**Simulating Cloth Using DirectX 11**

***Abstract***

*This report is a brief exploration into cloth simulation. Covering Verlet integration initially and how it applies forces to individual particles. The construction of a simulation of cloth is then discussed, with an explanation of constraints and why they are an essential aspect. How constraints are setup is detailed to give a better understand.*

**Introduction**

The simulation of cloth first started in the 1930's with a paper on the geometry of the structure of cloth (Pierce, F.T. 1937). It was not until the 1980's when multiple papers where published on different methods of cloth simulation. It is a highly desired method in computer graphics especially games as it adds a lot of realism to a scene.

For this report, aspects of cloth simulation will be discussed, demonstrated and concluded. Firstly, the method known as Verlet integration which is used for calculating the trajectories of particles within computer graphics and simulations will be researched and discussed. With Verlet integration covered, cloth simulation will be explored, covering the following topics: Initialisation, constraints and anchors. Following the information on cloth simulation, descriptions will be given on the setup of indexes and buffers of the constraints within the code that accompanies this document. To conclude the document a brief analysis will be given on the

**Verlet Integration**

At the centre of every simulation, there is a particle system. Usually, within the implementation of a particle system, each particle has two needed variables: position x and velocity v. The new position x' and velocity v' are computed within the time-stepping loop with the equations below. The most common go-to method within real-time particle systems is the Euler integration method. (Jakobsen. 2003).

where,

X, V - Initial position and velocity respectively,

X', V'- Resulting position and velocity respectively,

∆t - Time step,

a - Acceleration using Newton's law f=ma (where f is the total force acting on the particle)

Although this very basic Euler integration is effective in producing a good cloth simulation, it is not the most effective or accurate. Below is the rule for Verlet integration which stores the current position of x and its previous position x\*. Keeping the time step fixed:

Verlet integration is widely used in the simulation of molecular dynamics. Due to the velocity being implicitly given, it is quite a stable approach and this results in the velocity and position being less likely of coming out of sync. From the rules above it gives an accurate approximation of the current velocity from the distance travelled in the last step. Changing the '2' in the rules can add drag to the simulation, which in turn adds to the realism.

At the end of each step the current position x of the particle gets stored in the x\* variable.

The downside to Verlet integration is that it handles the changing of time steps badly, it requires two steps to get going, which means initial conditions are vital and it is unclear how it deals with the changing of accelerations from the formulation. (Dummer. 2004).

**Cloth Simulation**

A set of particles are used to form the model of a cloth object. Each particle subject to forces (gravity, wind and drag) and various constraints (to maintain the shape of the object and to prevent interpenetration with other objects. The particle's equation of motion from the application of forces comes from Verlet integration explained in the previous section.

Constraints are an essential component for cloth simulation. These constraints create a system of equations where the position of particles are linked together. At each simulation time step the system is solved by relaxation (by enforcing the constraints one by one by 'x' amount of iterations.

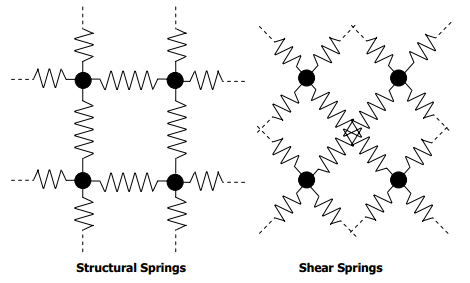
There are two types of constraints; horizontally and vertically connecting constraints are structural constraints and diagonally connecting constraints are shear constraints, as illustrated in figure 1

Figure - Structural and Shear Constraints (nVidia 2007)

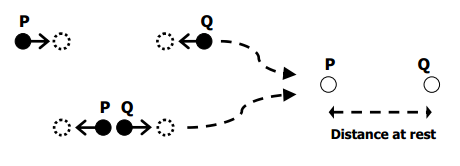
To start a constraint the distance between two particles needs to be calculated. The distance at rest is taken. The constraint keeps the particles within a set distance of the distance at rest. See figure 2 for a representation of two particles P and Q for distances and what the constraint will do if the particles are not within the distance at rest (Mosegaard. 2009).

