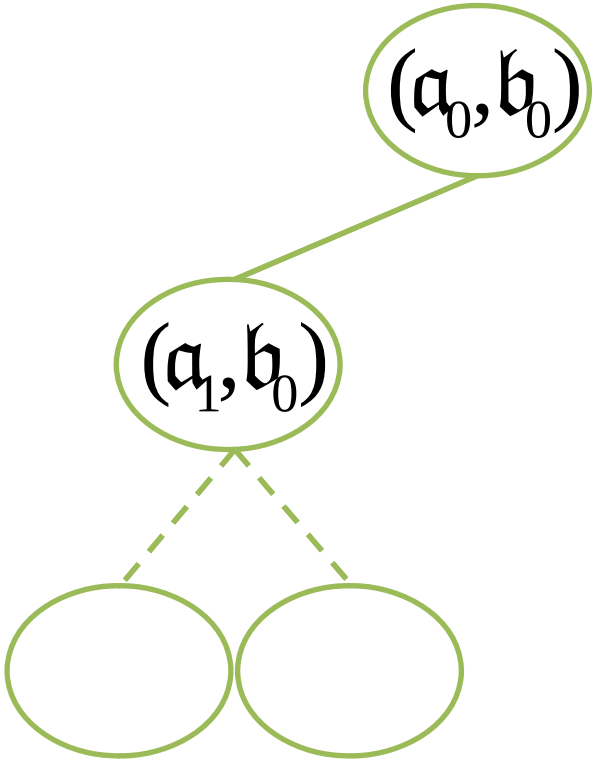
BVH Collision Test



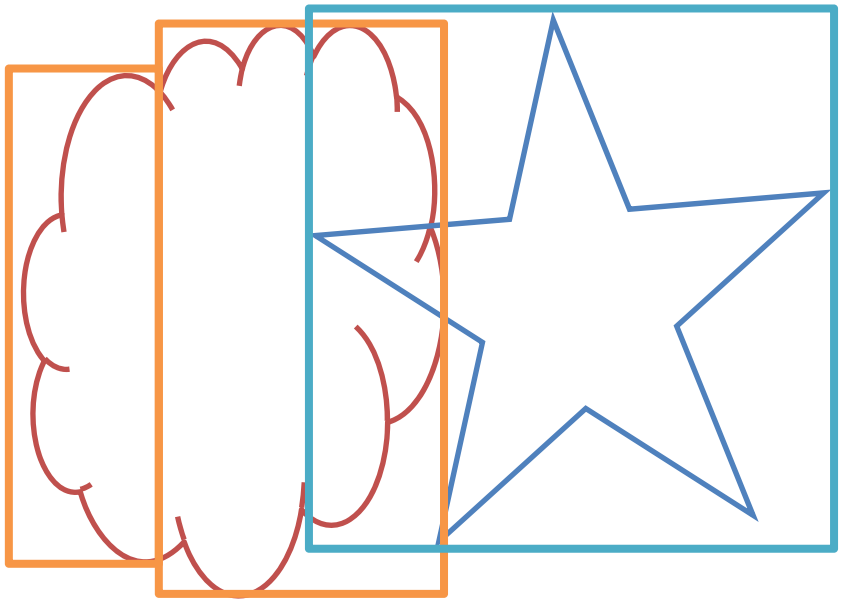
$(\mathfrak{a}_0, \mathfrak{b}_0)$

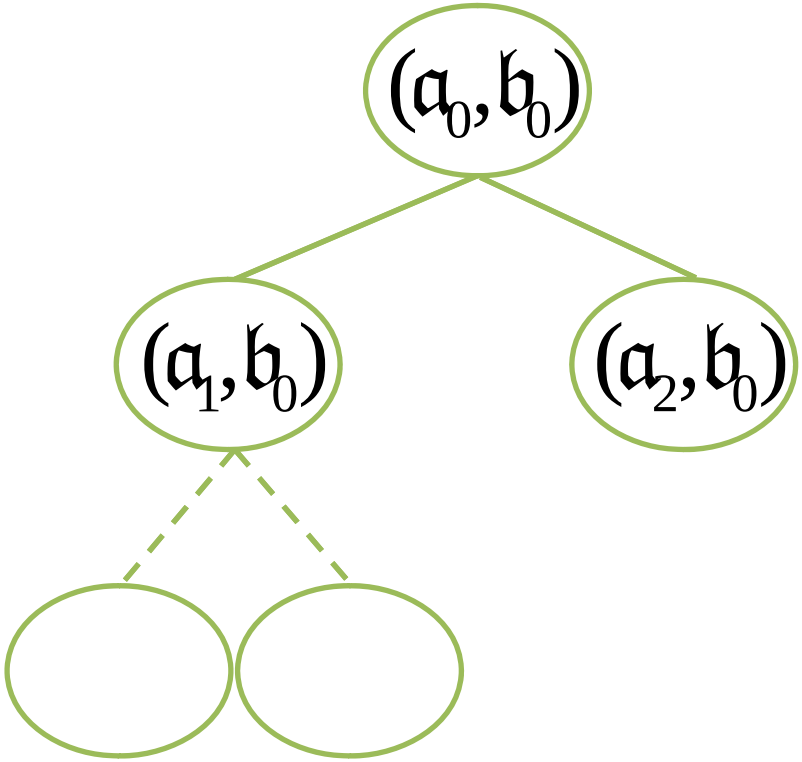
BVH Collision Test



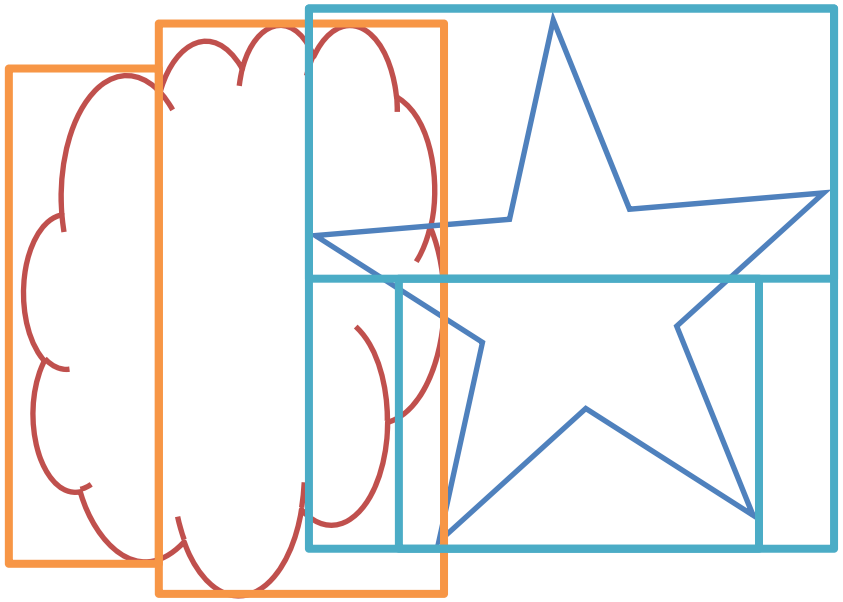


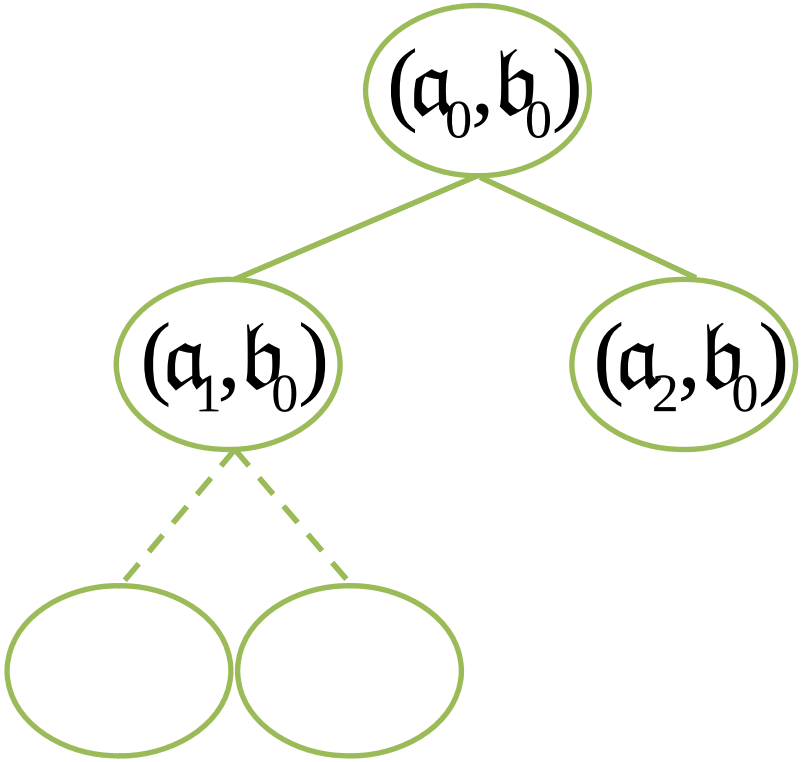
BVH Collision Test



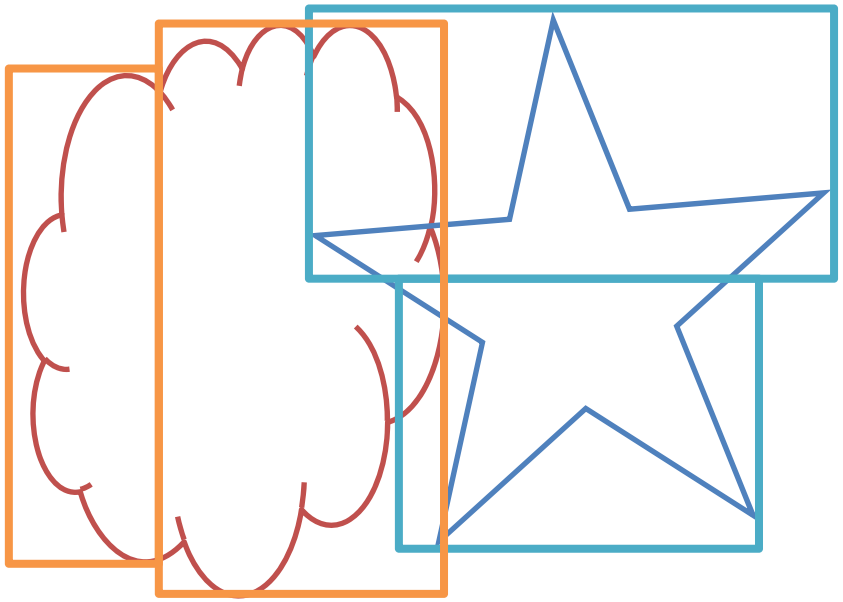


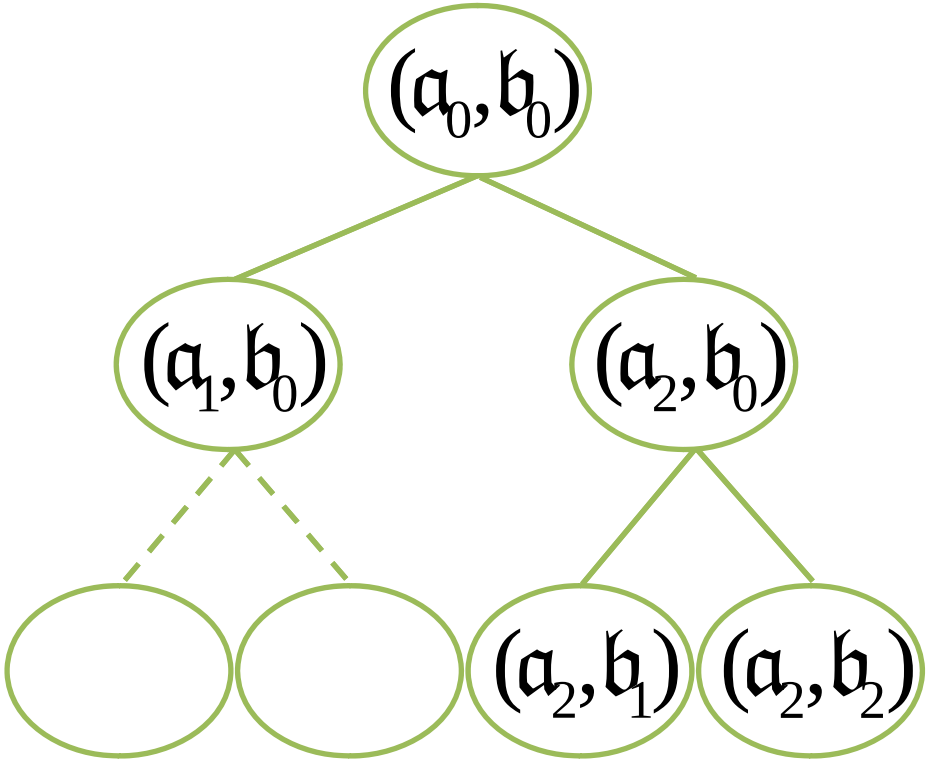
BVH Collision Test



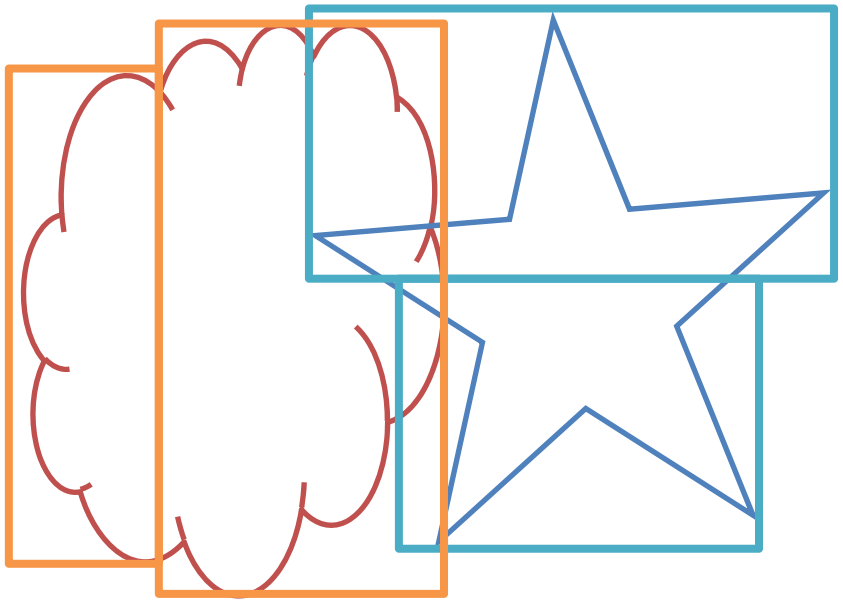


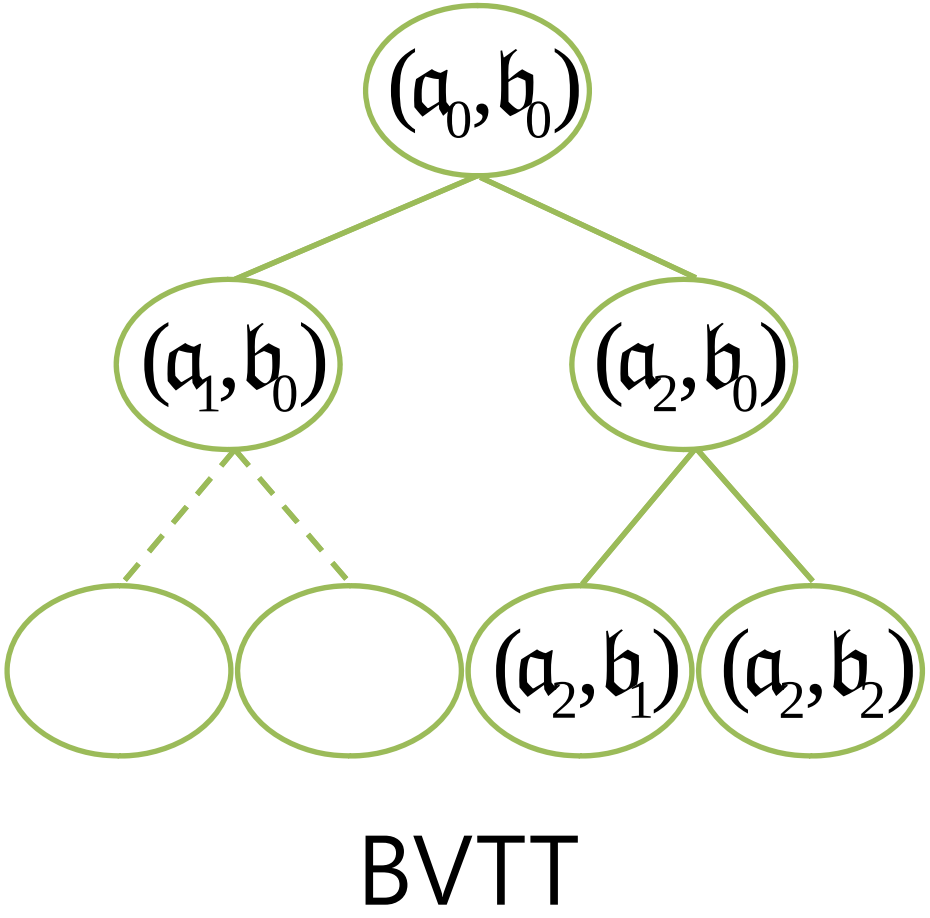
Bounding Volume Traversal Tree





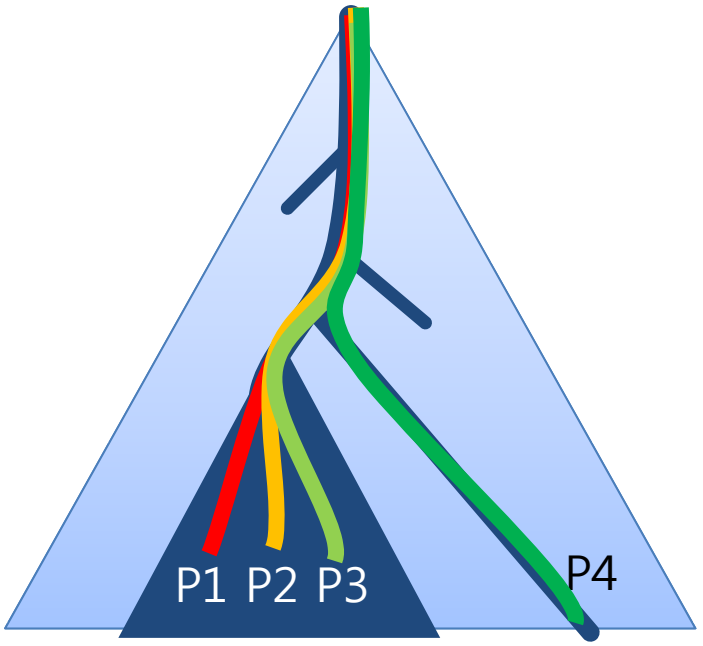
Bounding Volume Traversal Tree





Idea

- Traverse the BVTT in parallel
  - Concurrency
  - Dependency
  - Efficient load balancing

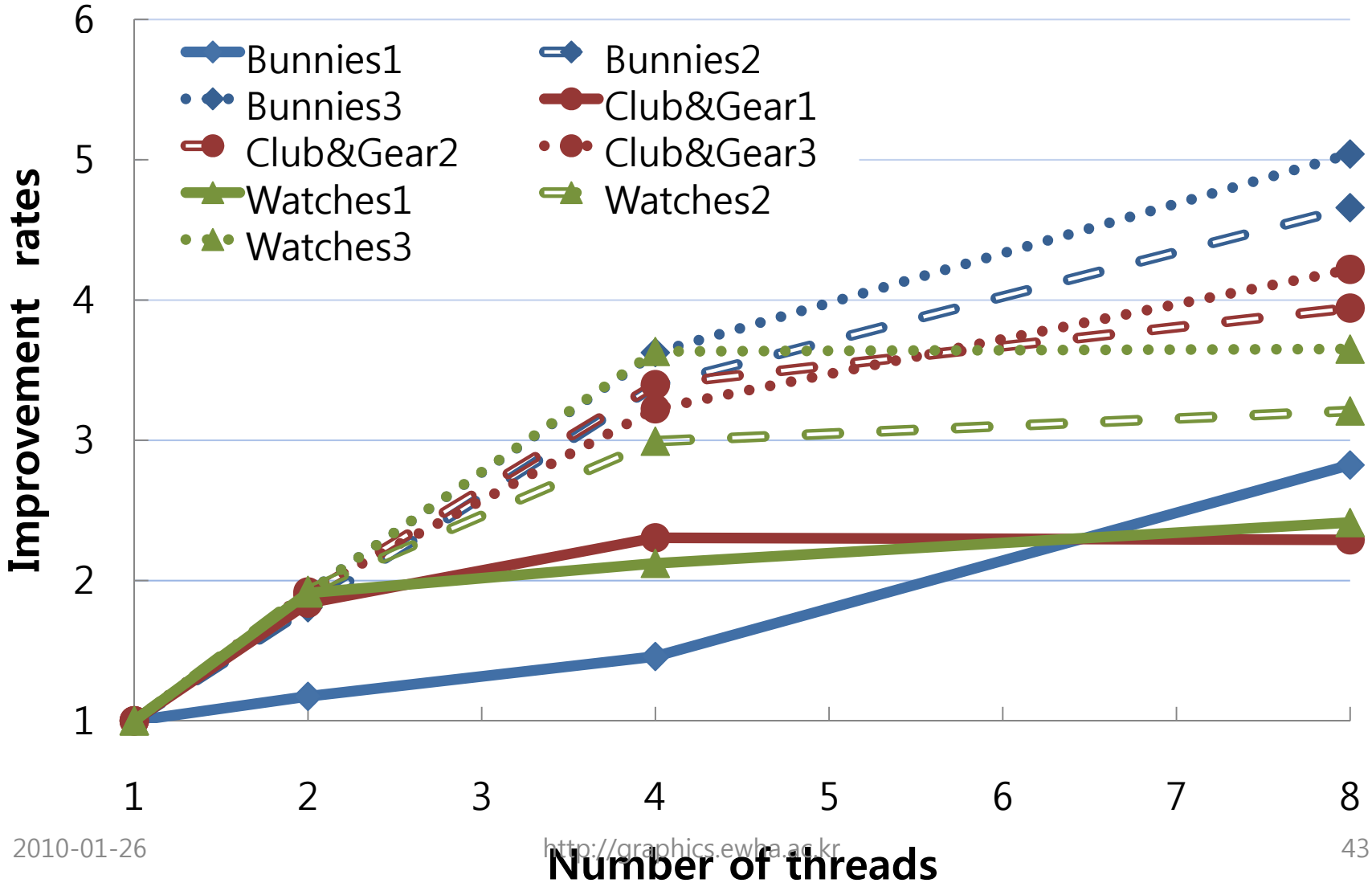


BVTT

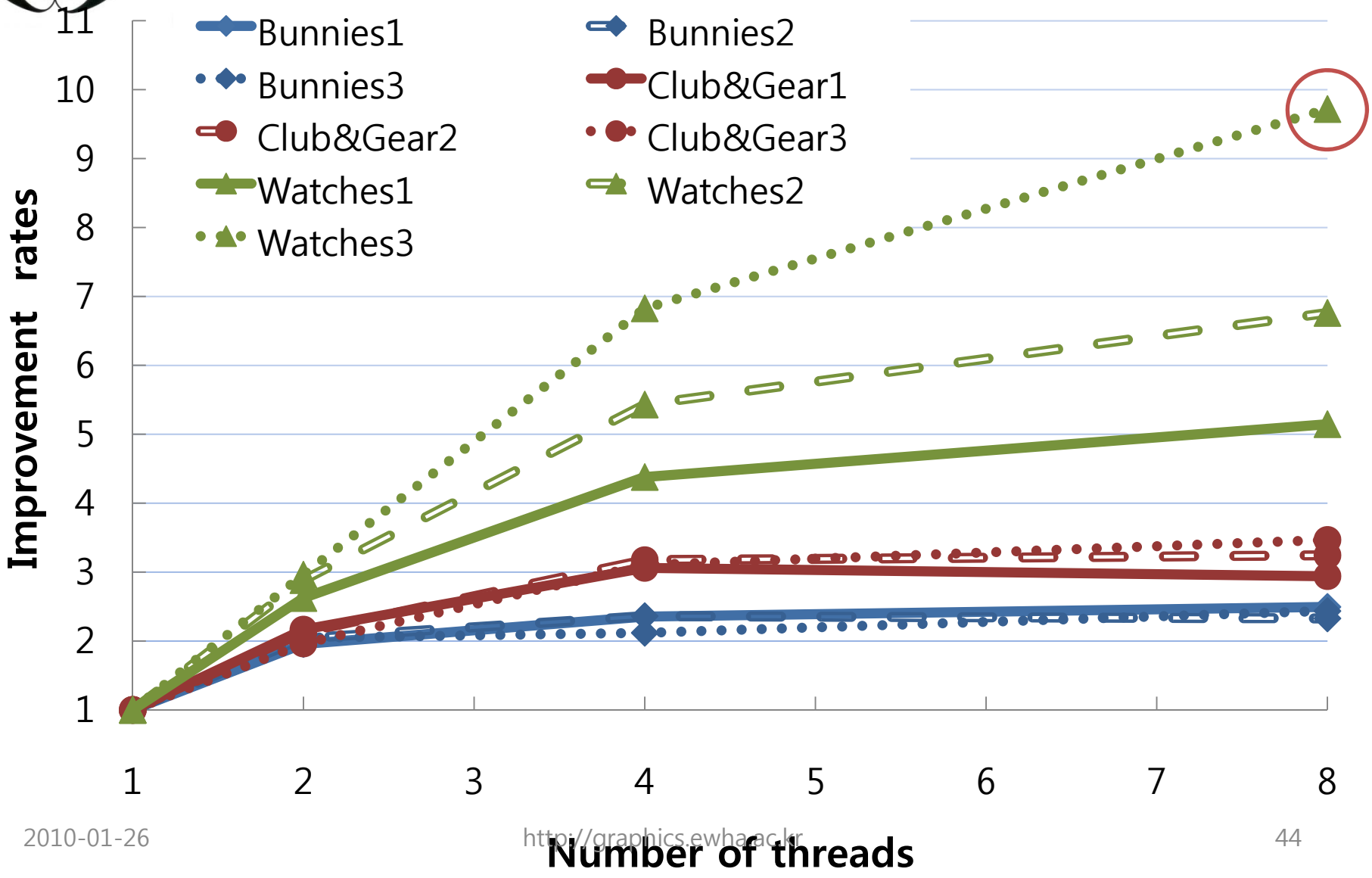
Parallel Collision Detection Demo

Parallel Distance Calculation Demo

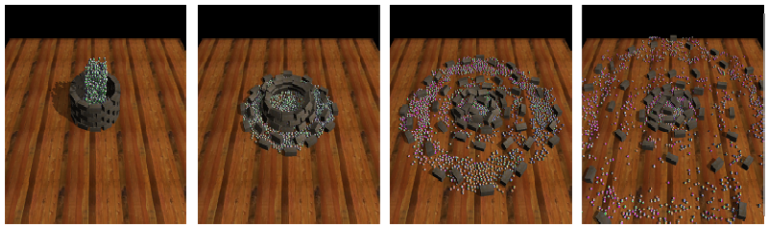
Collision Detection Results



Distance Calculation Results



Summary



Virtual Physics

Game Softwares

Ewha Collision Package

PolyDepth (PD query)

C^2A (Continuous query)

PPA (Collision, Distance)

GPU-SAP (N-body)

Other Physics Engines

Future Work

- Collision detection for fracturing
- N-body penetration depth
- Real-time generalized penetration depth

Thank you!

김영준

kimy@ewha.ac.kr

실시간 강체 동역학 시뮬레이션 기술

한국과학기술연구원
영상미디어센터

김 진 욱

Physically Based Simulation

- Newton's physics law

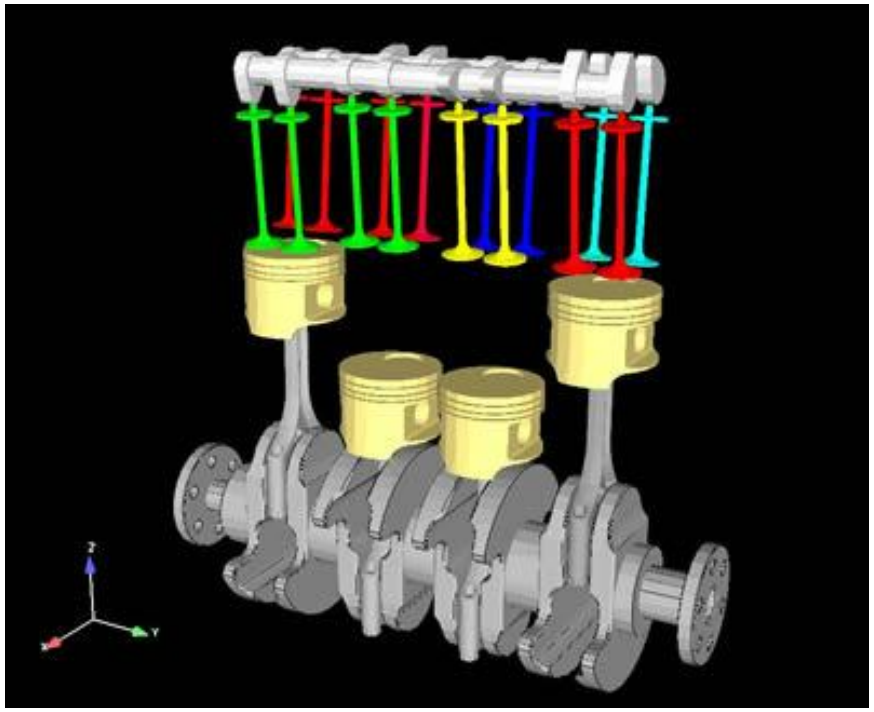
$$f = m\ddot{x}$$

  - If you push something, it will move

- Different governing equations
  - Particle
  - Rigid body
  - Articulated body
  - Deformable body
  - Fluid system

Engineering

- Everywhere modeling and simulation are required
  - Adaptive control
  - Structure simulation
  - Mechanism design



Engine block simulation
(ADAMS, MSC)

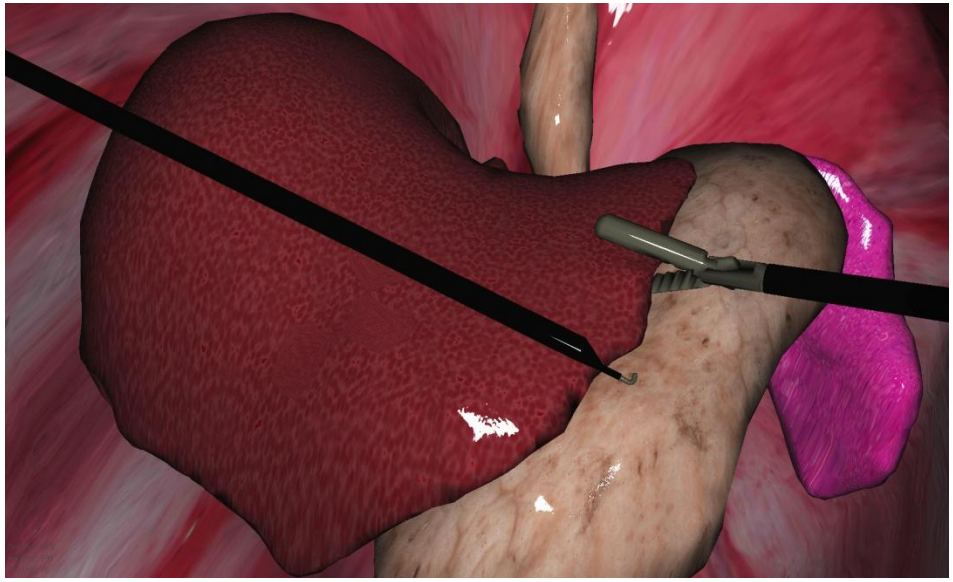
Entertainment

- To enhance visual realism
  - Virtual reality
  - Digital contents
  - Video games

Monster Inc.(PIXAR)

Medical Simulation

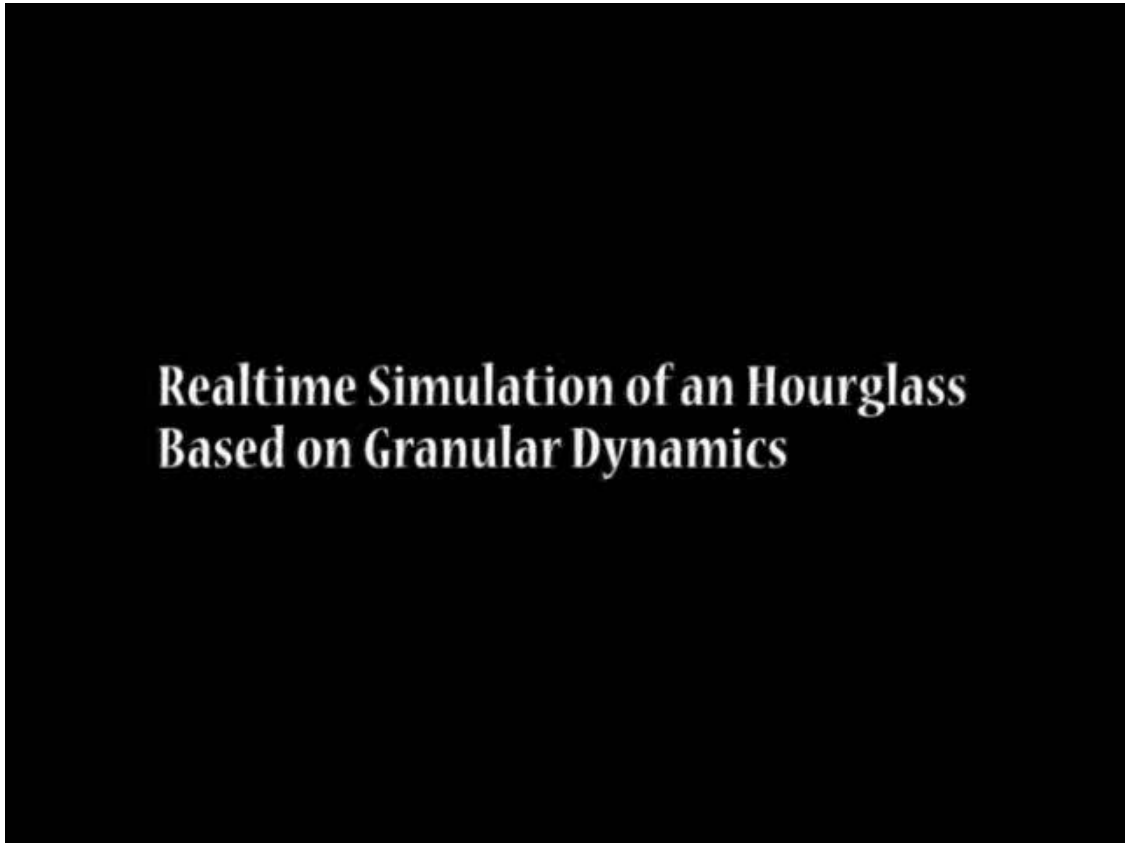
- To predict experimental medical treatment
  - Virtual surgery
  - Tissue simulation
  - Developing medical instruments



VR based endoscopic surgery(M. Srinivasan, MIT)

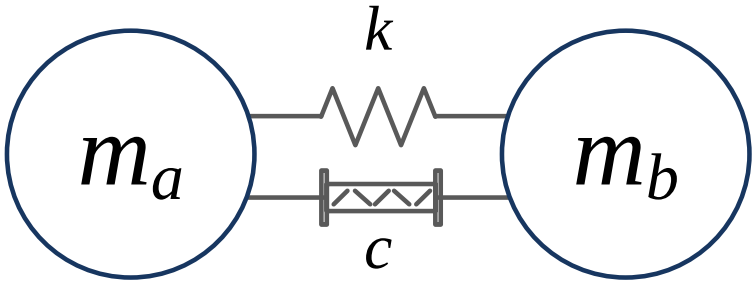
Particle System

- Simplified body without volume



Granular Dynamics(R. Yasuda, U.Tokyo)

Mass-Spring-Damper System



$$m\ddot{x} = f$$

$$f = -kx - c\dot{x}$$

- Easy to implement
- Hard to define proper k and c
- Tends to be unstable

Forward Dynamics

- Evaluate force applied to the particle
  - gravity, spring, damper, collision, …
- Calculate acceleration

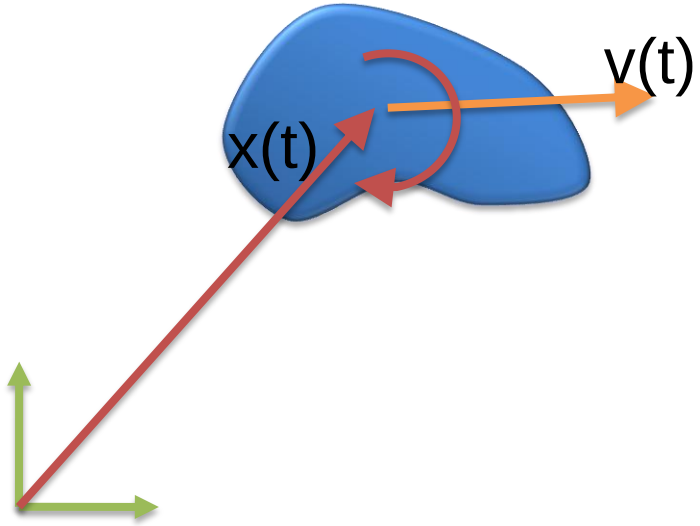
$$a = f / m$$

- Step forward in time
  - Integrate dynamics

$$\ddot{x} = a$$

Rigid Body System

- Body with a volume
- Dynamics state
  - Position/orientation
  - Linear/angular velocity

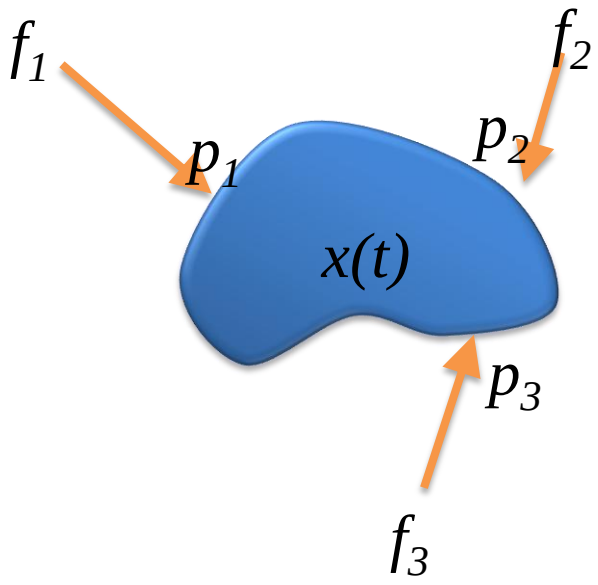


Rigid Body Dynamics

- Newton-Euler equation

$$f = m\dot{v}$$

$$\tau = I\dot{w} + w \times (Iw)$$



$$f = \sum f_i$$

$$\tau = \sum (p_i - x(t)) \times f_i$$

Collision & Contact

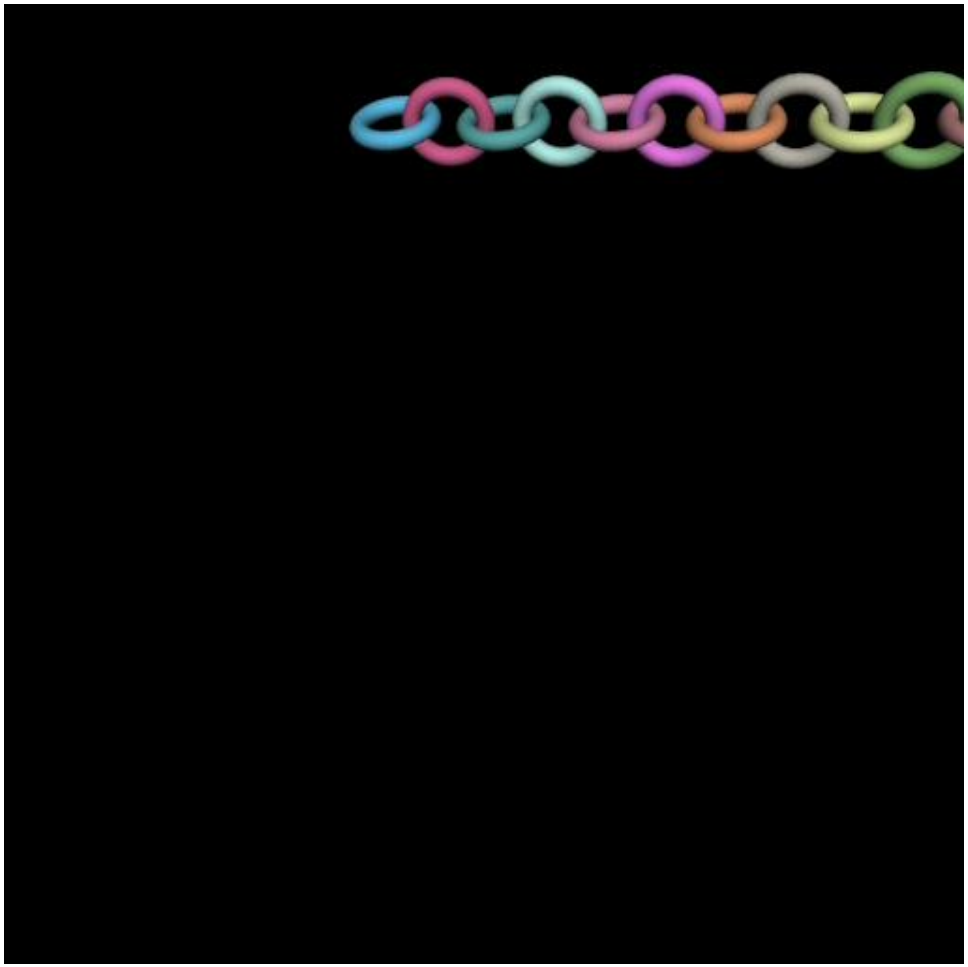
- Collision
  - A body hits another body.
  - Relative normal velocity is negative.
  - ***Impulse*** governs the dynamics.

- Contact
  - A body rests against another body.
  - Relative normal velocity is negligible.
  - ***Resting force*** prevents inter-penetration.

