1 Ray Tracing [14]
1.1 Complexity [3]
We want to render a scene of $N$ objects with one single light source onto an image of $M$ pixels (the image resolution is $\sqrt{M} \times \sqrt{M}$). Any object can be reflective and refractive, but we enforce a maximum recursion depth of $K$.
What is the worst-case complexity of ray tracing for the total image?

1.2 Light sources [2]
How is this complexity changed when we have $L$ light sources?

1.3 Forward ray tracing [3]
Give a 1-to-2-sentence explanation of why forward ray tracing is not directly practical.
1.4 What is the bug that caused the artifacts in this image? [ /3]

1.5 Which sphere has the bigger index of refraction? [ /3]
2 Transformations /6

2.1 Matrix form /3
What is the $4 \times 4$ matrix in homogeneous coordinate form corresponding to a 3D translation by $(a, b, c)$?

2.2 Normal transform /3
In a ray tracer, when an object is transformed by a linear transformation described by matrix $M$, how must we transform the surface normal after ray intersection? Give both a one- or two-sentence explanation and a formula.

3 Local shading /8

3.1 Coefficients /3
Describe what the image will look like if the scene contains no real light source but the ambient light color is $(1,1,1)$.

3.2 Give a one-sentence description of the Fresnel effect. /3

3.3 Dimensionality of BRDFs /2
How many dimensions does an anisotropic BRDF have?
How many dimensions does an isotropic BRDF have?
4 Ray-Cylinder Intersection [22]

4.1 Implicit cylinder [2]
Give the implicit equation for an infinite cylinder centered on the z axis and with radius r.

4.2 Explicit ray [2]
Give the explicit (parametric) equation for a ray with origin R and direction D.

4.3 Ray-cylinder intersection equation [4]
Write the quadratic equation for the intersection of a ray with an infinite cylinder centered on the z axis. Solve your equation for t.
4.4 Ray-cylinder intersection pseudo code [8]

Using the result from the previous question, write the pseudocode for the intersection method:

```cpp
bool Cylinder::intersect(const Ray &r, Hit &h, float tmin);
```

Don’t forget to compute the surface normal, but don’t worry about the material.
4.5 General cylinder [ /3]
In your ray tracer, without writing additional code, how would you use the code from question 4.4 to render arbitrary infinite cylinders (arbitrary position and arbitrary orientation)?

4.6 Non-infinite cylinders [ /3]
How would you modify the code from question 4.4 to render non-infinite cylinders. That is, the cylinder only goes from \( z_1 \) to \( z_2 \). You do not need to render the caps of the cylinder. You can assume that \( z_2 \geq z_1 \).