Figure - Distance between P and Q with constraint effects. (nVidia 2007)

Between each particle collision, constraints and objects are used to check whether a particle is within the object or not. If the particle is within the object, it is moved to a position at the surface of the object usually to the closest point of the particles current position. The following three steps are followed for the implementation of a complete algorithm for every simulation step:

* **Step 1:** For every particle that is not an anchor point:  
  *Apply force through equation of motion*
* **Step 2:** For every particle that is an anchor point:  
  *Update position*
* **Step3: For every relaxation step:**
  + **Step 3a**: For every spring constraint:  
    *Enforce spring constraint*
  + **Step 3b**: For every particle:
    - For every collisions object:  
      *Enforce collision constraint*

With the method of setting up the constraints into a single constraint buffer and shader resource view is that due to parrallel program, threads are executed in a random order. This leads to unwanted results which will end with a cloth that does not act like cloth. To solve this batches of constraints are setup. Each of the batches setup constraints that affect as little particles as possible. No batch can share constraints. Figure 3 is a visual representation to get a better understanding:

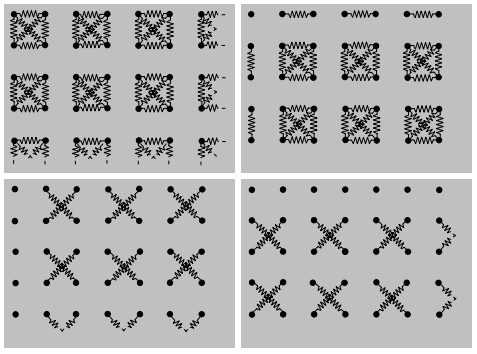


Figure 3- Partitioning of the total set of spring constraints in four sets of independent batches of constraints. (nVidia 2007)

Cloth simulation can be done on the CPU but due to the high multi-core processing power, it is typically executed on the GPU. The particles are stored in a buffer and at each simulation step they are processed through several rendering passes. Forces can be applied to the particles in parallel, this also applies for the enforcement of the collision constraints. This means Step 1 and Step 3b are implemented as one rendering pass through the vertex shader stage. Despite there not being that many anchor points, the position setup of each can also be done in parallel. Step 2 is implemented as one rendering pass through the pixel shader stage (nVidia, 2007).

A more powerful and very useful shader for cloth simulation is making use of compute shaders. They expand Microsoft Direct3D 11 beyond graphics programming. Just like geometry and pixel shaders, they are implemented with HLSL, but technically that is the only similarity. Compute shaders provide high-speed general purpose computing by taking advantage of the GPUs generous amount of parallel processors. The particles can be passed through to the shader using UAV and the shader uses RWStructuredBuffers to access the information (Microsoft, 2014).

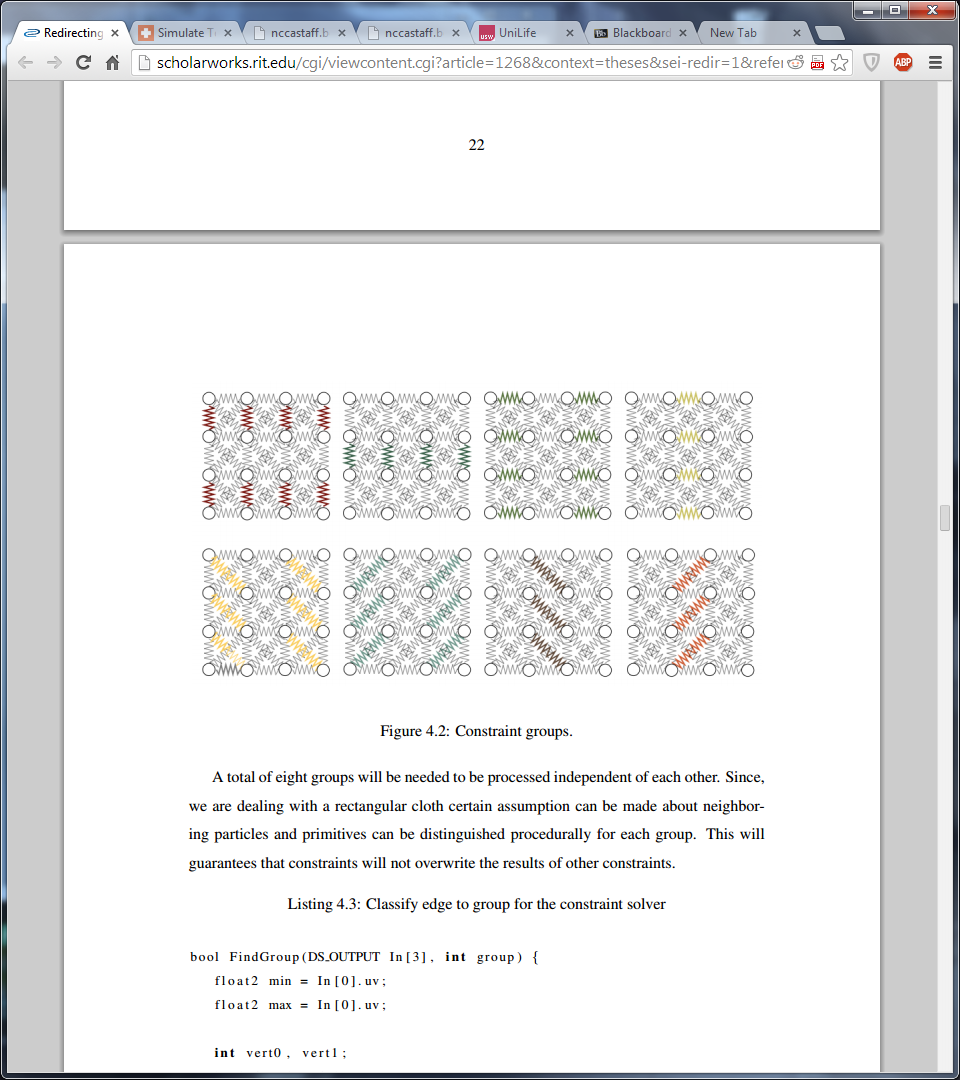
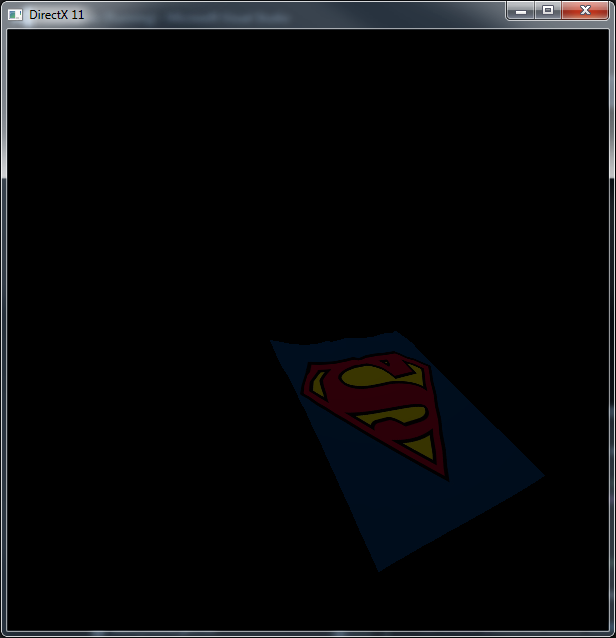
**Index Buffers and Updating Spring Constraints**

