International Journal of Advanced Science and Technology
Vol.119 (2018), pp.145-152
http://dx.doi.org/10.14257/ijast.2018.119.13

The Hardware Accelerated Physics Engine with Operating Parameters Controlling the Numerical Error Tolerance

Youngsik Kim

Department of Game and Multimedia Engineering, Korea Polytechnic University
kys@kpu.ac.kr

Abstract

This paper designed the dedicated hardware pipeline for interactive 3D entertainment system considering parallelization characteristics of rigid body physics simulation. This paper also analyzed the large parallelization characteristics of independent rigid group units connected to joints in hierarchical collision detection and design a hardware structure suitable for each parallel processing stage by extracting small data parallelization characteristics of each constraint row unit of LCP equation. Also, this paper performed control such as the number of loop repetitions by performing hardware trade-off in which numerical error is reduced within visual error tolerance by setting error attenuation parameter (ERP) and constraint force mixing (CFM) as driving parameters.

Keywords: *Hardware Accelerator, Physics Engine, Open Dynamics Engine, Numerical Error Tolerance*

1. Introduction

The importance of physics engine for simulating the motion of an object about physical phenomena (mass, speed, friction, collision) of nature in the interactive 3D game, virtual reality, 3D animation and robotics industry is becoming more and more important [1-4]. In a car racing game, it is better to use a physics engine simulation than a keyframe animation created by an artist, as it slips by centrifugal force when cornering and collides head-on with other cars. It would be much more natural to model an automobile as a rigid body with mass, a moment of inertia, velocity, *etc*., and simulate the kinematic motion of the body due to collision or friction.

Physics-based rigid body simulations simulate the kinematics of an object with mass properties connected by a joint. When a rigid body is subjected to a force or a collision, the new velocity is solved by the linear equation of motion, and the new position of the object is calculated by integrating it repeatedly in time step by computer numerical analysis. In the dynamic world, it is necessary to perform mathematical operations in parallel with dedicated hardware pipelines to obtain collision processing of many bodies and solution of differential equations by iterative integration and real-time physics simulation. Such an iterative integral calculation inevitably causes a numerical error, and it is impossible to completely eliminate it. Therefore, in 3D entertainment application, it is essential to design a hardware parameter that can control the number of iteration step between accuracy and speed of calculation in order to control numerical error congestion while increasing animation frame rate within the visual tolerance limit.

Efforts to accelerate software/hardware acceleration of real-time physics-based simulation have recently begun, but many studies have not been conducted [1-11]

Received (May 3, 2018), Review Result (July 20, 2018), Accepted (July 21, 2018)

[13-19] [21]. Some of these efforts have been a parallel mapping of physics-based simulation algorithms to high-performance CMP (chip multiprocessor) systems [1-4], and efforts to map them to graphics processing units (GPUs) that process 3D graphics [5-11] there was. [15-17] [21] There are some attempts to construct real-time physics simulation by using dedicated hardware. Hardware resources are wasted to configure a customized system with a high-performance CMP system. Moreover, because the GPU is focused on accelerating rendering and display, it is difficult to map physically based simulations with a continuous feedback loop directly, and the interim results are interactively accessed in the GPU graphics pipeline It is not easy to do [3]. Therefore, the hardware accelerator for the real-time physics engine for the interactive 3D entertainment system should be configured separately from the GPU-based 3D graphics acceleration.

In order to design the real-time physics engine hardware accelerator structure, as shown in Figure 1, the simulation environment is constructed as a real application program using the physics engine API. Then, the parallelism of the core module is analyzed to find data parallel processing technique of mathematical operation, Numerical error, visual error tolerance, throughput and delay time should be reflected in the design.