Examples

accurate frictional, rolling behavior

Examples



non-convex geometries

Examples

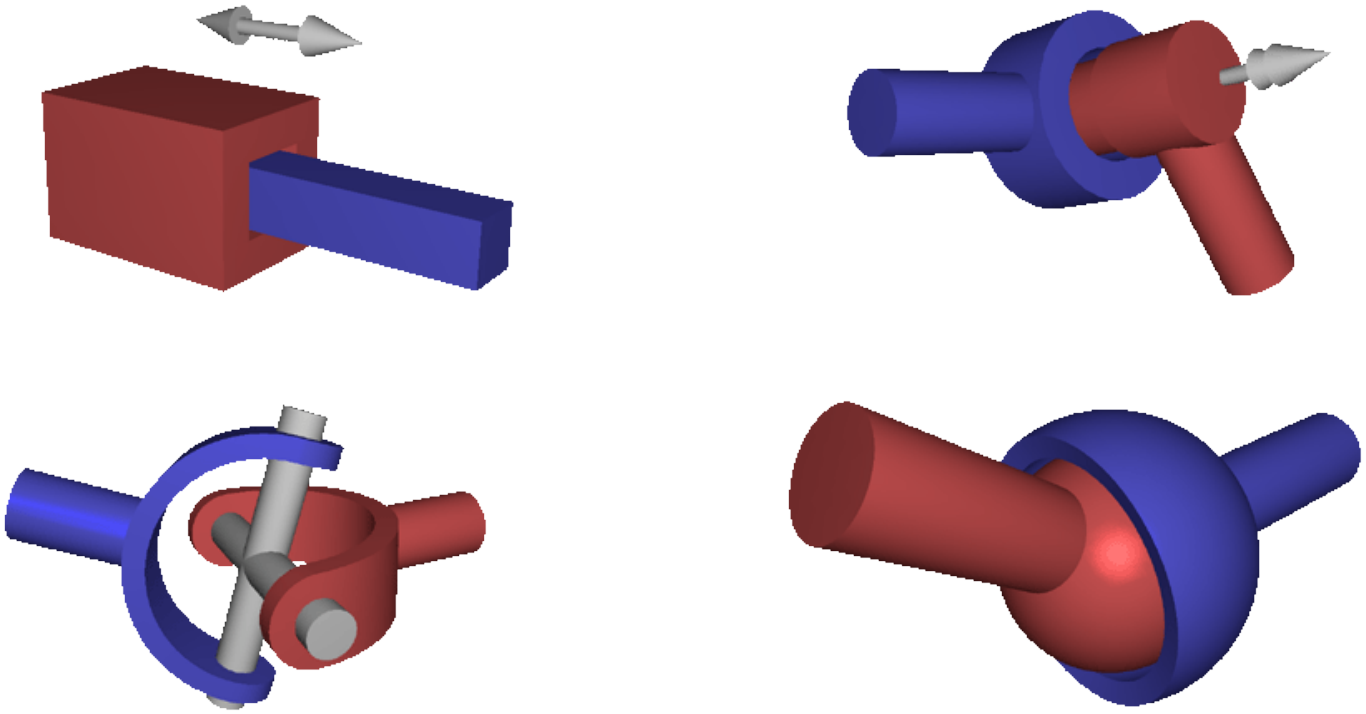
stress test(300 stacked objects)

Articulated Body System

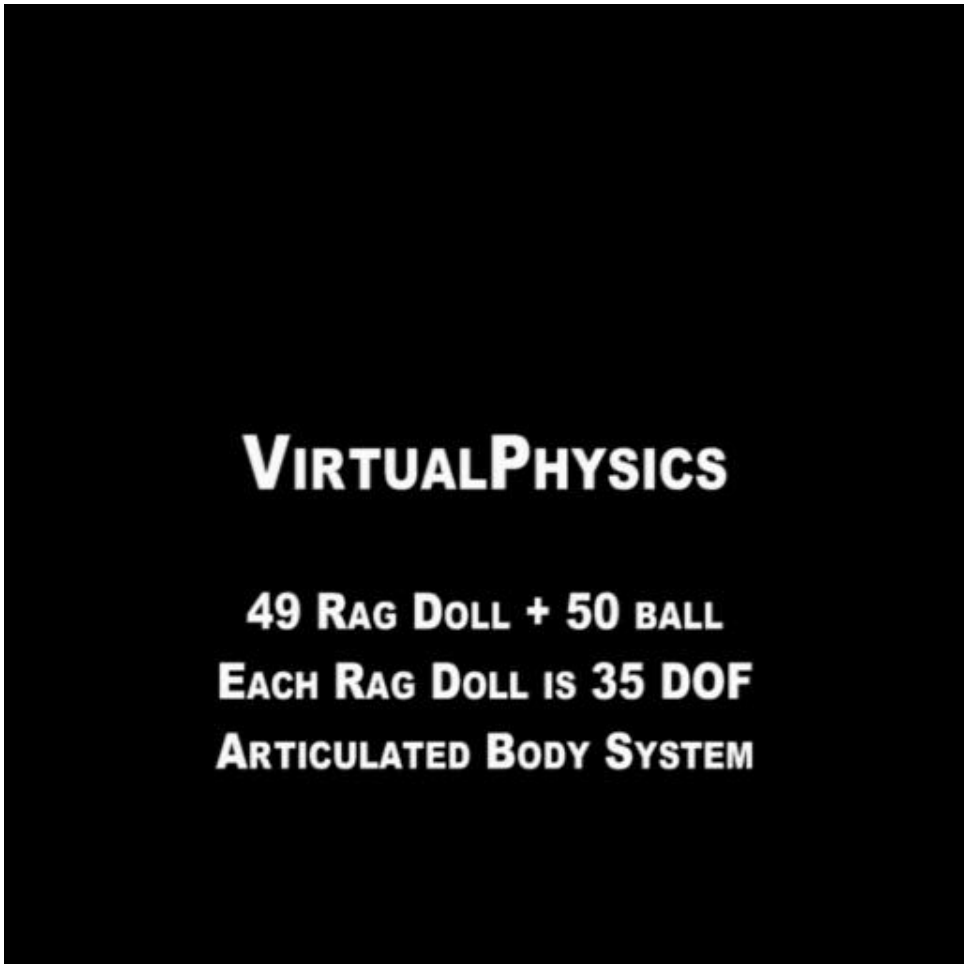
- A set of constrained rigid bodies
- Lagrangian approach
  - Solve dynamics with implicit constraint equations
  - In most of the available physics engine
  - $O(n^3)$
- Independent coordinates approach
  - Express dynamics only with the independent variables
  - Popular in Robotics
  - $O(n^3)$: Composite body algorithm
  - O(n) : Featherstone algorithm

Joint Constraints

- Joint variable as an independent coordinate



Examples



49 rag dolls(35DOF each)

Rigid Body Dynamics

- Newton's Law

$$f = ma$$

- In 3D case,

$$f = ma$$

$$m = I\alpha + w \times (Iw)$$

- Using Lie group notation,

$$F = J\dot{V} - ad_V^* JV$$

Lie Group

- Lie group
  - a group and a differentiable manifold
  - a set with a binary operator
  - locally similar to Euclidean space
- Lie algebra
  - a vector space with a binary operator(Lie bracket)
  - Linearization of Lie group at the identity
  - Exponential maps Lie algebra to Lie group

Why Lie Group?

- Examples of Lie group
  - General linear group

$$GL(n,\mathbf{R}) = \{M \in \mathbf{R}^{n\times n} \mid \det M \neq 0\}$$

  - Rotation group

$$SO(n) = \{R \in \mathbf{R}^{n\times n} \mid RR^T = I, \det R = 1\}$$

  - Special Euclidean group

$$SE(3) = \{(R, p) \mid R \in SO(3), p \in \mathbf{R}^3\}$$

SO(3): Rotation Group

- Matrix multiplication rules SO(3)

$$R = R_1 R_2$$

$$RR^T = R_1 R_2 (R_1 R_2)^T$$

$$= R_1 R_2 R_2^T R_1^T = I$$

- Differentiating a rotation matrix

$$RR^T = I$$

$$\dot{R}R^T + R\dot{R}^T = \dot{R}R^T + (\dot{R}R^T)^T = 0$$

so(3): Lie Algebra of SO(3)

- $\dot{R}R^T$ is a skew symmetric matrix.

$$\dot{R}R^T = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix} = [w]$$

$$w = (w_x, w_y, w_z) \in \mathbf{R}^3$$

- Similarly $R^T\dot{R}$ is skew symmetric as well.
- so(3) is a set of 3X3 skew symmetric matrix

under the Lie bracket defined as

$$[w, v] = [w \times v]$$

or in matrix representation

$$\big[[w],[v]\big] = [w][v] - [v][w]$$

Exponential Map

- The exponential of a general square matrix

$$e^{A} = I + A + \frac{1}{2}A^{2} + \frac{1}{3}A^{3} + \cdots$$

- If A is in so(3),

$e^{\theta[w]} = I + \sin\theta[w] + (1-\cos\theta)[w]^{2}$ is in SO(3).

- Corresponds to a rotation about w by an angle θ

SE(3): Special Euclidean Group

- aka homogeneous transformation matrix

$$\begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}, R \in SO(3),\ p \in \mathbf{R}^3$$

- Ordered pair (R, p) with group multiplication

$$(R_1, p_1)\cdot(R_2, p_2) = (R_1 R_2, R_1 p_2 + p_1)$$

or in matrix representation

$$\begin{bmatrix} R_1 & p_1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} R_2 & p_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1 R_2 & R_1 p_2 + p_1 \\ 0 & 1 \end{bmatrix}$$

se(3): Lie Algebra of SE(3)

- a pair of two 3dim vector $V = (w, v)$

or

$$\begin{bmatrix} [w] & v \\ 0 & 0 \end{bmatrix}, w, v \in \mathbf{R}^3$$ in matrix representation

- Lie bracket of se(3)

$$[(w_1, v_1), (w_2, v_2)] = (w_1 \times w_2, w_1 \times v_2 - w_2 \times v_1) = ad_{V_1} V_2$$

- Exponential map

$$\exp(w, v) = e^{\begin{bmatrix} [w] & v \\ 0 & 0 \end{bmatrix}}$$

Adjoint map

$$\mathrm{Ad}_T : se(3) \to se(3)$$

$$Ad_T V = T\,V\,T^{-1} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} [w] & v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} R^T & -R^T p \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} [Rw] & p \times Rw + Rv \\ 0 & 0 \end{bmatrix}$$

$$Ad_T V = (Rw,\ p \times Rw + Rv)$$

or in matrix representation

$$Ad_T V = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix}$$

Generalized Force

- se(3)*, dual of se(3)

$$F = (m, f)$$

f : force vector
m : moment vector

- Transformation of se(3)*

  dual Adjoint operator $\quad F_B = Ad^*_{T_B^A} F_A$

Generalized Inertia Tensor

- triple (I, m, r)
  - I: 3X3 inertia matrix of the rigid body
  - m: mass of the rigid body
  - r: 3dim. offset vector
  - in matrix representation

$$J = \begin{bmatrix} I & [r] \\ -[r] & m1 \end{bmatrix}$$

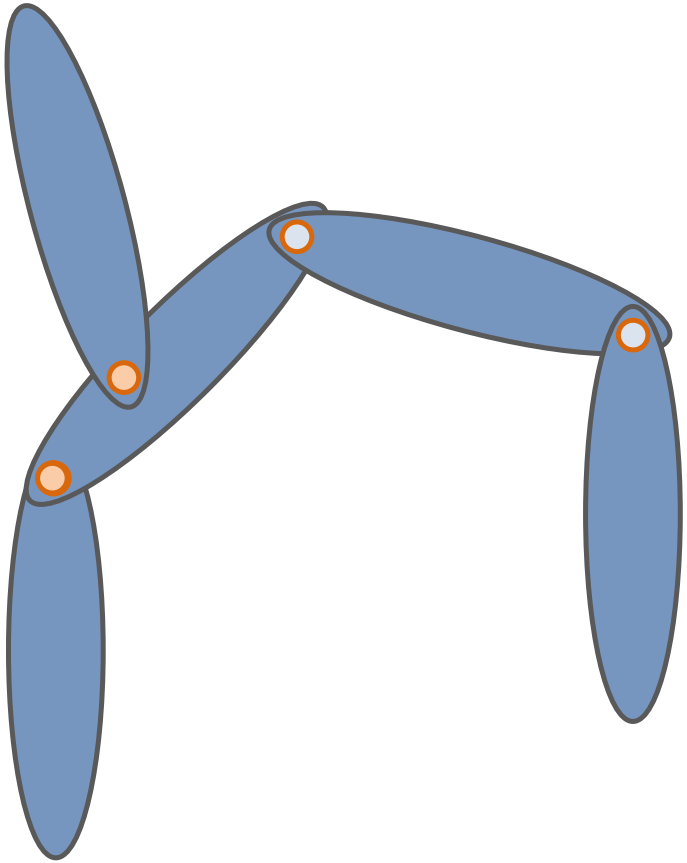
- maps se(3) to se(3)*

$$JV = (Iw + r \times v, mv - r \times w)$$

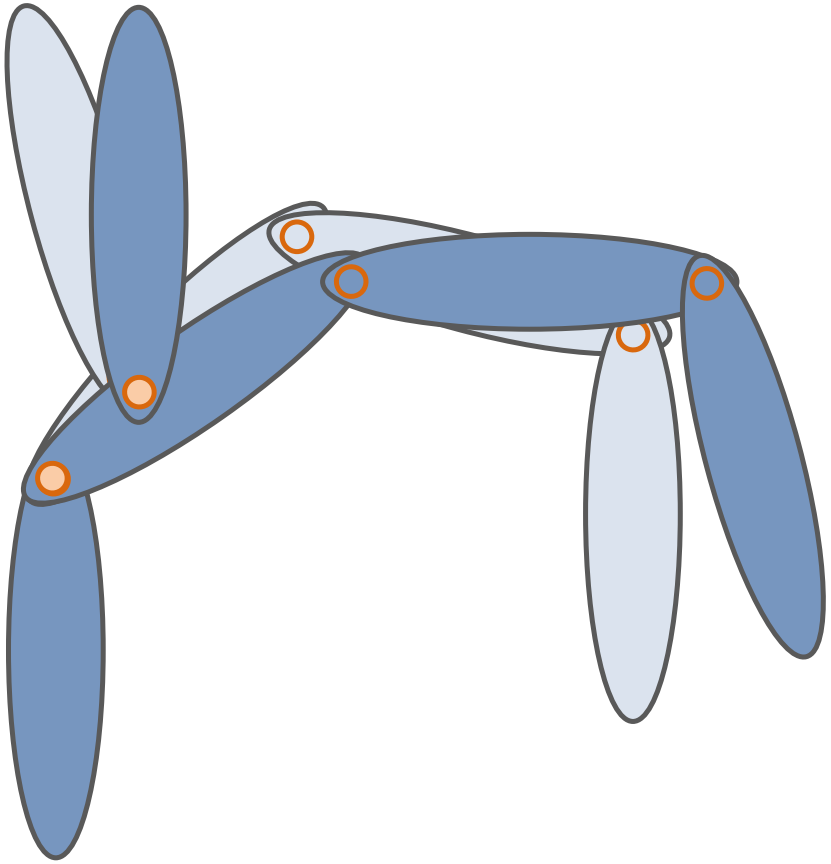
- Transformation of generalized inertia tensor

$$J_B = Ad_T^* J_A Ad_T$$

Articulated Rigid Body System



Articulated Rigid Body System



Joint

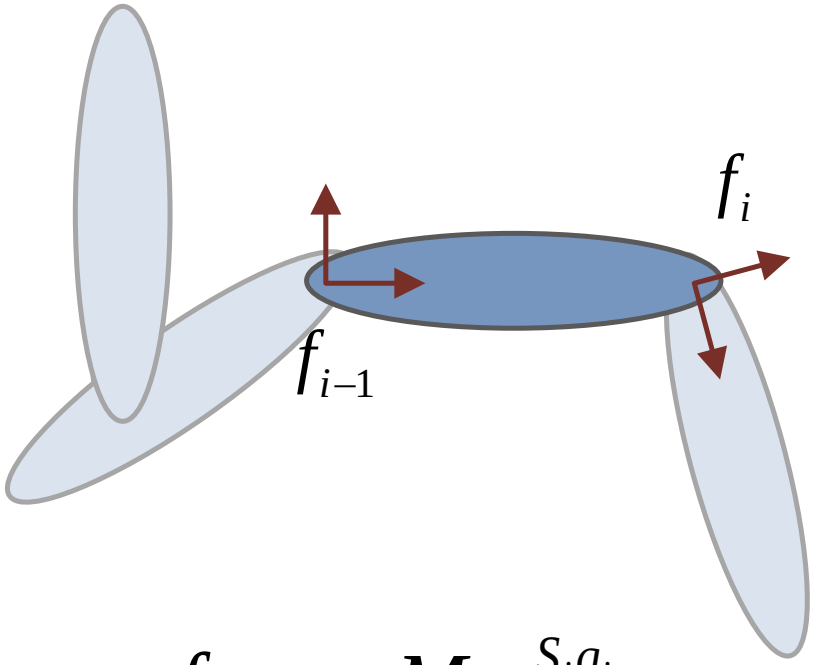
- Allows relative motion between adjacent bodies
- Can be parameterized through exponential map if the joint is 1DOF

$$T = e^{Sq}$$

$$T^{-1}\dot{T} = S\dot{q}$$

- Example
  - Rotational joint $S = (w, 0)$
  - Prismatic joint $S = (0, v)$

Kinematics



$$f_{i-1,i} = M_i e^{S_i q_i}$$

$$f_i = f_{0,1} \cdots f_{i-1,i}$$

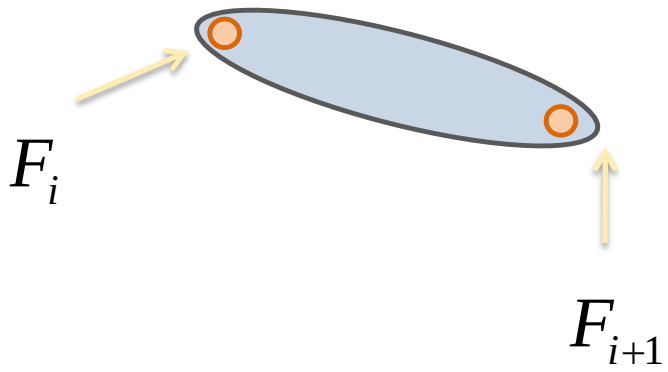
$$= f_{i-1} f_{i-1,i}$$

Velocity and Acceleration

$$
\begin{aligned}
V_i &= f_i^{-1}\dot{f}_i \\
&= (f_{i-1}f_{i-1,i})^{-1}(\dot{f}_{i-1}f_{i-1,i} + f_{i-1}\dot{f}_{i-1,i}) \\
&= f_{i-1,i}^{-1}f_{i-1}^{-1}\dot{f}_{i-1}f_{i-1,i} + f_{i-1,i}^{-1}f_i^{-1}f_{i-1}\dot{f}_{i-1,i} \\
&= f_{i-1,i}^{-1}V_{i-1}f_{i-1,i} + f_{i-1,i}^{-1}\dot{f}_{i-1,i} \\
&= Ad_{f_{i-1,i}^{-1}}V_{i-1} + S_i\dot{q}_i
\end{aligned}
$$

$$
\dot{V}_i = Ad_{f_{i-1,i}^{-1}}\dot{V}_{i-1} + S_i\ddot{q}_i + ad_{V_i}S_i\dot{q}_i
$$

Dynamics



$$F_i = Ad^*_{f_{i,i+1}^{-1}} F_{i+1} + J_i \dot{V}_i - ad^*_{V_i} J_i V_i$$

$$\tau_i = \langle S_i, F_i \rangle$$

Recursive Inverse Dynamics

initialize

$$V_0, \dot{V}_0, F_{n+1}, \dot{q}, \ddot{q}$$

Forward recursion

i=1:n

$$f_{i-1,i} = M_i e^{S_i q_i}$$

$$V_i = Ad_{f_{i-1,i}^{-1}} V_{i-1} + S_i \dot{q}_i$$

$$\dot{V}_i = Ad_{f_{i-1,i}^{-1}} \dot{V}_{i-1} + S_i \ddot{q}_i + ad_{V_i} S_i \dot{q}_i$$

Backward recursion

i=n:1

$$F_i = Ad^*_{f_{i,i+1}^{-1}} F_{i+1} + J_i \dot{V}_i - ad^*_{V_i} J_i V_i$$

$$\tau_i = \langle S_i, F_i \rangle$$

Recursive Forward Dynamics

Forward recursion

i=1:n

$$f_{i-1,i} = M_i e^{S_i q_i}$$

$$V_i = Ad_{f_{i-1,i}^{-1}} V_{i-1} + S_i \dot{q}_i$$

Backward recursion

i=n:1

$$\hat{J}_{i-1} = J_{i-1} + Ad^{\dagger}_{f_{i-1,i}^{-1}} (J_i - \frac{1}{\Omega_i}(\zeta_i \otimes \zeta_i))$$

$$b_{i-1} = -ad^{*}_{V_{i-1}} J_{i-1} V_{i-1} + Ad^{*}_{f_{i-1,i}^{-1}} (c_i + \frac{T_i - \langle S_i, c_i \rangle}{\Omega_i} \zeta_i)$$

$$c_{i-1} = \hat{J}_{i-1} ad_{v_{i-1}} S_{i-1} \dot{q}_{i-1} + b_{i-1}$$

$$\zeta_{i-1} = \hat{J}_{i-1} S_{i-1}$$

$$\Omega_{i-1} = \langle S_{i-1}, \zeta_{i-1} \rangle$$

Forward recursion

i=1:n

$$\dot{q}_i = \frac{1}{\Omega_i} \left(\tau_i - \left\langle Ad_{f_{i-1,i}^{-1}} \dot{V}_{i-1}, \varsigma_i \right\rangle - \langle S_i, c_i \rangle \right)$$

$$\dot{V}_i = Ad_{f_{i-1,i}^{-1}} \dot{V}_{i-1} + S_i \ddot{q}_i + ad_{V_i} S_i \dot{q}_i$$

Comparison to Spatial Notation

- Lie group notation

$$\hat{J}_{i-1} = J_{i-1} + Ad^{\dagger}_{f^{-1}_{i-1,i}} \left(\hat{J}_i - \frac{(\hat{J}_i S_i) \otimes (\hat{J}_i S_i)}{\langle S_i, \hat{J}_i S_i \rangle} \right)$$

- Spatial notation by Featherstone

$$I^A_{i-1} = I_{i-1} + {}_{i-1}X_i \left(I^A_i - \frac{I^A_i s_i s_i^T I^A_i}{s_i^T I^A_i s_i} \right) {}_i X_{i-1}$$

Implementing Lie Group

- Lie group elements can be represented as a matrix form.
- However it is not a clever idea to use matrices when computing operations on Lie group.

Implementing Lie Group

$$Ad_T V = (Rw,\ p \times Rw + Rv)$$

$$= \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix}$$

$$Ad_T^{\dagger} J = \left(R^T (I + [r][p] + [p][r] - m[p]^2) R,\ m,\ R^T (r - mp)\right)$$

$$= \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \begin{bmatrix} I & [r] \\ -[r] & m1 \end{bmatrix} \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix}^T$$

Operation Count

	Geometric computation		Matrix based computation		ratio	
	x	+	x	+	x	+
$Ad_T V$	24	18	63	48	**38%**	**37.5%**
$ad_V W$	18	12	36	30	**50%**	**40%**
JV	24	18	36	30	**67%**	**60%**
$Ad_T^{\dagger} J$	101	66	459	378	**22%**	**17%**

SE(3) Class

- Substitution `SE3 ← SE3`
- Multiplication `SE3 ← SE3 * SE3`
- Inverse `SE3 ← Inv(SE3)`
- Exponential map `SE3 ← Exp(se3)`
- Adjoint map `se3 ← Ad(SE3, se3)`

`dse3 ← dAd(SE3, dse3)`

se(3) Class

- Substitution `se3 ← se3`
- Addition `se3 ← se3 + se3`
- Multiplication `se3 ← scalar * se3`
- Reciprocal product `scalar ← se3 * dse3`
- Lie bracket `se3 ← se3 ^ se3`

Generalized Inertia Class

- Substitution `Inertia ← Inertia`
- Multiplication `dse3 ← Inertia * se3`
- Transformation `Inertia ← Ad(SE3, Inertia)`

Articulated Inertia Class

- Substitution `AInertia ← Ainertia`

  `AInertia ← Inertia`
- Addition `AInertia ← Ainertia + AInertia`
- Multiplication `AInertia ← scalar * AInertia`

  `dse3 ← AInertia * se3`
- Kronecker product `AInertia ← dse3 * dse3`
- Transformation `Inertia ← Ad(SE3, Inertia)`

Implementing Algorithm

$$\hat{J}_{i-1} = J_{i-1} + Ad^{\dagger}_{f^{-1}_{i-1,i}} \left(J_i - \frac{(\hat{J}_i S_i) \otimes (\hat{J}_i S_i)}{\langle S_i, \hat{J}_i S_i \rangle} \right)$$

1. **dse3 ← AInertia * se3**
2. **AInertia ← dse3 * dse3**
3. **scalar ← se3 * dse3**
4. **AInertia ← scalar * AInertia**
5. **AInertia ← Ainertia - AInertia**
6. **AInertia ← Ad(SE3, AInertia)**
7. **Ainertia ← Inertia + AInertia**

$$\zeta = \hat{J}_i S_i$$

$$\Gamma = \zeta \otimes \zeta$$

$$\Omega = \langle S_i, \zeta \rangle$$

$$\Theta = \frac{1}{\Omega} \Gamma$$

$$\Phi = \hat{J}_i - \Theta$$

$$\Psi = Ad^{\dagger}_{f^{-1}_{i-1,i}} \Phi$$

$$\hat{J}_{i-1} = J_{i-1} + \Psi$$

Code snippet

$$\hat{J}_{i-1} = J_{i-1} + Ad^{\dagger}_{f_{i-1,i}^{-1}} \left(\hat{J}_i - \frac{(\hat{J}_i S_i) \otimes (\hat{J}_i S_i)}{\langle S_i, \hat{J}_i S_i \rangle} \right)$$

```
SE3 f[n];

Inertia J[n];   AIneria AJ[n];

se3 S[n];       dse3 ksi;

...

ksi = AJ[i] * S[i];

AJ[i-1] = J[i-1] + Ad(Inv(f[i]), AJ[i] -
    (1/(S[i] * ksi)) * (ksi * ksi));
```

Lie Group Formulation

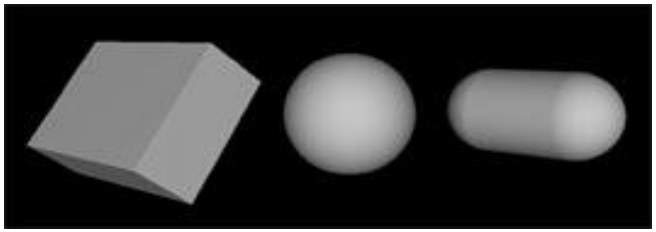
- looks beautiful
- offers further insight into the high-level structure of kinematics and dynamics
- is computationally efficient when implemented carefully

VirtualPhysics

- C++ library for realtime multi-body dynamics simulation
- based on Lie group formulation
- arbitrary joint modeling
- collision detection between geometric primitives
- analytic or penalty based collision response
- contact response by solving LCP
- mixed dynamics: working with kinematically controlled objects
- collision like joint limit
- dynamic configuration: excessive force can break joints

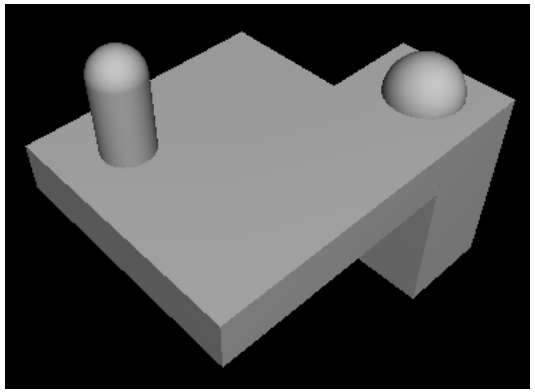
vpGeom

- Abstract class to represent primitive geometries



vpBox vpSphere vpCapsule

- Consider a rigid body as a set of primitive geometries



- Target of collision detection

vpBody

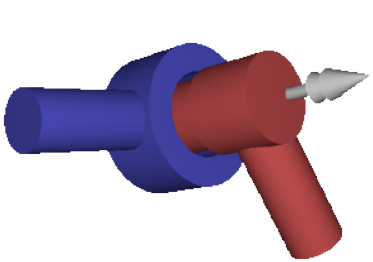
- A class to model rigid bodies
- consists of several vpGeom instances
- can be connected to another body with vpJoint.
- has a reference to vpMaterial
- can be set ground

vpMaterial

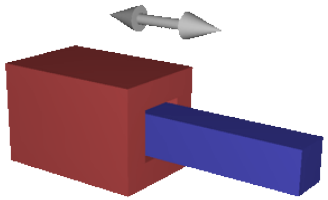
- A class about material properties
  - density
  - coefficient of restitution
  - coefficient of static/dynamic/spinning friction

vpJoint

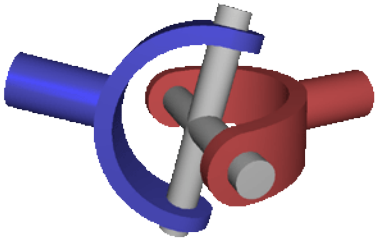
- Abstract class to connect two adjacent bodies
- Can be broken
  - , when a normal force/torque applied to the joint exceeds its threshhold.



vpRJointv

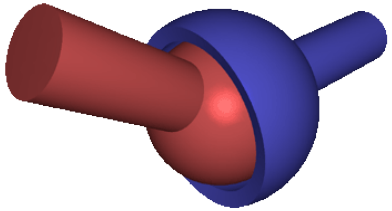


pPJoint



vpUJoint

vpSJoint



vpBJoint

vpWJoint

vpNDOFJoint

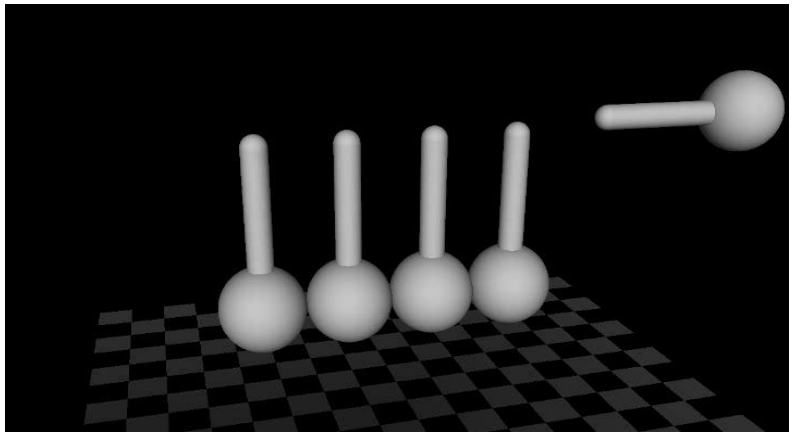
vpWorld

- A class to model a virtual world
- Add bodies to vpWorld
- Set global attributes such as gravity
- Set simulation attributes such as integration time step
- Initialize and run

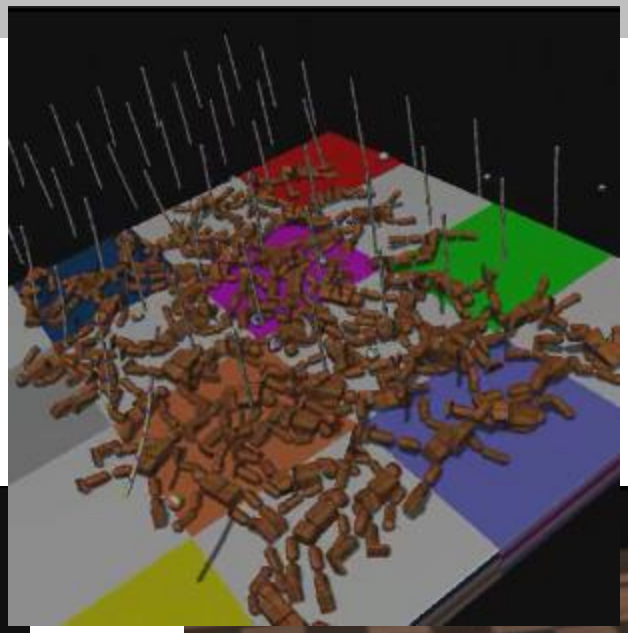
Example: a pedulum

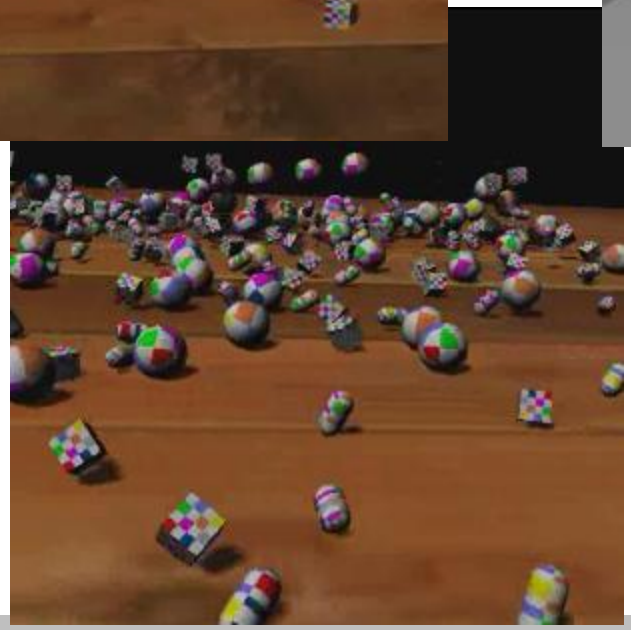
- Add a rotational joint

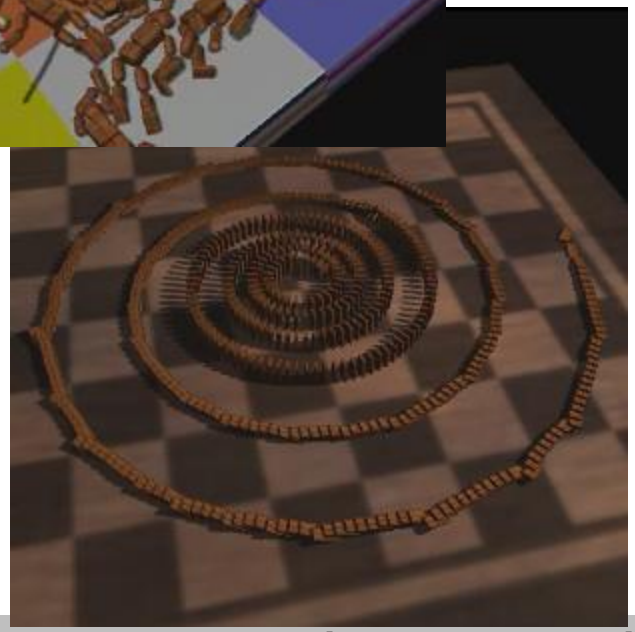
```
vpBody G, B;
vpSphere sph(0.5);
vpCapsule rod(0.2,2);
B.AddGeometry(&rod, Vec3(0));
B.AddGeometry(&sph, Vec3(0,0,-1));
vpRJoint J;
G.SetJoint(&J, Vec3(0));
B.SetJoint(&J, Vec3(0,0,1));
J.SetAxis(Vec3(0,1,0));
G.SetGround();
vpWorld world;
world.AddBody(&G);
world.Initialize();
while ( true )
    world.StepAhead();
```



http://virtualphysics.imrc.kist.re.kr







Thank you

Any Question?

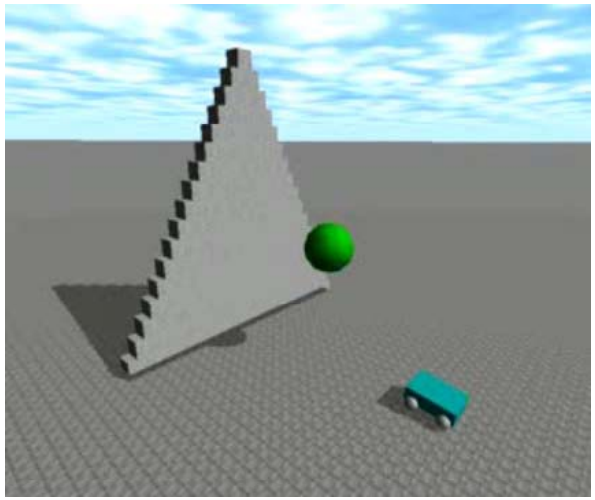
Physically Based Character Simulation

Yoonsang Lee,
Movement Research Lab., Seoul National University

Why Simulation?

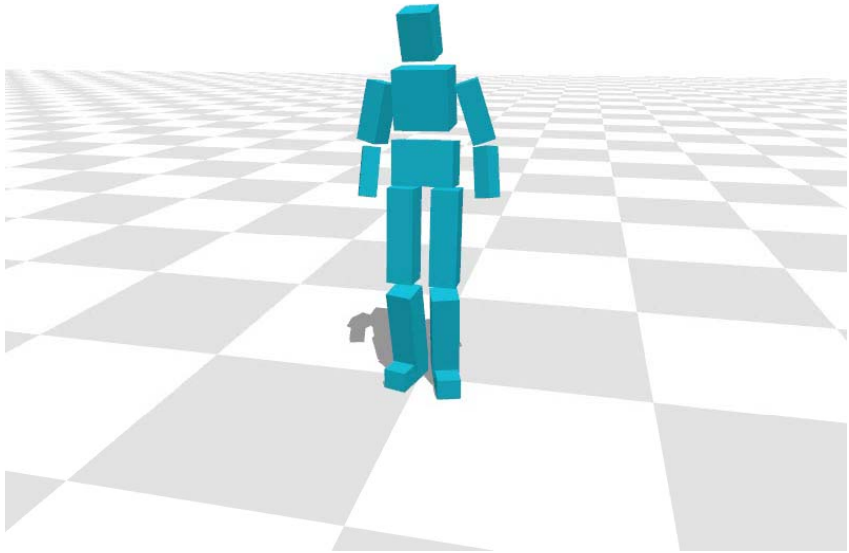
- Animating characters in dynamic environments is difficult
  - Physically correct interaction with environment is required

- Simulation automatically produces physically valid results



Character Control

- But, simulation of character requires controller that computes internal muscle forces (joint torques)

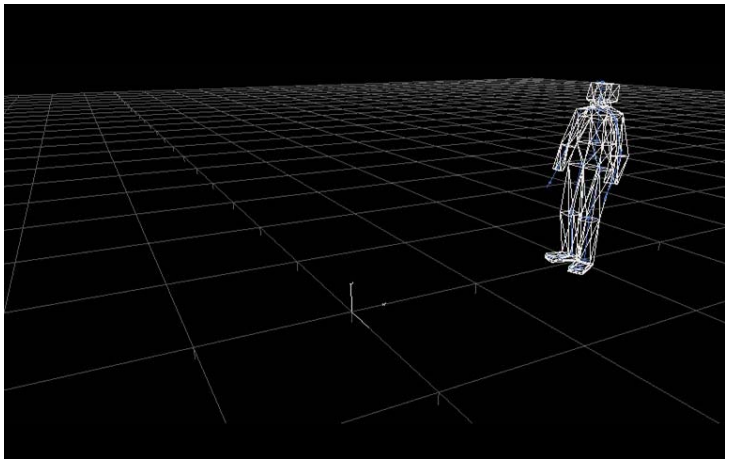


Simulation without internal forces

Character Control

- Character control is difficult problem
  - Maintaining balance
  - Reproducing various style of behavior

- Humanoid Robots : A lot hand tuning..



