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

XXX number

October 4, 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.
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.

1. Proof: Consider in E^d, two convex sets S_1 and S_2 for $(p_1, p_2, \cdots, p_k)$ and $(q_1, q_2, \cdots, q_\kappa)$, respectively.

$$S_1 = \{s_1 | s_1 = \sum_{i=1}^{k} \alpha_i p_i, \text{where } \alpha_i \geq 0, \sum_{i=1}^{k} \alpha_i = 1\}$$
$$S_2 = \{s_2 | s_2 = \sum_{i=1}^{\kappa} \beta_i q_i, \text{where } \beta_i \geq 0, \sum_{i=1}^{\kappa} \beta_i = 1\}$$

Then we denote the intersection of S_1 and S_2 by $S_0 = S_1 \cap S_2$, $\forall s_0 \in S_0$,

$$\begin{cases} s_0 & = \sum_{i=1}^{k} \alpha_i p_i, \text{where } \alpha_i \geq 0, \sum_{i=1}^{k} \alpha_i = 1 \\ s_0 & = \sum_{i=1}^{\kappa} \beta_i q_i, \text{where } \beta_i \geq 0, \sum_{i=1}^{\kappa} \beta_i = 1 \end{cases}$$

Add two equation together, and we can have

$$s_0 = \sum_{i=1}^{k} \frac{\alpha_i}{2} p_i + \sum_{i=1}^{\kappa} \frac{\beta_i}{2} q_i$$

Embed $(p_1, p_2, \cdots, p_k)$ and $(q_1, q_2, \cdots, q_\kappa)$ together to generate a new set of points $(v_1, v_2, \cdots, v_{k+\kappa})$, where $v_i = \begin{cases} p_i & i \leq k \\ q_{i-k}, i > k \end{cases}$, and its coefficients $\gamma_i = \begin{cases} \alpha_i/2 & i \leq k \\ \beta_{i-k}/2, i > k \end{cases}$

Rewrite s_0 into the forms by v_i, γ_i that

$$s_0 = \sum_{i=1}^{K} \gamma_i v_i$$

where $K = k + \kappa$, and , $\gamma_i \geq 0, \sum_{i=1}^{K} \gamma_i = \sum_{i=1}^{k} \frac{\alpha_i}{2} + \sum_{i=1}^{\kappa} \frac{\beta_i}{2} = 1$

Thus

$$S_0 = \{s_0 | s_0 = \sum_{i=1}^{K} \gamma_i v_i, \gamma_i \geq 0, \sum_{i=1}^{K} \gamma_i = 1\}$$

which implies *the intersection of two convex set is still a convex set*.

2. Firstly, let us talk about how we store the division information and the corresponding time complexity.

   1)A data structure is designed by

a. store all the edges by the form that starts from one of the point, and points to another point, i.e. ray's origin and ray's direction(without normalization), $r = \vec{o} + t\vec{d}$, where $t \in [0, 1]$

b. store all the edges belonging to which set of facets, respectively.
   1.1)The time complexity for storing all the edges by ray's origin and direction and storing all the edges belonging to which facets is $O(n)$, since we only need to traverse all the edges and calculate the direction.
   1.2)The space complexity for constructing the data structure is $O(4e + 2e)$=$O(e)$.
   Denote $e = \#$ edges, we use two vec2 to store origin and direction for each edge, and please notice that each edge belongs to two facets at most, thus we use two integers to store belonging facets for each edge.

   2)Since we know the bounding box of all points of the polygon, we can change p_1, p_2 to the upper bound and the lower bound of the bounding box. It is also easily to prove that the line cannot intersect the polygon outside p_1p_2. Then we convert the line into a line segment.

   3)Main idea of our algorithm:

   for an edge $r = o + td$, and given line segment p_1p_2: $r_p = p_1 + t(p_2 - p_1)$

2.1 using cross product to identify whether the edge lies at the same side of p_1p_2, if yes, return false, else, further check by 2.2

2.2 now just need to consider the intersection of two rays, finally check the range of t

3)Pseudocode of the algorithm:

Algorithm 1 find the intersected polygons

for $\vec{o}, \vec{d} \leftarrow e$ **do**
  // using cross product to check whether lies at the same side
  $q_1 \leftarrow o, q_2 \leftarrow o + d$
  $c_1 = p_1\vec{q}_1 \times p_1\vec{p}_2$
  $c_2 = p_1\vec{q}_2 \times p_1\vec{p}_2$
  if $c_1c_2 >= 0$ // On the same side, or potentially tangent with polygons **then**
    continue
  end if
  // calculate the corresponding t
  $\epsilon \leftarrow 1e-5$
  $t \leftarrow \frac{p_1-o}{d-p_2+p_1+\epsilon}$
  if $\epsilon < t < 1$ **then**
    mark polygon(e) as $True$
  end if
  continue
end for

4)Complexity analysis: The space complexity is $O(e)$ obviously.
The time complexity is $O(e)$, too. Because we only need to traverse all the edge.