Hierarchical Bounding Sphere FFD-AABB Algorithm for Fast Collision Handing of 3D Deformable Objects on Smart Devices

Jae-Hong Jeon[1], Min Hong[2], Min-Hyung Choi[3], Young-Sik Jeong[4]
[1]Department of Computer Science, Soonchunhyang University, Korea
[2]Department of Computer Software Engineering, Soonchunhyang University, Korea
[3]Department of Computer Science and Engineering, University of Colorado Denver, USA
[4]Department of Multimedia Engineering, Dongguk University, Korea
jjhong@sch.ac.kr, mhong@sch.ac.kr, Min.Choi@ucdenver.com, ysjeong@dongguk.edu

Abstract

Due to the recent enriched micro-level hardware and advanced computer software technology, portable yet powerful smart devices draw fervent responses from users and consequently the world's smart device market has been rapidly expanded. In addition, since smart devices can be used anytime and anywhere using the wireless internet environment, the traditional PC-oriented works are swiftly moving to smart devices. Recently, the demands for representing more realistic 3D deformable objects have been increased for smart device based 3D game, virtual reality, 3D mobile advertisement, augmented reality and so on. However, current smart devices cannot sufficiently process the details of complex 3D object representation and associated physics based animation yet. This paper proposes a new optimized 3D object simulation and collision handling method specifically targeted for smart devices. The proposed hierarchical bounding sphere FFD-AABB (Free Form Deformation -- Axis Aligned Bounding Box) algorithm provides efficient 3D object simulation, since it quickly rejects unnecessary complex collision tasks by executing simple hierarchical bounding sphere distance tests. We have conducted experimental tests under various iOS environments and have performed comparative performance analysis with previous methods. The proposed method shows on average 34% improvement in dynamic simulation and collision handling procedures.

Keywords: Smart device, Mobile game, Bounding sphere tree, Physically-based simulation, FFD-AABB algorithm, 3D deformable objects.

1 Introduction

Most real-world objects can be simply classified as rigid objects and as deformable objects, and we need to well represent both rigidity and deformability of 3D objects for realism. Unlike rigid object modeling which is relatively simple and requires cheap computational cost, deformable object modeling is highly complex and requires much heavier computational cost for accurate simulation. Despite these demanding operations for 3D deformable object physically-based modeling and simulation, along with high quality rendering of sophisticated objects and scenes, current PC and Console platforms with proper GPUs have successfully demonstrated reasonable frame rates and plausible realism in real-time [1-2].

Recently, the computing power of smart devices has been strikingly improved; evidenced by fast multi-core processors and GPUs, increased memory capability, and newly released mobile 3D graphic engine. Accordingly the demands for experiencing plausible levels of realism with 3D computer graphics are also increased among smart device users. In addition, the realistic representation of 3D deformable objects with real-time interactivity is one of essential components for 3D mobile based game, virtual reality, advertisement, augmented reality, and so on. However, unlike PC-based hardware, the computing power of smart devices is not sufficient for plausible 3D deformable object simulation and physics based animation yet, partly because most deformable objects include abundant nodes for geometric representation which then cause complex collision related computation and rendering burdens. Therefore, an optimized object modeling and collision handling approach is necessary for an effective 3D deformable object simulation on smart devices.

The realistic 3D representation of natural objects has been steadily craved from the mobile game developers. Although some popular PC or Console based games are already crossed over into smart device environments, it is not a simple conversion task because still hardware specifications of smart devices are not thoroughly sufficient enough. For example, limited computational power, long latency time of wireless internet, low capacity of memory, and restricted view size of devices are all collectively considered as present challenges. On the other hand, smart device users would like to play mobile games and applications anywhere and anytime even during their transit times and they definitely desire to be immersed in the 3D realistic real world environment with smart devices.

