Helen Fang
EID: hyf72

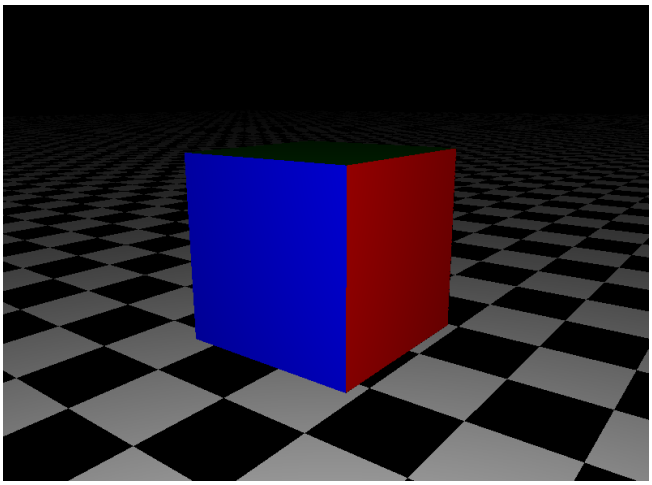
Subdivision Algorithms

General

The OpenGL/camera controls/shaders portions of this are taken almost exactly from the menger project. One thing I changed is to initialize the camera a fixed distance away from the mesh's bounds, to account for different mesh sizes.

The mesh class stores faces using a nested vector instead of uvec3 or uvec4s, in order to generalize the representation and allow for mixed face dimensions. This means that any algorithm can be used with any mesh and this can be navigated using keyboard controls (see README for details).

For reference, here is the original menger cube again (since I will be using it to demo each algorithm):



OBJ loading

`subdivision.cc/load_from_file()`

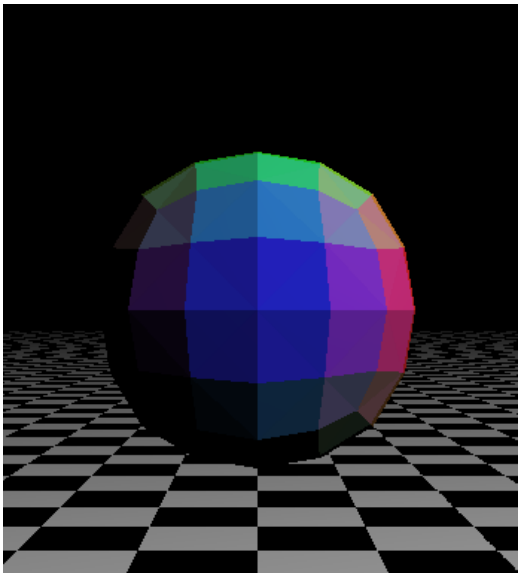
I think this is reasonably straightforward string tokenizing. For each line I read in from the file, I check the first char to load either as a face or a vertex. (For simplicity, I ignored all the other info). After, I searched for the delimiting spaces in the rest of the string and parsed ints/floats as appropriate.

