

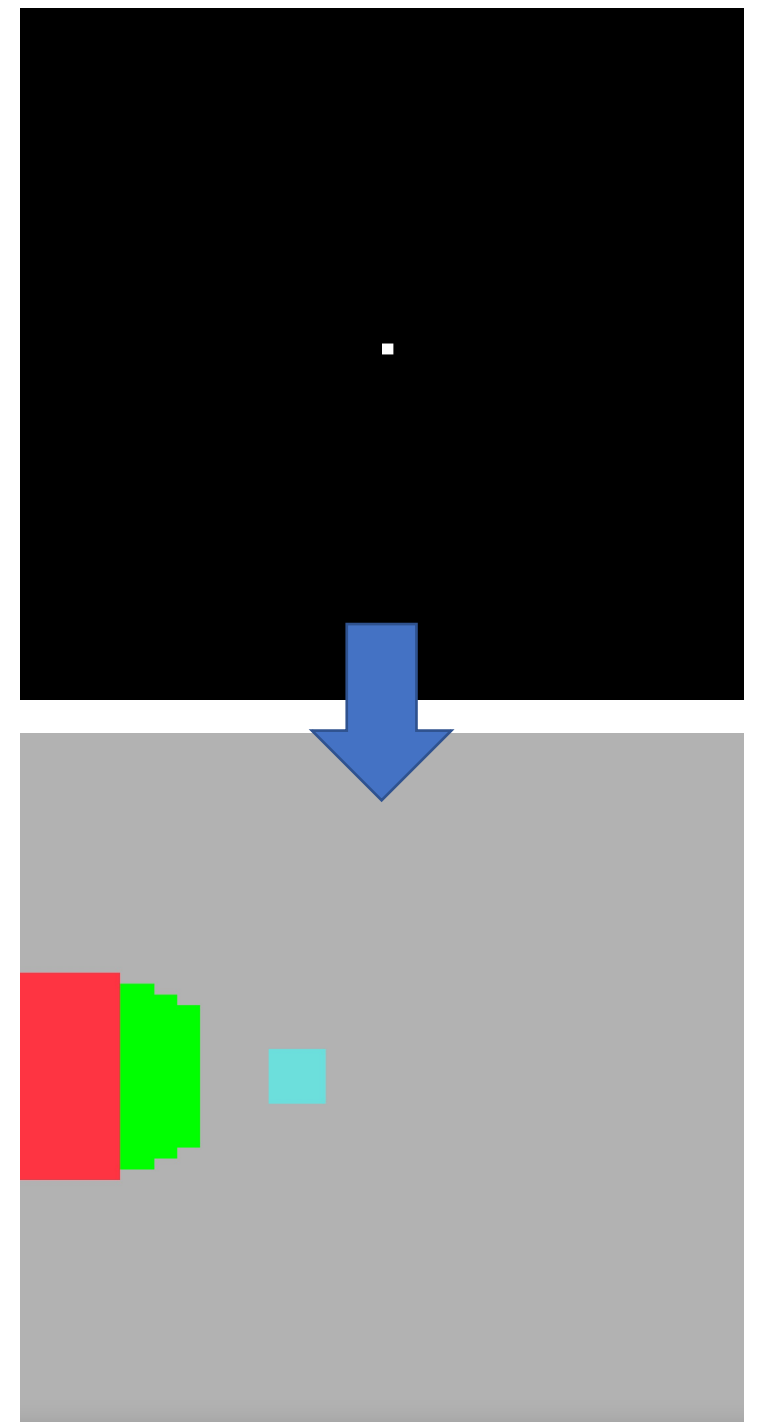
ITU - Graphics Programming

Exercise 11 – ray-tracing

Henrique Debarba

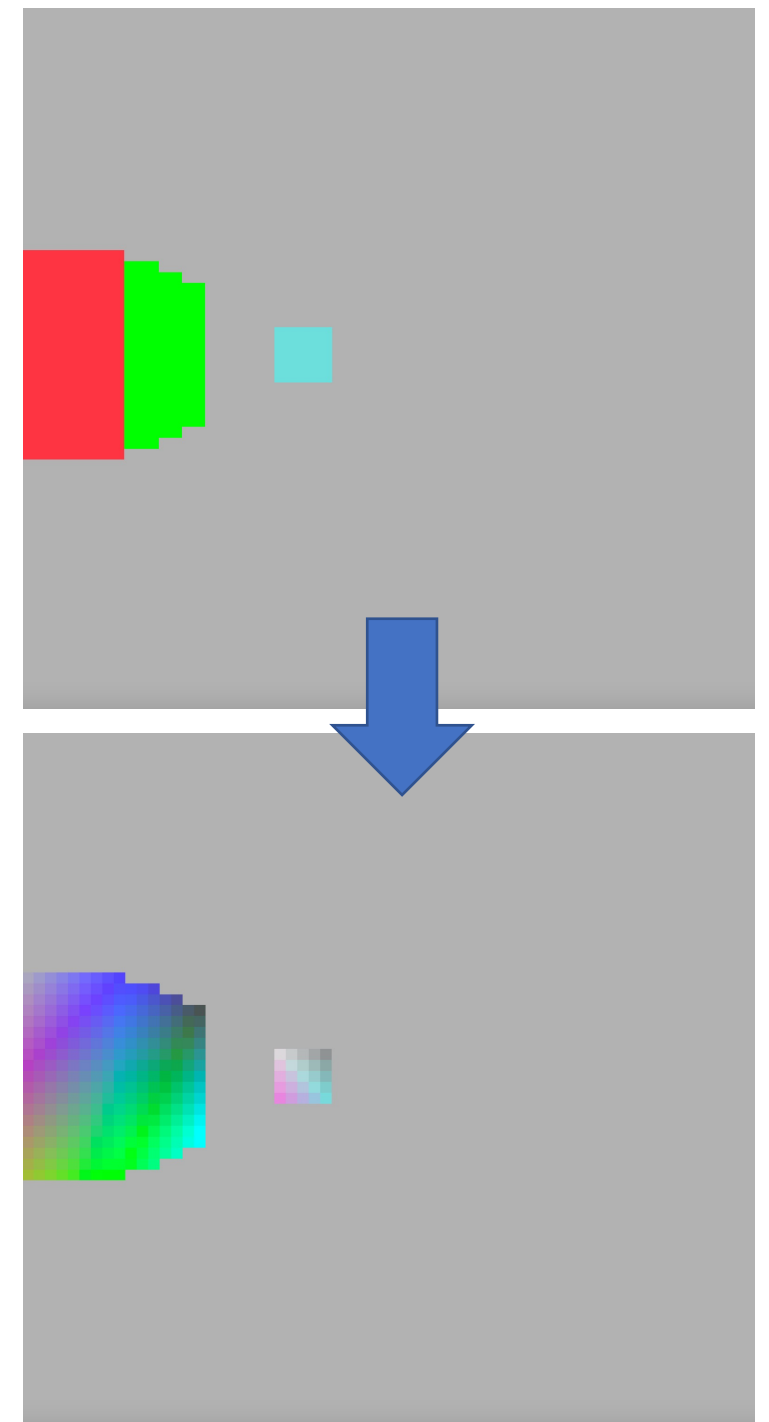
11.1 casting primary rays

- The application starts with a black screen and a single white pixel.
- Examine the methods in the `rt_renderer.h` file.
- Create the primary rays
 - Primary rays are the rays that are sent out from the projection convergence point (the camera position in our case)
 - Follow instructions in the “render” method in the `rt_renderer.h` file
- Notice that the small blue square in the resulting image is a reflection. It is hard to realize that if we don't have lighting!