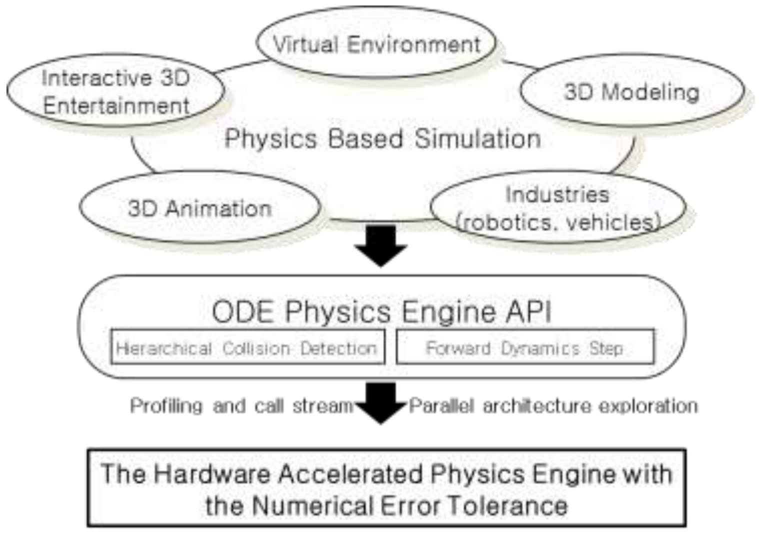
Virtual Environment
Interactive 3D Entertainment
3D Modeling
Physics Based Simulation
3D Animation
Industries (robotics, vehicles)
ODE Physics Engine API
Hierarchical Collision Detection
Forward Dynamics Step
Profiling and call stream
Parallel architecture exploration
The Hardware Accelerated Physics Engine with the Numerical Error Tolerance

Figure 1. The Hardware Accelerated Physics Engine

2. Related Works

This paper proposed a new algorithm for hardware collision detection, which is a core module of physics engine [16-17]. Collision detection using existing GPUs will continue to compete with the GPU's inherent rendering process, which will inevitably degrade the frame rate. In ODE, hierarchical collision detection is performed using AABB boundary box, which is basically easy to implement. In [16-17], collision detection is performed on a more efficient k-DOP basis, and collision detection is performed on the triangular mesh. In this study, k-DOP-based collision sorting is handled, but it will be processed in parallel in independent collision space. In addition to triangular meshes, various geometries (spheres, cylinders, boxes, capsules, triangular meshes, Planes) to handle data parallelism.

For the dynamics step forward module, ODE used Baraff's Dantzig LCP solver [18-19] algorithm to find the collision force satisfying the linear condition. [21] Ageia PhysX includes physics processing engine (PPU) for the physics engine, as well as CPU and GPU. In addition, the unit is separately provided. The internal structure of the PPU includes a PPU control engine (PCE), a data movement engine (DME), and an island processing engine (IPE). Particularly, it can be seen that the collision detection and the LCP equation solution are parallel-processed by providing several IPE blocks. This paper designed a pipeline structure to prevent numerical error congestion by processing data parallelism on the basis of Baraff 's LCP solver algorithm and controlling the number of iterations according to ERP / CFM drive parameters.

As shown in Figure 2, Thomas Y. Yeh *et al.*, who designed the ODE-based Parallax [1] architecture, reduced die size and speed by reducing the number of mantissa bits in a floating- [15], and furthermore, a floating-point computer with a reduced number of bits in a CMP system was shared [23]. Despite these trends, numerical error tolerance control hardware accelerated physics engine research is still lacking. In particular, hardware structure studies to control driving parameters such as collision detection large parallel processing and LCP solver data parallel processing pipeline repetition It was not done.

There have been efforts [7-11] to map real-time physics algorithms to GPUs that process common 3D graphics. GPU based particle simulation [7-8], matrix multiplication study [9], collision detection study [10-11]. GPU companies ATI / AMD [5] and nVIDIA [6] have added physics engine capabilities to the latest GPUs such as Radeon Vega and Volta. NVidia series supports Newton kinematics acceleration under the name of Quantum Effects Technology and provides the Compute Unified Device Architecture (CUDA ™) Physics Engine SDK for this. ATI / AMD similarly provides the CTM ™ (Close to Metal ™) physics engine SDK. The game industry is using the Havok FX physics engine SDK [12] in numerous games, including Half-Life 2 and Dead Rising. The Havok FX Physics Engine SDK provides the ability to parallelize multiple GPUs, such as ATI (CrossFire) or nVIDIA (SLI). Havok FX SDK has effects such as explosions Physics simulations are handled by the GPU like shader model 5.0, and gameplay physics simulations are handled by the CPU. In other words, many physics simulations are still being processed by the CPU.