Mobile devices base applications have been widely applied to various fields [3-5] and some smart device game and application developers have been successfully applied a

*Corresponding author: Young-Sik Jeong; E-mail: ysjeong@dongguk.edu
DOI: 10.6138/JIT.2013.14.5.13

primitive physics engine to represent the dynamic behavior of objects. One of good examples is "Angry Birds," a best seller mobile game application developed by Rovio Entertainment. Since most smart devices are equipped with limited computing power, this game application only uses a simple 2D based physics simulation for the control and propulsion of slingshot, for the representation of gravity, and 2D collision detection and responses. On the other hand, 3D physics engine has been rarely adopted for smart device games and applications yet [6]. As the performance of current smart devices has been greatly correlated with new hardware technology, it is predicted that smart devices which have similar specification of a conventional PC will be widely spread soon [7].

In this paper, we designed and implemented a 3D deformable object simulation application based on iOS (Apple's mobile OS) to optimally represent 3D deformable objects. The traditional FFD-AABB algorithm is improved using hierarchical bounding sphere to be specifically tailored to the smart device platform. This application is implemented using OpenGL|ES Graphics library which can be utilized on most smart devices [8]. Since one of the most critical hindrances for 3D deformable object simulation on a mobile platform is time-consuming collision handling between objects, our work is particularly focused on those performance optimizations for the collision handling process. This paper describes the performance comparison on 4 different iOS based mobile devices with 4 different experimental models. Also we provide a detailed profile analysis for the entire 3D deformable object simulation processes in major simulation steps including collision detection and response to show the computation cost improvements in each step.

The remainder of this paper is organized as follows. Section 2 explains the recent work done using FFD based deformable object simulation. In Section 3 we discuss about the implementation of 3D deformable object simulation with smart devices and explain the methodology of proposed hierarchical bounding sphere based FFD-AABB algorithm to achieve fast computational performance on smart devices. In Section 4, we show some simulation results using our proposed method, and compare it with the recent previous methods to prove the performance improvement and Section 5 is the conclusion of this paper.

2 Related Work

The simulation of deformable objects requires calculating the alteration information of vertices, edges, faces, and collision handling, so it usually takes plenty of computational cost and efforts. Thus representation of 3D deformable objects has the conflicting goals between achieving higher dynamic realism of deformable objects and providing faster computation at the same time. The traditional physically-based simulation has been deeply researched to achieve the increased levels of realism with the dynamic and physical nature of object behaviors under various external influences, but it still requires relatively high computational burden and it is not even suitable for smart devices at all. Therefore, we believe that Free Form Deformation (FFD) based method is one of the promising solutions for modeling and animation of 3D deformable objects to alleviate those issues, because it provides relatively fast computational performance even though it is not a one of the most accurate physics based methods.

Barr suggested a globally and locally defined geometric object deformation using a transform function to quickly simulate twisting, bending, or tapering motion of objects [9]. Since Barr's method only provided the restricted transformation, Sederberg and Parry suggested a general FFD method using some 3D-control nodes in the embedded surface [10]. Chang and Rockwood suggested the skeleton warping method with iterative affine transformations to reduce the computational cost of FFD [11]. However, these methods only utilized the mathematical points of view and somewhat lacked physical principles for representing deformation. Some researchers have been applied physical models into the FFD system such as mass-spring system, Finite Element Method (FEM), and shape-matching deformation to improve the realistic and efficient representation for deformable objects [12-14].

Unlike relatively simple collision checking between rigid body objects, a robust collision handling between 3D deformable objects is one of the most time-consuming tasks in 3D animation, yet it is essential component for realistic behavior. So it is a primary area to be optimized for the 3D dynamic simulation under smart device environments. Jimenez et al. surveyed various comprehensive 3D object collision algorithms [15]. Sphere trees and Axis Aligned Bounding Box (AABB) were introduced to quickly detect collisions between 3D objects [16-17]. James and Pai proposed the Bounded Deformation Tree (BD-Tree) method to reduce the collision detection time using reduced deformable models [18]. They applied the bounding sphere hierarchy for output-sensitive collision detection and it was successfully applied for massive collision problems. The optimized spatial hashing method is introduced for fast collision detection of deformable objects [19]. Since computational power and view size of smart devices are strictly limited, the most important key factor of simulating 3D deformable objects in smart devices is efficiently balancing the trade-off between the accurate representation of objects and fast computation.