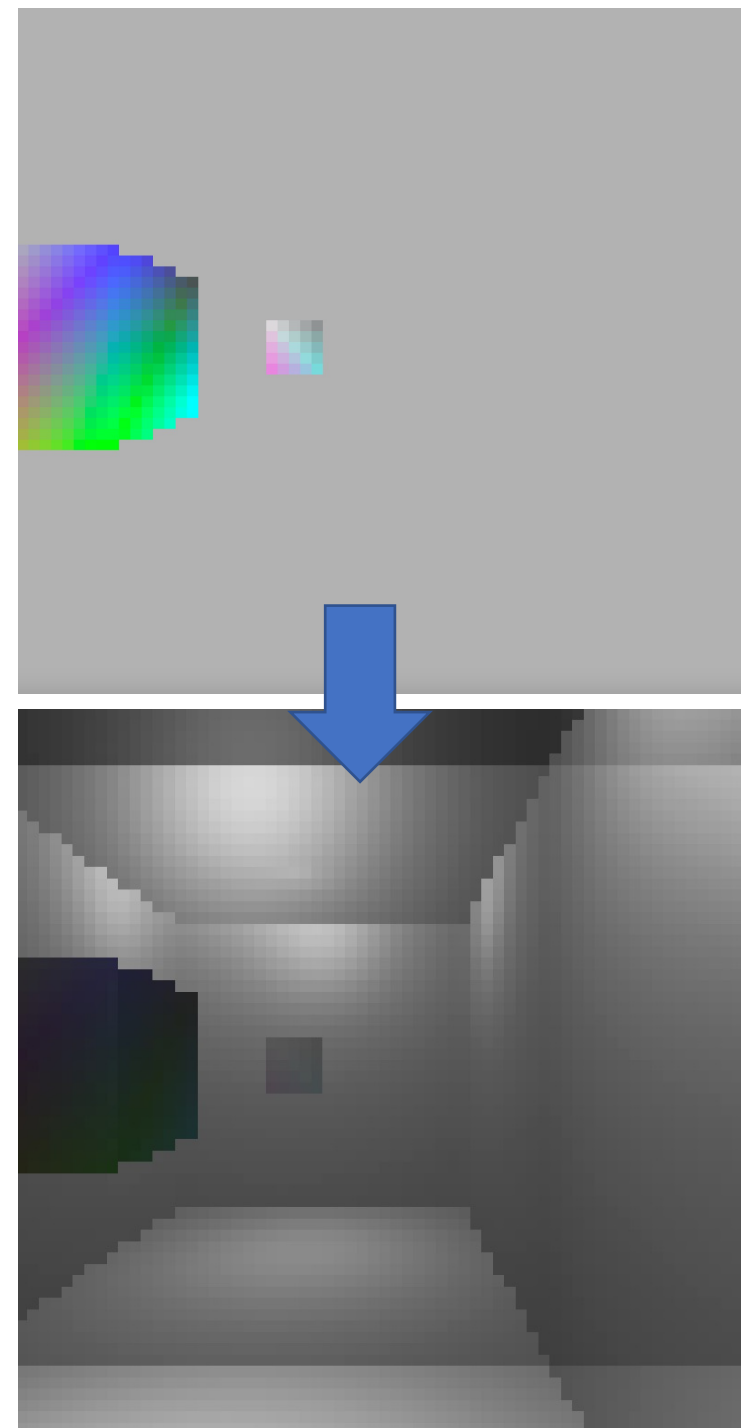
11.2 interpolation

- The RayModelIntersection function returns the barycentric coordinates of the triangle that was hit (if any).
- You should use these coordinates to interpolate colors and the normal of the current pixel.
- To access the vertices of the current triangle, you can use
 - `vts[hitInfo.hit_ID]`,
 - `vts[hitInfo.hit_ID+1]` and
 - `vts[hitInfo.hit_ID+2]`
- Remember that the normal vector should be normalized!



