CPS5124
Computer Graphics
Assignment: Image-order Rendering

Keith Bugeja
Sandro Spina

Department of Computer Science
University of Malta

Academic Year 2020/2021

Instructions

- This is **an individual** assignment. This assignment carries **100**% of the final **CPS5124** grade.
- A soft-copy of the report and all related files (including code) must be uploaded to the VLE by the indicated deadline. All files must be archived into a single .zip file. It is the student's responsibility to ensure that the uploaded zip file and all contents are valid. Please note that the standard plagiarism form must be signed and enclosed with the uploaded archive.
- Reports (and code) that are difficult to follow due to low quality in the writing-style, organisation or presentation will be penalised.

General

This is the assignment for CPS5124 (High Performance Graphics) for academic year 2020/21. In this assignment you are required to implement a number of image-order synthesis algorithms from first principles. Where the implementation of multiple strategies is required, make sure output is generated for each individual variant – **Task 2**, tone mapping: (i) sigmoid, (ii) linear; **Task 3**, image estimator: (i) uniform random, (ii) stratified.

Task 1

Image-order approaches are varied, but in general they may be classified as forward (starting from the light), backward (starting from the eye) or bidirectional. Key in point-based image-order estimators is the ray-casting function, a method for point-sampling scene geometry. You are required to build and test two-variants of this function:

`intersects` give a ray segment R and the set of scene geometry, return the point of intersection along R closest to its origin, if any;

`isOccluded` determine whether a ray segment R intersects any scene geometry.

Note that R is typically represented by the quadruple $\langle O, \hat{D}, t_{min}, t_{max} \rangle$, where O is the ray origin, $\hat{D}$ is the normalised direction, and t_{min} and t_{max} are the segment extrema. Only intersections bounded by the segment extrema are considered valid. Assume that scene geometry will be entirely comprised of parametric spheres, and for a sphere $S = (C, r)$, a point P lies on the surface of S if $(P_x - C_x)^2 + (P_y - C_y)^2 + (P_z - C_z)^2 - r^2 = 0$.

Furthermore, implement a pinhole camera system and a matching ray generation scheme. Develop an image-order renderer that stores the distance of each primary ray intersection into a frame buffer. If a ray does not intersect any geometry, record a finite maximum value instead. Finally, implement a post-processing operator to transform the scalar intensity value $I(x)$ of each pixel x, recording the new value $I'(x)$ in the frame buffer instead:

$$I'(x) = 1 - \frac{I(x)}{1 + I(x)}$$

Marks: 20

Task 2

In Heckbert's notation, the ray grammar for Whitted-style ray tracing [3] is given as $LD(S^*)E$; develop a recursive ray tracer that implements the grammar above. In particular, L events should be modelled by point sources, S events by perfectly specular and transmissive surfaces (delta-distribution BSDFs), and D by Lambertian diffusors; model E-type events using a pinhole camera system. The intensity for each pixel should be computed as a weighted integral over the image plane, such that:

$$I_{pixel} = \int_P I(x \rightarrow eye)\, H(p)\, dp,$$

where x is a visible point seen from the eye through p; $H(p)$ is a box linear filter. Furthermore, generate output using two different tone mapping operators, a sigmoid tone mapper:

$$I'_\lambda(x) = \frac{I_\lambda(x)}{1 + I_\lambda(x)},$$

and a linear tone mapper:

$$I'_\lambda(x) = \frac{I_\lambda(x)}{\max I_\lambda},$$

where I_λ is an individual component of the intensity coefficient vector. In this case assume $I \in \mathbb{R}^3$, where components represent the red, green and blue colour curves respectively. Also assume scene geometry to be entirely comprised of spheres. The scene should be lit by a configurable number of emitters. The recursion depth, although parameterisable, should not exceed 20 bounces; the system will default to an enclosing volume, should a ray-object intersection fail.

