International Journal of Modeling, Simulation,
and Scientific Computing
Vol. 4, Suppl. 1 (2013) 1340002 (20 pages)

DOI: 10.1142/S1793962313400023

DESIGN AND IMPLEMENTATION OF REAL-TIME PHYSICS SIMULATION ENGINE FOR e-ENTERTAINMENT

MYUNGSOO BAE* and YOUNG J. KIM†

Computer Science and Engineering
Ewha Womans University, Seoul, Korea
**msbae@ewha.ac.kr*
†*kimy@ewha.ac.kr*

Received 20 December 2012
Accepted 28 December 2012
Published 2 April 2013

We present our recent research results regarding the designing and implementation of real-time physics simulation engines, which aim at developing physics-inspired e-entertainment such as computer games, mobile applications, interactive TV and other smart media in Korea. Our real-time physics engine consists of three functional components: rigid body dynamics simulation, deformable body simulation, and data-driven physics simulation. The core simulation techniques to realize these simulation components include real-time collision detection and response, large-scale model simulation, and character model control. In this paper, we highlight these features and demonstrate their performances. We also showcase some of the gaming applications that we have integrated our physics engine into.

Keywords: Real-time physics simulation engine; rigid body dynamics simulation; deformable body simulation; data-driven simulation.

1. Introduction

Recently, digital contents have been on the rise as a key industry in the knowledge and information society with strong internet and wire–wireless communications infrastructure. It has become a core industry, creating high valued products, as digital content is relatively easy and inexpensive to manufacture and to distribute. Thus, many industrialized countries have been investing a significant of resources to boost the digital content industry and dominate the market.

For e-entertainment content such as computer games, real-time physics simulation technology is a very important factor as this technology helps users become immersed in computer games they play. Along with the improved computer hardware performance for gaming and the recent development of computer graphics

†Corresponding author.

techniques, offline physics simulation techniques, mainly used for movie special effects, are improving to be capable of real-time processing for computer games.

Real-time physics simulation technology usually refers to that which visually and physically simulates interactions between computer objects at interactive rates in a realistic manner. The real-time rendering techniques in recent computer games achieved nearly the same realism of the offline rendering used in film-making. On the other hand, since the quality of object interactions in the games are far from being realistic, the development of interesting and immersive contents that mimic the real world can be limited.

Yet as modern computer games become more diverse in terms of genre and as the gaming hardware utilized continues to improve, demands for realistic simulation are being generated. Therefore, it is expected that physics engines will be used more and more in a large number of game titles that will be developed in the near future.

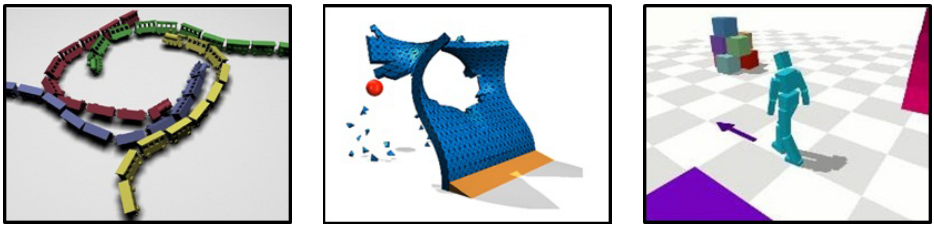
Many gaming companies and e-entertainment content developers in Korea also realize that the physics simulation techniques are the core factors for developing killer titles. However, there are some difficulties in using physics engines to develop physics-based games; these are expensive license fees, absence of a customized development environment, and the fear of failure to produce games due to a lack of understanding the challenging simulation techniques.[1] Because of these reasons, most Korean gaming companies either completely depend on well-known, existing physics engines for developing physics-based games, or simply avoid using the techniques. Thus, creating a real-time physics simulation engine or alternative technologies has drawn national attention in Korean gaming community, and it is considered urgent to resolve these problems.

The goal of our research project was to develop core technologies which perform physically-based simulation accurately with high-rate real-time performance for e-entertainment, including collision response, destruction and stacking objects.

These technologies can be used for various real-time e-entertainment contents such as computer games, interactive TV, movie special effects, mobile contents, virtual reality, etc. These techniques may become killer applications for high-performance hardware such as multi-core CPUs and GPUs. Moreover, these techniques can be applied to broader areas of such as computer-aided engineering (CAE)-based simulation, robot motion planning, battle simulation, virtual medical simulation and remote surgery and e-learning contents.[1]

Ultimately, we aim to reduce production costs and promote an efficient development environment for both Korean and worldwide game makers through the technologies developed in our research project. We also open-source some core algorithms for real-time physics simulation including massive-data processing, efficient data structure, numerical analysis, and allow researchers in the various fields of engineering to use the software.

Our real-time physics engine consists of three functional components: rigid body dynamics simulation, deformable body simulation and data-driven physics simulation as shown in Fig. 1. In this paper, we highlight the main features of our real-time



(a) Rigid body dynamics simulation

(b) Deformable body simulation

(c) Data-driven physics simulation

Fig. 1. Three main components of our real-time physics simulation engine.

physics simulation engine and their performances. The rest of this paper is organized as follows. Section 2 presents related work. Section 3 describes our research project overview. Section 4 summarizes the main features and performances of our physics simulation engine. Section 5 concludes our paper.

2. Related Work

Real-time physics simulation technologies appeared in late 1990s, and few of companies such as HOVOK and AGEIA succeeded commercially in their businesses. According to the market share report[2] in 2009, HAVOK, NVIDIA PhysX, open dynamics engine (ODE) and BULLET shared about 64% of the game physics simulation market. These simulation engines are also referred to as game middleware that follows a modular approach to build and expand simulation components.

Just like our own simulation engine, we classify existing simulation techniques as real-time rigid body dynamics simulation, real-time deformable body physics simulation and data-driven physics simulation, and briefly survey the current techniques.

Real-time rigid body dynamics simulation. Real-time rigid body dynamics simulation techniques deal with how to realistically simulate collision responses between rigid bodies and/or articulated bodies. Research of real-time collision detection between rigid bodies and articulated bodies has been carried out for more than 20 years in various areas such as computer graphics, robotics and virtual reality.[3] On the collision detection side, the GAMMA research group at University of North Carolina (UNC), USA has extensively studied this subject, and has open-sourced a number of softwares for collision detection such as I-COLLIDE, RAPID, PQP, SWIFT++, PIVOT, QuickCD, DEEP, Solid, etc. On the collision response side, constraints-, penalty- and impulse-based techniques have been proposed and have been successfully combined with collision detection techniques.[4] Commercially popular physics engines such as Intel's Havok, NVIDIA PhysX, ODE, BULLET, Newton, etc. support collision detection and collision response for various object types.