ASIMO

Honda Motor Company

QURIO

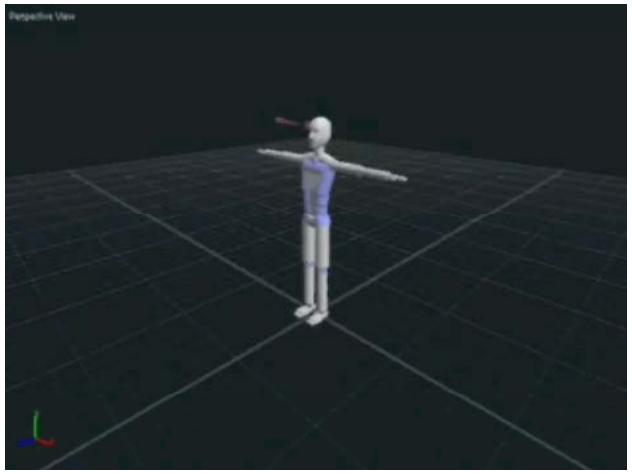
SONY CORPORATION

HUBO

KAIST

Character Control

- Current commercial physics engines supports character simulation of passive motion

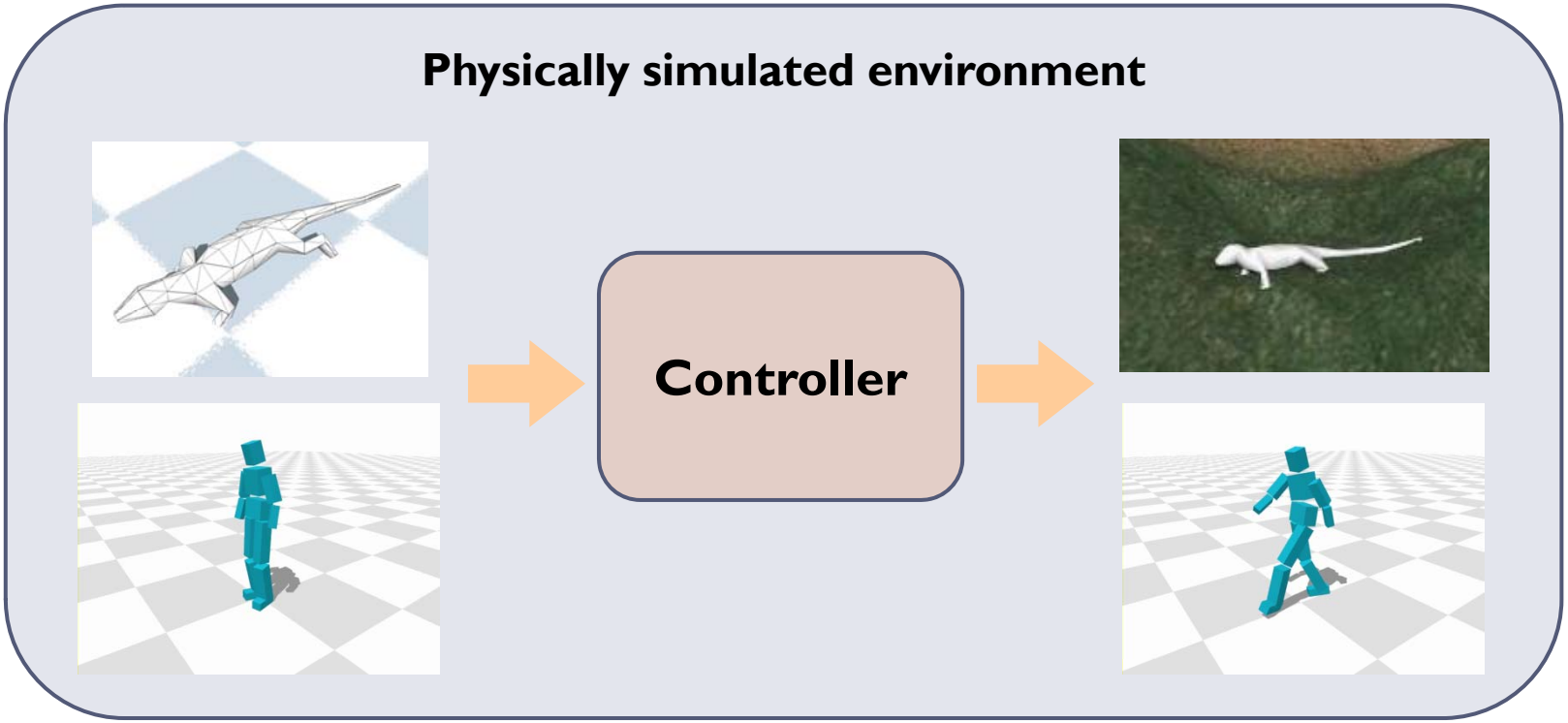


Endorphin

- Controller for active motion also can be applied to other area like robotics

Goal

- Control various characters interactively to perform desired motions in physically simulated environment



Goal

- Data-driven approach : use motion capture data
  - As a "guidance" of controller
  - Iguana, bird, human actor, …

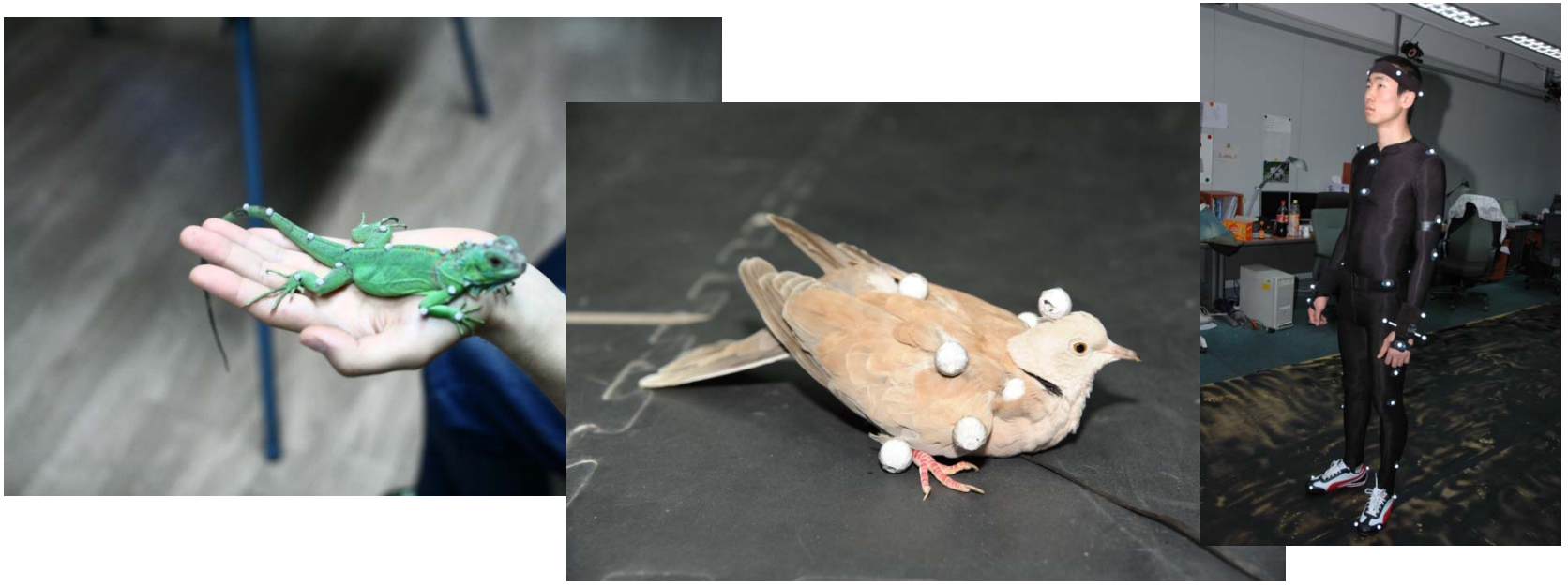


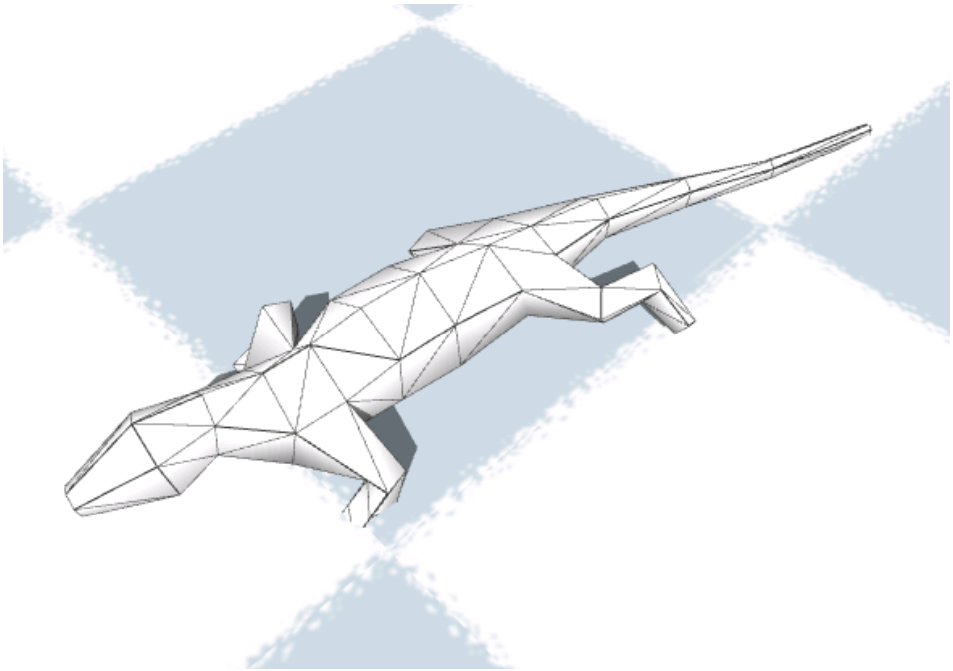
Table of Contents

Iguana Character Simulation

Iguana Motion Capture

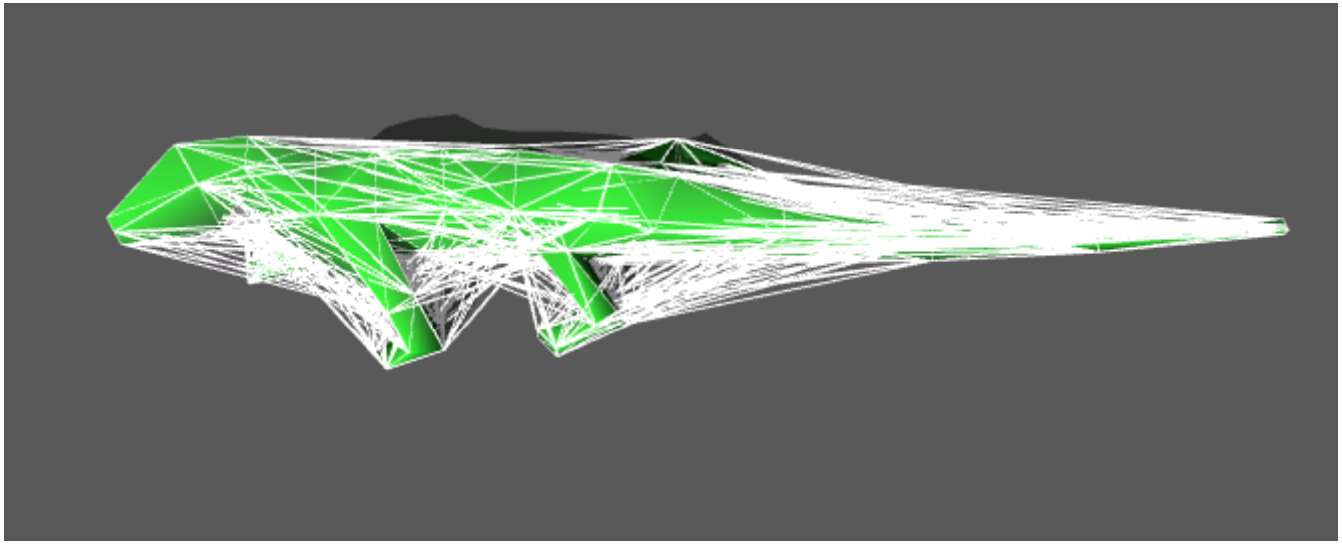
Iguana Simulation

- Target mesh
  - Skinning with dual quaternions [Kavan et al. 2008]

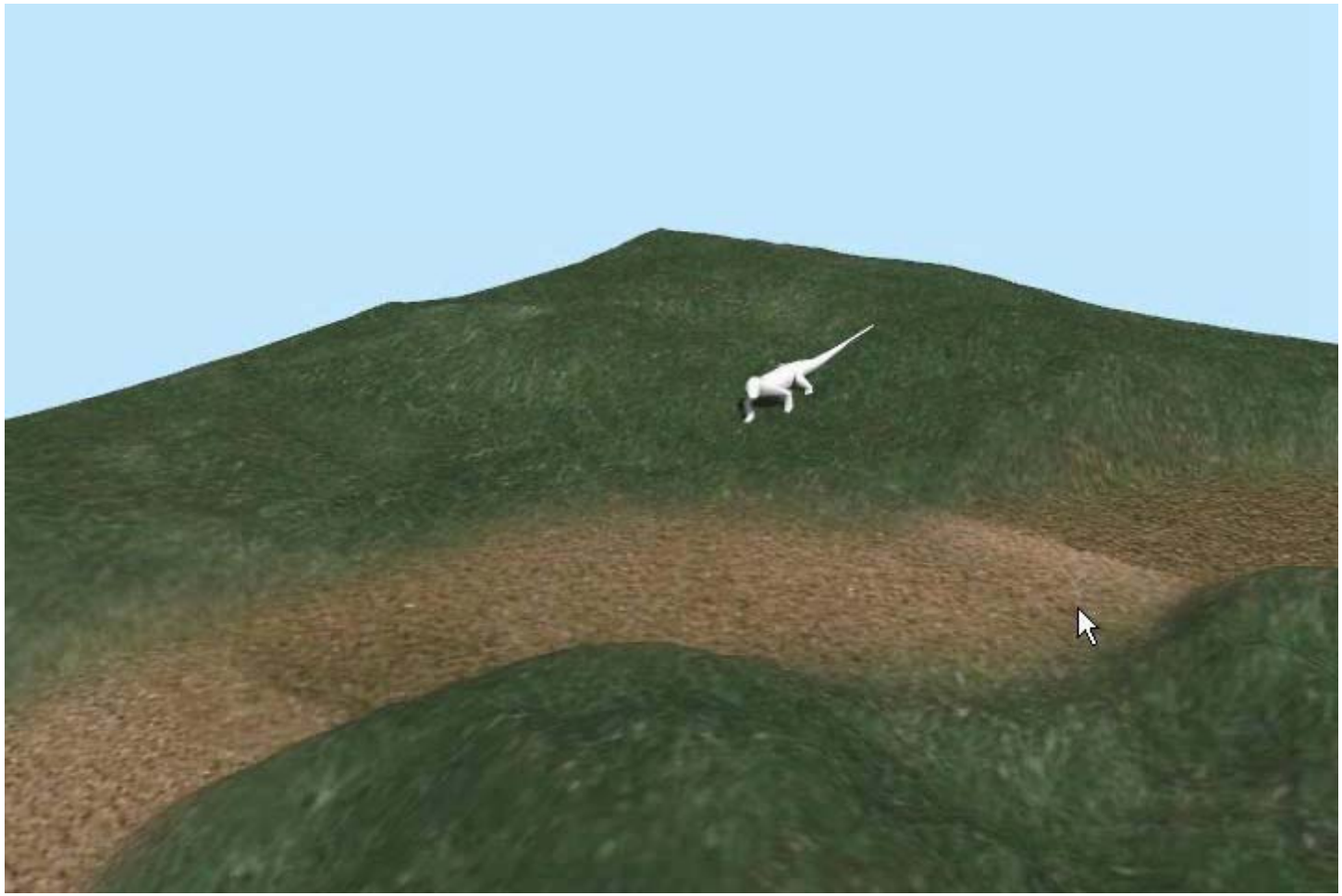


Iguana Simulation

- Flexible body model
  - Mass : proportional to the total area of neighboring triangles
  - Spring : between all pairs of vertices whose distance < 4

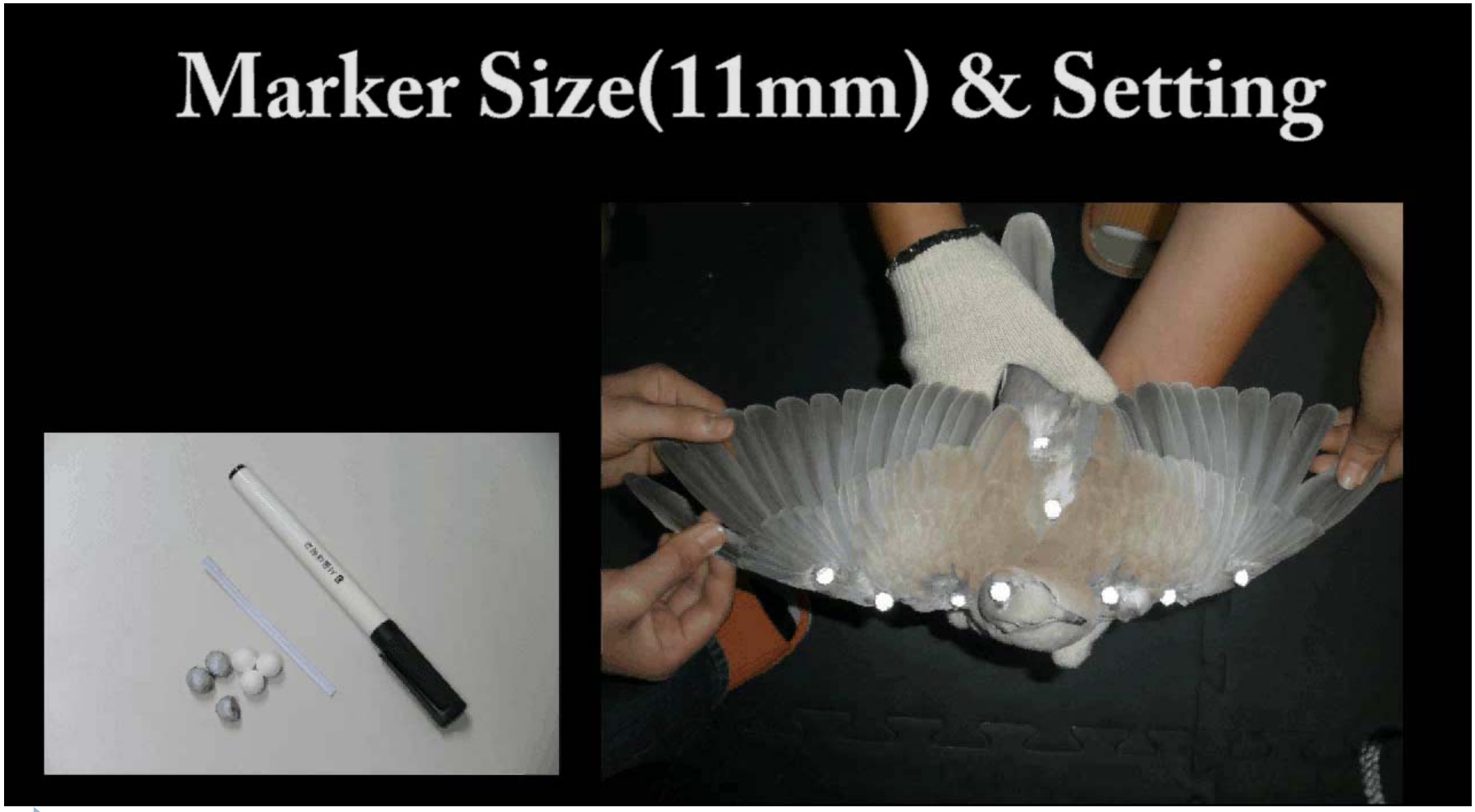


Iguana Simulation



Bird Character Simulation (Motion Capture)

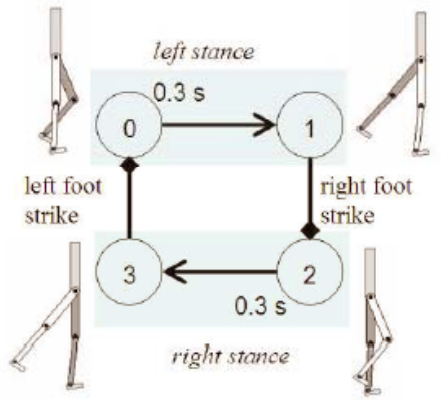
Bird Motion Capture



Biped Character Simulation

Related Work

- State machine approach
  - SIMBICON [Yin et al. 2007] : Relatively simple & robust
  - Hard to reproduce natural results & various human skills

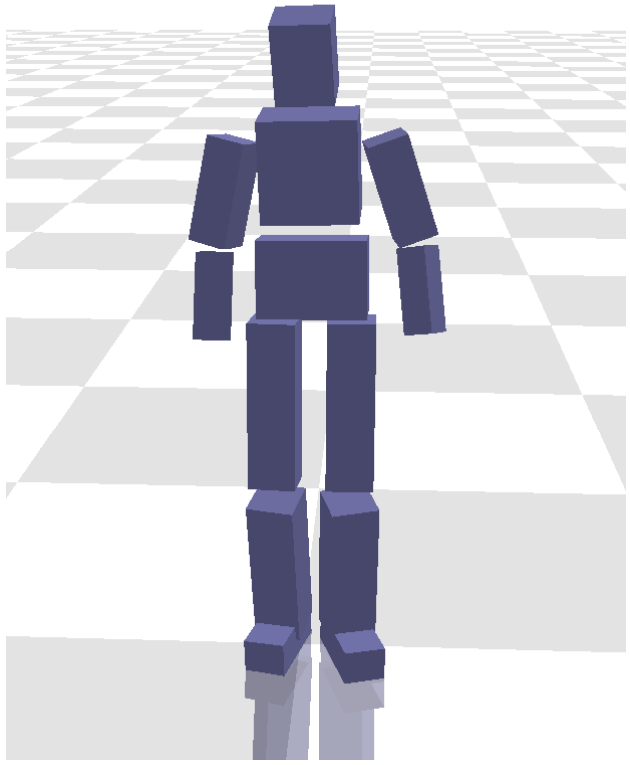


- Data-driven approach
  - Use mocap data as high quality & various human skills sources
  - Various methods proposed
    - Rectified motion set & simple regression model [Sok et al. 2007], QP [da Silva et al. 2008a], LQR [da Silva et al. 2008b], NQR [Muico et al. 2009]…

Our Approach

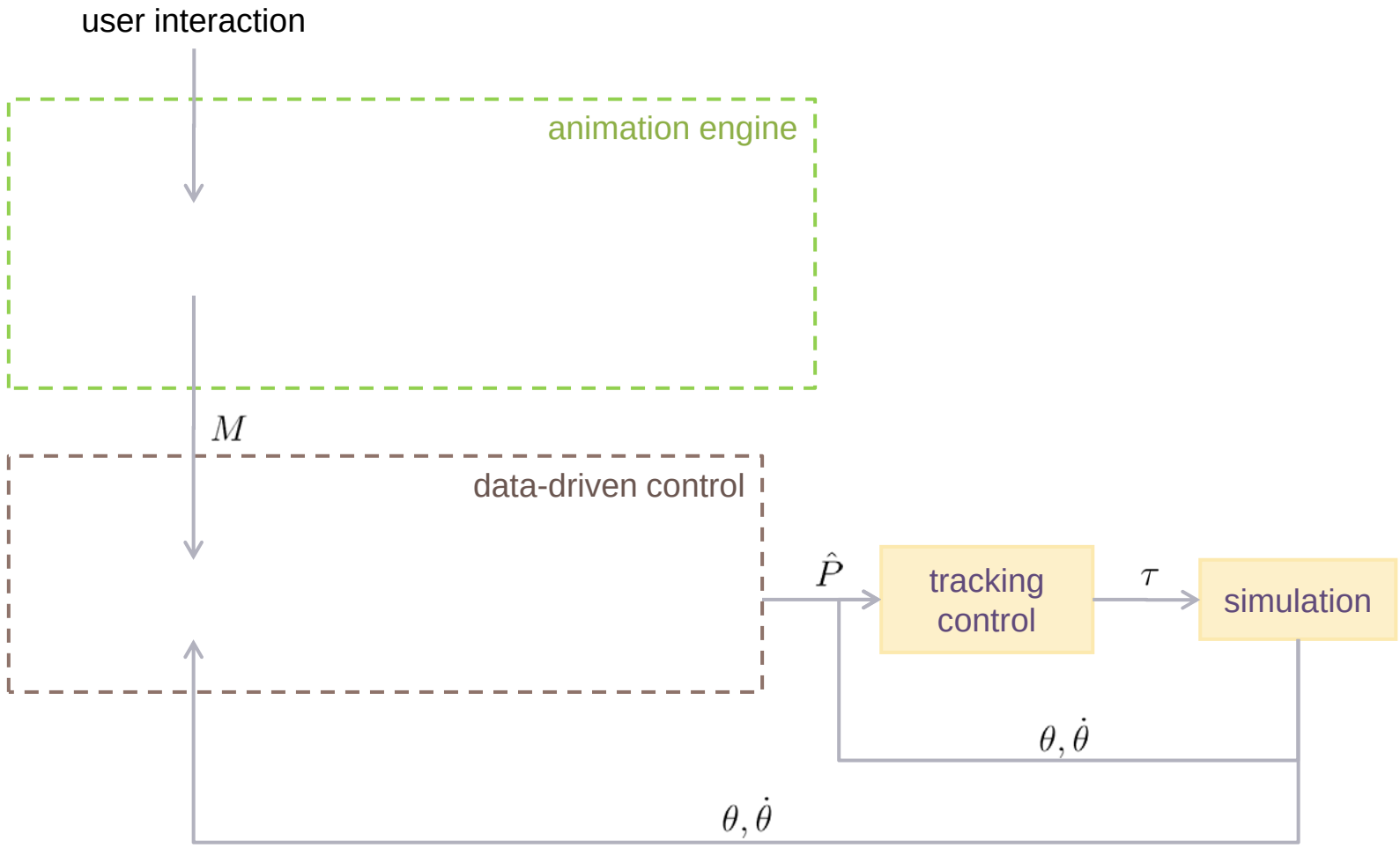
- Data-driven
  - Take advantage of high quality result & various human skill sources
- Focuses on reference motion
  - Modulate reference motion for synchronization & balancing
- Advantage
  - Existing data-driven animation techniques : editing, blending, motion graphs..
  - Simple dynamic controller : just tracking controller

Biped Character

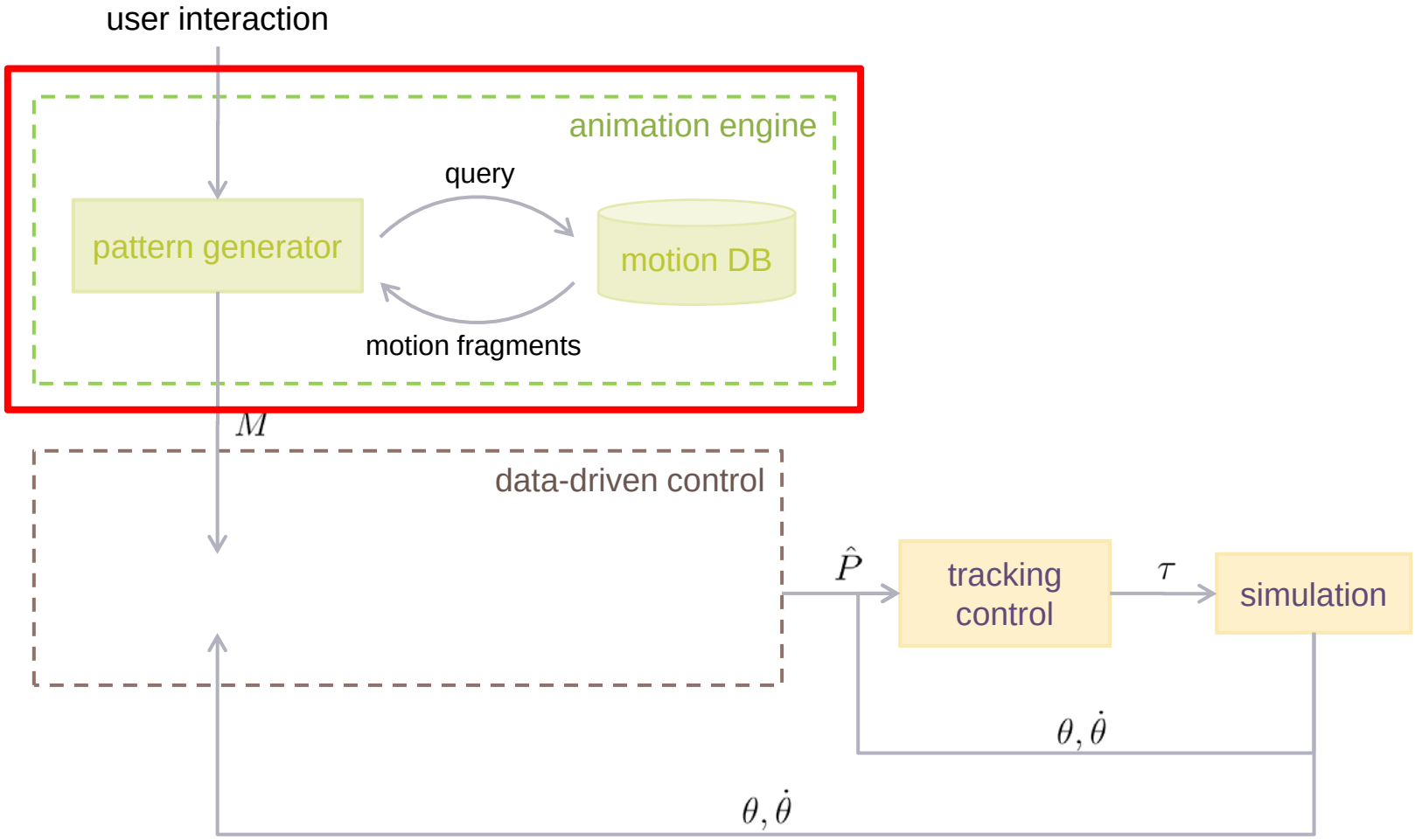


- 13 rigid body parts
- 12 ball-socket joints
- Total DOF is 42 (including 6 DOF of floating root joint)

Overview

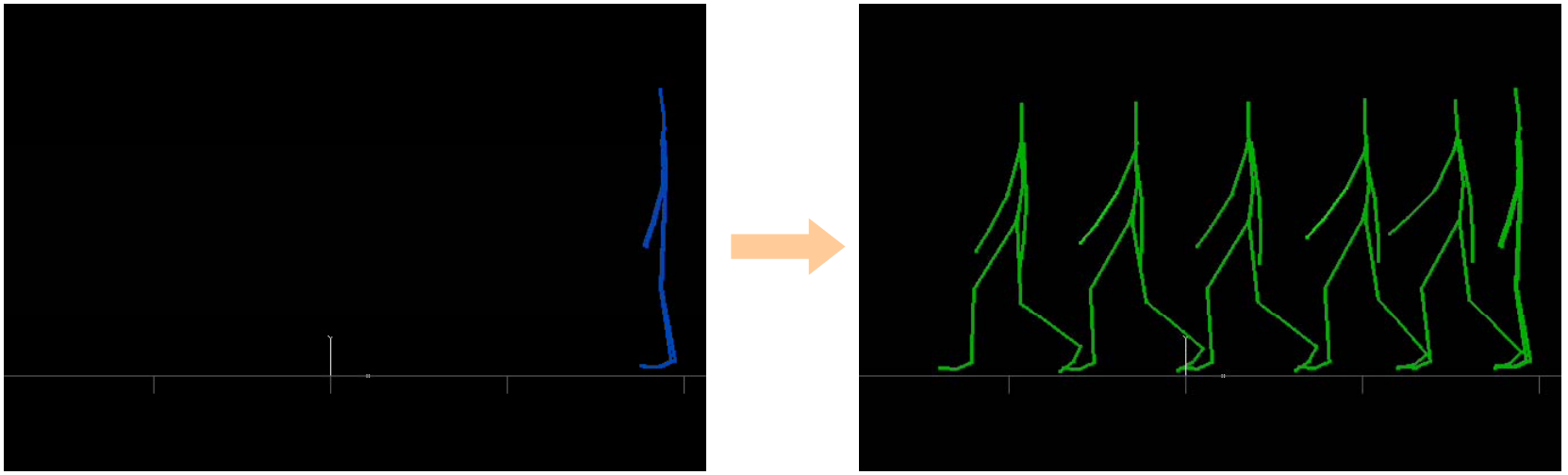

user interaction
animation engine
M
data-driven control
$\hat{P}$
tracking control
τ
simulation
$\theta, \dot{\theta}$
$\theta, \dot{\theta}$

Animation Engine



Motion Database

- Segment motion data into fragments where ground contact changes

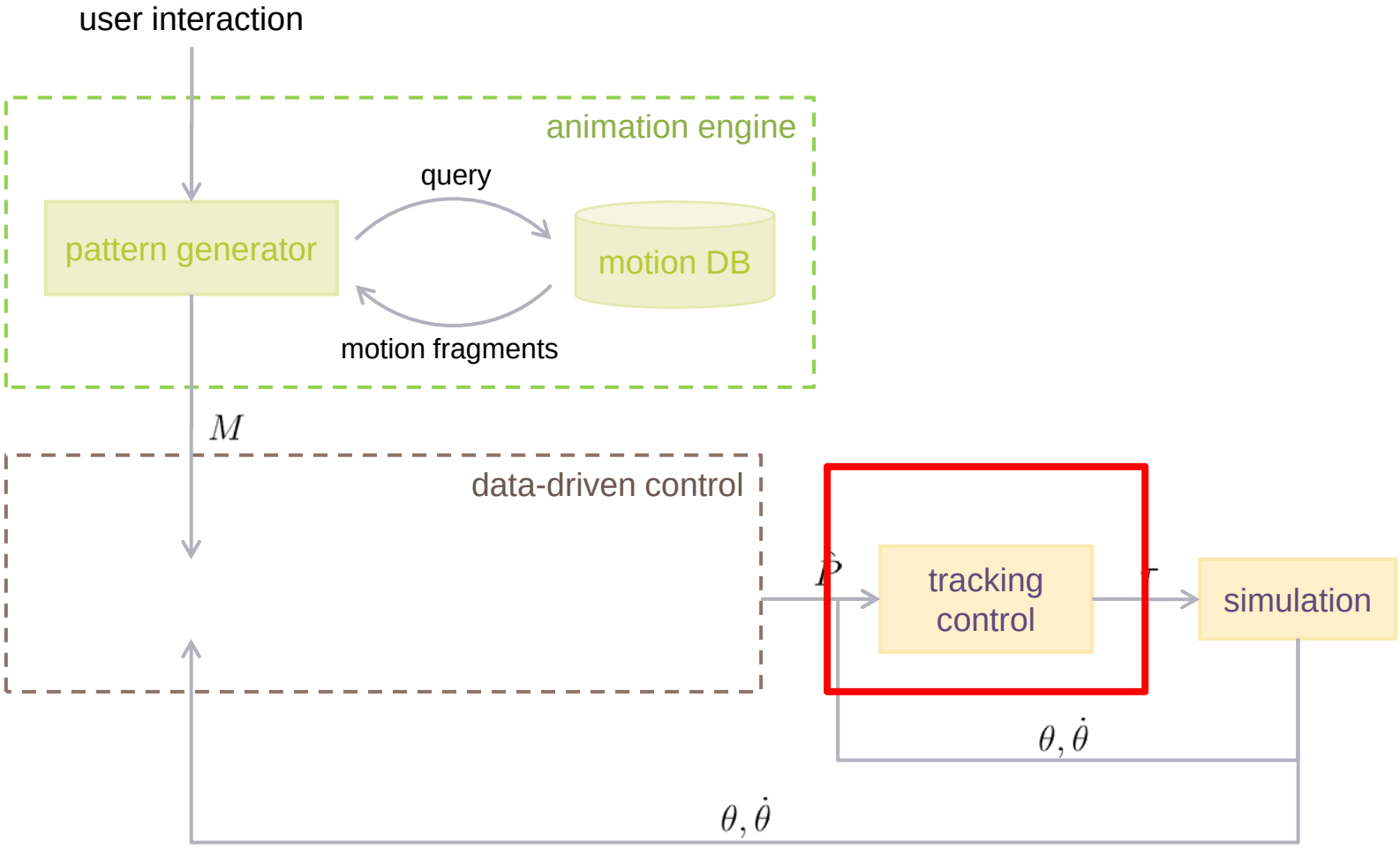


motion capture data

motion fragments

- Motion fragments maintained in a directed graph to allow transitioning between them

Tracking Control



Tracking Control

- Attempts to follow reference motion by

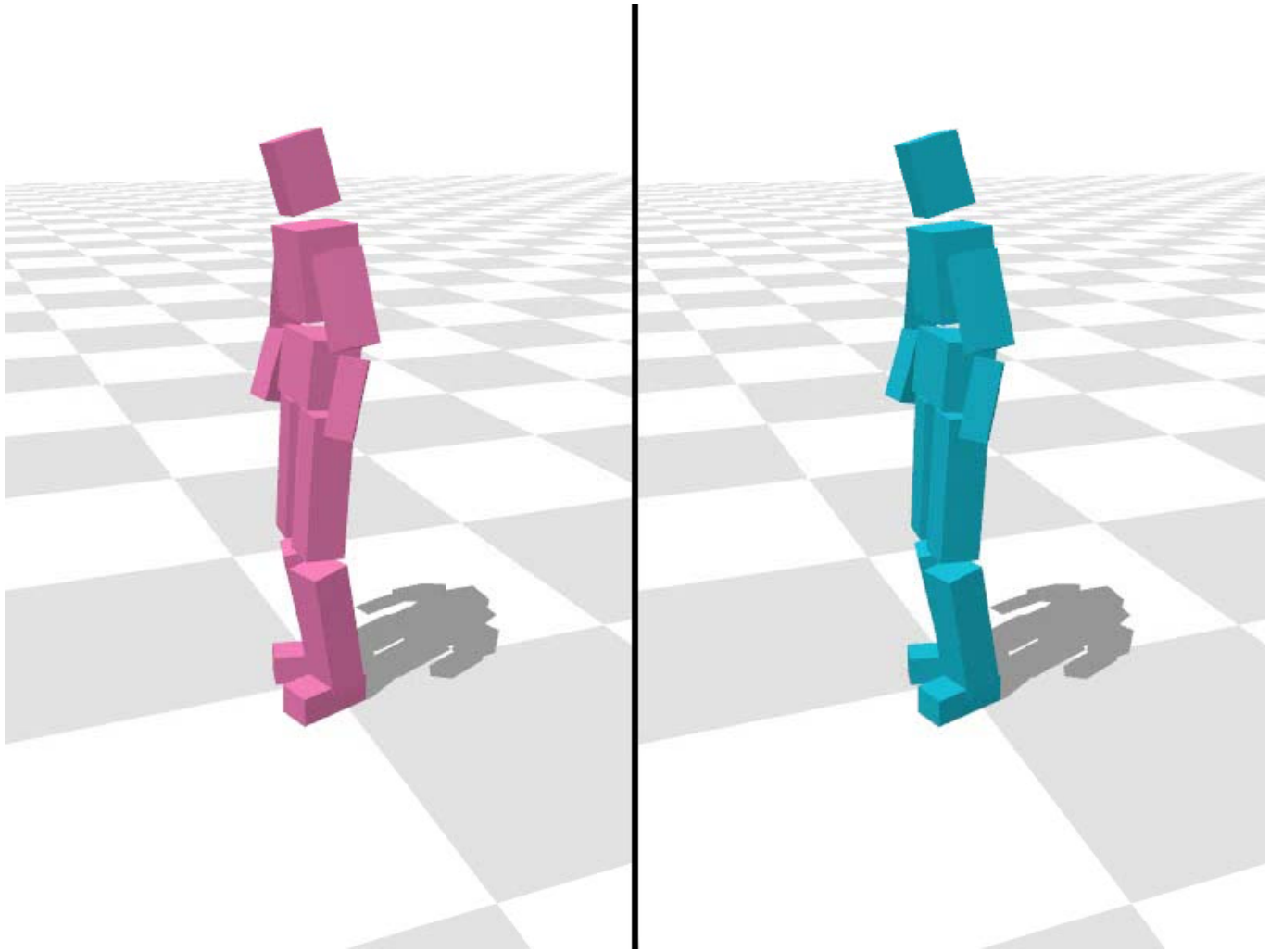
$$\ddot{\theta}_{desired} = k_t(\theta_r - \theta) + k_v(\dot{\theta}_r - \dot{\theta})) + \ddot{\theta}_r$$

$\theta_r, \dot{\theta}_r, \ddot{\theta}_r$: joint angles, joint velocities, joint accelerations from reference motion data

- Compute joint torques using inverse dynamics
- Penalty method for ground reaction force

- We used Virtual Physics to solve inverse dynamics and forward dynamics simulation

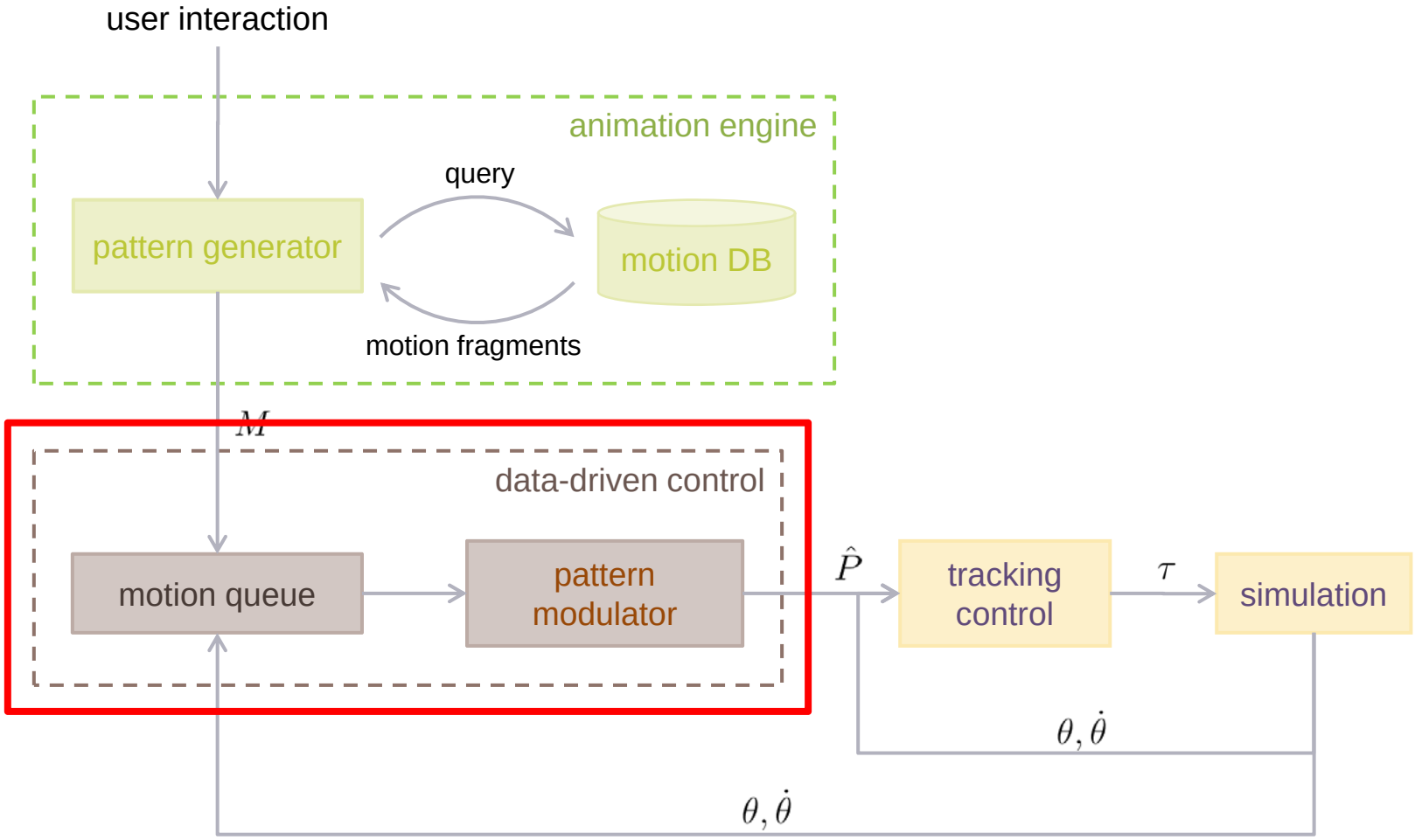
Tracking Result



motion capture reference

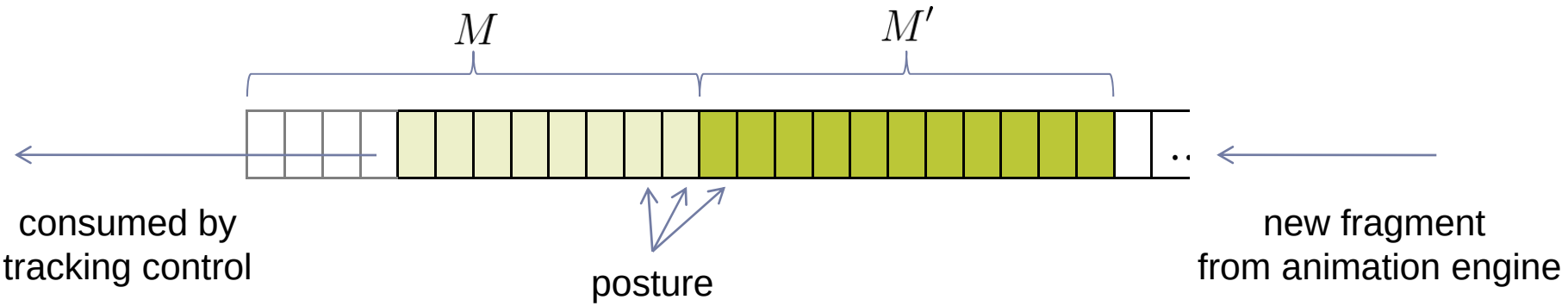
simulated (tracking only)