3 Implementation of Simulation for 3D Deformable Objects with Smart Devices

In traditional FFD method, 3D objects are divided with FFD grids and the collisions between deformable objects are calculated using these grids. However, since this FFD method only detects the collision based on overlapped areas between FFD grids which are not tightly attached to the surface of original objects, it leads severe unnatural motions such as premature bouncing-off or hovering without an exact surface-surface contact. To improve these particular issues, the FFD-AABB algorithm, which was based on AABB updated by FFD along with the embedded surface, was introduced to tightly approximate the surface of 3D deformable objects [20]. We previously implemented and tested 3D deformable object simulation with efficient FFD-AABB based collision handling for smart device environments as well [21]. The enhanced FFD-AABB algorithm to improve the simulation frame rate for traditional PC was also introduced by adding the bounding sphere based collision test between 3D deformable objects and the considerably fast results are achieved under 3D object collision situations [22].

Since FFD-AABB algorithm only requires to update 8 nodes which include local (s, t, and u) and global coordinates (x, y, and z) in each time step, the additional computation time is negligible to compare with traditional FFD method. We believe that the proposed FFD-AABB based algorithm is well suitable for the representation of 3D object deformation using the control of LOD (Level of Detail) and associated collision handling on smart devices with relatively low computational cost.

The collision detection and response process for 3D deformable objects hinders the fast simulation, because it requires the collision test among all objects on node basis. To improve the previous FFD-AABB based deformable object simulations, we implement and perform the test for these methods with smart devices and analyze the whole simulation process. Table 1 shows the averaged computational cost of the experimental results for major FFD-AABB algorithm processes using an iPod touch 5th generation. The number of element components for 5 alphabet characters and 5 dinosaurs are shown in Table 2.

This experimental test is performed with the traditional FFD-AABB method [20] and previous bounding sphere based FFD-AABB meted [21]. Most computing time is spent in FFD-AABB mesh updating and collision detecting between objects. Although it may mainly depend on the collision situations, one of the most key factors for FFD-AABB based simulation on smart devices is the collision testing step between 3D objects and it spends around 50% of the computational time of simulation as shown in Table 1. The simulated computational time for the first and second row on each object is calculated with the traditional FFD-AABB algorithm [20] and previous bounding sphere based FFD-AABB algorithm [22], respectively.

To reduce the overall cost of collision tests between objects, we have applied the two layers of bounding sphere collision test to quickly remove the unnecessary and time-consuming fundamental element based collision detection using a simple distance calculation. The bounding sphere data structure only requires an insignificant amount of memory which has to store only its radius and the central position information.

In this paper, we propose hierarchical bounding sphere based FFD-AABB algorithm for fast simulation of 3D object deformation on smart devices. The first-level of bounding sphere wraps whole 3D deformable objects which are depicted with red color as shown in Figure 1. The center of this bounding sphere can be readily calculated from minimum and maximum position of AABB

Table 1 Experimental Result with the Previous FFD-AABB Algorithm (ms)

Test Objects	Updating MSS	Updating MAP	Collision detecting	Collision response	Total Time
Alphabet	41.388	97.668	86.982	1.545	241.394
	41.537	97.853	53.051	1.528	206.830
Dragon	95.232	141.718	349.395	7.619	619.985
	95.233	141.627	301.697	7.442	561.297

Table 2 The Number of Components for Each Deformable Object

Object	Number of objects	Number of Vertexes	Number of Triangles	Number of FFD Cells
Alphabet	5	5,936	12,008	543
Dinosaur	6	4,500	8,976	744
Rabbit	2	6,041	11,830	448
Teapot	2	6,482	12,800	194