Although they can perform real-time dynamics simulation, more advanced techniques are required to handle the simulation involving a massive number of bodies or penetration depth computation for nonconvex objects with high performance. Recently, combining the software solution with a high-performance hardware platform, such as multi-core CPU and GPU, in order to accelerate simulation performance has become a trend.[5,6]

Real-time deformable body simulation. Real-time deformable body simulation techniques require detecting self-collision and collisions between deformable objects such as cloth, skin or fractured objects.

Cloth simulation techniques have been extensively studied in the past, but a few research challenges still remain. Popular commercial software products for cloth simulation such as SyFlex, Maya's nCloth, Hovok Cloth and Qualoth are mainly used for making digital movie content. However, these products have not been intensively used yet in games which require real-time performance. Moreover, existing commercial cloth simulation software products do not fully support cloth tearing. Although nCloth supports cloth tearing, the performance is not acceptable for computer games. NVIDIA PhysX also supports cloth tearing with some limitations. Many researchers have been working on cloth simulation to maximize realism.[1]

For skin deformation, HAVOK, Unreal and CryEngine provide modules of deformation object simulation by applying an efficient deflector-based collision response to the objects. However, few companies use real-time physics-based skinning technique in computer games. For example, the game "FIGHT NIGHT: ROUND 4" by EA sports using NVIDIA PhysX supported skin deformation simulation under limited conditions.[1]

Data-driven physics simulation. Data-driven physics simulation techniques are used to build controllers that reproduce various motions of articulated character model(s) in a physically realistic manner using motion database. These techniques can be used to create e-entertainment contents which support physically-based interactions between the character model and various environmental objects using carefully controlled joint motions.

Physics simulation softwares such as PhysX or HAVOK support physics simulation for articulated bodies but these are only passive simulations. However, the technology which controls a character model in a stable way is necessary to be able to generate active motions such as walking, running or swinging. Yet this technology is still in its infancy. In particular, controlling an articulated body is difficult due to the high dimensionality and nonlinearity of the problem, and it is often an underactuated system.

For these reasons, developing a controller for a character model while keeping its balance in a physically simulated virtual environment generally consumes much computational time and efforts. Hence, no one in the computer game industry has used these complex techniques, but these techniques have been used to test control algorithms on some humanoids such as ASIMO (Honda), QRIO (Sony) and

HUBO (KAIST). Partly related, the technique of controlling an upper body of the robot using motion capture data was developed at Carnegie Mellon University, and Endolphine from Natural Motion supports reproducing simple but physically realistic upper body motions in real-time.[1]

Real-time simulation technology in Korea. A few Korean gaming companies have taken a leading role in online gaming which combines the software solutions and a high-performance hardware platform such as multi-core CPU and GPU. However, only a handful of companies have successfully used physics simulation technology in their gaming titles mostly using third party physics engines such as PhysX.[7] Although some gaming companies developed customized in-house physics engines, these engines can only perform a limited degree of dynamics, which is far from full-physics simulation. In the domain of CAE, a simulation software package for dynamic analysis exists such as RecurDyn,[a] yet it may not be applied directly to game engines which require real-time performance.

Real-time physics simulation requires advanced technology to fully optimize both software and a modern hardware platform, such as multi-core CPU or GPU, however only few research groups in Korea have developed these techniques.

3. Project Overview

As previously explained, our real-time physics engine consists of three functional components: rigid body dynamics simulation, deformable body simulation and data-driven physics simulation. In this section, we highlight several technologies to be used in these simulation engine components. Our research has been carried out in five steps. Table 1 shows an overview of our research.

For the rigid body dynamics simulation component, we focus on rigid body and articulated body dynamics simulation including various collision responses, parallel algorithms for collision detection, and the method of simulation level-of-detail (SLOD). In a similar way, the technologies for deformable body simulation include real-time cloth simulation techniques, fracture simulation techniques for deformable model and parallel algorithms for cloth simulation or fracturing simulation.

Data-driven physics simulation technology is considered more advanced than rigid body (articulated body) dynamics simulation technology because the former technology requires the latter. The data-driven physics simulation component can reproduce physically-based motions using motion capture data from different types of objects. We blend motion capture data with physically-based simulation and generate dynamics controller for an articulated body using motion capture data.

These technologies need to run real-time physics simulation at the rates of higher than 30 FPS (Frames per Second). We give an overview of each component of our real-time physics simulation engine.

[a]http://www.functionbay.co.kr.

Table 1. Overview of the technologies for real-time physics simulation in our research.

Technology	Step 1	Step 2	Step 3	Step 4	Step 5
Collision detection (rigid & articulated)	Rigid body collision detection	Articulated body collision detection	Parallel collision detection	Massive-body collision detection	Rigid body approximation
Collision response (rigid & articulated)	Rigid body dynamics simulation	Articulated body dynamics simulation	Coupling dynamics simulations	Simulation level-of-detail	Integrated physics simulation
Collision detection (deformable)	Cloth collision detection	Parallel algorithm for cloth collision detection	Fracturing model collision detection	Hybrid parallel collision detection	High-performance cloth simulation
Collision response (deformable)	Skin character modeling	Deformable mesh simulation	Volume (element) mesh generation	Physically-based fracture simulation	Viscous, elastic and plastic model simulation
Data-driven simulation	3D character motion correction	Character motion balance control	Dynamic motion controller	Physically-based motion editing	Flying character simulation and control

3.1. *Real-time rigid body dynamics simulation*

Real-time dynamics simulation technology deals with how to simulate realistically collision responses such as collision, resting, sliding for rigid bodies and/or articulated bodies in real-time. As shown in Table 1, the rigid body dynamics simulation can be divided into two parts: the collision detection part[8–10] and the collision response part.

