Photon Mapping for Photorealistic and Non-Photorealistic Rendering

Parker Greene
pwgreene@mit.edu

I. Introduction

The rendering equation, introduced by Immel et. al. [1] and Kajiya [2], is much better estimated by photon mapping than simple ray tracing; however, unlike ray tracing, photon mapping relies on Monte Carlo sampling techniques, meaning the more photons mapped, the more realistic the rendered image appears. Thus, the method is much more computationally expensive than ray tracing, but with this increase in computation comes the power to render more complex effects, such as indirect illumination and caustics.

Though indirect illumination is one capability of photon mapping, I chose instead to focus on rendering caustics and more specifically, rendering lighting effects through transmitting surfaces. Another feature of photon mapping is that once the photons are mapped, the view camera can move and view the same scene, since the photons, though randomly distributed, do not move when the camera is moved. This view-independence feature was my primary motivation for exploring the possibility of view-independent non-photorealistic (NPR) rendering, since many methods of rendering artistic styles, stippling in particular, traditionally rely on generating random noise, which is difficult to keep consistent across multiple views.

A. Motivation

I was personally motivated to explore the non-photorealistic rendering of these artistic effects because of my background in drawing and painting. I was hoping to explore the more possibilities of imitating brush strokes with photon mapping, but this was a pretty unexplored area in computer graphics research, likely due to its difficulty. I instead spent much more time improving the rendering of realistic effects and pencil effects once I got a working implementation of the photon mapping algorithm.

II. Related Work

Photon Mapping was initially introduced by Jensen and Christensen [3] and my approach in this paper is primarily based on Jensen's algorithm. The photon mapping algorithm has seen a number of improvements, including adding a photon map for shadow photons [4], which I adapted to use as part of my implementation for rendering stippling effects.

Non-photorealistic rendering has been explored by a large number of computer graphics researchers; however, I am unaware of any research papers exploring the possibility of using photon mapping for stippling effects, though there has been plenty of research into rendering objects with this effect [5]–[7]. Li and Wang [8] utilized the photon mapping algorithm to develop a painterly rendering framework, using the the impressionist line integral convolution (ILIC) used by [9].

III. Approach

I built the photon mapping algorithm on top of the ray tracer we implemented in assignment 4 in C++. In order to map photons, I had to implement a photon map data structure, which is a kdTree that stores each mapped photon based on its location in the scene. Photons are structs as defined by listing 1. Each light in the scene creates photons and sends them randomly, according their distribution function, until the number of photons desired have been mapped. I implemented random photon casting for area lights by randomly sampling the area defined by their four corners and a random direction in the hemisphere of their normal. Spotlights sent photons from a single point in a cone defined by an angle parameter.

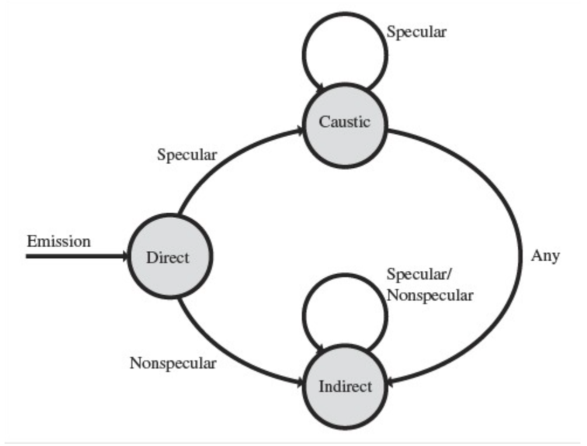


Fig. 1: A finite-state machine for mapping the various types of photons. Courtesy of [11]

```
struct Photon {
    Vector3f start;
    Vector3f position;
    Vector3f direction;
    Vector3f color;
    bool hitDiffuse = false;
    bool hitSpecular = false;
}
```

Listing 1: The photon data structure

My photon mapping implementation uses a Russian Roulette technique such that when a photon interacts with an object in the scene, its behavior depends on that object's material. Each material has some probability of absorbing, reflecting (diffuse or specular), or transmitting the photon. I used this type of photon mapping implementation in favor of one that instead sends multiple (reweighted) photons according to the material's BRDF due to the latter algorithm's branching factor and computational complexity.

