International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-4 Issue-4, April 2015

GPU based Cloth Simulation for Real Time Interaction using Multiple Haptic Interfaces

Sreeni K. G, Abhijith Joshi

***Abstract*—In this paper we propose a solution to the simulation of a real time deformable cloth for haptic interaction. The simulated environment consists of a deformable cloth, corners of which can be attached to a number of independent haptic devices through a client server mechanism. The users can feel the tensile force which is acting on the cloth due to its own weight through the haptic interface. A ball with a known mass is also rolled over the simulated cloth so as to effect an external force variation on the cloth. The cloth is modeled using a sufficiently dense mass spring model. A Graphic Processing Unit (GPU) is used at the server to speed up computation of cloth motion to make the computation time comparable with the haptic updation time of 1ms. We also use the environment as a possible gaming platform with several players interacting asynchronously using their respective haptics devices.**

***Index Terms*— Haptic rendering, Deformable object, GPU computation, CUDA, Parallelization.**

I. INTRODUCTION

The development of computer graphics technologies have greatly influenced the creation of interactive virtual environments over the past decades. Haptics technology in conjunction with graphics can enhance these environment and can provide a truly immersive experience to the user. Starting from simple non-deformable objects, now we are able to render complex deformable objects. Apart from the research aspects and medical applications, it has also picked up the market in the form of haptically intractable virtual games. In our work we propose a technique to virtually create a deformable cloth using a physically based mass-spring model which can also be used as a virtual gaming platform. Because of the computational requirement of simulating the deformable object with haptic interaction, the environment is simulated in a high performance GPU system. The users can interact with the simulated cloth using a haptic device which is connected to a GPU system using a client server based mechanism. The environment uses typically 4 haptic devices, each of which connected to the 4 corners of the cloth model through springs. Each haptic device connected to its respective client system sends the device position to the server for the updation of the cloth corners.

Manuscript published on 30 April 2015.
***** Correspondence Author (s)
Sreeni K. G, Department of Electronics and Communication, College of Engineering, Thiruvananthapuram, Kerala, India.
Abhijith Joshi, Department of Electrical Engineering, Indian Institute of Technology, Bombay, Powai, Mumbai, India.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

Similarly the reaction force on the haptic device is computed in the server and it sends back to the clients to get the respective haptic force feedback. An external object is also simulated which rolls over the cloth depending on the position of the haptic device. The users may move the haptic device in such a way so as to avoid the ball falling from their side. Once the ball falls off the cloth the game ends and it can be placed back on the cloth to start playing the game again. The users can very well feel the tensile force which is acting on the corners of the cloth through the haptic device when the ball bounces on it. Thus the key challenge in designing the environment involve updating the springs defining the cloth fast enough so that haptic rendering can be done at a rate faster than 1kHz and a graphics rendering at a rate 30 frames per second. We use a GPU based system that computes the spring updates at the server. Fig. 1a illustrates the simulated deformable cloth and the ball during interaction with the haptic device. Here users are connected to the four corners of the cloth which sags due to its own weight and the additional weight of the ball. Depending on the exact positions of these corner points, the ball will roll on this surface and the amount of deformation changes with the progression of the game.

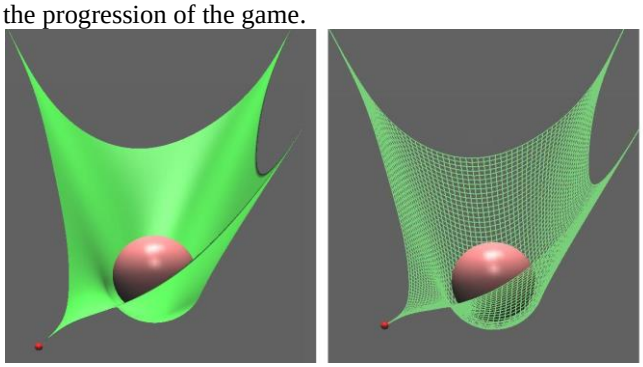


Fig. 1: Graphic illustration of the cloth with the bouncing ball during the interaction (a) cloth surface shaded using a known light source direction (b) mesh structure of the cloth with a size of 60 × 60 nodes for the purpose of clarity of illustration.

