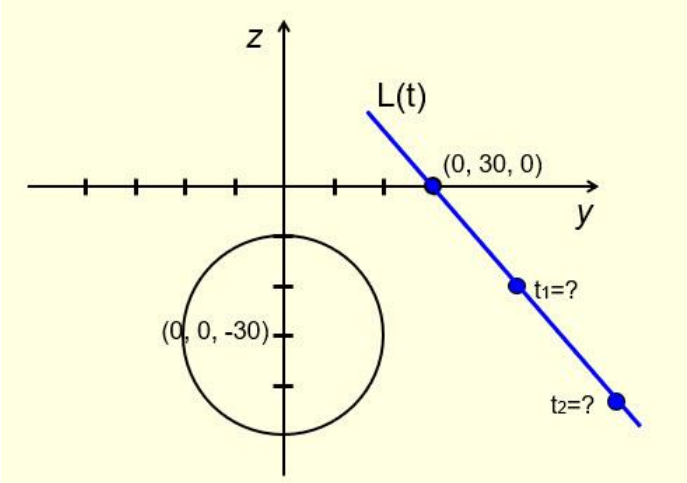
CS535 Computer Graphics
Solution set – Homework Assignment 5 (40 points)

Due: 11/05/2021

1. What are the parameters of the intersection points of the ray *(0, 30, 0) + t(0, 3, -4)* with the sphere centered at the *(0, 0, -30)* with radius *20*? You need to know how to do this to implement the ray tracing algorithm. (5 points)

Sol.

Let t1 and t2 be the parameters of the first and second intersection points of the ray with the sphere, respectively (see the following figure for an illustration of the intersection points).



The implicit equation of the sphere is:

$$x^2 + y^2 + (z + 30)^2 = 400$$

The parametric equation of the ray is:

$$L(t) = (0, 30, 0) + t(0, 3, -4) = (0, 30 + 3t, -4t)$$

where $t \in R$.

By substituting representation of L(t) into equation of the sphere, we get

$$(0)^2 + (30 + 3t)^2 + (-4t + 30)^2 = 400$$

or

$$5t^2 - 12t + 280 = 0$$

By solving this equation for t, we get

$$t_1 = \frac{12 + \sqrt{-5456}}{10} \qquad t_2 = \frac{12 - \sqrt{-5456}}{10}$$

These are not real numbers, so the ray does not intersect the given sphere.

2. What are the barycentric coordinates and ray parameter of the intersection point of the ray *(-40,0,0) + t(1,0,-1)* with the triangle with vertices *(-20,-10,-50)*, *(20,-10,-50)* and *(0,30,-50)* ? You can get the barycentric coordinates and the ray parameter even if the intersection point is outside the triangle. You need to know how to do this to implement the ray tracing algorithm as well. (5 points)

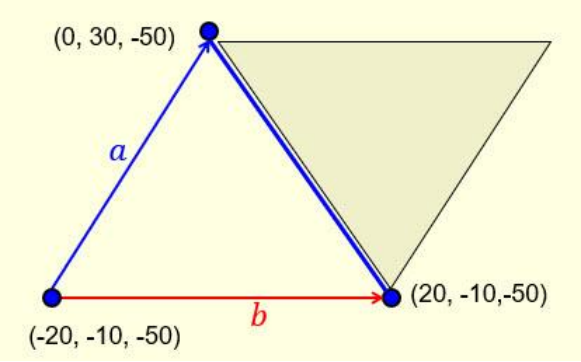
Sol:

Refer to the triangle shown below. As illustrated, we can use a combination of two vectors to represent the intersection point of the ray with the plane that contains the triangle in the following form:

$$\begin{aligned}(-20,-10,-50) + a[(0,30,-50) - (-20,-10,-50)] \\ + b[(20,-10,-50) - (-20,-10,-50)] \\ = (-20,-10,-50) + (20a, 40a, 0) + (40b, 0, 0) \\ = (-20 + 20a + 40b, -10 + 40a, -50) \qquad (*)\end{aligned}$$

where a and b are real numbers.

If the intersection point is a point within the triangle, we must have $0 \le a \le 1$, $0 \le b \le 1$ and $0 \le a + b \le 1$. The last condition is to ensure that we don't get anything from the grey part of the following figure.