I also stored photons in three different photon maps: direct, indirect, and caustic. The direct photons would not be used to compute direct lighting, since methods used in traditional ray tracing can better estimate direct lighting effects. The indirect map was stored photons that had hit at least one non-specular surface. Finally, the caustic map was used to store photons that had only interacted with specular or transmissive surfaces. A visual of the usage of each of these maps is given in figure 1.

I mainly focused on the effects I could achieve by rendering the photons from the caustic maps, so I will proceed to detail my methods for colored caustics and diffraction, and then detail how photon mapping can be applied to NPR.

A. *Colored Caustics*

Caustics can be rendered by calculating direct lighting of the scene through ray tracing techniques, then gathering the nearest k photons from the caustic map. I used a cone filter increase the weight of photons that are closer to the gathering point in order preserve the sharpness of caustics. The cone filter is applied by weighting the contribution of each photon to the radiance estimate by $1 - \frac{1-d_p}{kr}$, where d_p is the distance of the photon to the gathering point, k is the filter factor (can be adjusted depending on the desired effect), and r is the radius of the gathering sphere. Applying this filter helps a lot with reducing blur at the edges of caustics.

Colored caustics can be rendered by updating the color field of each photon as it is transmitted through a colored surface. When a photon passes through a surface, its color vector is multiplied by the color of that object. This allows for photons to adjust their color more than once, as in the case where a photon enters and exits more than one colored transmissive surface.

B. *Diffraction*

Physically-based diffraction can be handled quite nicely by photon mapping. My approach to rendering light cast through a diffractive surface, such as a prism, is as follows: when a photon hits a diffractive surface, split the photon into multiple different colors and refract them at slightly different indices of refraction. For example, a standard white photon sent through a transmissive, non-diffractive glass surface would refract at a refraction index of around 1.52, but when the surface is diffractive, a red photon would refract at 1.52, a yellow photon at 1.53, green photon at 1.54 and so on.

C. Non-Photorealistic Rendering: Stippling

The most experimental part of my project was an implementation of stippling effects using photon mapping. I looked into many different approaches to how stippling effects can be rendered on scenes, but as far as I know, there are no existing methods that use photon mapping.

Rendering stippling using photons is based off the idea of a shadow photon map. Standard photons are sent from each light in the scene along some ray r, and when they first hit an object at point p, ray r continues to be followed, and if there are any intersections of the scene along r after p, then a shadow photon is deposited there. This mapping method is shown in figure 2. The scene is rendered as white at every location by default, but because the shadow photons have a "negative color," they show up as gray or black points. The more negative the color, the darker the spot that shows up on the final image. With a small enough radius, only a couple of photons will be gathered for any given point, producing the dotted effect of stippling.

The advantage of using photon mapping to render stippling over other methods that rely on generating random noise is the ability to move the camera view and preserve the location of the stippled points. Stippling generated with some sort of random noise usually has to recalculate the stippled points, which means there isn't any coherence preserved across multiple views. However, though great at preserving coherence when the camera is moved, the limitation of this method is it can only handle movement of the camera and not movement of the light source or scene objects, due to the fact that photons are randomly sent from the light source onto the scene.

IV. Results

A. Caustics

Figure 3 shows the ability of the photon mapping algorithm implementation to handle colored caustics quite accurately. Figure 4 displays the handling of overlapping transmissive surfaces. The purple-red caustics are rendered by photons sent through both the glass and the wine. I also had to handle substances such as the wine that grew opaque when looking through a thicker part of them. This can be noticed in the difference between colors of the shallow and deeper angles of the liquid.

B. Diffraction

Figure 5 displays the same prism (with a refractive index varying between wavelenghts) at two different angles. Five wavelengths were sampled to yield quite accurate results.

C. Stippling Effects

Figure 6 shows the differences in the choice of lighting for rendering the stippling effect. The area light creates a soft shadowing effect and produces a higher concentration of stippled dots under the sphere and better mimics the effect of stippling.

