Scientific Journal Impact Factor (SJIF): 1.711
International Journal of Modern Trends in Engineering and Research
IJMTER
ISSN (ONLINE) : 2349 - 9745
ISSN (PRINT) : 2393 - 8161

Real Time Simulation of Cloth Like Object

Ashish V. Kumbhar, Amit A. Ghadage, Akash K. Shinde

[1]*CSE, DACOE, Karad,*
[2]*CSE, DACOE, Karad*
[3]*CSE, DACOE, Karad*

Abstract— Modeling and animation of cloth has experienced important developments in recent years. Concept regarding to simulation of cloth can be implemented by using both OPENGL and CUDA C. So we can examine how to incorporate effects of wind fields in cloth simulations. Use of CUDA C language helps to do execution faster. CUDA C is a GPU Programming language we have used which run on our nvidia graphics processor. We perform fast and stable time integration of a physically-based model, as well as wind drag effects to enhance realism.
Keywords-Wind Simulation, Cloth Animation, Physically Based Modeling, Parallel computing, Computational model, CUDA (Compute unified device architecture).

I. INTRODUCTION

While the simulation of wind is an area of vast interest in aerodynamic engineering, however aerodynamic effects are obviously capable of enhancing the realism of an animated scene and thus are an important part of a cloth simulation system. Simulation concept is also very useful for making video games so that game animation looks like a reality.

II. EXISTING SYSTEM AND RELEVANCE

Concept of cloth simulation had developed by using OpenGL but coding of OpenGL requires extra efforts to calculate coordinate system. Coordinate system has three dimensional aspects. To calculate 3D points in OpenGL, transformation, rotation, scaling methods are used and mathematical equations calculate coordinates of 3D object. These equations based on various tricky algorithms. To show final animation, CPU requires more time to calculate that coordinate system. Due to this problem, animation hasn't real time effects.

III. PROBLEM DESCRIPTION

The major advantage of OpenGL based approaches consists in the fact that the evolution of the wind flow over time is calculated. It enables us to model global effects like convection and diffusion on a physical basis. We present a model to exploit these wind models for calculating the interaction of deformable objects with the air flow by a boundary condition method. As already stated by paper [4] "a velocity field of its own isn't visually interesting until it starts moving objects", in our case clothes with the wind. On the one hand the wind deforms the objects which on the other hand change the wind flow.

For every deformable object the velocity value of the surrounding wind field for every vertex of the representing mesh is tracked. The wind velocity at the vertex positions of the object is recorded. Additionally, the normals of these vertices are stored. Finally, for every marked cell in the scene the previously stored normals are averaged in one space cell which are used to update the velocity. Thus, the boundary conditions are met and yet aerodynamic forces are obtained. A different issue is how to deal with the inside of (rigid) objects. The method to set boundary conditions as described above does not account for the interior of objects. The path of the wind flow is checked for object intersection, whereby the collision detection of the cloth simulation system provides a simple method to deal with this issue [1].

Particle calculation on Wind Fields

Here we combine the idea of creating wind fields by predefined flow primitives with particle calculation in given flow fields. To define a wind scene we first built up the air flow by simple primitives such as parallel directed wind fields, vertices, etc. We then use a particle calculation method in the defined wind field to determine windless areas. This method is very easy to implement and yields very plausible and nicely looking results.

Global Wind Fields [4]

A simple approach to generate complex air flows is to define a wind field by mathematical functions which assign to each point in space a unique velocity value. While this works for simple objects, this approach is not feasible at all with deformable objects like textiles. It will not give real time effects as better as possible. Another more serious drawback of OpenGL-based model for our application consists in the lack of interaction with objects. The wind flow defined by the primitives will not react on objects in the scene which means for example that tissues in the lee of other objects will be affected by the wind flow as well. To solve the described problems we propose a model which combines the simple global wind flow techniques with a particle calculation method through CUDA C. Here, particles are moved along the wind field to determine the effect of objects in the scene.

