

# Distributed Adaptive Meshes for Simulation and Animation

by Matt Galloway (Pembroke College)

## Technical Abstract

Triangulations, or meshes, are widely used by computers for simulation and animation purposes. They are a discrete representation of a physical, real world object which the computer can use to perform simulation and animation of real situations. A mesh comprises vertices and elements where a vertex is a point in space and an element is the connection of two or more such vertices to form discrete shapes. These can be any shape desired, but usually triangular elements are used, meaning triangles in 2-dimensions and tetrahedrons in 3-dimensions. Such a mesh can be structured or unstructured, where in a structured mesh the elements are of uniform geometry, whereas in an unstructured mesh the elements can be of any desired geometry. Unstructured meshes are the usual choice for simulation and animation because arbitrarily shaped objects can then be modelled.

Many uses have been found for simulation and animation using meshes including, but not limited to, Finite Element Analysis (FEA) which is a technique where the mesh has some form of physical equations applied to the vertices and elements, leading to the computation of the vertex displacements for given boundary conditions (e.g. forces, moments, displacements, constraints). This allows a designer to simulate the response of an object using a computer as many times as necessary, without physically manufacturing then testing the object. In the case of a complex object such as a car, this will clearly save much time and money.

During simulation and animation it is often required that the mesh is adapted at run time. This can mean many things, including refining the mesh at some location to increase the element density and hence the accuracy of results, or fracturing the mesh to change the topology of the object. This could be used to model how an object breaks during a fracture process, for instance if a designer wants to understand how their object will crack at failure.

Another feature of simulation and animation which is currently being researched is the concept of running the software over more than one computation unit, thus in a parallel, distributed manner. This is useful when the mesh becomes very large because spreading the computation can speed up the overall simulation procedure as each computation unit (process) can act upon different elements at the same time. Spreading the computation introduces problems where an element on one process needs information about an element on a different process. Keeping the inter-process communication to a minimum is the key to an efficient algorithm.

In simulation and animation software there are usually two distinct types of code

which can be described as layers. These are the geometry layer and the physics layer. The physics layer is responsible for the physical laws determining the forces inside elements and displacements of vertices, and also for determining when and where the mesh should be adapted. The geometry layer is responsible for maintaining the connectivity of the mesh during adaptive processes.

This project aims to develop the software which performs the geometry layer functions with the ability to store triangular & tetrahedral meshes and to fracture such meshes. The library of code created should be able to be used by a programmer of a simulation to add their physics code on top and not have to know how the geometry layer code maintains mesh connectivity. The library should contain structures to store the various mesh features such as vertices & elements and also contain algorithms to perform fracture of selected elemental interfaces & allow computation in a parallel manner. The project makes use of the C++ programming language and also extensions to the language including the Standard Template Library (STL) and Boost. To perform the parallel functions, the Message Passing Interface (MPI) is used.

Data structures and algorithms were designed and implemented to perform the full geometry layer functions for fracture simulation, including parallelisation. The software is packaged into a library called Distributed AdaPtive Triangulation Algorithms (DAPTA). In addition to developing the library, test programs were written to show how DAPTA can be used, including a simple FEA simulation. For this, a simple physics model was implemented including a method for determining where fracture should occur, where the results show that mesh connectivity is maintained as required. These test programs make use of the OpenGL framework for displaying graphics. The parallel routines were also tested by distributing a mesh over six processes, initiating fracture on some elements and then checking if all processes obtain the same mesh structure after swapping relevant information. It was found that the algorithms worked as expected.

At the end of the project, a library was produced which can represent triangular meshes and maintain connectivity during fracture, but it can easily be extended to contain other element types (e.g. rectangular) and other adaptive routines (e.g. mesh refinement). Documentation was written along with test programs to explain how the library can be used. The project aims were therefore met.

Along with meeting the project aims, developing the various data structures and algorithms has provided experience in using the C++ programming language for writing scientific applications, along with experience in effective use of the STL and other libraries.