Catmull-clark

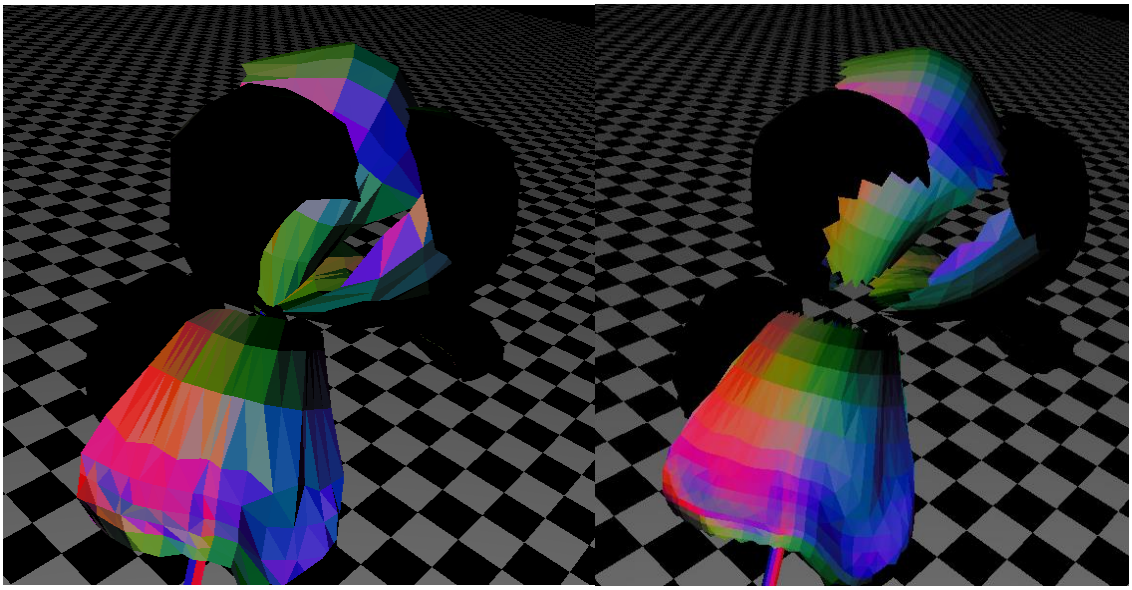
`subdivision.cc/catmull_clark()`

I coded up this algorithm first. The most difficult part of this for me was figuring out how to encode edge-face adjacencies in order to calculate the edge points, and I ended using a hash map (using a hash

function I found off math.stackexchange to hash my uvec2's). Additionally, I was careful about maintaining counterclockwise order when reconstructing my faces, making sure to add my vertices in a specific order. Otherwise, the code was pretty standard, I found the algorithm off of Wikipedia.



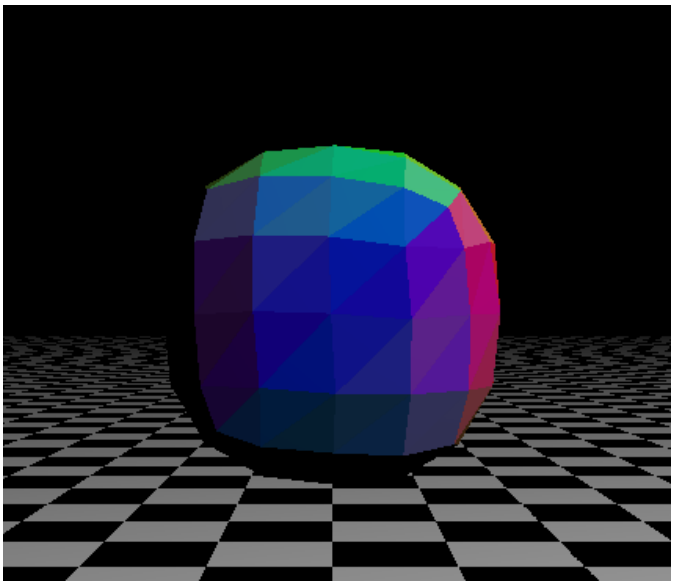
1 - Menger cube after 2 catmull-clark subdivisions



