Report for Computer GraphicII, HW1 3D convex hull algorithm and collision detection

Dai ZiJia 2022233158

October 5, 2022

1 Part 1 (20 points)

1. (5 points) Prove the intersection of two convex set is still a convex set.

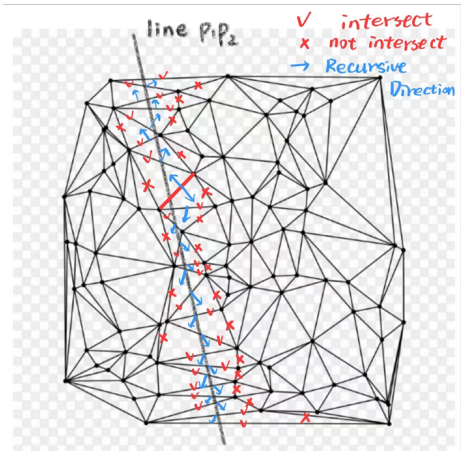
   Pf. Assume that there are two convex set P and Q. $\forall x, y \in P \cap Q$. Since P is a convex set $\Rightarrow$ segment xy is in P. Similarily, segment xy is in Q. So segment xy is in $P \cap Q \Rightarrow P \cap Q$ is still a convex set, which means the intersection of two convex set is still a convex set.

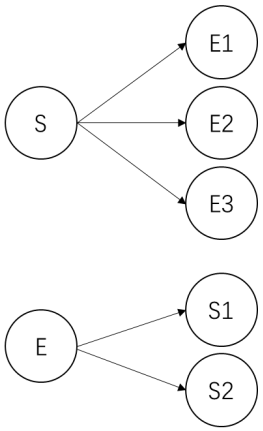
2. (15 points) If a plane is divided into polygons by line segments, please design a data structure to store the division information so that for the given line passing two points p_1 and p_2 on the plane, it is efficient to find all the polygons intersected with the line. Please provide the main idea and pseudocode of the algorithm and give the complexity analysis.

   Data structure:

```
struct Edge{
  Surface s1;
  Surface s2;
  Surface s3;
  Point p1,p2;
  int index;
};
struct Surface{
  Edge e1;
  Edge e2;
  Edge e3;
  int index;
};
```

   Main idea: As the following picture, find the first edge intersect with p_1p_2. The plane linked with the first edge will intersect with p_1p_2 too. Then search the edge of plane to find another edge intersect with p_1p_2. At the same time, save the result in an array. This is a recursive process and stop until the intersected edge is boundary.





Complexity analysis: The algorithm1 takes $O(n)$ to find the first intersect edge. For algorithm2 takes $O(n)$ to visit surface link to the intersected edge. So, the total time complexity is $O(n)$.

Pseudocode:

Algorithm 1: Find the first intersect edge

Input: line p_1p_2, Edges
Output: The first intersect edge

```
for e in Edges do
    if e intersect with p1p2 then
        isE_{e->index} = True;
        break;
    end
end
return e
```

Algorithm 2: Find intersect surfaces

Input: line p_1p_2, Edges, Surfaces, first intersect edge
Output: isE,isS

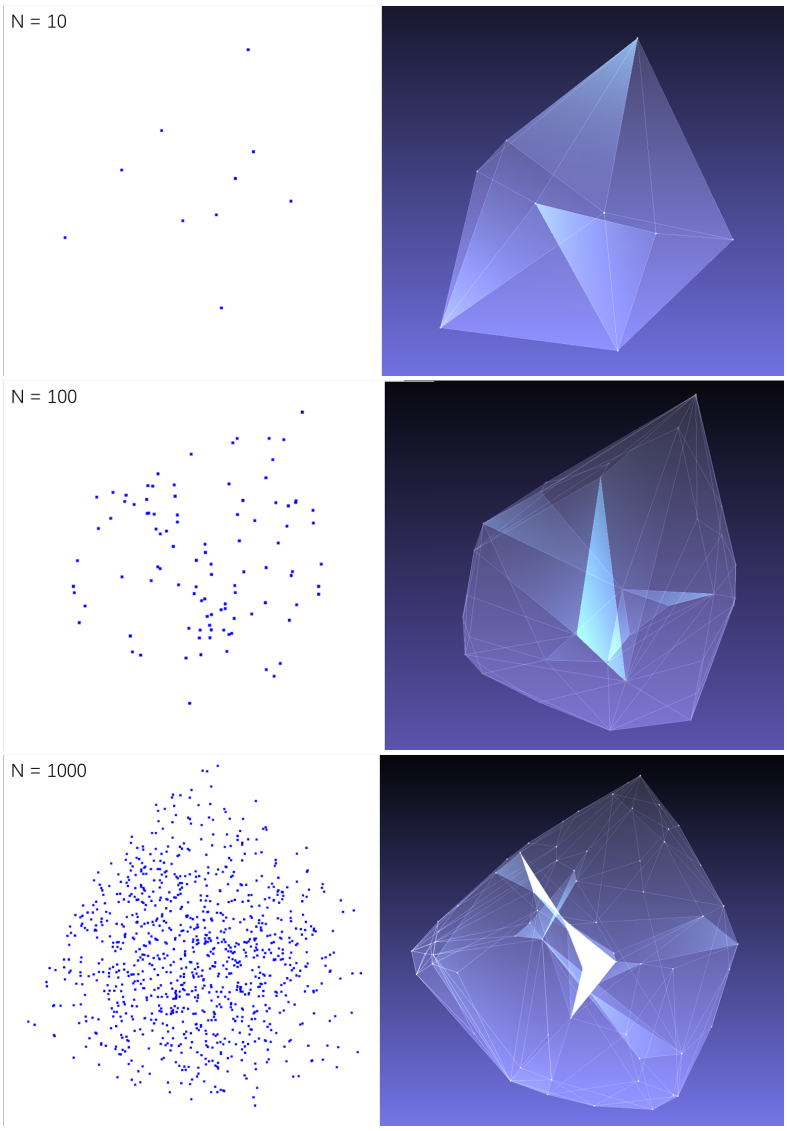
```
1  Findsurface(e): foreach surface s next to edge e do
2      if isS_{s->index} is not True then
3          isS_{s->index} = True;
4          Findedge(s);
5      end
6  end
7  Findedge(s): foreach edge e next to surface e do
8      if isE_{e->index} is not True and e intersect with p1p2 then
9          isE_{e->index} = True;
10         Findedge(e);
11     end
12 end
13 Findsurface(first intersect edge);
14 return isE,isS
```