In the collision detection part, we focus on rigid body collision detection, articulated body collision detection, parallel collision detection, multi-body collision detection and rigid body approximation. In real-time simulation, it is important to detect collisions and compute penetration depth at interactive rates for various types of objects. Therefore, our collision detection techniques deal with rigid body and/or articulated body, including nonconvex models and generalized triangular meshes. For accelerating collision detection performance, parallel algorithms are commonly used. In our research, multi-core CPUs and/or GPUs are utilized for parallel collision detection. Generally, physics simulation for complex models requires a huge number of computations. Since physics simulations for e-entertainment such as computer games requires real-time performance, we propose model or simulation approximation.

In the collision response part, we focus on rigid body simulation, articulated body collision simulation, coupled rigid/deformable simulations, simulation

level-of-detail and an integrated physics engine. The techniques for dynamics simulation (collision response) for rigid and articulated bodies should be able to handle various types of collision responses such as colliding contact, resting contact, friction and also deal with various types of joints for articulated bodies. Moreover, coupling between different types of simulations for rigid body, articulated body and deformable objects to simulate collision responses between them. Just like the level-of-detail technique is useful for real-time rendering, the SLOD algorithm makes dynamics simulation for complex models such as articulated body perform efficiently in real-time. Finally, a physics engine integrating different simulation media is necessary to support various types of physics simulations.

3.2. *Real-time deformable body simulation*

Like rigid body dynamics simulation, real-time deformable body simulation can also be divided into collision detection and response.

The collision detection part mainly includes techniques for cloth or fracturing models and parallel collision detection techniques. The cloth collision detection method includes collision detections between cloth models and within a cloth model using continuous collision detection (CCD).[11,12] The cloth simulation also supports cloth tearing. Generally, the collision detection module is considered as a bottleneck in cloth simulation. An efficient update of bounding volume hierarchy (BVH) for models is an important technique to increase the collision detection performance. Thus, we developed BVH updating techniques for both cloth and fracturing models. To accelerate collision detection performances, we also developed parallel algorithms using multi-core CPUs and/or GPUs for deformable models. Some other efficient techniques, such as an adaptive cloth simulation and an effective update of data structure, can be applied for high-performance cloth simulation.

In collision response, we mainly work on skin character modeling, real-time deformable model simulation, volume mesh generation, fracturing model simulation and viscosity/elasticity/plasticity model simulation. The skeleton-based skin character and its deformation can be used for character animation. A skin character should be naturally deformed without including distortion especially along joint areas. We also deal with real-time physically-based simulation for various types of deformable models such as solid, shell and rod using the continuum dynamics and the finite element method (FEM). We also developed a technique for generating a finite element mesh from a dense mesh.

Our physics engine realistically simulates various physical phenomena after extreme deformation of a deformable mesh such as fracture. It also supports various material properties such as viscosity, elasticity and plasticity.

3.3. *Data-driven physics simulation*

In this component, we aim to simulate physically-based motions of an articulated character using motion capture data.[13,14] In detail, we focus on techniques for

motion data correction of a 3D physical model, keeping the balance of the articulated character, physically-based simulation by real-time motion modulation, editing dynamic motion of the character, and physically-based simulation of a flying character and its motion control. As a result, we can physically realistically simulate the quadruped character, the biped character and the flying character.

We modify the motion capture data of a quadruped character for physical reproduction of the character under various ground conditions. Our technique allows a biped character to keep its balance while physically realistically reproducing its motions from motion capture data. We developed a controller which automatically simulates biped locomotion stably using real-time motion data modulation. This technique can control walking motions while steering its direction interactively and keeping its balance against external forces.

In addition, the human motion editing technique allows the user to edit dynamic motions interactively by controlling the momentum and external force of the simulated character. Thus, this editing technique can generate various dynamic human motions while earlier techniques can only replay the character motions that are stored in motion database. Finally, we researched physically-based 3D flying character simulation and flight motion control.

4. Main Engine Features

In this section, we provide the details of the main features and demonstrate the performances of each component of our real-time physics simulation engine. All the features in each component have been integrated into our real-time dynamics simulation engine, VirtualPhysics.[b]

4.1. *Real-time rigid body dynamics simulation*

Collision detection. Here, we developed CCD software for rigid and articulated body, a parallel algorithm using multi-core CPUs, a parallel algorithm using GPUs and a hybrid volumetric approximation method.

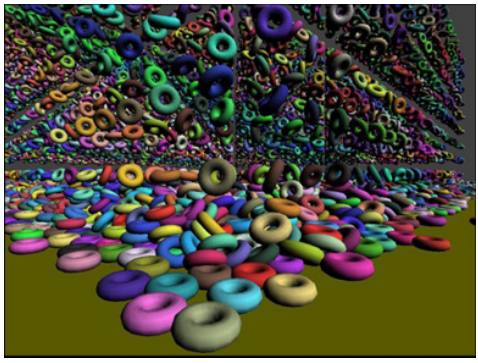
Our CCD software supports nonconvex rigid bodies. We developed a high-performance CCD software, called C^2A (controlled conservative advancement)[15] [see Fig. 2(a)], which supports polygon soup models. Based on C^2A, we also developed a fast penetration depth library, PolyDepth[c] that requires only 12 ms to compute the penetration depth for a polygon-soup model consisting of 152 K triangles.

The CCD for articulated body handles articulated bodies with nonconvex triangle meshes using Taylor models and temporal culling (CATCH).[16] We also expanded C^2A by applying C^2A to the links of articulated bodies using this method [see Fig. 2(b)]. As a result, we achieved up to 300 FPS collision detection performance

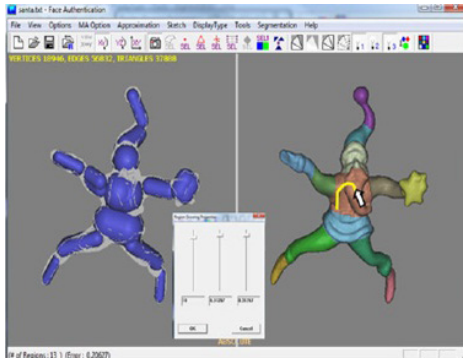
[b]http://virtualphysics.imrc.kist.re.kr.
[c]http://graphics.ewha.ac.kr/polydepth.

(a) C^2A

(b) CATCH



(c) gSaP



(d) Hybrid volumetric approximation

Fig. 2. Real-time rigid body dynamics simulation with high-performance collision detection techniques.

for an articulated body consisting of 20 K triangles of the articulated body with 12 links and the chess environment consisting of 100 K triangles.

