行政院國家科學委員會專題研究計畫 成果報告

運用 CUDA 在多核心平台上進行平行運算模擬多層不可伸展衣服
研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 98-2221-E-009-143-
執 行 期 間 ：98 年 10 月 01 日至 99 年 09 月 30 日
執 行 單 位 ：國立交通大學資訊工程學系（所）

計 畫 主 持 人：黃世強

計畫參與人員：碩士班研究生-兼任助理人員：方士偉
碩士班研究生-兼任助理人員：劉 政旻
碩士班研究生-兼任助理人員：陳 奕辰
碩士班研究生-兼任助理人員：曾韋閔
碩士班研究生-兼任助理人員：林冠辰
碩士班研究生-兼任助理人員：鄭游駿
碩士班研究生-兼任助理人員：陳 蔚恩
碩士班研究生-兼任助理人員：葉喬之

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 99 年 12 月 26 日

前言

Continuous collision detection is widely used in virtual reality applications and physics-based simulations as well as in robot navigation planning. We developed parallel scheme for improving continuous collision detection and elementary test processing. We also employed CUDA for rendering objects on a multi-core platform. We implemented our system for simulating garments.

Typically a four-stage process is employed for performing collision detection:(1) construction of the bounding volume hierarchies; (3) updating the bounding volume hierarchies; (2) traversing pairs of bounding volume hierarchies for culling; and (4) filtering and the elementary test processing. Much work has been done in the first three stages of the process for improving the collision detection performance. The last step, the elementary test processing, has attracted much attention recently. In the elementary test processing, the potentially colliding triangle pairs are computed for collisions. A significant amount of time is spent in the elementary test processing in physics based simulation, such as deformable objects interacting with each other. However, a majority of the potentially colliding triangle pairs do not collide. These can trigger false-positive collision events. The situation deteriorates for multiple stacked thin deformable surfaces, for example, piles of cloth draping over objects. The false positive events could be eliminated by applying a low cost filter for improving the computation cost in the elementary test processing, thereby improving the overall performance of collision detection.

Lastly, we employed CUDA for rendering objects. Users can use our simulation system to visualize the results.

研究目的

Continuous collision detection improves the computation of the contact information for interacting objects in dynamic virtual environments. The computation cost is relatively high in the phase of the elementary test processing. In virtual environments, such as crowds in large urban models, there is a large portion of feature pairs that do not collide but the computation is relatively of high cost. We developed a robust approach for solving the scalability of the collision detection problem by applying four distinct phases. Moreover, a parallel collision detection scheme was developed for improving the speed of collision detection. We also built a system for rendering objects based on CUDA.

文獻探討

I. Continuous collision detection

Collision detection is important in the simulation of deformable objects, such as cloth simulation [Breen et al. 1994; Volino and Magnenat-Thalmann 2000; Bridson et al. 2002; Baraff and Witkin 2003; Choi and Ko 2005; Govindaraju et al. 2005; Harmon et al. 2008; Ye 2008; Volino et al. 2009]. A comprehensive survey on collision detection for deformable objects can be found in [Teschner et al. 2005]. A brute force approach for collision detection is to perform an elementary test for every two features of triangles. The run–time complexity is bounded by O(n2), where n is the number of features. Employing bounding volumes would cull away a large portion of non-colliding triangles, such as the methods in [Hubbard 1993; Gottschalk et al. 1996; Klosowski et al. 1998; van den Bergen 1999; Mezger et al. 2003; Smith et al. 1995; van den Bergen 1999]. The hierarchies of k-DOPs [Klosowski et al. 1998] are widely used because of its high culling effectiveness.