Marks: 20

Task 3

Whitted-style recursive ray tracing cannot model complex illumination phenomena such as caustics, soft shadows (from area/volume lights), glossy and translucent materials, and complex inter-reflections. This puts a serious damper on the realism of images generated by the algorithm. Path tracing [2] is an image-order rendering technique based on stochastic methods [1] that models $L(D|S)^*E$ ray paths, which capture many of the complex phenomena observed in the real world. Build a path tracing implementation that includes:

Area Lights Support for soft shadows through the inclusion of spherical area light sources. Make use of point sampling to determine area light coverage for any point in the scene. Given an area light source, the reflected radiance in the direction $x \to x'$ is given by the estimator:

$$\langle L(x \to x') \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{L(y_i \to x)\, f_r(y_i \to x \to x')}{p(y_i)}\, G(y_i, x) \qquad (1)$$

where the geometry term G is defined as:

$$G(x, y) = \frac{(N_x \cdot x \to y)(N_y \cdot y \to x)}{|y - x|^2}\, V(x, y),$$

and the visibility function V:

$$V(x, y) = \begin{cases} 1 & \text{if x and y are mutually visible,} \\ 0 & \text{if x and y are not mutually visible.} \end{cases}$$

f_r is the bidirectional reflectance distribution function (ratio of reflected differential radiance along $x \to x'$ to incident differential irradiance along $y_i \to x$). In the estimator above, $p(y_i)$ is the probability of sampling point y_i on the light source. A naïve approach that uniformly samples the surface of a sphere yields a sampling probability of A^{-1} throughout, for a sphere with area A. This approach is acceptable, although slower to converge.

Glossy and Translucent Materials Provide suitable reflectance functions for glossy and translucent materials, characterised by a roughness parameter $\beta \in [0,1)$ that specifies the magnitude of perturbation of the mirror direction. Like diffuse surfaces, glossy and translucent materials cannot be modelled using a delta reflectance function; instead, an estimator similar to Equation 1 is used. In this case, however, the sampling domain is the distribution of reflected directions due to the surface material at x. Given a surface with normal N and roughness β, and an incident direction ω_i, the mirror direction may be stochastically perturbed as follows:

$$\omega_o = \frac{P(\beta, \xi_1, \xi_2) + R(\omega_i, N)}{|P(\beta, \xi_1, \xi_2) + R(\omega_i, N)|}, \tag{2}$$

where $P(\beta, \xi_1, \xi_2)$ is a perturbation function defined as:

$$P(\beta, \xi_i, \xi_2) = \beta \left(\sqrt{(1 - z^2)} \ (\cos(2\pi\xi_2)\, \hat{\mathbf{x}} + \sin(2\pi\xi_2)\, \hat{\mathbf{y}}) + (1 - 2\xi_1)\, \hat{\mathbf{z}} \right).$$

The perturbation function generates a point on the unit sphere through the random variates ξ_1 and ξ_2, and scales it by β; $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ are unit length coordinate axes. For translucency, the mirror reflection function R in Equation 2 can be replaced by a transmission function T such that:

$$\omega_o = \frac{P(\beta, \xi_1, \xi_2) + T(\omega_i, N, ior)}{|P(\beta, \xi_1, \xi_2) + T(\omega_i, N, ior)|},$$

where ior is the ratio of the refractive indices of the two media at the interface.

Image Estimator Use two sampling strategies in primary ray generation: uniform random and stratified sampling. In particular, uniform random sampling a sub-pixel position x_i in region R on the image plane is accomplished by:

$$x_i = w\xi_1\, \hat{\mathbf{x}} + h\xi_2\, \hat{\mathbf{y}},$$

where $\xi_1, \xi_2 \in [0,1)$ are two uniform random variates, and w and h specify the width and height of R. Sampling a sub-pixel position x_i using a stratified approach:

$$\begin{aligned} e &= \lfloor s^{1/2} \rfloor, \\ x_i &= \frac{w}{e}(i \bmod e + \xi_1)\, \hat{\mathbf{x}} + \frac{h}{e}(\lfloor i/e \rfloor + \xi_2)\, \hat{\mathbf{y}}, \end{aligned}$$

where s is the maximum sample count and i the sample index, $0 \le i < e^2$. The orthogonal unit vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ form part of the orthonormal basis of the camera system and characterise the image plane.

Russian Roulette Your implementation should adopt an iterative paradigm of computation and employ next event estimation (i.e., on each event, generate a light-terminating subpath by sampling a light source, to speed up convergence). Note that you should still provide an option for generating images using naïve path tracing (i.e., single path from eye to light source). Paths generated using next event estimation should be terminated without the introduction of systematic bias, through Russian roulette:

```
...
float q = max(T);
if (random() > q)
```

```
          break;
        T *= 1 / q;
        ...
```

where T is the current path throughput.

Marks: 55

Task 4

Create (and render) an original scene to showcase your path tracing implementation. Impress us!

