

RAYTRACER

COMS W4160

Jason Scherer: jps19@columbia.edu
Aurélien Sérandour: as3538@columbia.edu

1 list of commands

1.1 General commands

- *size width height*: size of the output image in pixels
- *maxdepth depth*: number of allowed bounces of lights (maximum depth or recursive raytracing)
- *output filename*: name of the image file (a ppm file)

1.2 Camera

- *camera lookfromx lookfromy lookfromz lookatx lookaty lookatz upx upy upz fov*: camera parameters as defined in OpenGL. *fov* is in radians.

1.3 Lighting

- *directional x y z r g b*: create a directional light whose direction is $(x\ y\ z)$ and color is $(r\ g\ b)$.
- *point x y z r g b*: create a point light whose location is $(x\ y\ z)$ and color is $(r\ g\ b)$.
- *omni x y z r g b radius*: create a sphere of light whose location is $(x\ y\ z)$, radius *radius* and color is $(r\ g\ b)$
- *attenuation const linear quadratic*: compute the attenuation caused by the distance d to the light-source by dividing the intensity calculated by $const + linear.d + quadratic.d^2$.
- *ambient r g b*: set the ambient illumination of the scene at $(r\ g\ b)$.
- *softShadowsRadial p*: number of radial subdivisions of omni lights.
- *softShadowsRotate q*: number of rotate subdivisions of omni lights.

Note that the colors $r\ g\ b$ are float between 0 and 1 and not integer between 0 and 255.

1.4 Geometry

- *sphere x y z radius*: create a sphere whose center is $(x\ y\ z)$ and radius *radius*
- *maxverts number*: set the maximum number of vertices to come - **mandatory before *vertex* commands**
- *maxvertnorms number*: : set the maximum number of vertices associated with a normal to come - **mandatory before *vertexnormal* commands**
- *vertex x y z*: create a vertex whose position is $(x\ y\ z)$
- *vertexnormal x y z nx ny nz* : create a vertex whose position is $(x\ y\ z)$ and its normal's direction is $(nx\ ny\ nz)$.
- *tri v1 v2 v3*: create a triangle with the three vertices represented by their indices.
- *trinormal v1 v2 v3*: create a triangle with defined normals with the three vertices represented by their indices.

1.5 Transformation

- *translate x y z*: translate along $(x\ y\ z)$.
- *rotate x y z angle*: rotate around $(x\ y\ z)$ with an angle *angle* (in degrees)
- *scale x y z*: scale by a factor x in x-direction, y in y-direction and z in z-direction.
- *pushTransform*: store the current transformation matrix.
- *popTransform*: retrieve the last stored transformation matrix.

1.6 Materials

- *ambient r g b*: set material ambient color to $(r\ g\ b)$.
- *diffuse r g b*: set diffuse color to $(r\ g\ b)$.
- *specular r g b*: set specular color to $(r\ g\ b)$.
- *shininess s*: set shininess exponent to s .
- *emission r g b*: set emissive color to $(r\ g\ b)$.
- *reflectivity k*: portion of color coming from reflected light.
- *refractivity k*: portion of color coming from refracted light.
- *refractiveindex n*: refractive index of the material.

The color at each point is $C = K_a + K_e + \sum_{i=1}^n S_i L_i (mK_a + K_d \max(l_i \cdot n_i, 0) + K_s \max(l_i \cdot n_i, 0)^{\text{shininess}})$

1.7 Post-processing

- *supersampling n*: size of the antialiasing mask.
- *jittering*: method of antialiasing.
- *random*: method of antialiasing.
- *rotated*: method of antialiasing - force supersampling to be 2.

2 Soft shadows

Soft shadows require the use of an *omni* light. Instead of performing one visibility test between the scene point and the light, many tests are performed to estimate the fraction of the light viewed by this point (see Fig.1). The sphere of light is considered to be a disk facing the scene point. The disk is sampled in several areas (see Fig.2). The color computed is multiplied by the portion of the light viewed by the scene point.