II. LITERATURE REVIEW

There are several haptic rendering techniques based on both deformable and non-deformable object models. Generally visual rendering of a simple deformable object like cloth can be simulated with a CPU where the graphic refresh rate is roughly around 30Hz. But when a haptically interacted deformable object is simulated the computation must be carried out at a higher rate since a minimum 1 Khz refresh rate is required by the haptic rendering loop.

© Copyright: All rights reserved.

However, such a speed is difficult to achieve for a deformable object. Hence one often uses local deformation models (such as Gaussian deformation [8]), but these are not physically valid models and works only when the deformation is very small. For an object like a cloth which is amenable to a large amount of global deformation due to haptic interactions, as shown in Fig. 1a, local deformation models do not work. The same argument holds good for a gaming platform that involves handling deformable objects.

Our work is based on real time deformable cloth modeling using a physically based mass-spring model. This application enables multiple users or haptic devices to interact with a single simulated cloth. For illustration we have used a piece of cloth clamped at four fixed locations for interaction through haptic devices and a variable point of interaction based on the rolling of a ball with fixed weight and volume. The computation time may be reduced if the cloth is modeled using only a fewer number of springs, when the interaction is quite non-smooth. For a smoother interaction, one requires a dense grid of springs. This constraint forced us to use a Graphic Processing Unit (GPU) on a separate thread for the computation of deformation of the object.

The haptic research area may be subdivided into three classes- human haptics, machine haptics and computer haptics. A good introduction to the classification is given in [6], [12]. Computer Haptics is a rapidly emerging area of research that is concerned with the techniques and processes associated with generating and displaying the touch and feel of virtual objects to a human operator through a force reflecting device. The process by which desired sensory stimuli are imposed on the user to convey information about a virtual haptic object is called haptic rendering. As user manipulates the haptic device, its position, called the haptic interface point (HIP), and orientation are acquired and the collision of HIP with the object is detected. Once the collision is detected, the interaction force is computed by a rendering algorithm and conveyed to the user through the haptic device. With point based haptic rendering the reaction force on touching an object is calculated as $\mathbf{f} = -K\mathbf{d}$ where K is the stiffness of the object and $\mathbf{d}$ is the vector denoting the amount of penetration of HIP in the virtual object [1]. The problems with single point haptic rendering for rigid objects have been solved by the rendering method (‘god object rendering’) proposed by Zilles and Salisbury in 1995 [18] by introducing the concept of proxy.

The literature in the field of deformation modeling for haptic applications can be divided broadly into two categories: geometry- based deformations and physics-based deformations. Geometry based deformation manipulates the vertices or the volume of an object locally to deform the object and does not focus on the physics involved behind the deformations. Here the contact vertex and the surrounding vertices are displaced according to a Gaussian [8] or a polynomial function to locally deform the surface. The geometry-based deformation can also be modeled using splines. The spline based methods assign control points to a certain volume. During the interaction, these control points are manipulated to bring about a smooth deformation of the object [2], [5], [13]. On the other hand physics-based deformation models work with the physical laws involved behind the haptic interactions. These methods are difficult to implement in real-life applications because of the computational complexity, although the force feedback to the haptic device can be estimated from the model itself during the interaction. Physics based models either use mass-spring models or finite elements models. Each vertex in a mass-spring model, is considered as a particle with a finite mass, attached to adjoining vertices through a network of springs and dampers [16]. With finite element methods (FEMs), the object is divided into small elements. The properties of these elements and the nature of contact forces with other elements are defined and calculated at every step. Even though FEMs are accurate in modeling deformation these models are computationally complex. Modifications are often done to reduce the computation time for FEM models [17]. We focus our attention on mass-spring systems to model deformation due to availability of dampers for smooth haptics interaction. This technique has also been a favorite in computer graphics to model the behavior of cloth [7], [10] soft tissues, muscles [3] and facial expressions [14].

