Parallelization of local and global mesh operations using Scotty3D

URL

The website for our final project will be hosted via GitHub. The link is https://alan7996.github.io/15418-project/.

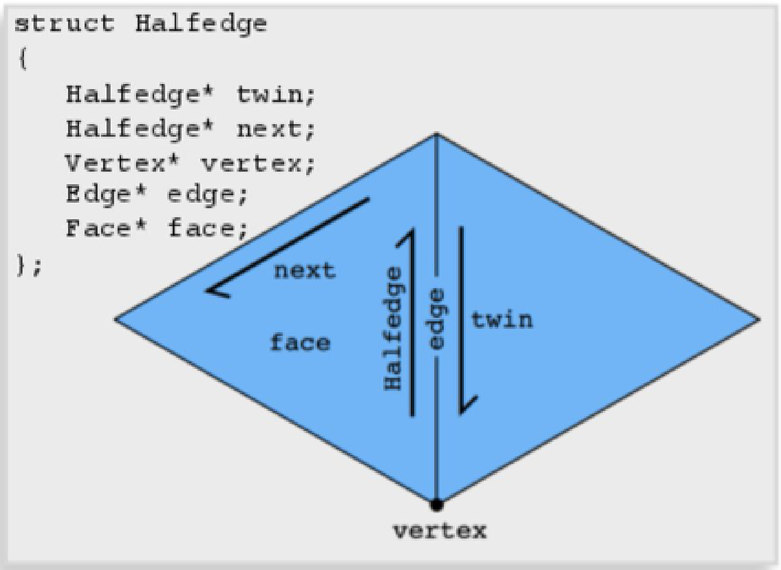
Summary

We are going to implement various parallelizations of local and global mesh operations provided by Scotty3D, a custom 3D graphics software package used in CMU's computer graphics course. We will experiment with ISPC / OpenMP / OpenMPI to analyze the performance differences that varying parameter setups and implementation strategies may introduce.

Background

Manipulating meshes is integral to any modern 3D graphics and modelling software. The range of supported mesh manipulating operations varies greatly from implementation to implementation, often depending on the canonical mesh representation used by the software. Scotty3D is both a modelling and rendering package capable of rasterization, raytracing, and particle simulation with linear blend skinning. The modelling aspect of this software was built with underlying design philosophy respecting the halfedge mesh data structure. Naturally, our goal for this project is to design, implement, and analyze various parallelization schemes for mesh operations on halfedge meshes. We define local operations to be any mesh-altering operation that is applied to a select subset of faces, edges, vertices, or halfedges. Similarly, we define global operations to be any mesh-altering operation that gets applied to every element of faces, edges, vertices, or halfedges.

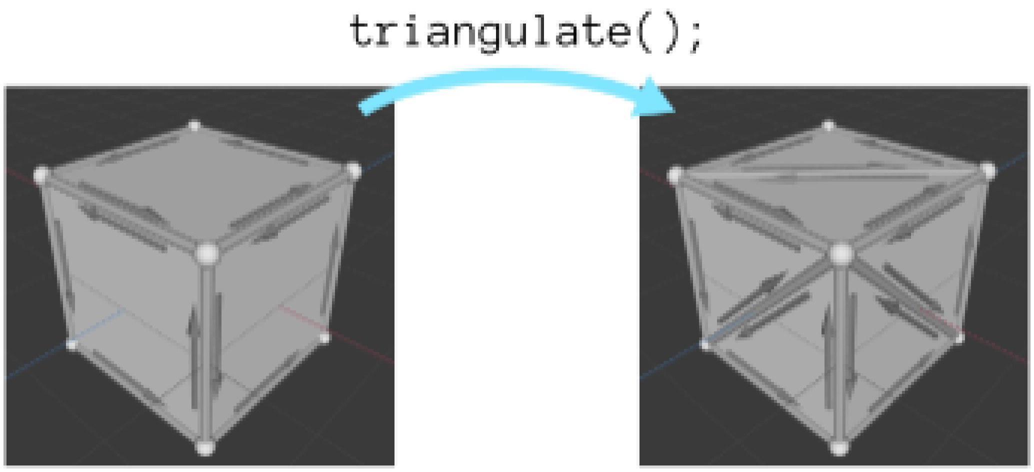
An halfedge mesh structure is a local connectivity description which allows for fast local topology changes and has clear storage locations for data associated with vertices, edges, faces, and face-corners. The fundamental building block of halfedge mesh structure is an halfedge. Let us define an edge as a segment connecting two endpoint vertices. An halfedge defines exactly one half of an edge using fields such as a starting vertex, a twin halfedge defining the other half of the edge, and a next halfedge defining the next edge of the face the halfedge belongs to. A loop of twin halfedges traverse an edge, a loop of next halfedges traverse a face, and a loop of twin of next halfedges traverse a vertex. Refer to Scotty3D documentation and 15462/662 Computer Graphics lecture slides for more detailed explanation on implementation specifications and characteristics.



```
struct Halfedge
{
   Halfedge* twin;
   Halfedge* next;
   Vertex* vertex;
   Edge* edge;
   Face* face;
};
```

As such, the halfedge mesh representation easily supports local modifications of any given manifold mesh - just update halfedge connections accordingly. However, global operations on halfedge meshes cannot be performed with a couple pointer reassignments like in local operations. We must loop through appropriate faces, edges, vertices, or halfedges to identify appropriate elements to modify and apply the changes. The current expectation and implementation of Scotty3D are that these loops and inner logics work in a completely serial manner in order to ease the difficulty of student assignments. This is exactly the area that parallelism can significantly improve performance by spreading the original serial workload across multiple processors / tasks / threads.