Marks: 5

Support Files

This document is accompanied by a number of scene definition files in JSON. You can choose to parse these files and build your environments dynamically or hardcode each of the individual configurations in your program. A sample scene definition file is included below for your convenience; each scene definition is comprised of a list of categories (e.g. cameras, materials, lights and shapes) and a scene composition block which refers to instances of objects within these categories. The `id` field is unique within a category (e.g. shapes), but not across different categories, and is used to identify objects within a category registry. The `type` field specifies the factory used to construct the named object. For instance, a camera object constructed by a pinhole camera factory is shown below:

```
{
 "cameras" : [{
    "id"       : "pinhole camera",
    "type"     : "pinhole",
    "fov"      : 45,
    "aspect"   : 1.0,
    "position" : [0, 0, -15],
    "target"   : [0, 0, 0]
  }]
}
```

The object can be uniquely identified as `pinhole camera`. The `position` and `target` fields are used to construct a frame of reference for the camera, implicitly assuming an *up* vector $[0\ 1\ 0]^T$. Material definitions follow a similar pattern: a scattering coefficient (`rho` for reflective materials; `rhoR, rhoT` for materials both reflective and transmissive) controls the fraction of incident light that is scattered. Materials that refract light provide a scalar refractive index, `eta`, defined as the ratio of the speed of light in a vacuum to that in the particular medium (e.g. for glass $\eta \approx 1.5$).

```
{
  "materials" : [{
    "id"   : "chalk",
    "type" : "diffuse",
    "rho"  : [1.0, 1.0, 1.0]
  }, {
    "id"   : "glass",
    "type" : "fresnel dielectric",
    "rhoR" : [0.9, 0.9, 0.9],
    "rhoT" : [0.9, 0.9, 0.9],
    "eta"  : 1.5
  }]
}
```

A *renderable*, or synthesisable, object is composed of two primary components: its material appearance (see above) and its shape. While the material models how light behaves when it interacts with the surface of an object at a microscopic level, its shape determines the geometric composition of the object and thus, the surface itself. The `shapes` category is a list of shape instances that can be used when constructing a scene, as part of a renderable object. Implement the `sphere` shape type, characterised by a `centre` point and a `radius`.

```
{
 "shapes" : [{
    "id" : "floor",
    "type" : "sphere",
    "centre" : [0, -505, 0],
    "radius" : 500
  }, {
    "id" : "ball",
    "type" : "sphere",
    "centre" : [2, -3.5, -1],
    "radius" : 1.5
  }, {
    "id" : "bulb",
    "type" : "sphere",
    "centre" : [2, 3.5, -1],
    "radius" : 1.5
  }],
}
```

All light sources, be they infinitesimal point sources or emissive objects, are declared in the `lights` category. Implement support for `point` and `area` sources. Since the latter are necessarily tied to some renderable geometry, the `shape` field for `area` type light sources links to an entry in the `shapes` category.

```
{
  "lights" : [{
    "id" : "point light",
    "type" : "point",
    "position" : [0, 1.0, -1],
    "power" : [300, 300, 300]
  }, {
    "id" : "area light",
    "type" : "area",
    "shape" : "bulb",
    "power" : [30, 30, 30]
  }]
}
```

The `scene` container combines various elements from the individual category lists together into a single environment, defined by a camera, light sources and rendering primitives. Differentiate between two types of primitives: (i) `geometric` primitives, which are specified in terms of a `shape` and `material` entry from their respective categories, to render a non-emissive object, and (ii) `emissive` primitives, geometric primitives that act as light emitters. These require an additional field, `light`, to define the object's emissive properties. Finally, the `renderer` embodies details such as the type of rendering algorithm, whether Whitted-style recursive ray tracing (`WRT`) or path tracing (`PT`), the size of the frame buffer in pixels (`dimensions`), the number of ray samples traced per pixel (`samples`), the maximum ray depth (`depth`) and the output filename (`output`).

```
{
"scene" : {
    "camera" : "pinhole camera",
    "lights" : ["area light"],
    "primitives" : [{
      "id" : "area light",
      "type" : "emissive",
      "shape" : "bulb",
      "light" : "area light",
      "material" : "chalk"
    }, {
      "id" : "floor",
      "type" : "geometric",
      "shape" : "floor",
      "material" : "chalk"
    }, {
      "id" : "ball",
      "type" : "geometric",
      "shape" : "ball",
      "material" : "glass"
    }],
    "renderer" : {
      "type" : "PT",
      "dimensions" : [512, 512],
      "samples" : 4096,
      "depth" : 6,
      "output" : "../output/assignment_0x.ppm"
    }
  }
}
```