Parallel approaches using CPUs and/or GPUs to accelerate collision detection performance have recently become an active research area. Our parallel algorithm uses multi-core CPUs for fast collision detection and distance computation between rigid models. The algorithm calculates penetration depth between oriented bounding box (OBB) for collision detection and approximated distance swept sphere volumes (SSV) for distance computation in parallel ways.[17] As a result, we obtained up to 240 FPS collision detection performance and up to 1000 FPS distance computation performance for a mesh with 69 K triangles using 8 concurrent threads. Our algorithm achieves up to 5 and 9.7 times improvement using eight core CPUs compared to a single core CPU for collision detection and distance calculation, respectively.

Our GPU-based parallel algorithm can handle collision detection for a million bodies in real-time using a variant of SaP (sweep and prune) collision culling

method. This algorithm uses workload balancing, principal component analysis (PCA) and spatial subdivisions.[18] The software library is called gSaP.[d] Using gSap, we achieved 161 ms/frame collision detection performance and 18 ms/frame simulation performance for up to 1M arbitrarily moving boxes and 16 K torus models, respectively [see Fig. 2(c)].

To accelerate proximity queries for physically-based animation, we simplify rigid objects. The hybrid volumetric approximation method approximates rigid triangular meshes tightly using SSV in both automatic and sketch-based way [see Fig. 2(d)].[19] Our approximated volume provides a visually reasonable result for rigid body dynamics.

Collision response. We developed rigid body and articulated body dynamics solvers, hybrid dynamics, coupling dynamics simulations, SLOD technique, a parallel algorithm for dynamics simulation and an integrated physics engine.

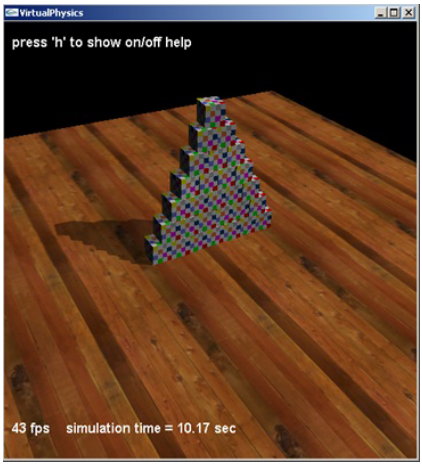
The solvers deal with collision response between rigid bodies or between articulated bodies. The rigid body dynamic solver provides numerically stable dynamics simulation and also supports spring-damper simulation and user-defined force fields such as wind as shown in Fig. 3(a). The articulated body dynamics solver supports various types of joints with constraints such as revolute joint, sliding joint and elastic spring [see Fig. 3(b)]. It provides a numerically stable, articulated body dynamics simulation by applying integral algorithms such as the implicit Euler integrator. This solver is effective since computing performance is proportional to the total number of degree of freedom (DOF) in an articulated body.

The hybrid dynamics of forward and inverse dynamics is used to meet diverse demands for passive and active dynamics simulation. Our simulation library couples the dynamics simulations of rigid body, articulated body and deformable body using an implicit data structure to optimize the performance. As a result, we obtained up to 60 FPS (real-time) simulation performance for up to 1 K rigid bodies, 1 K degree of freedom (DOF) articulated body and 10 K DOF deformable body.

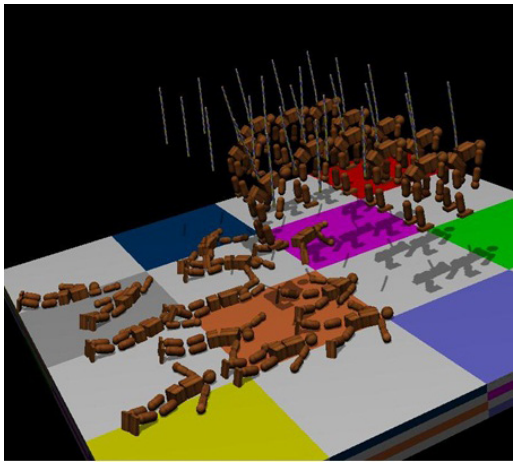
The SLOD technique activates or deactivates each joint of an articulator body based on the joint velocity. This technique minimizes the active DOF of the system by rigidifying the joints with a small movement. In our simulation test consisting of 60 DOF articulated body system, we were able to increase the simulation performance up to 69% using the SLOD algorithm [see Fig. 3(c)]. In addition, we also used a parallel algorithm for dynamics simulation of rigid body and articulated body to accelerate the simulation based on a lookup table of all possible colliding pairs of rigid bodies and articulated bodies in parallel way.

Finally, the integrated physics engine simulates seamlessly the rigid body, articulated body and deformable body. The engine also uses various collision detection algorithms selectively, and adaptively simulates according to user-based interactive

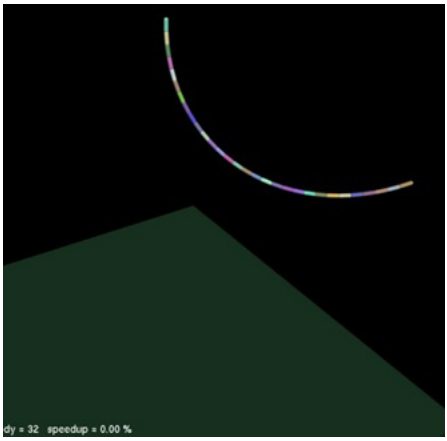
[d]http://graphics.ewha.ac.kr/gSaP.



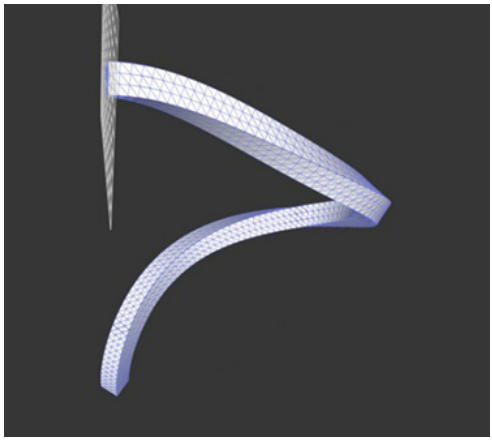
(a) Rigid body stacking simulation



(b) Articulated body simulation



(c) SLOD



(d) Simulation coupling

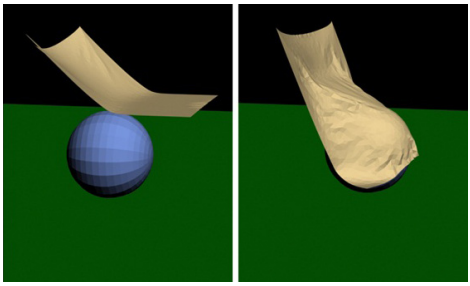
Fig. 3. Real-time rigid body dynamics simulation.

input or environmental conditions [see Fig. 3(d)]. In the test, we obtained computing performance proportional to the total number of cores in the parallel computer.