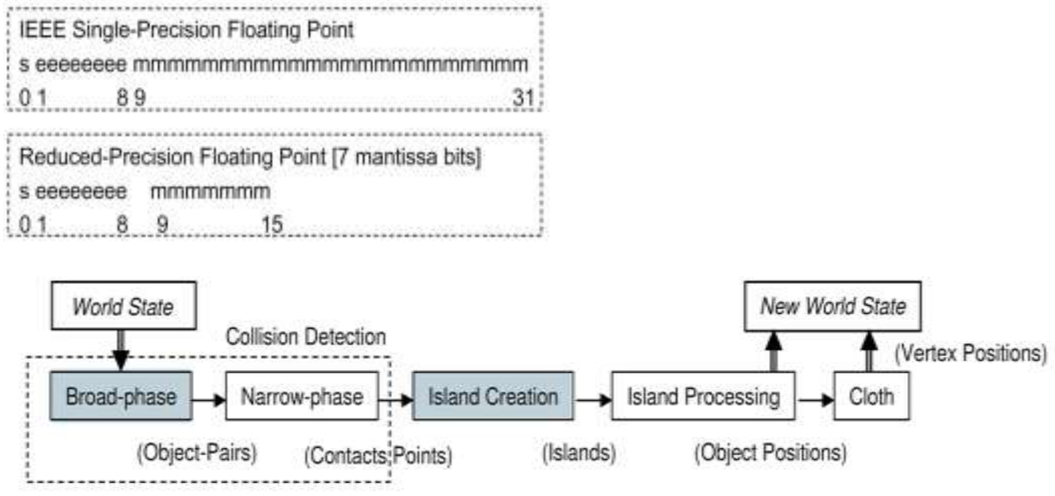

Figure 2. Numerical Error-tolerant Physics Engine to Reduce Bit Accuracy [15]

3. Design of Hardware Accelerated Physics Engine

3.1. Physics Engine based on Open Dynamics Engine

This paper designed a hardware accelerator for real-time processing of rigid-body physics simulation based on widely used and open source ODE (open dynamics engine) physics engine [22]. Figure 3 shows a series of operation flows that create a rigid body in the kinematics world, process collisions, and advance the time step.

The key modules of the ODE physics engine in the figure flow diagram are hierarchical collision detection and forward dynamics step. Hierarchical collision detection constructs an axis aligned bounding box (AABB) bounding box of a rigid body to detect collision. The dynamic step forward is obtained by repetitive integration of the solution of the LCP (linear complementary problem) equation according to the constraint conditions such as the radius of rotation of the joint, the friction coefficient, the buffer coefficient, and the applied force. This is used to calculate the final linear position and angular position of the body. In order to detect collision efficiently, a hierarchical conflict space and a bounding volume hierarchy (BVH) should be configured to search for a conflicted space and to cull a bounding volume that does not cause a collision. Then, we must solve the various errors that occur when we solve the LCP equation and perform the integration according to the time step. In other words, numerical errors in ODE algorithms for fast performance should not cause constraint violation. Examples of constraint violations include the passing of rigid bodies through other bodies, jumping without reason, being infinitely distant from one another, falling joints, or incorrectly bending the joint axes out of alignment. To reduce this error, an error reduction parameter (ERP) is implemented to correct every time step joint error. Also, in the LCP equation, the relationship between the constraint applied to the body and the force is expressed as constraint force mixing parameter to express the rigidity and softness of the body.

```
Operation Flow of ODE Physics Engine
1. Create a dynamics world.
2. Create bodies in the dynamics world.
3. Set the state (position and velocities) of all bodies.
4. Create the joints (constraints) that connect bodies.
5. Create a collision world and collision geometry objects.
6. While ($time < timemax$)
   (a) Apply force to rigid bodies as needed.
   (b) Adjust joint parameters as needed.
   (c) Hierarchical collision detection -
       . Performs collision space search and performs AABB collision selection.
       . Collision checking is performed for each pair of geometric pairs that are
         likely to collide, for all impact points, create a contact joint and add it to the
         contact joint group.
   (d) Forward dynamics step.
       . Constraining force Find the solution of the LCP equation and update the
         velocity of the body.
       . Integrals are performed to update the new line displacements and angular
         displacements of the body.
       . Increase one simulation step.
   (e) Remove all joints in the contact joint group.
   (f) Renders all geometry in the kinematics world in a new position and
       orientation.
7. Destroy kinetic world and collision world.
```