The global operations we intend to tackle will be detailed in later **Goals and Deliverables** sections. To showcase an example here, we will parallelize triangulation operation. This operation takes a not necessarily triangular mesh and outputs a mesh consisting only of triangular faces. This is a per-face operation where we inspect each face and check if it is already a triangle or not. If it is not a triangle, we generate new faces using a fan triangulation. Since the fan triangulation approach on any given face does not affect any mesh connectivity of neighboring faces, we can trivially parallelize this operation by for example writing a GPU shader code for fan triangulation and generating a task calling this helper function per face. Unfortunately, most of the other global operations are not as easy as this example.



The Challenge

A mesh is an intertwined connectivity description between lists of faces, edges, vertices, and halfedges. Each element contains multiple references to other elements to maintain the entire mesh as manifold and valid. However, once we start individualizing the components and process them on parallel, we are bound to encounter consistency issues with references. Refer back to the triangulation example from **Background**. Even though the process sounds simple, there are actually multiple areas where there may be assignment conflicts between overlapping elements. For example, let's imagine we ran triangulate on a cube as in above image. Consider the bottom left and bottom right faces. We can see that the new edges created by fan triangulation both uses the center vertex. If an implementation of triangulation were to reassign points on this vertex, which value should we preserve - the result from bottom left face or the result from bottom right face? This problem implies that we need to firstly try to keep every operation we do within a parallel segment completely local so that no merge conflict can occur, and secondly devise a coherent and consistent rule that defines how to communicate and solve merge conflicts.

Conceptual constraints aside, there are other technical challenges as well. In terms of the workload distribution, since our mesh structure is specialized to support local connectivities, it is very hard to predict the workload balance between different parts of the mesh. Usually global operations perform series of per-element manipulation and the element vectors are not necessarily in any spatial order, meaning we can expect to exploit almost no locality in between steps. Since each global operation performs a significantly different set of manipulations on the mesh, it is impossible to claim that we have a case of high or low communication to computation ratio as this depends on the exact steps we need to take to complete the operation.

Resources

We will be using Scotty3D as our starter codebase. We will start from the public base code of Scotty3D which has no reference implementations of anything and then import parallelization support libraries to work on our project.

We do not have any book or paper we are using as a reference.

We will be using our own computers to run and test our implementation. The codebase is meant to be run on any personal computer environments, so we do not need any special computing machines. We also can cover both Windows and MacOS environments to check basic cross-platform capabilities.

Goals and Deliverables

We describe our goals and deliverables by development stage.

Stage	Plan to Achieve	Hope to Achieve	Reason	Deliverable
Set Up	Incorprate two out of the three parallel frameworks (ISPC, OpenMP, OpenMPI) into the existing codebase	Incorprate all three	This is necessary for implementing the parallel system and conducting experiments	Successful compile messages

Stage	Plan to Achieve	Hope to Achieve	Reason	Deliverable
Implementation	Parallel implementation of Triangulate	Parallel implementation of linear subdivision	For these operations, we expect there to be a significant gain in performance from parallelization because we can divide these opertations into smaller independent pieces.	Parallel operations produce correct results
Gather Benchmarks	Find three benchmarks that are big and representative of the problem space	Find five benchmarks	We need good benchmarks to test and improve the performance of our parallel system	A summary of each benchmark
Testing -- Load Balancing	Experiment with at least two load balancing scheme for each operation	Experiment with more load balancing scheme	From the lectures and assignments, we have learned that the performance of a parallel system vary greatly depending on the load balancing. We work to experiment to find the best option for each mesh opertation.	Speedup and time table detailing the results of the experiments
Testing -- Other	compare the performance of each parallel framework with various number of processors on our personal computers	test on the ghc and psc machines	We want to gain more insights into the performance of the system	Speedup and time table detailing the results of the experiments

Stage	Plan to Achieve	Hope to Achieve	Reason	Deliverable
Analysis	Detailed review of the experiments that were done, including the roadblocks that we have encountered, a comparison of the performance, and indepth analysis of likely explanation of the result			Project Report

At the poster seesion, we plan to show the speedup graphs for the systems we plan to implement. If we are able to modify the Graphical User Interface, we hope to add a live demo so that the speedup can be experienced more directly.

Overall, the system will be capable of conducting global mesh operations in parallel, and we hope to achieve close to linear speedup.

Platform Choice

Scotty3D is a 3D graphics software package that both team members are greatly familiar with due to previous exposures. The package is almost entirely built with C++, meaning the codebase is compatible with the parallelization strategies we intend to implement. This software is also designed to be ran on a single computer, making it easier to control external variables (such as other users sharing computing resources with us) and examine minor performance differences. However, due to the innate limitation of Scotty3D being ran on a single machine with serial implementations, we can expect to significantly improve the performance of our mesh operations with good parallelization schemes.

Schedule

The schedule below refers to the development stages outlined in the Goals and Deliverables section above.

Week of	To Do
Nov. 13	11/15 Propject Proposal Due, start Set Up
Nov. 20	Finish Set Up, begin Implementation, complete Gather Benchmarks
Nov. 27	Finish implementation, start Testing -- Load Balancing, 12/2 Milestone Report Due
Dec. 4	finish Testing -- Load Balancing, complete Testing -- Other
Dec. 11	complete analysis, 12/14 Final Report Due, 12/15 Poster Session

The schedule above outlines our general workflow. We will try to complete the genral work flow first before attempting the "Hope to Achieve" goals.