Initially the development for the cloth simulation started with a basic implementation of a plane by closely following the vertices and indices setup of the CGBasicTerrain. Certain unneeded information was removed such as the methods for use with heightmaps.

When setting up the constraints they were done within one batch and store in one buffer and as found out through research this came with unwanted visuals for the cloth. The method of setting up four batches was then implemented but this also came with some visual issues, they were better than a single batch but not perfect, due to some constraints jumping around randomly.

To solve this issue a paper that detailed an eight batch method was followed (Huynh, 2011). Each batch deals with individual direction constraints based on an odd and even vertex method. The batches setup are basically the same, the only changes made were to do the shear constraints like the vertical constraints where a row was skipped instead of a column like the horizontal batch. See figure 4.

The information for each batch of constraints were passed through to a compute shader via its own shader resource view. The shader was then updated on each pass with the batches. This produced a cloth with no jumping random jumping particles (Huynh, 2011).



**Conclusion**

In conclusion, cloth simulation is a very visually pleasing method. Seeing it in games adds so much realism to the visuals. Since the first implementation of a cloth simulation hardware has advanced a lot and allowed for GPUs to carry out allow of the processing necessary to produce a realistic simulation. Nowadays nearly all of the execution is conducted by the GPU in a very timely fashion. Compute shaders providing even faster execution than geometry and pixel shaders which then allows the use of cloth simulation in games (real-time) to increase the realistic look of a virtual world. With the ever advancing hardware cloth and other methods can only get better.

**References**

Jakobsen, T. 2013. *Advanced Character Physics*. Gamasutra.com. Accessed: 4/3/2014.

Dummer, J. 2004. *A Simple Time-Corrected Verlet Integration Method.* Accessed: 4/3/2014.

nVidia. 2007. *White Paper, Cloth Simulation*. Accessed: 28/2/2014.

Mosegaard, J. 2009. *Cloth Simulation Coding Tutorial.* 28/2/2014.

Microsoft, 2014. *Compute Shader Overview.* Accessed: 28/2/2014.

Huynh, D. 2011. *Cloth Simulation using Hardware Tessellation*. Accessed 2/3/2014.

Pereira, L. 2010. *C++ Cloth Simulation*. Accessed 2/3/2014.