Depth of field, Bokeh and Lens Aberrations
CS500 Final Project

Josu Cubero Ruiz de Gopegui

Spring 2020

Abstract

Ray tracing is a computer graphics technique that simulates how light reflects and refracts in material surfaces to generate an image as close to reality as possible. In this document, there are covered some effects that can be added to a ray tracer that mimic how cameras work when taking pictures.

1 Depth of Field

This technique is commonly found in photography and it causes some objects to look out of focus based on the distance the light has to traverse before hitting the lens of the camera. Cameras let light come to the image plane through the lens aperture. Most computer graphics are based on a pinhole camera, where the area of the aperture is a single point in space and therefore, every object in the scene is in focus. To simulate depth of field in those cases, post processing effects are applied. However, to get a realistic depth of field effect, where objects get blurrier the further they are from the focus plane the ray tracer must replicate how cameras work in real life.

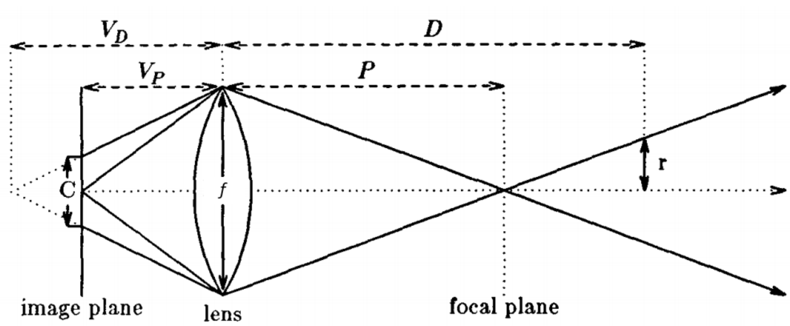


Figure 1: Circle of confusion [4]

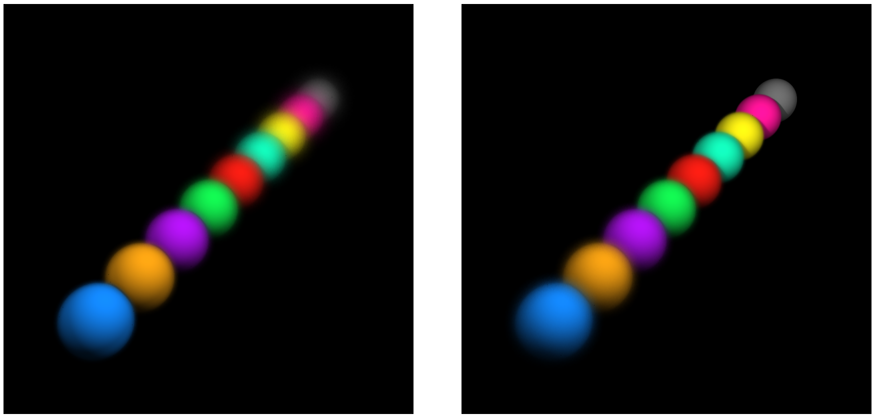
Cameras can modify depth of field using two main variables, the focal length V_p, and the aperture size f. The focal length, changes the distance at which the image plane is from the lens, and thus, the distance from the lens to the focal plane. The focal length also modifies the rate at which the objects get out of focus. The aperture size, determines how big is the area through which light comes to the image plane in the back of the camera. The aperture also affects exposure, which changes how bright the image looks and can be also modified by letting the shutter stay open for a longer or shorter time.

