# CS 207 Final Project

### Prompt 3 - Tetrahedral Mesh

Yung-jen Cheng, Jeffrey Shen

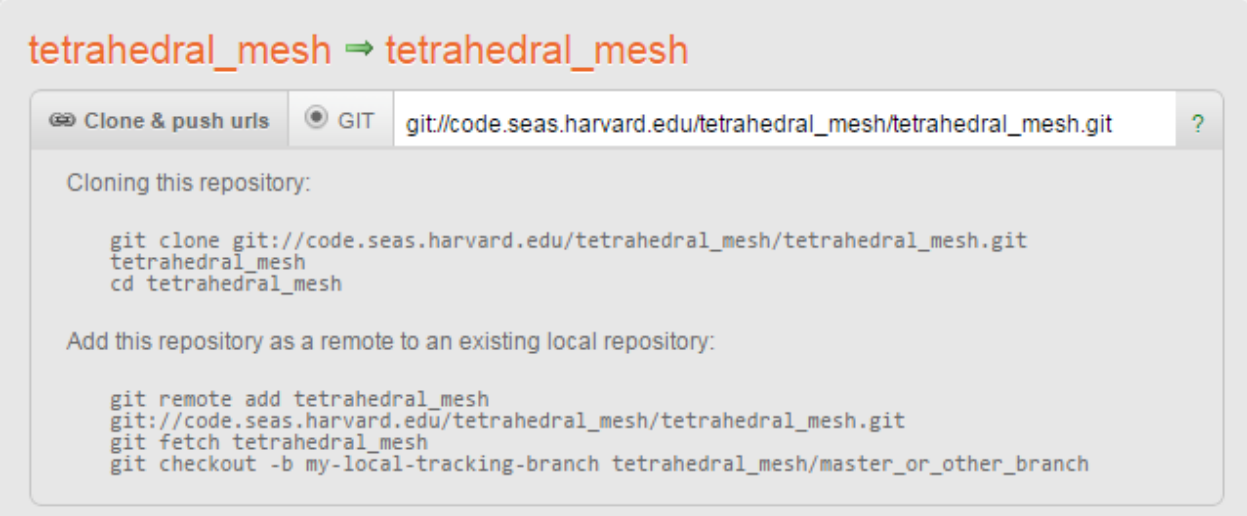
## Motivation:

In previous work, we have already had experience in using 2D meshes to simulate mass-spring model and shallow water simulations (triangular mesh). These 2D meshes model objects that consists of surfaces with internal pressure well.

**Tetrahedral mesh are more natural models for more solid objects.** We have provided a tetrahedral mesh object that can used to simulate bouncing balls and 3D fluid flow.

## Getting the Code:

The code can be found on
https://code.seas.harvard.edu/tetrahedral_mesh/tetrahedral_mesh/trees/master




**Steps to clone the repository**

## Using the Code:

The tetrahedral mesh class has many analogous functions to the triangular mesh.

### Files to include: (be sure you have these files in the right places)

```
#include "Graph.hpp"
#include "CS207/Util.hpp"
#include "CS207/Point.hpp"
```

### Declaration of a Mesh:

The mesh is templated on 3 parameters. This is similar to the triangular mesh.

```
template <typename N, typename E, typename T>
class Mesh { … };
```

Example:

```
typedef Mesh<NodeData, EdgeData, TetrahedralData> MeshType;
MeshType mesh;
```

In the above declaration, NodeData, EdgeData, and TetrahedralData are user-defined structures that can be passed into mesh to store values associated with Nodes, Edges, and Tetrahedrons.

To access the templated structure in a node:
Example:

```
MeshType::Node n = mesh.node(0);
NodeData ndata = n.value();
```

To access the templated structure in an edge:
Example:

```
MeshType::EdgeIterator ei = mesh.edge_begin();
EdgeData eData = (*ei).value();
```

To access the templated structure in a tetrahedron:

```
MeshType::Tetrahedral tet = mesh.tetrahedral(0);
TetrahedralData tetData = tet.value();
```

### Adding Nodes and Tetrahedrons to the Mesh:

Adding nodes and tetrahedral are primarily dealt with `mesh.add_node()` and `mesh.add_tetrahedral()`.

```
Node add_node(const Point& p, const node_value_type& node_value)
Tetrahedral add_tetrahedral(const Node& n0, const Node& n1, const
Node& n2, const Node& n3, const tet_value_type& tet_value =
tet_value_type())
```

