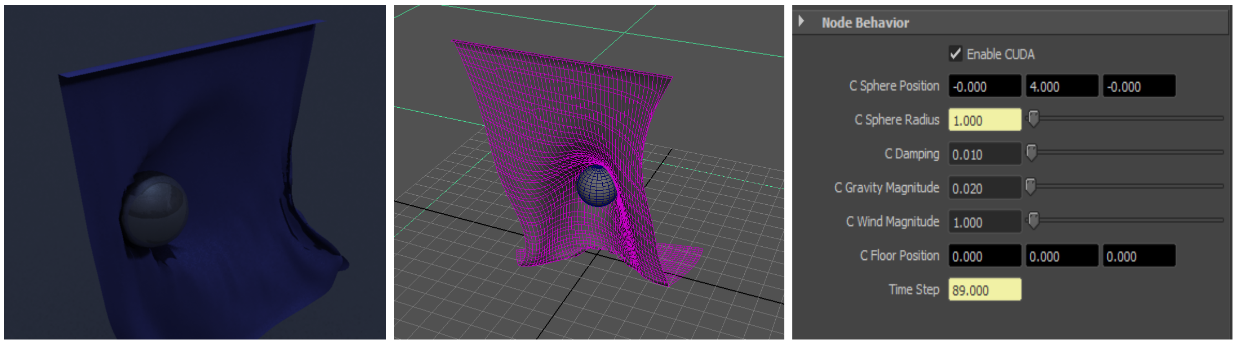
# TNCG13 SFX - Tricks of the Trade 2013

Sathish Kottravel (satko730)

# Cloth Deformer  Plugin Using Hybrid GPU-CPU Parallelization



Node Behavior

[✓ Enable CUDA]

| | | | |
|---|---|---|---|
| C Sphere Position | -0.000 | 4.000 | -0.000 |
| C Sphere Radius | 1.000 | | |
| C Damping | 0.010 | | |
| C Gravity Magnitude | 0.020 | | |
| C Wind Magnitude | 1.000 | | |
| C Floor Position | 0.000 | 0.000 | 0.000 |
| Time Step | 89.000 | | |

Figure 1: ClothDeformer a) Snapshot from final rendering b) Scene setup c) Deformer attributes

## Introduction

In this project a basic Cloth Deformer plugin is developed using C++ Maya API. This deformer invokes solver that uses Verlet-Integration method to solve mass-spring system. Mass-spring system is constructed by the deformer for a given polygonal mesh (cloth). Verlet-Integration is simple and easy approach to solve any mass-spring system.

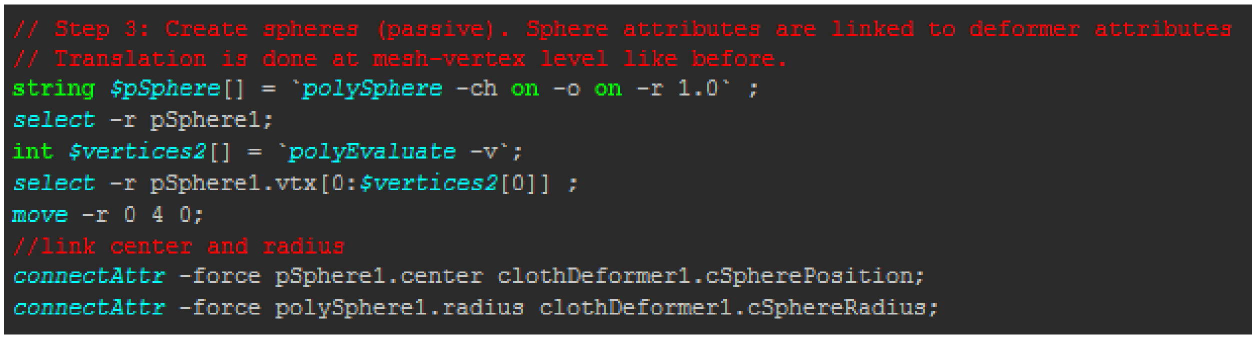
The solver exploits parallelism from both GPU and CPU. Force equation is solved in GPU using CUDA. The structural constraints are solved in CPU and uses OpenMP to invoke parallelism.