GPU based parallel computing has been very effective in the past few years. Authors in [9] proposes a method of modeling a mass-spring system taking advantage of the parallel processing in the GPU. Also, some alternative implementations were shown in [4] for mass-spring based systems on a GPU. [15] used a GPU-based implementation along with the haptic force feedback to achieve a haptic rate of 450 Hz which is quite less than the required rate of 1Khz. In [11] GPU-based deformation, to achieve both haptic and graphic rendering was shown. In our implementation we demonstrate a mass-spring based deformation with a haptic refresh rate of 1kHz using a GPU.

The parallel GPU based implementation uses Compute Unified Device Architecture (CUDA) developed by Nvidia. CUDA is a general purpose parallel computing architecture that allows to leverage the computational horsepower of Nvidia GPUs using a large number of parallel threads. CUDA has become a very popular parallel computing model since the last decade due to its high computing capability using a cheaper hardware. Now a days, GPUs are very common in personal computers. One can easily use a GPU card to speed up any computationally intensive application which is data parallel in nature. In this paper we are present our work based on the use of such a cheaper GPU to handle computationally intensive physics-based deformation so that each machine can potentially serve a s server. We briefly explain the client server architecture for the developed multiuser haptic interface. If all users are collocated, one single computer suffices to connect several haptic devices independently at different USB ports. If users are connected through Internet, then one of the PCs act as the server and the rest act as clients. If the renderings are performed at individual clients, there is very little communication overhead as each client just needs to broadcast its current HIP position. However, this requires that each client is equipped with a GPU. In our implementation we perform the rendering at the server only and the rendered data is communicated to all clients for haptic interaction and visual display.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

There is, however, a significant communication overhead and the users experience will depend on network delay and delay jitters. In this paper we refrain from discussing the communication issues as our primary focus here is on haptic rendering.

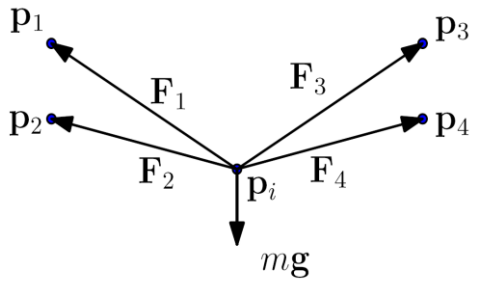


Fig. 2: Illustration of modeling the elemental cloth using a 4-connected spring system.

III. MATHEMATICAL MODELING

In our method, the cloth is modeled using a mass-spring model with each node in the model is connected to a maximum of 4 nearest neighbors through springs as shown in Fig. 2. The nodes on the four sides of the cloths have 3 neighbors except for the corner nodes which have only two. Consider any arbitrary inside node $\mathbf{p}_i$ with mass m. Let the nodes $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ and $\mathbf{p}_4$ be positions of the 4 neighboring nodes of the node $\mathbf{p}_i$. Then the total force $\mathbf{F}_{Ti}$ acting on any inside node $\mathbf{p}_i$ is given by

$$\mathbf{F}_{Ti} = \sum_{j=1}^{4} \mathbf{F}_{ij} + \mathbf{F}_g + \mathbf{F}_{ei} - \beta_1 \mathbf{v}_i \qquad (1)$$

where $\mathbf{F}_{ij}$ denotes the spring force between nodes i and j, $\mathbf{F}_g = m\mathbf{g}$ denotes the gravitational force due to the self weight of the node (a part of the weight of the cloth), $\mathbf{v}_i$ denote the velocity of the node *i* and $\beta_1 < 1$ denotes the damping factor to make the system stable. We have also added an external force component $\mathbf{F}_{ei}$ on each node which is zero unless the cloth is interacting with an external object. This force component is added in order to account for the interaction with the moving ball and is explained in section 3.1. If the natural length, l_N of the spring is known the spring force between any connected node *i, j* can be calculated as

$$\begin{aligned} \mathbf{F}_{ij} &= k_s\left(\left|\mathbf{p}_i - \mathbf{p}_j\right| - l_N\right)\mathbf{u}_{ij} \qquad if \ \ \left|\mathbf{p}_i - \mathbf{p}_j\right| > l_N \\ &= 0 \qquad\qquad\qquad\qquad else \end{aligned} \qquad (2)$$

Here k_s denote the spring constant which is related to the stiffness of the cloth material. The vector $\mathbf{u}_{ij}$ denote a unit vector along the vector directed from node *i* to node *j* Equation (2) assumes that there is no squeezing force on the springs.