Data-Driven Control



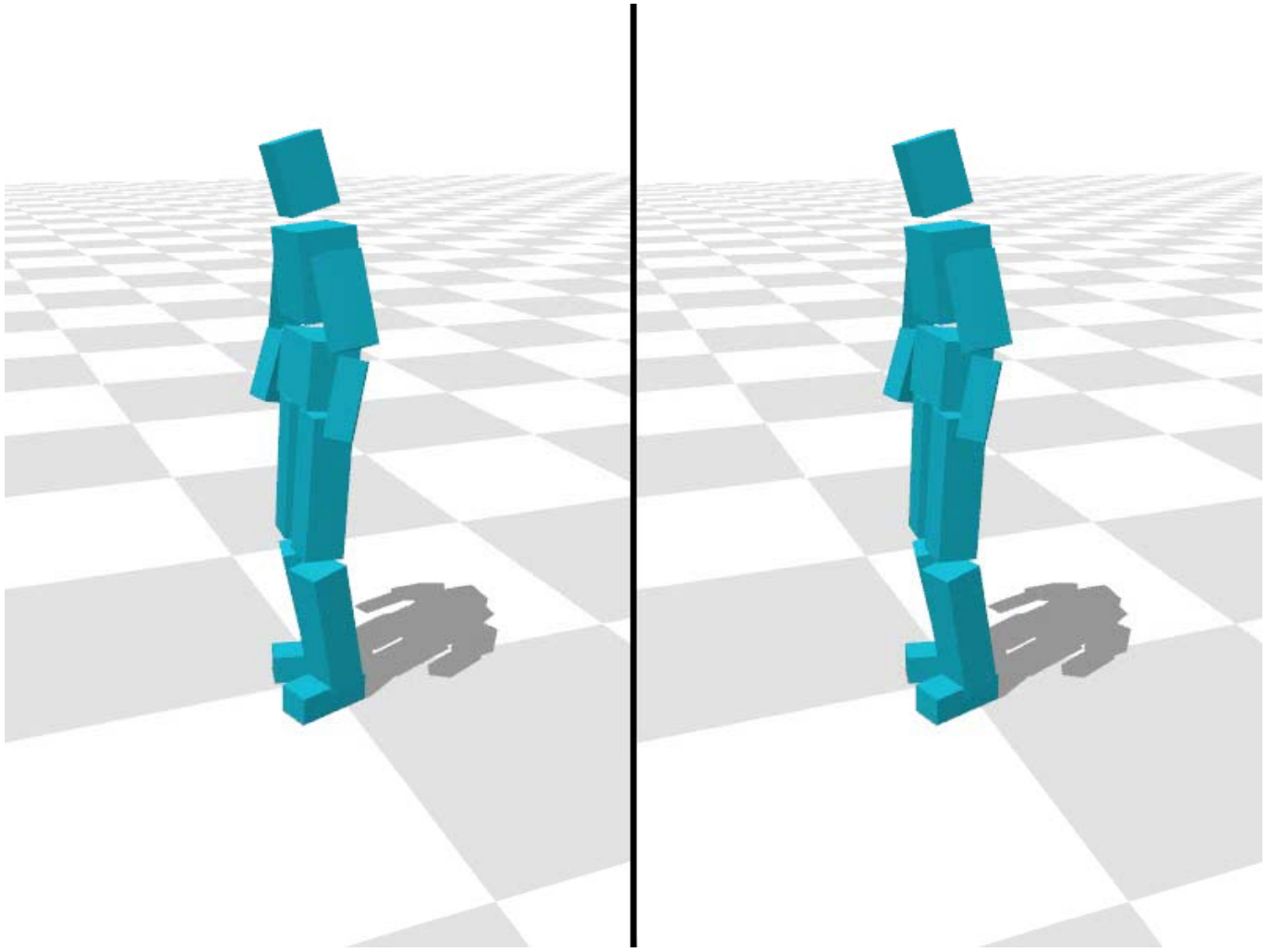
Synchronization

- Motion queue



- Synchronize reference motion & simulated biped at contact changes by inserting or deleting postures

Synchronization Result



simulated (tracking only)

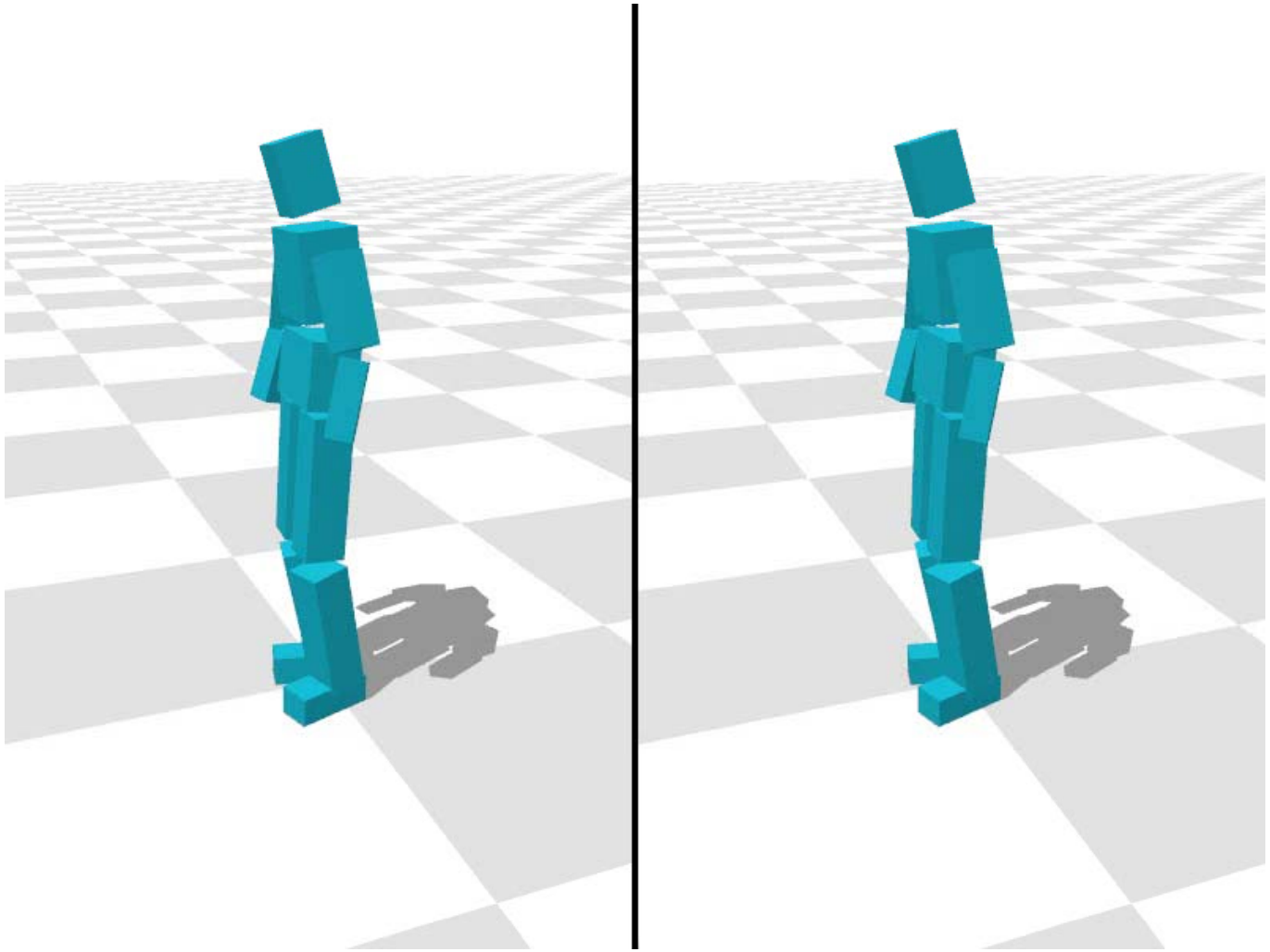
simulated (+ synchronization)

Balancing

- Compute continuous stream of target poses
  - based on current reference frame & simulated biped

- Feedback on Stance Hip
- Feedback on Swing Hip and Stance Ankle

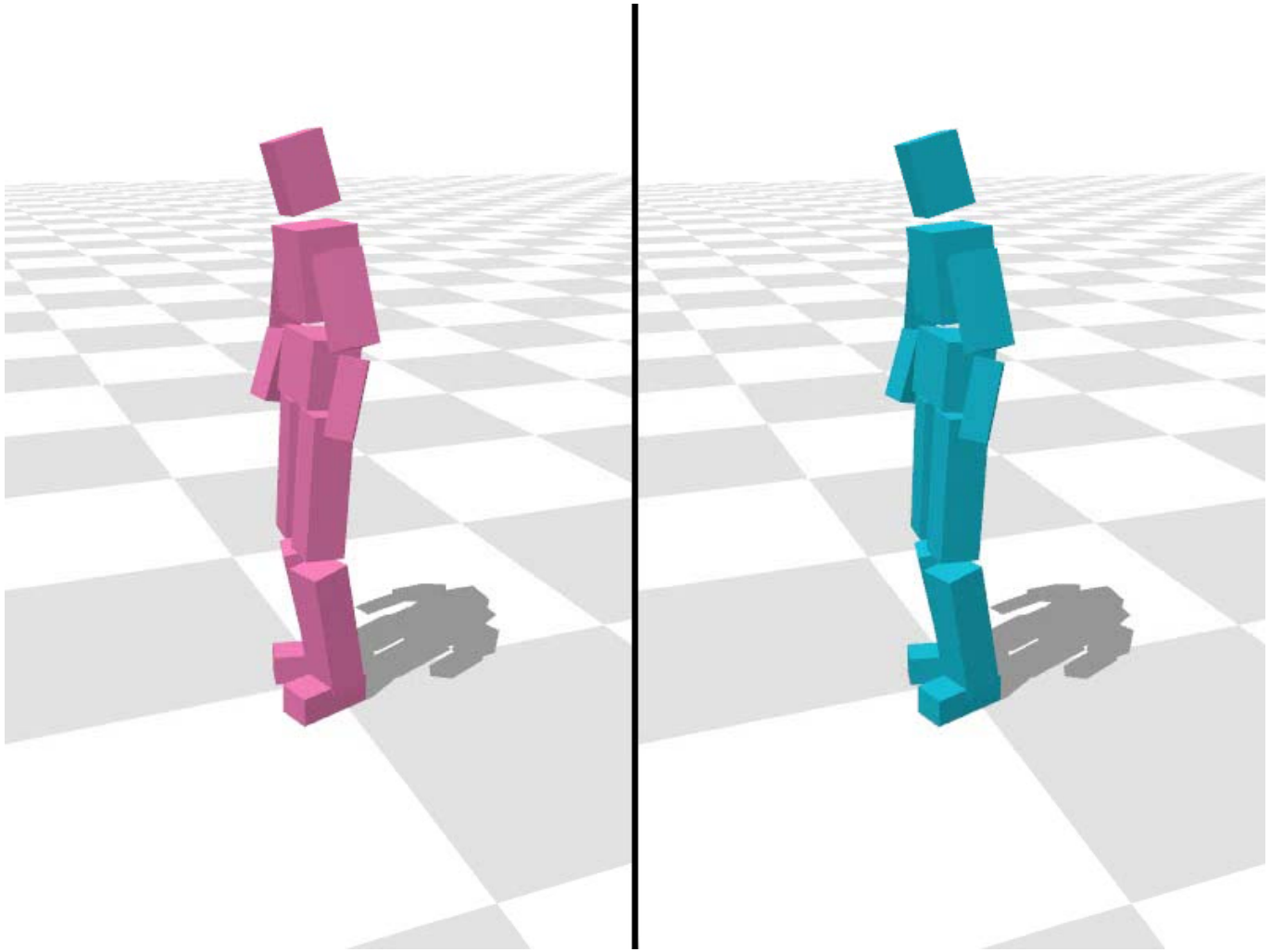
Balancing Result



simulated (+ synchronization)

simulated (+ synchronization + balancing)

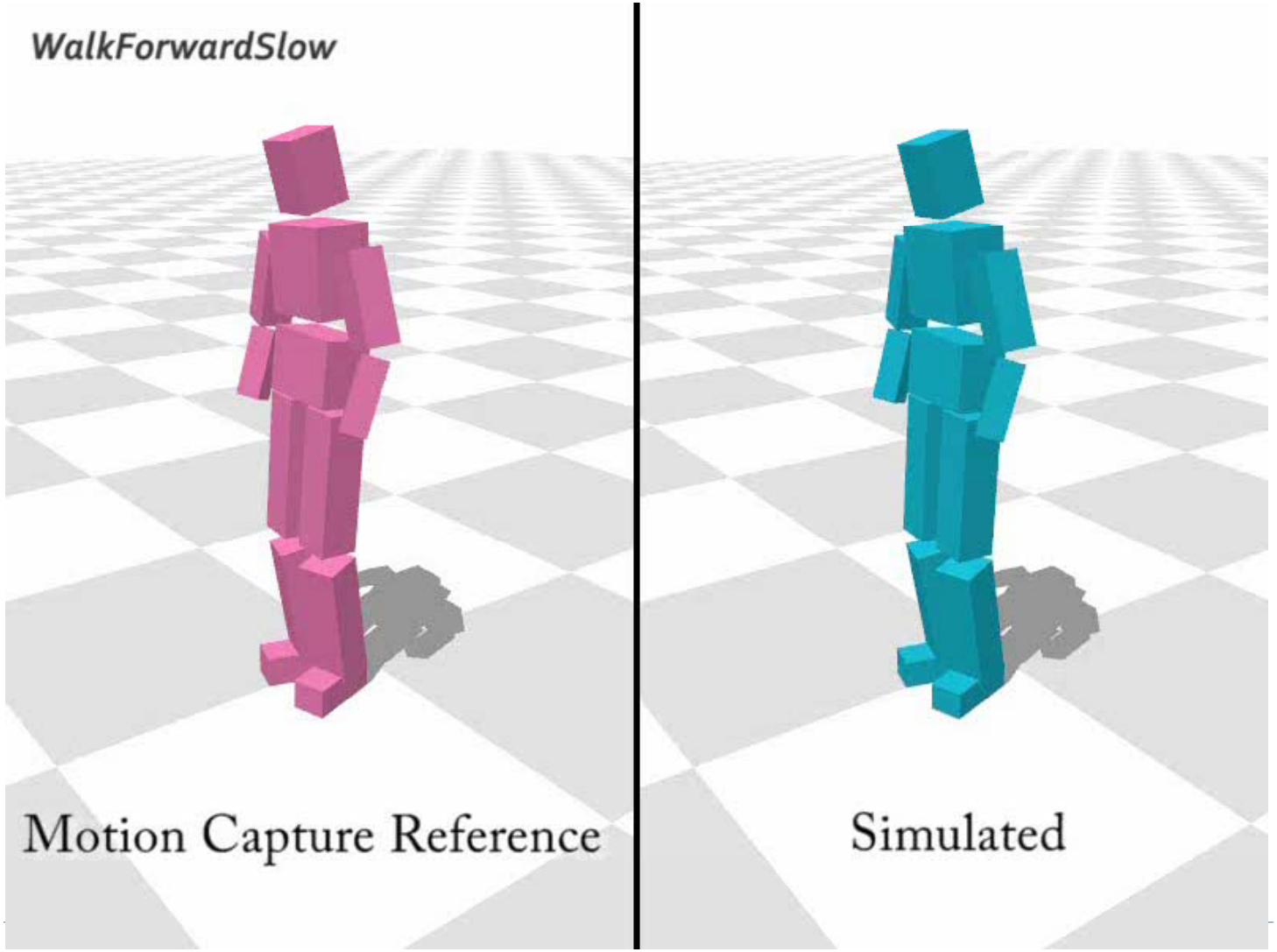
Balancing Result



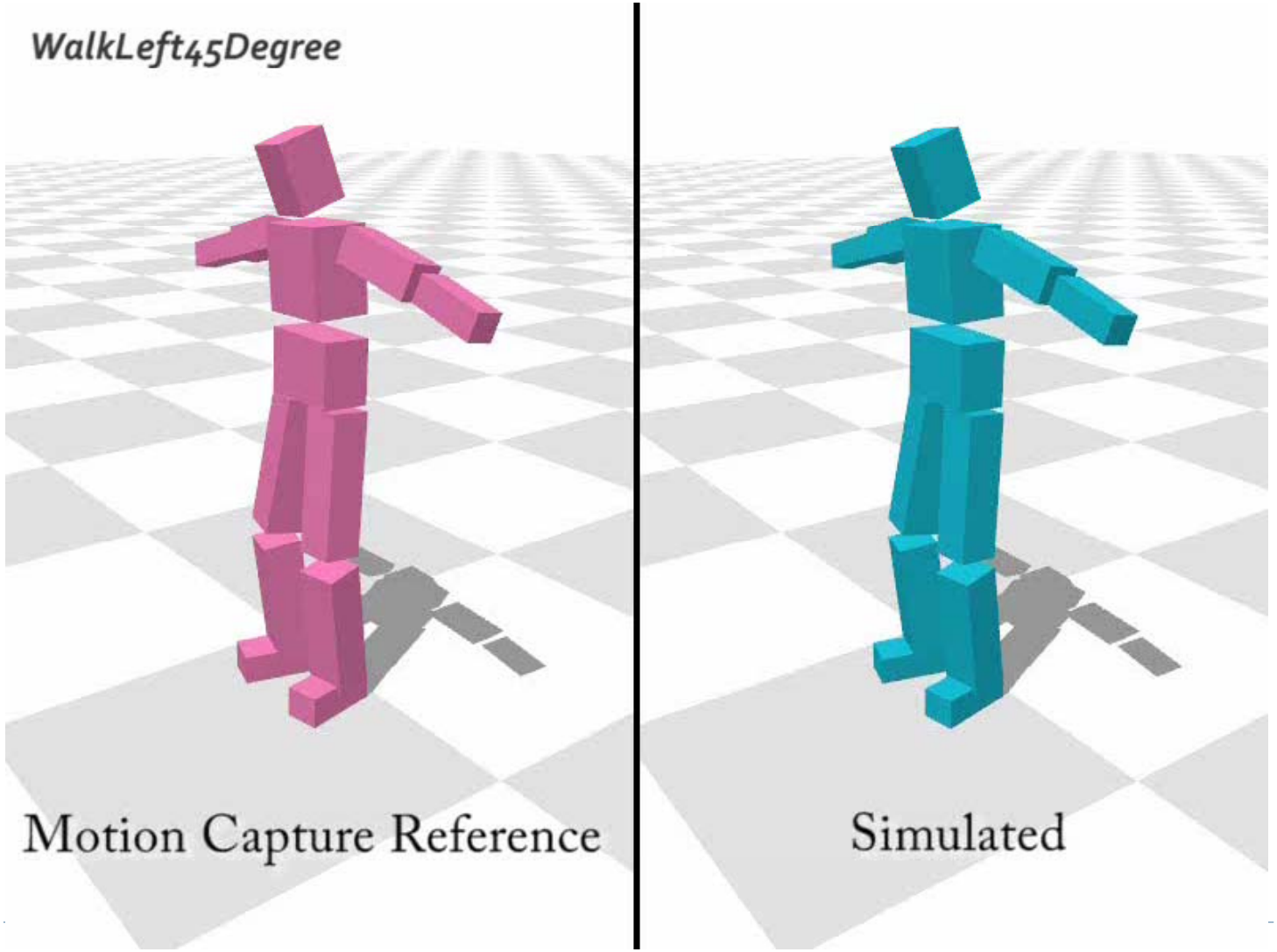
motion capture reference

simulated (+ synchronization + balancing)

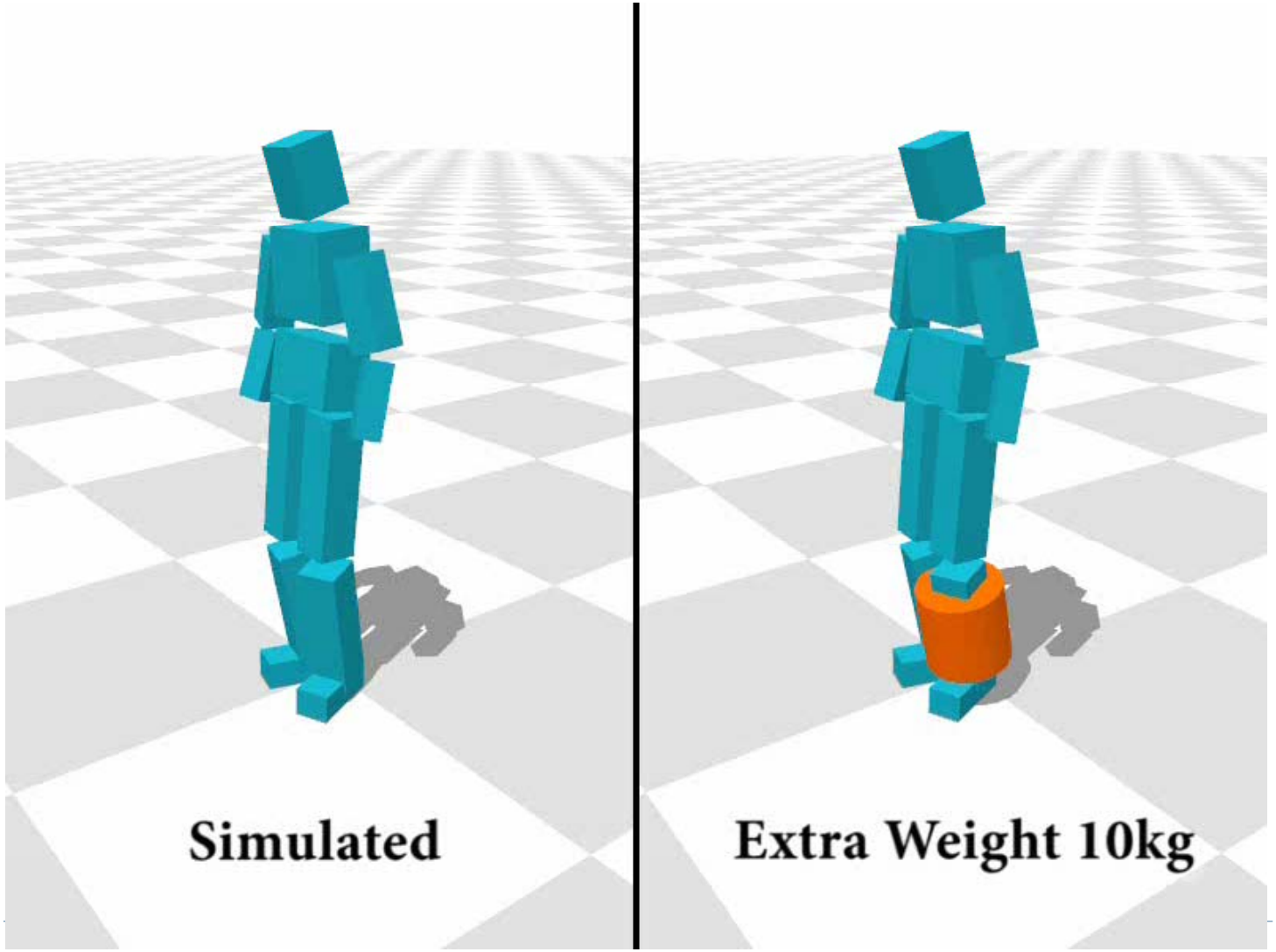
Result - Walking



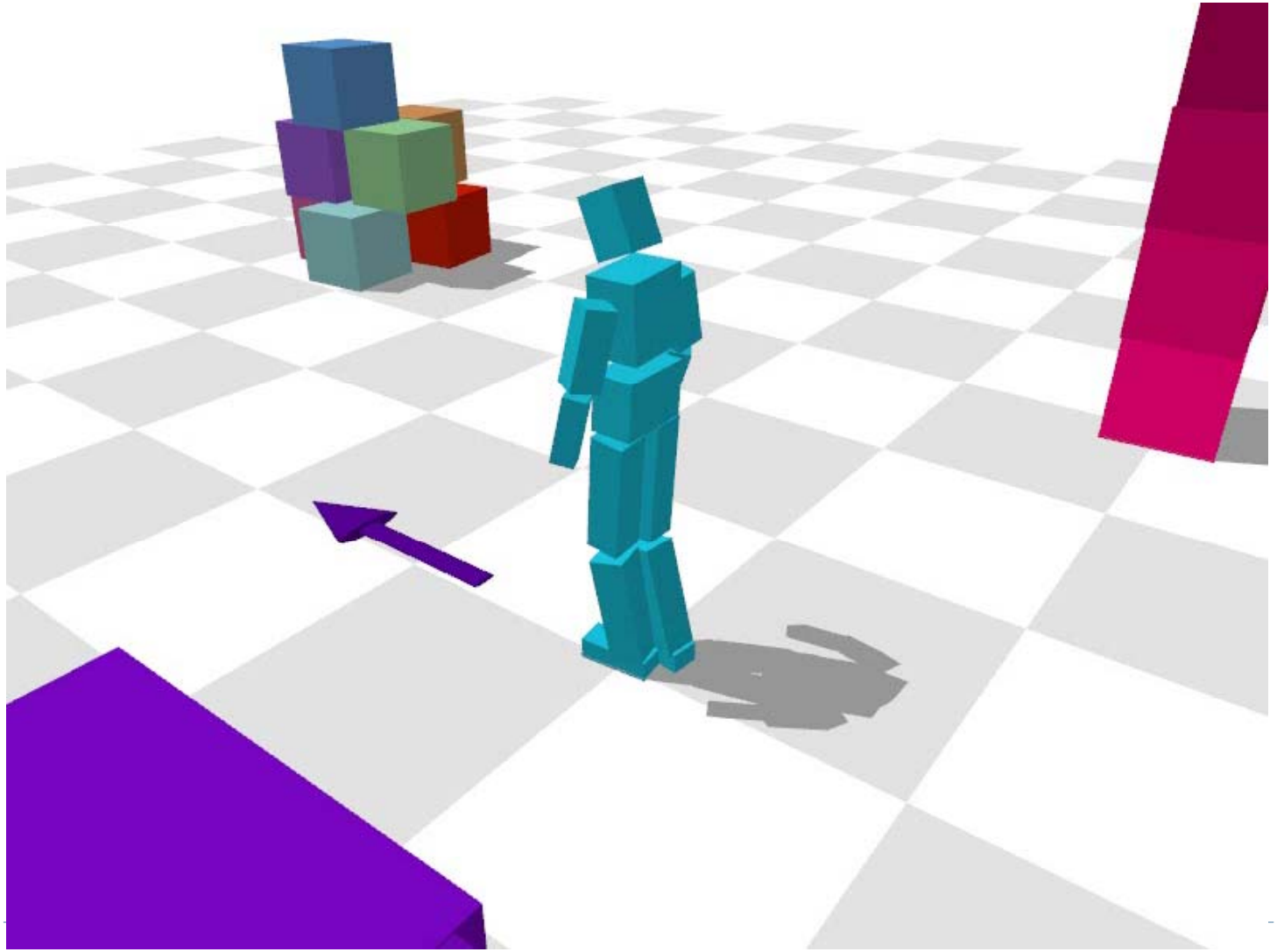
Result – Turning & Spinning



Result – Robustness Test



Result – Interactive Control



Interactive Collision Detection for Large-Scale Deformable Objects

Sung-Eui Yoon
Scalable Graphics Lab.
KAIST

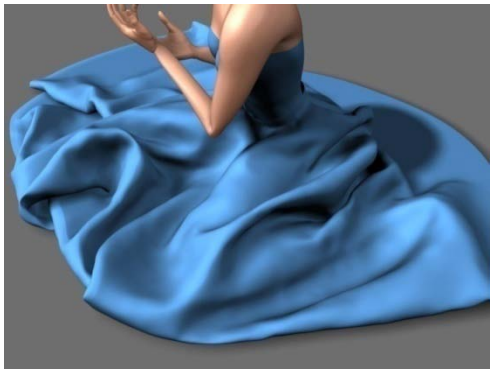
http://sglab.kaist.ac.kr/~sungeui/

KAIST

Collision Detection

- **Collision detection is used in various fields**
  - **Game, movie, scientific simulation and robotics**

<Figure from PIXAR>



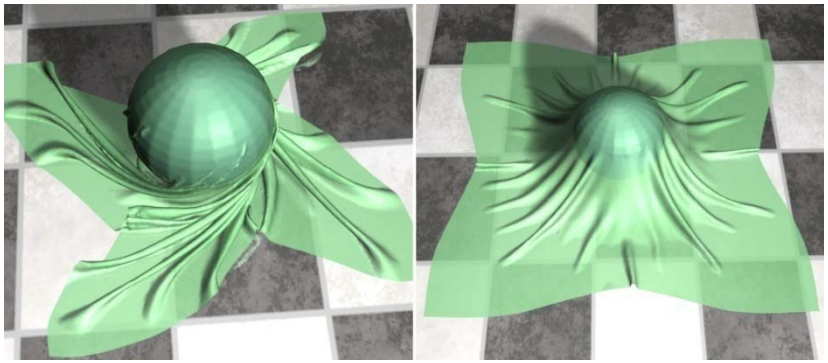
<Figure from C. Lauterbach >

<Figure from AION >

KAIST

Goals

- **Achieve interactive performance for exact collision detection between large-scale deformable models**
  - **E.g., deforming models consisting of tens or hundreds of thousand triangles**



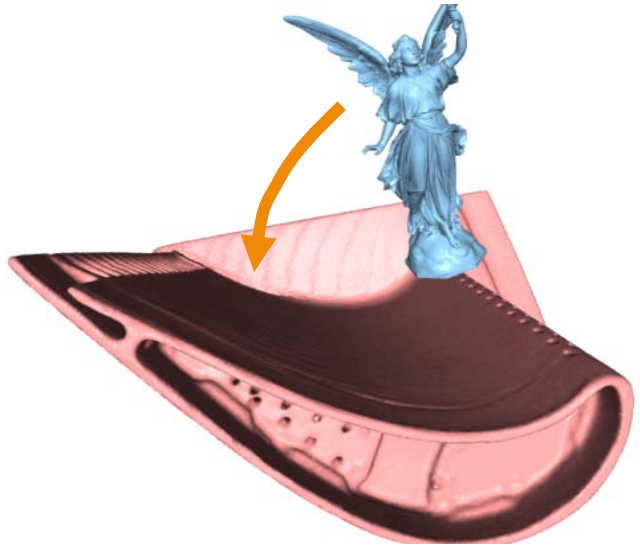
<Cloth-ball, 94K triangles>

<Breaking dragon, 252K triangles>

KAIST

Large-Scale Applications

- Entertainment (games/movies)
- E-heritage applications
- Virtual reality, etc



KAIST

Overview

- **HPCCD: Hybrid parallel continuous collision detection**
- **FASTCD: Fracturing-Aware Stable Collision Detection**
- **HCCMeshes: Hierarchical-Culling oriented Compact Meshes**

KAIST

Overview

- **HPCCD: Hybrid parallel continuous collision detection**
- **FASTCD: Fracturing-Aware Stable Collision Detection**
- **HCCMeshes: Hierarchical-Culling oriented Compact Meshes**

KAIST

Discrete vs. Continuous

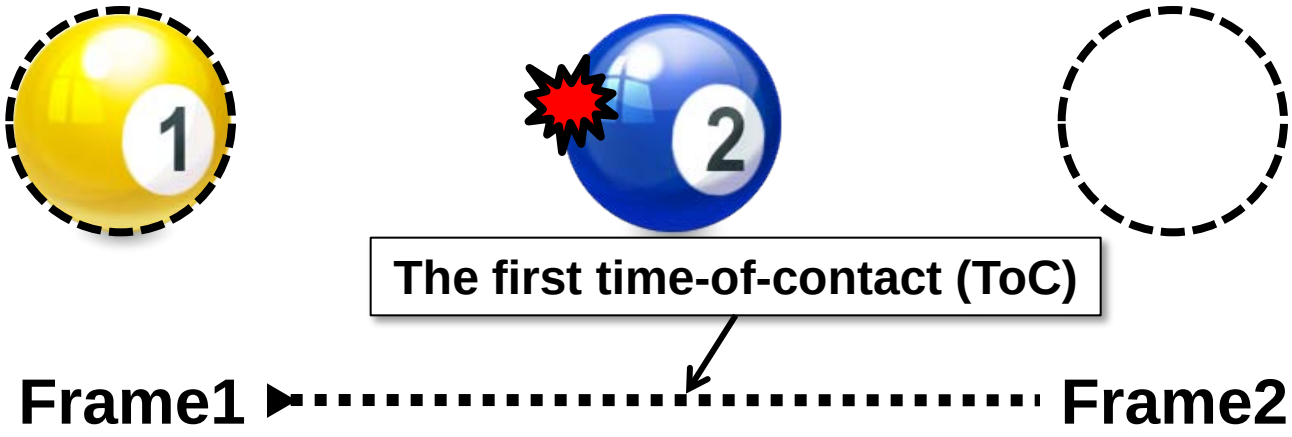
- **Discrete collision detection (DCD)**
  - **Detect collisions at each frame**
  - **Fast, but can miss collisions**

Miss collisions

Frame1 **Frame2**

Discrete vs. Continuous

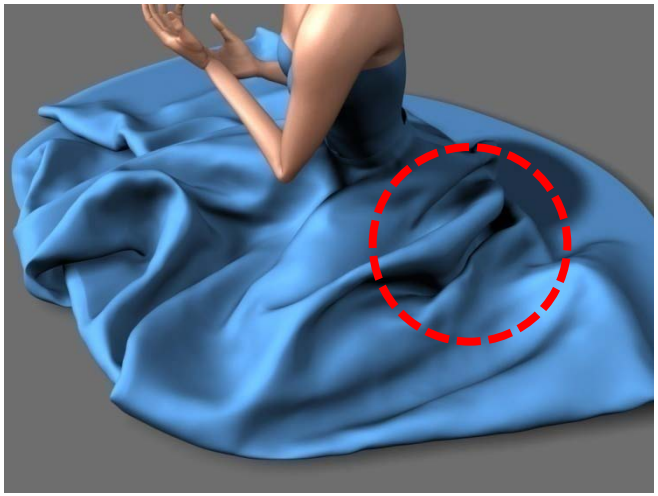
- Discrete collision detection (DCD)
- **Continuous collision detection (CCD)**
  - **Identify the first time-of-contact (ToC)**
  - **Accurate, but requires a long computation time**
  - **Not widely used in interactive applications**



Inter- and Self-Collisions

- **Inter-collisions**
  - **Collisions between two objects**

- **Self-collisions**
  - **Collisions between two regions of a deformable object**
  - **Takes a long computation time to detect**



From Govindaraju's paper

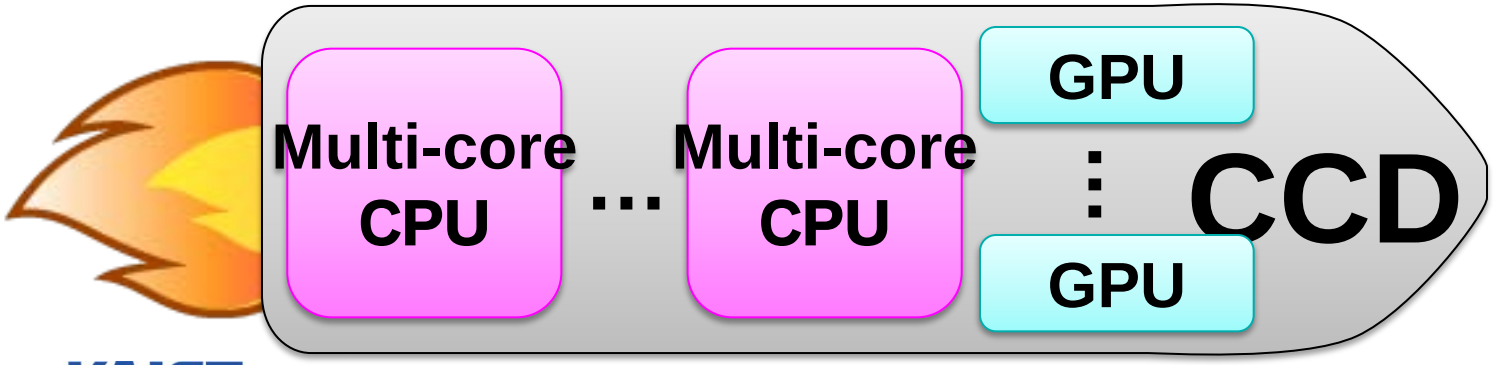
Parallel Computing Trends

- Many core architectures
  - Multi-core CPU architectures
  - GPU architectures
- Heterogeneous architectures
  - Intel's Larabee and AMD's Fusion
- Designing parallel algorithms is important to utilize these parallel architectures

Main Contributions

- **A novel, hybrid parallel CCD method**
  - **Utilize both multi-core CPUs and GPUs**
  - **No locking in the main loop of CD**
  - **GPU-based exact CD between two triangles**
- **High scalability & interactive performance**

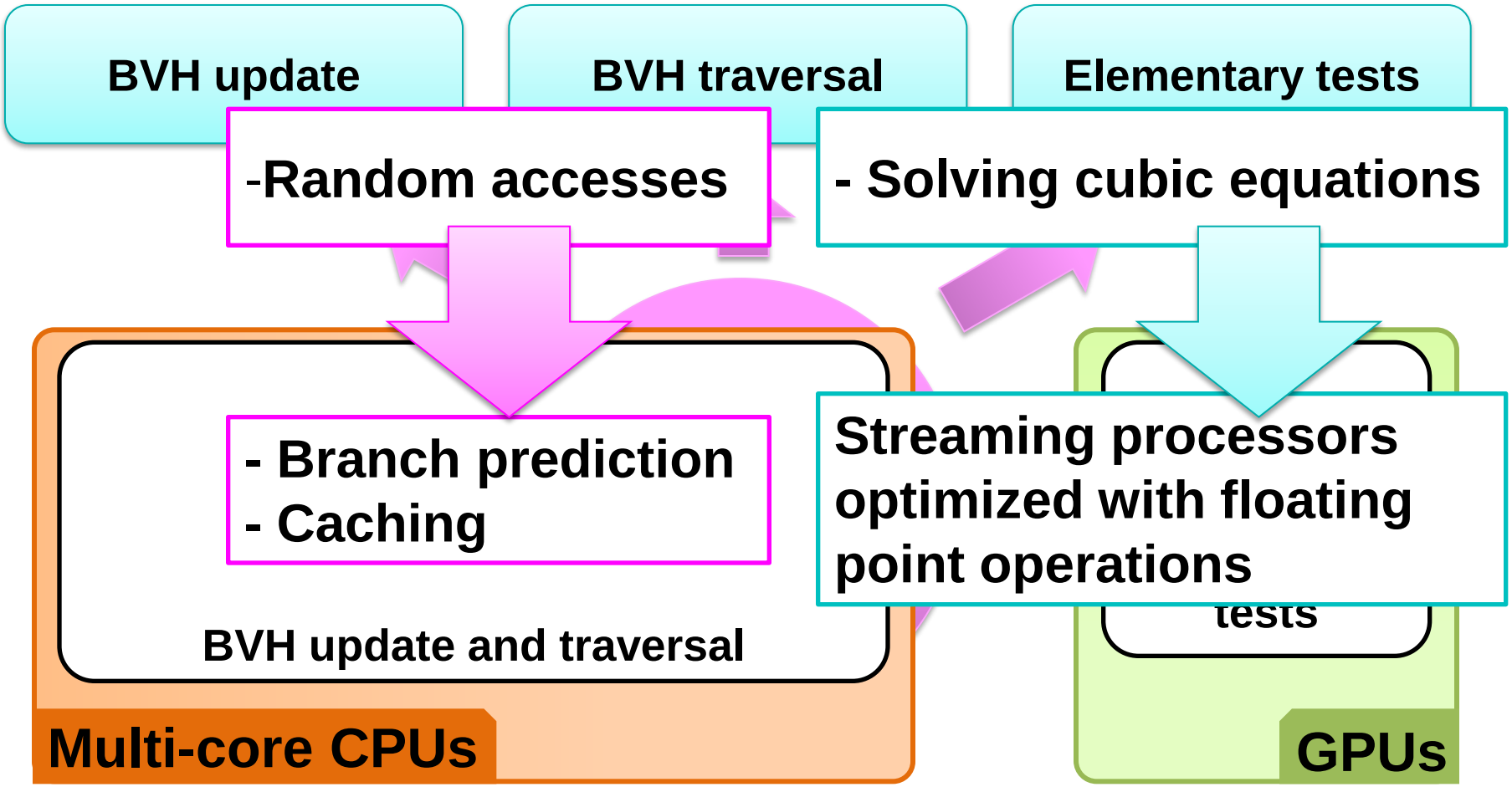
- Kim et al. PG 09
- Received a distinguished paper award