Continuous collision detection can be used for computing precise contact times for two objects. Continuous collision detection has been applied in the simulation of rigid bodies [Redon et al. 2005]. For handling deformable objects, an approach was proposed in [Liu et al. 1996] for computing the contact time between two triangles by checking the fifteen feature pairs of these two triangles. Each test for a feature pair involves solving a cubic equation for the times when the feature pair is coplanar. The shortest distance of the feature pair is then computed. If the distance is less than or equal to a predefined threshold, the feature pair collides. Subsequently, this method was adopted in [Provot 1997; Bridson et al. 2002; Wong and Baciu 2005; Hutter and Fuhrmann 2007]. Solving the cubic equations takes the most amount of time. It is unnecessary to solve the cubic equations as a large portion of feature pairs are not coplanar over the simulation time interval. Based on that, Tang et al. [Tang et al. 2010] proposed an approach to filter the feature pairs that are not coplanar so as to avoid solving the corresponding cubic equations. They employed a numerical iteration method, Interval Newton, to solve the cubic equation of the feature pair if the filter test is passed.

An assignment scheme was proposed in [Wong and Baciu 2006] for assigning features to adjacent triangles. Each edge and vertex in the mesh is assigned to one of their adjacent triangle in a randomized manner. Their method significantly reduces the number of duplicate tests for feature pairs. The idea was adapted in [Hutter and Fuhrmann 2007; Curtis et al. 2008; Tang et al. 2009a].

High-level culling technologies have been proposed for reducing the computation cost in self-collision detection. Volino et al. [Volino and Magnenat-Thalmann 1994] proposed a method for partitioning surfaces into low-curvatured regions. There would be no self-collisions for the regions with low curvature. Originally, the method was adopted in discrete collision detection. The idea was extended in [Provot 1997; Wong and Baciu 2005; Tang et al. 2009a] for continuous collision detection. Collision detection can be performed at interactive rate for complex deformable objects by employing parallel computation techniques. An approach was proposed in [Kim et al. 2009] for treating inter-collisions and self-collisions as inter-CD tasks. Independent inter-CD tasks are assigned to different threads for performing parallel update and traversal of bounding volume hierarchies on a multi-core platform with graphics processing units. In [Tang et al. 2009b], a parallel collision detection method was developed for storing the front nodes temporarily as the start nodes for the traversal of bounding volume hierarchies at each frame.

II. Parallel collision detection

隨著平行化技術的成熟，可形變物體的碰撞偵測，也將觸角延伸向這個領域。Kim et al. [12] 提出了“inter-CD task”，將資料互相獨立的inter-CD task 分派給不同的執

行緒，達成BVH traversal 及更新的平行化。Tang et al. [21]改進了先前的論文[22]，提出"front node" 來暫存每一次做BVH traversal 的起始點，來加速找出所有elementary test 的過程。

III. CUDA for rendering objects
Popov et al. [14] presented an algorithm of the SAH-based kd-tree on GPUs by increasing the coherence of memory accesses during construction of the kd-tree signi_cantly. Later on, Zhou et al. [21] presented a real time algorithm to construct kd-tree on GPUs. It constructs tree nodes completely in breadth first search order by exploiting the large scale parallelism of GPUs. For bounding volume hierarchies, Lauterbach et al. [10] presented a hybrid algorithm to construct SAH-based hierarchies on GPUs.

研究方法
I. Continuous collision detection

We developed a robust approach for solving the scalability of the collision detection problem by applying four distinct phases. First, k-DOPs are used for culling non-proximal triangles. Second, the feature assignment scheme is used for minimizing the number of potentially colliding feature pairs. Third, an intrinsic filter is employed for filtering non-coplanar feature pairs. Forth, we use a direct method for computing the contact time that is more efficient than the numerical Interval Newton method. We have implemented our system and have compared its performance with the most recently developed approaches.

