FRI-ONLINE-1-MIP-03

REVIEW OF SEVERAL TECHNIQUES FOR ACCELERATING PHYSICAL SIMULATIONS ON THE GPU

Prof. Tzvetomir Vassilev, PhD
Department of Informatics and Information Technologies,
University of Ruse
Phone: +359 82 888 475
E-mail: tvassilev@uni-ruse.bg

Abstract: *This paper reviews several techniques for accelerating physical simulations on the graphics processing unit (GPU). In the current paper they are applied to mass-spring cloth model and cloth-body and cloth-cloth collision detection, but they can be utilized in many other paradigms, where computations can be parallelized. The first technique uses OpenGL GLSL with compute shaders and shader-storage buffers for implementing the entire simulation, cloth model and image space based collision detection, on the GPU. The second utilizes CUDA (OpenCL) for implementing the cloth model with image space based collision detection and visualization in OpenGL. The last one uses again CUDA for the cloth model, but NVidia OptiX ray tracing engine for accelerated collision detection on the GPU. The last approach overcomes some drawbacks of the image space based collision detection. The conclusion compares the three techniques with their advantages and disadvantages and give ideas of possible applications.*

Keywords: *Physical Simulation, Cloth modelling, GPU programming, Parallel programming*

INTRODUCTION

Physical simulation has become very important in computer graphics and has been used to produce realistic animation for several decades now. Very often realistic clothing has to be visualized on human bodies in different types of applications: computer games, garment design and electronic commerce. Garment simulation on a human body usually includes two tasks: first implementing a suitable cloth model and second developing collision detection (CD) technique, which is closely related to the cloth model. The principle of CD in cloth simulation on a human body is to find if a cloth vertex penetrates the body (cloth-body CD) or if there are collisions between cloth pieces (cloth-cloth CD).

This paper reviews several techniques for accelerating physical simulations on the graphics processing unit (GPU). They are applied to mass-spring cloth model and cloth-body and cloth-cloth collision detection, but with some modification they can be utilized in many other paradigms, where computations can be parallelized. In all implementations the model is based on a mass-spring system with velocity modification to constrain elasticity.

Technique 1 uses OpenGL GLSL with compute shaders and shader-storage buffers for implementing the entire simulation, cloth model and image space based collision detection, on the GPU. Technique 2 utilizes CUDA (or OpenCL) for implementing the cloth model with image space based collision detection and visualization in OpenGL. The last one uses again CUDA for the cloth model, but NVidia OptiX ray tracing engine for accelerated collision detection on the GPU. The last approach overcomes some drawbacks of the image space based collision detection

The rest of the paper is organized as follows. The next section reviews previous work on cloth simulation and CD on GPU. Section 3 describes the utilized cloth model. Section 4, 5 and 6 describe in detail the three techniques. Section 6 gives results of the experiment and the last section concludes the paper.

There are many kinds of seismic protection in the field of civil engineering: kinematic foundations, sliding girdle with fluoroplastic for earthquake-proof building, the protecting trench around a building, earthquake-proof buildings, flexible ground floor, rubber isolation bearings (Cooper, A., & Wilson, A., 2002). Seismic forces are directly proportional to the mass of the building and reach their maximum value while resonant vibrations of the system "building-

foundation". Non-traditional methods of isolating the structure from its foundation enable an isolated part of the building to vibrate at a frequency which is different from the frequency of the base (non-isolated) part of the building. Then the phenomenon of resonance of the system "building-foundation" does not occur and seismic forces do not reach their maximum value. Thus, special earthquake protection fights with the causes of the dynamic load - seismic forces produced by the system "building-foundation" (Kotler, P., Haider, D. H., & Rein, I., 1993).

RELATED WORK

Cloth simulation

Physical simulation of elastic objects and surfaces has been addressed by researchers since the end of 1980s. Terzopoulos et al. (Terzopoulos D., J. Platt, A. Barr, K. Fleischer, 1987) proposed elastic deformable surfaces and utilised the finite element method and energy minimisation techniques usually used in mechanical engineering.

A more detailed survey on physically based cloth modelling techniques can be found in several papers (Magnenat-Thalmann N., P. Volino, 2005), (Nealen A., M. Müller, R. Keiser, E. Boxerman, M. Carlson, 2006).