We solved above problem by using following architectural use:

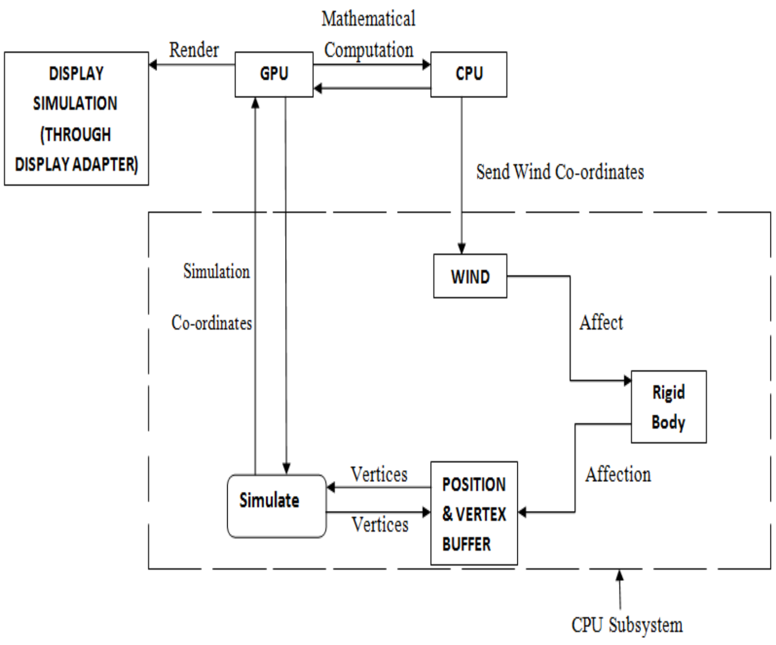


Fig.1 Architectural diagram

IV. IMPLEMENTATION

Particle Calculation through OpenGL:

Two-Dimensional Evaluators:

In two dimensions, everything is similar to the one-dimensional case, except that all the commands must take two parameters, u and v, into account. Points, colors, normals, or texture coordinates must be supplied over a surface instead of a curve. Mathematically, the definition of a Bezier surface patch is given by

$$S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u)\, B_j^m(v)\, P_{ij}$$

Where, *P*ij are a set of *m***n* control points, and the *B*i are the same Bernstein polynomials for one dimension. As before, the *P*ij can represent vertices, normals, colors, or texture coordinates [2].

Two-Dimensional Example:
A Bezier Surface draws a wireframe Bezier surface using evaluators, as shown in Figure 2 below. In this example, the surface is drawn with nine curved lines in each direction. Each curve is drawn as 30 segments.

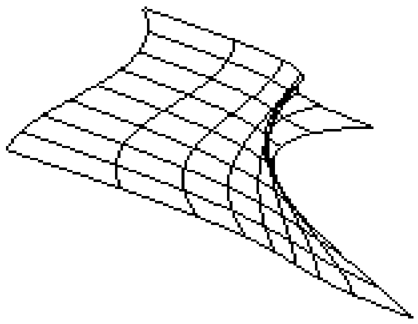


Fig.2 Wireframe object

The GLU NURBS Interface
The GLU provides a NURBS (Non-Uniform Rational B-Spline) interface built on top of the OpenGL evaluator commands [2]. The basis function is a cubic B-spline, but the knot sequence is nonuniform, with a multiplicity of 4 at each endpoint, causing the basis function to behave like a Bezier curve in each direction. Figure 3 shows the surface as a lit wireframe.

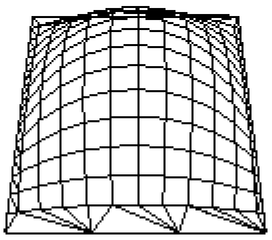
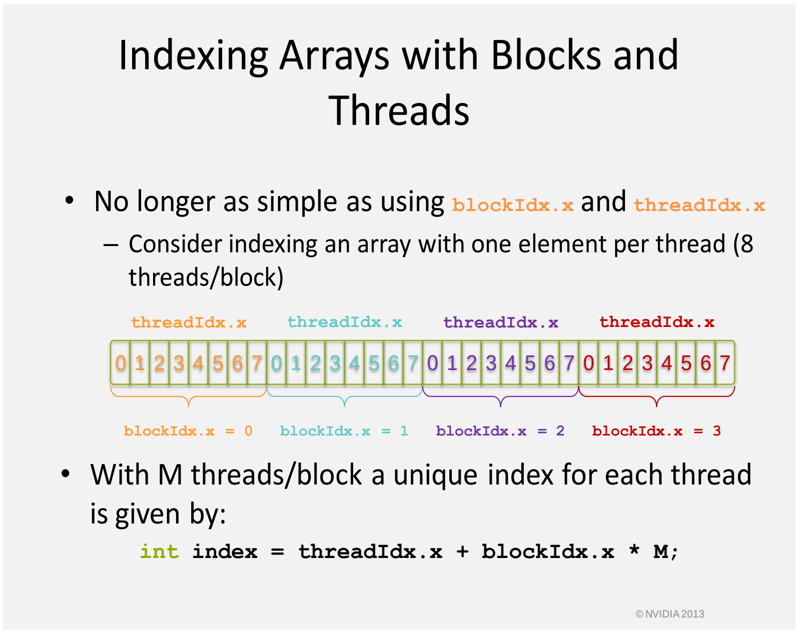