Figure 3. The Operation Flow of ODE Physics Engine

These ERP and CFM parameters reduce numerical errors, but they have a significant impact on performance time. The following equation 1 is ODE spring-buffer systems corresponding to ODE constraint equations and spring constants kp and damping constants kd, expressed as ERP and CFM. J is the Jacobian matrix, v is the velocity of the rigid body, c is the right vector, lambda is the restorative force required for constraint, and h is the step size.

$$J \times v = c + CFM \times lambda$$
$$ERP = h \times kp / (h \times kp + kd), CFM = 1/(h \times kp + kd) \quad (1)$$

3.2. Hardware Accelerated Physics Engine

This paper designed a dedicated hardware physics engine accelerator that efficiently controls numerical errors within visual error tolerance for the 3D entertainment system. To do this, we design a parallel hardware pipeline of hierarchical collision detection and dynamics step forward, which is a core module that takes a lot of execution time in the ODE physics engine. We develop a simulation module that operates in a cycle accurate and parallel pipeline for hardware exploration of hardware pipelines in two core modules. ODE We design a parallel pipeline architecture that optimizes the number of pipelines, the bottleneck section of the pipeline stage, and the step delay time by profiling the actual application and analyzing the parallelism. In ODE real application, hierarchical collision detection and dynamic step forward call stream are extracted and used as inputs to cycle accuracy simulation module.

The real-time physics engine hardware accelerator of this study is dedicated to physics simulation in addition to the CPU and GPU (Figure 4). Figure 5 shows a block diagram of the real-time physics engine hardware accelerator architecture proposed in this study and explains parameter extraction and hardware/software comparison verification method through application profiling.

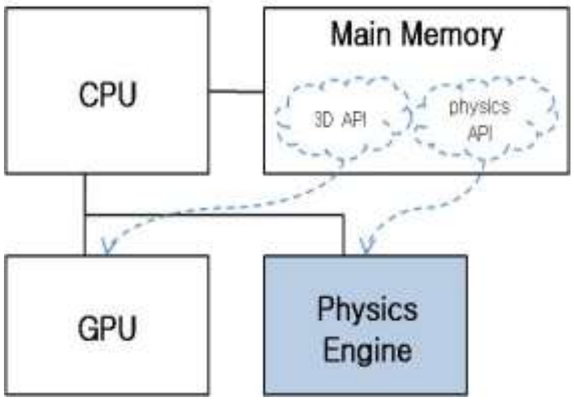

Figure 4. The 3D Entertainment System including the Proposed Real-Time Physics Engine Hardware Accelerator, CPU, and GPU

Figure 5 shows the ODE application execution flow (left), cycle accuracy simulation (center), and hardware parallel pipeline Verilog RTL (right) in three parts. Hierarchical collision detection and forward dynamics step grayed out in the ODE application execution flow are part of the hardware design. The ODE software engine extracts the call stream of the hierarchical collision detection module, processes it by hardware, and designs a structure that can receive and render the new position of the body. The hierarchical collision sensing part and the dynamic step advancing part that are to be hardware accelerated have a structure in which pipelines are provided in parallel, or the loop is repeatedly performed according to the parallelization characteristic. Such a parallel pipeline architecture can control operation with drive parameters such as ERP / CFM.