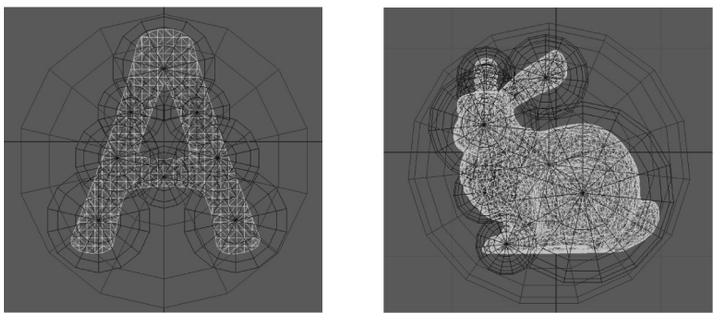

Figure 1 Structure of Proposed Two-Level Bounding Sphere Tree Based FFD-AABB Algorithm (Red Lines: First-Level of Bounding Sphere Tree, Blue Lines: Second-Level of Bounding Sphere Tree)

for each deformable object. For the first-level bounding sphere collision test, the distance from one first-level bounding sphere to other all first-level bounding spheres should be checked to determine the collision between first-level bounding spheres. When the distance between these bounding spheres is less than sum of their original radius, these two first-level bounding spheres are collided each other. When the collision is detected by these first first-level bounding spheres, the proposed algorithm proceeds into the more elaborated collision detection process using second-level bounding sphere test.

The second-level bounding spheres are allocated in the inside of the first-level bounding sphere and they are somewhat overlapped each other to robustly detect the collision regardless of their deformation. We predefined the center of second-level bounding spheres by choosing the one of object nodes, thus they automatically follow the object when the object is deformed by external influences.

When the second-level bounding spheres are collided each other, the nodes inside of these bounding spheres go into spatial hashing process to quickly find the PCPs (potential collision pairs) of bounding box nodes. Finally, the element components based detailed and time-consuming collision test between each node and plane in the same bucket of this hash table (PCPs) is proceeded to determine the actual collision between 3D deformable objects. When the collision between two objects is detected, the impulse-based collision response with velocity and geometric correction with position are applied to guarantee the correct collision handling.

Figure 2 shows the simulation process of proposed hierarchical bounding sphere based FFD-AABB algorithm. When the first-level bounding spheres are collided each other, the proposed algorithm goes into the collision test using second-level bounding spheres. Since we need to check the areas where the second-level bounding spheres are overlapped each other, the proposed method has to perform the spatial hashing to find the PCPs of the bounding box nodes on relatively narrowed parts.

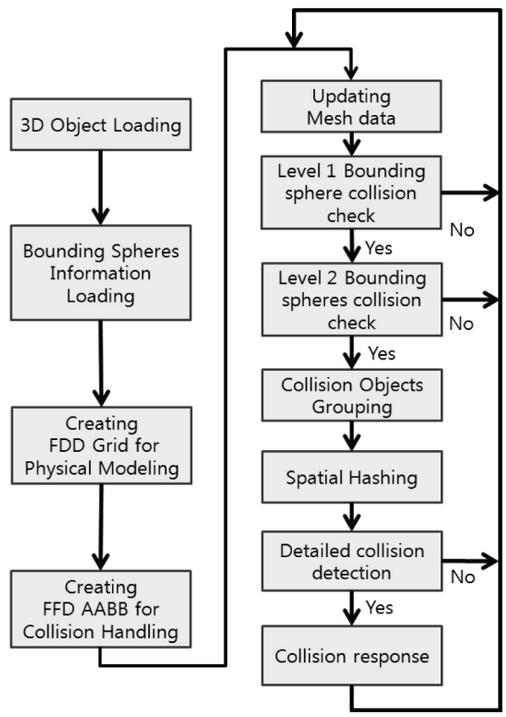

Figure 2 Simulation Process for 3D Deformable Objects with the Proposed Algorithm

Therefore, the proposed method avoids the expensive element level of collision test for entire objects, utilizes the fast distance check with the first-level bounding spheres, and reduces the range of actual element level of collision test with the second-level bounding spheres.