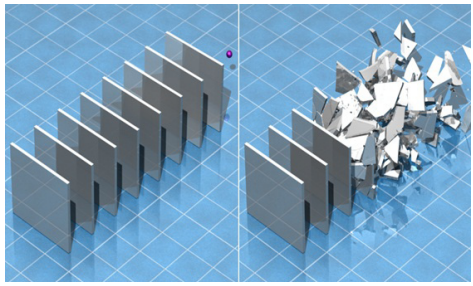
4.2. *Real-time deformable body simulation*

Collision detection. We developed a cloth collision detection technique, parallel algorithm using multi-core CPUs, fracturing model collision detection algorithm, hybrid parallel algorithm using multi-core CPUs and GPU and high-performance cloth simulation technique.

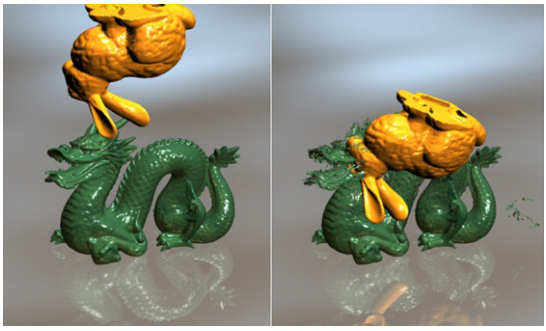
The cloth collision detection algorithm includes self-collision detection, and performs BVH-based CCD. It also applies representative triangles to avoid duplicate



(a) Cloth simulation



(b) Fracturing model simulation



(c) Fracturing model simulation using the hybrid parallel algorithm

(d) Adaptive cloth simulation

Fig. 4. Real-time deformable body simulation with fast collision detection techniques.

computation [see Fig. 4(a)].[20] In the test, we obtained up to 200 FPS collision detection performance between cloths consisting of up to 3 K triangles, and also achieved up to 60 FPS for self-collision detection performance. We applied the selective reconstruction technique and the parallel algorithm using multi-core CPUs to the cloth collision detection algorithm. Then, we obtained up to 40 FPS performance for collision detection between cloths and within a cloth consisting of up to 4 K triangles.

Our efficient collision detection method for fracturing models improved the collision detection performance by updating BVH selectively on fractured parts.[21] This method also includes an efficient culling algorithm for fracturing models. As a result, our method takes less than 260 ms for CCD for a fracturing mesh consisting of up to 200 K vertices and less than 100 ms for discrete CD for a fracturing mesh consisting of up to 100 K vertices of the fracturing mesh [see Fig. 4(b)].

Our hybrid parallel algorithm utilizes multi-core CPUs and GPU for collision detection between fracturing models consisting of tens or hundreds of thousands triangles. The algorithm includes the effective distribution of workload between multi-core CPUs and GPU and supports various deforming models and self-collision detection. We achieved up to six times (for 17 K vertices) and eight times (for 192 K

vertices) improvement using four CPU threads and one GPU compared to one CPU thread [see Fig. 4(c)].

Finally, the high-performance cloth simulation uses an adaptive cloth simulation algorithm [see Fig. 4(d)] and an efficient update of a data structure. The adaptive cloth simulation algorithm uses multi-resolutions adaptively for fast and stable physics simulation. As a result, we obtained up to 50 FPS collision detection performance for up to 8 K vertices using 4-core CPUs and GPU (GTX480). This is about five times performance improvement compared to a single CPU core. We also achieved up to 5 FPS of simulation performance for fracturing models with 190 K vertices.

Collision response. We developed a skin character simulation technique, deformable model simulation technique, a finite element mesh (volume) generation, a simulation technique for fracture or fragmentation of meshes and for deformable mesh with viscosity, elasticity or plasticity.

Our skin model simulation method is based on a linear blending algorithm using motion, skeleton hierarchy, skin mesh vertices and the weights. The method simulates a skin character without distortion in the joint area. In our simulation test, we obtained up to 250 FPS of simulation performance for a skinning model with 4400 vertices.

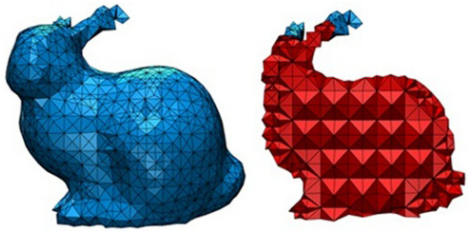
In real-time deformable model simulation, we can control the material properties of deformable models such as the shell, rod and solid by modeling dynamic deformation of elasticity using the continuum dynamics (and FEM for solid deformable mesh) [see Fig. 5(a)]. We achieved up to 100 FPS simulation performance for a deformable mesh consisting of up to 10 K vertices.

We developed an algorithm to create a finite element volume mesh from a dense mesh [see Fig. 5(b)]. To generate the finite element mesh from a dense mesh, we first compute the signed distance field (SDF) of the body-centered cubic (BCC) lattice using the fast marching method. Then, an octree-based adaptive finite element mesh is obtained using the SDF. This method generates a high-quality element mesh with good dihedral angles. In the test, it takes less than 3 seconds for generating and updating the element mesh consisting of up to 250 K tetrahedra.