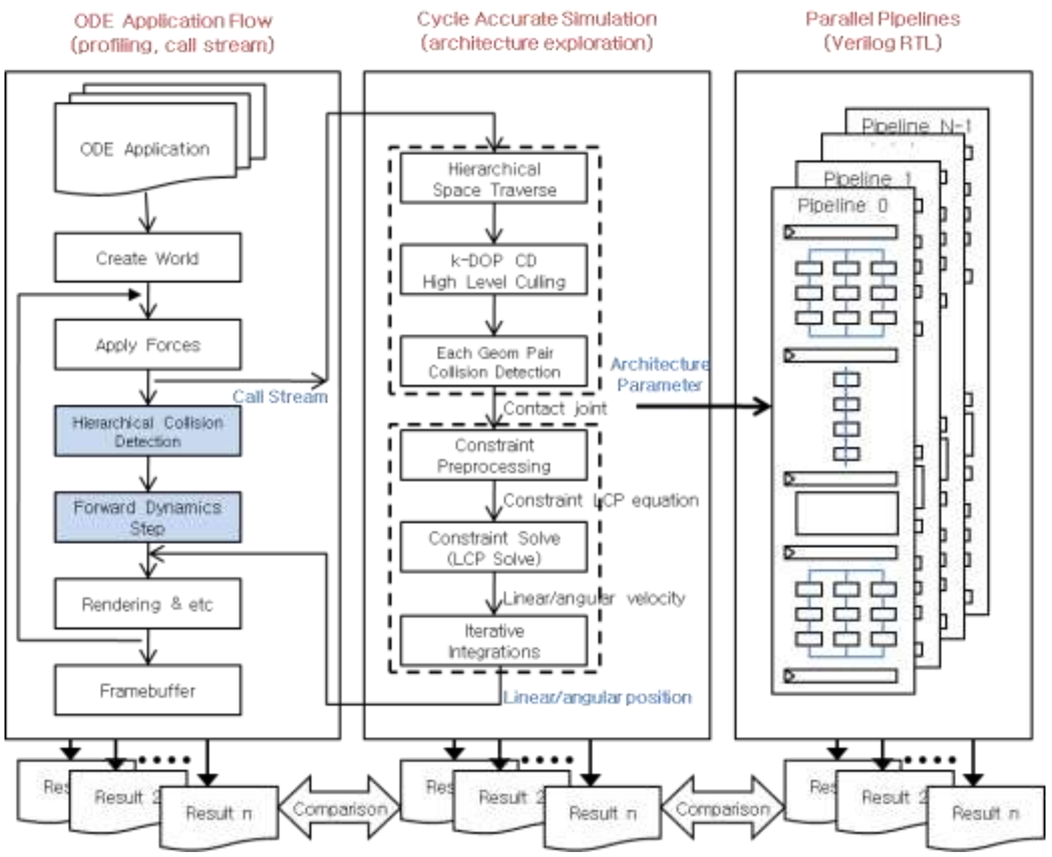

Figure 5. The Proposed Real-Time Physics Engine Hardware Pipeline Structure and Simulation Verification Environment

4. Performance Evaluation

As shown in Figure 6, the parallel pipeline structure is designed by profiling the ODE real application and analyzing the parallelism to optimize the number of pipelines, the bottleneck section of the pipeline stage, and the step delay time. Figure 6 is a sample of a simulation of simulating collision by rushing eight cars in a four-wheeled vehicle at the same time onto a brick wall. As shown in the figure, in the whole frame processing, the collision detection execution time takes a lot in the front part where the collision occurs, and the execution time of the dynamics step forward in the middle part increases. Overall, the average collision detection time and dynamics step forward time are 26.2% and 24.9%, respectively.

Setting the constraint force mixing (CFM) parameter (0 to 1) to a large value increases the softness of the rigid body and reduces the numerical error of the simulation, but increases the softness and can violate the non-penetrating condition. It is helpful to increase the number of iterations per step to reduce the numerical error, but it also slows down the execution speed. In this study, a driving control algorithm based on ERP / CFM driving parameters that maximizes the execution speed and allows visual error be designed and applied to the pipeline. In order to efficiently interface between the proposed physical engine hardware accelerator and CPU / GPU, this paper designed the DMA connection list type call stream transfer and FIFO queue data transfer structure.