11.3 lighting

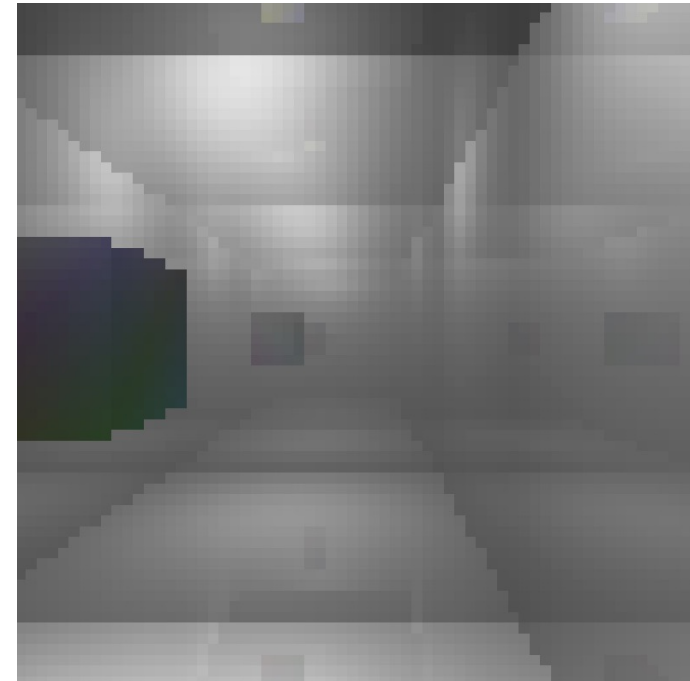
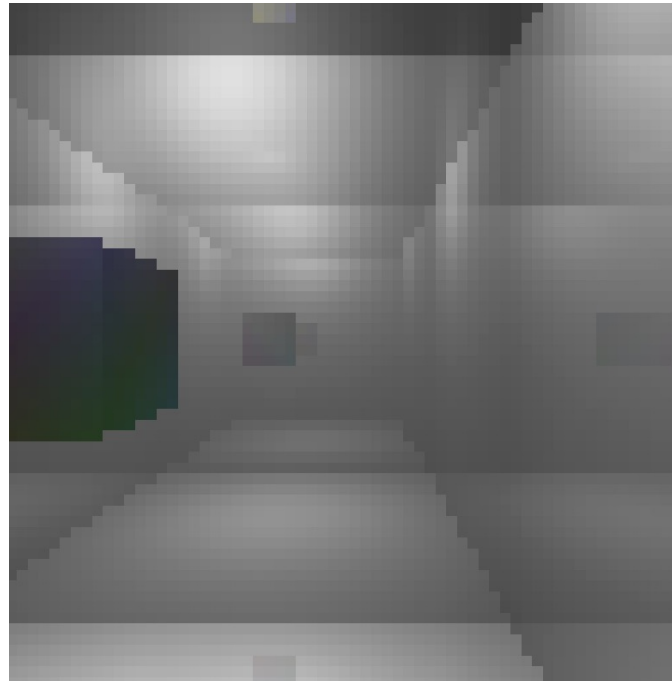
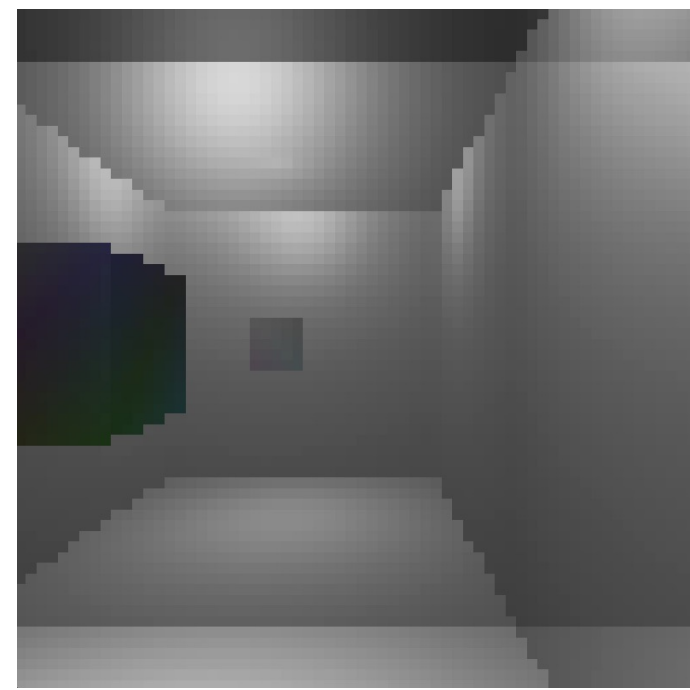
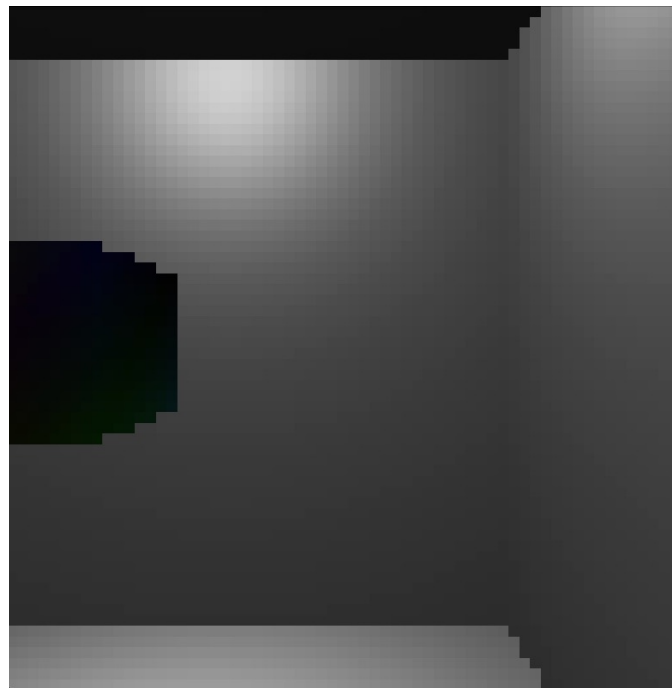
- Implement the Phong reflection model
- Replace `col = i_col;` with the light reflection that you have computed.



Final result

Key mapping:

- 1 - one intersection (aka ray-casting rendering)
- 2 - one reflection
- 3 - two reflections
- 4 - three reflections
- 5 - four reflections (not shown in the image)



11.4 shadows (optional)

- Check for light is occluded by any geometry before (this should approximately double the computation cost of your ray tracer!)
- (no screenshots for that, sorry)