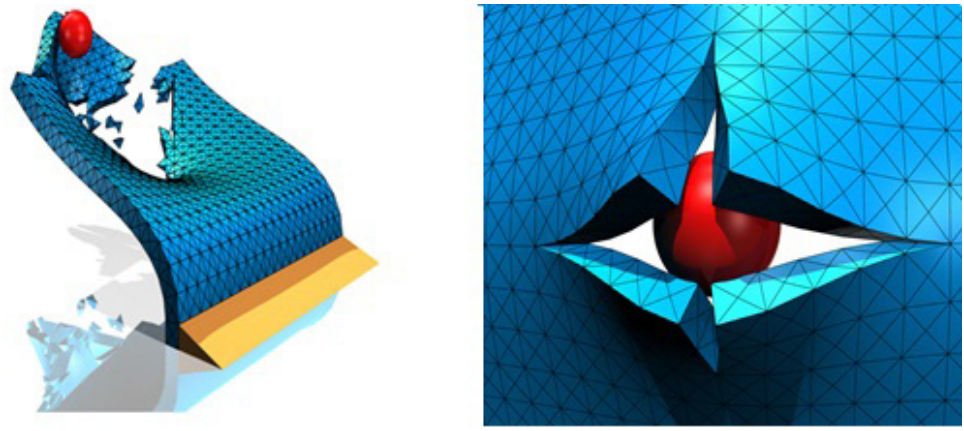
The simulation technique for mesh fracture or fragmentation handles collision detections and collision responses between broken pieces at fracturing or fragmenting events [see Fig. 5(c)]. The algorithm patternizes the fracture and fragmentation phenomena and reconstructs the element mesh locally.

To simulate viscous, elastic or plastic deformable mesh, we deal with negative volume for stabilizing simulation and use both a less dense finite element mesh and a highly dense surface mesh. The algorithm uses the inversion of penetration depth and the reproduction of modified gradient to handle a negative volume to stabilize the simulation. We also develop a method to map the vertices from a dense surface mesh to those on the spare element mesh which enables deforming the dense surface mesh quickly by deforming the coarse finite element mesh.

(a) Deformable model simulation



(b) Finite element mesh generation



(c) Simulation of deformable mesh with various material properties

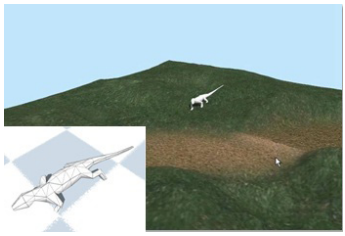
Fig. 5. Real-time deformable body simulation.

4.3. *Data-driven physics simulation*

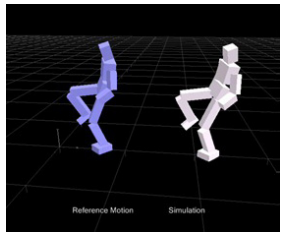
In data-driven physics simulation, we developed a motion data correction method for physically-based locomotion reproduction of a quadruped character, a technique for keeping the balance of a 3D biped character, a dynamic controller of motion reproduction, a dynamic motion editor and a bird flight simulation technology.

The 3D iguana character reproduces physically-based locomotion of a character from the motion capture data as shown in Fig. 6(a). We can physically simulate the character in a virtual environment under various ground conditions. The directional change of a moving character and various locomotions can also be physically simulated.

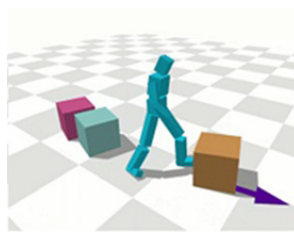
We can also physically reproduce a realistic locomotion of a dynamic biped character [see Fig. 6(b)] from motion capture data. The balance of the character is maintained by controlling its angular momentum and the linear momentum. Our biped character can reproduce various human gaits for a longer period than



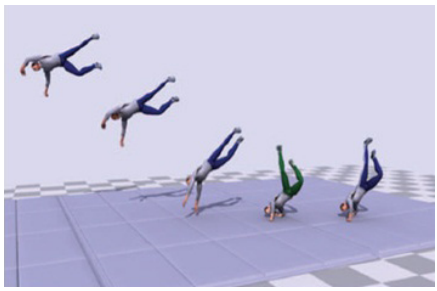
(a) Simulation of the iguana's locomotion



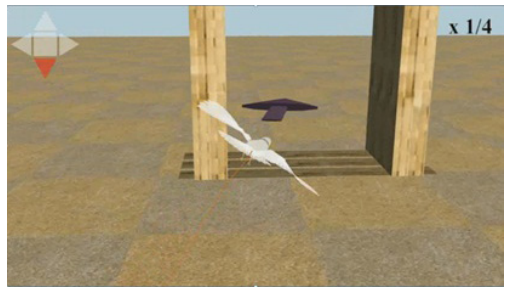
(b) Reproduction of biped model's kicking



(c) Interactive dynamic model control



(d) The dynamic motion editor



(e) Bird flight simulation

Fig. 6. Real-time data-driven physics simulation.

the original locomotion in the database. In our test, we obtained eight different types of dynamic locomotion reproductions lasting up to five times longer than the original data. Our human character can also reproduce a kicking motion at a standstill while keeping its balance as shown in Fig. 6(b).

The dynamic controller allows the character to reproduce motions stably by modulating motion data in real-time.[22] The balance feedback method and the technique for synchronizing reference motions and character contact state are used for real-time motion data modulation. We can also simulate dynamic human walking by spinning or keeping his balance against an external force using the controller [see Fig. 6(c)]. Other results include keeping balance under various conditions of simulation, interactively-controlled biped navigation and stable reproduction of various human gaits such as marching, U-turn, walking back, etc. Compared to previous data-driven controllers, our controller does not require any pre-computed control model or nonlinear optimization. Therefore, it is computationally efficient and easy to implement.

We can edit existing dynamic motions without breaking physical plausibility by controlling the momentum and external force.[23] The editor includes the functions for analyzing the momentum information of motion data, editing a sequence of dynamic motions in a more physically plausible way, and the complex motion editing function [see Fig. 6(d)].

Not only did we apply the data-driven physics simulation technology to quadrupeds and bipeds, we also applied it to a flying character. We modeled the physical characteristics of a flying character using marker-based bird flight motion capture data and combined the aerodynamic model[24] with the simulation model.[25] By doing so, we were able to perform realistic feather simulation. Our control algorithm repeatedly extracts state-action pairs by analyzing the motion capture data statistically. Then the flying character can be controlled using the state-action pairs in almost real-time [see Fig. 6(e)]. Previous approaches are based on optimization to control the flying character which takes more time to compute.

4.4. *Software and applications*

Many techniques we developed in our research project are open-sourced to the public, and the source codes have been released in the form of a software library. For example, FAST[e] (interactive CCD for nonconvex polyhedral), CATCH[f] (CCD for articulated models using Taylor models and temporal culling), C^{2}A[g] (controlled conservative advancement), gSaP (GPU-based SaP), OpenCCD[h] (CCD API) and VirtualPhysics are all currently available on their respective web sites. VirtualPhysics provides an integrated library for simulation techniques that we have developed in this research project [see Fig. 7].