Fig. 3 Nurbs surface example

Parallel Computing Calculation:



Indexing Arrays with Blocks and Threads

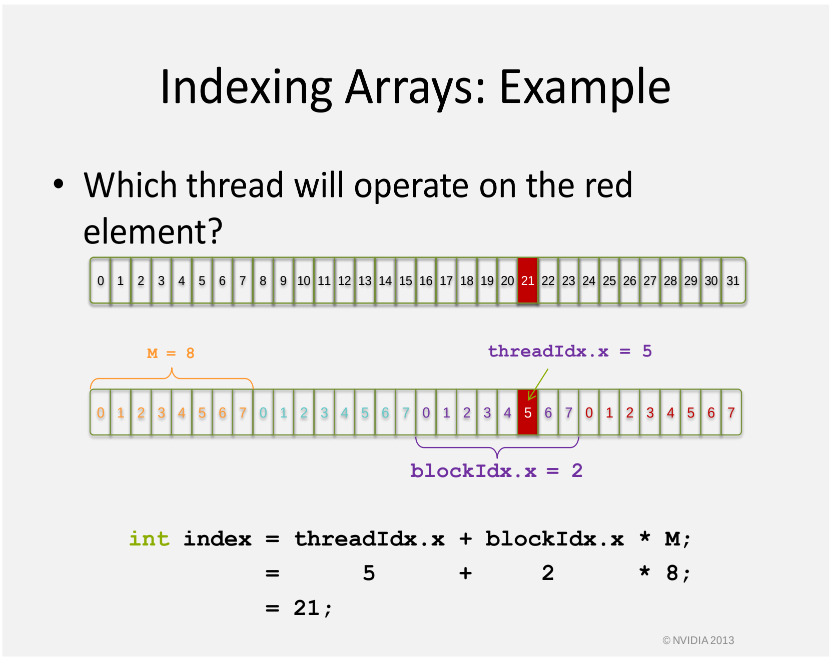
- No longer as simple as using `blockIdx.x` and `threadIdx.x`
  - Consider indexing an array with one element per thread (8 threads/block)

threadIdx.x
threadIdx.x
threadIdx.x
threadIdx.x
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
blockIdx.x = 0
blockIdx.x = 1
blockIdx.x = 2
blockIdx.x = 3

- With M threads/block a unique index for each thread is given by:

```
int index = threadIdx.x + blockIdx.x * M;
```

© NVIDIA 2013



Indexing Arrays: Example

- Which thread will operate on the red element?

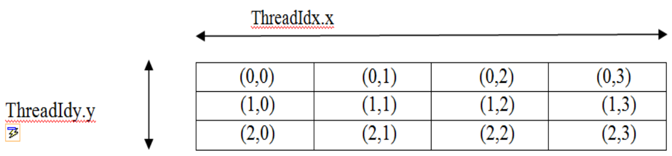
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
M = 8
threadIdx.x = 5
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
blockIdx.x = 2

```
int index = threadIdx.x + blockIdx.x * M;
          =      5      +     2      * 8;
          = 21;
```

© NVIDIA 2013

Cloth Particals Calculation:

int index = threadIdx.y + threadIdx.x * blockDim.y + blockIdx.x * blockDim.x * blockDim.y;



(0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)
(2,0)	(2,1)	(2,2)	(2,3)