II. Parallel collision detection
我們提出兩個平行化處理的架構，與不同 k 值的 k-DOPs 做搭配，藉由數據的分析，得知在什麼樣 的平行化環境下，如何選用最適合的 bounding volume 來實作，才能夠在碰撞偵測的過程中，帶給我們最高的效能。我們的系統中也加入了高階層篩選和元件分配的機制。藉由這些實驗數據的分析，將幫助我們依實驗環境的不同，挑選合適的 k-DOPs。

爲了比較 k-DOPs 在一般的平行化系統下的表現，我們提出了兩種版本的平行化處理環境。所有被找到的可能碰撞元素(三角形 PCTPs)都會被儲存在名爲 PCTPArray 的一維陣列中。在平行化環境下，系統事先產生 n_{tot} 條的執行緒，閒置執行緒被稱爲 t_{idle}，數量記爲 nidle。其中一條執行緒，將負責處理 BVH traversal 的工作，並把找到的 PCTPs 放到 PCTPArray 中，我們稱它爲 ttra ；若有需要的話，我們會指定一條執行緒，專門來溝通 ttra 與其他閒置中的執行緒。

Static Task Scheduling. 在這個版本中，我們將不依靠額外的執行緒來溝通系統中的其他執行緒。換句話說， t_{tra} 的工作範圍，除了全程的 BVH traversal ，也就是確認 bounding volume 之間是否有互相重疊，判斷哪些三角形對必須通過基層測試來確認碰撞，並將這些三角形對標示爲 PCTPs 以外，還得負責將找到的 PCTPs ，分配給其他執行緒執行基層計算。

Dynamic Task Scheduling. 顯然地，在上述的方法中，所有處於閒置狀態下的執行緒將被迫等待 t_{tra} 找到所有可能的 PCTPs 後，才能開始動工。然而讓大多數的執行緒閒置，空轉著等待單一條執行緒的方式，是有許多改進空間的。與此同時，我們也修改 t_{tra} 的工作方式。t_{tra} 不再讓空閒的執行緒等待它找到所有存在的 PCTPs 後，才能經由 t_{tra} 的平均分配，開始基層測試的計算。

爲了達成這個目標，我們決定將 PCTPs 包裝成一個個固定大小的封包。在 t_{tra} 執行 BVH traversal 的過程中，一旦發現有封包被裝滿，便 動態地從 n_{idle} 條 t_{idle} 中找出一條空閒的執行緒來接管這個裝滿的封包。我們發現這樣的改變是值得的。經過這樣的調整後，我們能在整體效能上，進一步取得改進。

III.CUDA for rendering objects
We adopted multithreading techniques and streaming SIMD extensions (SSE) for ray packets. A dynamic loading balancing scheme is employed for multiple threads on a CPU. On the other hand, we perform ray tracing on a GPU with CUDA. Then we assign tasks to the CPU and the GPU as evenly as possible for load balancing.

結果與討論

We have built out system and tested for performance.

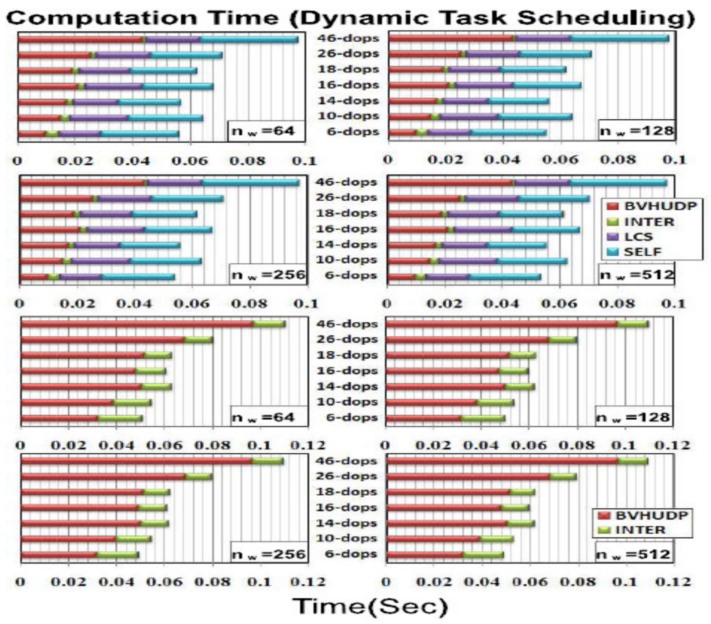
I. Continuous collision detection
We implemented our algorithm and performed experiments. All the experiments were performed on an Intel(R) 2.66 GHz quadcore CPUs with 4GB main memory and one thread was used. The computation was all carried out in double precision floating point for solving the cubic equations. The type of bounding volume was k-DOPs. We had implemented 6-, 10-, 14-, 16-,18- and 26-DOPs. For the update and traversal of bounding volume hierarchies, we implemented them based on Intel SSE instruction set. The shortest distance between the features is computed for the times t in an ascending order [Hutter and Fuhrmann 2007]. On average we found that the performance for using 14-DOPs, 16-DOPs and 18-DOPs were the best. We compared our algorithm to I-Newton (implemented in [Kim et al. 2009]) and NPF (non-penetration filter in [Tang et al. 2010]) The type of bounding volume was 16-DOP in all methods. Moreover, we employed our proposed feature assignment scheme for avoiding duplicate elementary tests. We recorded the simulation data (such as positions of vertices) and the same set of data was then employed for each method. In this way, all methods would have to process the same amount of potentially colliding triangle pairs.