Under a steady state (there is no ball placed on the cloth and there is no change in any of the HIPs) all the springs are active because of the non-zero mass attached to the nodes. The acceleration of the reference node $\mathbf{p}_i$ due to the force $\mathbf{F}_{Ti}$ is given by $\mathbf{a}_i = \mathbf{F}_{Ti}/m$. We solve for the position of each node using the Euler equation as given in equation (3).

$$\begin{aligned} \mathbf{v}_i^{(k+1)} &= \mathbf{v}_i^{(k)} + \mathbf{a}_i^{(k)}\Delta t \\ \mathbf{p}_i^{(k+1)} &= \mathbf{p}_i^{(k)} + \mathbf{v}_i^{(k+1)}\Delta t \end{aligned} \qquad (3)$$

where $\mathbf{a}_i^{(k)}, \mathbf{v}_i^{(k)}$ and $\mathbf{p}_i^{(k)}$ denote the acceleration, velocity and position of the node *i*, respectively, in the k^{th} iteration. The value Δt denotes the time to update the node positions. The movement of any node in the model affects the resultant force on the neighboring nodes through equation (1) and the movement spreads to the remaining part of the cloth quickly. Since force on each node $\mathbf{p}_i$ is computed independently we use parallel computing with help of a GPU to reduce the computation time which is essential in any haptic feedback system.

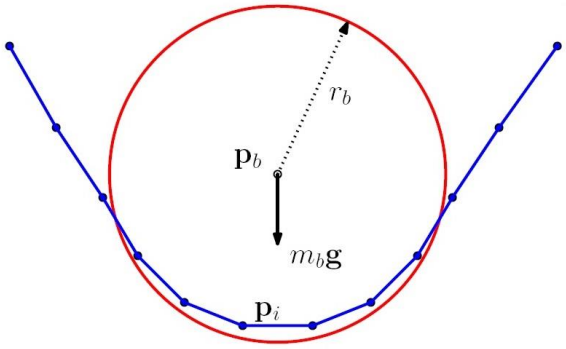


Fig. 3: Illustration of interaction of ball with the cloth.

A. Interaction between the Cloth and the External Object

In order to illustrate the interaction of the ball of radius r_b with the cloth, consider the mass-spring model in 2D as shown in Fig. 3. During interaction with the ball, the cloth nodes try to penetrate into the ball as shown. The amount of penetration of nodes inside the ball depends on the stiffness of the ball. Let m_b and k_b denote the mass and stiffness of the ball, respectively. The external force $\mathbf{F}_{ei}$ as given in equation (1) on each penetrated cloth node is given by

$$\mathbf{F}_{ei} = k_b\left(r_b - \left|\mathbf{p}_i - \mathbf{p}_b\right|\right)\frac{\mathbf{p}_i - \mathbf{p}_b}{\left|\mathbf{p}_i - \mathbf{p}_b\right|} \qquad if \ \ \left|\mathbf{p}_i - \mathbf{p}_b\right| < r_b \qquad (4)$$

If M denotes the number of nodes penetrated inside the ball, then the total force on the ball is given by

$$\mathbf{F}_b = \mathbf{F}_{gb} - \sum_{i=1}^{M} \mathbf{F}_{ei} - \beta_2 \mathbf{v}_b \qquad (5)$$

where $\mathbf{F}_{gb} = m_b\mathbf{g}$, denote the gravitational force due to the weight of the ball and $\beta_2\mathbf{v}_b$ denote the damping term $(\beta_2 < 1)$ proportional to the ball velocity $\mathbf{v}_b$. The equations of motion for the ball also can be solved using the Euler equation as given in equation (6)

$$\begin{aligned} \mathbf{v}_b^{(k+1)} &= \mathbf{v}_b^{(k)} + \mathbf{a}_b^{(k)}\Delta t \\ \mathbf{p}_b^{(k+1)} &= \mathbf{p}_b^{(k)} + \mathbf{v}_b^{(k+1)}\Delta t \end{aligned} \qquad (6)$$

where $\mathbf{p}_b$ is the position of the center of the ball and $\mathbf{a}_b = \mathbf{F}_b/m_b$ denotes the acceleration of the ball.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

