

1. (a)

- Ambient light: intensity of reflected ambient light is  $RL$  where  $R$  is the ambient material property and  $L$  the intensity of the ambient light emitted by the light source. [2]
- Diffuse light: intensity of the reflected diffuse light is  $R((l - p)^t n)L$  where  $R$  is the diffuse material property and  $L$  the intensity of the diffuse light emitted by the light source. [3]
- Specular light: intensity of the reflected specular light is  $R(r^t(v - p))^\alpha L$  where  $R$  is the specular material property,  $L$  the intensity of the incoming light,  $\alpha$  the shininess exponent, and  $r$  the direction of perfect reflection. [3]
- To compute  $r$ :  $r + s = 2(s^t n)n$  with  $s = (l - p)/\|l - p\|$ , so  $r = (2((l - p)^t n)n - l + p)/\|l - p\|$  [2]

Total: [10]

1. (b)

- Gouraud shading evaluates the illumination model at each vertex of a polygon and interpolates the colours in the interior of the polygon. [2]
- Phong shading evaluates the illumination model for each pixel in the viewing plane using interpolated surface normals given for the vertices of the polygon. [2]
- Advantages of Phong shading are better quality of resulting shades, in particular specular highlights and for large polygons [2]
- Disadvantage is that information about lights, material properties and normals must be available until the fragment shading phase in the graphics pipeline. This makes it slower and harder to implement than Gouraud shading. [3]

Total: [9]

1. (c)

- Specular reflection depends only on the direction to the viewer and to the light source, but does not consider a surface orientation. Hence, rotating the polygon using Phong's illumination model with fixed viewer and light position will not change the specular highlights. However, for the scratched polygon the highlights also depend on the surface orientation given by  $t$ . [3]
- In order to consider the surface orientation in Phong's illumination model one could modify the normal used in the computations based on the scratch direction  $t$ . One could choose the normal  $n$  orthogonal to  $t$  such that that component of  $l$  in direction  $n$  is maximal, i.e. set  $n = d - (d^t t)t$ . (Alternative answers acceptable, e.g. anisotropic specular material property). [3]

Total: [5]

2. (a)

- Translate the light source to the origin by vector  $-l$ . [1]
- Apply perspective projection  $x = -xl_y/y$ ,  $y = -yl_y/y$ ,  $z = -zl_y/y$ . [2]
- Translate back by vector  $l$ . [1]
- Complete transformation:  $x' = l_x - \frac{x-l_x}{(y-l_y)/l_y}$ ,  $y' = 0$ ,  $z' = l_z - \frac{z-l_z}{(y-l_y)/l_y}$  [2]  
 (Alternative answer using homogeneous coordinates is acceptable).

Total: [6]

2. (b)

- The projection of  $p$  lies on the line  $p + lt$ ,  $t \in \mathbb{R}_0^+$ . [1]
- Intersecting this line with the plane  $n^t x - d = 0$  [1]
- gives  $t = \frac{d-n_x p_x - n_y p_y - n_z p_z}{n_x l_x + n_y l_y + n_z l_z}$  [3]
- So the projected point is  $q = p + \frac{d-n_x p_x - n_y p_y - n_z p_z}{n_x l_x + n_y l_y + n_z l_z} l$  [1]
- In coordinates this gives:

$$q_x = \frac{p_x(n_y l_y + n_z l_z) - p_y n_y l_x - p_z n_z l_x + d l_x}{n_x l_x + n_y l_y + n_z l_z}$$

and similar

$$q_y = \frac{p_y(n_x l_x + n_z l_z) - p_x n_x l_y - p_z n_z l_x + d l_y}{n_x l_x + n_y l_y + n_z l_z}$$

$$q_z = \frac{p_z(n_x l_x + n_y l_y) - p_x n_x l_y - p_y n_x l_x + d l_z}{n_x l_x + n_y l_y + n_z l_z}$$

[4]

- So the homogeneous projection matrix is

$$\begin{bmatrix} n_y l_y + n_z l_z & -n_y l_x & -n_z l_x & d l_x \\ -n_x l_y & n_x l_x + n_z l_z & -n_z l_y & d l_y \\ -n_x l_z & -n_y l_z & n_x l_x + n_y l_y & d l_z \\ 0 & 0 & 0 & n_x l_x + n_y l_y + n_z l_z \end{bmatrix}$$

[4]

Total: [14]

2. (c)

- An extended light source may only be partially “visible” from a point. The areas where the light source is fully occluded are in the full shadow; the areas where the light source is fully visible are fully lit; the areas in between are partially lit depending on how much of the light source is visible from the position. Projecting the polygons on a plane will only produce areas fully lit or not lit at all without soft shadows. [2]
- A simple approach to soft shadows is to sample the light source and compute a shadow polygon for each sample point. The occlusion maps of the samples are combined into an attenuation map where each pixel stores the number of sample points on the light source that are occluded. (Alternative answers possible, e.g. a modification of the above merges shadow maps instead of occlusion maps—see layered attenuation maps; single sample approach with inner/outer penumbra based on distance to lit/shadow pixel from shadow/lit pixel in shadow map; shadow volume approaches). [3]

Total: [5]

3. (a)