Table 1. *Speedup factors for the elementary test processing: Comparison to I-Newton and NPF.*

	Collision type	I-Newton	NPF & I-Newton
N-body	Inter-collision	1.44x	1.68x
	Self-collision		
Cloth-ball	Inter-collision	1.39x	1.30x
	Self-collision	1.37x	1.29x
Garments	Inter-collision	1.44x	1.32x
	Self-collision	1.89x	1.69x
Dragons	Inter-collision	1.30x	1.23x
	Self-collision	1.32x	1.24x
Cloth-torus	Inter-collision	1.38x	1.28x
	Self-collision	1.25x	1.23x
Bunnies	Inter-collision	1.43x	1.34x
	Self-collision	1.39x	1.27x

II. Parallel collision detection
我們主要的貢獻為提出兩個平行化處理碰撞偵測的演算法，並瞭解 k-DOPs 是如何地在效能提昇上扮演重要的角色。我們將著眼點放在：什麼樣維度的 k-DOPs 能在平行化的系統下取得更好的表現。藉由實驗數據的幫助，選擇最適合的 k-DOPs 來實作，將大大地幫助效能上的改進，同時也朝系統最佳化的目標再進了一步。結果如圖 Graph 1.
選用 Cloth-ball 時，比起單執行緒的版本，各種維度的 k-DOPs 平均可取得 2.9X，2.5X， 2.8X， 2.5X， 2.8X，2.7X 和 2.6X 的加速，比起 StaticTaskScheduling 有著進一步的改進。加速的情形以 6-DOPs 最明顯。且在整體效率上， 6-DOPs 也同樣有著最佳的表現。使用 N-body 時，獲得的加速分別為 1.8X， 1.8X， 1.9X， 2.0X， 2.0X， 2.0X 和 2.2X。在沒有 self-collision 的情況下，46-DOPs 仍然在加速上取得領先，整體效能還是以 6-DOPs 最優。

Graph 1.



III. CUDA for rendering objects

We performed CPU ray tracing on Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHZ 2.67 GHz, 3.25GB RAM platform.There are four cores on the CPU, so we employed four threads to get better efficiency. For GPU ray tracing, CUDA process runs on NVIDIA Quadro FX 4600 graphics card that 1.0 compute capability is supported. The overall performance, however, could be improved up to around 50%.

