Name: ______________________ **Student ID:** ______________________

1) Describe how to test if a ray with origin $\vec{x}_0$ and direction $\vec{d}$ intersects an infinite cylinder centred on the y-axis with radius 1.

The cylinder is described implicitly by $x^2 + z^2 = 1$. Plug the explicit ray equation $\vec{x}(s) = \vec{x}_0 + s\vec{d}$ in and rearrange:

$$\begin{aligned} (x_0 + sd_x)^2 + (z_0 + sd_z)^2 &= 1 \\ \Leftrightarrow \quad (d_x^2 + d_z^2)s^2 + (x_0 d_x + z_0 d_z)s + (x_0^2 + z_0^2 - 1) &= 0 \\ \Leftrightarrow \quad As^2 + Bs + C &= 0 \end{aligned}$$

If the discriminant $B^2 - 4AC$ is negative, there are no intersections. Otherwise, check if the two real roots

$$s = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

are in range $[s_{\min}, s_{\max}]$ for valid intersection.

2) Which is faster, and why: raytracing or rasterizing a single triangle?

In practice, rasterizing is faster. While implementations differ, there is fundamentally less arithmetic needed to test if a 2D pixel centre is inside a 2D triangle than if a 3D ray intersects a 3D triangle, and similarly there are more efficient data structures to cull away unnecessary tests for 2D rasterization.

3) Explain a problem that can happen with shading a triangle mesh if smoothly interpolated normal vectors are used.

Smoothly interpolated normals are not actually orthogonal to the surface, so geometric formaulas depending on that may go wrong. For example, a reflected ray (for a mirror shader) may go inside the object instead of bouncing off it — see class notes for the picture.

4) What is an effect that pathtracing approximates which regular raytracing (like assignment 3) cannot?

A variety of global illumination effects: any of caustics, color bleeding, indirect illumination.

5) Describe how to incorporate shadows into a matte shader using ray tracing.

Before adding the contribution of a light to the total incident light at the surface point, trace a secondary "shadow" ray to the light source and don't add the light if there are any intersections, i.e. objects blocking the path to the light.

Name: ______________________ **Student ID:** ______________________

6) Why is clipping of some sort necessary for the Z-buffer algorithm when used with perspective projection via 4×4 matrices and homogeneous coordinates?

The projection and homogenization maps vertices behind the camera to positive depths, and vertices between the camera and near clipping plan to negative depths. A point linearly interpolated between them (in the course of Z-buffer rasterization) can erroneously be assigned a depth in the rendered range, showing up in the image when it actually shouldn't be rendered since its true depth is outside of the range.

7) Describe how to test if two points, $\vec{p}$ and $\vec{q}$, are on the same or different sides of the plane containing a triangle with vertices $\vec{x}_0$, $\vec{x}_1$, and $\vec{x}_2$.

Use the signed volume "orientation" predicate:

$$\mathrm{orient}(\vec{a}, \vec{b}, \vec{c}, \vec{d}) = (\vec{b} - \vec{a}) \cdot (\vec{c} - \vec{a}) \wedge (\vec{d} - \vec{a})$$

If $\mathrm{orient}(\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{p})$ *and* $\mathrm{orient}(\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{q})$ *have the same sign, they are on the same side of the plane.*

8) Given n points stored in a BVH of spheres, develop an efficient algorithm for finding the closest point to the origin.

The key is to avoid traversing any part of the tree which cannot contain the closest point. In particular, we only look at a branch in the tree if the bounding sphere, at its closest to the origin, is closer than the smallest distance seen so far.

- Set closest distance so far $d = \infty$ and closest point $\vec{p}$ undefined.
- Push the root on to a stack.
- While the stack is not empty:
- ...Pop node N off the stack.
- ...If N is a leaf node containing point $\vec{q}$, and $\|\vec{q}\| < d$:
- ......Set $d = \|\vec{q}\|$ and $\vec{p} = \vec{q}$.
- ...Else for each child sphere C of N:
- ......Let $\vec{c}$ and r be the centre and radius of C.
- ......If $\|\vec{c}\| - r < d$: *// does C come closer to the origin than d?*
- ....Push C on to the stack.
- Return $\vec{p}$.