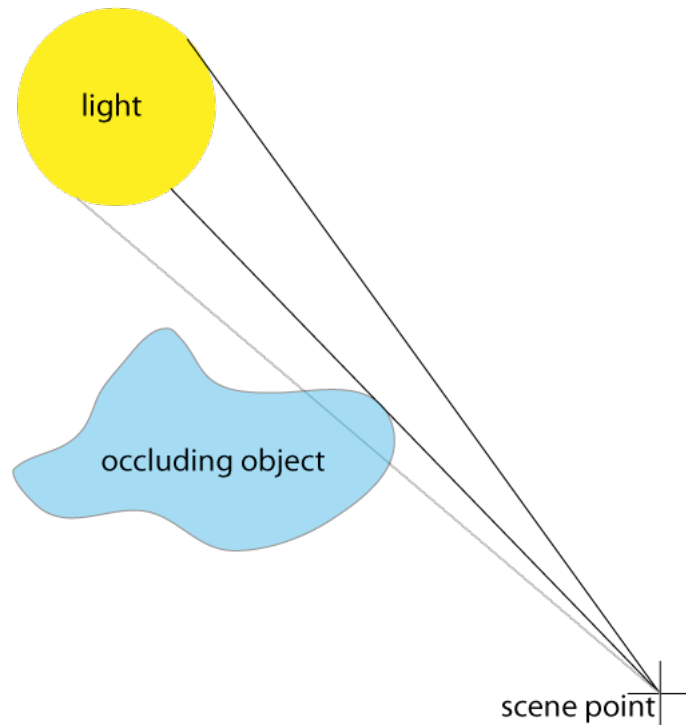


Figure 1: multiple ray test

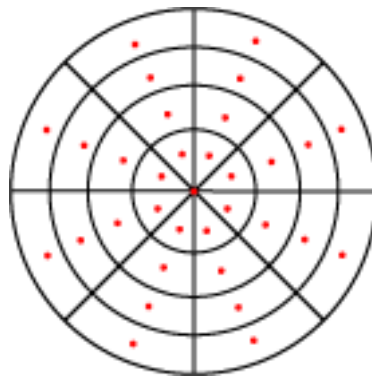


Figure 2: disk samples for $\text{softShadowsRadius} = 4$ and $\text{softShadowsRotate} = 8$

3 Supersampling - Antialiasing¹

We added antialiasing to our project. For a supersampling value n , the method is to cast n^2 rays for each pixel and compute the average of them. Practically, this means multiplying the size of the image by n and grouping pixels from a $n \times n$ mask to create a unique one. There are several methods to cast these n^2 rays

¹images from Wikipedia

3.1 Grid - by default

Each pixel is subdivided in a $n \times n$ grid and each ray goes through the center of each sub-pixel.

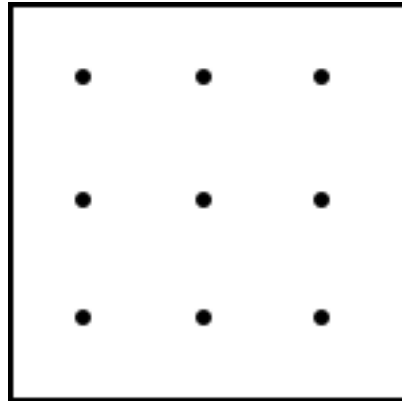


Figure 3: 3×3 grid

3.2 Random choice - command *random*

Instead of using a grid, rays can go randomly through any point of the pixel. With this method, you can choose exactly the number of rays to cast because there is no bound to a regular grid. However, in our code, we keep $n \times n$ rays.