### Tetrahedron Specific Functions

Volume: returns the volume of the tetrahedral. This function is useful when one needs the tetrahedron's volume to interact with forces. Lastly, negative volume signifies that the tetrahedral has inverted in the process.

```
double volume()
```

Example:
```
MeshType::Tetrahedral tet = mesh.tetrahedral(0);
double tetVolume = tet.volume();
```

Surface: returns true if the tetrahedron is not enclosed by four tetrahedrons and has at least one surface exposed. Otherwise, false. This function is useful when one needs to know if a tetrahedron is on the surface of a 3D object or enclosed inside a 3D object.

```
bool isSurface()
```

Example:
```
MeshType::Tetrahedral tet = mesh.tetrahedral(0);
bool isTetOnSurface = tet.isSurface();
```

### Traversing through Nodes, Edges, and Tetrahedrons

Traversal through each of the objects above can be achieved by using each class' iterator: NodeIterator, EdgeIterator, and TetrahedralIterator.

NodeIterator:
```
node_iterator node_begin()
node_iterator node_end()
```
Example: `mesh.node_begin(); mesh.node_end();`

EdgeIterator:
```
edge_iterator edge_begin()
edge_iterator edge_end()
```
Example: `mesh.edge_begin(); mesh.edge_end();`

TetrahedralIterator:
```
tet_iterator tetrahedral_begin()
tet_iterator tetrahedral_end()
```
Example: `mesh.tetrahedral_begin(), mesh.tetrahedral_end()`

IncidentIterator: travesing all the Edges incident to this node.
```
incident_iterator edge_begin()
incident_iterator edge_end()
```
Example:
```
MeshType::Node n = mesh.node(0);
n.edge_begin();
n.edge_end();
```

### Obtaining all adjacent Tetrahedrons of Nodes, Edges, and Tetrahedrons

In order to calculate forces or manipulate tetrahedrons' values based on adjacent tetrahedrons, we have provided the following functionalities.

To obtain all of the tetrahedrons that a node is a part of, we can use:
```
vector<Tetrahedral> nodeAdjTetrahedral()
```

Example:

```
MeshType::Node n = mesh.node(0);
vector<MeshType::Tetrahedral> nAdjTets = n.nodeAdjTetrahedral();
```

To obtain all of the tetrahedrons that an edge is a part of, we can use:

```
vector<Tetrahedral> edgeAdjTetrahedral()
```

Example:

```
MeshType::EdgeIterator ei = mesh.edge_begin();
vector<MeshType::Tetrahedral> eAdjTets = (*ei).edgeAdjTetrahedral();
```

To obtain all of the tetrahedrons that a tetrahedron is adjacent to, we can use:

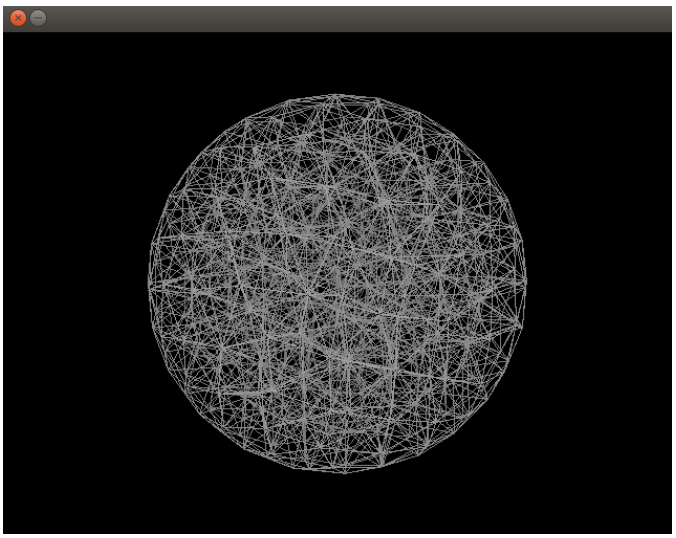
```
vector<Tetrahedral> tetAdjTetrahedral()
```

Example:

```
MeshType::Tetrahedral tet = mesh.tetrahedral(0);
vector<MeshType::Tetrahedral> tetAdjTets = tet.tetAdjTetrahedral();
```

## Example:

The tetrahedral mesh can be ran on any file with a pair of .nodes and .tets extensions. In the data\ folder, we have included sphere87, sphere676, and sphere2696 .nodes and .tets files.