Since the photons are mapped independently of the camera, multiple views of a stippled scene can be generated without having to remap photons. This video shows a 360 degree view of the sphere from 6b. Frames are rendered in less than one second on the CPU.

Other, more complex geometries can be handled by the stippling effect generation without any modifications, as shown in figure 7.

V. Conclusion and Future Work

Through this project, I learned much more about physically-based rendering techniques and am very satisfied with the realistic-looking scenes that my implementation could render. As for the non-photorealistic stippling effect, I explored a novel approach to achieving it using photon mapping, which allowed for very interesting 3D scenes that could be easily rendered from multiple views without having to recompute the photon map.

As for future work, the photon mapping could be extended to handle subsurface scattering as well as participating media and reflections defined by BDRFs. I am much more interested, however, in exploring the possibilities of other NPR effects using photon mapping, and improving the stippling effect. One improvement that could be done would be to randomly send shadow photons *directly* from the light source, such that angles further from the direction of light would have a higher chance of casting a shadow photon.

VI. Source Code

All source code can be found at https://github.com/pwgreene/Renderer

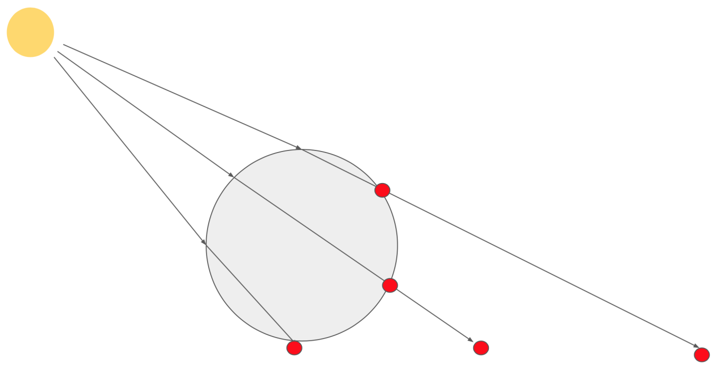


Fig. 2: An example of mapping shadow photons. The light is the yellow circle and shadow photons are mapped at the red points.

References

[1] Immel, David S.; Cohen, Michael F.; Greenberg, Donald P. "A radiosity method for non-diffuse environments" (PDF), Siggraph 1986

[2] Kajiya, James T. "The rendering equation" (PDF), Siggraph 1986

[3] Jensen, Henrik Wann and Niels Jrgen Christensen: "Photon maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects". Computers and Graphics 19 (2), pp. 215-224, 1995.

[4] Jensen, Henrik Wann: "Global Illumination Using Photon Maps". Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering), pages 21?30, 1996.

[5] A. Lu et al., Nonphotorealistic volume rendering using stippling techniques. Proc. Visualization 2002, 81?89

[6] Lu et. al., 2003, "Illustrative Interactive Stipple Rendering". IEEE Transactions on Visualization and Computer Graphics, VOL. 9, NO. 2, APRIL-JUNE 2003.

[7] J. Kruger and R. Westermann. Efficient stipple rendering. In *Proceedings of IADIS Computer Graphics and Visualization*, 2007.

[8] M. T. Li and C. M. Wang. "A 3D Painterly Rendering Framework for Photon Mapping," Journal of Information Science and Engineering. 2010.

[9] C. M. Wang and J. S. Lee, "Using ILIC algorithm for an impressionist effect and stylized virtual environments," Journal of Visual Languages and Computing, Vol. 14, 2003, pp. 255-274.

[10] Jensen, Henrik Wann. "A Practical Guide to Global Illumination Using Photon Maps". SIGGRAPH 2000 Course 8.

