**Simulating Cloth Using Direct X 11**

**Real-Time Computer Graphics**

**Jak Boulton**

**10026037**

|  |
| --- |
| *Abstract*  *This report discusses an overview of how simulation of cloth physics can be implemented using the massive parallel processing power of the graphics processing unit (GPU) normally used to draw high polygon count models and high resolution textures within videogames. The techniques involved are explored with a focus on the compute shader and the benefits it brings to simulating cloth in real-time. Finally, the report is concluded with a look over the compute shader and other options for cloth simulation.* |

**1. Introduction**

Cloth simulation has been a topic of interest since the emergence of the original works of Terzopoulos among other authors (Huynh, 2011) (Terzopoulos, 1987). Since then many techniques and methods have been developed and evolved to simulate both realistic and real-time approximations of cloth physics. The general focus has however been on realistic simulation rather than real-time approximations. Due to the recent surge of yearly improving graphics hardware; cloth simulation and computation has become both more and more complex and easier to render in real-time. The use of graphics hardware and the newest versions of graphics APIs (such as DirectX 11) have allowed cloth simulation to run in real-time with great performance and relative physical accuracy.

This report provides an overview of cloth simulation and the techniques involved in order to compute cloth simulation on the GPU at interactive frame rates. The report begins with a look at the most common method of computing the fluid-like movement of the cloths physics; *Verlet Integration* before exploring an overview of the basic method for cloth simulation. Creation of the index buffers required to compute the cloth constraints and how they are computed per-frame on the GPU is explored. The report is then concluded by summarising the reasons for using the compute shader method rather than other options explained throughout.

**2. Verlet Integration**

At the core of every simulation is a particle system that manages how each of the particles reacts based on the system setup and force application. The method most often used (in real-time systems) is the *Euler integration* method (Jakobsen, 2003). An approximation of the next position for any given particle is based on these rules:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |
|  |  |

Where:

|  |  |
| --- | --- |
|  | The initial and next particle positions; |
|  | The initial and next particle velocities; |
|  | The time step between simulation steps; |
|  | The particle acceleration calculated using Newtons law. |
|  |  |

This is a very simple form of Euler Integration (Howell, 2012). The main problem with using this equation (or group of rules) to define the movement for each of the particles within cloth simulation is that fabrics do not react to forces in this way in the real world and so the simulation would appear too rigid and not believable. Another particle management system more suited to cloth simulation is known as *Verlet Integration* (Counts, 2012)(Dummer, 2005)*.* This differs from Euler Integration by representing particles without the use of a velocity. Instead of storing both the current position and velocity for each of the particles, the previous position is stored. Once the current and previous positions are stored, a fixed time step is used with each integration step (Fiedler, 2006).

This is calculated as below:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |
|  |  |

Where:

|  |  |
| --- | --- |
|  | The previous particle position. |

As seen in equation (2), there is no velocity taken into account within the calculation of the next position. The calculated position is known as *Inertia* and is calculated using the previous and next particle positions. The acceleration or in the equation denotes the forces actively being applied to the particle, this can be used for forces such as gravity and/or wind for example. The outcome when using Verlet integration over a more basic model like Euler are that simulations such as cloth appear to hold their kinetic energy from previous simulation steps causing for a more fluid or fabric behaviour (BlueThen, 2011).

**3. Cloth Simulation**

Simulating materials in real-time systems such as video games has been the goal for many algorithm authors for many years. When treated as a lattice of particles, cloth simulation becomes a problem that can be solved through the means of data parallelism (Zeller, 2007). Modern GPUs often have over one-thousand cores allowing for a large number of threads to be running at once (Nvidia, 2012). Taking these two points into account, cloth simulation is an ideal candidate for GPU computation.

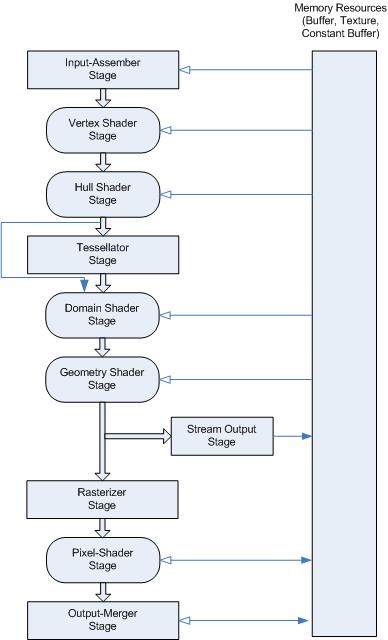
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