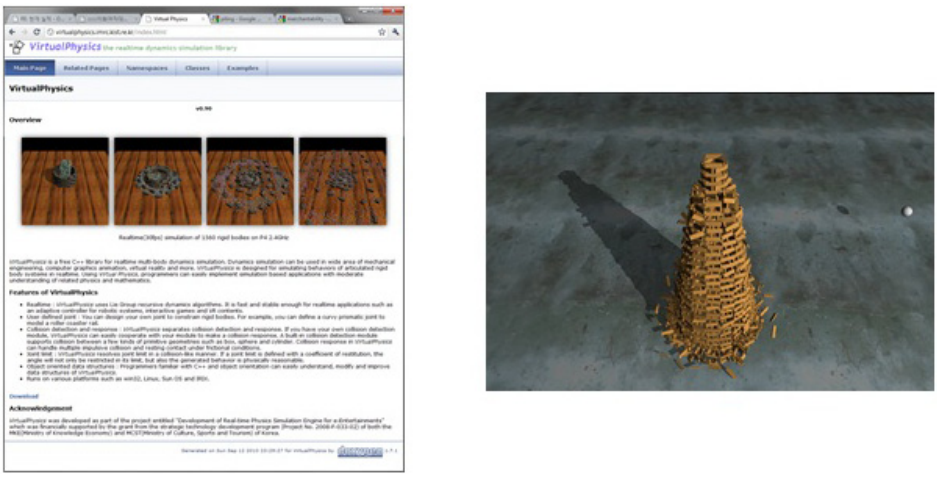
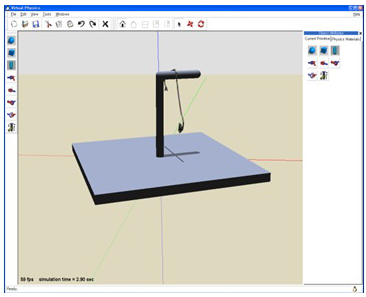
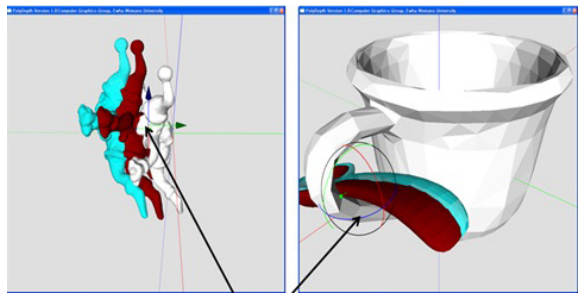


Fig. 7. The real-time physics simulation engine, VirtualPhysics. The homepage of the real-time physics software, VirtualPhysics (left) and an example of physics simulation using VirtualPhysics (right).

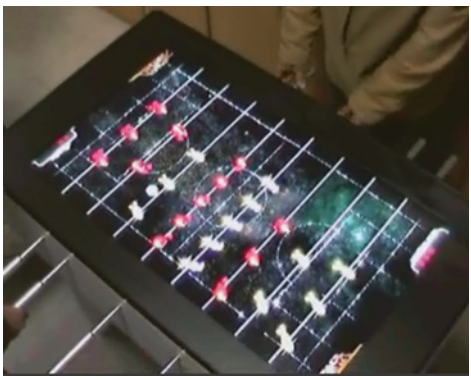
[e]http://graphics.ewha.ac.kr/FAST.
[f]http://graphics.ewha.ac.kr/CATCH.
[g]http://graphics.ewha.ac.kr/C2A.
[h]http://sglab.kaist.ac.kr/OpenCCD.



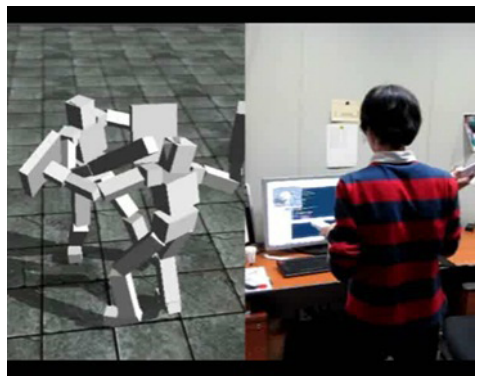
(a) The physics engine editor



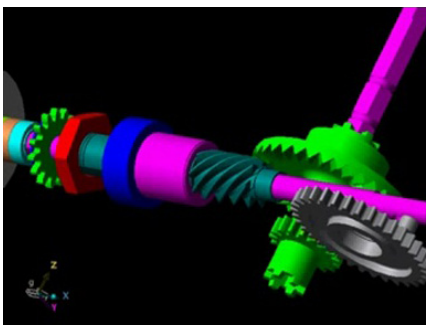
(b) The test environment of the collision software framework



(c) Space Foosball



(d) Interactive Fighter



(e) RecurDyn

(f) Monster Chaos

Fig. 8. Our software prototypes.

We also developed application softwares to benchmark our real-time physics engine. Figure 8 shows such examples. These prototypes are based on VirtualPhysics. Currently, VirtualPhysics supports diverse OS environments including Windows, Linux and iOS. The physics engine editor [see Fig. 8(a)] allows users to edit simulation conditions and verify the simulation results. The collision software

framework [see Fig. 8(b)] is a framework for our integrated collision algorithms that are meant to be easily used for game engines or simulation software. Space Foosball [see Fig. 8(c)] allows users to experience the foosball game in virtual space through a tangible interface similar to actual foosball.[26] Interactive Fighter [see Fig. 8(d)] is a game platform prototype using VirtualPhysics and the motion sensor Wiimote for virtual fighting with swords based on data-drive character simulation. RecurDyn [see Fig. 8(e)], which is the FunctionBay's multi-body dynamics engine for CAE (computer aided engineering), is integrated with our parallel collision technology for higher performance dynamics simulation. Monster Chaos [see Fig. 8(f)] is a 3D physical game developed for the iPhone and iPad on Apple iOS. This game allows a user to allocate defense towers in real-time to protect the base from enemies.

5. Conclusion

In this paper, we have highlighted the main features and performances of our real-time physics simulation engine developed for e-entertainment. Our real-time physics engine consists of three functional components: rigid body dynamics simulation, deformable body simulation and data-driven physics simulation.