4 Experimental Results for 3D Deformable Objects using Smart Devices

The experimental test was performed on iPhone 5, iPad (3rd generation), iPod touch (5th generation), and iPad 1 and the proposed algorithm was coded on Mac OS 10.8 under iOS 6.X. The proposed method was implemented

in 3D using OpenGL|ES. In order to test the performance of proposed algorithm on smart devices, we created four different deformable object models such as alphabet letters, dinosaurs, rabbits and teapots with different elastic properties. Since alphabet letters have the both convex and concave features in each character, these examples are selected to be tested for fair comparison under various circumstances. The dinosaur model includes sharp gradient changes in surface contour, so it may lead the challenging collision problems during the deformable simulation. These objects are freely falling down to ground and they are collided against each other or ground as shown in figure 4. The number of vertices, triangles, and FFD cells for each object is shown in Table 2.

Figure 3 shows the results of the experimental test based on fps (frame per seconds) for each objects using iPhone 5. Deformable object simulation was performed using the proposed hierarchical bounding sphere based FFD-AABB algorithm and it was compared against previous one-level bounding sphere FFD-AABB algorithm. This result shows that the proposed method demonstrates much faster performance than previous method in all cases. The overall performance order for each smart device was iPhone 5 (7.4 fps) > iPad 3rd generation (4.4 fps) > iPod 5th generation (3.7 fps) > iPad 1 (1.65 fps). iPhone 5, showing the best performance for deformable object simulation, provides 13.11 fps for simulation of deformable alphabet letters.

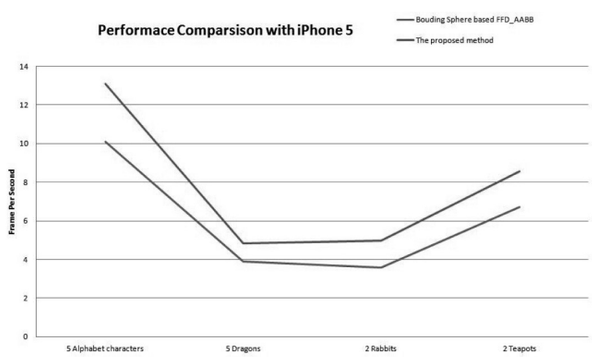

Figure 3 Performance Result of Comparison Using iPhone 5 (fps)

In addition, the computation time comparison between the proposed method and the previous one-level bounding sphere FFD-AABB algorithm is performed for all examples based on each major simulation step using iPhone 5 and the result is shown in Table 3. The computational time for the first and second row on each object is calculated with previous bounding sphere based FFD-AABB algorithm [22] and with the proposed hierarchical bounding sphere based FFD-AABB algorithm, respectively. The major simulation steps for simulation are divided in 4 steps: updating mass-spring system, updating FFD-AABB cells, collision testing, and collision response. Although it may be different in improvement rate according to the collision situations of simulation, the collision testing time is significantly reduced in the proposed hierarchical bounding sphere based FFD-AABB algorithm.

Although these results did not reach around 20 fps which is a minimum speed for plausible real-time applications, the proposed hierarchical bounding sphere based FFD-AABB algorithm achieved around 34.44% improvement in average performance (fps) for all testing examples compared with the previous bounding sphere based FFD-AABB algorithm. While more complex and deeper hierarchical bounding tree structures can be applied to figure out the detailed collision resolution for objects in close proximity, we believe that it may necessitate extra computational cost to navigate the bounding trees, more complex algorithm to manage the simulation, and additional memory resources.