***Figure 1:*** *Particles in a regular order*

The algorithm for modelling real-time cloth is as follows:

|  |
| --- |
| * **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*   * **Step 3**: For every relaxation step:   + **Step 3a**: For every spring constraint:   *Enforce spring constraint*   * + **Step 3b**: For every particle:   *For every collision object*  *Enforce collision constraint.* |

As mentioned above the cloth object is treated as a group of particles that together model the cloths surface, as shown in figure 1. Every particle is subject to a group of forces; external forces such as gravity, wind and drag and various constraints; shape constraints which define the shape of the cloth and collision constraints which ensure that the cloth does not clip into other objects within the scene (Choi, 2005). There is also a great deal of research and implementation into methods that treat the cloth as a collision object against itself (Bargmann, 2003). Anchor constraints are also applied to specific particles to hold the entire cloth in space. These constraints, the forces and the other structural/shear constraints are the required group of elements allowing for a realistic simulation of cloth.



***Figure 2:*** *The Graphics Pipeline for DirectX 11.*

Cloth simulation is often implemented on the GPU through a stage of the graphics pipeline known as the geometry shader (Zeller, 2007). This is due to the ability to pass adjacency information for surrounding particles. The main issue with this however is that the this requires a group of rendering passes using the stream output stage forcing duplication of buffers since you cannot set a buffer for both read and write capabilities. Direct X 11 offers many other options for implementing parallel algorithms on the GPU using pipeline stages. As an example, the tesselator and hull shader could be used since they offer adjacency information that is much less restricted than previous versions of Direct X. Data duplication would still be required due to the inability to set a shader as both read and write. The other option would be to use the compute shader capabilities with Direct X 11.

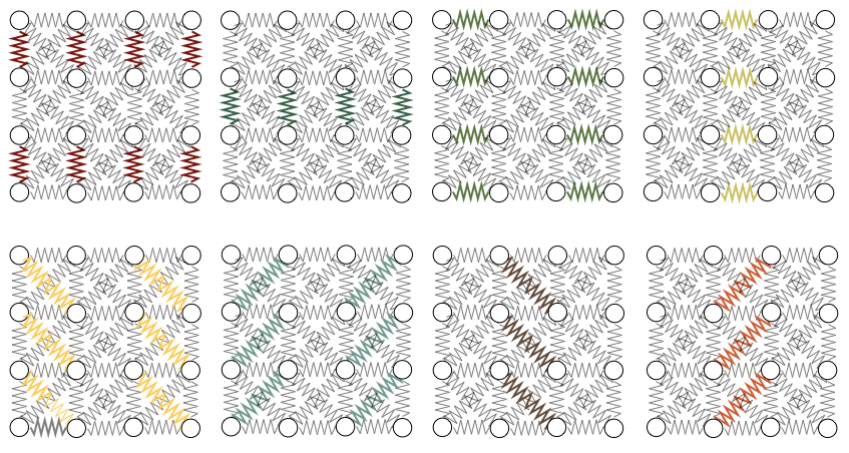
The compute shader is a programmable shader stage that offers programming of the GPU for more general uses on top of the intended graphics potential. The benefits of using compute shaders over other previously mentioned options are that there is no need to force data into a format which the graphics pipeline ‘understands’. There is no need to duplicate buffers in order to alter vertex positions since there are buffer types such as *RWStructuredBuffer*s and *RWByteAddressBuffers* that as the naming prefix suggests, offer both read and write access (Microsoft, 2014).

**4. Index Buffers and Updating Constraints**

Cloth simulation requires the use of spring constraints that dictate the rest distance between particles and their neighbours. There are two main types of spring constraint in cloth simulation; structural and shear. Structural constraints are vertical and horizontal and make up the basic structure of the cloths surface and shear constraints are diagonally crossed over each quad of particles to ensure the cloth keeps to the shape originally defined.

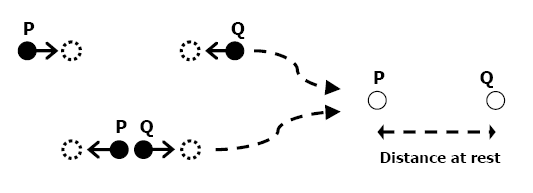
|  |  |
| --- | --- |
|  |  |
| ***Figure 3:*** *Structural constraint layout* | ***Figure 4:*** *Shear constraint layout* |

These constraints work by checking the distance between the two points involved before moving them closer together or further apart dependant on the particle positions at a given simulation step. The problem with simply setting up a single buffer and shader resource view for the constraints is that in parallel programming, threads are executed in a non-deterministic order leading to unpredictable results. To solve this, batches of constraints are setup based on independence of each constraint. What is meant by this is that each constraint operates on two particles, during the processing of each batch of constraints only one constraint must process each particle. The result is that the particle buffer must be organised into eight batches containing independent constraint sets. The sets are visually represented in figure 5 (Huynh, 2011).