IV. HAPTIC AND VISUAL RENDERING

The haptic devices are connected to the corners of the deformable cloth using springs. Movement of any haptic device causes movement of the corresponding cloth corner to which it is connected. The force on the haptic device is computed using Hook's law. If $\mathbf{h}_p$ denotes the Haptic Interface Point (HIP) and $\mathbf{c}_p$ denotes the position of the node to which it is connected, the reaction force on the haptic device can be computed by equation (7).

$$\mathbf{F}_h = -K(\mathbf{h}_p - \mathbf{c}_p) \tag{7}$$

where K denotes the stiffness of the spring attached to the haptic device. Here we assume natural length of the spring to be zero.

The simulated cloth is visually rendered as a mesh as shown in Fig. 1b with the help of a 2D grid matrix. This representation in 2D also helps to find the neighborhood points. The visual rendering loop runs on a separate thread with a refresh rate of 30 Hz. As discussed in section 3 the positions of all the nodes are computed in a GPU to accelerate the computation. During the interaction the cloth bends or folds and hence the normal at the nodes are also computed dynamically before being displayed on the screen. The normal $\mathbf{n}_i$ at any internal node $\mathbf{p}_i$ is computed as

$$\mathbf{n}_i = \frac{\mathbf{p}_{i1} - \mathbf{p}_{i2}}{|\mathbf{p}_{i1} \times \mathbf{p}_{i2}|} \tag{8}$$

This computed normal further helps in providing shades to the cloth with a known light source direction. An example of such a graphic rendering has been shown earlier in Fig. 1a.

V. CUDA BASED GPU-COMPUTATION

GPU-based high performance computers play a significant role in large-scale modeling. The model for GPU computing must use a CPU and GPU/GPUs together in a heterogeneous computing model. GPUs are organized into a number of Streaming Multiprocessors (SM), which are in turn composed of number of Streaming Processors (SP), that share control logic and instruction cache. The number of SMs and the number of SPs in these SMs vary with different GPU architecture. CUDA C is an extension of C language that allows programmer to define C functions, called kernels, that are executed N times in parallel by N different CUDA threads, as opposed to regular C functions that execute only once. A kernel is defined using the `__global__` specifier. The number of CUDA threads that execute the kernel for a given kernel call is specified using a new `(<<<. . . >>>)` execution configuration syntax. Each thread that executes the kernel is given a unique thread ID accessible within the kernel using the built-in "`threadIdx`" variable.

In CUDA "`threadIdx`" is a 3-component vector, that can identify threads within a thread block. This provides a natural way to compute across the elements in a domain such as a vector, matrix or volume. Number of threads per block are limited, since all threads of a block should reside on the same processor core and must share the limited memory resources of that core. A kernel can also be executed by multiple similar looking thread blocks, so that the total number of threads is equal to the number of threads per block, times the number of blocks. These blocks are organized into a grid of thread blocks.

A. Parallel Algorithm

The batch of threads that execute a kernel is organized as a grid of thread blocks. The following pseudo code will be implemented inside the kernel:

Algorithm 1 CUDA Kernel Pseudo Code

```
for (All neighboring mass nodes) do
    Compute internal forces
    Add external forces
    Compute velocity of the node
    Update of node position
end for
```

B. Implementation

Cloth is a 2D structure of $N \times N = N_v$ nodes. The computation at each node consists of spring forces depending on neighboring nodes, gravitational force and the external force. This procedure of calculation and update is done for all the N_v nodes. The algorithm for this situation is highly data-parallel and therefore a good match for CUDA based GPU programming model. We parallelize then solver by executing blocks of hundreds of threads that span the whole mass-spring system. Computations for every point of the cloth are performed by separate threads. Force calculations in the cloth model require node position vectors, and the other constants to be transferred from CPU to GPU device. These are transferred to the GPU device only once. The kernel function executes N_v number of threads, one per node. Every thread calculates the effective force, velocity and position vector for the corresponding node and stores it in the global device memory.