- Store the vertices of the polygon in a vertex list  $p = (p_1, p_2, \dots, p_n)$  (sequence is important). We look at the front side of the polygon if the sequence of points is counter-clockwise when projected on the display (alternative orientation convention fine as well if also used in algorithm). [2]
- To determine whether we look at the front or back of  $p$  from position  $v$ :
  - choose three non collinear points  $p_l, p_k, p_m$  from the list  $p$  with  $l < k < m$  (make sure that  $p_k - p_l \neq C(p_m - p_k)$  for some constant  $C$ ;  $C$  can be computed from the first coordinate and checked for the remaining) [2]
  - Let  $v_1 = p_k - p_l, v_2 = p_m - p_k, n = v_1 \times v_2$  [2]
  - Let  $d = v - p_k$  [1]
  - If  $n^t d < 0$  return front side [1]
  - If  $n^t d > 0$  return back side [1]
  - If  $n^t d = 0$  return neither [1]

Total: [10]

3. (b)

- Let  $P = (p_1, \dots, p_n)$  be the points of the original polygon. [1]
- For each edge  $e = (t_l, t_{(l+1) \bmod 3})$ ,  $l = 0, 1, 2$ , of the triangle: [1]
- For each vertex  $p$  of the polygon  $P$ : [1]
- If  $p$  is outside (to the right of  $e$ ), delete  $p$  from  $P$  [2]
- If the edge from  $p$  to the next vertex crosses  $e$ , add intersection point to  $P$  before the next vertex [2]
- Outside test:  $a = t_{(l+1) \bmod 3} - t_l, b = p - t_l, [-a_y; a_x]^t b < 0$  [2]
- Compute intersection: call line intersection algorithm for lines  $t_l + s_0(t_{(l+1) \bmod 3} - t_l)$  and  $p + s_1(q - p)$  where  $q$  is the next point after  $p$ . If  $0 \leq s_1 \leq 1$ , then the intersection is  $p + s_1(q - p)$ . [1]

Total: [10]

3. (c)

- The intersection consists of at most one polygon. [1]
- If there were two (or more) polygons  $A, B$ , then there are points  $a \in A$  and  $b \in B$  which lie in both original polygons. With this the line between  $a$  and  $b$  also has to lie in both original polygons as they are convex. This line connects  $A$  and  $B$  and hence  $A$  and  $B$  cannot be disconnected components. [4]

Total: [5]

4. (a)

- A BSP-tree (binary space partition tree) is a binary tree where each internal node splits a set of objects into a set in front and a set behind a plane. In general the leaf nodes contain the models which may be cut along the planes; flat objects lying inside the planes can, e.g., be assigned to the “in front” region. [4]
- Construction principle for simple, flat polygons in 3D:
  - (i) choose a polygon and use it to define a partition plane [1]
  - (ii) Split the remaining set of polygons into two sets of polygons in front of and behind the plane. [1]
  - (iii) If a polygon intersects the plane cut it along the plane into multiple polygons and sort them in the same way (or, in particular in the context of ray tracing, store those polygons in both children) [1]
  - (iv) Recursively continue splitting the polygon sets [1]

Total: [8]

4. (b)

- The ray/object intersection tests can be sped up by determining the regions of the BSP-tree that the ray intersects and only consider the objects in these regions: similar to depth sorting we traverse the BSP-tree such that the leafs are visited in the same sequence the directed ray intersects them to find the first intersection point. [3]
- Let the ray be given by  $p + td$  ( $p \in \mathbb{R}^3$ ,  $d \in \mathbb{R}^3$ ,  $t \in \mathbb{R}_0^+$ ). For a BSP-tree node  $C$  with partition plane  $n^t x + s = 0$  ( $n \in \mathbb{R}^3$ ,  $s \in \mathbb{R}$ ,  $x \in \mathbb{R}^3$ ) find the first intersection recursively: [1]
  - If  $C$  is a leaf node, intersect  $p + td$  with each object in  $C$  and return closest intersection point or NULL if none found. [1]
  - Let  $N$  be the child node of  $C$  which contains the starting point  $p$  of the ray depending on the sign of  $n^t p + s$ . Let  $O$  be the other child node. [2]
  - Recursively traverse node  $N$  to obtain potential intersection point  $q$  with objects in the  $N$  area. [1]
  - If  $q$  is NULL, test intersection of the ray with the partition plane: [1]
    - \*  $n^t(p + td) + s = 0$ ,  $tn^t d = -n^t p - s$ : the intersection is not empty if  $n^t p - s = 0$  (ray starts in plane or lies in plane), or  $n^t d \neq 0$  and  $(n^t p + s)/(n^t d) < 0$  (single intersection point). [2]
    - \* If intersection is not empty, recursively traverse node  $O$  to obtain potential intersection in the  $O$  area. [1]

Total: [12]

4. (c)

- Model fog as a density map  $f : \mathbb{R}^3 \rightarrow \mathbb{R}_0^+$ : assign a density value to each point in space. The higher the density, the thicker the fog. [2]
- The main step in rendering a density map is to find how much light is reflected and how much is transmitted by a ray. This is done by integration: move in the given direction in small steps, adding the density of the current position to the total as we move along. If the total is large, the density map is more or less opaque in that direction; if it is small, it is translucent. [3]

Total: [5]