KAIST

Pacific Graphics 2009

Task Distribution



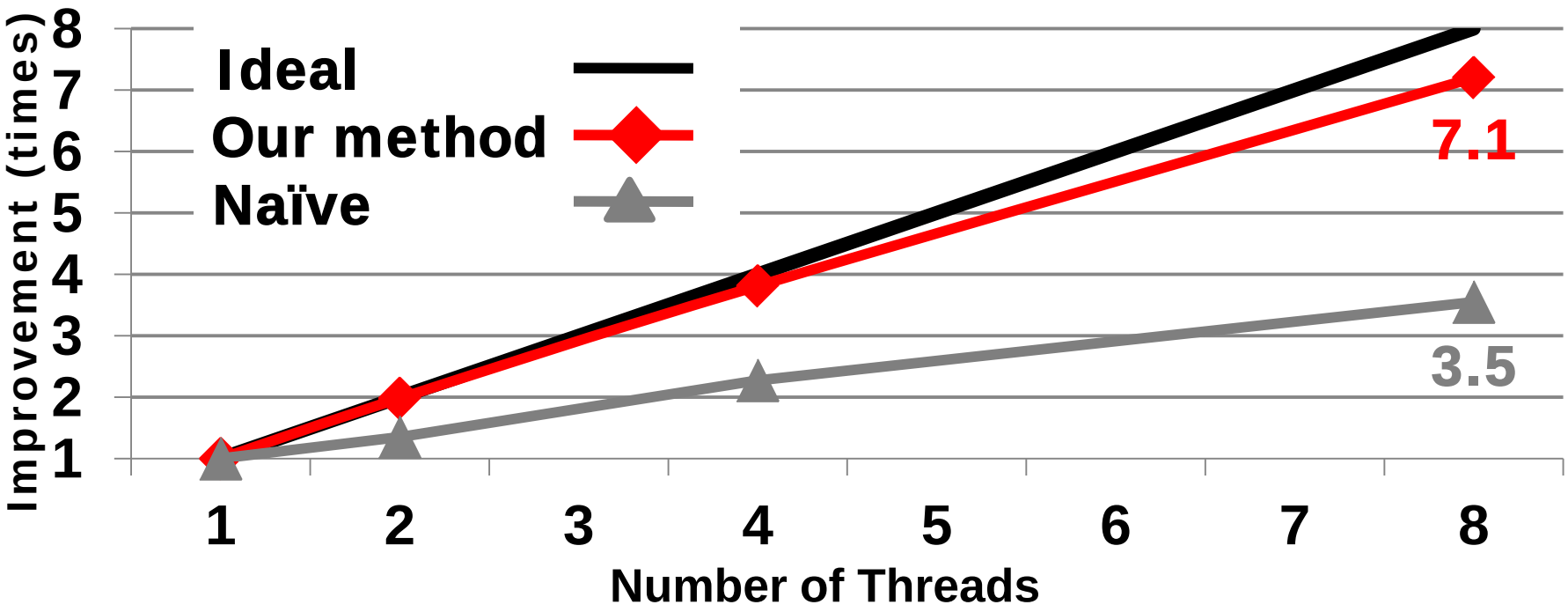
Testing Environment

- **Machine**
  - **One quad-core CPU (Intel i7 CPU, 3.2 GHz)**
  - **Two GPUs (Nvidia Geforce GTX285)**
- **Run eight CPU threads by using Intel's hyper threading technology**

KAIST

Results

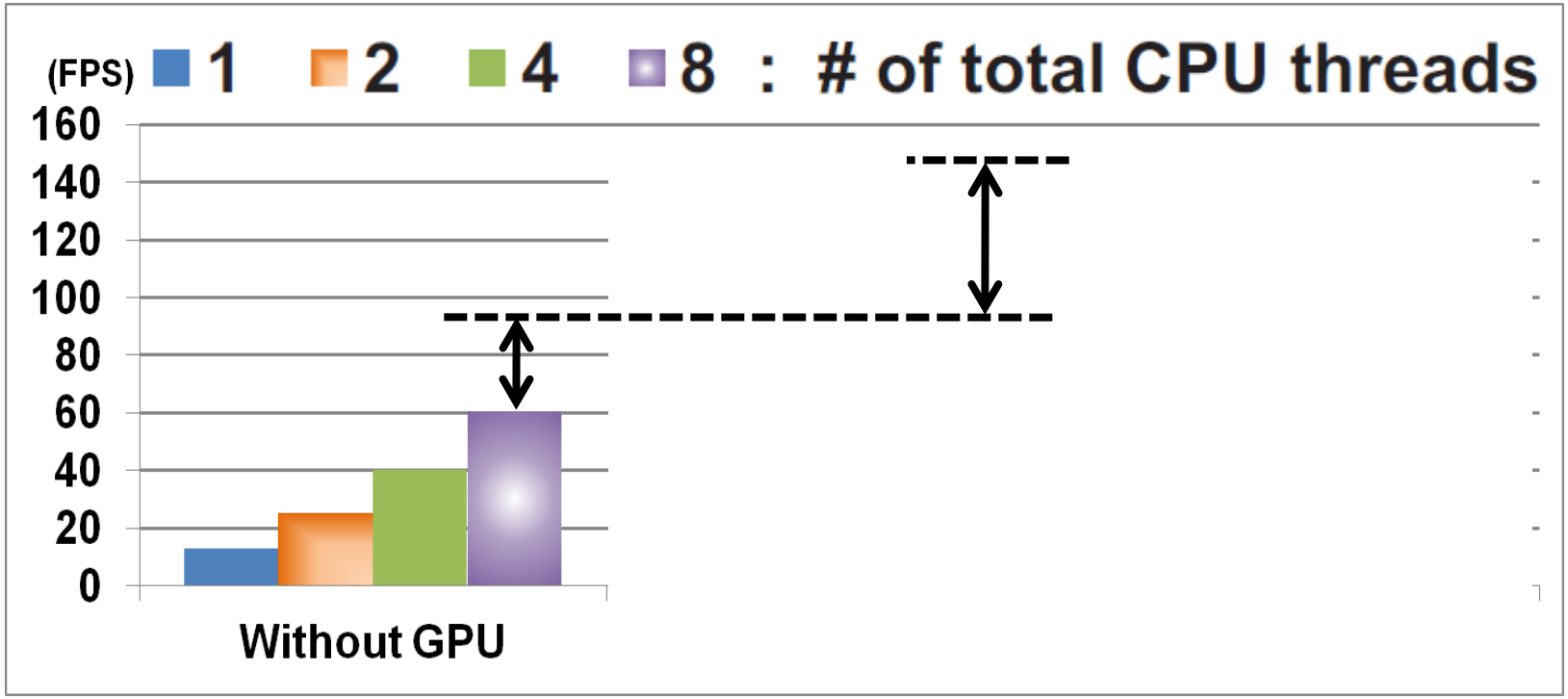
Results of Our CPU-based Parallel CCD



- Remove locking in the main loop of CD
- Employ efficient dynamic load-balancing based on inter-CD task units

KAIST

Results of HPCCD



- **As the number of GPUs is increased, we get higher performances**

KAIST

Limitation

- **Low scalability for small rigid models**

Summary

- A novel, hybrid parallel algorithm
  - Utilize both multi-core CPUs and GPUs
- High scalability

The implementation code will be available as

OpenCCD library

(http://sglab.kaist.ac.kr/OpenCCD)

  - Show 19-140 FPS for various deformable models consisting of tens or hundreds of thousand triangles

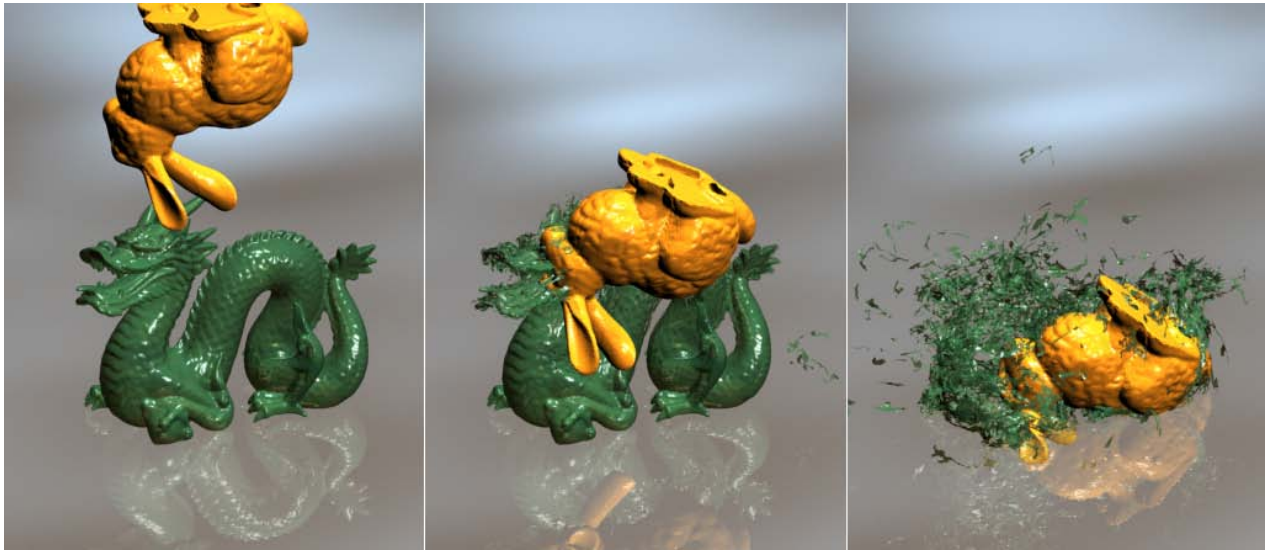
Overview

- **HPCCD: Hybrid parallel continuous collision detection**
- **FASTCD: Fracturing-Aware Stable Collision Detection**
- **HCCMeshes: Hierarchical-Culling oriented Compact Meshes**

Fracturing in Simulations

- More widely used in various applications to create more realistic interactions
- Presents significant challenges to CD
- Self-CD can take more expensive time with fracturing models
  - Fractured segments can be located in a near proximity

Examples (Video)



252K

142K

KAIST

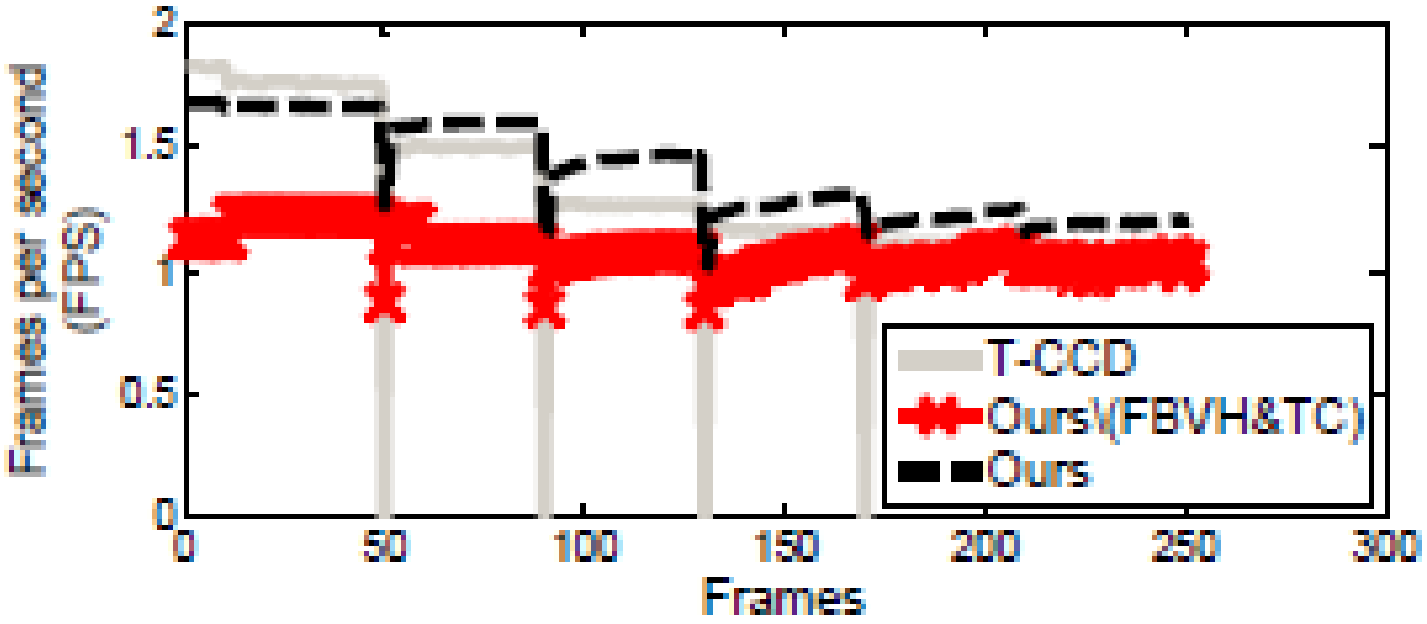
Research Challenges with Fracturing Models

- **Most prior works assume fixed topologies**
  - **Can take prohibitive times when models change their topologies**
- **Self-CD can take more expensive time with fracturing models**
  - **Fractured segments can be located in a near proximity**

Our Approach

- **Fracturing-aware stable CD**
  - **Incrementally update meshes and BVHs by utilizing topological changes of models**
  - **Design a simple self-CD culling method without much pre-computations**

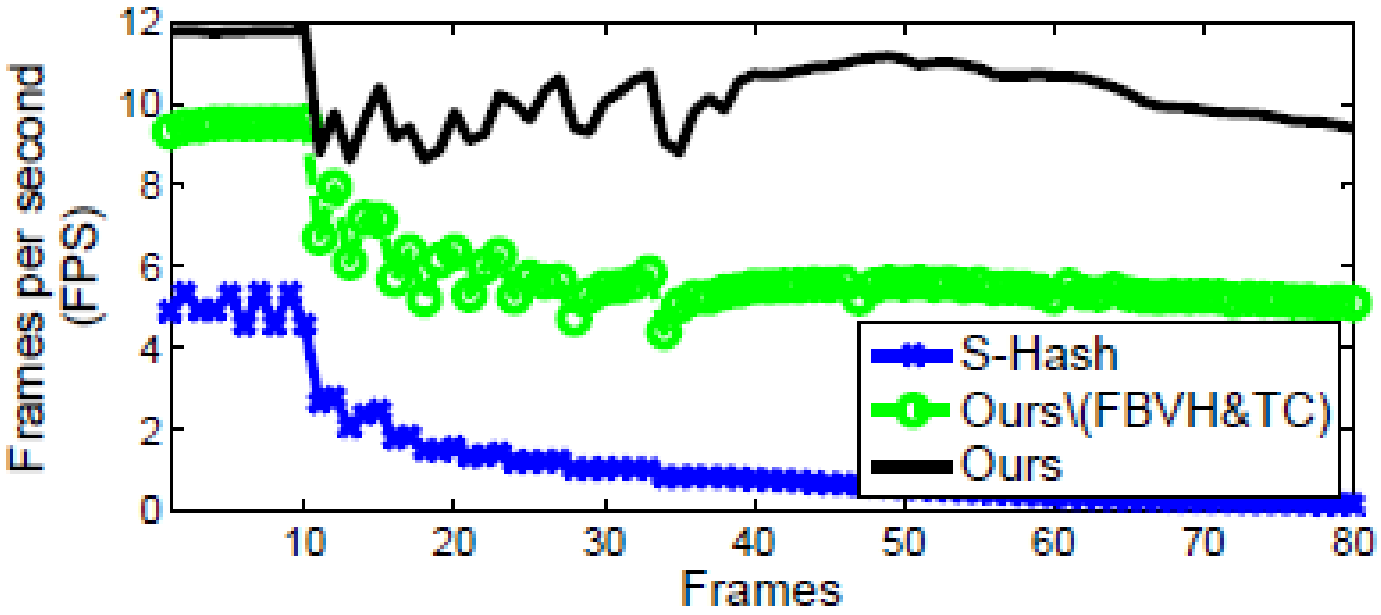
Results with the Exploding Dragon Model



T-CCD: Tang et al. [2008]

KAIST

Results with the Breaking-Wall Model



S-Hash: Optimized spatial hashing (Teschner et al. 03]

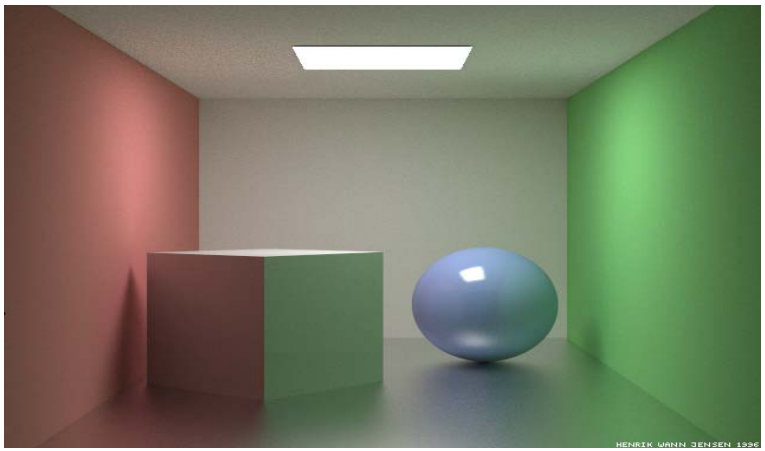
KAIST

Overview

- **HPCCD: Hybrid parallel continuous collision detection**
- **FASTCD: Fracturing-Aware Stable Collision Detection**
- **HCCMeshes: Hierarchical-Culling oriented Compact Meshes**

Hierarchical Culling and Traversal

- **Widely used in many search based algorithms**
  - **Various proximity queries**
  - **Ray tracing**



Path tracing

Photon mapping

[Images are from Jensen's homepage]

KAIST

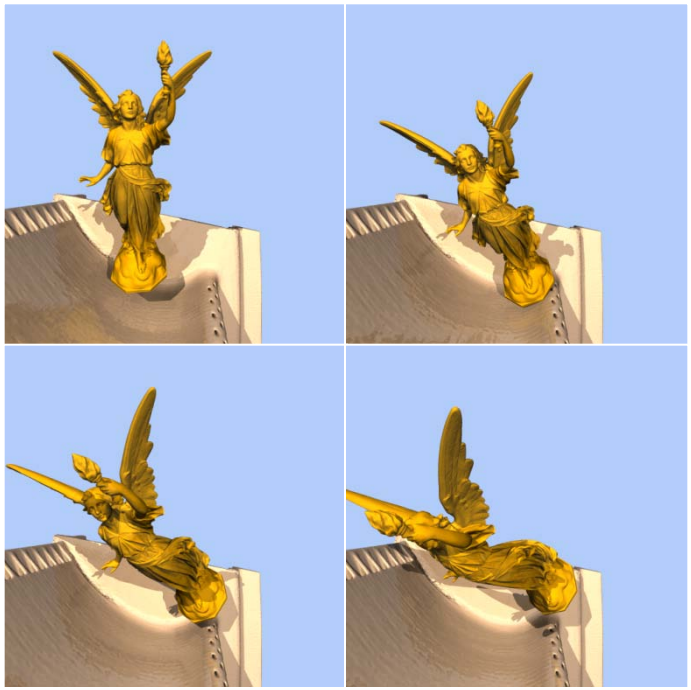
Ray Tracing – Massive Model

St. Matthew

- **128M triangles**
- **256M BVH nodes**
- **14GB data**
- **Size of memory < 4GB**
- **Rendering time : 40 min ↑**

KAIST

Collision Detection– Massive Model



- **30M triangles**
- **60M BVH nodes**
- **3.2GB data**

Lucy & CAD turbine

KAIST

Random-Accessible Compressed Data

- **Compression methods of meshes and hierarchies**
  - **Reduce the memory requirements**
  - **Supports random accesses on meshes and hierarchies**
  - **Can be useful to many different applications**

[Kim et al. Eurographics 2010 (to appear);
Kim et al., TVCG 09;
Yoon and Lindstrom, VIS 07]

→ 1st rank at ACM SIGGRAPH Student Competition

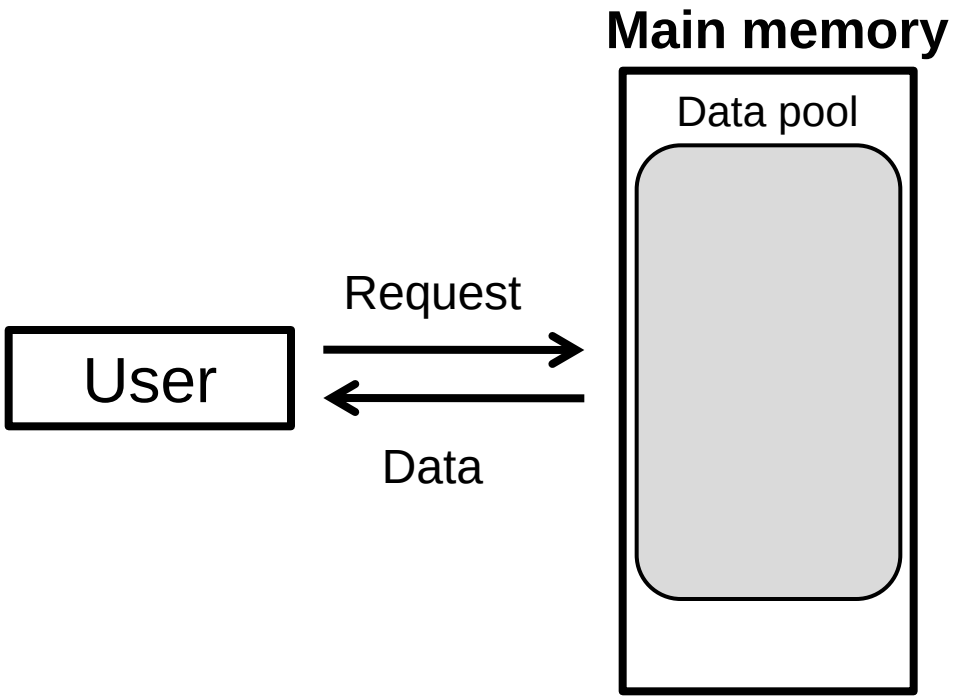
KAIST

Hierarchical-Culling oriented Compact Meshes (HCCMeshes)

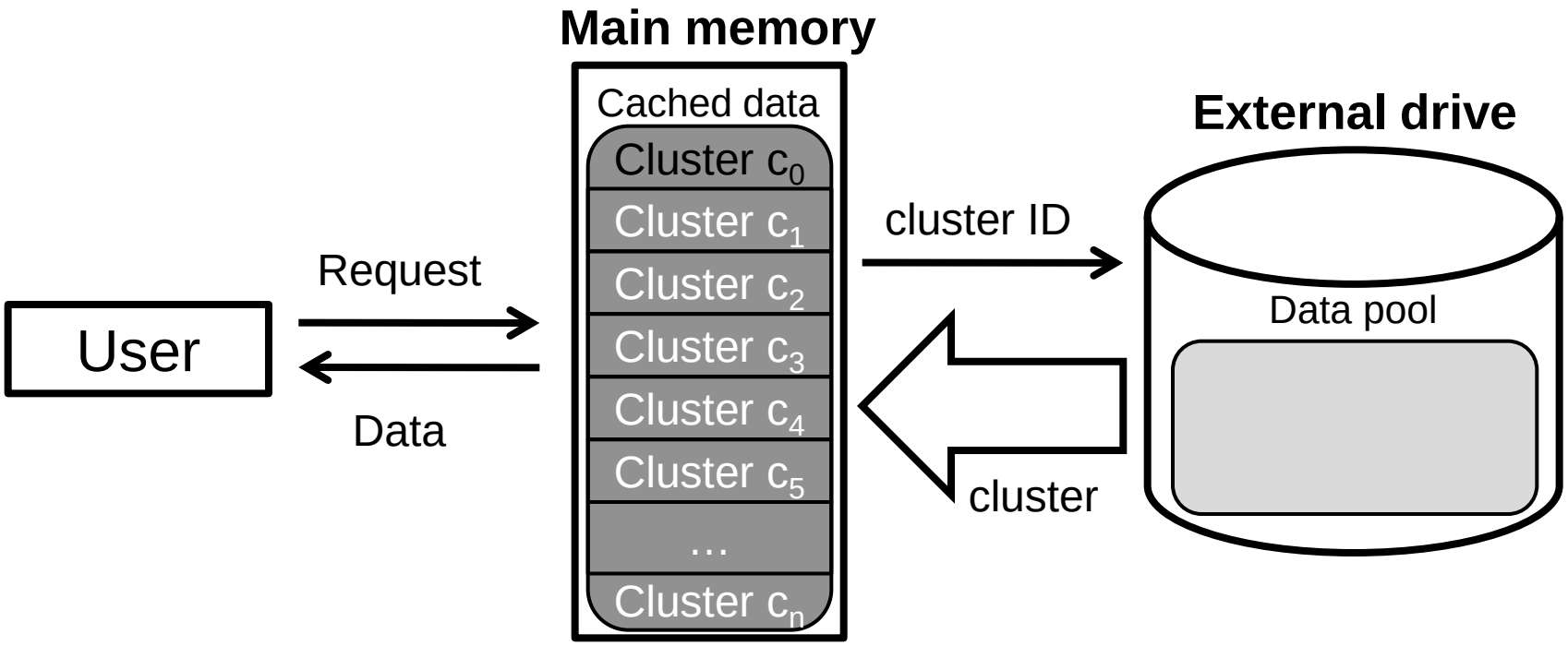
- **Consists of two parts:**
  - **i-HCCMeshes (in-core representation)**
  - **o-HCCMeshes (out-of-core representation)**

KAIST

Data Access Framework

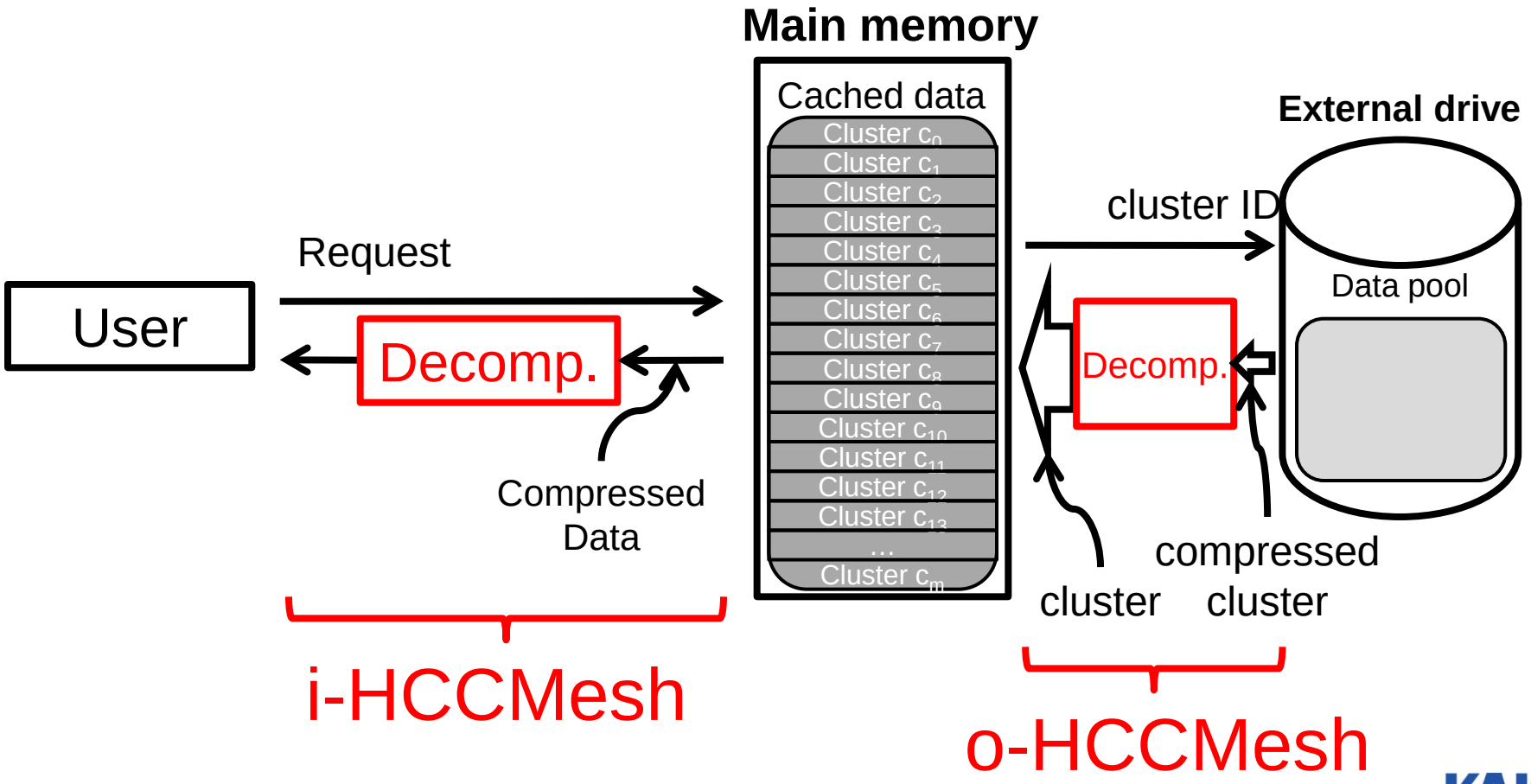


Data Access Framework - Out-of-Core Technique



HCCMeshes

Support hierarchical random access!



Main Benefits

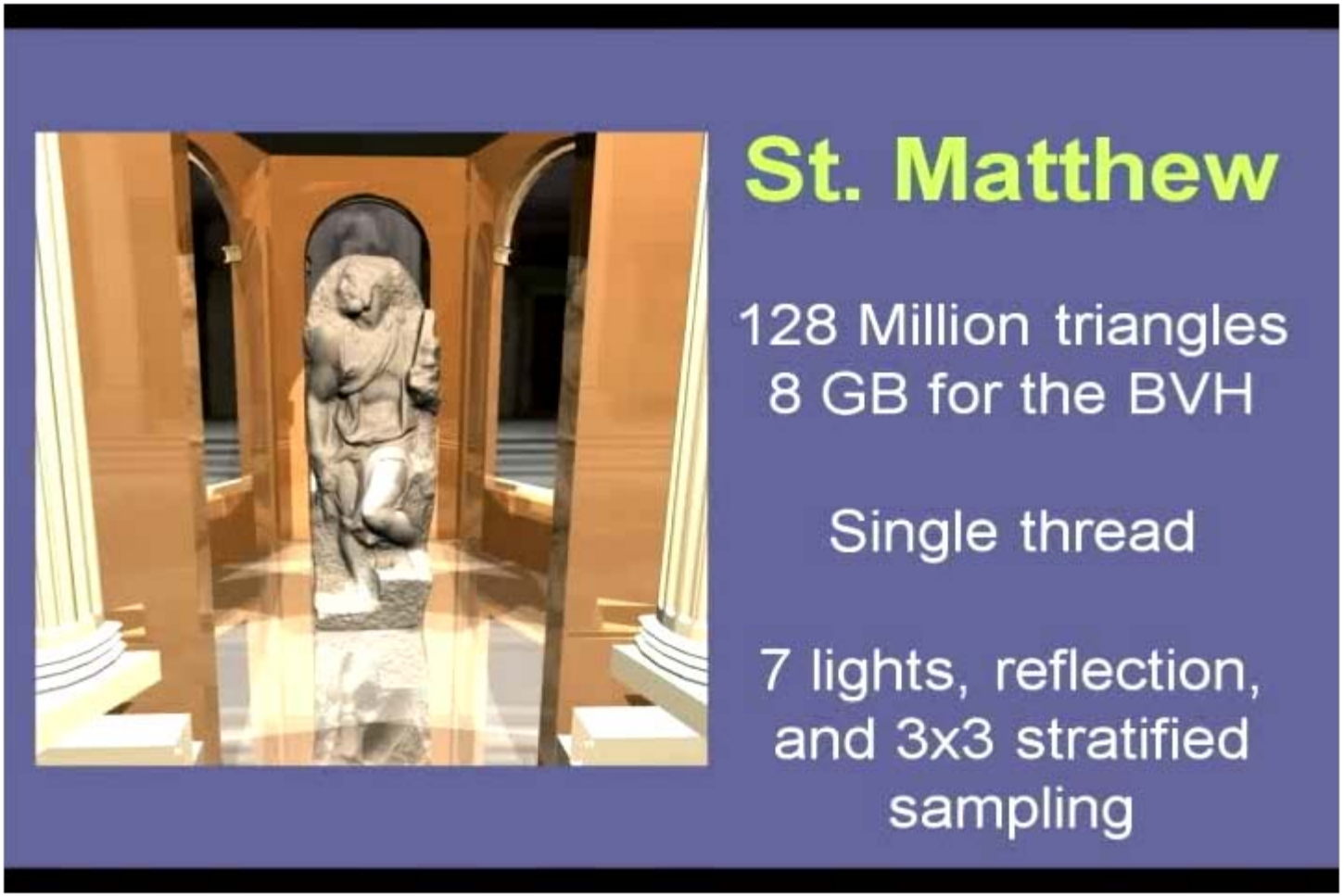
- **Use a lower memory space and working set size**
  - **o-HCCMeshes have 20:1 compression ratios**
  - **i-HCCMeshes have 6:1 compression ratios**
- **Improve runtime performance**

KAIST

Applications

- Whitted-style ray tracing
- LOD-based ray tracing
- Collision detection
- Photon mapping
- Non-photorealistic rendering

Results



KAIST

Conclusions

- **Discussed three different techniques for interactive CD among large-scale deforming models**
  - **HPCCD: Hybrid parallel continuous collision detection**
  - **FASTCD: Fracturing-Aware Stable Collision Detection**
  - **HCCMeshes: Hierarchical-Culling oriented Compact Meshes**

KAIST

Acknowledgements

- **Research collaborators**
  - **DukSu Kim, JaePil Heo, TaeJoon Kim, Pio Claudio, BooChang Moon, YongYoung Byun, SeungYong Lee, YongJin Kim, JaeHyuk Heo, John Kim**

- **Funding sources**
  - **Ministry of Knowledge Economy**
  - **Microsoft Research Asia**
  - **KAIST seed grant**
  - **Samsung**
  - **Korea Research Foundation**

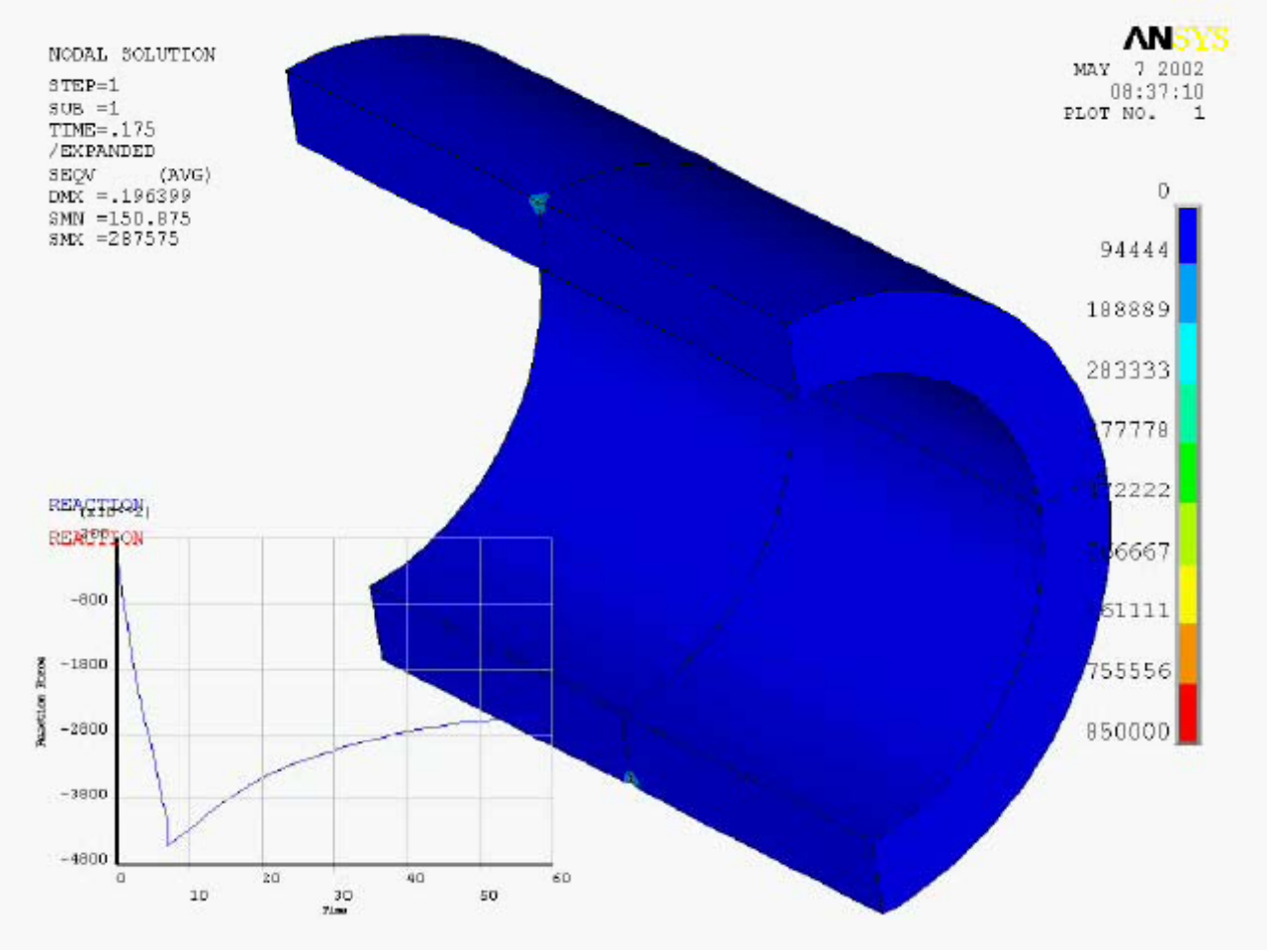
KAIST

Real-Time Simulation of Flexible Bodies

CG Lab.