Out of a large number of threads, many threads are simultaneously writing to the shared array in the GPU global memory. At the same time other threads might be reading values of the neighboring nodes. Here, a condition may arise that a node may read older value from one neighbor and newer value from the other. In normal situation, it is expected that all the threads should read all the neighboring values which are computed in last kernel execution. To avoid reading newer values, in the current execution, we use a technique similar to double buffering, which is used in computer graphics. We use two arrays, one for storing results from current execution and the other, which is already filled up in previous execution. After each execution these arrays are swapped. Typical implementation transfers the input data from CPU to GPU, executes the kernel function and again transfers the results back to the CPU. In our implementation all computations are done on the GPU and only the required data is transferred back to the CPU. A frame rate of 1KHz is to be maintained for the haptic force feedback. Hence only 4 corner points, connected to the haptic devices, are transferred back to the CPU, per kernel invocation. For graphic rendering, the position vectors are transferred only after 30ms to maintain a graphic frame rate of approximately 30Hz.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

The pseudo code for the parallel implementation is described below:

In Fermi architecture, each SM can accommodate maximum of 1536 threads simultaneously, at a given instance of time. Also each SM has 32786 number of 32-bit registers with a maximum of 64 registers per thread. The kernel, in our implementation uses 31 registers per thread for its execution. With the above mentioned limitations only 1024 threads (4-blocks, 256 threads/block) are being launched.

$$\text{Occupancy} = \frac{\text{No of threads in SM for given kernel}}{\text{Max. no. of threads in SM}} = \frac{1024}{1536} = 0.67$$

Algorithm 2 Parallel Implementation Pseudo Code

```
loop
   deform_kernel<<<blocks,threads>>> (arguments)
   Get the data from GPU to CPU.
   Update the ball position.
   Update the haptic feedback force.
   if (time elapsed == 30ms) then
      Get the new node positions of the cloth
      Update the normal at each cloth nodes
      Update the graphics output
   end if
end loop
```

Occupancy plays an important role in achieving the peak performance out of the given CUDA/GPU hardware. The more the occupancy, more will be the number threads that can reside in an SM. These threads will be ready to run, when a running thread is requesting a time consuming operation such as memory access. The switching between the threads in such circumstances results into improved performance. The low occupancy in our application kernel, therefore becomes the performance limiter. The ball position is updated on the host side. Hence we need to transfer the data required for this update to the host, which is also one of the performance limiters.

We use a 3-DOF Falcon haptic device from NOVINT for haptic rendering which is having roughly a 10 cm cube interaction space. For haptic interaction we use the API called HAPI and the virtual environment is visually rendered with OpenGL.

VI. PERFORMANCE RESULTS

The algorithm was implemented and tested on the following systems.

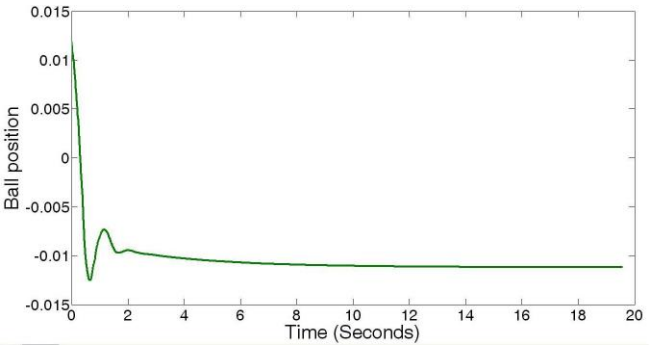
(1) CPU: Intel core i7-920 processor, 2.66 GHz, 8 GB RAM
(2) GPU: Nvidia Graphic Card 560 Ti Fermi-architecture 384cores (256 threads/block)