[11] Pharr, M., AND Humphreys, G. 2010. Physically Based Rendering, Second Edition: From Theory To Implementation, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

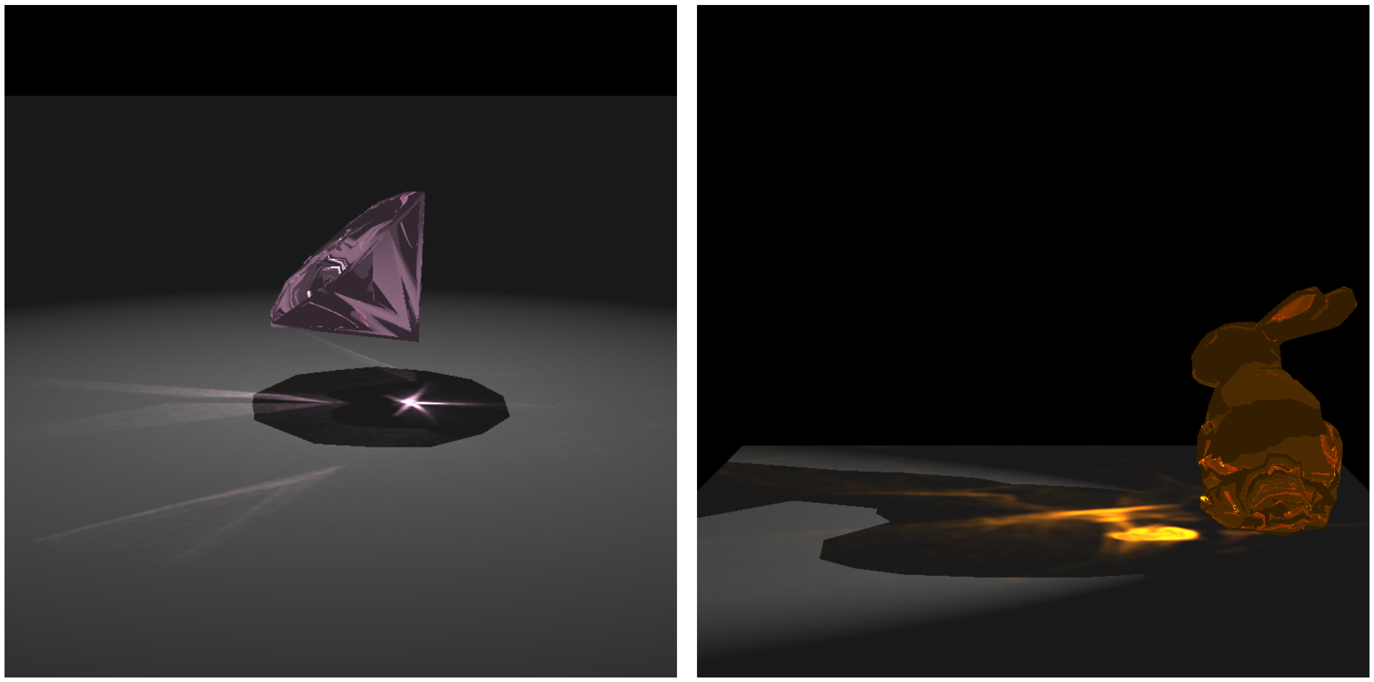


Fig. 3: The color of caustics can be estimated estimated very accurately. An animation for the pink diamond animation can be found at https://www.youtube.com/watch?v=NwYMuA0i4MA