The Cloth Deformer is applied on a polygonal plane which is treated as active object that represents the cloth. The solver in the deformer requires two passive objects (sphere and plane). Two types of forces can act on cloth, gravity and wind. The magnitude of the gravity and wind can be controlled from the attributes.

## Plugin

This plugin has been developed using Visual Studio. But instead of using Visual Studio wizard provided with Maya API, special CMake scripts are written for this project. These CMake scripts can search for Maya libraries, include directories and generates visual studio solution. Similarly CUDA CMake scripts are used to located Cuda API.

Cloth Deformer plugin consists of several attributes as shown in Figure1c. Most of the attributes in cloth deformer node can be connected to attributes of other nodes in the scene. In Figure 2, MEL snippet has been shown that connects attributes of polygonal sphere (center and radius) to the cloth deformer attributes. Similarly floor, wind, gravity and other attributes can be connected from other Maya nodes.



```
// Step 3: Create spheres (passive). Sphere attributes are linked to deformer attributes
// Translation is done at mesh-vertex level like before.
string $pSphere[] = `polySphere -ch on -o on -r 1.0` ;
select -r pSphere1;
int $vertices2[] = `polyEvaluate -v`;
select -r pSphere1.vtx[0:$vertices2[0]] ;
move -r 0 4 0;
//link center and radius
connectAttr -force pSphere1.center clothDeformer1.cSpherePosition;
connectAttr -force polySphere1.radius clothDeformer1.cSphereRadius;
```

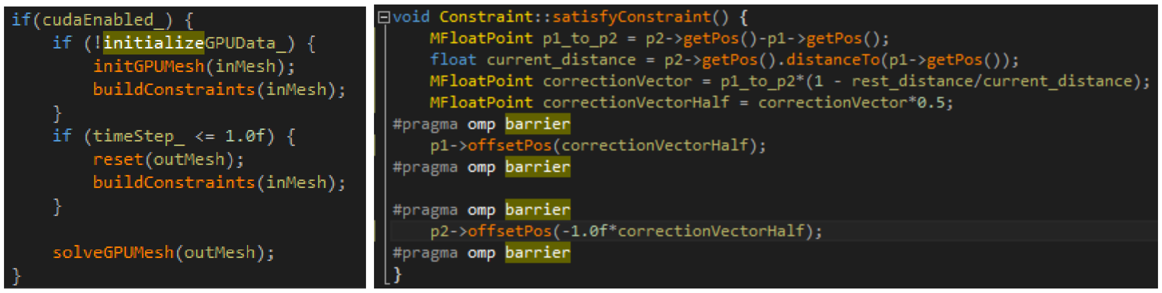
Figure 2 Sample script that used to setting up initial scene. Connecting sphere radius and center to Cloth Deformer attribute.

Cloth Deformer plugin has *MPxDeformerNode* as parent node. By default parent node has input and output geometry. During *compute()* function of the plugin, GPU data structure and other helper structures are initialized. This initialization is done when *compute()* function is executed for the first time. This whole setup is reinitialized again when playback time is set to frame1. This is shown in Figure 3a, C++ snippet.

During the solver phase, 2 types of displacement information are required for each vertices. They are displacement from previous time frame and displacement from current time frame. This is used in the force-equation solver to predict the next displacement. At this phase each node should have been associated with the resultant forces acting upon them.

For simplicity, uniform mass is assigned to each vertex. But the underlying data structure already supports non-uniform mass. After solving the force and predicting the displacement, it is necessary to perform collision detection. In this project, sphere and plane are two passive objects used for collision and interaction.

After this step, constraints between each displaced vertices are restored. Solving constraints is very essential step and it is also computationally intensive. Constraints are not GPU friendly. In other words, solving constraints on GPU would lead to very high thread divergence.With special preparation it is possible to solve constraints on GPU, which is beyond the time scope of this project. But as an alternative option, constraints can be solved using CPU multi threading. OpenMP is used in this project. OpenMP has been preferred for two reasons, one its less intrusive, two resolving racing condition is much easier (Figure 3b).



```
if(cudaEnabled_) {
    if (!initializeGPUData_) {
        initGPUMesh(inMesh);
        buildConstraints(inMesh);
    }
    if (timeStep_ <= 1.0f) {
        reset(outMesh);
        buildConstraints(inMesh);
    }

    solveGPUMesh(outMesh);
}
```

```
void Constraint::satisfyConstraint() {
    MFloatPoint p1_to_p2 = p2->getPos()-p1->getPos();
    float current_distance = p2->getPos().distanceTo(p1->getPos());
    MFloatPoint correctionVector = p1_to_p2*(1 - rest_distance/current_distance);
    MFloatPoint correctionVectorHalf = correctionVector*0.5;
#pragma omp barrier
    p1->offsetPos(correctionVectorHalf);
#pragma omp barrier

#pragma omp barrier
    p2->offsetPos(-1.0f*correctionVectorHalf);
#pragma omp barrier
}
```