The advanced realism can be achieved by adjusting the number of mesh and FFD cell. Although the computational cost will be increased, we can use relatively dense mesh and FFD cell to represent the objects and can achieve more detailed simulation results.

Table 3 Computation Time of Major Steps for Each Experiment Using iPhone 5

Objects	Updating MSS	Updating CELL	Collision Testing	Collision Response	Total Time
Alphabet	22.773	52.175	22.373	0.580	98.979
	22.637	52.600	5.376	0.582	76.279
Dinosaur	46.991	62.215	135.472	4.766	257.136
	47.007	62.267	85.487	4.779	207.247
Rabbit	70.657	90.408	131.467	5.859	280.354
	70.636	90.399	40.473	5.866	201.394
Teapot	41.990	63.362	40.465	0.838	148.592
	41.995	63.384	18.468	0.838	116.599

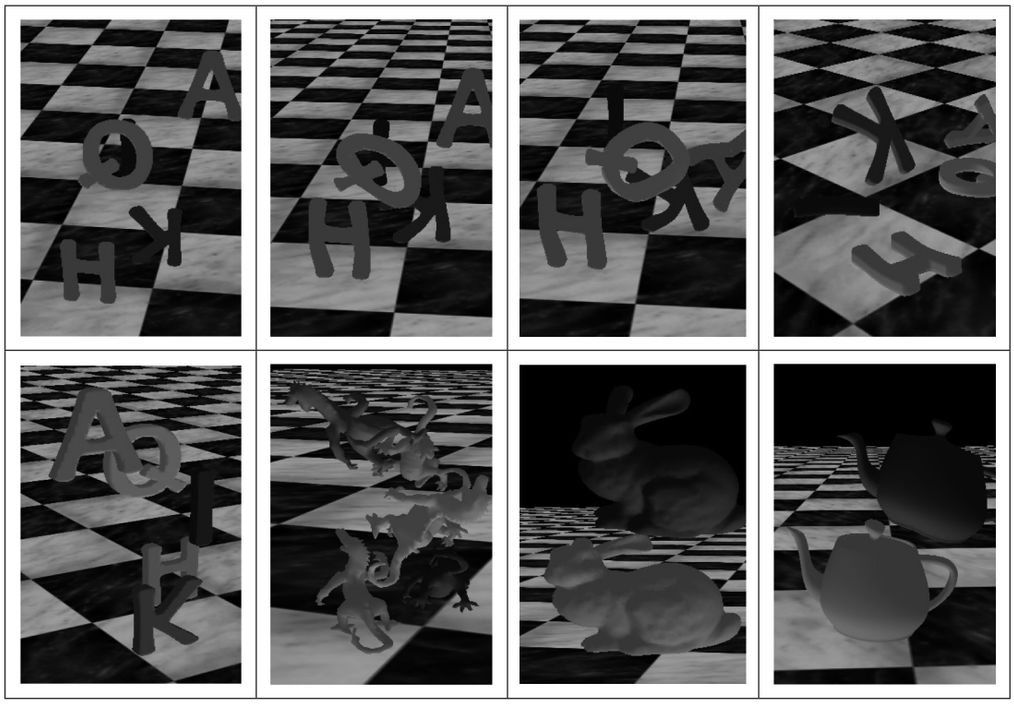

Figure 4 Snapshots of Experimental Simulation Results with the Proposed Hierarchical Bounding Sphere Based FFD-AABB Algorithm Using iOS Device

5 Conclusion

In this paper, we extended the previous bounding sphere based FFD-AABB algorithm to provide the enhanced performance in collision handling process between 3D deformable objects on iOS smart devices. We added the two-level bounding sphere tree for collision test to promptly remove the unnecessary expensive element level of collision test. The computational performance improvement of the proposed method is roughly 34% faster than previous bounding sphere based FFD-AABB algorithm, given relatively densely populated geometric structures.