Kwangwoon University

Deformation in Mechanical Engineering



Character Skinning in CG

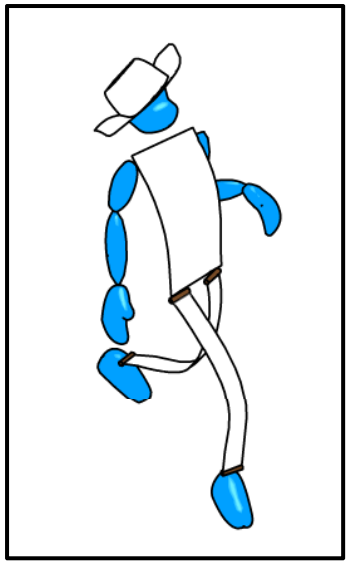
Hair Simulation in CG

Surgery Simulation

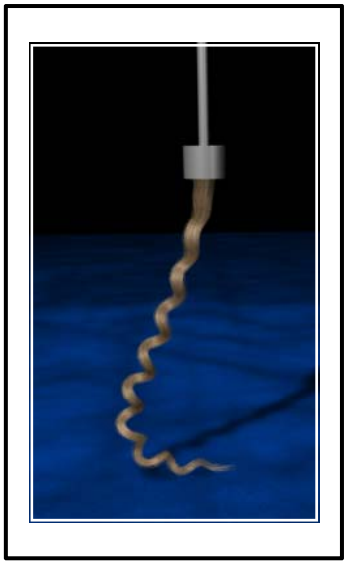
Overview

- Elastodynamic deformation of solids, shells, and rods

3D Solids



3D Thin Shells



3D Thin Rods

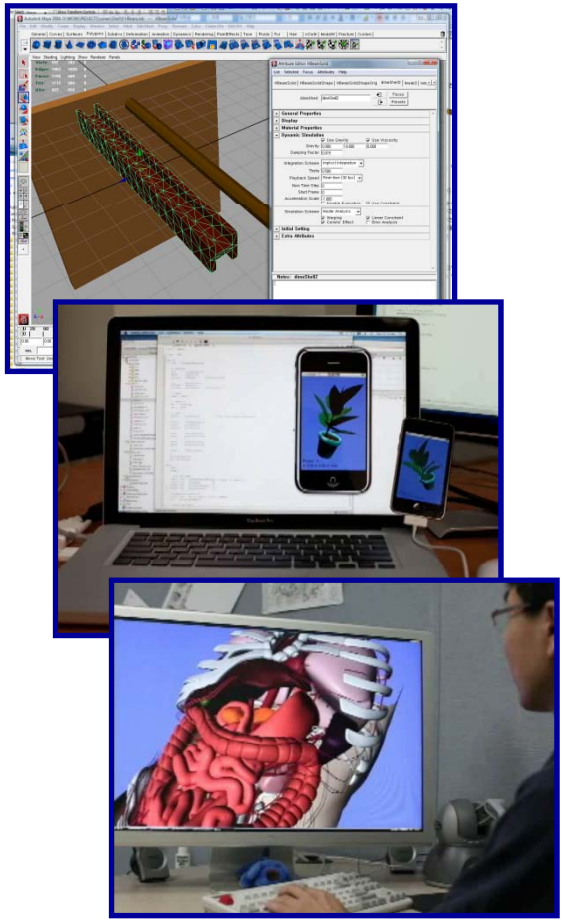
Overview

- **Authoring in the Maya platform**
  - Geometry
  - Material constants
  - Precomputation

  Exporting

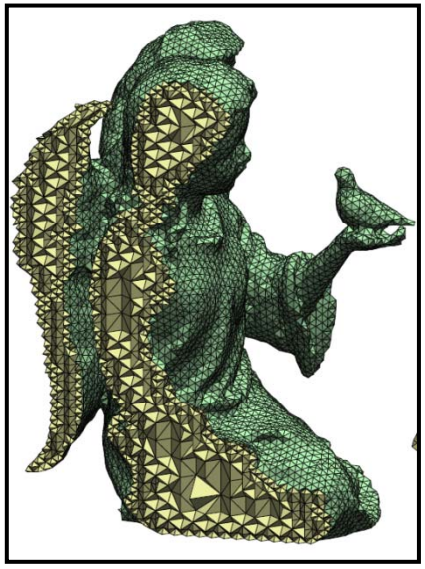
  - Runtime simulation

- **Simulation in multi-platforms**
  - iPod Touch
  - Haptic device



Future Research Directions

- **Extension of elastodynamic deformation**



Meshing

Crack/Fracture

Viscosity/Plasticity

Ongoing, Scheduled Work

Dynamic Deformation

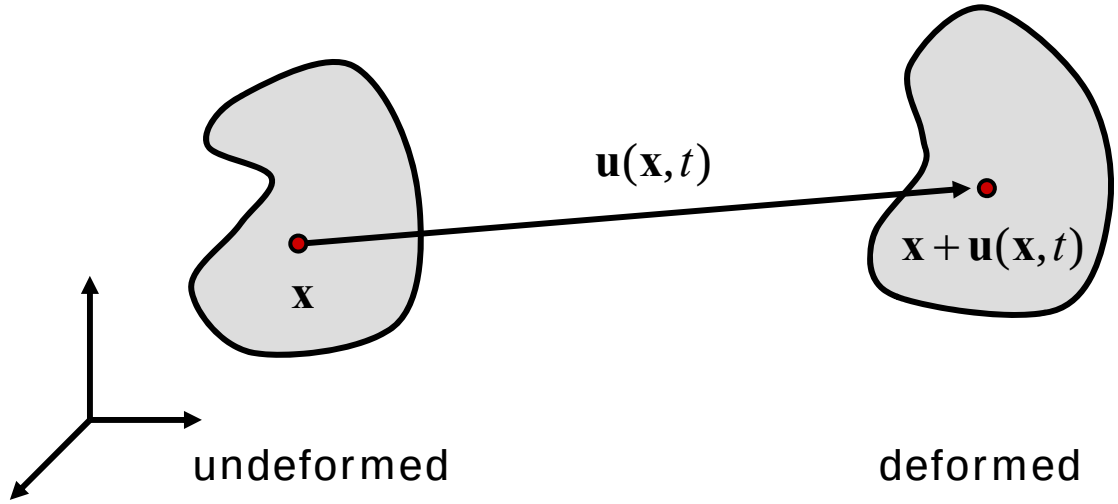
- **Governing equations: Euler-Lagrange equations**

kinetic energy

elastic potential energy

$$\frac{d}{dt}\left(\frac{\partial T(\mathbf{u})}{\partial \dot{\mathbf{u}}}\right)+\frac{\partial U(\mathbf{u})}{\partial \mathbf{u}}=\mathbf{f}_{ext}$$

external force



Finite Element Discretization

- **Governing equations: Euler-Lagrange equations**

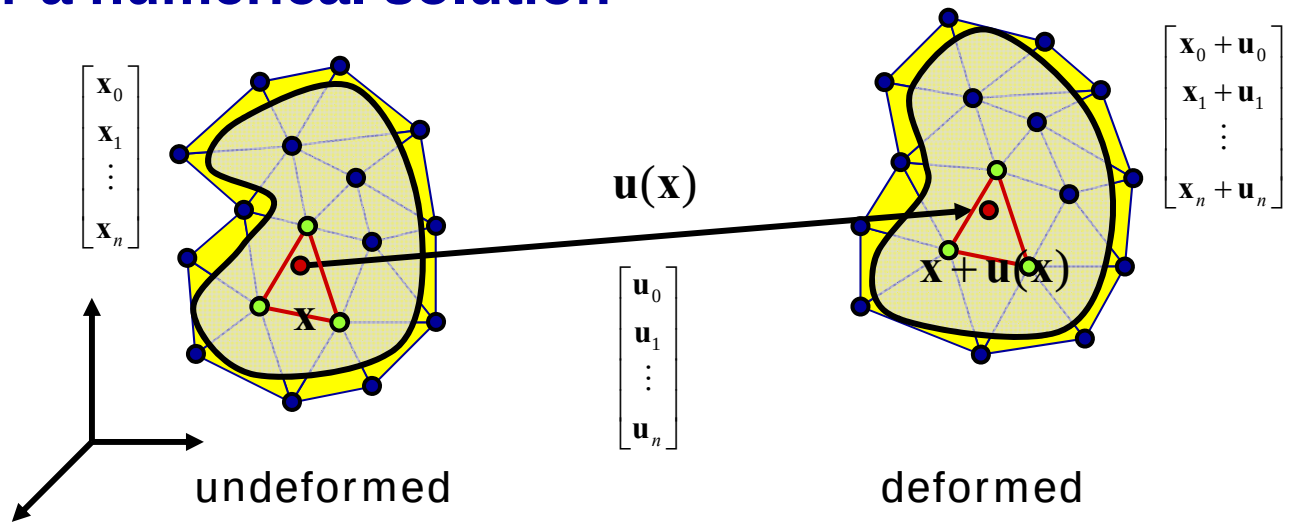
kinetic energy

elastic potential energy

$$\frac{d}{dt}\left(\frac{\partial T(\mathbf{u})}{\partial \dot{\mathbf{u}}}\right)+\frac{\partial U(\mathbf{u})}{\partial \mathbf{u}}=\mathbf{f}_{ext}$$

external force

- **For a numerical solution**



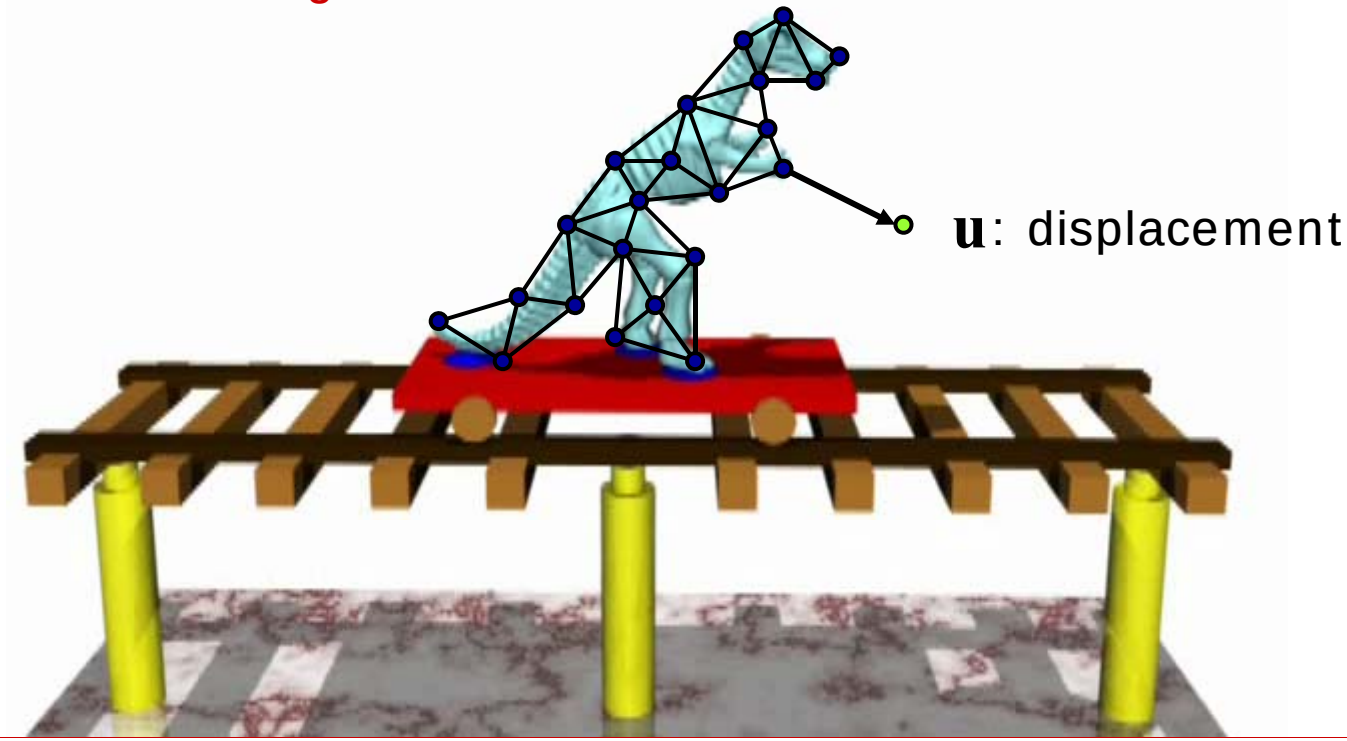
Real-Time Deformation of 3D Solids

- **Elastodynamic equation for FEM**

$3n \times 3n$ matrix, $3n \times 1$ vector

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}(\mathbf{u}) = \mathbf{F} \quad \Rightarrow \quad \mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F}$$

Time-consuming!!! Linearization Not real-time!!!



Real-Time Deformation of 3D Solids

- **Modal analysis in *local coordinate frames***

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}(\mathbf{u}) = \mathbf{F} \quad \Rightarrow \quad \mathbf{M}\ddot{\mathbf{u}}^L + \mathbf{C}\dot{\mathbf{u}}^L + \mathbf{K}\mathbf{u}^L = \mathbf{R}^{\mathrm{T}}\mathbf{F}$$

Linearization

G. Eigenvalue problem

$$\mathbf{M}\boldsymbol{\Phi}_i = \lambda_i \mathbf{K}\boldsymbol{\Phi}_i$$

→ Modal basis

$$\mathbf{u}^L(t) = \boldsymbol{\Phi}\mathbf{q}(t)$$

$$\mathbf{u}^k = \int_0^{t^k} \mathbf{R}(t)\Phi\dot{\mathbf{q}}(t)dt \quad \Leftarrow \quad \mathbf{M}_\mathbf{q}\ddot{\mathbf{q}} + \mathbf{C}_\mathbf{q}\dot{\mathbf{q}} + \mathbf{K}_\mathbf{q}\mathbf{q} = \mathbf{Q}$$

Modal warping

Real-time performance without linearization artifacts!!!

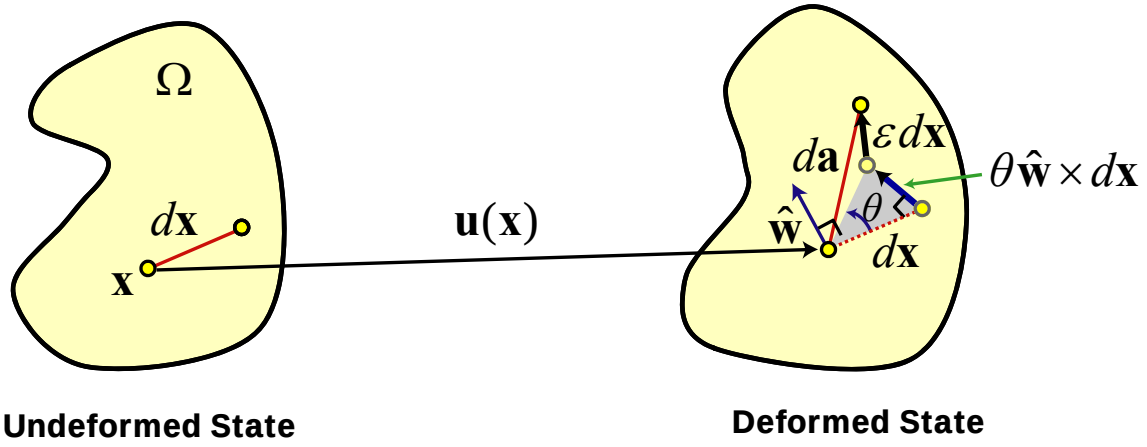
Modal Rotation for Solids

- **Infinitesimal deformation = rotation + strain**
  - Element-wise rotation vector

$$\mathbf{w}_e(\mathbf{x}) = \tfrac{1}{2}\nabla \times \mathbf{u}(\mathbf{x}) = \mathbf{W}_e \mathbf{u}_e$$

  - Globally-assembled rotation vector

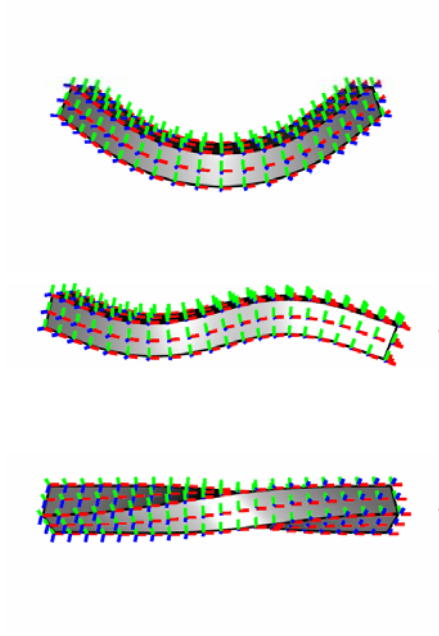
$$\mathbf{w}(t) = \mathbf{W}\mathbf{u}(t) = \{\mathbf{W}\Phi\}\mathbf{q}(t) = \Psi\mathbf{q}(t)$$

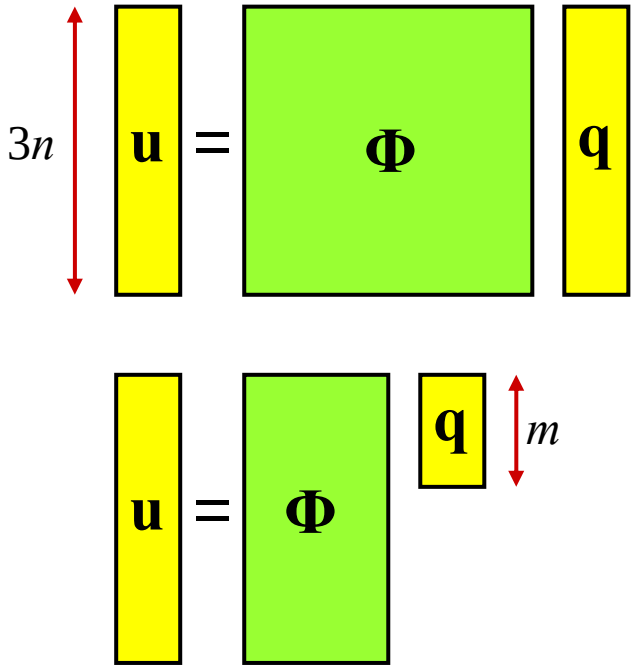


Modal Truncation

- **Exploit dominant low frequency mode shapes only!**
  - Higher modes heavily damped and die out fast.

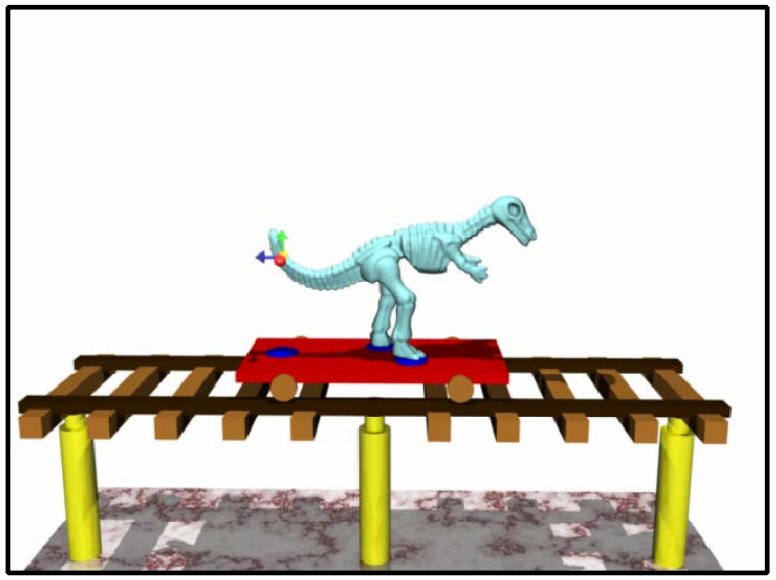
$$\mathbf{u}(t) = \tilde{\mathbf{R}}\boldsymbol{\Phi}\,\mathbf{q}(t) = \tilde{\mathbf{R}}\boldsymbol{\Phi}_i\, q_i(t)$$



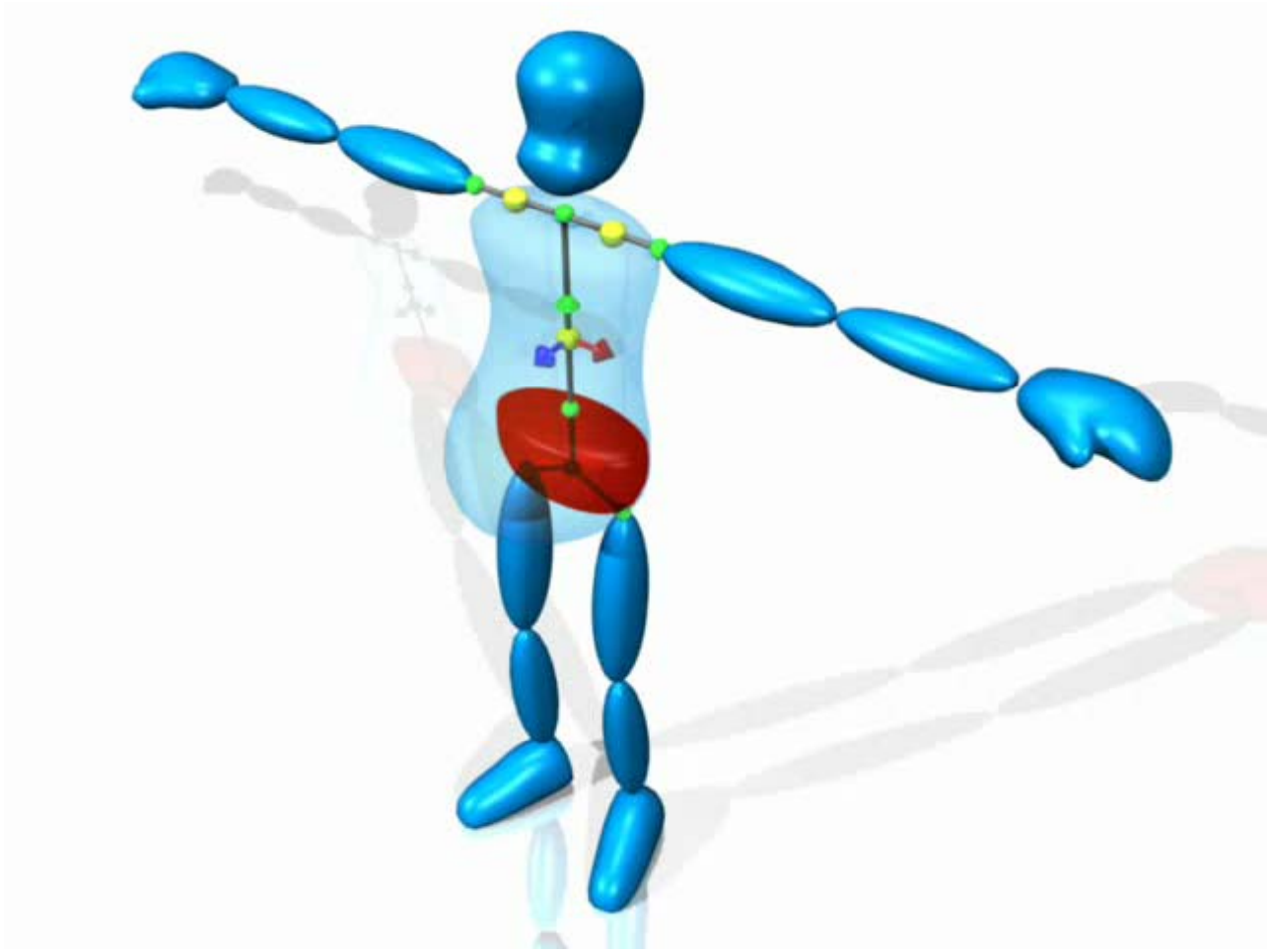


Deformation using GPU

- **Programmable graphics H/W for large models**
  - Written in NVIDIA Cg + DirectX
  - Small extra computation in vertex program: 11 lines, 47 inst.

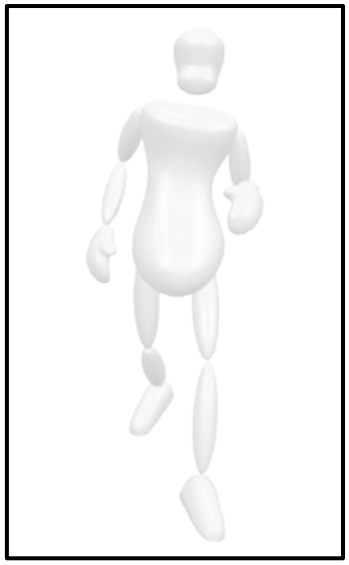


Passive Deformation as 2ndary Motion

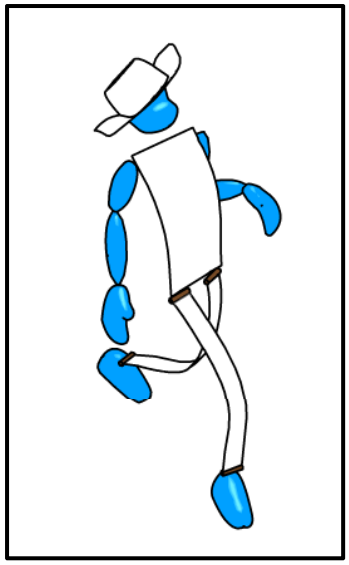


Real-Time Deformation of 3D Thin Shells

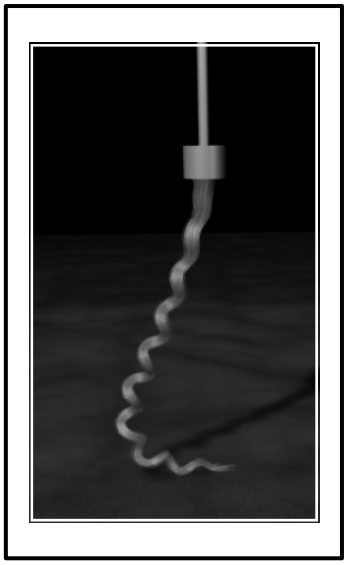
- Elastodynamic deformation in 3D



3D Solids



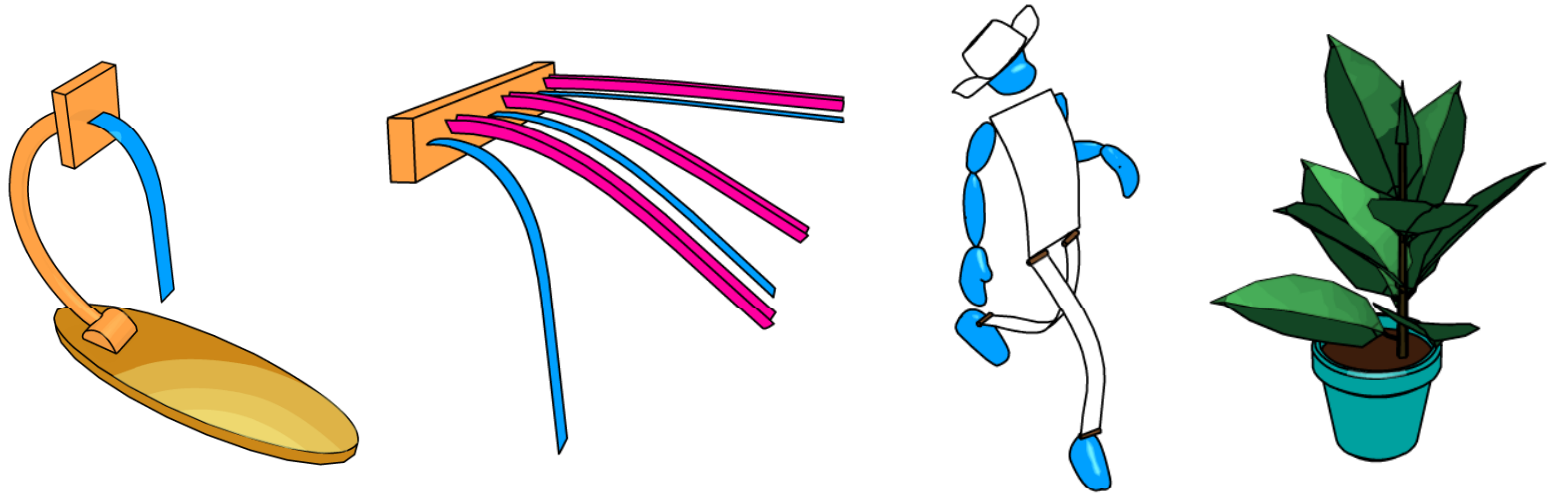
3D Thin Shells



3D Thin Rods

Thin Shell Structures

- Thin shells are almost two-dimensional structures.



Dynamics of Thin Shell Structures

- **Elastic potential energy of a thin shell**

$$E = \underbrace{(k_A E_A + k_L E_L)}_{\text{Membrane energy}} + \underbrace{k_B E_B}_{\text{Flexural energy}}$$

- **Elastic force due to the elastic potential energy**

$$\mathbf{K}(\mathbf{u}) = \frac{\partial E(\mathbf{u})}{\partial \mathbf{u}}$$

Modal Warping for 3D Thin Shells

- **Modal analysis in *local coordinate frames***

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}(\mathbf{u}) = \mathbf{F} \quad \Rightarrow \quad \mathbf{M}\ddot{\mathbf{u}}^{L} + \mathbf{C}\dot{\mathbf{u}}^{L} + \mathbf{K}\mathbf{u}^{L} = \mathbf{R}^{\mathrm{T}}\mathbf{F}$$

Linearization in ***LOCAL*** coordinate frames

G. Eigenvalue problem $\mathbf{M}\boldsymbol{\Phi}_i = \lambda_i \mathbf{K}\boldsymbol{\Phi}_i$ → Modal basis $\mathbf{u}^{L}(t) = \boldsymbol{\Phi}\mathbf{q}(t)$ ↓

$$\mathbf{u}^{k} = \int_{0}^{t^{k}} \mathbf{R}(t)\Phi\dot{\mathbf{q}}(t)dt \quad \Leftarrow \quad \mathbf{M}_{\mathbf{q}}\ddot{\mathbf{q}} + \mathbf{C}_{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{K}_{\mathbf{q}}\mathbf{q} = \mathbf{Q}$$

Modal warping

Real-time performance without linearization artifacts!!!

Modal Rotation for Shells

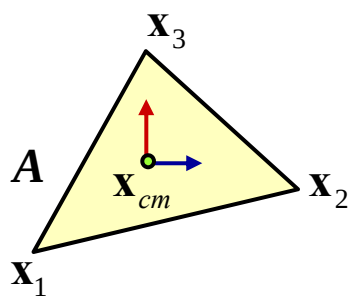
- **Rotation of a triangle near the rest state**

$$\arg\min_{\omega_A} \sum_{i=1}^{3} \| \mathbf{p}_i + \omega_A \times \mathbf{p}_i - \mathbf{q}_i \|^2$$

$$\mathbf{p}_i = \mathbf{x}_i - \mathbf{x}_{cm}, \quad \mathbf{q}_i = \mathbf{a}_i - \mathbf{a}_{cm}$$

  - Element-wise rotation vector

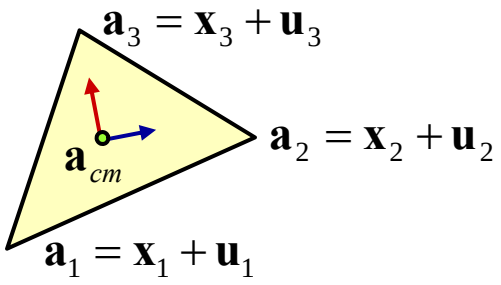
$$\omega_A(\mathbf{u}_A) = -\left(\left[\sum_{i=1}^{3} \mathbf{p}_i \times \mathbf{p}_i \times \right]^{-1} \left[\mathbf{p}_1 \times | \mathbf{p}_2 \times | \mathbf{p}_3 \times \right] \right) \mathbf{u}_A$$



Undeformed state

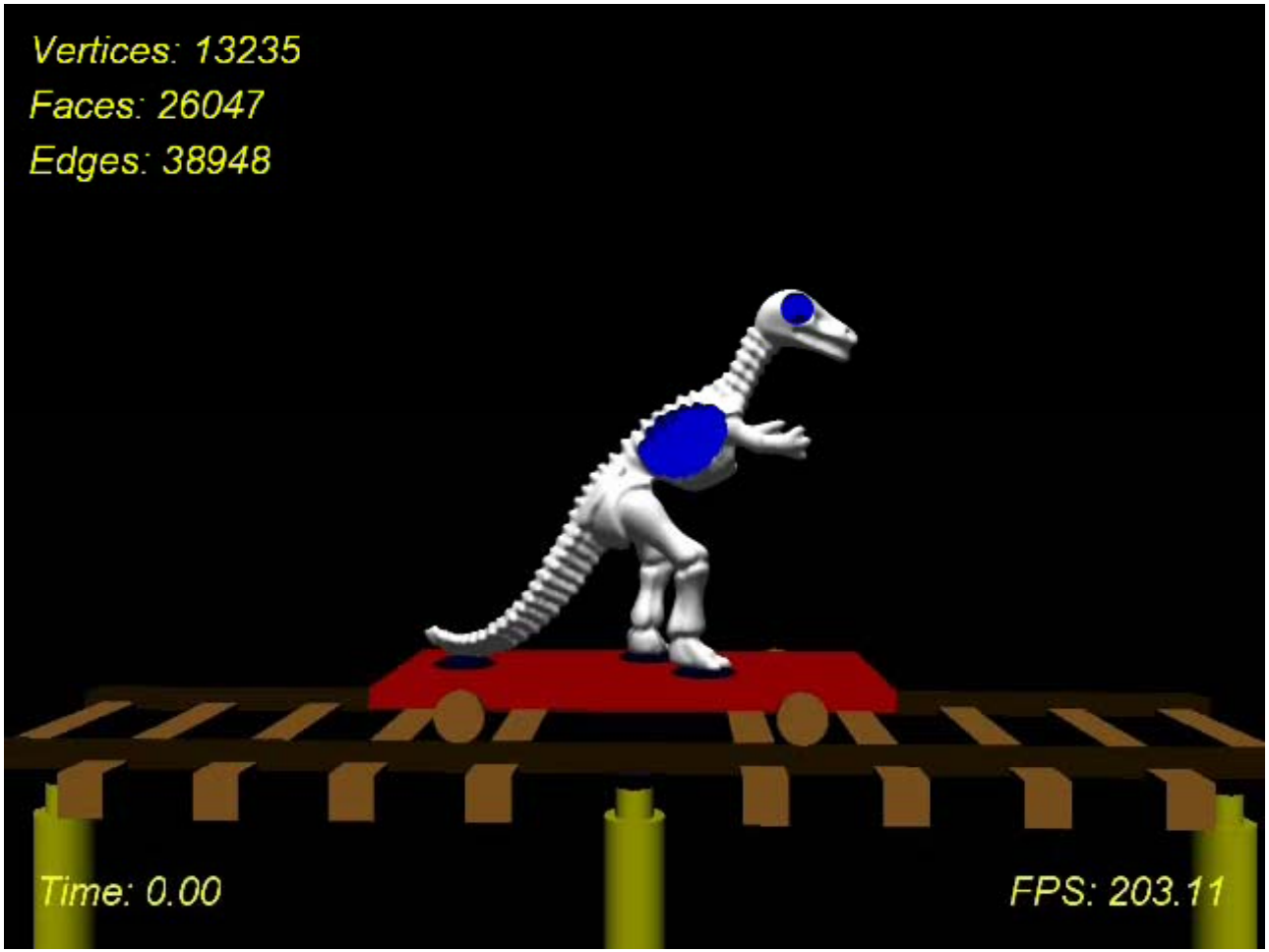
$\omega_A(\mathbf{u}_A)$

Rotation vector of a triangle A



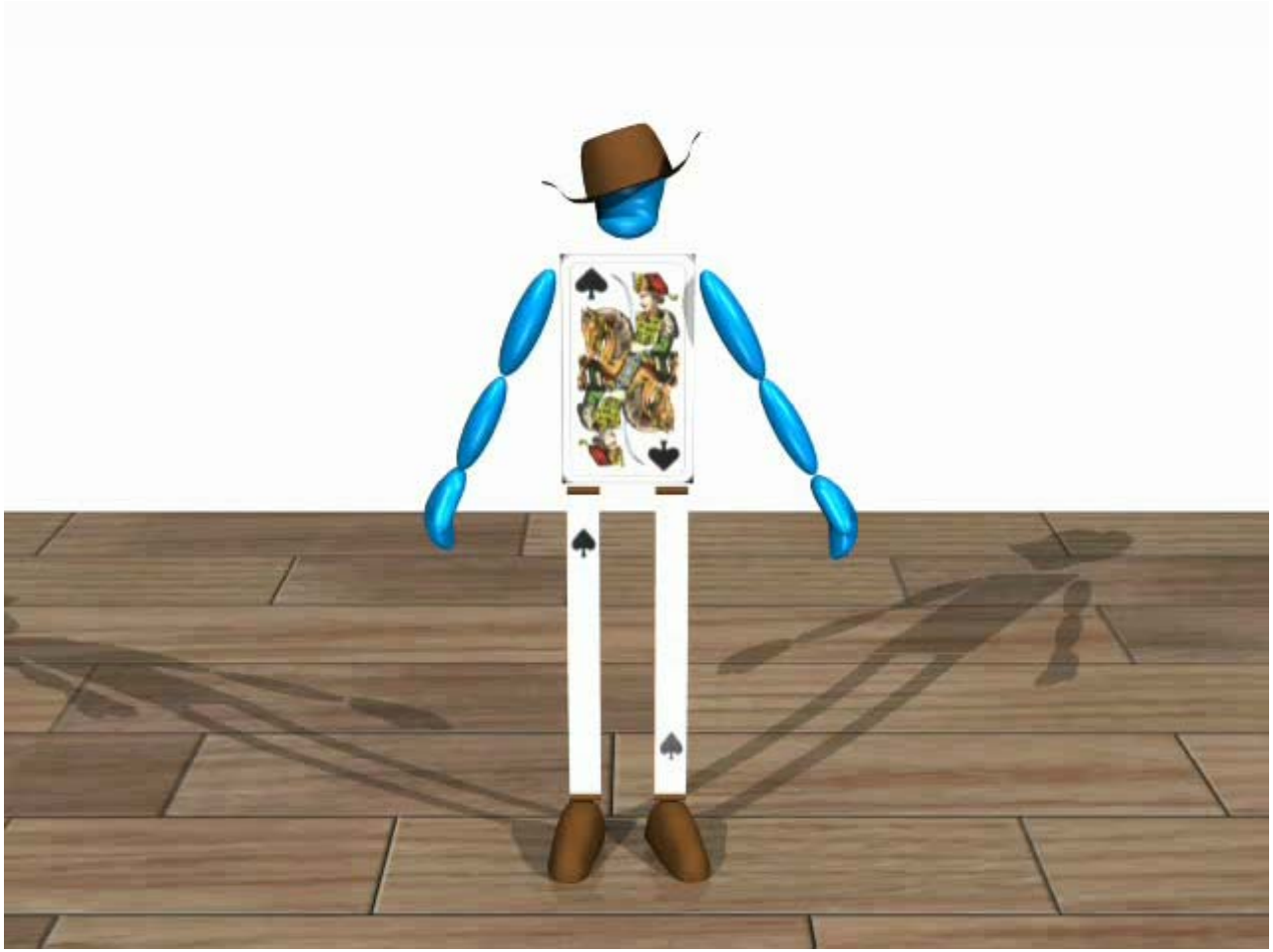
Deformed state

Deformation using GPU



Programmable graphics hardware for large mesh

Passive Deformation as 2ndary Motion



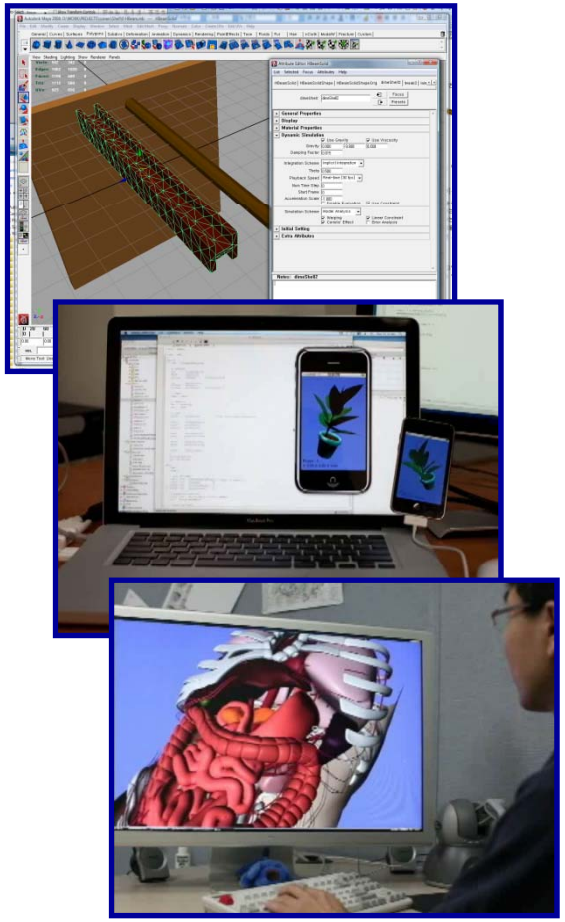
Framework 4 Flexible Body Simulation

- **Authoring in the Maya platform**
  - Geometry
  - Material constants
  - Precomputation

  Exporting

  - Runtime simulation

- **Simulation in multi-platforms**
  - iPod Touch
  - Haptic device



Haptic Platform

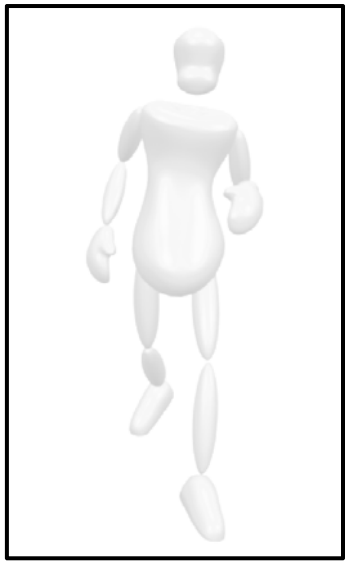
- **Features**
  - Phantom Omni + GPU

iPod Touch Platform

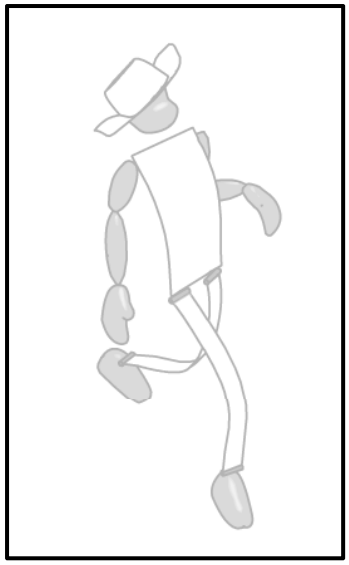
- **Features**
  - 3-axis linear accelerometers for excitation forces
  - Multi-touch capabilities for drawing/simulation modes

Real-Time Simulation of 3D Thin Rods

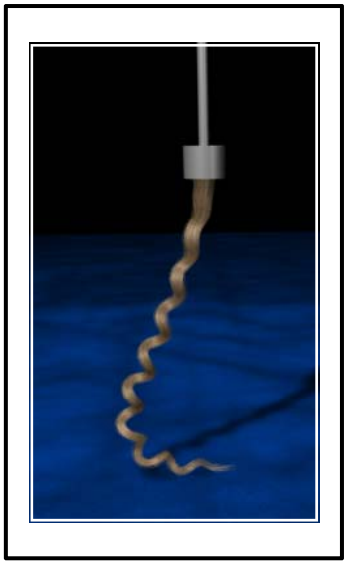
- **Elastodynamic deformation in 3D**



3D Solids



3D Thin Shells



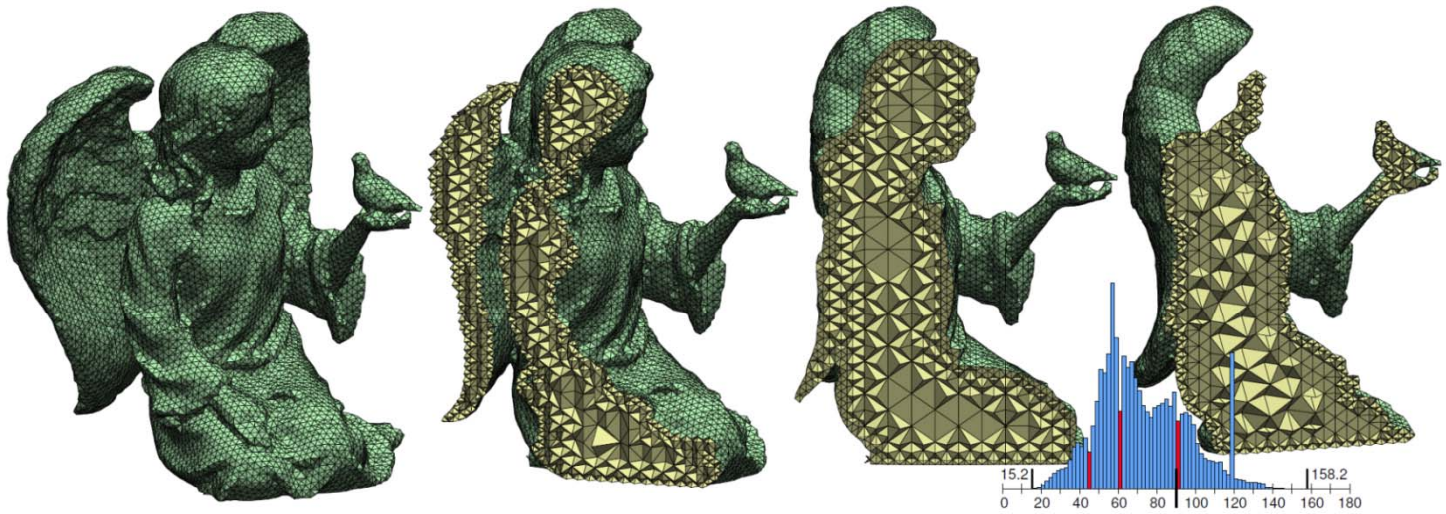
3D Thin Rods

Next Step

- Hair simulation

Next Step

- **Meshing for flexible body simulation**



Isosurface stuffing by Labelle and Shewchuk

Next Step

- Corotational FEM for crack/fracture simulation

Star Wars: The Force Unleashed by Parker and O'Brien

Next Step

- Elasticity, viscosity, and plasticity

Fast viscoelastic behavior by Wojtan and Turk

Questions?

Virtual Foosball:

A Tangible Implementation of the Virtual Physics Library

everyware

CREATIVE COMPUTING GROUP

Yunsil Heo
NML, Inc.

Hyunwoo Bang
New Media Lab., Seoul Nat'l Univ.

Jinwook Kim
Imaging Media Research Center,
KIST

Young J. Kim
Computer Graphics Lab., Ewha Womans
Univ.

