Collision Detection between Cloth and Other Objects Using a GPU Based Ray-Tracing

Tzvetomir Vassilev

Abstract: *This paper presents an approach for cloth-body and cloth-cloth collision detection in computer graphics simulations of clothing. It is an object-space based algorithm implemented in CUDA an OptiX ray-tracing engine on the GPU. The underlying idea behind this work is to speed up the solution of the collision detection problem by utilizing the excessive computational capacity of contemporary GPUs. Results of the experiment are presented at the end of the paper.*

Key words: *Collision Detection, Ray tracing, GPU computation, Cloth Simulation*

INTRODUCTION

The main objective of this work is to develop an efficient approach for collision detection (CD) in computer graphics simulations of clothing. Collision detection is the most time consuming stage in realistic cloth simulations. It is so due to a number of factors as: complexity of the CD problem; the deformable nature of cloth; cloth objects are usually situated near or on the same surface of other objects in the scene which results in multiple collisions in dynamic scenarios; achieving realism requires highly detailed object surfaces.

Over the past few years the continual development of graphics processing unit (GPU) technology made these devices attractive for more than just rendering. The today's modern GPUs are very powerful computationally, capable of parallel processing. They outperform the modern central processing units (CPUs) in floating-point calculations. In addition high-level programming languages have been developed for GPUs, which made them popular for speeding up all kinds of computational tasks.

The approach presented in this paper utilizes the NVIDIA OptiX ray-tracing engine for solving the CD problem. OptiX programs are executed on the GPU, which will reduce the workload on the CPU and is expected to improve the simulation speed in general.

The rest of the paper is organized as follows. The next section reviews previous work on CD on GPUs. Section 3 describes the utilized cloth model and gives basic idea about the way a simulation is carried out. Section 4 presents the object space technique for CD on the GPU. Section 5 gives results of the experiment and Section 6 concludes the paper.

PREVIOUS WORK ON COLLISION DETECTION ON GPUs

Due to the computational complexity of the CD task and its impact on the overall simulation speed in various computer graphics applications, significant research has been done on the topic by the Computer Graphics community. This resulted in wide variety of approaches and solutions. Initially, most of them ran on the CPU only. Later approaches involved the GPU implicitly (without direct GPU programming). In recent approaches it is common for the GPU to explicitly take part in or entirely solve the CD problem. Due to GPU's stream processing oriented architecture implementing an entire CD algorithm to run on the GPU only may not be always feasible. Thus, hybrid approaches that involve both the CPU and GPU in the CD process are quite common.

The techniques for CD could be divided in two major groups – based on geometrical interference tests in object-space and based on z-buffer interference tests in image-space. A common feature of the object-space approaches is the utilization of hierarchical structure(s) for reducing the complexity of the problem. There are two main strategies for building such hierarchies. The one is focused at space or voxel subdivision while the other is focused at the objects in the scene and relies on different types of bounding volumes and in particular bounding boxes.

In 2003 Knot et al [6] presented CInDeR – an image-space algorithm for CD on the GPU. The GPU is used implicitly via hardware frame buffer operations which implement virtual ray-casting in order to detect static interference between solid objects. Kim et al [5] proposed hybrid parallel continuous CD method that takes advantage of hybrid multi-core architectures – using the general-purpose CPUs to perform bounding volume hierarchy (BVH) traversal and culling while GPUs are used to perform the actual tests for collisions which are reduced to solving cubic equations. Another hybrid approach is presented in the work of Georgii et al [1,2]. On the contrary of the work of Kim et al the GPU here is used to locate couples of possibly colliding polygons via ray tracing and depth peeling techniques. Afterwards, the actual tests for exact interaction are performed on the CPU. A hybrid object-space CD algorithm which mainly uses the GPU is developed by Zhang and Kim [10]. It is suitable for deformable polygonal objects. Given two objects the algorithm can detects all possible pairwise primitive-level intersections between them using two hierarchies of axis aligned bounding boxes (AABBs). The two balanced AABB trees are used as input streams over which the GPU performs tests for overlapping. An encoding / decoding strategy is also used to transfer only the list with possible collisions instead of the entire output streams which saves GPU – CPU communication bandwidth. Finally the tests for intersection between primitive-level elements (e.g. triangles) are implemented using the CPU. This approach is not capable of detecting self-collisions. Pabst et al [7] propose another hybrid CPU/GPU CD technique for rigid and deformable objects based on spatial subdivision. The algorithm is developed with scalability in mind on both the CPU and GPU sides and implements highly parallel spatial subdivision method to quickly prune away non colliding parts of the scene in a broad phase. Then a narrow phase with GPU-optimized exact collision tests follows. The CPU is used for BVH updates. A pure GPU based CD approach for rigid bodies is presented by Gress et al [3]. It operates in object-space and utilizes balanced AABB trees as BVH. The algorithm maps the AABB trees onto GPUs and performs a breadth-first search on the trees. During the traversal of hierarchy, occlusion query is used to count the number of overlapping AABB pairs and recursive AABB overlapping tests in object space is implemented using GPUs. In a later research the algorithm is modified and implemented for deformable parameterized surfaces [4]. The modifications include real-time generation of BVH on the GPU via stenciled geometry images representing the individual parameterized surfaces.