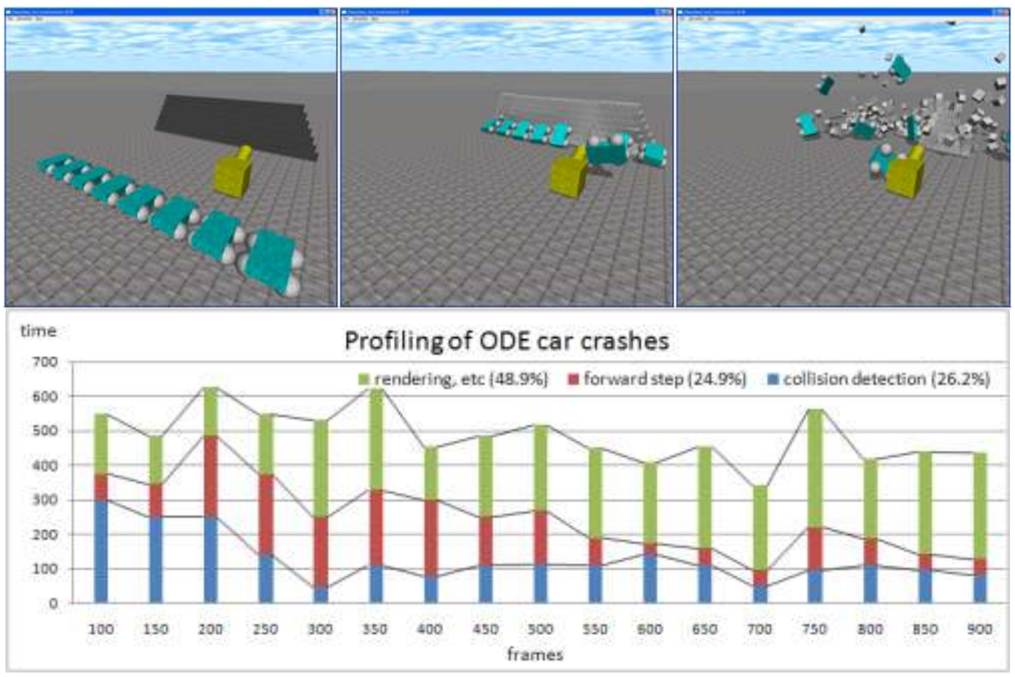

Figure 6. Profiling of Execution Time of ODE Applications

5. Conclusion

In order to verify the hardware accelerator architecture of the proposed real-time physics engine, this paper modified the source code of ODE and build a cycle accuracy simulator that operates in parallel and perform performance evaluation. Using the actual game application programs using ODE API, API function output format change, function call stream is extracted and used as the cycle accuracy simulator input. The performance evaluation metric of the software simulator shows the speed and idle ratio of each pipeline stage, parallelization ratio, numerical error relation with ERP / CFM driving parameters, visual error tolerance with repetition frequency, FIFO analysis, throughput and delay time, To optimize the design of the physical engine pipeline and to design efficient interface structures with existing CPU / GPUs. The proposed structure is implemented by Verilog HDL RTL.

Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. 2016-0-00204, Development of mobile GPU hardware for photo-realistic real time virtual reality).

References

[1] T. Y. Yeh, P. Faloutsos, S. Patel and G. Reinman, "ParallAX: An Architecture for Real-Time Physics", In 34th International Symposium on Computer Architecture (ISCA), **(2007)** June.
[2] C. J. Hughes, R. Grzeszczuk, E. Sifakis, D. Kim, S. Kumar, A.P. Selle, J. Chhugani, M. Holliman and Y.-K. Chen, "Physical Simulation for Animation and Visual Effects: Parallelization and Characterization for Chip Multiprocessors", In 34th International Symposium on Computer Architecture (ISCA), **(2007)** June.
[3] T. Y. Yeh, P. Faloutsos and G. Reinman, "Enabling Real-Time Physics Simulation in Future Interactive Entertainment", In ACM SIGGRAPH Video Game Symposium (Sandbox), **(2006)** July.
[4] V. Kokkevis, S. Osman, E. L., "High-performance physics solver design for next generation consoles", In Game Developers Conference, **(2006)**.
[5] ATI CrossFire architecture : http://www.amd.com/
[6] nVIDIA SLI architecture : http://www.nvidia.com