Virtual Foosball

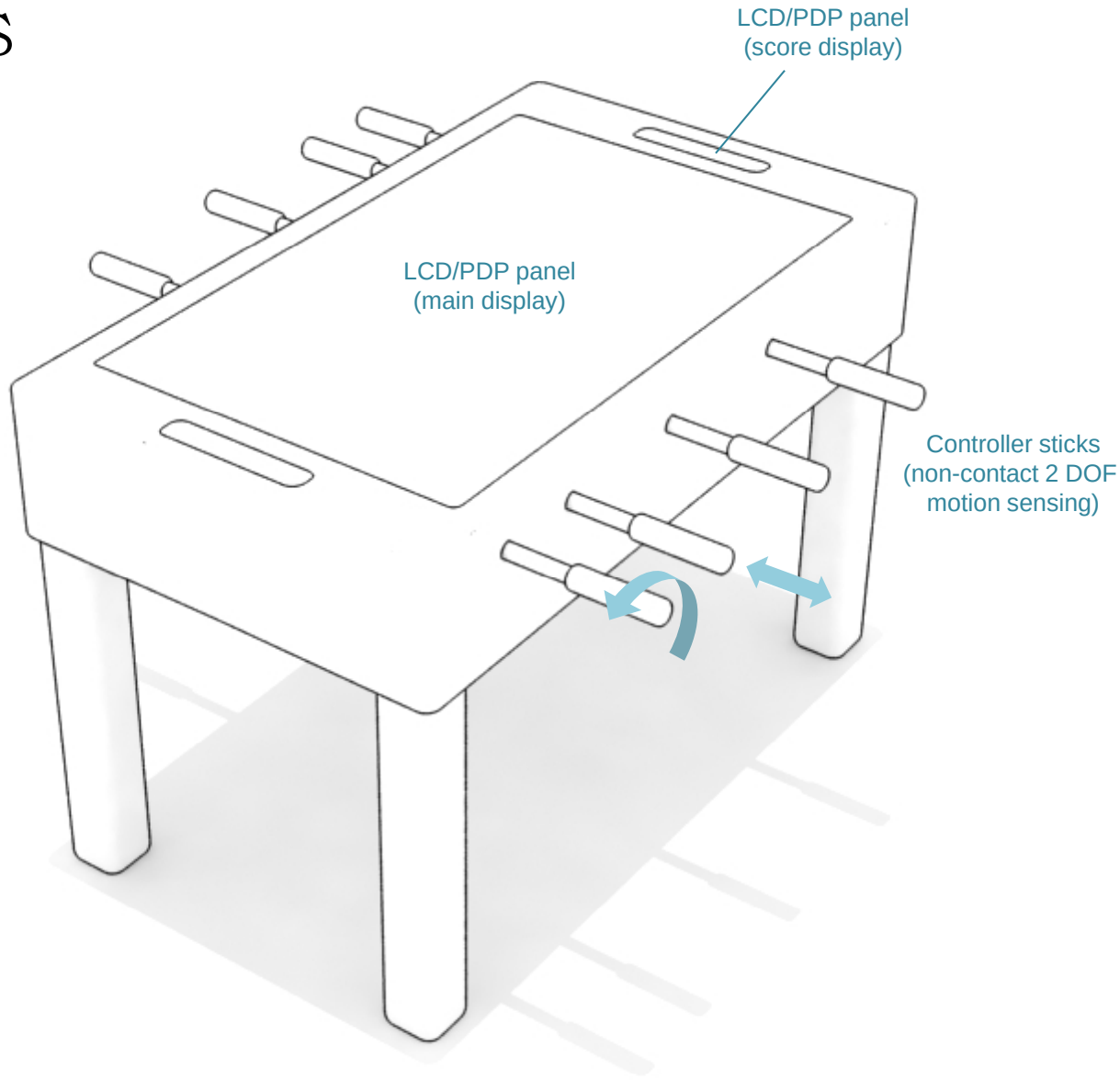
Concepts

everyware

Concepts

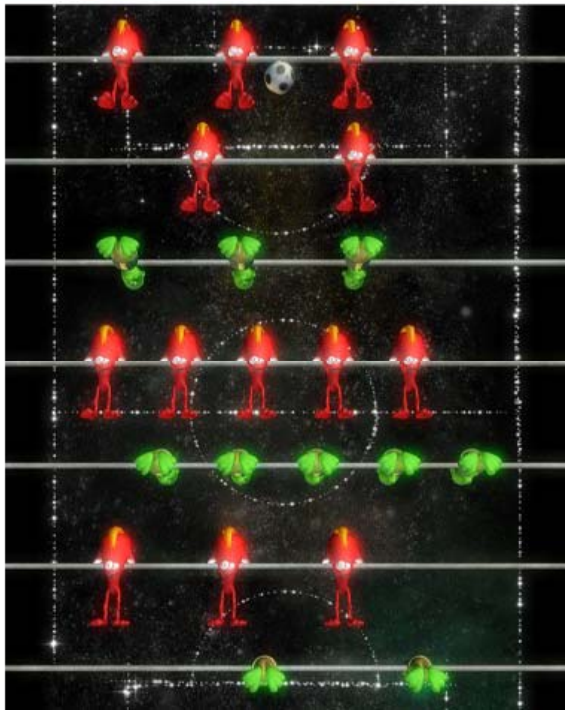
Concepts

Concepts


LCD/PDP panel
(score display)
LCD/PDP panel
(main display)
Controller sticks
(non-contact 2 DOF
motion sensing)

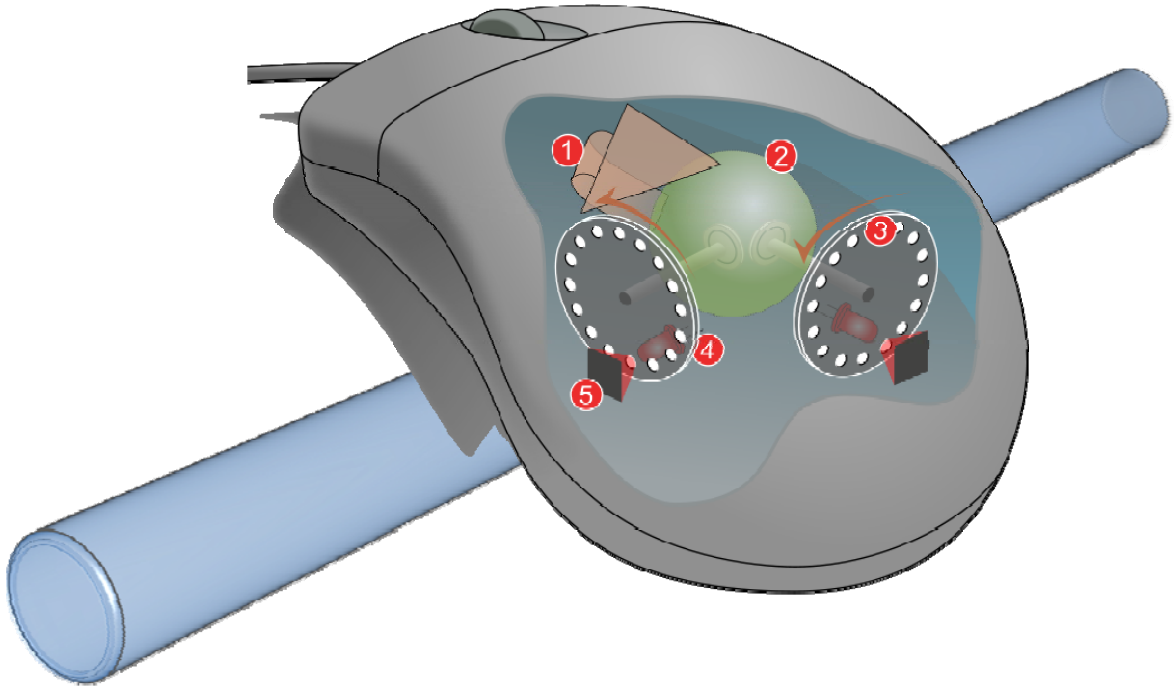
Concepts


SPACE FOOSBALL
오픈 베타 테스트

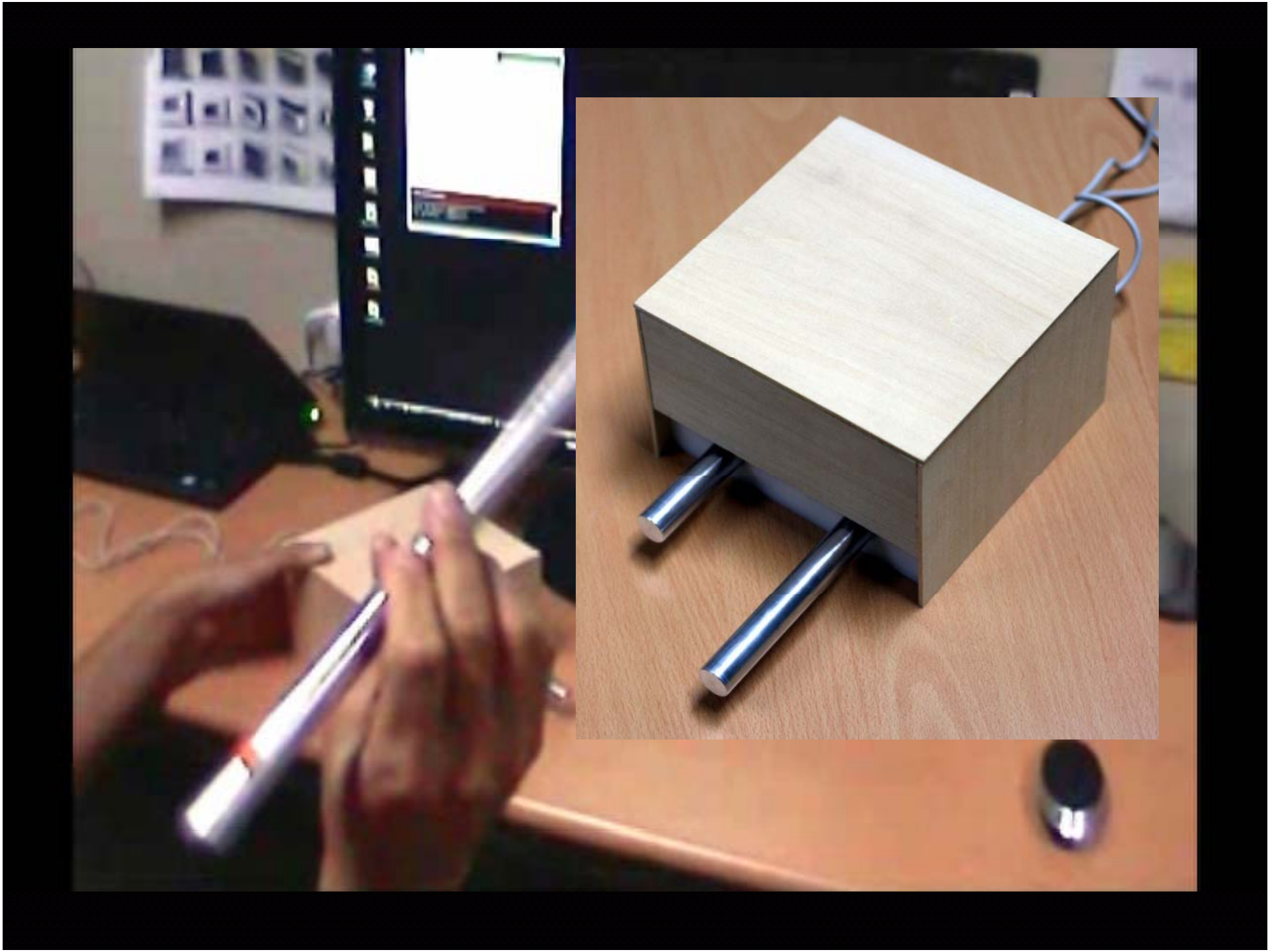


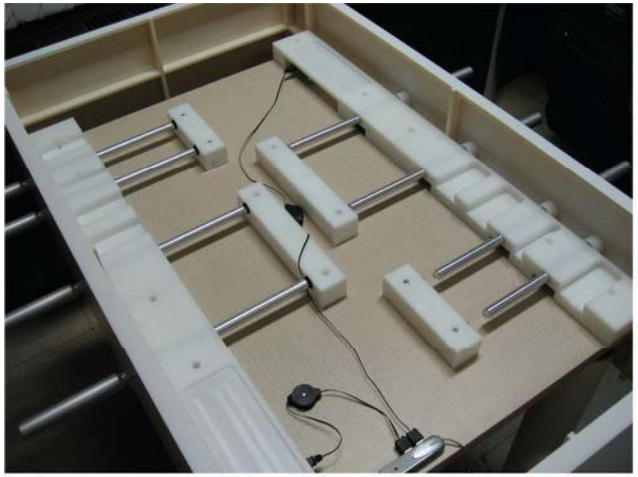
Hardware

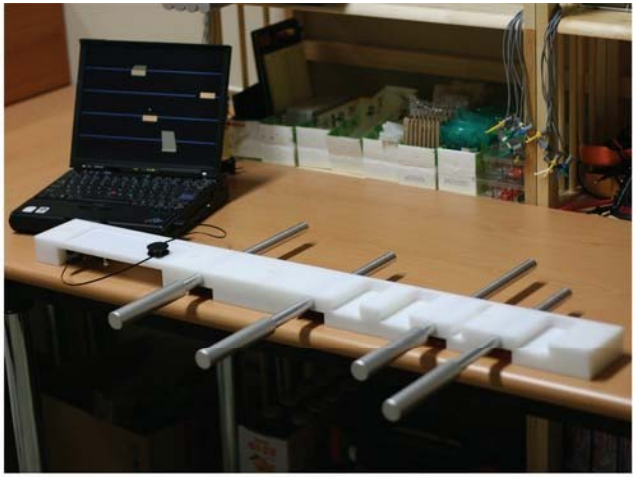
everyware

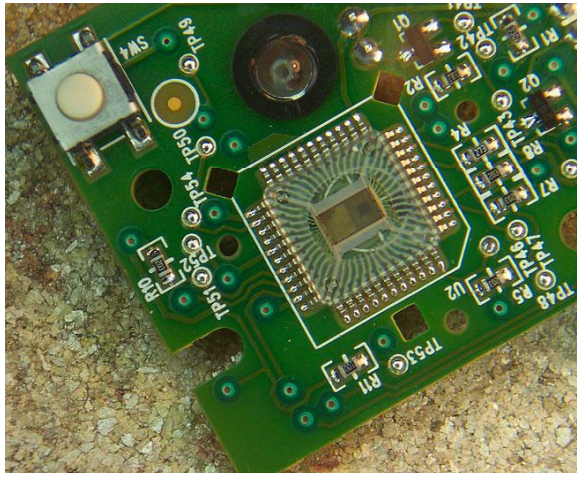


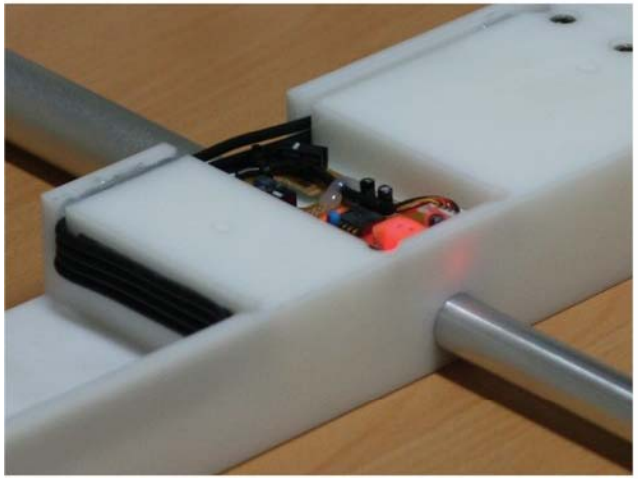
1: Moving the rod turns the ball.
2: X and Y rollers grip the ball and transfer movement.
3: Optical encoding disks include light holes.
4: Infrared LEDs shine through the disks.
5: Sensors gather light pulses to convert to X and Y vectors.

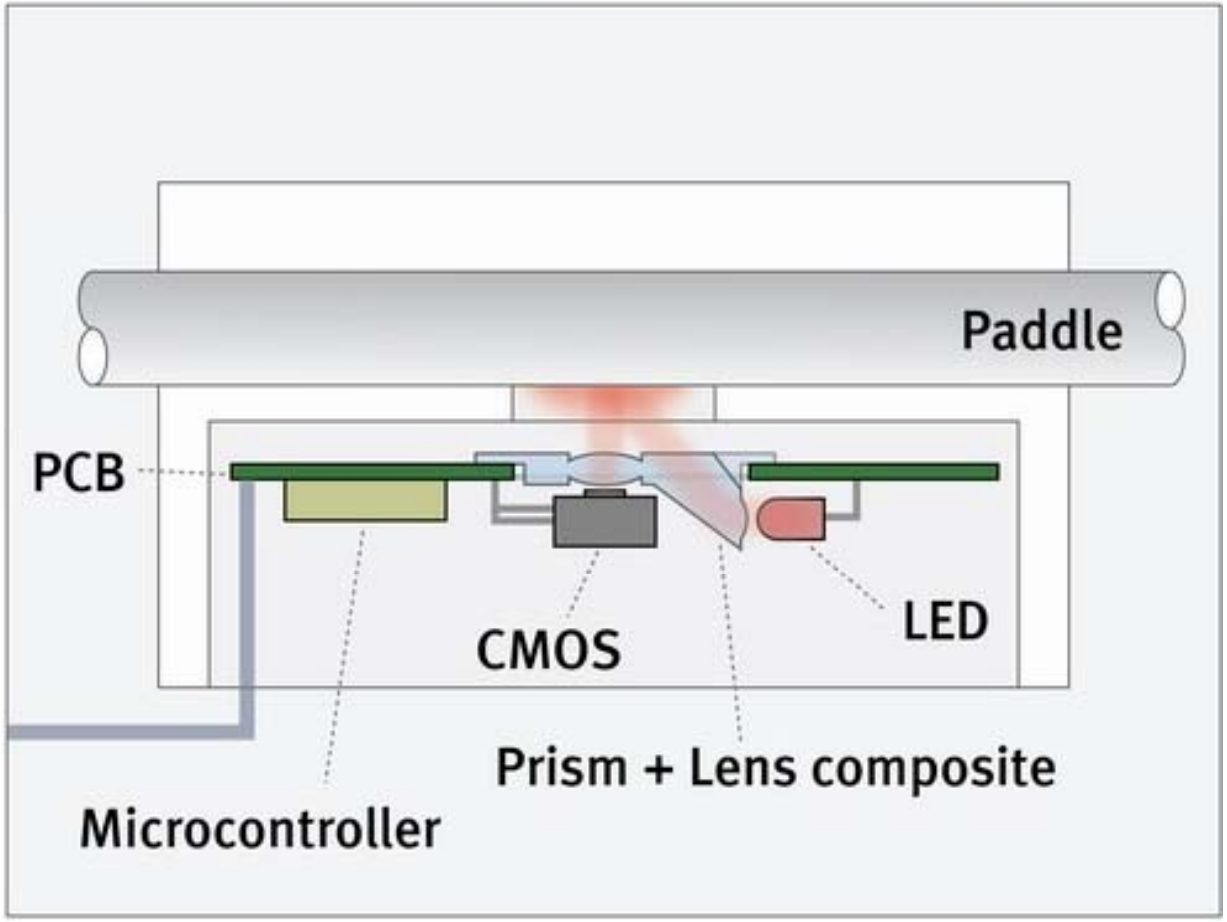


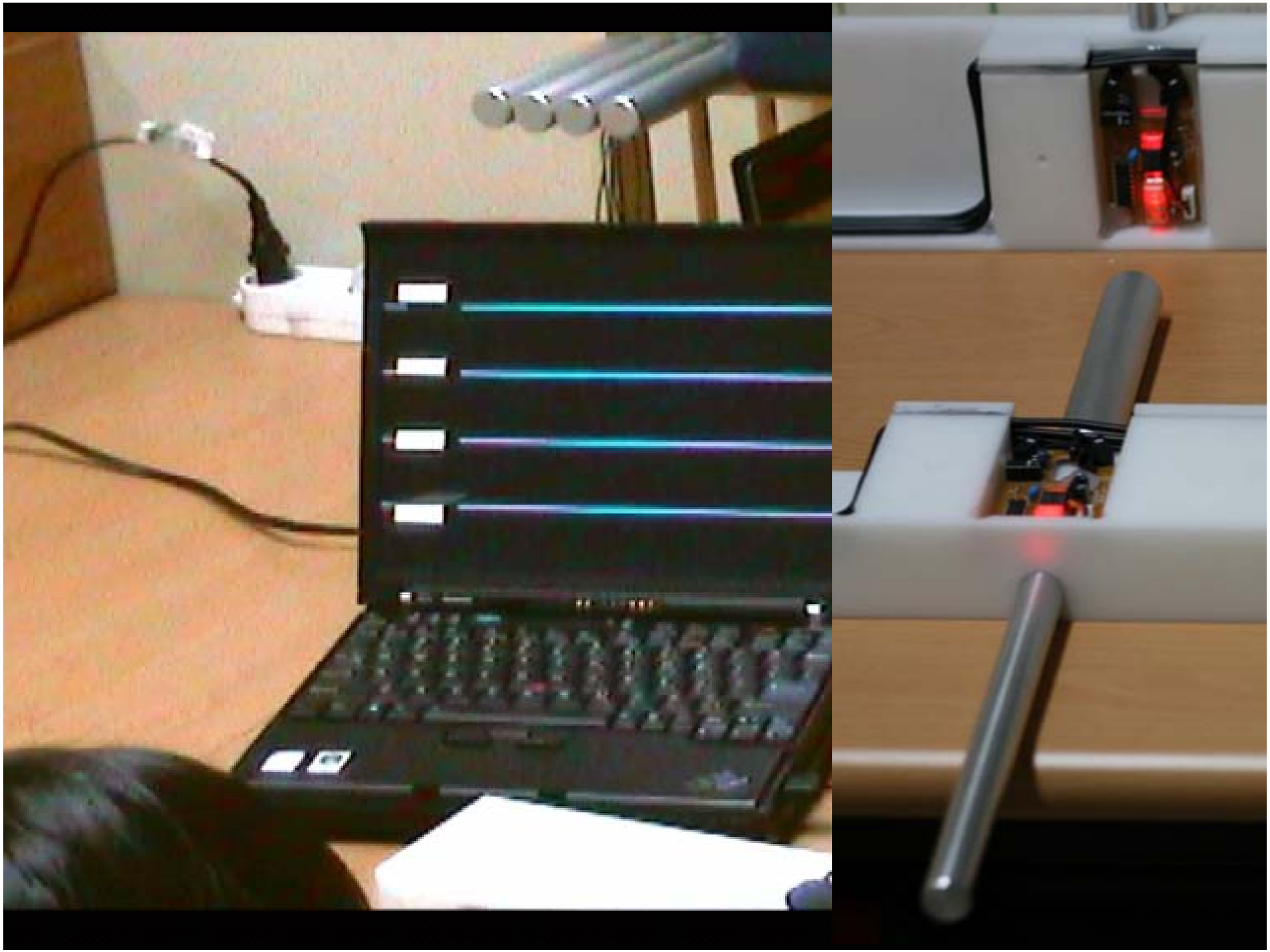


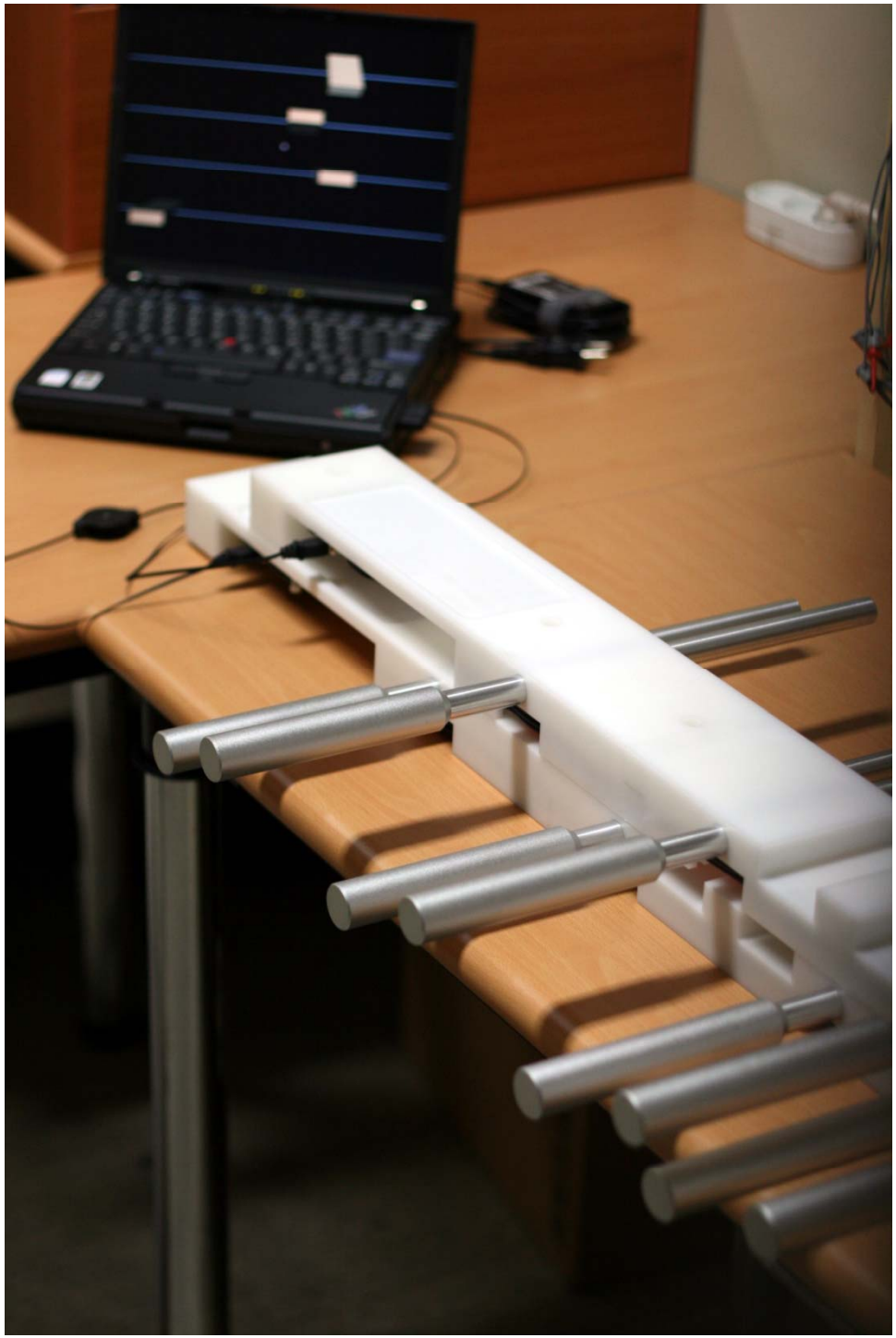


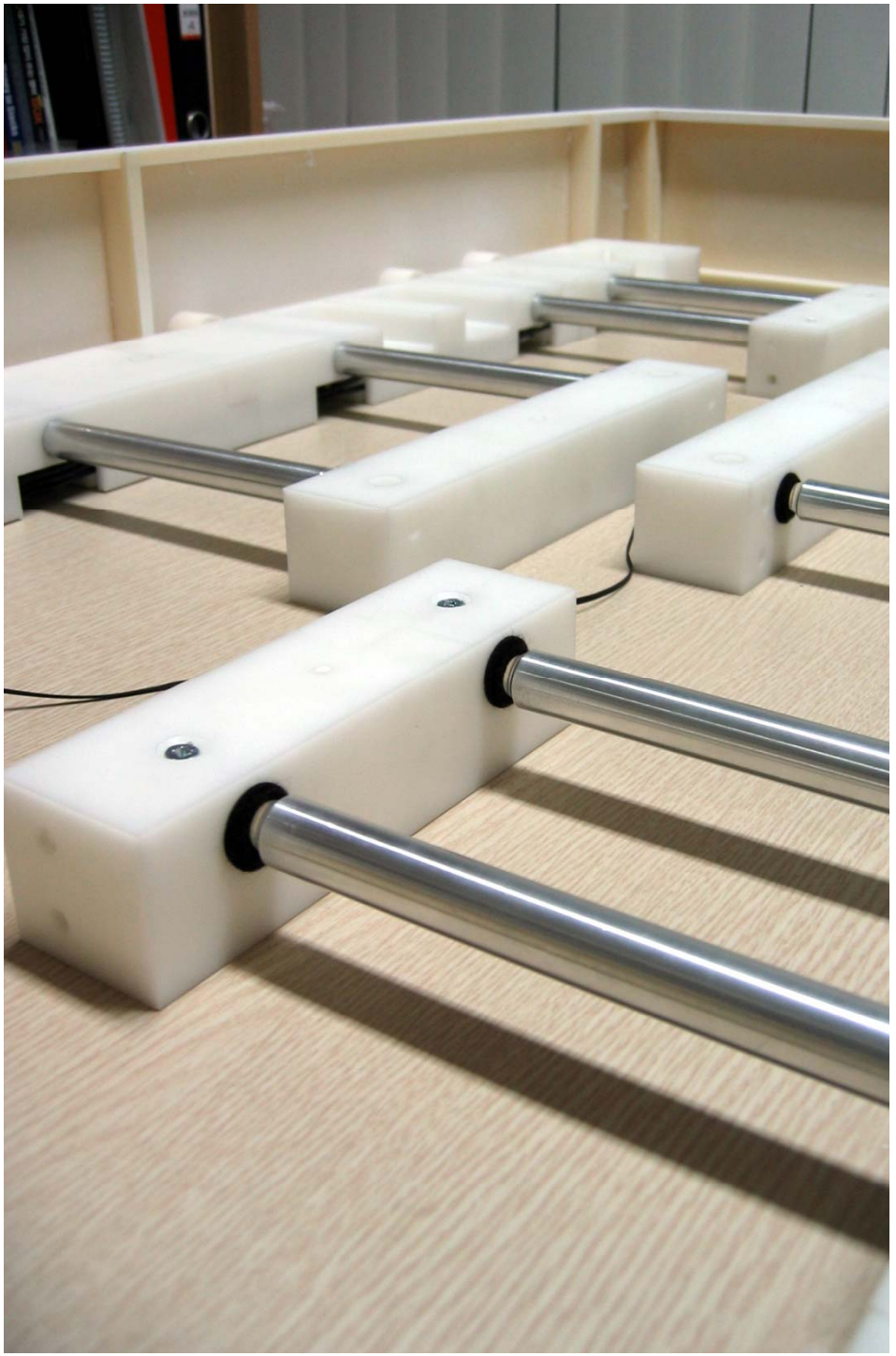


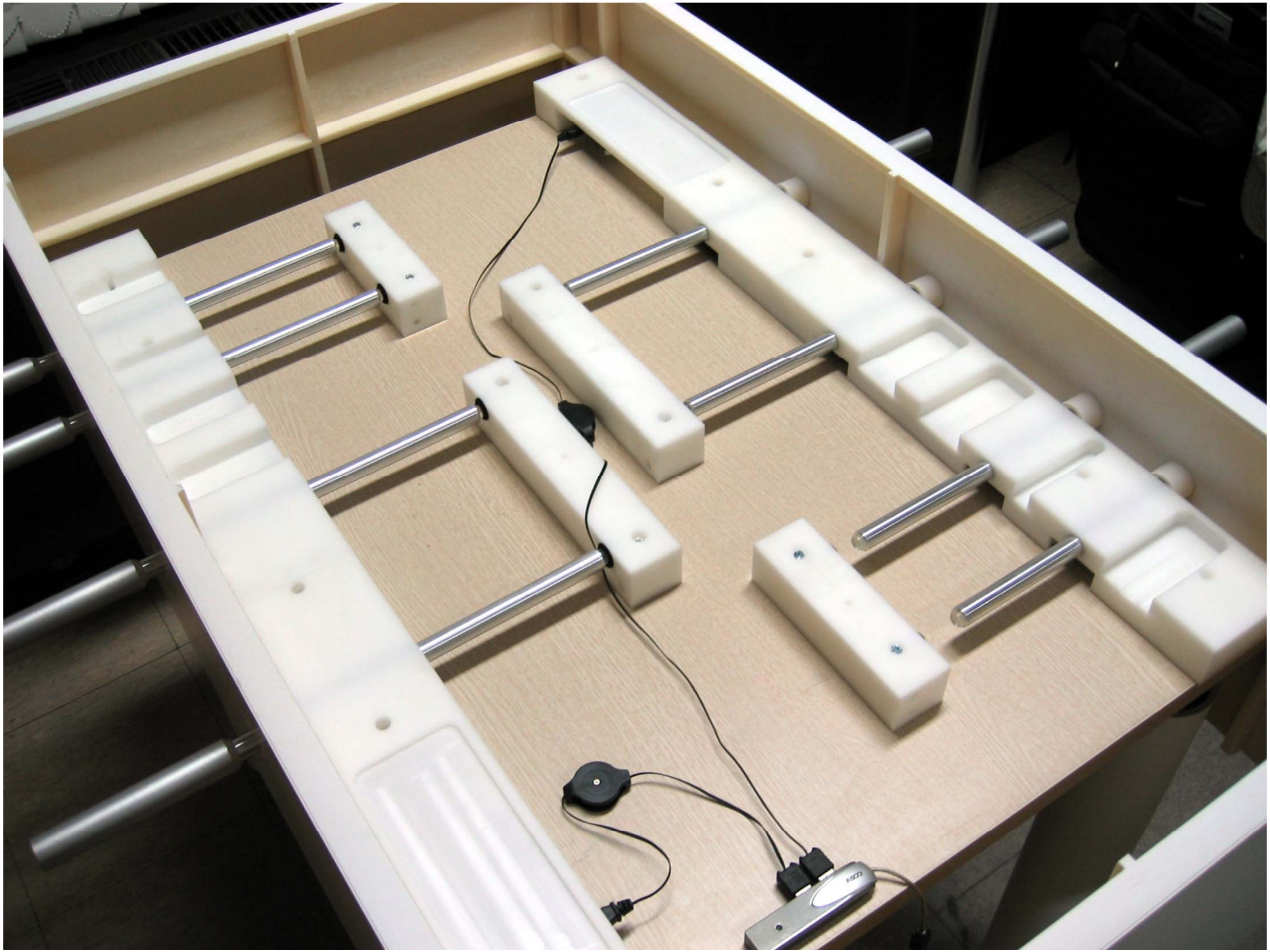



Paddle
PCB
CMOS
LED
Prism + Lens composite
Microcontroller









Virtual Foosball

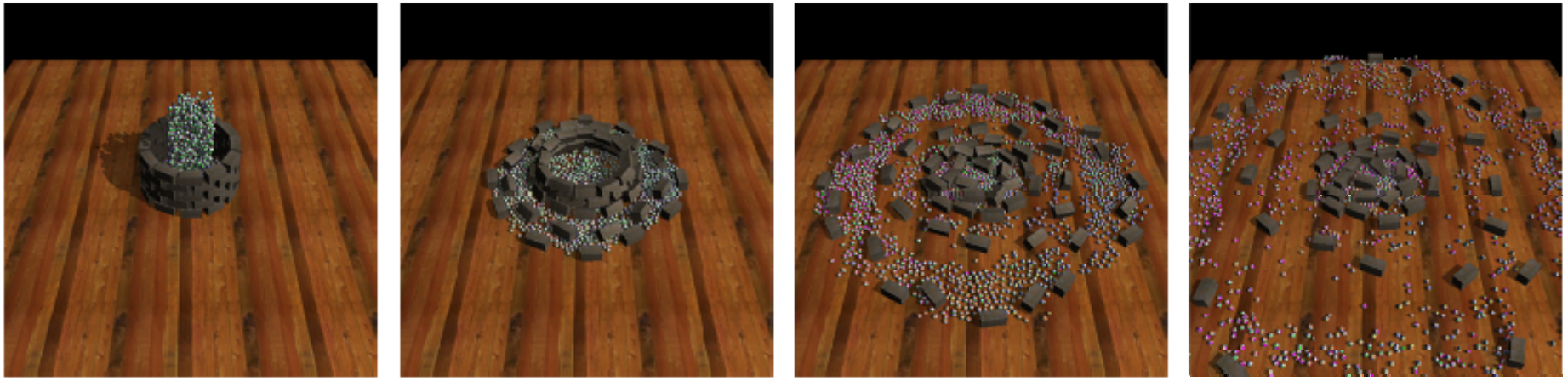
Physics Engine

everyware

PhysX™
by
NVIDIA.

Virtual Physics

REALTIME DYNAMICS SIMULATION LIBRARY

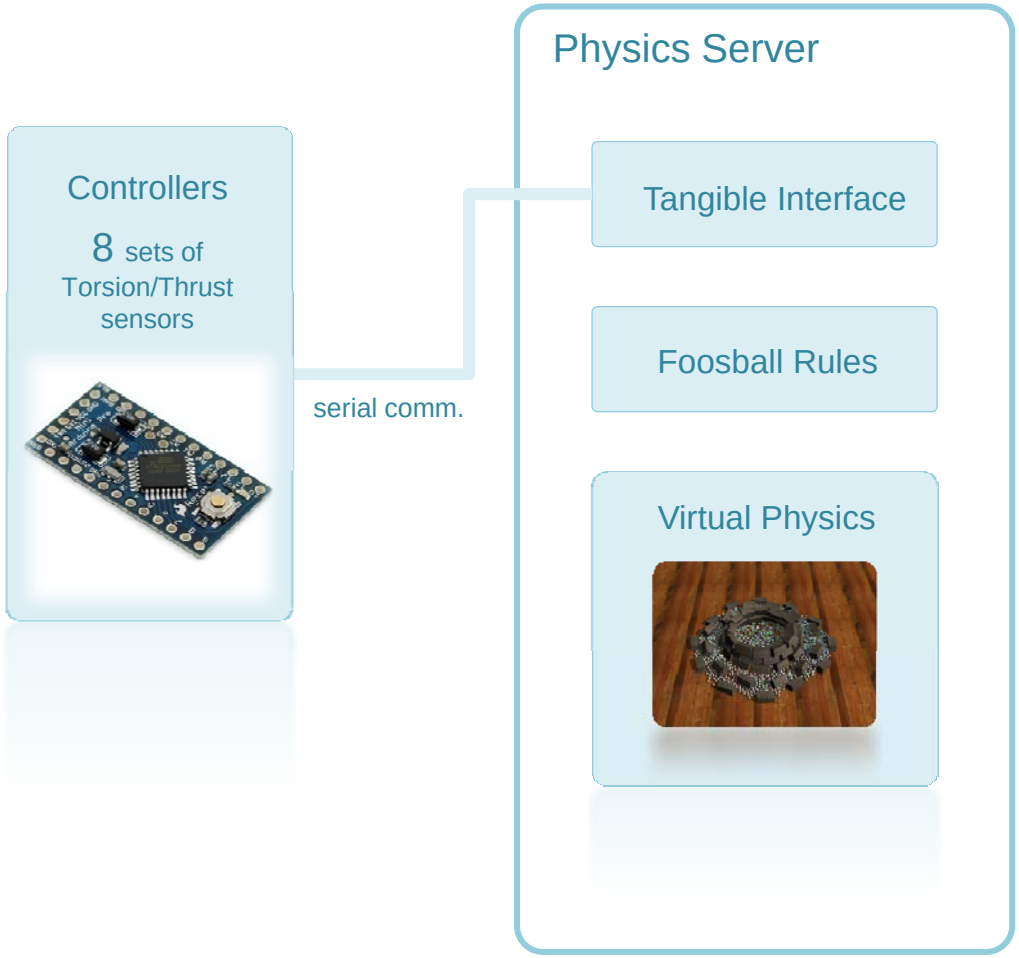


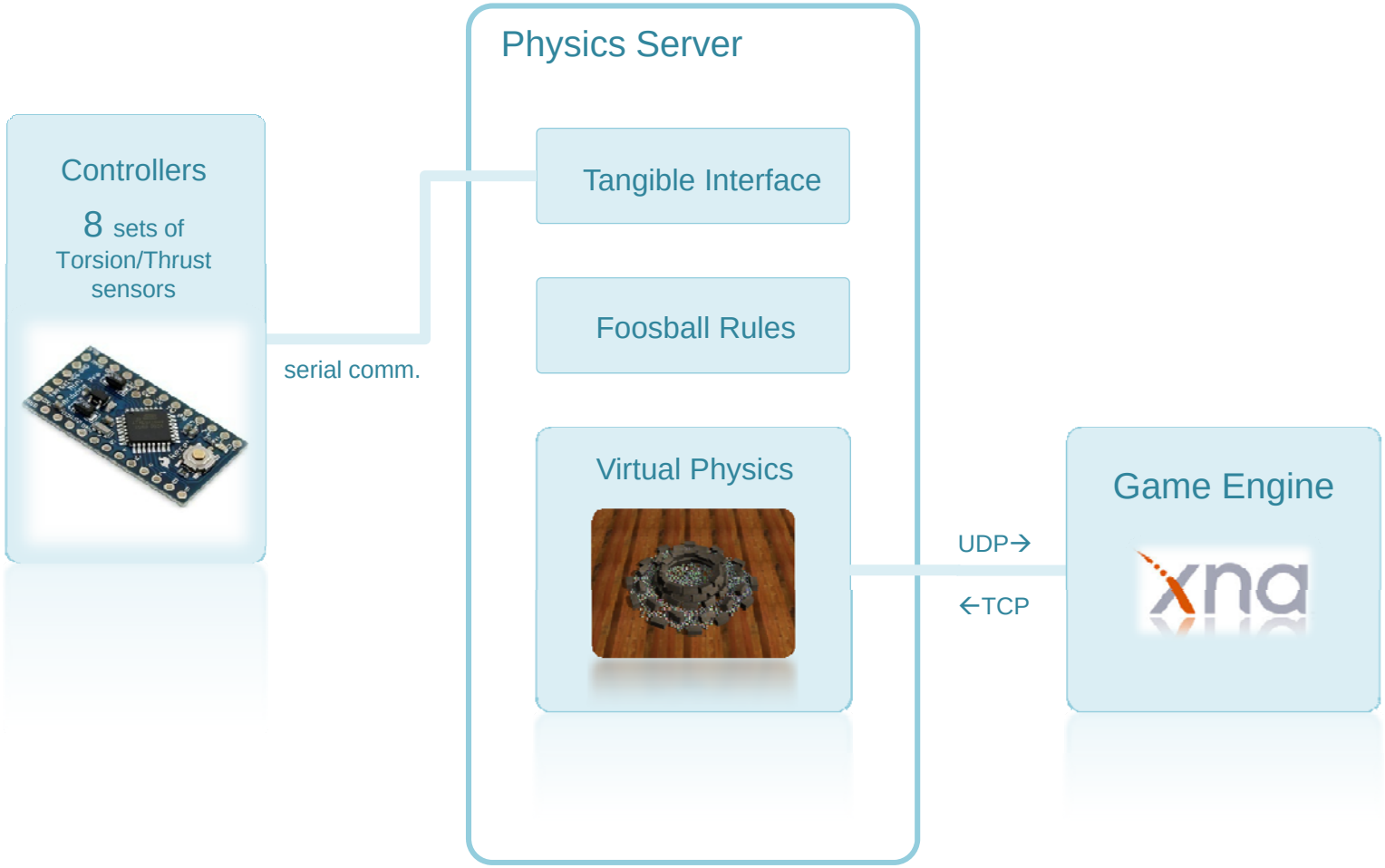
Virtual Physics

Physics Server

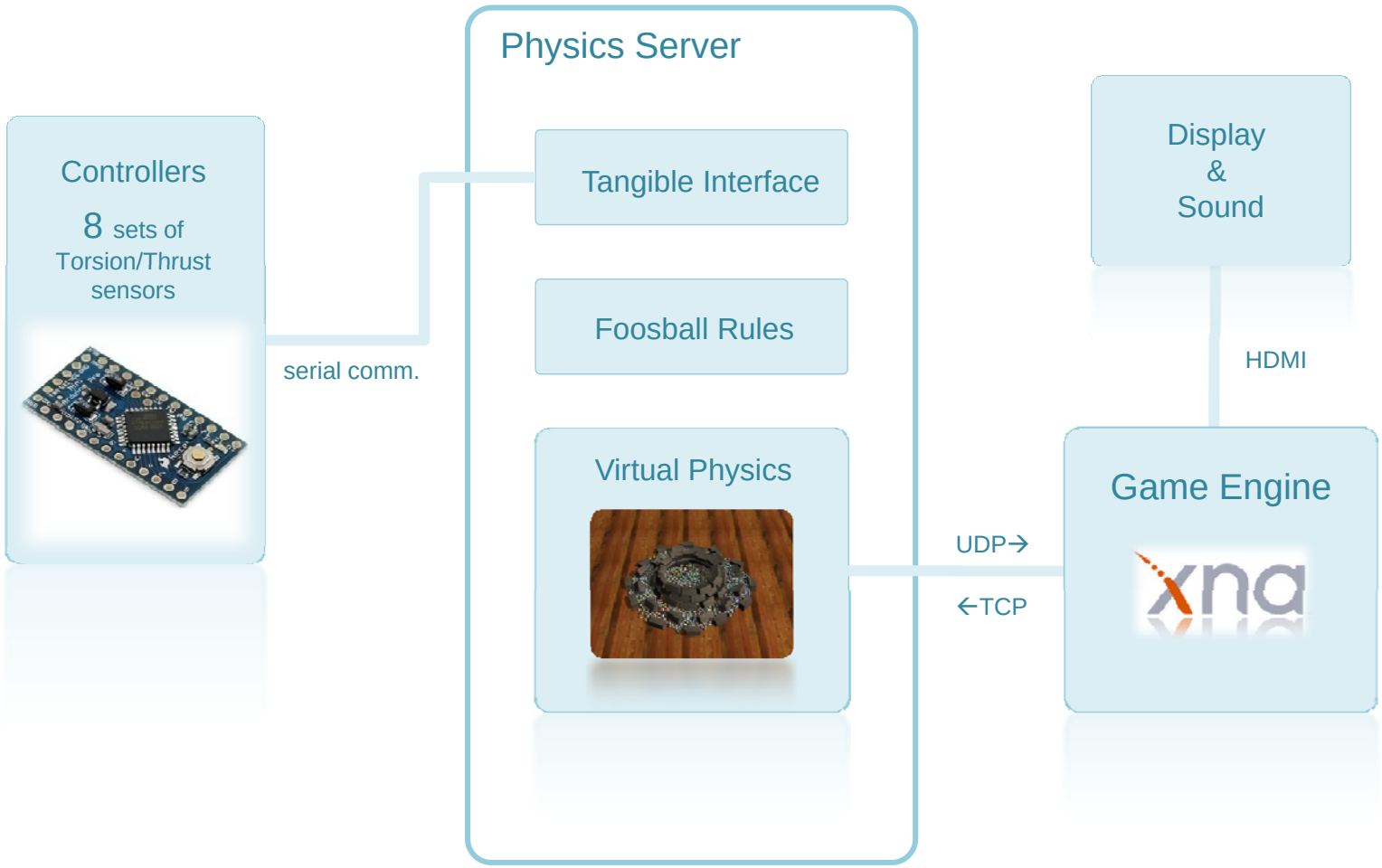
Foosball Rules

Virtual Physics


Physics Server
Controllers
8 sets of Torsion/Thrust sensors
serial comm.
Tangible Interface
Foosball Rules
Virtual Physics