The main technologies for rigid body dynamics simulation and deformable body simulation components include collision detection and collision response between rigid bodies, articulated bodies and/or fracturing deformable bodies for high-performance simulation. Moreover, parallel algorithms are utilized for multi-core CPUs and/or GPUs to increase the simulation performance. Since our goal is to develop a real-time physics engine for e-entertainment such as computer games, we focused on performance rather than accuracy. However, our results show that our software enables high-performance physics simulation as well as provides visually reasonable results.

Using the data-driven physics simulation, we physically realistically simulate a quadruped character, a biped character and a flying character to reproduce their dynamic motions (locomotion simulation or bird flight simulation) based on motion capture data. This can be achieved stably under various environmental conditions. The biped character in particular can reproduce various dynamic human walking while keeping his balance against an external force using the dynamic controller. Also, existing dynamic motions of the biped character can be edited with physical plausibility using the dynamic motion editor.

Most of these simulation features are integrated into our real-time dynamics simulation engine, VirtualPhysics, which is constantly being upgraded with new features of real-time simulation. This powerful simulation engine will provide various features for developing more fascinating e-entertainment contents.

Acknowledgments

This work was supported in part by IT R&D program of MKE/MCST/KOCCA (KI001818) and NRF grant funded by the Korea government (MEST)

(No. 2012R1A2A2A01046246, No. 2012R1A2A2A06047007). This research project was jointly conducted with Seoul National University, Korea advanced institute of science and technology, Korea institute of science and technology, and Kwangwoon university.

References

1. Annual Report of Korean Innovation Research Program 2009–2011: Development of Real-time Physics Simulation Engine for e-Entertainment, Ministry of Knowledge Economy (MKE), 2010–2012.
2. DeLoura M., Middleware showdown, Game Developer Magazine, August 2009.
3. Ericson C., *Real-time Collision Detection*, Morgan Kaufmann Publishers, 2010.
4. Bergen G. V. D., Gregorius D., *Game Physics Pearls*, A K Peters Ltd., 2010.
5. Huagen W., Zhaowei F., Shuming G., Qunsheng P., A parallel collision detection algorithm based on hybrid bounding volume hierarchy, *CAD/Graphics: 7th Int. Conf. CAD and Computer Graphics*, pp. 521–528, 2001.
6. Zhou K., Hou Q., Wang R., Guo B., Real-time KD-tree construction on graphics hardware, *ACM Trans. Graph.*, **27**(5):126, 2008
7. 2010 Korean Game White Paper, *Korea Creative Content Agency* (*KOCCA*), 2010.
8. Gottschalk S., Lin M., Manocha D., OBB-Tree: A hierarchical structure for rapid interference detection, *SIGGRAPH96: Proc. of the 23rd Annual Conf. Computer Graphics and Interactive Techniques*, pp. 171–180, 1996.
9. Lin M., Manocha D., Collision and proximity queries, *Handbook of Discrete and Computational Geometry*, pp. 787–808, 2004.
10. Yoon S., Salomon B., Lin M., Manocha D., Fast collision detection between massive models using dynamic simplification, *SGP04: Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing*, pp. 136–146, 2004.
11. Tang M., Manocha D., Tong R., Multi-core collision detection between deformable models, *SPM09*: *SIAM/ACMJoint Conf. Geometric and Physical Modeling*, pp. 355–360, 2009.
12. Kim D, Heo J. P., Huh J, Kim J, Yoon S. E., HPCCD: Hybrid parallel continuous collision detection, *Comput Graph Forum* (*Pacific Graphics*), **28**(7):1791–1800, 2009.
13. Yin K., Loken K., Van De Panne M., Simbicon: Simple biped locomotion control, *ACM Trans. Graph.* (*SIGGRAPH 2007*), **26**(3):105, 2007.
14. Coros S., Beaudoin P., Van De Panne M., Robust task-based control policies for physics-based characters, *ACM Trans. Graph.* (*SIGGRAPH Asia*), **28**(5):170, 2009.
15. Tang M., Kim Y. J., Manocha D., C2A: Controlled conservative advancement for continuous collision detection of polygonal models, *ICRA09: Proc. IEEE Int. Conf. Robotics and Automation*, pp. 356–361, 2009.
16. Zhang X., Redon S., Lee M., Kim Y. J., Continuous collision detection for articulated models using Taylor models and temporal culling, *ACM Trans. Graph.*, **26**(3):15, 2007.
17. Lee Y., Kim Y. J., Simple and parallel proximity algorithms for general polygonal models, *J. Comput. Animation and Virtual Worlds* (*CASA special issue*), **21**(3–4): 365–374, 2010.
18. Liu F., Harada T., Lee Y., Kim Y. J., Real-time collision culling of a million bodies on graphics processing units, *ACM Trans. Graph.* (*SIGGRAPH ASIA*), **29**(6):154, 2010.

19. Bae M., Kim J., Kim Y. J., User-guided volumetric approximation using swept sphere volumes for physically-based animation, *J. Comput. Animation and Virtual Worlds*, **23**(3–4):385–394, 2012.
20. Tang M., Curtis S., Yoon S., Manocha D., Interactive continuous collision detection between deformable models using connectivity-based culling, *SPM08: Proc. ACM Symp. Solid and Physical Modeling*, pp. 25–36, 2008.
21. Heo J., Seong J., Kim D., Otaduy M. A., Hong J., Tang M., Yoon S., FASTCD: Fracturing-aware stable collision detection, *SCA10: Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 149–158, 2010.
22. Lee Y., Kim S., Lee J., Data-driven biped control, *ACM Trans. Graph.* (*SIGGRAPH 2010*), **29**(4):129, 2010.
23. Sok K. W., Yamane K., Lee J., Hodgins J. K., Editing dynamic human motions via momentum and force, *SCA10: Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 11–20, 2010.
24. Wu J.-C., Popovic Z., Realistic modeling of bird flight animations, *ACM Trans. Graph.*, **22**(3):888–895, 2003.
25. Choi M. G., Woo S. Y., Ko H. S., Real-time simulation of thin shells, *Comput. Graph. Forum*, **26**(3):349–354, 2007.
26. Bang H., Heo Y., Kim J., Kim Y. J., Space Foosball: Coupling tangible interfaces with a real-time game physics engine, *VRIPHYS09: 6th Workshop on Virtual Reality Interactions and Physical Simulations*, pp. 49–58, 2009.