Hack in std::map with Eigen

Zhouyuan Chen

April 2024

1 Problem Description

Recently, I needed to load a giant volumetric data, which took the computer forever to load and compute. With $1000 \times 1000 \times 1000$ volumetric data, if someone needs to generate a mesh from it by directly building mesh with a specific pattern in each voxel and simplifying it. In my terrible experience, it is totally insane and impossible to do. It is definitely a big challenge to the machine, I mean, You should never push your machine like this.

The good news is that there is a technique called Marching Windows[ZYP+22], that makes this kind of data operation easier. The Marching Window's idea is easy, given a window's size, then you take the chunks one by one and do simplification. When you finish the simpification, you write the current chunk's mesh into the disk and move the another window. In the end, you will operate the all voxels and get the final simplified version volumetric tetrahedron.

And the thing is, each time you can only move half the window's size in each direction. Therefore, naturally, here comes a problem, how to find the relationship between two overlapping windows?

2 Play std::map in a Hacky Way

I had a closer look at the source code of the Marching Windows[ZYP+22] and then realized they used a trick to make the implementation much easier than I thought. They used the std::map with the position key. A cool technique that I never thought about before.

Then I begin to write the code, something like:

```
// define the mapping
std::map<Eigen::Vector3d,unsigned int> mapping;
// do something
...
// find mapping, let's assume I have a position p0 and idx
if(mapping.count(p0))
    mapping[p0] = idx
```

But the compiler was not happy about this. Error occurred.

```
[build] /usr/include/c++/11/bits/stl_function.h:400:20: error: no match for
'operator<' (operand types are   const   Eigen::Matrix<double, 3, 1>' and
'const Eigen::Matrix<double, 3, 1>')
[build]  400 | { return __x < __y; }
```

Then I realized I needed to overwrite the less comparison method to make the compilation pass. I add such a structure and build the std::map with it.

```
// struct for less comparison method
struct VectorComparer
{
    bool operator()(const Eigen::Vector3d& a, const Eigen::Vector3d& b) const
    {
        return std::lexicographical_compare(
```

```
            a.data(),
            a.data() + a.size(),
            b.data(),
            b.data() + b.size());
    }
};

std::map<Eigen::Vector3d, bool, VectorComparer> mapping;
```

For now, the compiler is happy.

Have to keep in mind that, std::map's runtime complexity in inserting and searching is $O(\log N)$. It is fast, but not that fast, I mean $O(1)$, like hash. This is a lazy way to make the code run faster than directly traversing the whole point set.

References

[ZYP+22] Wenhua Zhang, Yating Yue, Hao Pan, Zhonggui Chen, Chuan Wang, Hanspeter Pfister, and Wenping Wang. Marching windows: Scalable mesh generation for volumetric data with multiple materials. *IEEE Transactions on Visualization and Computer Graphics*, 2022.