[7] P. Kipfer, M. Segal and R. Westermann, “UberFlow: A GPU-Based Particle Engine”, In Graphics Hardware, **(2004)**.

[8] A. Kolb, L. Latta and C. Rezk-Salama, “Hardware-based Simulation and Collision Detection for Large Particle Systems”, In Graphics Hardware, **(2004)**.

[9] K. Fatahalian, J. Sugerman and P. Hanrahan, “Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication”, In Graphics Hardware, **(2004)**.

[10] G. Baciu, W. S.-K. Wong and H. Sun, “RECODE: an image based collision detection algorithm”, The Journal of Visualization and Computer Animation, vol. 10, no. 4, **(1999)**, pp. 181-192.

[11] N. K. Govindaraju, S. Redon, M. C. Lin and D. Manocha, “CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware”, In Graphics Hardware **(2003)**, pp. 25-32.

[12] Hovok FX physics engine : http://www.hovok.com.

[13] P. M. Hubbard, “Collision detection for interactive graphics applications”, IEEE Transactions on Visualization and Computer Graphics, vol. 1, no. 3, **(1995)** September, pp. 218-230.

[14] J. Eckstein and E. Sch¨omer, “Dynamic collision detection in virtual reality applications”, In WSCG’'99, Plzen, Czech Republic, University of West Bohemia, **(1999)** February, pp. 71-78.

[15] T. Y. Yeh, G. Reinman, S. J. Patel and P. Faloutsos, “Fool me twice: Exploring and exploiting error tolerance in physics-based animation”, ACM Tr. on Graphics, **(2007)**.

[16] A. Raabe, B. Bartyzel, J. K. Anlauf and Gabriel Zachmann, “Hardware accelerated collision detection-an architecture and simulation results”, Design Automation and Test in Europe, **(2005)**.

[17] G. Zachmann and G. Knittel, “An architecture for hierarchical collision detection”, In Journal of WSCG, University of West Bohemia, Plzen, Czech Republic, **(2003)**, pp. 149-156.

[18] D. Baraff, “Non-penetrating rigid body simulation”, State of the Art Reports of EUROGRAPHICS’'93, Eurographics Technical Report Series, **(1993)**.

[19] D. Baraff, “Fast contact force computation for nonpenetrating rigid bodies”, SIGGRAPH, **(1994)**.

[20] T. Iwashita and M. Shimasaki, “Algebraic Block Red-Black Ordering Method for Parallelized ICCG Solver with Fast Convergence and Low Communication Costs”, IEEE Transactions on Magnetics, vol. 39, no. 3, **(2003)**.

[21] Ageia PhysX Physics Processing Unit: http://physx.ageia.com.

[22] Open Dynamics Engine: http://www.ode.org.

[23] T. Y. Yeh, P. Faloutsos, M. Ercegovac, S. Patel and G. Reinman, “The Art of Deception: Adaptive Precision Reduction for Area Efficient Physics Acceleration”, The 40th International Symposium on Microarchitecture (MICRO), **(2007)** December.

Author

Youngsik Kim received the B.S., M.S., and Ph.D degree in Dept. Computer Science from the Yonsei University, Korea, in 1993, 1995, and 1999 respectively. He had worked for System LSI, Samsung Electronics Co. Ltd from Aug. 1999 to Feb. 2005 as a senior engineer. Since March 2005 he has been working for Dept. of Game & Multimedia Engineering in Korea Polytechnic University. His research interests are in 3D Graphics and Multimedia Architectures, Game Programming, and SOC designs.