The proposed hierarchical bounding sphere based FFD-AABB algorithm quickly determines the collision of 3D objects using the first-level bounding sphere based collision testing and rejects the detailed collision test when the actual collision is not occurred. When the collision is detected by the first-level bounding sphere based collision test, the second-level bounding sphere based collision test reduces the possible collision region and only the nodes within the region go into spatial hashing process to quickly find the PCPs. Since the expensive element level collision test is only performed for these PCPs, the proposed algorithm can substantially alleviate the computational time for collision detection and we believe that it is well suitable for 3D deformable objects on smart devices.

Since recent hardware technology is very speedily updated and small number of objects is usually required for smart devices because of the restricted view size, the realistic and natural simulation of 3D deformable objects on smart devices can be achieved in near future. We believe that the multi-core based programming can improve the overall performance of simulation for smart devices as well.

Acknowledgements

This paper is an extended version of CUTE 2012 conference paper [21]. This work was supported in part by the Soonchunhyang University Research Fund.

References

[1] Kyung-Sik Kim, *An Analysis of Domestic and Foreign Game Engine*, *Korea Society Broadcast Engineers Magazine*, Vol.10, No.1, 2005, pp.113-122.

[2] Ki-Seok Lee, Dong-Chun Lee, Hyang-Ki Kim, Ssang-Uk. Park and Chang-Jun Park, *Game Physics Technology*, *Electronic and Telecommunications Trends*, Vol.22, No.4, 2007, pp.53-63.

[3] Jiann-Shu Lee and Yen-Ru Shi, *A Mobile Phone Based Rock Classification System*, *Journal of Internet Technology*, Vol.10, No.3, 2009, pp.291-297.

[4] Victor Chan, Pradeep Kumar Ray and Nandan Parameswaran, *Mobile E-Health Monitoring: An Agent-Based Approach*, *IET Communications*, Vol.2, No.2, 2008, pp.223-230.

[5] Eun-Ha Song and Young-Sik Jeong, *GML Map Visualization on Mobile Devices*, *Journal of Information Processing Systems*, Vol.6, No.1, 2010, pp.33-42.

[6] KOCCA, *Trend of Mobile Game in Smart Environments*, KOCCA, 2011.

[7] *PassMark Software*, *Mobile Benchmark*, 2012, http://www.mobilebenchmark.net

[8] *OpenGL Headline News*, *Khronos*, 2012, http://www.opengl.org

[9] Alan H. Barr, *Global and Local Deformation of Solid Primitives*, *ACM Siggraph Computer Graphics*, Vol.18, No.3, 1984, pp.21-30.

[10] Thomas W. Sederberg and Scott R. Parry, *Free-Form Deformation of Solid Geometric Models*, *ACM Siggraph Computer Graphics*, Vol.20, No.4, 1986, pp.151-160.

[11] Yu-Kuang Chang and Alyn P. Rockwood, *A Generalized de Casteljau Approach to 3D Free-Form Deformation*, *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, New York, July, 1994, pp.257-260.

[12] Petros Faloutsos, Michiel Van dé Panne and Demetri Terzopoulos, *Dynamic Free-Form Deformations for Animation Synthesis*, *IEEE Transactions on Visualization and Computer Graphics*, Vol.3, No.3, 1997, pp.201-214.

[13] Steve Capell, Seth Green, Brian Curless, Tom Duchamp and Zoran Popović, *Interactive Skeleton-Driven Dynamic Deformations*, *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, San Antonio, TX, July, 2002, pp.586-593.

[14] Alec R. Rivers and Doug L. James, *FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation*, *ACM SIGGRAPH 2007 Papers*, San Diego, CA, August, 2007, p.82.

[15] P. Jiménez, F. Thomas and C. Torras, *3D Collision Detection: A Survey*, *Computer and Graphics*, Vol.25, No.2, 2001, pp.269-285.