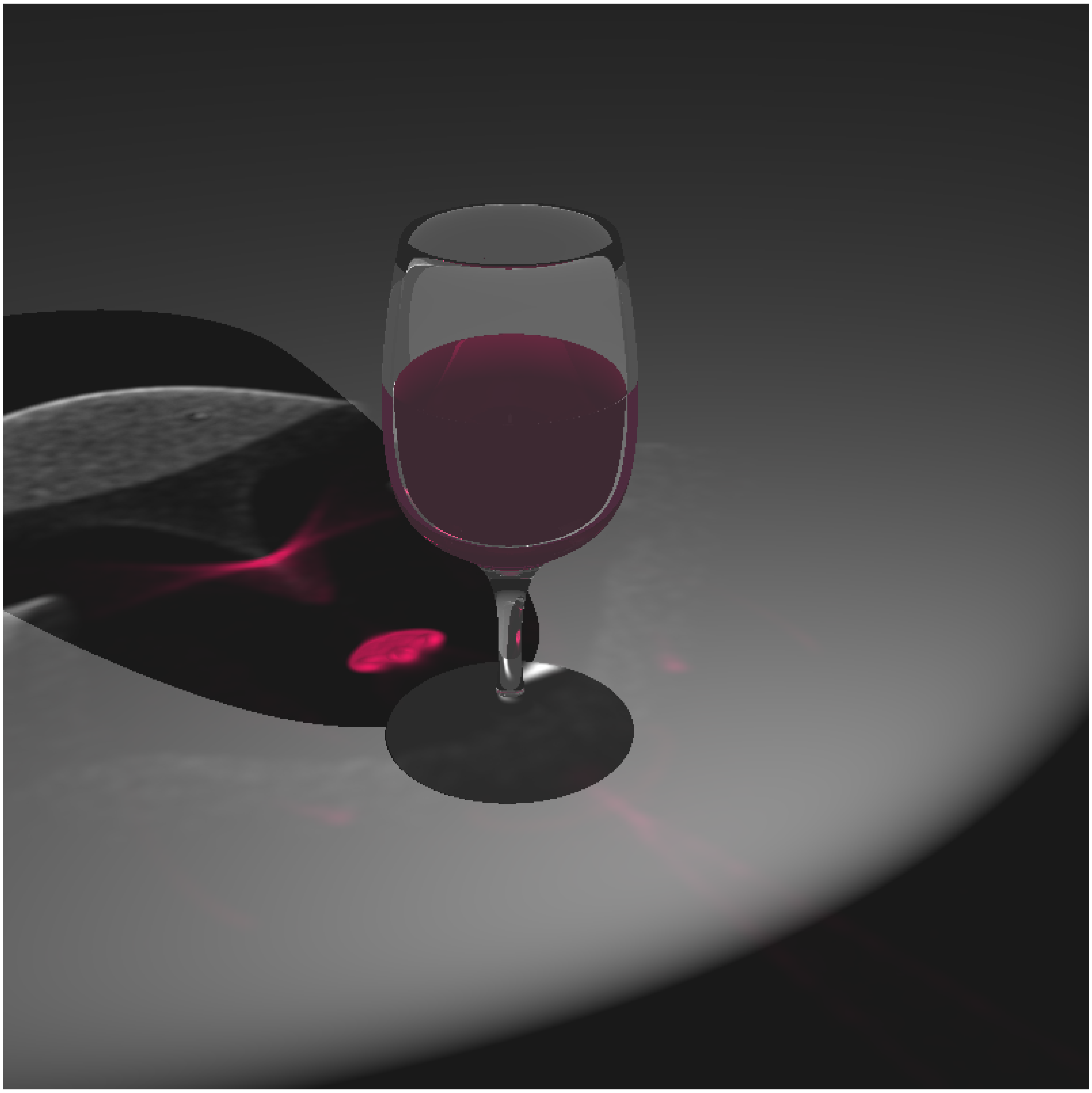


Fig. 4: Layering of transparent surfaces shown by a wine glass and wine. Rendered using 100000 photons. The liquid gets darker with more depth.