UDP : simulation objects posture refresh @ ~10000 times/s
TCP : simulation mode switching orders @ ~60 times/s



UDP : simulation objects posture refresh @ ~10000 times/s
TCP : simulation mode switching orders @ ~60 times/s

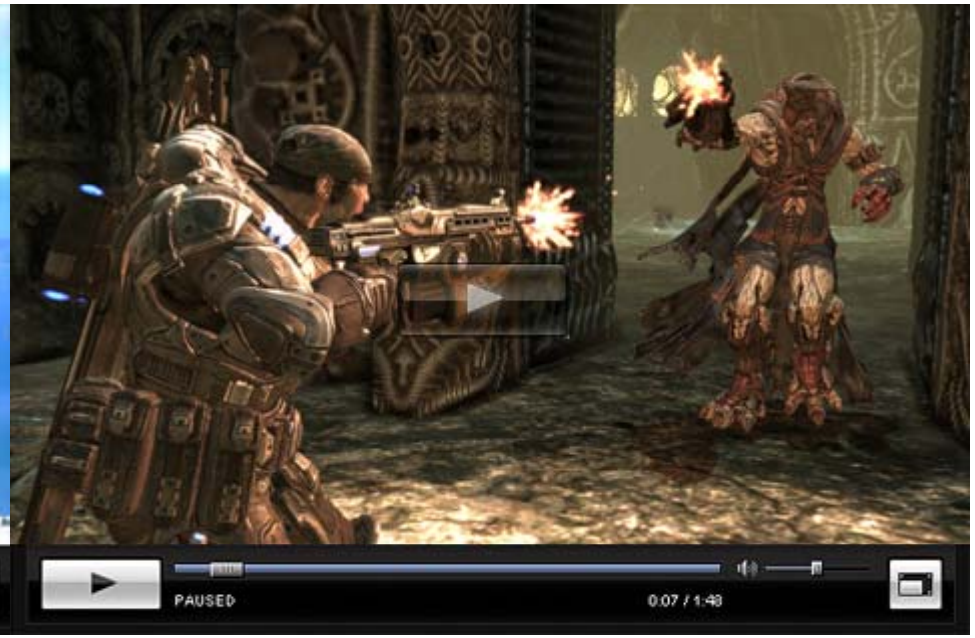
Virtual Foosball

Render Engine

everyware

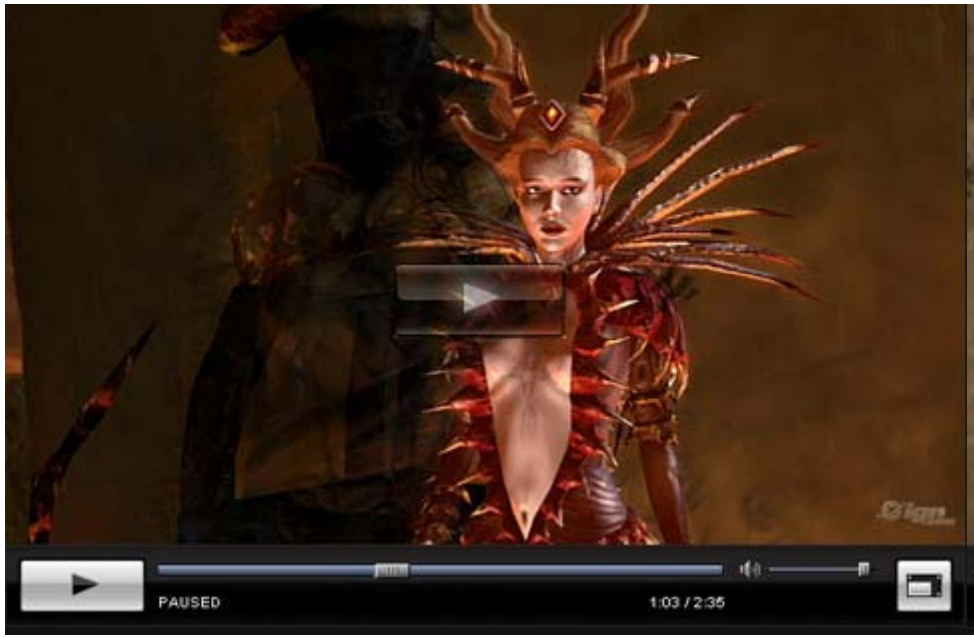
The Dead Engine
Dead Space, Dante's Inferno

The Dead Engine
As Seen in: Dead Space, Dante's Inferno



Unreal Engine
As Seen In: Gears of War, Mass Effect, BioShock, Unreal Tournament, GRAW, Borderlands, Brothers in Arms, Mirror's Edge, Singularity ...

Naughty Dog Game Engine
As Seen in: Uncharted: Drake's Fortune, Uncharted 2: Among Thieves



Geo-Mod Engine
Red Faction: Guerrilla

Anvil Engine
Assassin's Creed series, Prince of Persia, Shaun White Snowboarding

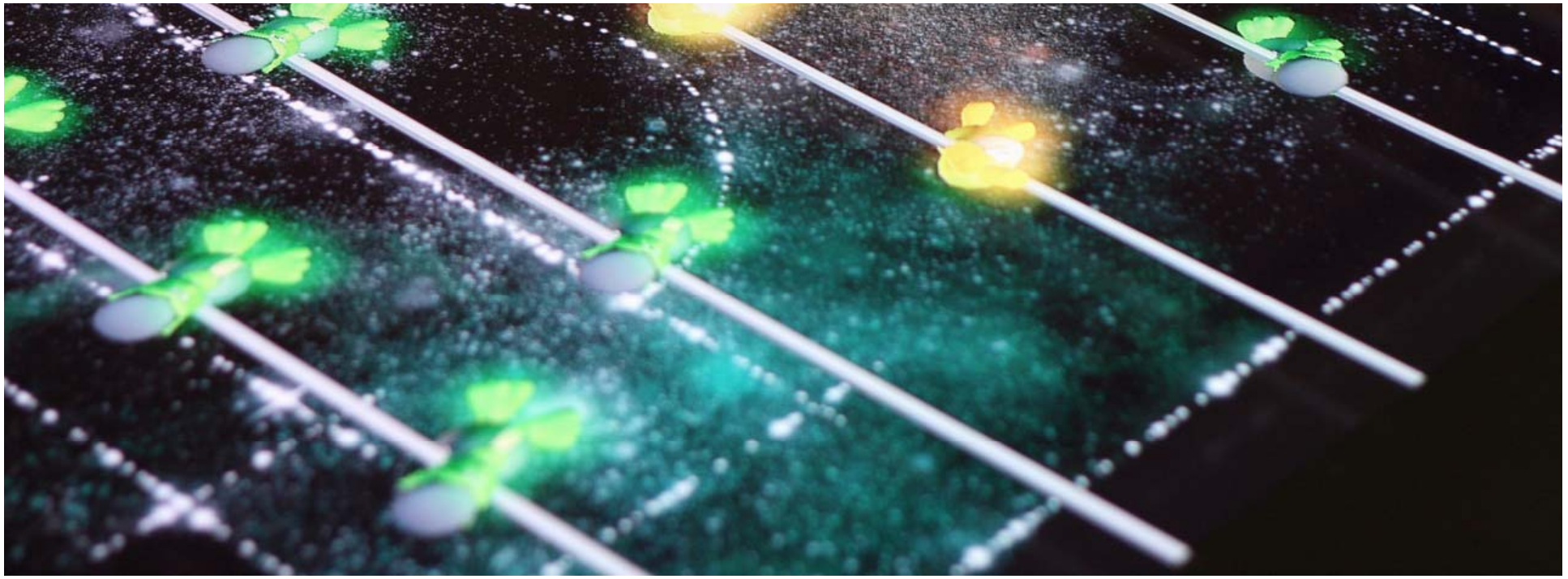
EGO Engine
Colin McRae: DiRT, Race Driver: GRiD, Operation Flashpoint, F1

IW Engine
Call of Duty Series

CryENGINE
As Seen In: Far Cry, Crysis, Crysis Warhead, Crysis 2, Aion: Tower of Eternity

RAGE Engine
As Seen In: Rockstar Presents Table Tennis, GTA IV + Episodes, Midnight Club: Los Angeles, Red Dead Redemption, L.A. Noire (rumoured)

Realtime shadow
Global Illumination
Vertex Shader
Pixel Shader
Normal/Parallax Map
Halo
3D Audio
Motion Blur
Depth of Focus
Render to texture



Realtime shadow
Global Illumination
Vertex Shader
Pixel Shader
Normal/Parallax Map
Halo
3D Audio
Motion Blur
Depth of Focus
Render to texture

Virtual Foosball

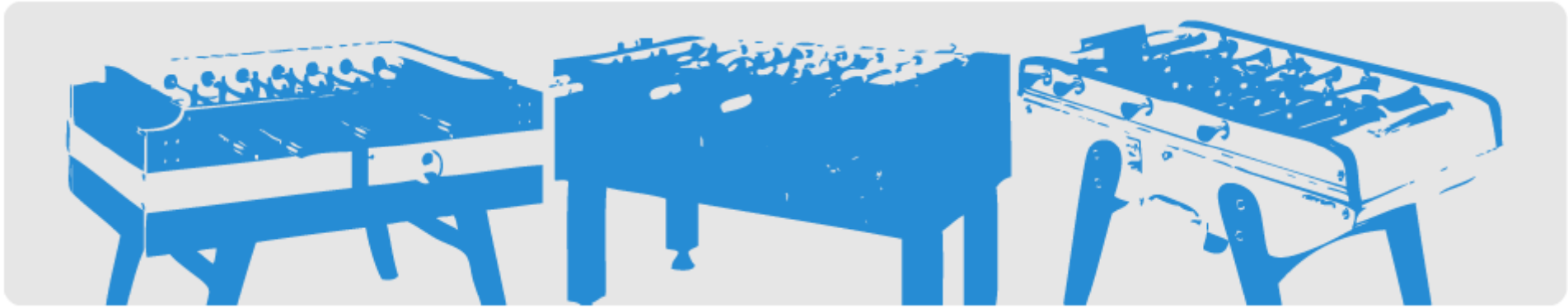
Sales

everyware


SOUTH
AFRICA
2010
FIFA
WORLD CUP


adidas
JABULANI
OFFICIAL MATCH BALL
SOUTH
AFRICA
2010
FIFA

The British Foosball Association



Yunsil,

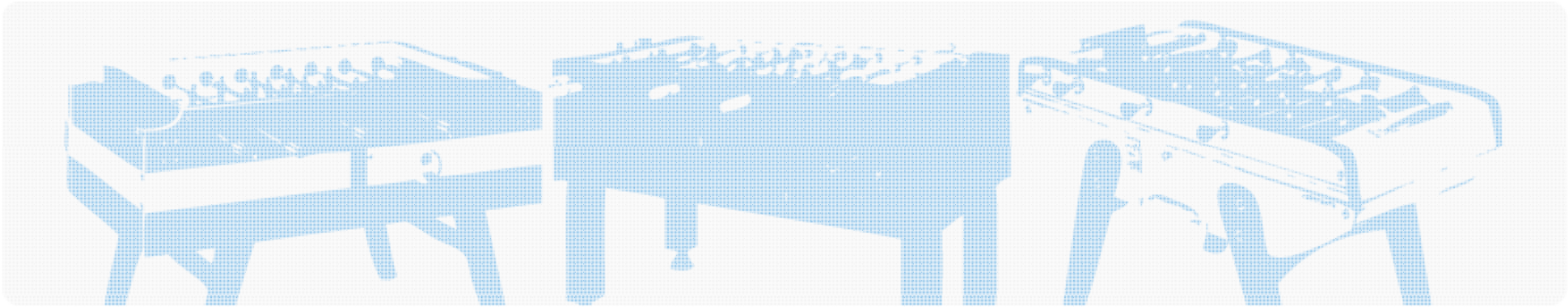
I'm interested in your space table football - I contacted you through Vimeo.
Firstly, I think you should think about entering the XMOS FoosTech Ideas Competition 2009 (http://tinyurl.com/foostech and http://xmos.com/foostech).
Secondly, I think we should discuss the mass market for a game of this nature.
Some of what you have created is unique and would be essential in any commercial implementation. Some however is going to be difficult to commercialize. For example, you have created a conventional table football table but with the table replaced by a screen. The important and unique thing is the decoupling of the tangible user interface (the rods) from the display and physics engine.
My opinion is that to mass commercialize a game like this, you need to build on what already exists. Everyone already has a screen, so I don't believe you are going to find many buyers for your massive new screen - the unit price is going to be simply too high!
So ideally, I would have a set of "handles" which are small joystick-style devices which I can place on my desk. I can play against my computer using my monitor. Crucially and interestingly, if the engine is correctly optimized, I can play against an opponent over the internet.
This builds on existing technology and uses your physics engine and tangible interfaces to create something that would genuinely have mass-market appeal.
I am a Computer Science PhD student so would be happy to help you if you think what I have written is sensible. I think I could find someone to help fund the development of a console-style game.

Best wishes,

Jonathan

--
Jonathan May
Sponsorship & Commercial Development
British Foosball Association
+44 (0)7767 847278

The British Foosball Association



Yunsil,

I'm interested in your space table football... I contacted you through Vimeo

... the unit price is going to be simply too high! So ideally, I would have a set of "handles" which are small **joystick-style devices** which I can place on my desk. I can play against my computer using my monitor. Crucially and interestingly, if the engine is correctly optimized, I can play against an opponent **over the internet**...

--
Jonathan May
Sponsorship & Commercial Development
British Foosball Association
+44 (0)7767 847278

Virtual Foosball

The Upgrade

everyware


PUSH BUTTON TO START.
JUNK YARD
FOOSBALL


START
SETTINGS
JUNK YARD
FOOS BALL
SONY
VAIO
JUNKYARD
FOOSBALL

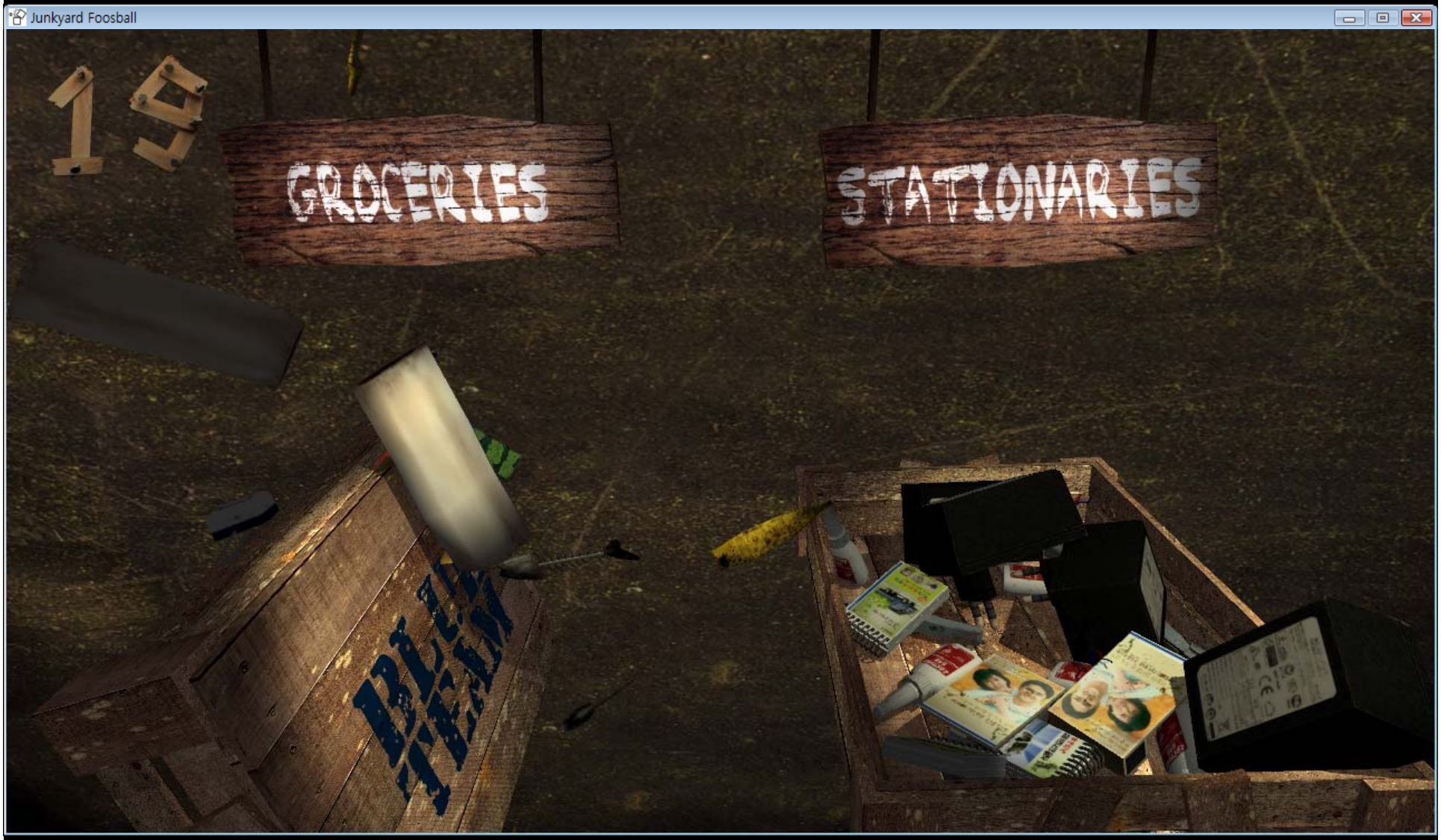

everyware
JUNKYARD
FOOSBALL

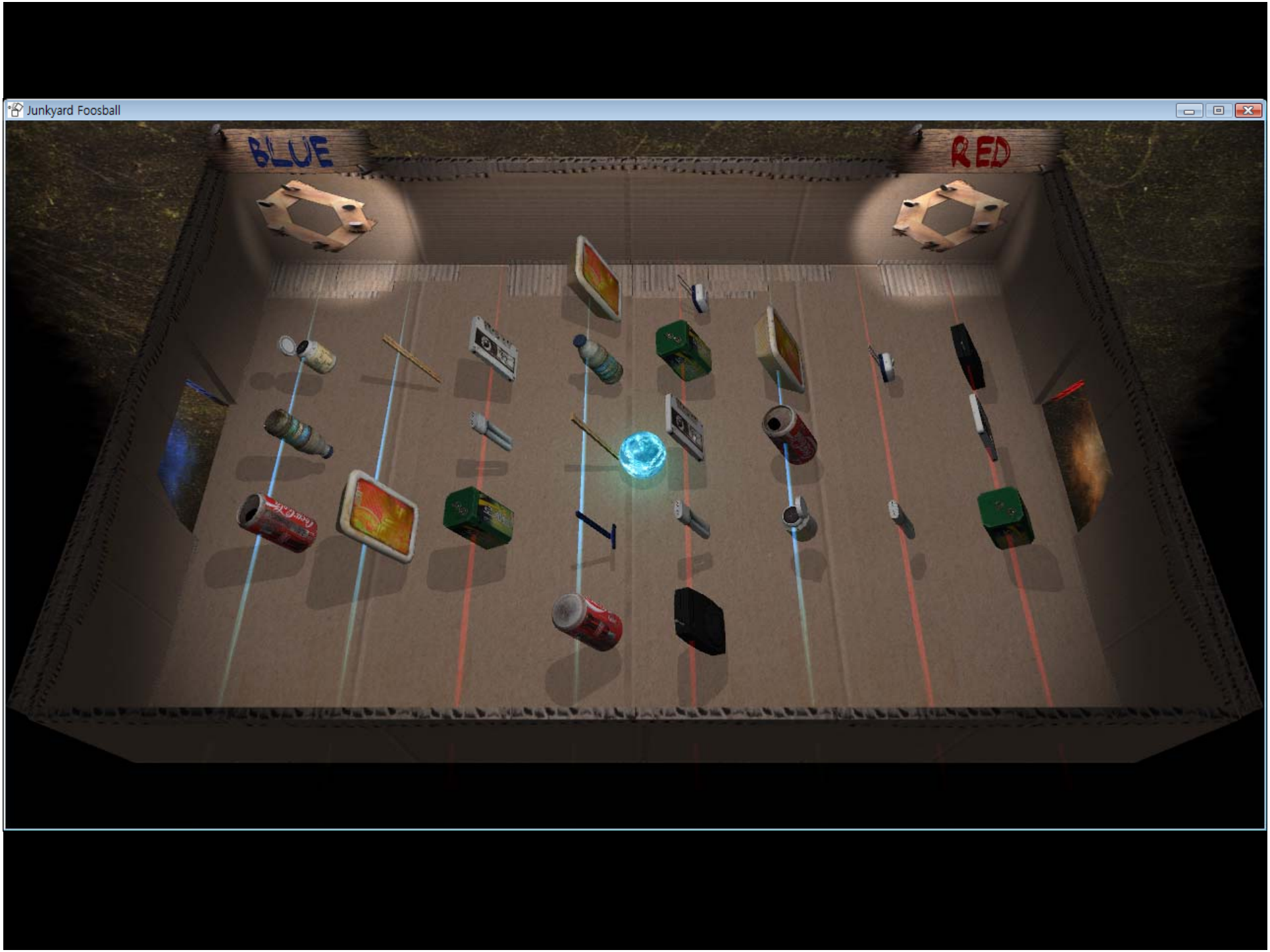

everyware
JUNKYARD
FOOSBALL
everyware
JUNKYARD
FOOSBALL
everyware
JUNKYARD
FOOSBALL


JUNKYARD
FOOSBALL

- ✓ Wireless
- ✓ High speed comm.
- ✓ HD sensing resolution
- ✓ Hybrid physics engine
- ✓ Multiplayer
- ✓ Mass-production Ready


Junkyard Foosball
START
SETTINGS
JUNK YARD
FOOSBALL


Junkyard Foosball
GROCERIES
STATIONARIES


Junkyard Foosball
BLUE
RED

Realtime shadow
Global Illumination
Vertex Shader
Pixel Shader
Normal/Parallax Map

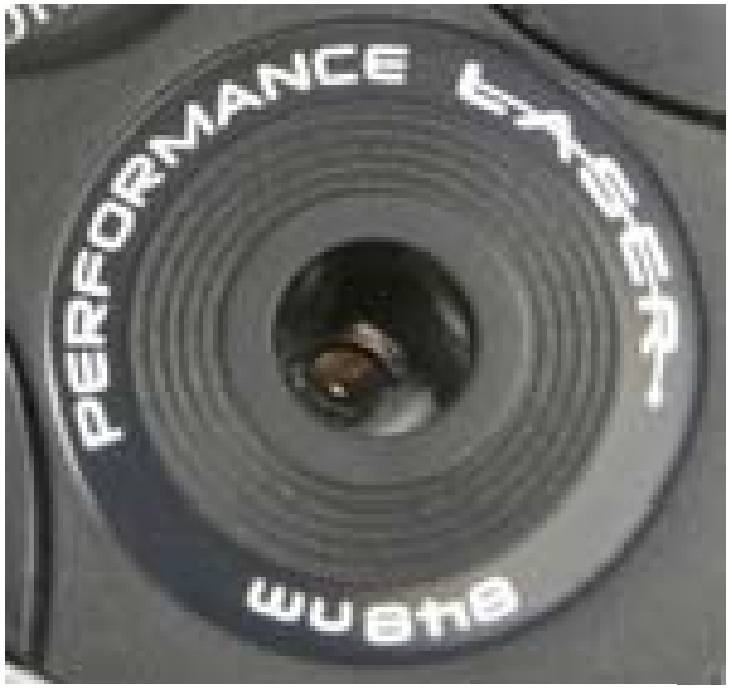
Halo
3D Audio
Motion Blur
Depth of Focus
Render to texture

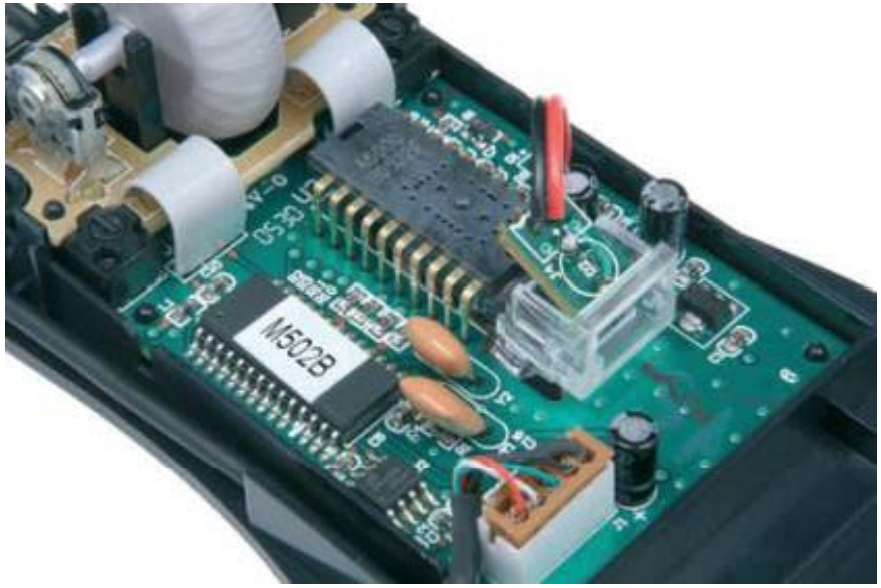
Virtual Foosball

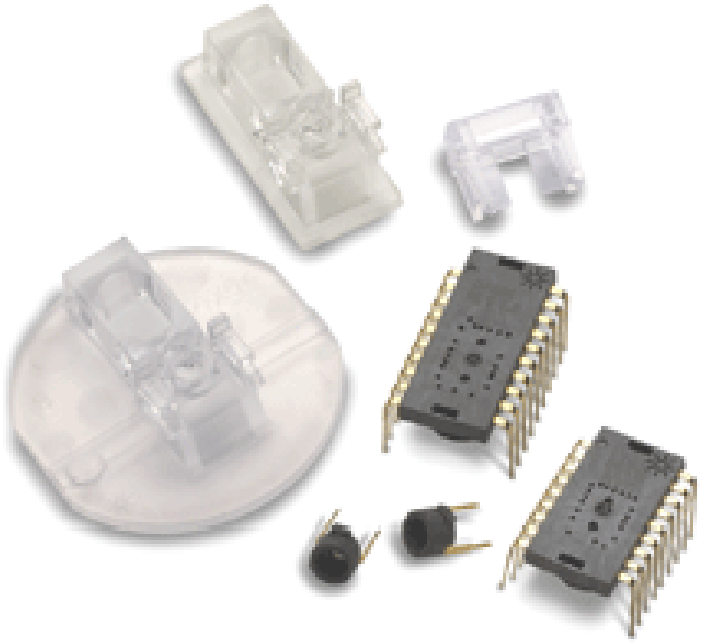
Technology

everyware


Logitech
Logitech
300
320
200
210
310


PERFORMANCE LASER

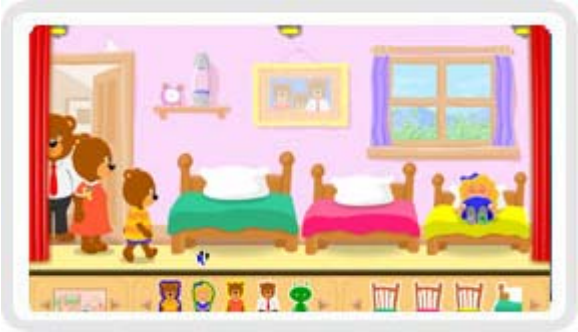

M502B



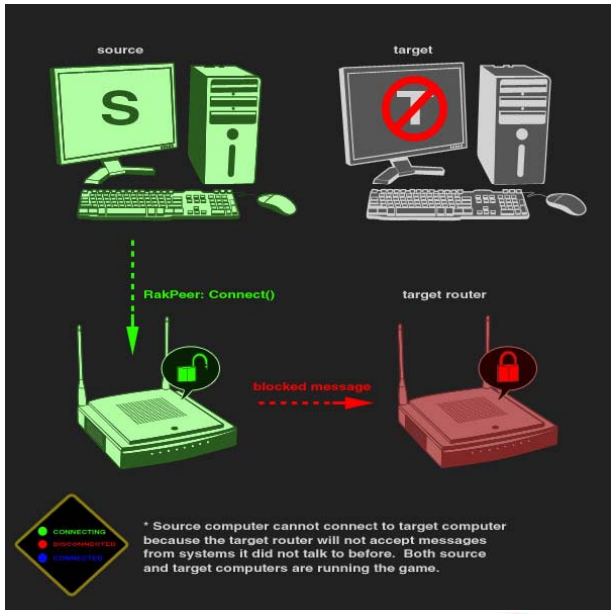
One PC, Multiple Mice, Endless Possibilities

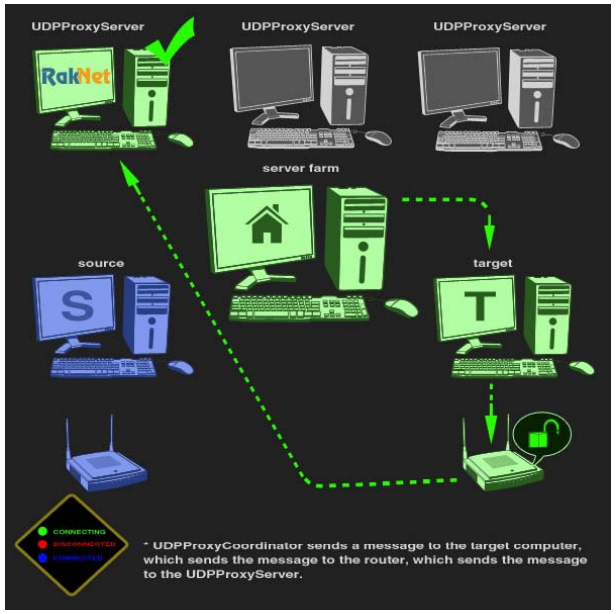
Make your PowerPoint presentations interactive

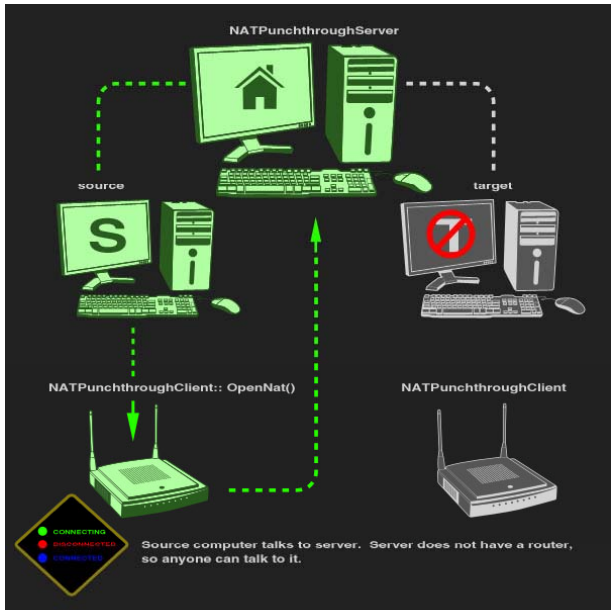
Download Beta



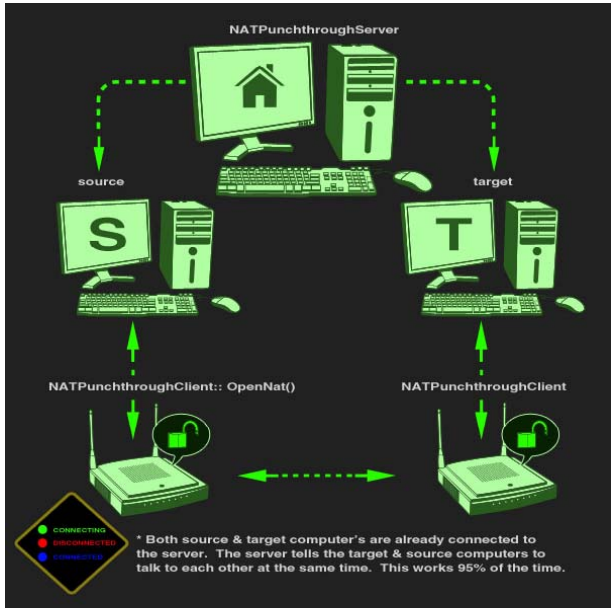

Multipoint Mouse
2551

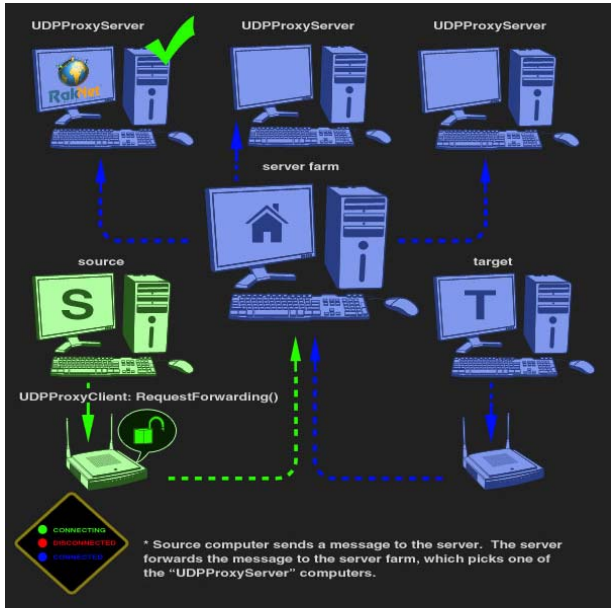


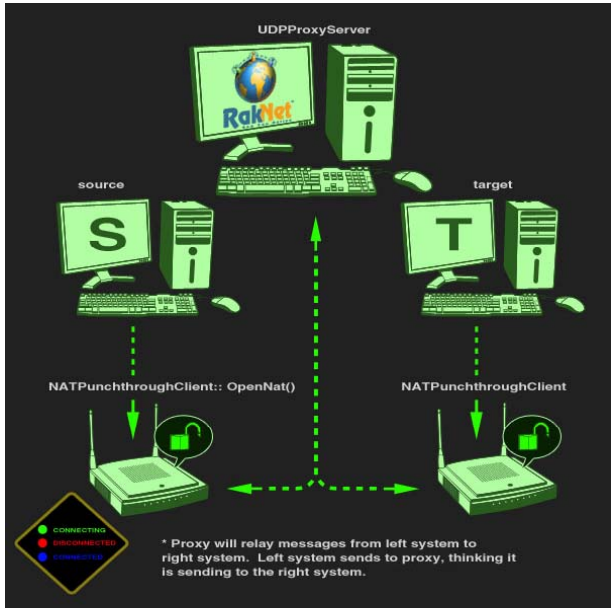


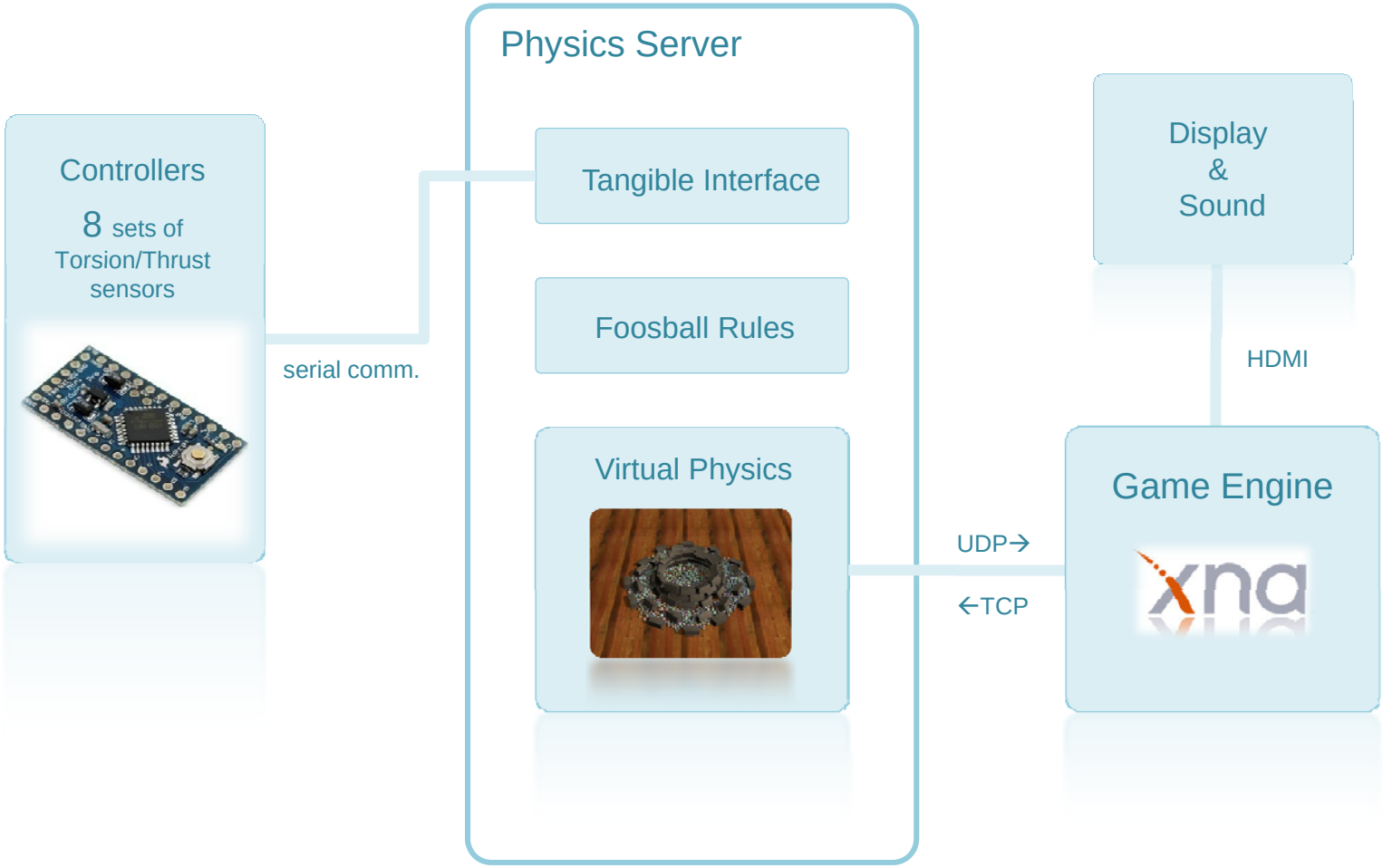


“NAT Punch-Through”









UDP : simulation objects posture refresh @ ~10000 times/s
TCP : simulation mode switching orders @ ~60 times/s

Virtual Physics

VP.net

Game Engine

VP.net

Up to 250 Mice

Multipoint SDK

Game Engine

VP.net


Up to 250 Mice
Multipoint SDK
Game Engine
VP.net
Up to 250 Mice
Multipoint SDK
Game Engine
VP.net
RakNet
Up to 250 Mice
Multipoint SDK
Game Engine
VP.net


Up to 250 Mice
Multipoint SDK
Game Engine
VP.net
Virtual Physics
Up to 250 Mice
Multipoint SDK
Game Engine
VP.net
Virtual Physics
RakNet
Up to 250 Mice
Multipoint SDK
Game Engine
VP.net
Virtual Physics