Figure 3 a) Initialization and resetting the plugin. b) Resolving racing conditions in OpenMp

## Discussion and Future Work

Basic ClotherDeformer implementation is done and it shows reasonable stability. Hybrid CPU-GPU parallelization has been implemented. Force equation solver is GPU accelerated. Constraint solver is CPU accelerated. Performance of constraint solved can be improved either by porting it to GPU or by using better data structure that can support optimized acceleration on CPU. The type of constraint that has been used in this project is Structural constraints. It preserves structure of the cloth. Another type of constraint that can be improve cloth like behaviour is Bend Constraints. Bend constraints remove any edgey structures in the cloth during simulation. But this will increase computational load of constraint solver.

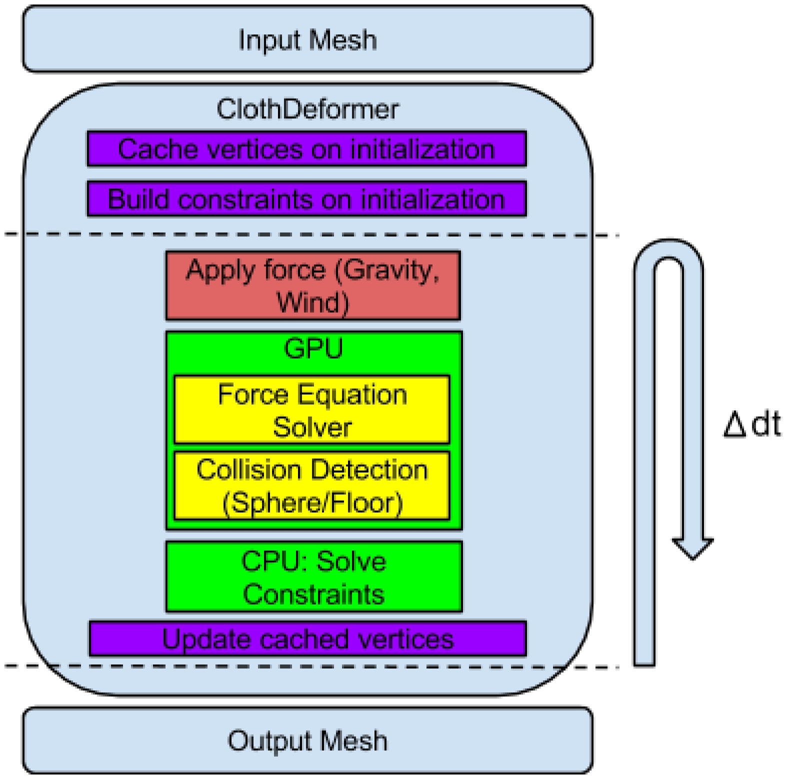

Input Mesh
ClothDeformer
Cache vertices on initialization
Build constraints on initialization
Apply force (Gravity, Wind)
GPU
Force Equation Solver
Collision Detection (Sphere/Floor)
CPU: Solve Constraints
Update cached vertices
Δdt
Output Mesh

Figure 4 ClothDeformer overview. dt is time step. At every time step displacement is calculated.

## Result

For demo scene with minimal animation has been setup. Final rendering of the scene is done using Mental Ray with indirect lighting option as Image Based Lighting. IBL uses HDR panorama image to emit light (provided in the course webpage). ( Link )

## References

[1] Computer Graphics Lab, http://cg.alexandra.dk/
[2] CUDA Programming Guide 3.0
[3] CUDA by Example.
[4] Blob Physics By Mick West, "Inner Product" - Game Developer Magazine
[5] Maya API Reference