Later some implementations appeared, based on a mass-spring system with explicit integration (Provot X, 1995) and (Vassilev T.I., B. Spanlang, Y. Chrysanthou, 2001), which are simple to implement and with low computational cost. One of their main drawbacks is the super-elasticity, i.e. the computer model is much more elastic than real cloth. To overcome this Provot (Provot X, 1995) proposed to modify cloth vertices positions and Vassilev et al. (Vassilev T.I., B. Spanlang, Y. Chrysanthou, 2001) velocities after each simulation step in order to constrain elasticity.

In the recent years some new simulation methods for deformable models based on geometry emerged. In general they are not as accurate as traditional physics based methods, but produce visually pleasing animations. Müller et al. (Müller M, B. Heidelberger, M. Teschner, M. Gross, 2005) proposed meshless deformation based on shape matching. The method is fairly fast and unconditionally stable. Bender et al. (Bender J., D. Weber, R. Diziol, 2013) used a shape matching technique on two types of regions to model cloth. The 2D triangular areas model resistance to stretching and shearing, while the edges increase the stretching stiffness of the simulated material. However, their model cannot naturally represent resistance to bending with shape matching. Therefore they had to introduce a quadratic bending term, however it does not guarantee unconditional stability in the time integration anymore.

Recently position-based dynamics (M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, 2007) has become popular in the computer graphics applications. Unlike physical simulation these methods compute the positions at each simulation step directly without storing forces and velocities, using external forces and a nonlinear system of constraints. The position-based approaches are stable and controllable and are well-suited for interactive environments. However, they are generally not as accurate as physics-based methods but provide visually pleasing results. So, their main application areas are virtual reality, computer games and special effects in movies and advertisements.

Tang et al (Tang M., R. Tong, R. Narain, C. Meng, D. Manocha, 2013) implemented the mass-spring cloth model of Provot entirely on the GPU. They do not mention how they handle the super elasticity, but their approach can use both explicit and implicit integration, however the time step of implicit integration is not larger in order to handle collisions of complex models.

The system, used in all the three techniques, is based on physically based mass-spring simulation, which manipulates velocities to solve a system of constraints about super elasticity of the cloth and the collision response.

Collision detection

The CD techniques can be divided in two major groups: based on geometrical interference tests in object-space and based on depth buffer interference tests in image-space.

Most common object-space techniques use bounding volumes. They partition space in a set

of boxes or spheres, which allow fast intersection tests. Bounding Volume Hierarchies (BVHs) place these volumes into tree structures to accelerate searching even further. Larsson and Akenine-Moller (Larsson T., T. Akenine-Moller, 2006) built a BVH with dynamically resizing existing Axis Aligned Bounding Boxes (AABBs). Another commonly used structure is the distance or radial fields, which store the shortest distances to a surface in a Cartesian grid (Friston S., A. Steed, 2019). Hermann et al (Hermann E., F. Faure, B. Raffin, 2008) proposed ray-traced collision detection for deformable bodies. A ray is shot from each surface vertex in the direction of the inward normal. A collision is detected when the first intersection belongs to an inward surface triangle of another body. This CD algorithm does not always work well for cloth simulation.

CLOTH MODEL

The cloth model utilised in all the three paradigms is based on the mass-spring model proposed by (Provot X, 1995) and modified by (Vassilev T.I., B. Spanlang, Y. Chrysanthou, 2001). The elastic model of a cloth piece is a rectangular topology mesh of $m \times n$ mass points, linked to each other by massless springs of natural length greater than zero. There are three different types of springs: stretch, shear, and bend, which implement resistance to stretching, shearing and bending.

For each cloth vertex the positions, velocities, and accelerations of the i-th mass point at time t are denoted: $\mathbf{p}_i(t)$, $\mathbf{v}_i(t)$, $\mathbf{a}_i(t)$, i=1,…, m x n. The system is governed by Newton's law:

$$\mathbf{a}_i(t) = \mathbf{f}_i(t) / m_i \tag{1}$$

where m_i is the mass of point $\mathbf{p}_i$ and $\mathbf{f}_i$ is the resulting force applied at that point. The force $\mathbf{f}_i$ is the sum of the internal and external forces.

The internal forces are due to the tensions of the springs. Utilising the Hook's law the overall internal force applied at point $\mathbf{p}_i$ is the sum of the forces caused by all springs linking this point to its neighbours:

$$\mathbf{f}_{int_i}(t) = -\sum_j k_{ij}\Delta\mathbf{l}_{ij} \tag{2}$$

where k_{ij} is a stiffness coefficient of the spring linking $\mathbf{p}_i$ and $\mathbf{p}_j$ and $\Delta\mathbf{l}_{ij}$ is the elongation of the same spring related to its natural length.