References:

I. Continuous collision detection and II. Parallel collision detection

1. BARAFF, D., AND WITKIN, A. 2003. Untangling cloth. *ACM Transactions on Graphics 22*, 3, 862–870.
2. BREEN, D., HOUSE, D., AND WOZNY, M. 1994. Predicting the drape of woven cloth using interacting particles. In *Computer Graphics Proceedings, ACM SIGGRAPH*, 365–372.
3. BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics 21*, 3, 594–603.
4. CHOI, K., AND KO, H. 2005. Stable but responsive cloth. In *ACM SIGGRAPH 2005 Courses*, ACM, 1.
5. CURTIS, S., TAMSTORF, R., AND MANOCHA, D. 2008. Fast collision detection for deformable models using representative triangles. In *Proceedings of the Symposium on Interactive 3DGraphics and Games*, 61–69.
6. GOTTSCHALK, S., LIN, M., AND MANOCHA, D. 1996. OBBTree: A hierarchical structure for rapid interference detection. In *Computer Graphics Proceedings, ACM SIGGRAPH*, 171–180.
7. GOVINDARAJU, N., KNOTT, D., JAIN, N., KABUL, I., TAMSTORF, R., GAYLE, R., LIN, M., AND MANOCHA, D. 2005. Interactive collision detection between deformable models using chromatic decomposition. *ACM Transactions on Graphics 24*, 3, 991–999.
8. HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. 2008. Robust treatment of simultaneous collisions. In *Computer Graphics Proceedings, ACM SIGGRAPH*, 23.
9. HUBBARD, P. 1993. Interactive collision detection. In *Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality*, 24–31.
10. HUTTER, M., AND FUHRMANN, A. 2007. Optimized continuous collision detection for deformable triangle meshes. *Proceedings of WSCG*, 25–32.
11. KIM, D., AND YOON, S., 2009. OpenCCD: Continuous collision detection API, used in HPCCD. http://sglab.kaist.ac.kr/OpenCCD.
12. KIM, D., HEO, J., HUH, J., KIM, J., AND YOON, S. 2009. HPCCD: Hybrid parallel continuous collision detection using CPUs and GPUs. *Computer Graphics Forum 28*, 7, 1791–1800.
13. KLOSOWSKI, J., HELD, M., MITCHELL, J., SOWIZRAL, H., AND ZIKAN, K. 1998. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics 4*, 1, 21–36.

14. LIU, J., KO, M., AND CHANG, R. 1996. Collision avoidance in cloth animation. *Visual Computer 12*, 5, 234–243.
15. MEZGER, J., KIMMERLE, S., AND ETZMUSS, O. 2003. Hierarchical techniques in collision detection for cloth animation. *Journal of WSCG 11*, 2, 322–329.
16. PROVOT, X. 1997. Collision and Self-collision Handling in Cloth Model Dedicated to Design Garments. In *Graphics Interface*, 177–189.
17. REDON, S., LIN, M., MANOCHA, D., AND KIM, Y. 2005. Fast continuous collision detection for articulated models. *Journal of Computing and Information Science in Engineering 5*, 126.
18. SMITH, A., KITAMURA, Y., TAKEMURA, H., AND KISHINO, F. 1995. A simple and efficient method for accurate collision detection among deformable polyhedral objects in arbitrary motion. In *Proceedings of the Virtual Reality Annual International Symposium*, 136–145.
19. TANG, M., AND MANOCHA, D., 2010. Non-penetration filters, source code used in Fast Continuous Collision Detection using Deforming Non-Penetration Filters. http://gamma.cs.unc.edu/DNF/DNF-code.cpp.
20. TANG, M., CURTIS, S., YOON, S., AND MANOCHA, D. 2009. ICCD: Interactive continuous collision detection between deformable models using connectivity-based Culling. *IEEE Transactions on Visualization and Computer Graphics 15*, 4, 544–557.
21. TANG, M., MANOCHA, D., AND TONG, R. 2009. Multi-core collision detection between deformable models. In *SIAM/ACM Joint Conference on Geometric and Physical Modeling*, 355–360.
22. TANG, M., MANOCHA, D., AND TONG, R. 2010. Fast continuous collision detection using deforming non-penetration filters. In *Computer Graphics Proceedings, ACM SIGGRAPH*, 7–13.
23. TESCHNER, M., KIMMERLE, S., HEIDELBERGER, B., ZACHMANN, G., RAGHUPATHI, L., FUHRMANN, A., CANI, M., FAURE, F., MAGNENAT-THALMANN, N., STRASSER, W., ET AL. 2005. Collision detection for deformable objects. In *Computer Graphics Forum*, 61–81.
24. VAN DEN BERGEN, G. 1999. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics, GPU, and Game tools 2*, 4, 1–14.
25. VOLINO, P., AND MAGNENAT-THALMANN, N. 1994. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In *Computer GraphicsForum*, 155–166.
26. VOLINO, P., AND MAGNENAT-THALMANN, N. 2000. Implementing fast cloth simulation with collision response. In *Computer Graphics International*, vol. 7(1), 9–4.
27. VOLINO, P., MAGNENAT-THALMANN, N., AND FAURE, F. 2009. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Transactions on Graphics 28*, 4, 105.
28. WONG, W., AND BACIU, G. 2005. Dynamic interaction between deformable surfaces and non-smooth objects. *IEEE Transactions on Visualization and Computer Graphics 11*, 3, 329–340.