**Static Picture of sphere2696 comprised of 2,696 tetrahedrons**

By combining forces such as the dashpot (3D equivalent of mass-spring and damping forces) and volume penalty (stops the tetrahedral from inverting) forces, we can simulate a bouncing ball with `test_tet_mesh.cpp`.

$$\mathbf{f}_{i,\text{dashpot}} = \sum_{j \in A_i} -\left( K_{ij}\left(|\mathbf{x}_i - \mathbf{x}_j| - L_{ij}\right) + C\frac{(\mathbf{v}_i - \mathbf{v}_j)\cdot(\mathbf{x}_i - \mathbf{x}_j)}{|\mathbf{x}_i - \mathbf{x}_j|}\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|}$$

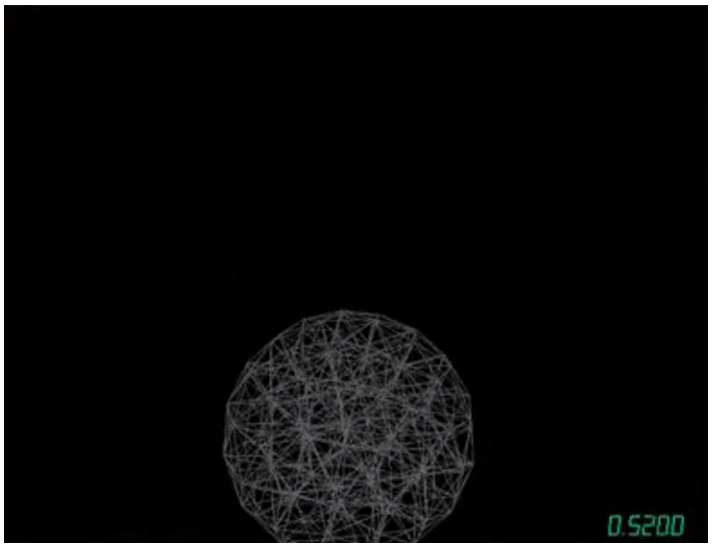
$$\mathbf{f}_{i,\text{volume}} = \sum_{k \in \mathbb{T}(i)} -K(V_k - V_k^0)\frac{\mathbf{x}_i - \mathbf{p}_k}{|\mathbf{x}_i - \mathbf{p}_k|}$$

where $K$ is a constant, $\mathbb{T}(i)$ is the set of tetrahedra adjacent to node $i$, $V_k$ and $V_k^0$ are the current and original volume of tetrahedron $k$, and

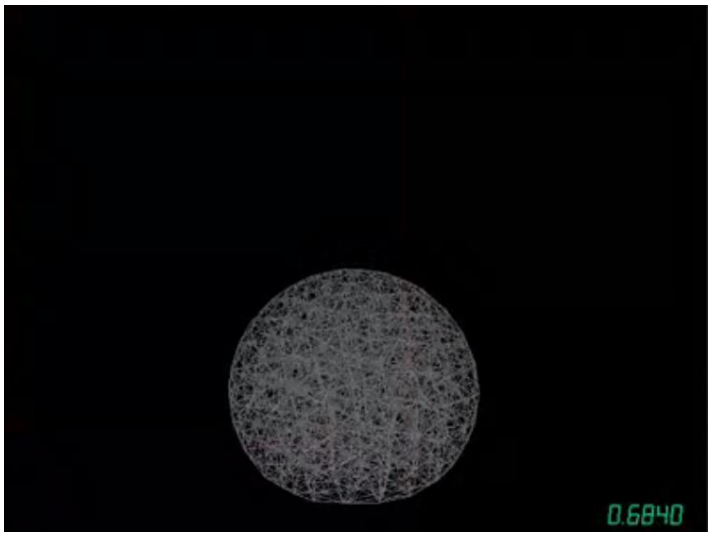
$$\mathbf{p}_k = \frac{m_0\mathbf{x}_0 + m_1\mathbf{x}_1 + m_2\mathbf{x}_2 + m_3\mathbf{x}_3}{m_1 + m_2 + m_3 + m_4}$$

The two forces are modeled as functors in test_tet_mesh.cpp as `DashpotForce` and `VolumePenaltyForce`. These are used in the same way as we have used past forces for mass spring.

## Simulation Videos:




http://youtu.be/rQhoYIBjHYY




http://youtu.be/RXdeS9w_ToQ

## Possible Extensions:

The tetrahedral mesh can be used to model other solid objects and the objects' interactions. In addition, the tetrahedral mesh can be used to simulate 3D fluid dynamics. Integrations with the collision module and other force modules would yield interesting results.