Lecture 3 (CGAL)

Panos Giannopoulos, Dror Atariah
AG TI
WS 2012/13
Exercises?
Outline

Convex Hull in 3d

Polyhedra and the Half Edge (or DCEL) data structure

Exercise 3
Convex hull of $n$ points is a convex polytope:

- $\leq n$ vertices (points)
- $\leq 2n - 4$ facets
- $\leq 3n - 6$ edges
Theorem

The convex hull of a set of $n$ points in $\mathbb{R}^3$ can be computed in $O(n \log n)$ time.

Here: $O(n^2)$-time algorithm
Convex Hull in 3d

Input: $P \subseteq \mathbb{R}^3$, $|P| = n$
Output: $\mathcal{CH}(P)$

1. Choose 4 points that are NOT coplanar. (Their convex hull is a tetrahedron.)
   Let $P_4 = \{p_1, \ldots, p_4\}$ be these points.
   
   $C \leftarrow \mathcal{CH}(P_4)$.

2. Let $P_r = \{p_1, \ldots, p_r\}$ for $r \geq 1$.
   ($p_5, \ldots, p_n$ are the remaining points.)

   For $r \leftarrow 5$ to $n$
   
   $\quad \triangleright$ Add $p_r$ to $\mathcal{CH}(P_{r-1})$.
   
   (construct $\mathcal{CH}(P_r)$ from $\mathcal{CH}(P_{r-1})$.)
   
   $C \leftarrow \mathcal{CH}(P_r)$.

3. Return $C$. 
Add \( p_r \) to \( CH(P_{r-1}) \):

1. If \( p_r \) lies inside (or on the boundary of) \( CH(P_{r-1}) \), then \( CH(P_r) = CH(P_{r-1}) \).

2. If \( p_r \) lies outside \( CH(P_{r-1}) \), find visible region of \( p_r \) on \( CH(P_{r-1}) \) (test every facet) (connected region of visible facets, enclosed by the horizon).

3. Replace visible region by facets (triangles) connecting \( p_r \) to it’s horizon.
3. Replace visible region by facets (triangles) connecting \( p_r \) to it’s horizon:

4. **Note:** Merge any new triangle with their co-planar faces of \( \mathcal{CH}(P_{r-1}) \) (if any).
The boundary of a 3d convex polytope can be interpreted as a planar graph:

We store the convex hull in the form of a *Doubly-Connected Edge List* (as for planar subdivisions)
Doubly-Connected Edge List:

- geometric and topological information stored in records (vertex, face, half-edge)
- two oriented half-edges for an edge (origin, destination)
- incident face of a half-edge lies to its left
- counterclockwise traversal of a face

The half-edge record of a half-edge $\vec{e}$ stores a pointer $\text{Origin}(\vec{e})$ to its origin, a pointer $\text{Twin}(\vec{e})$ to its twin half-edge, and a pointer $\text{Next}(\vec{e})$ to its next edge on the boundary of the face. The half-edge record also stores pointers $\text{Prev}(\vec{e})$ and $\text{IncidentFace}(\vec{e})$ to the previous edge and the face that it bounds, respectively.
For a cube it looks like:
CGAL::Polyhedron_3<PolyhedronTraits_3> for storing the convex hull (Uses a DCEL, no need for low lever operations from the user):

```cpp
template < class PolyhedronTraits_3,
    class PolyhedronItems_3 = CGAL::Polyhedron_items_3,
    template < class T, class I> class HalfedgeDS = CGAL::HalfedgeDS_default,
    class Alloc = CGAL_ALLOCATOR(int)>
    class Polyhedron_3;
```

Use a Kernel as `PolyhedronTraits_3`:

```cpp```
typedef CGAL::Simple_cartesian<double> Kernel;
typedef CGAL::Polyhedron_3<Kernel> Polyhedron;
```
Container (It provides all sorts of iterators):

```cpp
typedef CGAL::Simple_cartesian<double> Kernel;
typedef Kernel::Point_3 Point_3;
typedef CGAL::Polyhedron_3<Kernel> Polyhedron;
typedef Polyhedron::Vertex_iterator Vertex_iterator;

int main() {
    Point_3 p(1.0, 0.0, 0.0);
    Point_3 q(0.0, 1.0, 0.0);
    Point_3 r(0.0, 0.0, 1.0);
    Point_3 s(0.0, 0.0, 0.0);

    Polyhedron P;
    P.make_tetrahedron(p, q, r, s);
    CGAL::set_ascii_mode(std::cout);
    for (Vertex_iterator v = P.vertices_begin(); v != P.vertices_end(); ++v)
        std::cout << v->point() << std::endl;
    return 0;
}
```

It provides handles (e.g., Polyhedron::Halfedge_handle), as trivial, lightweight iterators.
(It supports plane information by default)

E.g., compute plane equation of a plane of a facet:

```
struct Plane_equation {
    template <class Facet>
    typename Facet::Plane_3 operator()( Facet& f) {
        typename Facet::Halfedge_handle h = f.halfedge();
        typedef typename Facet::Plane_3 Plane;
        return Plane( h->vertex()->point(),
                      h->next()->vertex()->point(),
                      h->next()->next()->vertex()->point());
    }
};
```
...  
typedef Kernel::Plane_3 Plane_3;
typedef CGAL::Polyhedron_3<Kernel> Polyhedron;

```cpp
int main() {
    Point_3 p( 1, 0, 0);
    Point_3 q( 0, 1, 0);
    Point_3 r( 0, 0, 1);
    Point_3 s( 0, 0, 0);
    Polyhedron P;
    P.make_tetrahedron( p, q, r, s);
    std::transform( P.facets_begin(), P.facets_end(), P.planes_begin(),
                     Plane_equation());
    CGAL::set_pretty_mode( std::cout);
    std::copy( P.planes_begin(), P.planes_end(),
               std::ostream_iterator<Plane_3>( std::cout, "\n"));
    return 0;
}
```
It supports I/O from/to .OFF file format:

```cpp
template <class PolyhedronTraits_3>
ostream& out << CGAL::Polyhedron_3<PolyhedronTraits_3>& P

template <class PolyhedronTraits_3>
istream& in >> CGAL::Polyhedron_3<PolyhedronTraits_3>& P
```

.OFF format
Exercise 3

- Implement the $O(n^2)$-time algorithm for computing the convex hull of a point set in $\mathbb{R}^3$.
- Use `CGAL::Polyhedron_3<PolyhedronTraits_3>`.
- Use predicates from the Kernel to check visibility.
- Construct the horizon by performing a depth first search.
- Possible functions for updating the current convex hull:
  - `erase_facet(...)`
  - `erase_connected_component(...)`
  - `add_vertex_and_facet_to_border(...)`
  - `add_facet_to_border(...)`
- Visualize the steps of the algorithm (partial results) with a .OFF viewer (e.g., Polyhedron_3 CGAL Demo).