Table 1 compares the time taken by CPU and GPU for updating the cloth with different cloth sizes. The timing refers to the thread that computes only the haptic rendering part. It can be seen from the table that the speed up factor with a GPU increases as the number of nodes increase. With a cloth size of 25 × 25 nodes the cloth can be updated very fast (within 0.38 ms), but the visual and haptic rendering will not be smooth due to the lack of an adequate number of points. It is not possible to have a smooth bend in the cloth when shown graphically with a less number of nodes. As the number of nodes increase, the graphic rendering becomes better and so does the haptic experience. But on the other hand it increases the cloth updation time which results in a sluggish haptic feedback, jeopardizing the user experience. For cloth dimensions upto 85×85 nodes the GPU time is less than 1 ms which gives an update rate of 1Khz while using double precision computation. With single precision computation the number of nodes can be as large as 135 × 135. Experimentally we found the single precision to be good enough for haptic rendering purposes, where one does save significantly in computation time.

Cloth size	CPU time (ms)	GPU time (ms)	Speed up DP	GPU time SP(ms)
25x25	2.34	0.38	6.15	0.35
50x50	9.38	0.43	21.82	0.36
85x85	27.64	1.01	27.36	0.44
100x100	37.79	1.31	28.85	0.85
135x135	60.06	1.91	31.44	0.97
150x150	84.98	1.97	43.14	1.25
200x200	151.53	3.27	46.34	1.85
250x250	240.59	4.7	51.19	2.24
300x300	343.65	5.92	58.05	2.53
350x350	468.66	8.76	53.5	3.15
400x400	612.58	10.56	58.01	3.62
450x450	771.68	13.78	56	4.11
500x500	956.15	15.92	60.06	5.15

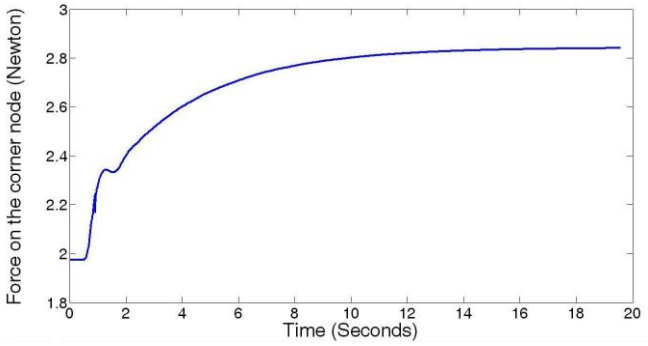
Table 1. : Computational time only for haptic rendering part (DP- Double Precision, SP- Single Precision).

The haptic device that we used for interaction with the cloth is having an interaction space of roughly 10cm cube. A cloth size of (50 × 50) to (70 × 70) nodes was enough for the current interaction space. In future, one may choose to use larger mesh size for a large interaction space. So we have added the computation comparison upto a size of 500 × 500 nodes in table 1. With a cloth size of 500 × 500 nodes an update frequency of approximately 200Hz can be achieved. In order to illustrate the change in reaction force on the haptic device, the ball is allowed to fall freely on a steady cloth. The force on the haptic device as well as position of the ball were recorded for this scenario. Fig. 4a illustrates position of the ball during the interaction. As the ball touches the cloth, it bounces and settles down. The corresponding reaction force on the haptic device is shown in Fig. 4b.



(a)

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.



(b)

Fig. 4: Illustration of the rendered force during interaction with the ball (a) position (y-coordinate) of the freely falling ball (b) magnitude of force on the haptic device (connected to one of the corner node) Vs time.

VII. CONCLUSIONS AND FUTURE WORK

In this paper we discuss the simulation of a real time deformable cloth with a haptic interface. The simulated environment consists of a deformable cloth, corners of which can be attached to a number of independent haptic devices. The users can feel the tensile force which is acting on the cloth corner through the haptic interface. A ball with a known mass and a given radius is also rolled over the simulated cloth so as to feel the external force variation in the cloth. The deformable cloth is modeled using a mass spring model. We use a Graphic Processing Unit (GPU) to speedup computation of cloth motion to make the computation time comparable with the haptic updation frequency. Thus we could increase the number of nodes in the cloth without compromising for the haptic computational requirement with the help of GPU.

In the current implementation structural spring model which has 4 neighboring nodes connected to every node is used. In case we want to model shearing force also, we will have 8 neighboring nodes for every node. This will however increase the computational requirement. This can be easily dealt with using GPU and will be explored further.