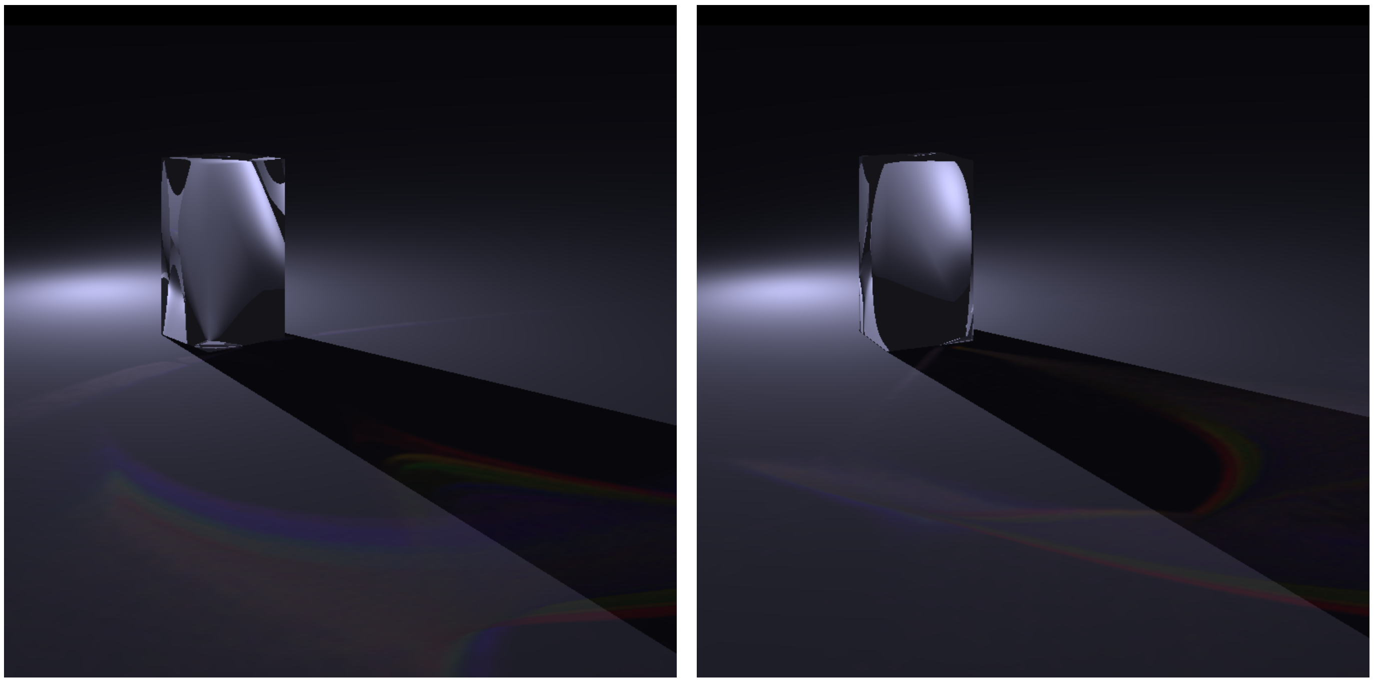
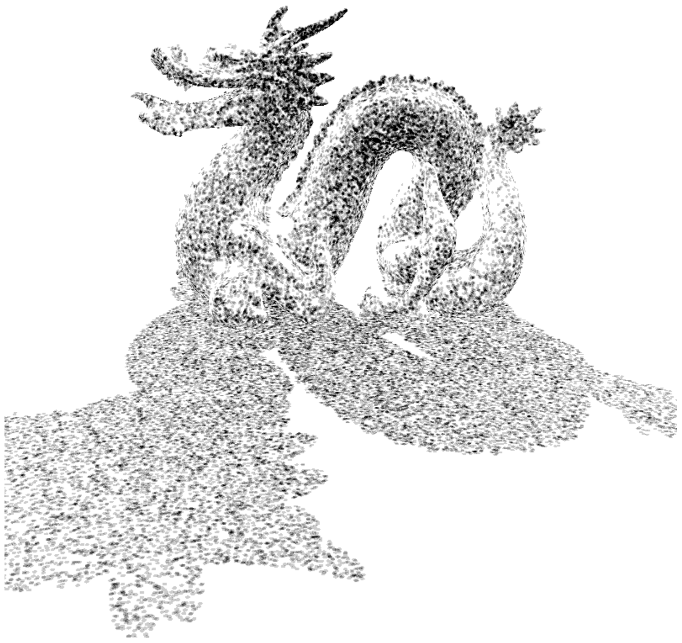


Fig. 5: Prism rendered at two different angles. The bands of light are more separated on the left, and they combine at the right. Rendered with 90000 photons. 5 wavelengths were sampled. A video with 90 degrees can be found at https://www.youtube.com/watch?v=irdqbK7mF7k

(a) Stippling effect on a sphere with a spotlight (b) Stippling effect on a sphere with an area light

Fig. 6: The difference in the choice of lights to render the stippling effect



(a) Stanford dragon rendered with a spot light. 50000 photons were used

(b) Bunny rendered with an area light. 30000 photons were used. Notice that the shadowed side of the bunny has much more shape than the lit side

Fig. 7: more complex geometries are handled quite nicely by the stippling effect using photon mapping