CLOTH MODEL

The elastic model of cloth, used in this work, is a mesh of $l \times n$ mass points, each of them being linked to its neighbours by massless springs of natural length greater than zero.

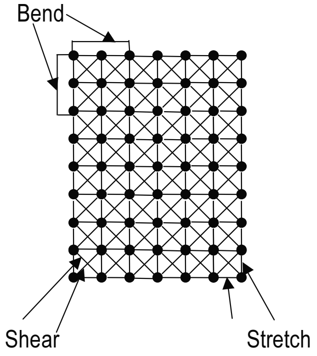


Fig. 1. Spring types in the cloth model

There are three different types of spring (Fig. 1):

- Springs linking vertices [*i*, *j*] with [*i*+1, *j*], and [*i*, *j*] with [*i*, *j*+1] are called “stretch” springs;
- Springs linking vertices [*i*, *j*] with [*i*+1, *j*+1], and [*i*+1, *j*] with [*i*, *j*+1] are called “shear” springs;
- Springs linking vertices [*i*, *j*] with [*i*+2, *j*], and [*i*, *j*] with [*i*, *j*+2] are called “bend” springs.

As the names indicate, the first type of spring implements resistance to stretching, the second – to shearing and the third – to bending.

Let $\mathbf{p}_{ij}(t)$, $\mathbf{v}_{ij}(t)$, $\mathbf{a}_{ij}(t)$, where $i=1,\ldots,l$ and $j=1,\ldots,n$, be respectively the positions, velocities, and accelerations of the mass points at time t. The system is governed by the basic Newton’s law:

$$\mathbf{f}_{ij} = m_{ij}\,\mathbf{a}_{ij}, \tag{1}$$

where m_{ij} is the mass of point ij and $\mathbf{f}_{ij}$ is the sum of all forces applied at point ij. The force $\mathbf{f}_{ij}$ can be divided in two categories.

Internal forces arise from the tensions of the springs. The overall internal force applied at the point ij is a result of the stiffness of all springs linking this point to its neighbours:

$$\mathbf{f}_{int}(\mathbf{p}_{ij}) = -\sum_{k,l} k_{ijkl}\left((\mathbf{p}_{kl}-\mathbf{p}_{ij}) - l^0_{ijkl}\frac{\mathbf{p}_{kl}-\mathbf{p}_{ij}}{\left\|\mathbf{p}_{kl}-\mathbf{p}_{ij}\right\|}\right), \tag{2}$$

where k_{ijkl} is the stiffness of the spring linking ij and kl, and l^0_{ijkl} is the natural length of the same spring.

The **external forces** can differ in nature depending on what type of simulation we wish to make. The forces most frequently included are:

- Gravity: $\mathbf{f}^{gr}_{ij} = m\mathbf{g}$, where $\mathbf{g}$ is the gravity acceleration;
- Viscous damping: $\mathbf{f}^{vd}_{ij} = -C_{vd}(\mathbf{v}_i\text{-}\mathbf{v}_j)$, where C_{vd} is a damping coefficient,
- Collision response.

From the above we may compute the force $\mathbf{f}_{ij}(t)$ applied to point ij at any time t. The fundamental equations of Newtonian dynamics can be integrated over time by a simple Euler method:

$$\begin{aligned}
\mathbf{a}_{ij}(t+\Delta t) &= \frac{1}{m_{ij}}\mathbf{f}_{ij}(t)\\
\mathbf{v}_{ij}(t+\Delta t) &= \mathbf{v}_{ij}(t) + \Delta t\,\mathbf{a}_{ij}(t+\Delta t)\ ,\\
\mathbf{p}_{ij}(t+\Delta t) &= \mathbf{p}_{ij}(t) + \Delta t\,\mathbf{v}_{ij}(t+\Delta t)
\end{aligned} \tag{3}$$

where Δt is a chosen time step. The Euler Equations 3 are known to be very fast and to give good results, provided the time step Δt is less than the natural period of the system $T_0 \approx \pi\sqrt{m/K}$, where K is the highest stiffness in the system. Numerous recent works in cloth simulation have shown that improvements in stability are possible by using implicit integration. However, for complex garments with mapping of Kawabata Evaluation System measurements to the spring properties, explicit integration still proved to be beneficial in terms of efficiency in our case [8]. The advantages of Euler integration became particularly apparent when computation of the collision detection and response, which require small time steps, were taken into consideration.