We can compute the distance to the focal plane by solving the following equation where n is the coefficient of refraction of the lens material, given by the Abbe constant, r_1 and r_2 are the radius of the lens diopters and S and S' are the distances from the lens to the image plane and the focal plane respectively (V_p and P in Figure 1).

$$\frac{1}{S} + \frac{1}{S'} = (n-1) * \left(\frac{1}{r_1} - \frac{1}{r_2}\right)$$

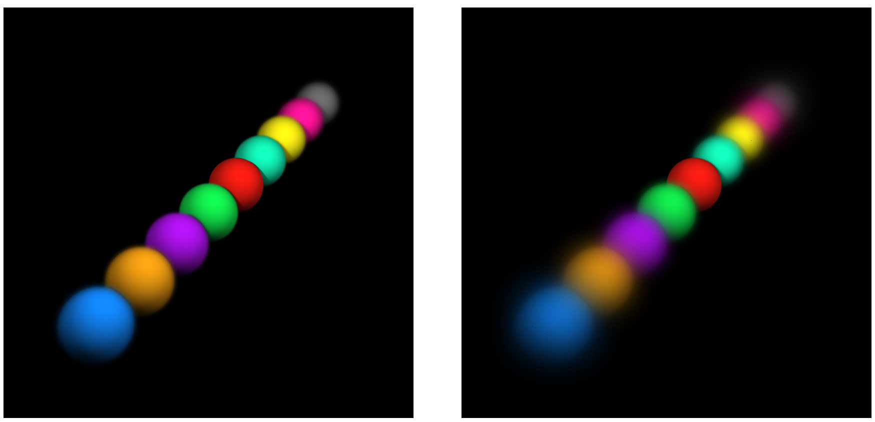
To compute the ray for a given pixel, the ray tracer will compute the ray for that pixel as normally and then compute the intersection of the ray with the focus plane. Taking that point as the target and a random point in the lens as a starting point, a new ray will be created. This process must be repeated taking random points in the lens and the final color will be given as the average of all the results.

```
vec3 compute_pixel_dof(Camera cam, Ray ray, int samples)
{
    vec3 color(0,0,0)

    /* focal length in this context is
    the distance from the camera to the focal plane*/
    vec3 focal_plane_point = cam.pos + ray.dir * cam.focal_length

    for (int i = 0 : samples)
    {
        // random point in the lens aperture
        vec3 start = cam.rand_point()

        Ray dof_ray(start, focal_plane_point)
        color += compute_color(dof_ray)
    }
    color /= samples
    return color
}
```



(a) Focal Distance 1.0 (b) Focal Distance 2.0

Figure 2: Focal Distance Examples



(a) Aperture 0.1 (b) Aperture 0.5

Figure 3: Aperture Examples

2 Shaped Bokeh

Bokeh is photography technique produced by placing the focus plane far from the the objects in the scene, such that they create an abstract, yet appealing picture. This effect can easily be obtained by setting the right values for the depth of field, however, this properties limit the blurred objects to look like circles. In order to change the shape of the objects, regardless of their original shape, it is possible to cover the aperture with a 2D polygon [2].

The problem now is how to define a polygon for the aperture and be able to sample random points uniformly inside it.

2.1 Uniform Sampling Polygons

A simple approach is to define explicit functions for defined polygons like a circle, a square or a triangle. However, this limits the amount of supported shapes to those that allow for uniform sampling.

2.2 Bounding Areas

This approach allows to sample uniform random points for any 2D polygon by making use of the previous method. A random point is sampled inside the bounding area (a square for example) and then, and a check is made to determine if the point lays inside the polygon.

```
vec3 rand_point_polygon(Polygon p)
{
    vec3 point
    do {
        point = p.bounding_area.get_rand_point()
    } while (!point_inside_polygon(point, p))

    return point
}
```