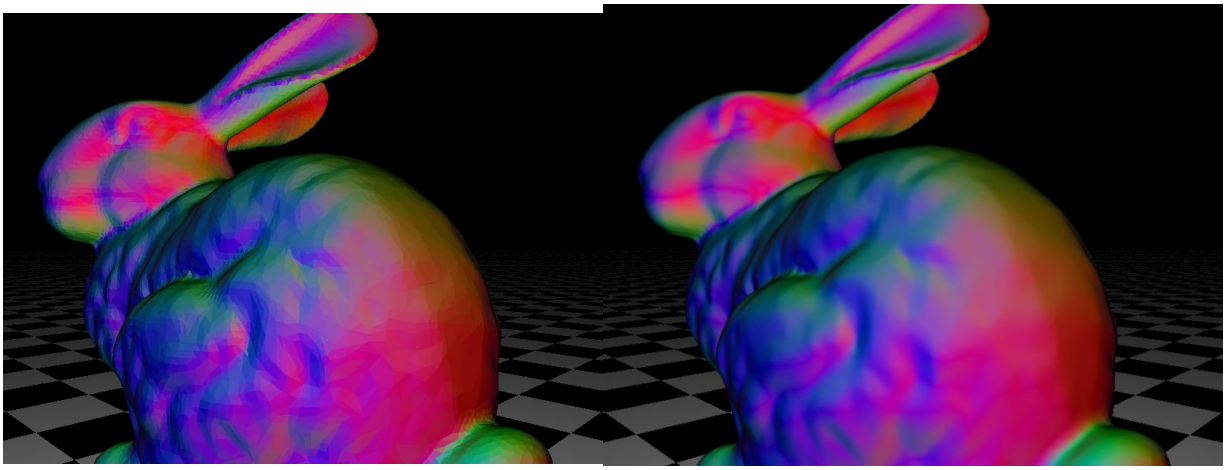
2- Before and after catmull-clark on an iris quad mesh

Loop

Because this only works with triangles, I added a hack to automatically split a face into triangles while I compute the subdivision, if I ever encounter a non-triangle face. I used a similar framework to catmull-clark where I used hash maps to save edge adjacency and "opposite vertex" information. Again, I followed a formula that I found online for the weight information, being careful to reconstruct faces with the vertices in a specific order.



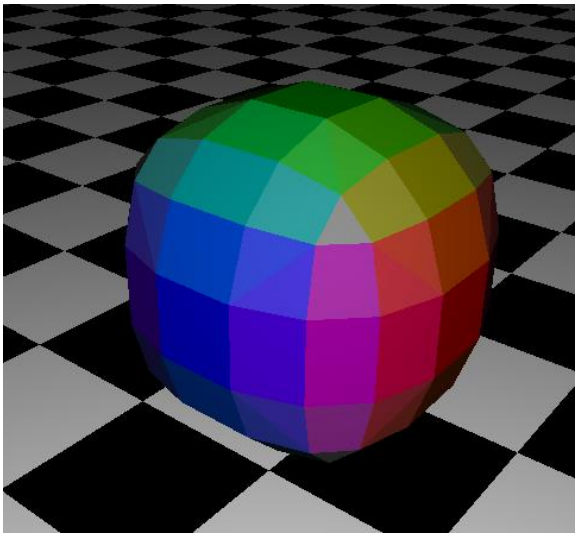
3- after 2 loop iterations on menger cube



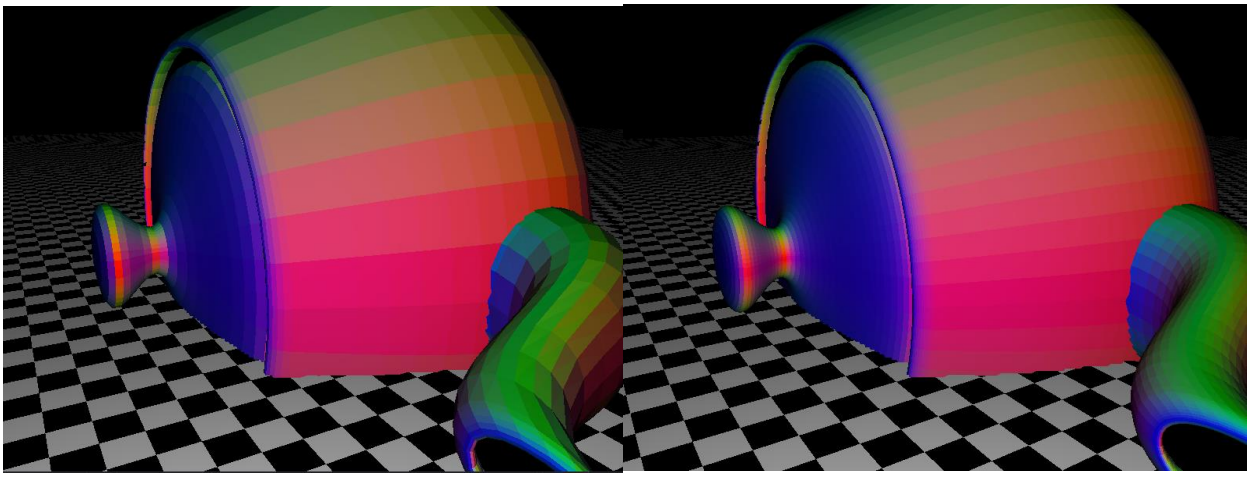
4- before and after loop subdivision on bunny trimesh