V. RESULT ANALYSIS

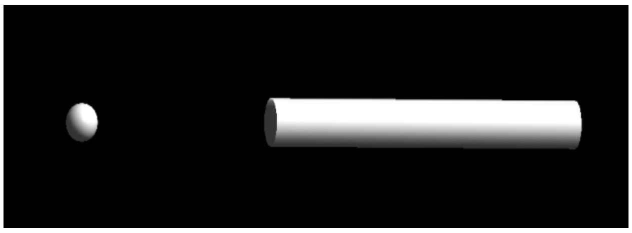


Fig. 4 Implementation screenshot 1

The fig.4 indicates that a ball thrown from a shooter goes towards cloth, collide over cloth and drop on the ground. We can see simulation of overall phenomenon with uninterrupted visualization as shown in above figure.

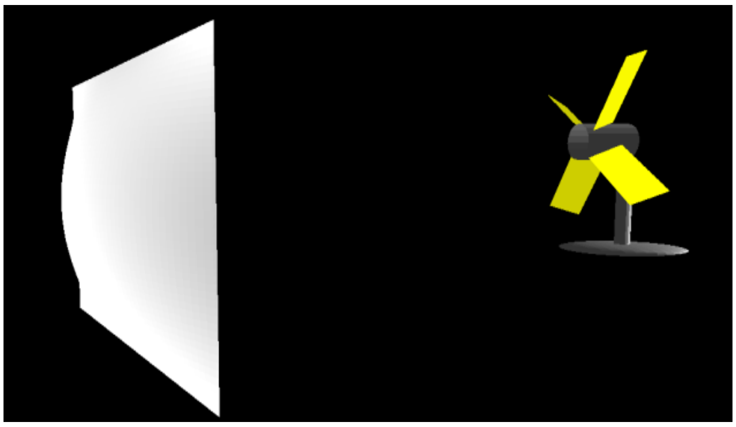


Fig.5 Implementation screenshot 2

The fig. 5 shows that how cloth gets affected by wind of fan which is placed forefront of cloth. We have made two models of above simulation. One is made up with OpenGL language only and other is of both OpenGL & CUDA C. We finally compared execution of both model. We saw that Visualization through second model is better than first model.

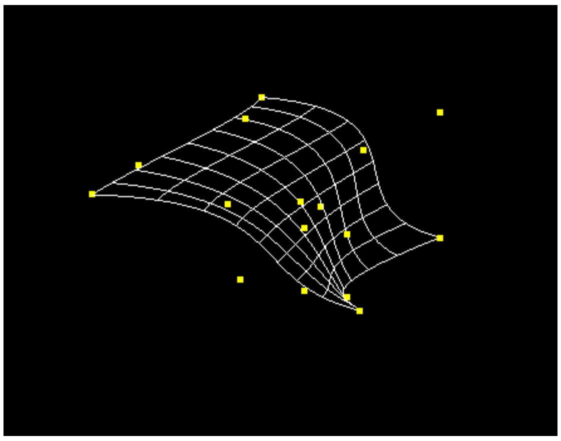


Fig.6 Implementation screenshot 3

Fig.6 shows cloth particles which will help to simulate cloth according to wind force applied by fan.

VI CONCLUSION AND FUTURE SCOPE

We have implemented simulation of cloth, by using OpenGL programming for graphical demonstration. There is visualization problem during simulation of model. This problem overcame by implementing CUDA C parallel computing.

REFERENCES

[1] Cloth Simulation Parameters from Video. In *Proc. SIGGRAPH Symposium on Computer Animation*, 2003.
[2] OpenGL Programming Guide (Addison-Wesley Publishing Company), Jan 1997.
[3] GPU Programming Guide Nvidia Publication, Oct. 2012
[4] M. Keckelsen, S. Kimmerle, and M. Wacker, “Modelling Effects of Wind Fields in Cloth Animations”, IEEE, 2003.
[5] OpenGL Super Bible!
[6] Getting Started with OpenGL, Peter Grogono, 1998, 2002, 2003.
[7] http://visualambition.wordpress.com/
[8] CUDA By Example (An introduction to general-purpose GPU programming) Jason sanders, edward Kandrot
[9] NVIDIA CUDA Reference Manual (Version 3.2 Beta), August 2010.
[10] Introduction to CUDA C, NVIDIA Corporation 2013.
[11] https://developer.nvidia.com/