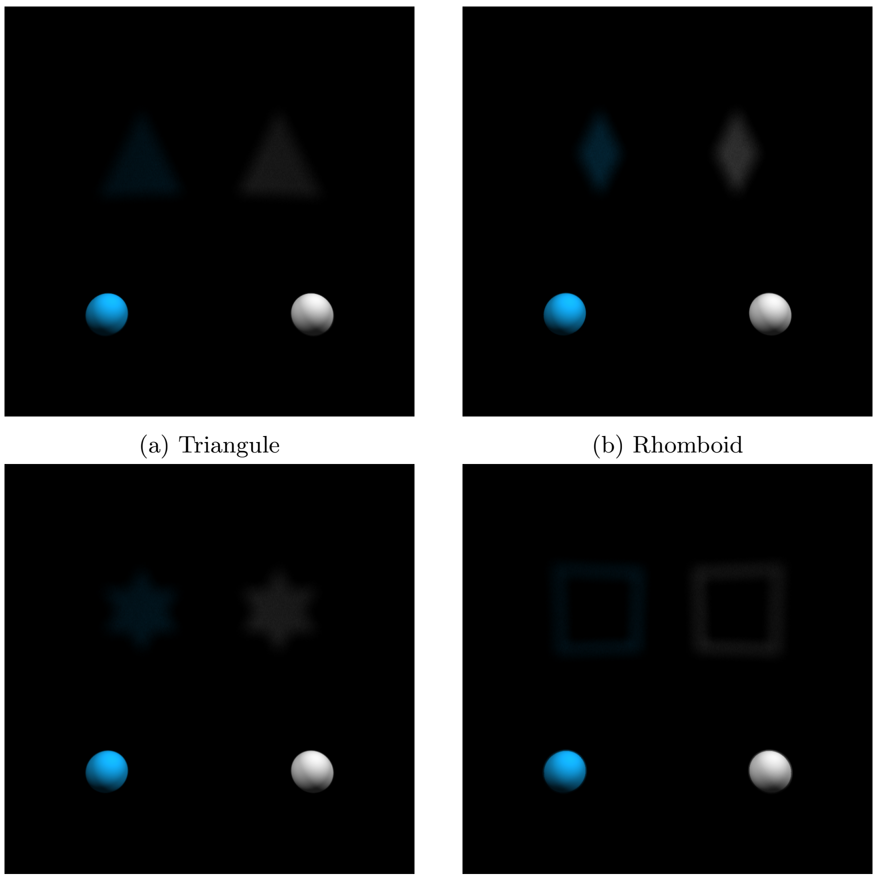
This algorithm gives the correct output but it will affect performance. The worse case scenario where the polygon takes the shape of a thin diagonal triangle will try to sample many points outside the polygon making extra computations for each pixel.

2.3 Alvaro's Method

As well as with bounding areas, the Alvaro's Method allows for sampling uniform random points in any 2D polygon. The method consists on triangulating the shape, giving each triangle a weighted heuristic value based on the area the triangle takes and sampling points inside each triangle, where the total heuristic value of the shape is one. To get a random point inside the triangle, we make use of its barycentric coordinates.

$$P_r = (1 - r_1)A + r_1(1 - r_2)B + r_1 r_2 C$$

```
vec3 alvaros_method(Polygon p)
{
    float triangle_h = rand(0,1)

    for (tringle : p.triangles)
    {
        if (triangle_h >= tringle.heuristic)
            return rand_point_triangle(triangle)
    }
}
```



(a) Triangule (b) Rhomboid (c) Star (d) Frame

Figure 4: Shaped Bokeh Examples

3 Lens Aberrations

Lens aberrations are produced because the way the light refracts through the lens material is not perfect, creating some interesting effects. The two aberrations covered in this document are on-axis. This means that the effect will become more pronounce the further the point in the image plane lies from the center of the optical axis.

3.1 Spherical Aberration

This aberration produces a change in the focal length based on the distance h from the optical axis at which the rays cross the lens.

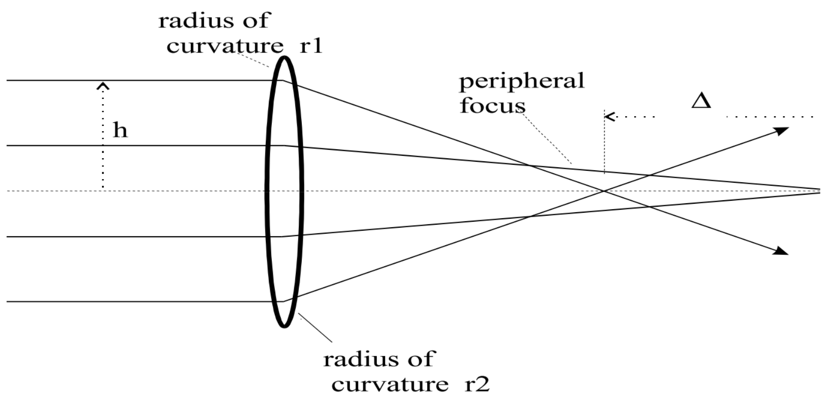


Figure 5: Spherical Aberration [1]

