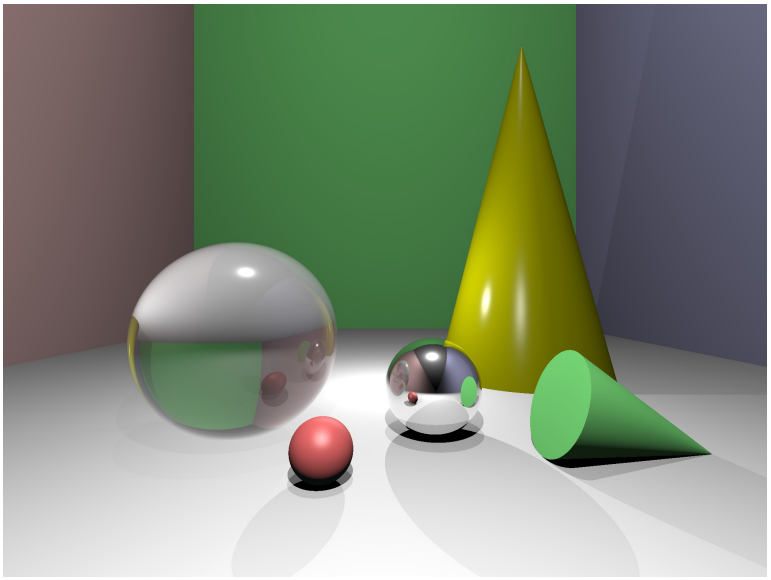
Università
della
Svizzera
italiana

Faculty
of Informatics

Computer Graphics

Assignment 4: Shadows, Reflection, and Refraction



In this assignment, you are challenged to implement the remaining main functionalities of every basic raytracer, i.e., shadows, reflections, and refraction. The image above shows what you can expect after correctly solving all the exercises. The exact result, as usual, will depend on the precise settings of material parameters, illumination, and tone mapping. As always, your solution is not expected to provide the same colors. However, you should obtain very similar reflection and refraction effects when using the correct object definitions. Remember to take care of the numerical issues when computing secondary rays for all exercises.
Start solving the assignment using your solution to the previous assignments. To create an image similar to the above, you should reuse the definitions of the objects of earlier assignments. In this assignment, you will be asked to add only one new object, the refractive sphere.

Useful functionality of the GLM library

`glm::reflect` and `glm::refract` are your potential good friends for this assignment. Make sure that you check how they work before using them.

Exercise 1 [4 points]

Add to your raytracer shadows. Probably the most convenient way of doing it is to include shadow computation into the function `PhongModel`. Simply make sure that you add a contribution of a particular light source to the color of a point only if the light source illuminates the point. If it does not, discard this light source from the light computation.

Exercise 2 [4 points]

Make your raytracer handle objects made of reflective materials. To this end, you will first need to extend the `Material` structure to keep the information about the reflectivity. The simplest solution is to store a boolean

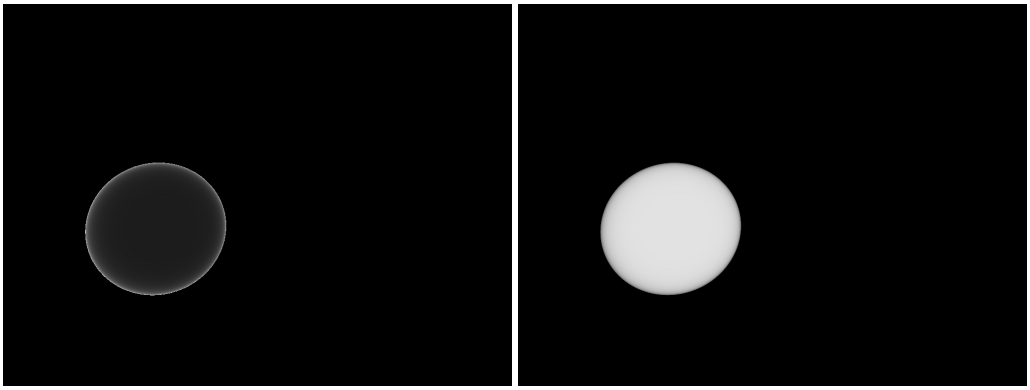
flag indicating whether the material reflects the light or not. For more refined control over the material properties, you can store a real number that indicates the portion of the light undergoing perfect reflection. The rest can come directly from the Phong model of the material. To demonstrate the implemented capabilities, make the sphere in the middle (originally blue; see the image above) reflect the light. Do not change the objects' size and positions so you can easily verify whether the reflection matches the image above.

Exercise 3 [5 points]

Implement the refraction effect. Similar to the previous exercise, you need to make sure that the `Material` structure can keep the information whether the object refracts the light or not. Additionally, you have to store the refractive index of the material. When implementing the refraction, be careful about the proper handling of ray intersections coming both from outside and inside of the object. Note that if you consider a glass object, sometimes the rays will be going from air to glass and, in other cases, from glass to air. You have to detect these cases as the refraction computation is different in both cases. One possibility is to keep at each execution of the `trace_ray` function a flag whether you are inside or outside of the object. The other possibility is to determine whether you go into or out of the object based on the angle between the normal vector and the ray direction at the intersection point. To get the full amount of points, it is enough if you get the effect right for a sphere made of solid material. To demonstrate the effect, add a refractive sphere to the scene. The center of the sphere should be at $(-3, -1, 8)$, and the radius 2. The index of refraction for the material of the sphere should be 2.0.

Exercise 4 [2 points]

Extend handling of the refraction by considering the Fresnel effect. In case you run into problems and need to debug your code, it might be useful to visualize for the transparent sphere the two coefficients for the Fresnel effect, i.e., the portion of the light reflected and the portion of the light refracted. Below, you can find two images that visualize them for our reference raytracer: reflection coefficient on the left and refraction on the right. Note that to visualize the coefficients, you need to disable the tone mapping and the gamma correction. The coefficients should be in the range $(0, 1)$. The correct behavior is that more light should undergo reflection as the primary rays approach the sphere's boundary.



Submission

Your submission **must** contain one ZIP-file with:

- a readme file or a PDF document file with information about which exercises you solved, the authors of the solutions, and the explanation of encountered problems, if any,
- an image file, *result.ppm*, containing the final image you could render,
- a directory named *code* containing all the source code used to generate the image.

The source code, upon compilation, should generate the image identical to the submitted *result.ppm* file. Your code should compile by calling *g++ main.cpp*. The ZIP file name must be of a form *surname1_surname2.zip* for

a team of two, and *surname.zip* for a single submission. Only one person from the team should submit the solution.

Solutions must be submitted via iCorsi by the indicated there deadline.