Doo-Sabin

I struggled the most with this algorithm, particularly when trying to rearrange the new points surrounding each original vertex into radial order. The approach that I ended up using was to 1) compute the average of the new vertices as the center 2) pick one of the surrounding points as an anchor point 3) order the other points based on the size of the angle formed by itself, the center, and the anchor. There were a lot of implementation details to handle, such as using the approximate face normal to decide when to add PI to the arccos value. An edge case that I struggled with was that acos(-1) either returned NaN, or otherwise messed up my calculations so I needed to just return PI if the angle was roughly straight. I've made it work for most, but unfortunately not all scenarios (ex. Doing loop -> doo-sabin will cause some weird empty patches).

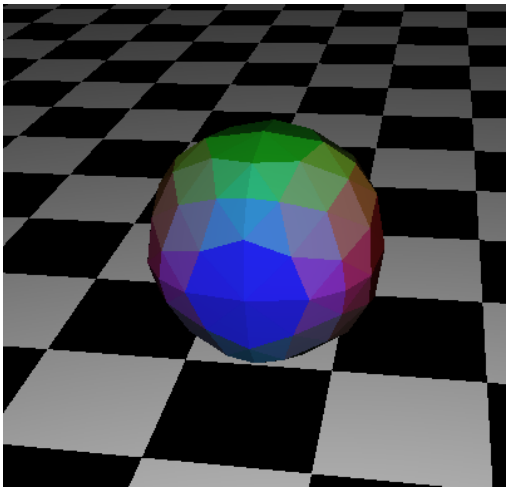


5- after 2 doo-sabin iterations on menger cube

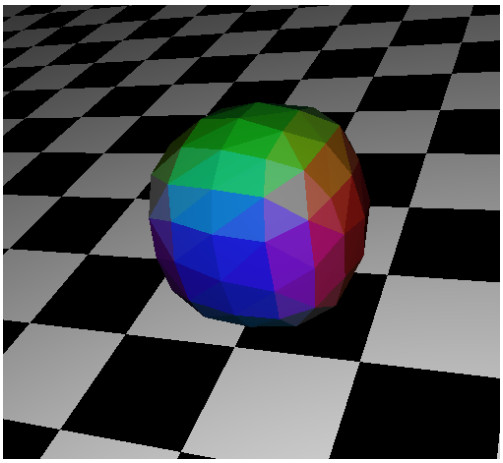


6- before, after doo-sabin on teapot quad/tri mixed mesh

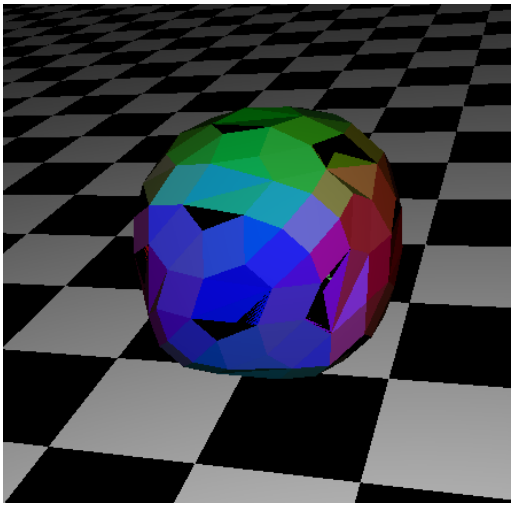
Mix & Match examples



7- catmull-clark, then loop



8 - doo-sabin then loop



9 - loop, then doo-sabin