5.0

2D Lens Flare

Yossarian King

Lens flare is an optical effect created by interreflection between elements of a lens when the camera is pointed toward a bright light. The result is a shifting pattern of translucent shapes and colors emanating from the light source. The effect is often seen in TV broadcasts when the sun enters the video camera's field of view.

In real life, lens flare is considered a defect, and camera manufacturers go to great lengths to eliminate it through special lens coatings. Video games, however, like to emphasize and exaggerate all the cooler aspects of reality, and lens flare is definitely cool. Real lens flare is due to complex interactions of light with surfaces in the optical system of a camera. Video game lens flare is all about appearances. This article shows how to implement an attractive lens flare effect using only a small amount of code and artwork, without needing to know anything at all about physical optics.

Approach

Real lens flares are created in the lens system of the camera and so naturally appear "on top of" the scene being viewed. Each element of a flare is a reflection of the light bouncing off a secondary lens and onto the primary lens. Since the lenses are in precise vertical alignment, the reflections fall along a line in the final image, where the distance of the reflection from the center of the image is proportional to the distance of the corresponding secondary lens from the primary.

These observations justify the treatment of lens flare rendering as a 2D problem. The flare is rendered as an overlay on the 3D scene, and the elements of the flare are rendered along a line intersecting the projected position of the light and the center of the screen, as shown in Figure 5.0.1.

At this point, we abandon all reference to physical optics and focus entirely on aesthetics. The lens flare effect is rendered with a small collection of textures, one for each style of flare element—circles, rings, hexagons, sunbursts, and so on—as shown in Figure 5.0.2. The gray-scale textures are combined with vertex colors to produce subtle coloring. Alpha blending is used to make the effect translucent. Elements are rendered in a variety of sizes.

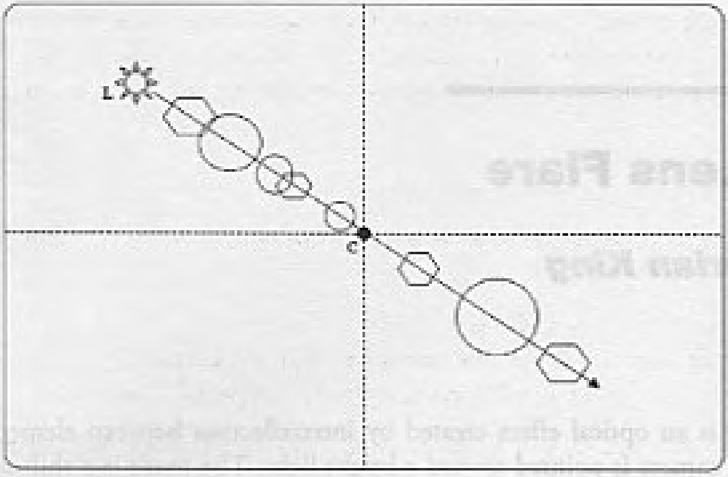

FIGURE 5.0.1. Lens flare rendering is a 2D problem. Elements of the lens flare are rendered along a line between the projected position of the light source and the center of the screen.

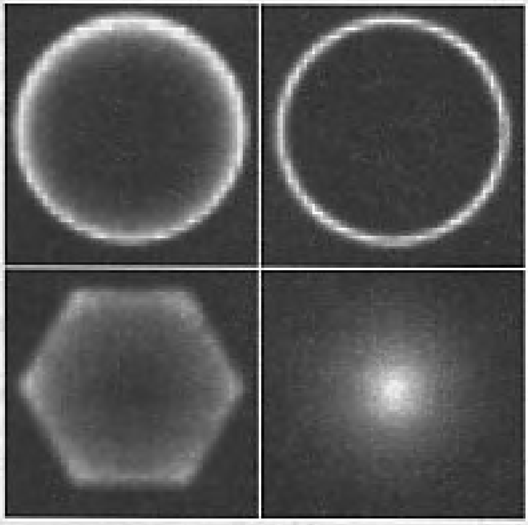

FIGURE 5.0.2. The lens flare effect is rendered using a small collection of gray-scale textures.