The external forces in this cloth simulation are three types: viscous damping ($-c\ \mathbf{v}_i(t)$), gravity and seaming forces applied to the edges to be stitched together.

$$\mathbf{f}_{ext\ i}(t) = -c\ \mathbf{v}_i(t) + \mathbf{f}_{gr} + \mathbf{f}_{seam\ i}(t), \tag{3}$$

The above equations allow to compute the force $\mathbf{f}_i(t)$ applied on the cloth vertex $\mathbf{p}_i$ at any time t. In the original method of Provot and Vassilev et al the fundamental equations of Newtonian dynamics are integrated over time by explicit Euler integration.

Another possibility is to use Verlet integration with half-step velocity as shown in equation 4. This makes it possible to take larger time steps and as a result the simulation should converge faster. The cloth simulation program is implemented as a NVIDIA CUDA kernel and runs entirely on the GPU.

$$\begin{aligned}
&\mathbf{a}_i(t) = \frac{1}{m_i}\mathbf{f}_i(t)\\
&\mathbf{v}_i\left(t + \tfrac{1}{2}\Delta t\right) = \mathbf{v}_i(t) + \frac{1}{2}\mathbf{a}_i(t)\Delta t\\
&\mathbf{p}_i(t + \Delta t) = \mathbf{p}_i(t) + \mathbf{v}_i\left(t + \tfrac{1}{2}\Delta t\right)\Delta t\\
&\mathbf{a}_i(t + \Delta t) = \frac{1}{m_i}\mathbf{f}_i(t + \Delta t)\\
&\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i\left(t + \tfrac{1}{2}\Delta t\right) + \frac{1}{2}\mathbf{a}_i(t + \Delta t)\Delta t
\end{aligned} \tag{4}$$

TECHNIQUE 1

The entire cloth simulation is implemented on the graphics processor using OpenGL shading language (GLSL). GLSL was preferred as the visualization is in OpenGL and this makes data exchange between the simulation and visualization modules easier and faster. In addition, with the compute shaders and shader-storage (SS) buffers available in OpenGL 4.3 and above the general purpose GPU programming is as flexible as in OpenCL and CUDA.

An image-space approach is used for collision detection. Using this technique it is possible to find a collision only by comparing the depth value of the garment vertex with the according depth information of the body stored in the depth maps. The graphics hardware is also exploited to generate the information needed for collision response, that is the normal vectors of each body point. This can be done by encoding vector co-ordinates (x, y, z) as colour values (R, G, B). Depth and normal maps are created using two parallel projections: one of the front and one of the back of the body (or object). For rendering the maps two orthographic cameras are placed at the centre of the front and the back face of the body's bounding box (BB). To increase the accuracy of the depth values, the camera far clipping plane is set to the far face of the BB and the near clipping plane is set to near face of the BB. Both cameras point at the centre of the BB. The depth maps are used to detect collisions and the normal maps to get the normal vector at the collision point, which is needed for the collision response. In case of a static human body the maps are generated only once before the simulation. In case of a moving body the maps have to be generated at each animation step, although if the body movements are known, they can be pre-computed. The main drawback of this approach is that it cannot handle collision detection if there are occlusions, e.g. an arm is in front or at the back of the torso. All the checks for collisions and the corresponding response are implemented on the GPU in GLSL.

TECHNIQUE 2

The cloth simulation is implemented on the GPU, but using NVidia's CUDA API. It can be also done in OpenCL, as the two APIs are very close.

Again the image-space approach is used for collision detection. However all the checks for collisions and the corresponding response are implemented on in CUDA or OpenCL. As the maps are generated using OpenGL this requires that the CUDA (OpenCL) and OpenGL shares their buffers. As shown in the results section this leads to some delays.

The visualization is again implemented in OpenGL, which also requires sharing buffers between OpenGL and the other libraries.

TECHNIQUE 3

The cloth simulation is again implemented on the GPU NVidia's CUDA API.

The collision detection is performed using ray-tracing to detect collisions, which is implemented with NVidia OptiX engine, as described in (Vassilev, T.I., 2012).