29. WONG, S., AND BACIU, G. 2006. A randomized marking scheme for continuous collision detection in simulation of deformable surfaces. In *Proceedings of ACM Int'l Conf. on Virtual Reality Continuum and Its Applications*, 181–188.
30. YE, J. 2008. Simulating inextensible cloth using impulses. In *Computer Graphics Forum*, vol. 27(7), 1901–1907.

III. CUDA for rendering objects

1. T. Aila and S. Laine. Understanding the efficiency of ray traversal on GPUs. In Proceedings of High Performance Graphics, pages 145-149, 2009.
2. A. Appel. Some techniques for shading machine renderings of solids. In Proceedings of the Eastern Joint Computer Conference, pages 37-45, 1968.
3. C. Benthin. Realtime Ray Tracing on Current CPU Architectures. PhD thesis, 2006.
4. M. Ernst, C. Vogelgsang, and G. Greiner. Stack implementation on programmable graphics hardware. In Proceedings of the Vision, Modeling, and
5. Visualization Conference 2004, Stanford, California, USA, November 16-18, 2004, pages 255-262, 2004.
6. T. Foley and J. Sugerman. KD-tree acceleration structures for a GPU raytracer. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, page 22, 2005.
7. D. Fussell and K. Subramanian. Fast ray tracing using kd trees. 1988.
8. J. Goldsmith and J. Salmon. Automatic creation of object hierarchies for ray tracing. IEEE Computer Graphics and Applications, 7(5):14-20, 1987.
9. V. Havran. Heuristic Ray Shooting Algorithms. Phd thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, 2000.
10. D. Horn, J. Sugerman, M. Houston, and P. Hanrahan. Interactive kd tree GPU raytracing. In Proceedings of the 2007 symposium on Interactive 3D graphics and games, page 174, 2007.
11. C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, and D. Manocha. Fast BVH construction on GPUs. In Computer Graphics Forum, pages 375-384, 2009.
12. T. M oller and B. Trumbore. Fast, minimum storage ray/triangle intersection. In ACM SIGGRAPH 2005 Courses, page 7, 2005.
13. C. NVIDIA. Compute Uni_ed Device Architecture, Programming Guide, version 2.1, 2008.
14. R. Overbeck, R. Ramamoorthi, and W. Mark. Large ray packets for real-time Whitted ray tracing. In IEEE/EG Symp. on Interactive Ray Tracing, 2008.
15. S. Popov, J. G unther, H. Seidel, and P. Slusallek. Experiences with streaming construction of SAH KD-trees. In Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing, pages 89-94, 2006.