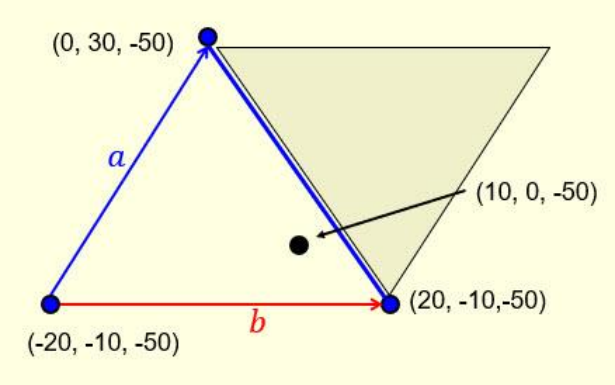


If the ray intersects a point of the plane that contains the triangle, then from equation of the ray $L(t) = (-40 + t, 0, -t)$ and the above representation of the plane (*), we must have

$$\begin{cases} -20 + 20a + 40b = -40 + t \\ -10 + 40a = 0 \\ -50 = -t \end{cases}$$

Solving this system of equations for t, a and b, we get $a = \frac{1}{4}$, $b = \frac{5}{8}$, $t = 50$.

Hence, the intersection point is $(10, 0, -50)$ with parameter t = 50.
Since a and b are both non-negative and the sum of a and b (=7/8) is between zero and 1, so it is a point inside the triangle. See the following figure for the location of (10, 0, -50).



To find the barycentric coordinates of the intersection point, set

(10, 0, -50) = u(-20, -10, -50) + v(20, -10, -50) + w(0, 30, -50),

and solve for u, v and w. We get u=1/8, v=5/8, w=1/4. So barycentric coordinates of the intersection point are (1/8, 5/8, 1/4).

3. What is the reason for using a ***binary tree*** to record the ray tracing process for each pixel of the screen? If none of the objects in the 3D scene is transparent, do we still need a binary tree to record the ray tracing process for each pixel of the screen? (5 points)

Sol:

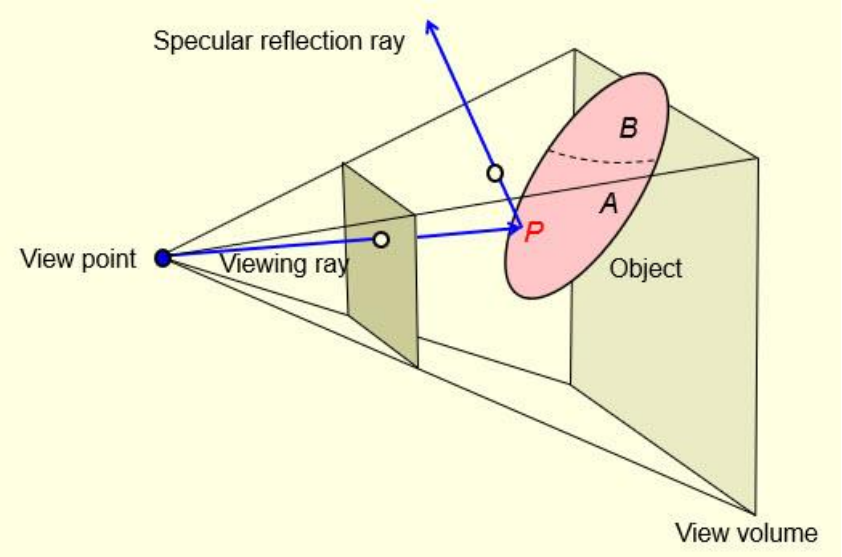
The ray tracing process generates a specular reflection ray and a refraction ray for the first point hit by a viewing ray. The specular reflection ray will return contribution from neighboring objects and the refraction ray will return contribution of light sources behind the object if the object is transparent. A binary search tree **(BST)** is a natural choice for this process because for each node, we can use the left branch to trace the work of the specular reflection ray and use the right branch to trace the work of the refraction ray.

If none of he objects in the 3D is transparent, then a binary tree is not needed, a **linear array** would be sufficient.

4. **Clipping** is not necessary for the ray tracing process? **Why?** (5 points)

Sol:

Even if a portion of an object is outside the view volume (like portion B of the object in the following figure), since each viewing ray generated in the ray tracing process is bounded by the four bounding planes of the view volume, the first intersection point returned by the viewing ray will always be a point on the visible portion of an object (the portion of the object that is inside the view volume), such as the point *P* in the figure below. If the specular reflection ray or refraction ray generated for an intersection point hits a bounding plane of the view volume, we simply return the background color for that specular reflection ray or the refraction ray. So there is no need for a clipping process in the ray tracing algorithm.