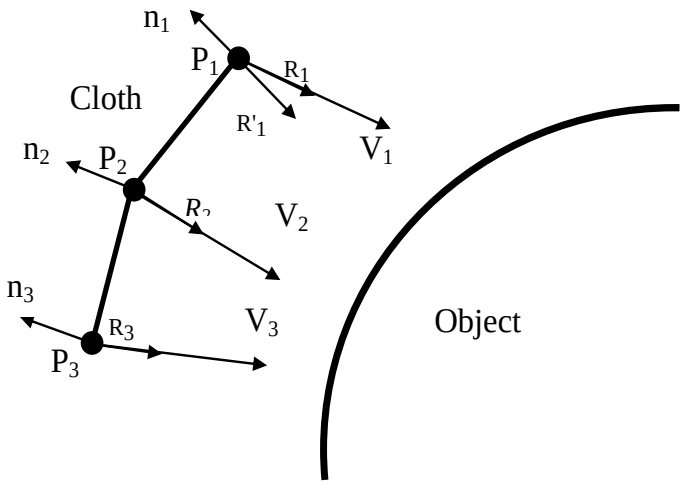

Fig. 1. Cloth vertices and their velocity and normal vectors

Ray-tracing can be utilised for CD as follows. A collision buffer of quadruplets XYZW is created. The size of the buffer is equal to the number of cloth vertices. For each cloth vertex a collision ray is constructed (R_1 in figure 1), which starts from the vertex and its direction is along the velocity of the cloth mass point. The length of the ray is set to some reasonable tolerance value, in our case 3 mm. The longer the ray, the slower the simulation, as intersection with more objects in the scene is detected. The ray is traced and if an intersection with another object (or cloth surface) 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 or cloth surface at the intersection point. The normal vectors at the cloth vertices are computed by the simulation and for the rigid objects in the scene they are pre-computed.
- W: the distance from that vertex to the object, computed by the ray-tracing engine.

If no collision is found for the first, then a second ray is cast (R'_1 in figure 1), which starts from the cloth vertex and its direction is opposite to the normal vector of the cloth surface at that vertex. The ray is traced again as explained above.

The cloth model is implemented in NVidia CUDA as it is easier to communicate with OptiX, which also uses CUDA. The algorithm is shown below.

RESULTS

The three techniques were implemented on a PC with a NVidia RTX 2070 graphics card, Windows 10 and OpenGL 4.6. NVidia Optix 6.5 API was used for technique 3 which has a hardware accelerated ray-tracing on RTX cards.

Performance test were run for the three techniques and time was measured for different numbers of cloth vertices. Results of the comparison is shown in figure 2. One can see that Technique 1 is the fastest and technique 2 is the slowest. However, as explained above technique 1 cannot detect collisions if there are occlusions of body parts.