16. S. Popov, J. G unther, H. Seidel, and P. Slusallek. Stackless kd-tree traversal for high performance GPU ray tracing. In Computer Graphics Forum, volume 26, pages 415-424, 2007.
17. T. Purcell, I. Buck, W. Mark, and P. Hanrahan. Ray tracing on Programmable graphics hardware. In ACM SIGGRAPH 2005 Courses, page 268, 2005.
18. A. Reshetov, A. Soupikov, and J. Hurley. Multi-level ray tracing algorithm. ACM TOG, 24(3):1176-1185, 2005.
19. S. Rubin and T. Whitted. A 3-dimensional representation for fast rendering of complex scenes. In Proceedings of the 7th annual conference on Computer graphics and interactive techniques, page 116, 1980.
20. I. Wald, S. Boulos, and P. Shirley. Ray tracing deformable scenes using dynamic bounding volume hierarchies. ACM TOG, 26(1), 2007.
21. I. Wald, P. Slusallek, C. Benthin, and M. Wagner. Interactive rendering with coherent ray tracing. In Computer Graphics Forum, pages 153-165, 2001.
22. K. Zhou, Q. Hou, R. Wang, and B. Guo. Real-time kd-tree construction on graphics hardware. In ACM SIGGRAPH Asia 2008 papers, page 126, 2008.
23. M. Zlatu_ska and V. Havran. Ray Tracing on a GPU with CUDA: Comparative Study of Three Algorithms. 2010.

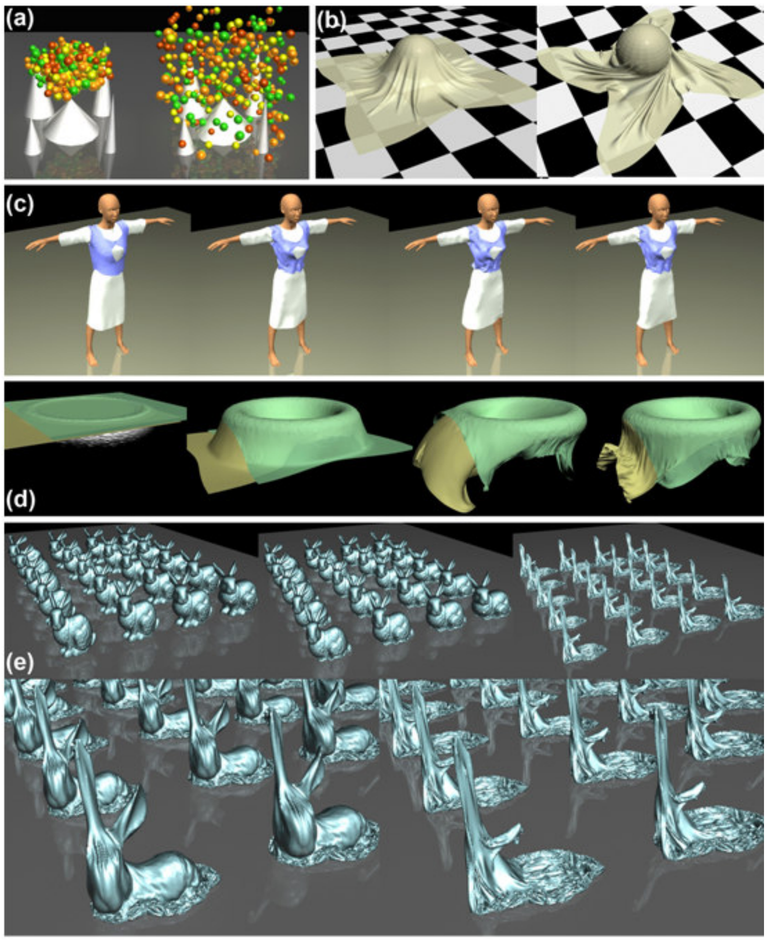


Fig. 1 Snapshots of experiments for collision detection.