[16] Philip M. Hubbard, *Collision Detection for Interactive Graphics Applications*, *IEEE Transactions on Visualization and Computer Graphics*, Vol.1, No.3, 1995, pp.218-230.

[17] Gino van den Bergen, *Efficient Collision Detection of Complex Deformable Models Using AABB Trees*, *Journal of Graphics Tools*, Vol.2, No.4, 1997, pp.1-13.

[18] Doug L. James and Dinesh K. Pai, *BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models*, *ACM Transactions on Graphics*, Vol.23, No.3, 2004, pp.393-398.

[19] Matthias Teschner, Bruno Heidelberger, Matthias Mueller, Danat Pomeranets and Markus Gross, *Optimized Spatial Hashing for Collision Detection of Deformable Objects*, *Proceedings of Eighth Int. Fall Workshop Vision, Modeling and Visualization (VMV'03)*, 2003, pp.47-54.

[20] SunHwa Jung, Min Hong and Min-Hyung Choi, *Collision Handling for Free-Form Deformation Embedded Surface*, *Image Processing*, *IET*, Vol.5, No.4, 2011, pp.341-348.

[21] Jae-Hong Jeon, Dong-Ik Oh, Min-Hyung Choi and Min Hong, *Implementation of 3D Deformable Objects on Smart Devices Using FFD-AABB Algorithm*, *International Conference on Ubiquitous Information Technologies & Applications (CUTE 2012)*, Vol.214, 2013, pp.833-840.

[22] JaeHong Jeon, Min-Hyung Choi and Min Hong, *Enhanced FFD-AABB Collision Algorithm for Deformable Objects*, *Journal of Information Processing Systems*, Vol.8, No.4, 2012, pp.713-720.

Biographies

Jae-Hong Jeon received BS degree in Department of Computer Software Engineering from Soonchunhyang University in 2012. Now he is undertaking a master degree of computer engineering course as a member of the Computer Graphics Lab at Soonchunhyang University. His research interests are computer game development, computer graphics, AR (Augmented Reality) and embedded motion capture.

Min Hong is an Associate Professor at the Department of Computer Software and Engineering, Soonchunhyang University in Asan, Korea. He received BS in Computer Science from Soonchunhyang University in 1995. He also received MS in Computer Science and PhD in Bioinformatics from the University of Colorado in 2001 and 2005, respectively. His research interests are in Computer Graphics, Mobile Computing, Physically-based Modeling and Simulation, Bioinformatics Applications, and u-Healthcare Applications. In present, he is a Director of Computer Graphics Laboratory at Soonchunhyang University.

Min-Hyung Choi is an Associate Professor at the Department of Computer Science and Engineering at University of Colorado Denver. He received his MS and PhD from University of Iowa in 1996 and 1999, respectively. His research interests are in Computer Graphics, Scientific Visualization and Human Computer Interaction with an emphasis on physically-based modeling and simulation for Neuroscience and Bioinformatics applications. Currently he is the Director of Computer Graphics and Virtual Environments Laboratory. In 2003, he received the NSF CAREER Award from National Science Foundation.

Young-Sik Jeong is a Professor in the Department of Multimedia Engineering at Dongguk University in Korea. His research interests include cloud computing, mobile computing, IoT (Internet of Things), and wireless sensor network applications. He received his BS degree in Mathematics and his MS and PhD degrees in Computer Science and Engineering from Korea University in Seoul, Korea in 1987, 1989, and 1993, respectively. Since 1993, he has been serving as an IEC/TC 100 Korean Technical Committee member, as the IEC/TC 108 Chairman of Korean Technical Committee, and as an ISO/IEC JTC1 SC25 Korean Technical Committee member. Also He is an Editor-in-Chief (EiC) of *Journal of Information Processing Systems*, an associate editor of *International Journal of Communication Systems* (IJCS) and an editor of *Journal of Internet Technology* (JIT), finally an associate editor of *Journal of Human-centric Computing* (HCIS) and so on. He is also is a member of the IEEE.