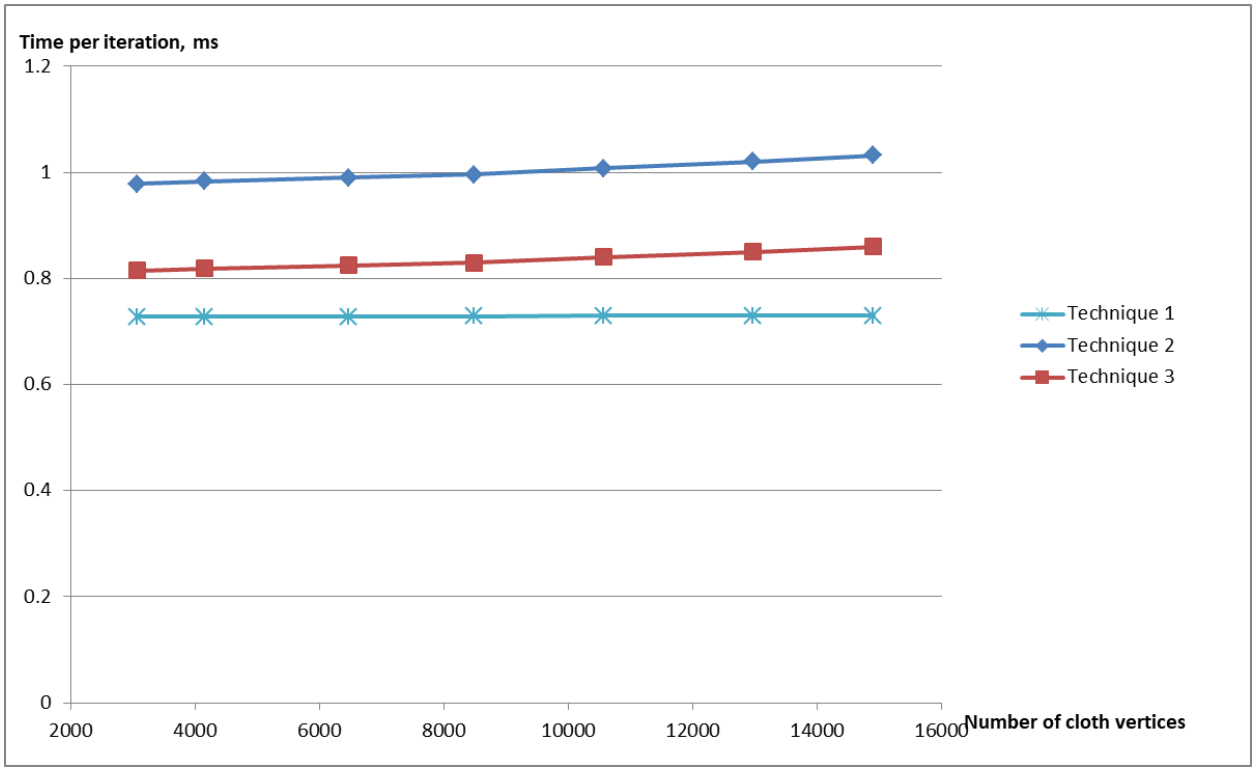

Fig. 2. Time per iteration vs the number of cloth vertices

CONCLUSION

This paper compared three techniques for accelerating physical simulations on the graphics processing unit. The following conclusions can be drawn:

- Technique 1 is the fastest, but significant drawback: cannot handle collisions if there are occluded objects, e.g. an arm is in front or at the back of the torso
- Technique 3 is a bit slower, but more general in detecting collisions
- Technique 2 is the slowest probably because of exchanging buffers between OpenGL and CUDA

REFERENCES

Terzopoulos D., J. Platt, A.Barr, K. Fleischer (1987). Elastically deformable models. *ACM Proceedings of SIGGRAPH* 21 (4), 205–214.

Magnenat-Thalmann N., P. Volino (2005). From early draping to haute couture models: 20 years of research. *The Visual Computer*, 21, 506–519

Nealen A., M. Müller, R. Keiser, E. Boxerman, M. Carlson (2006). Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25 (4), 809–836.

Provot X (1995). Deformation constraints in a mass-spring model to describe rigid cloth behaviour. *Proceedings of Graphics Interface*, 141–155.

Vassilev T.I., B. Spanlang, Y. Chrysanthou (2001). Fast cloth animation on walking avatars. *Computer Graphics Forum* 20 (3), 260–267.

Müller M, B. Heidelberger, M. Teschner, M. Gross (2005). Meshless deformations based on shape matching. *ACM Transactions on Graphics* 24(3), 471–478.

Bender J., D. Weber, R. Diziol (2013). Fast and stable cloth simulation based on multi-resolution shape matching, *Computers & Graphics* 37 (8), 945-954.

M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff (2007). Position based dynamics, *J. Visual Commun. Image Representation*, vol. 18, no. 2, 109–118.

Bender J., M. Müller, M. A. Otaduy, and M. Teschner (2013). Position based methods for the simulation of solid objects in computer graphics, in *Proc. Eurographics* - 2013.

Tang M., R. Tong, R. Narain, C. Meng, D. Manocha (2013). A GPU-based Streaming Algorithm for High-Resolution Cloth Simulation, *Pacific Graphics* 2013, Vol 32 (7).

Larsson T., T. Akenine-Moller (2006). A dynamic bounding volume hierarchy for generalized collision detection, *Computers & Graphics*, 30 (3), 451–460.

Friston S., A. Steed (2019). Real-Time Collision Detection for Deformable Characters with Radial Fields, *IEEE transactions on visualization and computer graphics*, 25 (8), 2019, 2611-2622.

Hermann E., F. Faure, B. Raffin (2008). Ray-traced collision detection for deformable bodies. *3rd International Conference on Computer Graphics Theory and Applications*, GRAPP Funchal, Madeira, Portugal, 293–299.

Vassilev, T.I. (2012). Collision Detection for Cloth Simulation Using Ray-Tracing on the GPU. *International Journal on Information Technology and Security*, No 4, 3-12.