2 Part 2 (80 points)

2.1 3D convex hull algorithm(55 points)

(note: you need to show the convex hull visualization result; remember to state the data structure you use; analysis the runtime with incremental number of points; don't make the example too simple(like the simple box or tetrahedron))

Result:



Data structure:

```
1  struct Point{
2    double x;
3    double y;
4    double z;
5    int index;
6
7    Point(double a = 0, double b = 0, double c = 0, int i = -1){ x = a; y
         = b; z = c; index = i;}
8  }P[MAX];
9
10 struct Surface{
11   Point p0;
12   Point p1;
13   Point p2;
14
15   Surface() {};
16   Surface(Point a, Point b, Point c) { p0 = a; p1 = b; p2 = c; }
17 }S[MAX];
```

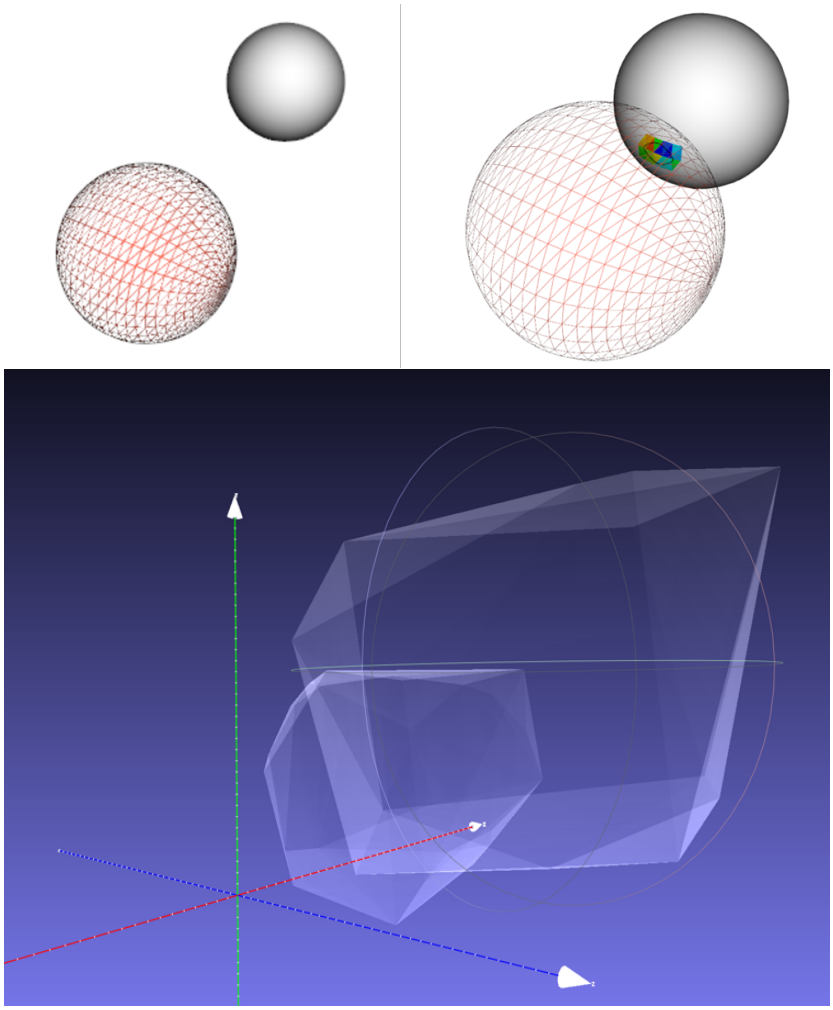
Runtime: As the points increases, the cost of time becomes larger. The time complexity is $O(n^2)$.

Number	100	1000	5000	10000
Time(ms)	18.6	83.2	815.6	5277.6

2.2 Collision detection(25 points)

(note: need collision visualization and algorithm description)

Collision visualization:



Discription:

- Broad phase. Using a sphere to enclose objects, if their center distance is less than the sum of their radii, then they are likely to collide.
- Narrow phase. I use OBB-SAT in this step. OBB is to determine the size and orientation of the box according to the set shape of the object, so that the most compact box can be selected to represent the object. Separating Axis law shows that two convex polygons, if we can find an Axis such that the projections of the two objects on this Axis do not overlap each

other, then there is no collision between the two objects, this Axis is called Separating Axis.