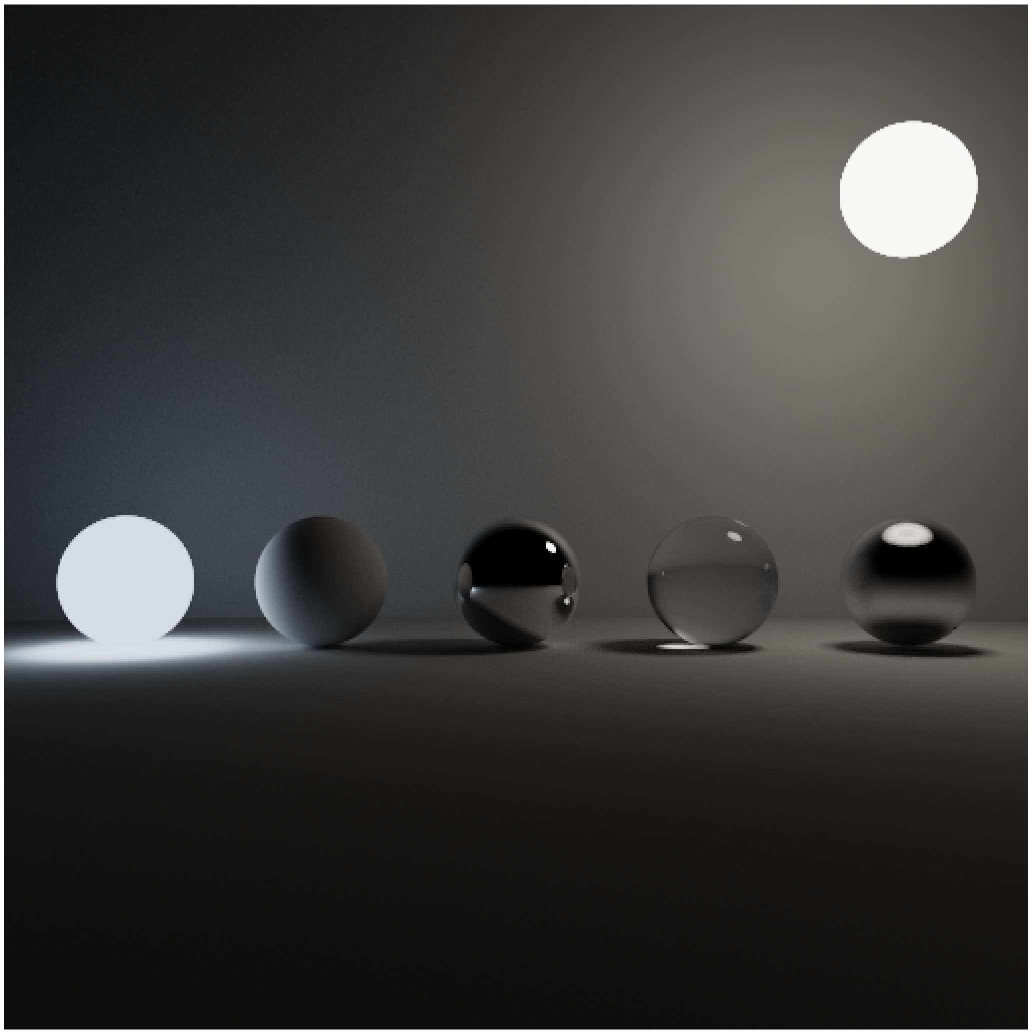

Figure 1: Sample rendering output for scene definition number 8.

Final Words

You are expected to produce a detailed report of your implementation. The report should feature rendering output from your implementation of the provided scene description files. Sample output images (see Figure 1) have also been included with your support files; use them as a reference during development. The images have been tone-mapped using the sigmoid operator described above, and rendered using a *left-handed* coordinate system. Feel free to experiment and try to enjoy this assignment; discuss ideas or difficulties with your class mates. Do not leave the assignment for the last week. The effort put into this assignment will be duly noted during marking. Good luck!

References

[1] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. In *ACM SIGGRAPH Computer Graphics*, volume 18, pages 137–145. ACM, 1984.

[2] J. T. Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.

[3] T. Whitted. An improved illumination model for shaded display. *Communications*, 1980.