The difference of the focal length can be expressed as follows:

$$\Delta = [f_0 - f(h)] \approx \frac{1}{2} K h^2$$

Where K is a parameter that describes the spherical aberration based on the lens inner and outer radius, r_1 and r_2; and the distances from the lens to the image and focal plane, S and S'.

$$K = \frac{1}{4 f_0 n(n-1)} \left[\frac{n+2}{n-1} q^2 + 4(n+1)qp + (3n+2)(n-1)p^2 + \frac{n^3}{n-1} \right]$$

$$\text{for } q = \frac{r_2 + r_1}{r_2 - r_1}, \text{ and } p = \frac{S' - S}{S' + S}$$

3.2 Chromatic Aberration

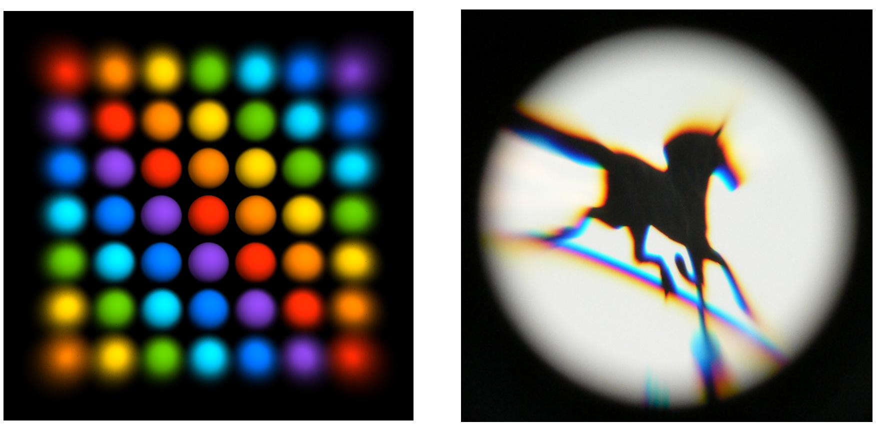
Similarly to the spherical aberration, the chromatic aberration occurs when the light rays refract the lens off center from the optical axis. By the properties of light and the material of the lens, the color channels with different wavelengths are going to be refracted differently, modifying the change of distance of the focal length.

Normally, the material index of refraction decreases with higher wavelengths, in order to compute the refraction value, the most common dispersion equations are the Schott and Sellmeier equations [3].

$$\text{Schott: } n^2(\lambda) = a_0 + a_1\lambda^2 + a_2\lambda^{-2} + a_3\lambda^{-4} + a_4\lambda^{-6} + a_5\lambda^{-8}$$

$$\text{Sellmeier: } n^2(\lambda) = a_0\frac{a_1\lambda^2}{\lambda^2 - b_1} + \frac{a_2\lambda^2}{\lambda^2 - b_2} + \frac{a_3\lambda^2}{\lambda^2 - b_3}$$

In both equations the wavelength of the light channel is represented as λ where the rest of the constants are properties of the material provided by the lens manufacturers.



(a) Spherical Aberration

(b) Chromatic Aberration

Figure 6: Lens Aberration Examples

4 References

References

[1] Lens aberrations and ray tracing.

[2] The blog at the bottom of the sea. Pathtraced depth of field and bokeh.

[3] Xiaohui Hu Fanjiang Xu Jiaze Wu, Changwen Zheng. Rendering realistic spectral bokeh due to lens stops and aberrations.

[4] Loren Carpenter Robert L. Cook, Thomas Porter. Distributed ray tracing.