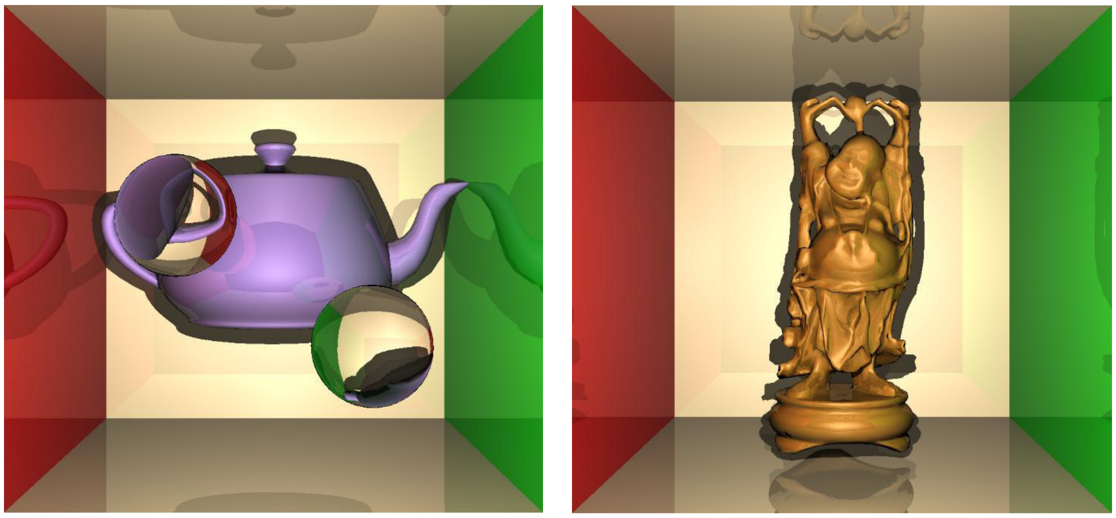


Figure 2. Rendering results.

國科會補助計畫衍生研發成果推廣資料表

日期:2010/12/26

國科會補助計畫	計畫名稱: 運用CUDA在多核心平台上進行平行運算模擬多層不可伸展衣服
	計畫主持人: 黃世強
	計畫編號: 98-2221-E-009-143-　　　　學門領域: 計算機圖學

無研發成果推廣資料

98 年度專題研究計畫研究成果彙整表

計畫主持人：黃世強	計畫編號：98-2221-E-009-143-
計畫名稱：運用 CUDA 在多核心平台上進行平行運算模擬多層不可伸展衣服	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	2	2	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	1	1	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	無

🟥	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

   ■達成目標

   □未達成目標（請說明，以 100 字為限）

   　　□實驗失敗

   　　□因故實驗中斷

   　　□其他原因

   說明：

2. 研究成果在學術期刊發表或申請專利等情形：

   論文：■已發表 □未發表之文稿 □撰寫中 □無

   專利：□已獲得 □申請中 ■無

   技轉：□已技轉 □洽談中 ■無

   其他：（以 100 字為限）

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）（以 500 字為限）

我們的研究是關於運用 CUDA 在多核心平台上進行平行運算模擬布料·

學術成就:

我們總共發表了三篇論文, 其中兩篇是國內, 另外一篇是國外·

我們提出的方法對運用 CUDA 在多核心平台上進行平行運算方面的研究，十分有助益·

技術創新:

我們完成兩個系統:

1. 在多核心平台上對碰撞運算

我們提出以動態形式分配工作給多核心處理器，以達到每一個核心的工作量相近，從而提高計算的效能·

2. 在 CUDA 和多核心平台架構上的 ray tracer

我們分配工作給核心處理器和圖像處理器，進行光線追蹤的平行運算·應用 CUDA 對光線追蹤進行平行運算，其中 ray 對基本元素三角面的碰撞技術，稍作變更，也可以應用在布料碰撞運算·

應用價值:

在多核心平台上對碰撞運算和應用 CUDA 技術，能夠在較短時間內完成複雜的計算·這些方法可以加快模擬的速度，縮短製作時間，不論在電影、遊戲及衣服設計都有廣泛應用·由於這些技術可以加快模擬的速度，模擬系統能夠達到互動的效能，使用者可以即時知道模擬的結果，對模擬作出改良，從而增加使用者的工作效率，對社會的進步有所助益·