The implementation of this cloth model on the GPU is described in details in [9]. The collision detection and response is image-space based and is also implemented on the GPU. In this work we use a similar implementation but it is based on CUDA. However, if we want to animate a dressed body the image-based approach is not suitable, because of the arms moving in front and back of the body and in this way the body depth map is shadowed. This approach either has to be modified or a pure object space approach has to be developed.

RAY-TRACING AND ITS APPLICATION FOR CD

Let us assume we have a 3D scene and want to render it in a RGB buffer of size (width x height) pixels. In ray-tracing we cast as many rays as the number of pixels (in our case width x height) from the camera to a rectangle of pixels situated behind the scene. Each ray is traced and collisions are sought with the objects in the scene. If a collision is found with an object, its material shader is called. In OptiX the material shader is a CUDA program that computes the colour of the pixel where the ray hit the object surface. In this way the output RGB buffer is filled with RGB colour values. As OptiX programs are executed on the GPU, the computations for each ray are done in parallel, which increases the performance.

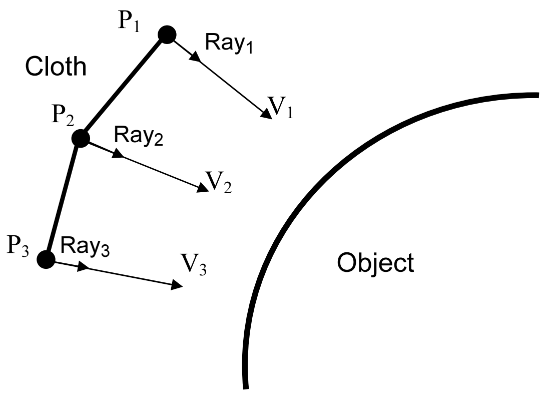


Fig. 2. Cloth mass points and their velocity vectors

To utilise ray-tracing for cloth-body and cloth-cloth CD we do the following. We construct a collision buffer of quadruplets XYZW. The size of the buffer is equal to the number of cloth vertices in the scene. At the beginning of each iteration all buffer values are set to (0,0,0,-1), which means no collision. For each cloth vertex a collision ray is constructed, which starts from the vertex and its direction is along the velocity of the cloth mass point (see fig. 2). The length of the ray is set to some reasonable tolerance value. In our case we experimented and set it to 5 mm. A collision material shader is set for all objects in the scene, including the cloth, for the collision ray type. If an intersection of a ray and a object is found, the collision material shader is called and it writes the following values in the output buffer: XYZ: the coordinates of the normal vector of the object surface at the intersection point; W: the distance from that vertex to the object.

The cloth simulation kernel, which is written in CUDA and has access to the collision buffer, is executed for each cloth vertex. After computing the resultant force on the cloth mass point, it checks the entry in the collision buffer. If there is a collision it computes a proper collision response using the coordinates of the normal vector.

RESULTS

The algorithms were implemented in C++ under Windows 7 and were tested on a laptop with a 2.27 GHz Intel core-i5 processor, 6 GB RAM and NVidia GeForce GT 330 graphics card. The cloth simulation on the GPU was implemented in CUDA. More details about the parallel implementation of the cloth model can be found in [9]. The OptiX ray-tracing engine was used for both detecting collisions cloth-object and cloth-cloth, as described above and for rendering the images. The tests were performed for two pieces of table cloth seaming together and a third piece on top of them draping on a sphere. The sphere is a polygonal model with 578 vertices and 1152 faces.

