1. There is a scene with one black triangle in it. When this scene is ray traced with a standard ray tracer, this is the result we get:

Describe what changes can be made to the raytracing algorithm in order to smooth out the jagged line.

2. Define the difference between local and global illumination.

3. If we generate a raytree that looks like this:

   a. Which surfaces are reflective?
   b. Which surfaces are refractive?
   c. Which surfaces are neither reflective nor refractive?
4. You are designing a very simple ray tracer with shadows and reflections (no refractions). It should render scenes with a single color channel (grayscale images). The critical part of this program is a function called `trace` that takes a `ray` and recursion `depth` as arguments and returns a pixel intensity. The `trace` function should call itself recursively where appropriate to accomplish the final task. In a top-level loop in the bigger program, `trace` would be called for a `ray` emanating from each pixel in the image with a recursion `depth` value of 0.

**Write a pseudo-code definition for `trace(ray, depth)`**.

Follow a consistent syntax and use any of the definitions provided below. If your code needs to make use of any other functions or global values, give a one line description of what you are accessing.

**Helpful Assumptions:**
- All lights have unit intensity (1) and do not fall off with distance.
- Calling `trace` with a ray that does not intersect any object should result in a background color.
- Every point on every object has half diffuse reflectance and half perfect-mirror reflectance.
- No object emits light on its own.
- 0 represents Black

**Global values:**
- `MAX_DEPTH` – maximum recursion depth
- `OBJECTS` – collection of objects in the scene
- `LIGHTS` – list of lights in scene
- `BACKGROUND_INTENSITY` – some background color (single channel)

**Utility functions:**
- `SomeObject | None = closest_object(Ray, OBJECTS)`
- `True | False = can_see(Ray, Light)`
- `OutgoingRay = reflect(IncomingRay, Object)`
- `Intensity = diffuse(Object, Ray, Light)`