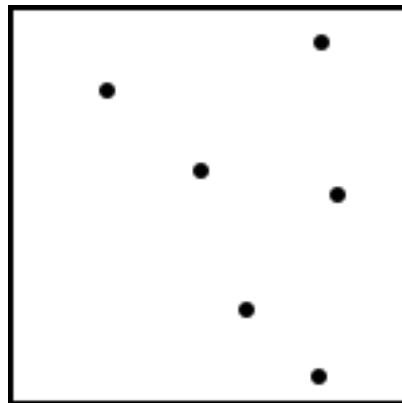


Figure 4: random choice of 6 rays in the pixel

3.3 Jittering - command *jittering*

In the jittering method, rays are chosen randomly within each sub-pixel. This is a combination of the two previous algorithms.

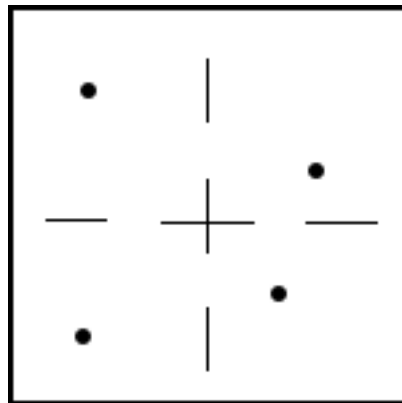


Figure 5: Jittering for a 2×2 grid

3.4 Rotate the grid mask - command *rotated*

In order to avoid aliasing on nearly horizontal or vertical lines, we can rotate the grid mask. We choose to rotate it of 15 degrees. because of boundaries issues, we limited the supersampling value to 2 with this method.

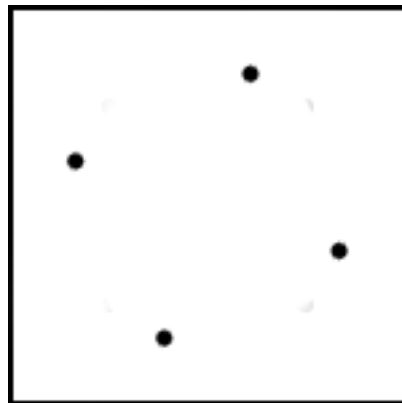


Figure 6: Rotated 2×2 grid

4 Refractivity

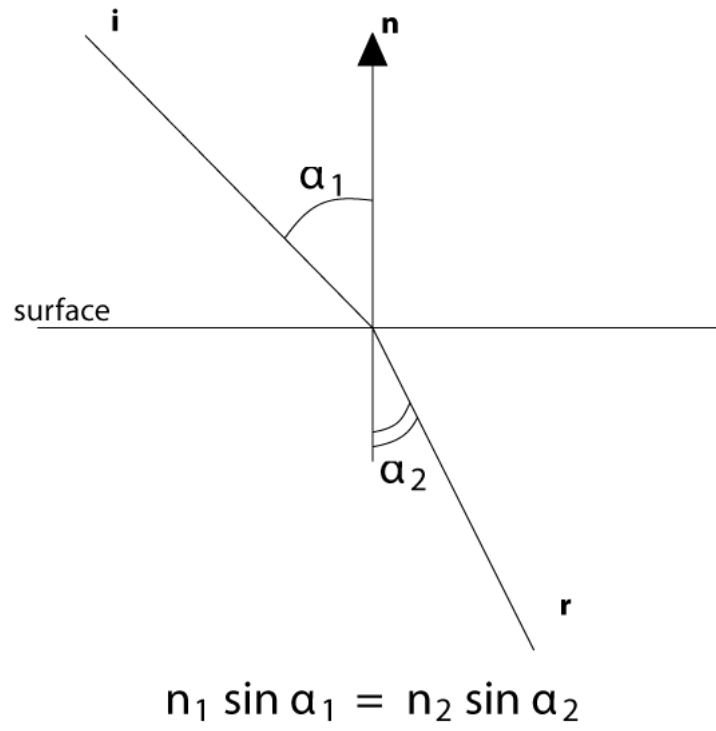


Figure 7: Fresnel refraction