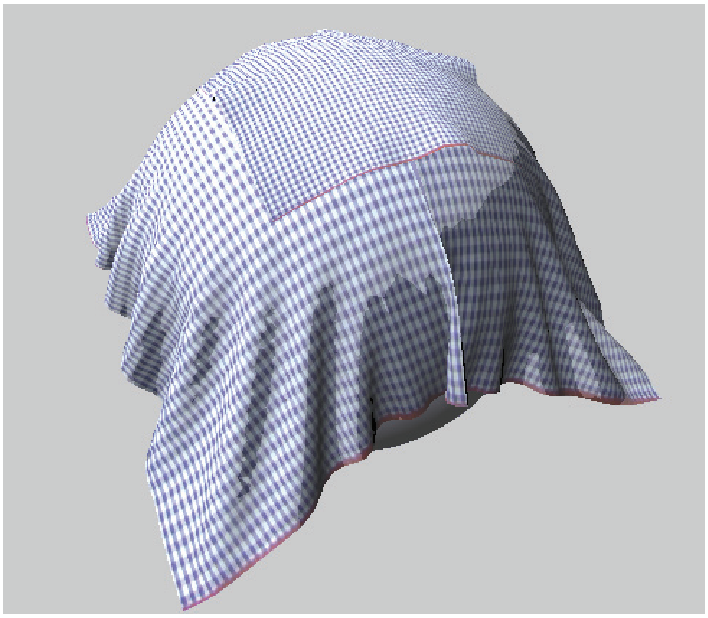


Fig. 3. Two table cloth pieces draping on a sphere

The result of the simulation is shown in Fig. 3. The cloth pieces have a total of 4252 vertices and the simulation ran at 232 iterations per second.

CONCLUSIONS AND FUTURE WORK

This paper presented a ray-tracing approach for cloth-object and cloth-cloth collision detection and response in cloth simulation, implemented on GPU. Its main advantage over the image-space based collision detection is that it can be applied for a dynamic simulation of garments on walking avatars. The following conclusions can be drawn:

- The ray-tracing CD for cloth simulation, described in this paper, produces good results and can be used real applications;
- The current implementation of OptiX has no interoperability with CUDA, which means the OptiX and CUDA buffers cannot be shared directly. This is currently done via OpenGL buffers shared with CUDA and OptiX. This probably slows down the simulation, because of switching contexts and acquiring and releasing the buffers. Hopefully in the next versions this implementation drawback will be overcome.

REFERENCES

[1] Georgii, J., J. Krüger, R. Westermann. Interactive Collision Detection for Deformable and GPU Objects. IADIS International Journal on Computer Science and Information Systems, October 2007, Vol. 2, Number 2.

[2] Georgii, J., J. Krüger, R. Westermann. Interactive GPU-based Collision Detection. IADIS Computer Graphics and Visualization, 2007.

[3] Gress, A., G. Zachmann. Object-space Interference Detection on Programmable Graphics Hardware. SIAM Conference on Geometric Design and Computing, Seattle, Washington, November 13-17 2003.

[4] Gress, A., M. Guthe, Reinhard Klein. GPU-based Collision Detection for Deformable Parameterized Surfaces. Computer Graphics Forum, September 2006, 25:3 (497-506), Presented at Eurographics 2006.

[5] Kim, D., J.P. Heo, J. Huh, J. Kim, S. Yoon . HPCCD: Hybrid Parallel Continuous Collision Detection using CPUs and GPUs. Computer Graphics Forum (Pacific Graphics) 2009.

[6] Knott, D., D. K. Pai. CInDeR - Collision and Interference Detection in Real-time using Graphics Hardware. Proceedings of Graphics Interface 2003, p 73-80. Halifax, Nova Scotia, 11-13 June 2003.

[7] Pabst, S., A. Koch, W. Strasser. Fast and Scalable CPU/GPU Collision Detection for Rigid and Deformable Surfaces. Eurographics Symposium on Geometry Processing 2010, Volume 29, Number 5.

[8] Vassilev, T., Spanlang, B., Chrysanthou Y. Fast Cloth Animation on Walking Avatars, Computer Graphics Forum, 2001, 3 (20), 260-267.

