Ray Tracer – Basic Concepts

Basic methods and description:

1. Pixel[][] renderImage(cameraInfo, objectsArray, lights)
    - build a 2d pixel matrix (2d array of pixel objects)
    - for i – 0 to n:
        - for j – 0 to n:
            - Convert the current matrix cell position to its location in the cartesian coordinates system
            - Create a ray which starts at the camera location and goes through the cell location (the converted location)
            - Run getPixelColor(cameraInfo, objectsArray, lights) and save the rgb value for that pixel
            - Insert the computed rgb value to the relevant cell of the matrix
    - **Return the updated image**

2. Double[] getPixelColor(cameraInfo, objectsArray, lights,ray, reflectionDepth, tranparencyDepth)
    - for each object in scene:
        - look for intersection
        - if there is an intersection, check if so far it's the closest one. If so, save it
    - if no intersection was found, return the background color
    - transparency check: if the surface transparency > 0 , run a recursive call to getPixelColor(), with the same parameters except tranparencyDepth which is going to be increased by one. Save this color as **objectBackground**.
    - Lights check: for each light source:
        - Run lightCheck(cameraInfo, objectsArray ,lights[i],ray,intersectionData)
        - accumulate the values: **diffuseColor, specularColor (**for all light sources)
    - Reflection check: if the surface reflectance > 0 , run a recursive call to getPixelColor(), with the same parameters except reflectionDepth which is going to be increased by one. Multiply the output of getPixelColor() by reflection color of the surface. Save this color as **reflectionColor**
    - Final calculation:
        - **finalColor** = (**objectBackground**) * (transparency) + (**diffuseColor** + **specularColor**) * (1 - transparency) + (**reflectionColor**)
    - **return finalColor**
    -

3. double lightCheck(cameraInfo, objectsArray ,lights[i],ray,intersectionData)
    - check if ray from light[i] is obscured or not
        - if the ray is not obscured:
            - using the light color – calculate the **diffuse color** of the intersection point
            - calculate the angle between the intersection point's normal and the light ray. This angle, the specular intensity and the specular color of the surface will help us calculate the **specular color** at the intersction point

    - if the ray is obscured:
        - calculate both color the same way
        - multiply both color by (1-shadow intensity).
- Return **diffuse color**, **specular color**