To be truly effective, the lens flare effect must animate convincingly with camera movement. The overall movement of the flare is determined by tracing the line from the projected light position through the center of the screen. Varying the size and translucency of the flare elements produces additional subtlety. This variation is achieved by scaling the size and alpha value of the flare elements based on the distance between the projected light position and the center of the screen; when the light is farther from the center, the elements are smaller and more transparent, and when the light is closer to the center, they become larger and more opaque.

A sample of the results of this approach are shown in Color Plate 4.

Implementation

Putting all this together boils down to performing the following steps for each element of the lens flare:

1. Determine position and size of the flare element.
2. Determine texture, color, and translucency of the element.
3. Render the element as a 2D sprite with the computed properties.

In this implementation, a flare is a collection of elements. Each element has the following static properties:

- **Texture.** The available textures (shapes).
- **Distance.** Proportional distance along the line from the light source to the center of the screen.
- **Size.** Normalized size of the element (before scaling).
- **Color.** Red-green-blue (RGB) color used to shade the element while rendering.
- **Alpha.** Translucency of the element (before alpha scaling).

The flare also has an overall scale factor and a maximum size, used to control the element sizes during rendering. These properties are all determined at initialization time. In the demo code, the properties can be determined randomly or loaded from a flare description file.

During rendering, the dynamic properties of each lens flare element are computed based on their static properties and the position of the light source on the screen. Texture and color of the flare are unaffected, but position, size, and alpha level are all dynamic, depending on the movement of the camera relative to the light source.

In pseudocode, the lens flare effect is rendered as follows:

```
function renderflare:
    flare       // flare object to be rendered
    (lx,ly)     // projected position of light on screen
    (cx,cy)     // center of flare (normally center of screen)
{
    // Compute how far off-center the flare source is.
    maxflaredist = sqrt(cx^2 + cy^2)
    flaredist = sqrt((lx - cx)^2 + (ly - cy)^2)

    // Determine overall scaling based on off-center distance.
    distancescale = (maxflaredist - flaredist)/maxflaredist

    // Flare is rendered along a line from (lx,ly) to a
    // point opposite it across the center point.
```

```
        dx = cx + (cx - lx)
        dy = cy + (cy - ly)

        for each element in flare
        {
            // Position is interpolated between (lx,ly) and
            // (dx,dy).
            px = (1 - element.distance)*lx + element.distance*dx
            py = (1 - element.distance)*ly + element.distance*dy

            // Size of element depends on its scale, distance
            // scaling, and overall scale of the flare itself.
            width = element.size * distancescale * flare.scale

            // Width gets clamped, so the off-axis flares keep a
            // good size without letting the centered elements
            // get too big.
            if (width > flare.maxsize)
                width = flare.maxsize

            // Flare elements are square (round) so height is
            // just width scaled by aspect ratio.
            height = width * aspectratio

            // Alpha is based on element alpha and distance scale.
            alpha = element.alpha * distancescale

            // Draw the element's texture with computed
            // properties.
            drawrectangle( element.texture, element.colour,
            alpha, px, py, width, height )
        }
    }
```

Source Code

The lens flare demo includes OpenGL source code and a Windows executable. The source code is separated into an API and sample code that uses the API. The API includes structures that define the properties of a flare as well as the following functions:

- **FLARE_initialize.** Initialize flare elements, given a list of properties by the caller.
- **FLARE_randomize.** Generate a list of flare elements with random properties.
- **FLARE_render.** Render the flare at a given screen position.

The demo uses these functions to create a lens flare effect controlled by the mouse. The mouse cursor is used as the screen location of the light source. Lens flares can be randomly generated or loaded from a file. See the README.TXT file included with the demo for additional details on the demo interface.