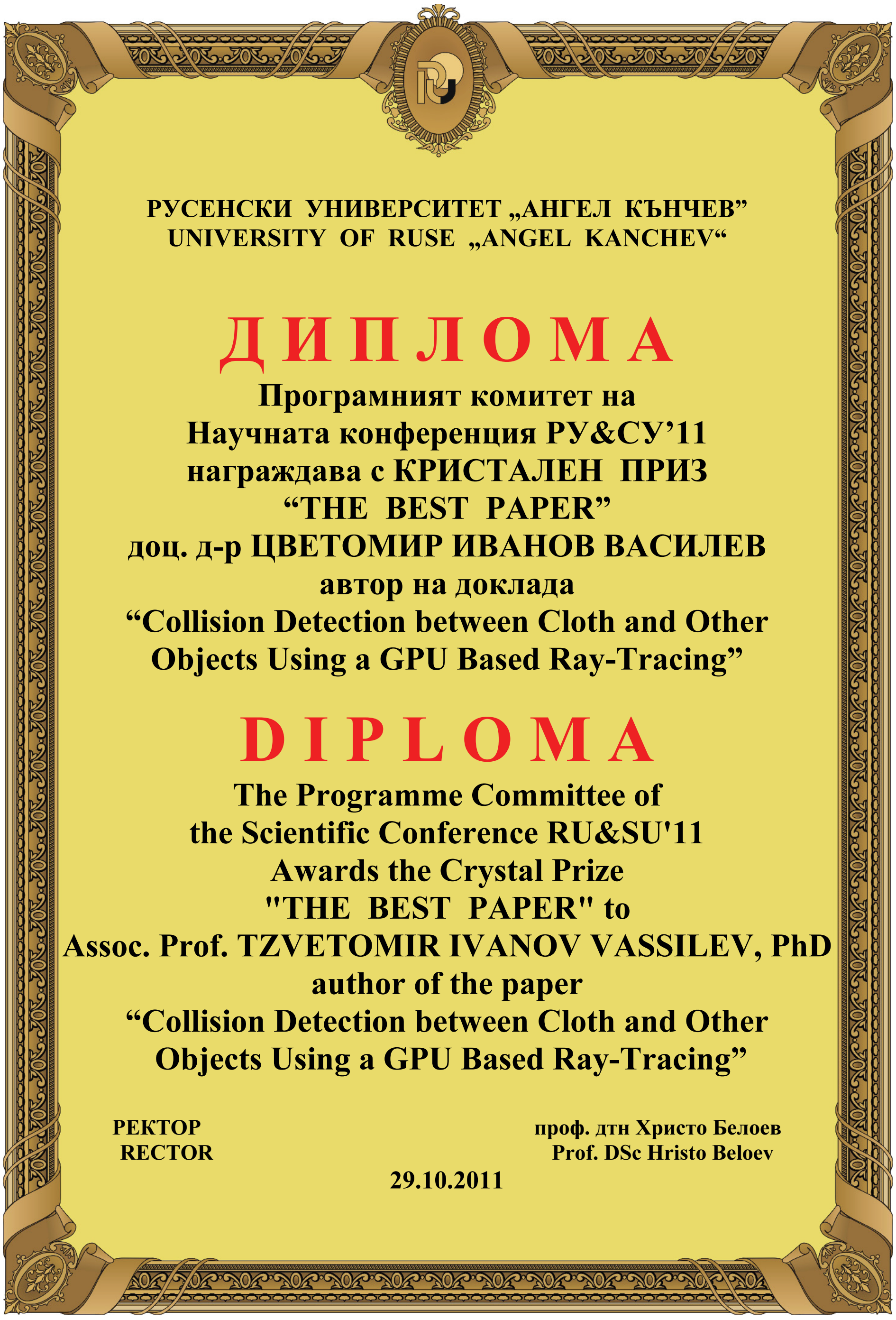
[9] Vassilev, T.I, Rousev, R.I., Algorithm and Data Structures for Implementing a Mass-spring Deformable Model on GPU, Proceedings of the RU conference, November, 2008, Ruse, Bulgaria.

[10] Zhang, X., Y. J. Kim. Interactive Collision Detection for Deformable Models Using Streaming AABBs. IEEE Transactions on Visualization and Computer Graphics Archive, March 2007, Volume 13 , Issue 2, p 318-329, ISSN:1077-2626.

ABOUT THE AUTHORS

Dr. Tzvetomir Ivanov Vassilev, Department of Informatics, University of Ruse, Phone: +359 82 888475, E-mail: TVassilev@uni-ruse.bg.

Докладът е рецензиран.



РУСЕНСКИ УНИВЕРСИТЕТ „АНГЕЛ КЪНЧЕВ”
UNIVERSITY OF RUSE „ANGEL KANCHEV“

Д И П Л О М А

Програмният комитет на
Научната конференция РУ&СУ’11
награждава с КРИСТАЛЕН ПРИЗ
“THE BEST PAPER”
доц. д-р ЦВЕТОМИР ИВАНОВ ВАСИЛЕВ
автор на доклада
“Collision Detection between Cloth and Other
Objects Using a GPU Based Ray-Tracing”

D I P L O M A

The Programme Committee of
the Scientific Conference RU&SU'11
Awards the Crystal Prize
"THE BEST PAPER" to
Assoc. Prof. TZVETOMIR IVANOV VASSILEV, PhD
author of the paper
“Collision Detection between Cloth and Other
Objects Using a GPU Based Ray-Tracing”

РЕКТОР
RECTOR

проф. дтн Христо Белоев
Prof. DSc Hristo Beloev

29.10.2011