REFERENCES

[1] C. Basdogan, S. D. Laycock, and A. M. Day. 3-Digree of Freedom Haptic Rendering.
[2] S. Coquillart. Extended free-form deformation: a sculpturing tool for 3D geometric modeling. In SIGGRAPH'90, pages 187–196, NY, USA, 1990.
[3] D. d'Auliganc, R. Balaniuk, and C. Laugier. A haptic interface for a virtual exam of the human thigh. In Robotics and Automation, ICRA'00, pages 2452–2457, San Francisco, CA, USA, April 2000.
[4] Randima Fernando. GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics. Pearson Higher Education, 2004.
[5] William M. Hsu, John F. Hughes, and Henry Kaufman. Direct manipulation of free-form deformation. In Computer Graphics (SIGGRAPH'92), pages 177–184, NY, USA, 1992.
[6] S. D. Laycock and A. M. Day. A survey of haptic rendering techniques. Computer Graphics Forum, 26(1):50–65, March 2007.
[7] J. Liu, M. Ko, and R. Chang. A simple self-collision avoidance for cloth animation. Computers and Graphics, 22:117–128, 1998.
[8] C. J. Luciano, P. P. Banerjee, and S. H. R. Rizzi. GPU-based elastic-object deformation for enhancement of existing haptic applications. In Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering, Scottsdale,AZ, USA, September 2007.
[9] Jesper Mosegaard and Thomas Sangild Sorensen. GPU accelerated surgical simulators for complex morphology. In Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality, VR '05, pages 147–154, 323, Washington, DC, USA, 2005. IEEE Computer Society.
[10] Hing N. Ng and R. L. Grimsdale. Computer graphics techniques for modeling cloth. IEEE Computer Graphics in Textiles and Apparel, 16:28–41, 1996.
[11] M. De Pascale, G. De Pascale, D. Prattichizzo, and F. Barbagli. A GPU-friendly method for haptic and graphic rendering of deformable objects. In Proceedings of EuroHaptics, pages 146–151, Munich Germany, June 5-7 2004.
[12] J. Kenneth Salisbury and Mandayam A. Srinivasan. Phantom-based haptic interaction with virtual objects. IEEE Computer Graphics and Applications, 17(5):6–10, 1997.
[13] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In SIGGRAPH'86, pages 151–160, NY, USA, 1986.
[14] D. Terzopoulos and K. Waters. Physically-based facial modeling analysis, and animation. Visualization and Computer Animation, 1:73–80, 1990.
[15] Srensen T.S. and Mosegaard J. Haptic feedback for the GPU-based surgical simulator. In Proceedings of Medicine Meets Virtual Reality, pages 523–528., 2006.
[16] A. Witkin. An introduction to physically based modeling: Particle system dynamics. Technical report, School of Computer Science, Carnegie Mellon University, 1997.
[17] Y. Zhuang and J. F. Canny. Haptic interaction with global deformations. In ICRA'00, pages 2428–2433, 2000.
[18] C.B. Zilles and J.K. Salisbury. A constraint-based god-object method for haptic display. Intelligent Robots and Systems, IEEE/RSJ International Conference on, 3:3146, 1995.

Sreeni K. G. received his B. Tech degree in Electronics from the College of Engineering, Adoor, Kerala, India in 2000. He received his M. Tech degree in Power Electronics from the National Institute of Technology, Calicut, India in 2003. He received his Ph.D from IIT Bombay in the year 2013. Now he is working an Assistant Professor in College of Engineering, Thiruvananthapuram, India. His research interests include developing algorithms for haptically rendering virtual objects.

Abhijith Joshi received his B. Tech degree in Electrical & Electronics Engineering from Visvesvaraya National Institute of Technology, Nagpur, Maharashtra, India in 2009. He received his M. Tech degree in Electrical Engineering from the Indian Institute of Technology, Bombay in 2012. Currently he is working in Qualcomm India Pvt. Ltd., Bangalore as a Sr. Digital Design Engineer. His research interests include High Performance Computing, Algorithms, Low power Digital Design.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.