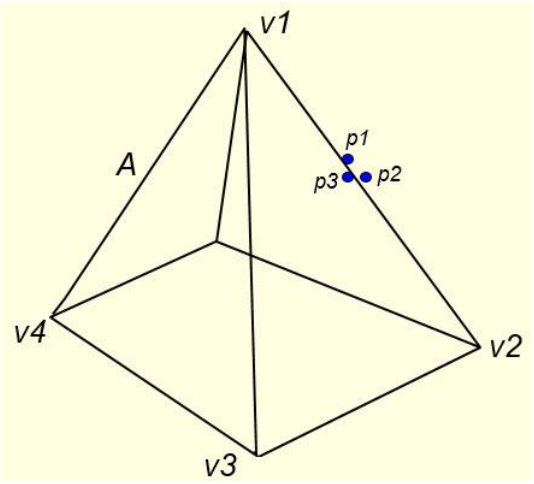


5. The ray tracing technique can be used to compute the volume of an object. Can the ray tracing technique be used to identify the outline of an object with respect to the view point? Justify your answer. (10 points)

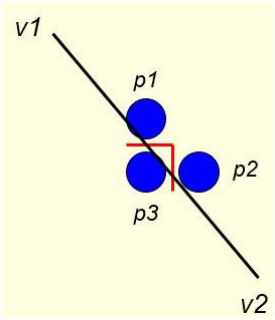
Solution:

The answer is YES.

The idea is, if the rays through two horizontally adjacent pixels intersect different objects, then we assume there is a vertical edge between these two pixels. If the rays through two vertically adjacent pixels intersect different objects, then we assume there is a horizontal edge between these two pixels.



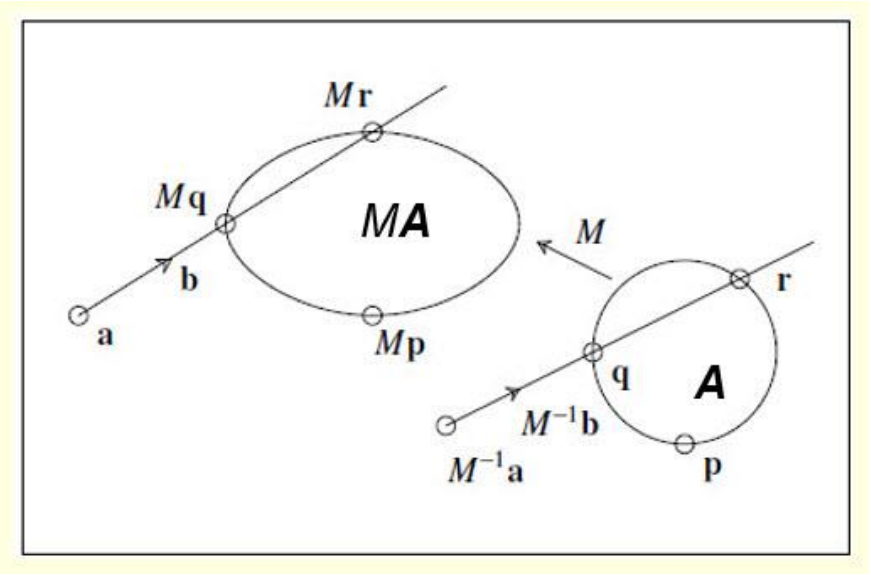
For instance, consider the pyramid A in the above figure. *p2* and *p3* are horizontally adjacent and the rays through these two pixels intersect different objects (one intersects A and one intersects the background), so we assume there is a vertical edge between *p2* and *p3* (see the red vertical line segment between *p2* and *p3* in the following figure). *p1* and *p3* are vertically adjacent and the rays through these two pixels intersect different objects (one intersects A and one intersects the background), so we assume there is a horizontal edge between *p1* and *p3* (see the red horizontal line segment between *p1* and *p3* in the following figure). This way we get an approximation of the edge *v1v2* which is part of the outline of A. Similarly, we can also get approximations for edges *v1v4*, *v2v3* and *v3v4*. For a screen with a reasonably good resolution, we would get a good approximation of the outline of A this way.



6. When ray trace an instance of an object transformed by a matrix M, we usually perform the ray tracing process in the space of the original object. What is the advantage of doing the tracing this way? (10 points)

Solution:

Doing it this way, we only need to maintain one ray-object intersection point computation procedure for each primitive object, instead of performing a separate ray-object intersection point computation for each instance of a primitive object.



For instance, in the above figure, ***A*** is a primitive object and *M**A*** is an instance of ***A*** through the mapping of the transformation matrix *M.* To find the intersection points of the ray *a+tb* with *M**A*** directly, we have to develop separate code to perform the ray-object intersection point computation for *a+tb* and *M**A***. By transforming the ray *a+tb* into the space of ***A***, we can use the ray-object intersection point computation procedure for ***A*** and the ray $M^{-1}a + t\,(M^{-1}b)$, and then transform the intersection points into the space of *M**A*** by *M*. This approach saves both computation time and software implementation.

However, if an object is itself a primitive, such as a unit cube, then the CSG tree representation of this object has one node only, a primitive node for the unit cube. Here we ignore the possibility of representing the object as the union (or, intersection) of two identical unit cubes.