Lecture 15 - Constructive Solid Geometry

Philip Caplan
CSCI 0422 - Geometric Modeling (Spring 2022)
Learning objectives

By the end of this lecture you will be able to:

- represent complex shapes from simpler ones using binary set operations: union, difference, intersection,
- use a binary space partition to perform discrete solid modeling operations,
- intersect polygons with planes, creating two polygons on either side of the plane.
Getting started...

Switch Host & Client today!

$ git pull
$ make update
$ cmake .
$ make template_class15_practice
$ cmake .

Compiling and running the exercise:

$ make class15_practice

Compiling and running the solution (after class):

$ make class15_practice_sol
Detective work: find the bug.

```cpp
// represents a simplified version of the HalfEdgeMesh container (only holds HalfEdge's)
std::set< std::unique_ptr<HalfEdge> , std::less<> > edges;

// add 5 edges to the container
for (int i = 0; i < 5; i++)
  edges.insert( std::make_unique<HalfEdge>() );

// assign the cost of all edges
double cost = 100;
for (auto & e : edges) {
  e->cost = cost ++;
  printf("cost = %g\n",e->cost);
}

// comparison used to determine order in priority queue
struct CompareCost {
  bool operator() ( const HalfEdge* e1 , const HalfEdge* e2) const {
    return e1->cost < e2->cost;
  }
};
std::multiset<HalfEdge*,CompareCost> pq;

// add the first edge in the container to the priority queue
HalfEdge* e = edges.begin()->get();
pq.insert(e);

// delete this edge from the simplified HalfEdgeMesh container
auto it = edges.find(e);
assert( it != edges.end() );
edges.erase(it);

// print the cost of the edge with the highest priority
e = * pq.begin();
printf("top edge cost = %g, queue size = %lu\n",e->cost,pq.size());
//pq.erase(e); // uncomment to show even more memcheck errors
```
Constructive Solid Geometry.

how would you create a mesh of this geometry?
Solid represented analytically via set operations.
Solid represented analytically via set operations.
Boolean operators in CAD software.
Representing a solid via CSG with a binary space partition.

- Represent each shape as a mesh.
- To perform a CSG operation \((\cap, \cup, -)\), determine "side" of each polygon w.r.t. all other polygons.
- Polygons might intersect, in which case they need to be split in two (front, back).
- How to determine side of each polygon w.r.t. other polygons? (efficiently)
Representing a solid via CSG with a binary space partition.

- Represent each shape as a mesh.
- To perform a CSG operation ($\cap$, $\cup$, $-$), determine "side" of each polygon w.r.t. all other polygons.
- Polygons might intersect, in which case they need to be split in two (front, back).
- How to determine side of each polygon w.r.t. other polygons? (efficiently)

**binary space partition**
Building a Binary Space Partition.

**main idea:** each polygon defines a plane → divide other polygons to be in front/back of this plane.

1. pick a starting polygon \( p \) (with normal \( n \)), space in front is \( P^+ \), behind in \( P^- \),
2. initialize front and back arrays of polygons (children),
3. loop through each polygon \( q \):
   (a) determine "side" (front or back) of each vertex in \( q \),
   (b) if all sides or the same, place \( q \) into either front or back appropriately,
   (c) otherwise, split \( q \) into two polygon and add to front and back branches,
4. build the front and back branches using the polygons that were just classified.
Intersecting planes with polygons.

Given a polygon $p$ with normal $n$ and some point $c \rightarrow$ defines a plane $\mathcal{P}$. Side $s$ of a point $v$ can be determined via the sign of

$$s = n \cdot v - n \cdot c.$$ 

$s < 0 \ ? \ v$ is in $\mathcal{P}^-$, otherwise $v$ is in $\mathcal{P}^+$. 
Intersecting planes with polygons.

Given a polygon \( p \) with normal \( \mathbf{n} \) and some point \( \mathbf{c} \rightarrow \) defines a plane \( \mathcal{P} \). Side \( s \) of a point \( \mathbf{v} \) can be determined via the sign of

\[
    s = \mathbf{n} \cdot \mathbf{v} - \mathbf{n} \cdot \mathbf{c}.
\]

\( s < 0 \) ? \( \mathbf{v} \) is in \( \mathcal{P}^- \), otherwise \( \mathbf{v} \) is in \( \mathcal{P}^+ \).
Final polygons may not be the nicest, but solid is represented.
Other uses of Binary Space Partitions.

hidden surface removal: which triangle is in front of another triangle?
Rest of class time for Project 3 work.