***Figure 5:*** *Constraint batches*

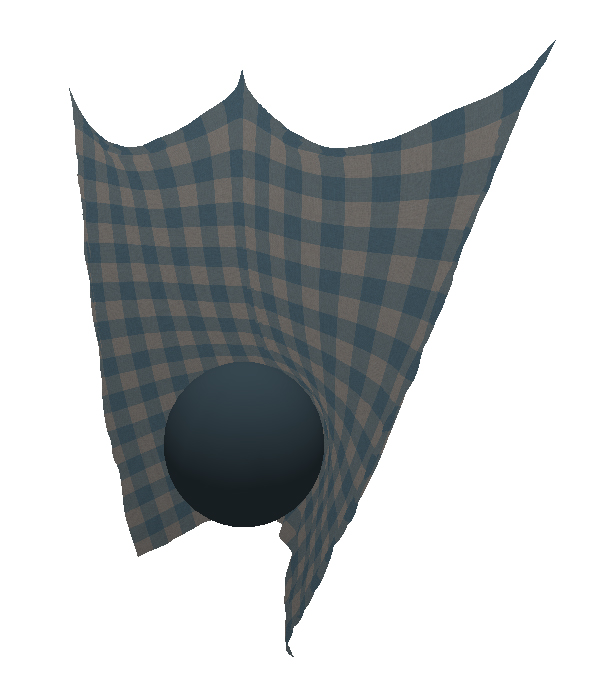
Updating the constraints is computed in the compute shader (as are the forces, collisions and anchors). The first step is to compute the vector pointing from the first particle to the second; this is found by subtracting the second particles position from the first. The length of this vector is compared to the rest distance defined in the constraint before a scale factor is computed and the particles are moved together or apart based on the distance. Figure 6 shows a visual representation of how the constraints are managed.



***Figure 6:*** *Distance constraint between particles P and Q.*

**5. Conclusions**

Verlet integration provides the particles with a model of motion in simulating cloth. It allows for fluid-like movement that suits how cloth moves when reacting to forces such as wind and gravity. Structural constraints, shear constraints and collision constraints ensure the surface of the cloth does not deform in ways that are not expected of cloth material. Modern graphics APIs and hardware offer many ways that developers are able to interact with the GPU to harness the parallel processing power, however a smaller subset of options are available to simulate cloth. Overall, the compute shader offers the best interaction while not enforcing limits or bottlenecks on setup or application performance.



***Figure 7:*** *Cloth.*

# 6. Bibliography

Bargmann, R., 2003. *Real-Time Cloth Simulation,* Zurich: Swiss Federal Institute of Technology.

BlueThen, 2011. *Updated Curtain and How to Improve your Verlet Cloth.* [Online]   
Available at: http://bluethen.com/wordpress/index.php/processing-app/updated-cloth-simulator-and-how-to-improve-your-verlet-cloth-simulator/  
[Accessed 5 March 2014].

Choi, K.-J., 2005. *Stable but Responsive Cloth,* Seoul: ACM SIGGRAPH.

Counts, J., 2012. *Simulate Tearable Cloth and Ragdolls With Simple Verlet Integration,* s.l.: tutsplus.com.

Dummer, J., 2005. *A Simple Time-Correted Verlet Integration Method.* [Online]   
Available at: http://lonesock.net/article/verlet.html  
[Accessed 5 March 2014].

Fiedler, G., 2006. *Fix Your Timestep!.* [Online]   
Available at: http://gafferongames.com/game-physics/fix-your-timestep/  
[Accessed 5 March 2014].

Howell, K., 2012. Euler’s Numerical Method. In: *Applied Differential Equations.* Alabama: Hayden-McNeil Publishing, pp. 191 - 210.

Huynh, D., 2011. *Cloth Simulation using Hardware Tessellation,* New York: B. Thomas Golisano College of Computing and Information Sciences.

Jakobsen, T., 2003. *Advanced Character Physics.* s.l., Gamasutra.

Microsoft, 2014. *Compute Shader Overview.* [Online]   
Available at: http://msdn.microsoft.com/en-us/library/windows/desktop/ff476331(v=vs.85).aspx  
[Accessed 5 March 2014].

Nvidia, 2012. *GeForce GTX 670 Hardware Specifications.* [Online]   
Available at: http://www.geforce.co.uk/hardware/desktop-gpus/geforce-gtx-670/specifications  
[Accessed 5 March 2014].

Terzopoulos, D., 1987. Elastically Deformable Models. *Computer Graphics,* 21(4), pp. 205 - 214 .

Zeller, C., 2007. *Cloth Simulation,* s.l.: Nvidia.