Report of Assignment 5

Zhihan Li

1600010653

May 16, 2018

Question 1

Answer Direct implementation of such hidden surface removal algorithm always performs $O(nmk)$ times intersection operation, because we have to judge whether a ray intersects with polygons and decide which the intersection to display, which means every pixel-polygon pair must be tested.

However, there are optimization strategies to this algorithm. An approach is to omit unnecessary judgement outside projected polygons. For example, we may use scan line algorithm to find admissable pixels for a polygon on a specific scan line, together with the distance to the camera stored pixel-wise. We may then find the closest polygon for each pixel according to such intervals.

This optimization strategy makes advantage of the sparsity of pixels related to a specific polygon — most polygon will be small enough and cover only a small fraction of area. Construction of scan lines takes $O(mk)$ time, while finding the closest may take only $O\left(\sum_{i=1}^{k} A_i\right)$ if a buffer of length $O(n)$ is used to store the partial closest distance, where A_i is the projected area to the screen (image space) of i-th polygon.

In real applications, $\sum_{i=1}^{k} A_i \sim O(mn)$. An important case is that these polygons forms a fine division of a surface. In this case, sum of the projected area is roughly CA, where A is the project area of the surface, and $C = 2$ if the surface is convex or slightly larger if not, which means $\sum_{i=1}^{k} A_i \approx CA \leq Cmn = O(mn)$. Another case is that some of the polygons go very large and approximately cover the whole screen. However, such polygon could only occur few times, which yields $O(mn)$ also.

In conclusion, $\sum_{i=1}^{k} A_i \sim O(mn)$ and the overall time complexity is

$$O(mn) + O(mk) = O(mn) \tag{1}$$

if k does not go very large. This explains the constant time complexity.

Question 2

Answer One explanation of the problem requires to judge whether polygon P lies in a half space split by the plane of planar polygon Q. The algorithm is given in Algorithm 1.

Data: List of vertices of planar polygon P $(P_1, P_2, \cdots, P_n)$ and Q $(Q_1, Q_2, \cdots, Q_m)$
Result: Whether P lies in one side of Q
$\mathbf{n} \leftarrow \overrightarrow{Q_1Q_2} \times \overrightarrow{Q_1Q_3}$;
$s = \operatorname{sgn} \overrightarrow{Q_1P_1} \cdot \mathbf{n}$;
if $s = 0$ **then**
  return *False*;
end
for *i from* 2 *to m* **do**
  $s' = \operatorname{sgn} \overrightarrow{Q_1P_i} \cdot \mathbf{n}$;
  if $s \neq s'$ **then**
    return *False*;
  end
end
return *True*;

Algorithm 1: Algorithm to check whether P lies in a half space split by the plane of Q

Question 3

Answer The program is given in `53LineSegment/main.py`. The initial value of d for different cases is given in the program.

Question 4

Answer We only have to show one eighth of the circle because augmentation can be done by $(x, y) \mapsto (x, -y), (-x, y), (y, x)$ consecutively. We can construct the circle in 270–315° because sloop of this interval lies in $[0, 1]$.

The program is given in `54Circle/main.py`.