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

Jiang Xinhong 2020533075

October 5, 2022

Acknowledgements: Deadline: 2022-10-5 23:59:59

You can choose C++ or Python, and no restrictions on programming framework. You can freely use frameworks such as openGL.

The **report** submits as a PDF file to gradscope, the programming part should package all the files include code, input files, executable file, readme.txt, and report. The **package** name is **your_student_name+student_id.zip**.

You will get Zero if the code not passing the plagiarism check.

1 Part 1 (20 points)

1. (5 points) Prove the intersection of two convex set is still a convex set.
   proof.
   Let C_1, C_2 be two convex set.
   Let us assume that $C_1 \cap C_2$ is non-convex. Then, there must be points $A, B \in C_1 \cap C_2$ such that line segment AB is not a subset of $C_1 \cap C_2$.
   Thus, we must have a point $X \in AB$ for which $x \notin C_1 \cap C_2$, now there are 3 possible cases:
   (i) $X \in C_1 - C_2$, which is contradicted to definition $X \in C_2$
   $(ii\ )X \in C_2 - C_1$, which is contradicted to definition $X \in C_1$
   (iii) $X \notin C_1 \cup C_2$, which is contradicted to definition $X \in C_1, C_2$
   Thus, the region of intersection is convex.

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: $S = \{\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}, \{(x_4, y_4), (x_5, y_5), (x_6, y_6), (x_7, y_7)\}, \dots\}$
   We use a polygon mesh to store it. A set of set in which stores serials of dots, representing the polygons. In each subset, the dots can constructed a polygon
   Algorithm main idea: Iterate over the collection of polygons, determinate whether are them across with p_1p_2, if 2 edges in the polygons are cross with p_1p_2, the the polygons is intersected with the line.
   Pseudocode:

Algorithm 1 polygons divided

Input: collection S, dots p_1, p_2
Output: interpoly = [True, False, True,...] `/* consistent with the list` S `order */`

```
1  for polygon in S do
2      for line segments in polygon do
3          if intersection with p1p2 then
4              record intersection
5          end
6      end
7      if p1p2 across with 2 edge then
8          interpoly[i] = True /* corresponding Bool value */
9      end
10 end
```

   Complexity:

The number of polygons is $O(n)$, and the edges of a polygons is $O(1)$. Thus, the complexity of whole algorithm is $O(n)$

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))

Main idea of algorithm:

The basic idea is to add points sequentially to the convex hull.

The specific operation is to initially select three non-collinear points (A, B, C) to form a closed graph composed of two planes ABC and CBA (they have different directions), and then add points one by one.

If this point is in the current convex hull, ignore it. Otherwise, imagine the point becomes a lamp, emitting light in all directions. Light hits the convex hull we've already got, creating bright and dark sides Deletes all points in the bright surface (that can be hit by the light) and reconstructs the plane from the shadow boundary with the new points.

Pseudo-code:

Our algorithm received a point cloud and output a surface mesh constructed by triangular .

Algorithm 2 Convex

Input: collection S, dots p_1, p_2

Initial Convex $= \{(a_1, a_2, a_3), (a_3, a_2, a_1)\}$

for *dots not visited* **do**

  `// add it to the convex hull`

  for *all surface in convex hull* **do**

    if *the point above the surface* **then**

      delet the surface from mesh

      add new surface constructed by the dot

    end

  end

end

Complexity:

For n dots in cloud, the edges and surfaces is also $O(n)$. As there are Two loops nested, each of at most $O(n)$, and the inner loop will decrease as the program progresses.

Thus the complexity is $O(n \log(n))$

visualization:

We randomly generate points :

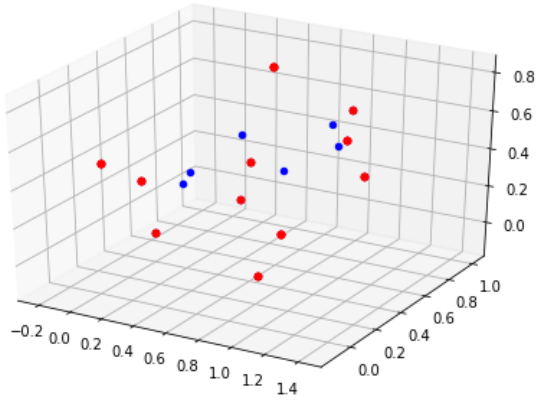

Figure 1: Scatter plot where the points on the convex hull is red while blue point is inner the hull

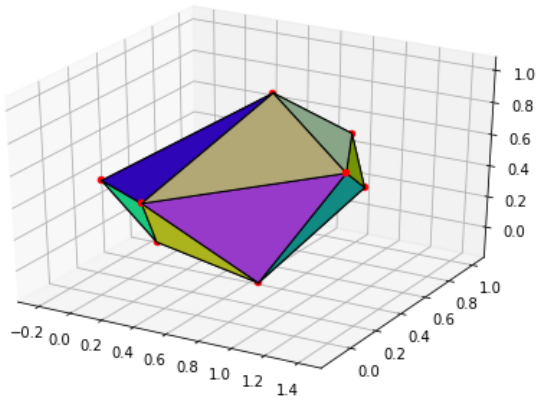

Figure 2: The convex hull we get

2.2 Collision detection(25 points)

(note: need collision visualization and algorithm description)