

1. (a) — seen

- 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 $R(d^t n)L$ 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, α the shininess exponent, and r the direction of perfect reflection. [3]
- To compute r : $r + s = 2(s^t n)n$ with $s = d/\|d\|$, so $r = (2(d^t n)n - d)/\|d\|$. [2]

Total: [10]

1. (b) — seen

- 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 as more computations are required and harder to implement than Gouraud shading. [3]

Total: [9]

1. (c) — unseen

- 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 the 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: [6]

2. (a) — seen

- A node of an octree is an axis-aligned cube in 3D. If it is a leaf node, it contains references to the points lying inside this cube. For a non-leaf node, the node's children are formed by partitioning the cube into eight smaller cubes along three planes through the centre of the cube with normals parallel to the coordinate axes. [3]
- A k D-tree partitions space into axis-aligned rectangular blocks. A node partitions space along a plane with normal orthogonal to one of the coordinate axis directions at an arbitrary position. It has two children which represent the point to the left and right of the partition plane. The points are stored in the leaf nodes. At each node a coordinate axis has to be chosen for the partition. [3]
- An octree always partitions space at the centre of a cube and is hence more suitable for uniformly distributed point sets. A k D-Tree partitions space at arbitrary positions (but still axis-aligned) and is hence able to better adapt to less uniform point distributions. [3]

Total: [9]

2. (b) — seen

- Choose a coordinate axis a , either following a regular pattern x, y, z, x, y, z, \dots or a choose the axis in which the distance between the vertices of the polygons is maximal (or some other strategy). [1]
- Choose a position p on the coordinate axis a to partition the points: compute the median coordinate or more efficiently just the middle point between the two extrema on the coordinate axis. [2]
- Split the polygons into two sets, one lying on the right, the other one lying on the left of the partitioning plane at p . Polygons which intersect the plane have to be split into two polygons: compute the two intersection points between the polygon and the plane and cut the polygon along the line segment between these two polygons into two separate polygons. [3]
- Repeat the process from the beginning for each of the two subsets until either a subset consists of less than a certain amount of polygons or a certain maximal depth has been reached. [2]

Total: [8]

2. (c) — unseen

- Starting with the root node of T , let a be the coordinate axis chosen for the root node and let v be the position of the split on a . [1]
- If the a coordinate value of q is smaller than v , select the left child, otherwise the right child (order chosen arbitrarily). [1]
- Recursively process the child in the same manner until a leaf node has been reached. [1]
- Once a leaf node has been reached compute the distances of all the points in the leaf node and choose the point r with the smallest distance. [2]
- Now traverse the tree again in the same manner, but visit all children which may contain a point at most $\|q - r\|$ away from q , i.e. if q lies on the left and $v < q + \|q - r\|$ visit both children and equivalently if q lies on the right. If this finds a closer point, update r and finally report r as the closest point. [3]

Total: [8]

3. (a) — seen

- Simple: edges of polygon do not intersect [1]
- Flat: polygon lies in a plane [1]
- Convex: given *any* two points inside the polygon, the straight line segment between the two points also lies inside the polygon [2]
- Such a polygon can be stored as a list of vertices (or simpler as an $n \times d$ array, etc.) such that the vertices are listed in the order they are visited when traversing the boundary. [2]
- If the ordered vertices are drawn in anti-clockwise order when viewed from the camera position, we look at the front-side, otherwise we look at the backside. [2]

Total: [8]

3. (b) — seen

- Choose three non-collinear points p_l, p_k, p_m such that $l < k < m$ [2]
- Let $n = (p_m - p_k) \times (p_l - p_k)$ (or equivalent) [3]
- Let $d = p - p_k$ (can choose any vertex) [1]
- If $n^t d > 0$, viewer can see the front [1]
- If $n^t d < 0$, viewer can see the back [1]
- If $n^t d = 0$, viewer can see neither [1]

Total: [9]

3. (c) — unseen

- For each line segment $a = w_l, b = w_{(l+1) \bmod n}$ for $l = 0, \dots, n - 1$: [1]
 - Solve the 2D linear equation system $p_1 + t(p_2 - p_1) = a + s(b - a)$ for $t, s \in \mathbb{R}$ [2]
 - If solution (t, s) exists ($p_2 - p_1$ not parallel to $b - a$) and $s \in [0, 1]$ store t in ordered list T of real numbers [2]
- If T is not empty, the clipped line segment consists of the M line segments from point $p_1 + t_l(p_2 - p_1)$ to $p_1 + t_{l+1}(p_2 - p_1)$ where the t_l are taken in order from the list T and $l = 0, 2, 4, \dots, 2(M - 1)$. [3]

(essentially equivalent to polygon clipping, but not on a raster; variations exist: above is Liang-Barsky, Sutherland-Hodgeman with equations ok, too)

Total: [8]

4. (a) seen

- I. Input: viewing position v , look-at point a , up vector u such that viewing plane is centered at a and orthogonal to $a - v$ and u is parallel to y -direction. [1]
- II. For each pixel (x, y) , $x = 0, \dots, X\text{-Resolution}$, $y = 0, \dots, Y\text{-Resolution}$ do
 - (1) Create a ray from the viewing position v in direction d such that it passes through (x, y) in the viewing plane.
 - (2) Set colour of (x, y) to the return value of `raytrace(v,d)`. [2]
- III. function `raytrace(v,d)`:
 - (1) Initialise position t on ray l from v in direction d to infinity and the nearest object n to empty. [1]
 - (2) For each object o in the scene
 - (a) Compute intersection p of l and o closest to v
 - (b) If p exists and it is closer to v than t , set t to p and n to o . [2]
 - (3) If n is empty, return background colour Else [1]
 - (a) If n is reflective and we haven't reached the maximum recursion depth level, compute perfect reflection vector r of d at t and call `raytrace(t, r)` to obtain reflected colour c_r . [1]
 - (b) If n is transparent and we haven't reached the maximum recursion depth level, compute refraction vector r' of d at t and call `raytrace(t, r')` to obtain refracted colour c_t . [1]
 - (c) For each light source $k = 1, \dots, m$ at position l_k , cast ray from t to l_k . If this line segment intersects with any of the other objects, t is in the shadow of this object. Otherwise compute the amount of light c_k reaching t from k . [2]
 - (d) Return combination of colours c_r, c_t and $c_k, k = 1, \dots, m$. [1]

Total: [12]

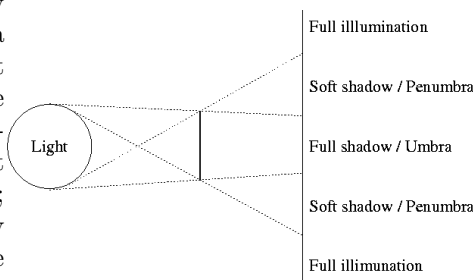
4. (b) seen

- Let $m = (v^t n)n$ [2]
- Let $e = v - m$ [1]
- Then $w = m + e/2 = (v + m)/2 = (v + (v^t n)n)/2$ [3]

Total: [6]

4. (c) unseen

- 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 of the occluding surface; 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.



[4]

- For each extended light source cast multiple rays from the surface point to the extended light source to cover the whole light source surface. [2]
- The percentage of the rays that are not blocked by other surfaces gives the percentage of light reaching the point from the extended light source. [1]

Total: [7]