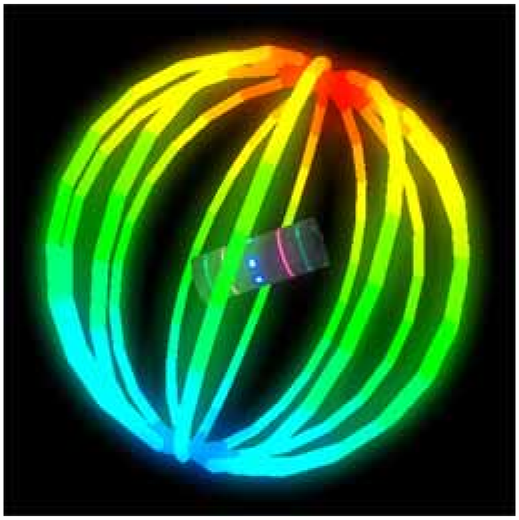
NVIDIA

Technical Report

Glow Sample



DEVELOPMENT

Glow

Glows and halos of light provide powerful visual cues about the brightness of objects. In reality, halos around bright objects are caused by light scattering in the atmosphere and by scattering within the tissues and fluid of our eyes. When viewing reproductions of reality, either in printed media, computer graphics, or film, our eyes are picking up light from sources that cannot reproduce the intensities we receive from real-world objects. For example, in looking at a photograph of a neon sign, the intensity of light from the image is several thousand times less than the intensity we'd receive from an actual neon sign. In viewing the photograph, the neon still appears to be bright because the photograph reproduces the glowy halo around the sign. The halo creates the perception of brightness, and even subtle halos have this effect.

Traditional real-time 3D graphics has no concept or means of creating glow around objects, so real-time scenes have appeared flat with subdued intensities, even when specular highlights are used. With the introduction of "Wreckless" (Bunkasha Games, Masake Kawase) on the Xbox, and more recently "Tron 2.0," "Splinter Cell," "Half-life 2," "Far Cry," and "World of Warcraft," real-time 3D games are rapidly adopting full-scene glow effects. The resulting scenes are truly stunning, featuring either life-like or fantastic effects, and they are much more appealing than the flat look of 3D graphics without any glow.

This paper describes, in brief, a technique very similar to that developed by NVIDIA and Monolith Productions for the popular "Tron 2.0" game, released by Buena Vista Interactive in 2003. More detail about the approach and about integrating the effect into a shipping game is provided in the "GPU Gems" article referenced below.

The blur effect is created by post-processing a rendering of the 3D scene. First, the ordinary 3D scene is rendered to the flip chain back buffer. During this rendering, the strength of the sources of glow at each pixel on screen is written to the alpha channel of the back buffer. The back buffer is then copied to a render-target texture (RTT) of lower resolution than the back buffer. This downsampled render-target texture is read several times in a separable convolution operation which blurs the glow sources out into soft halos of glow. The RTT containing the sources of glow is first read in several passes of rendering to another RTT to create a blur in the horizontal direction. This horizontal blur RTT is then read in several more passes of rendering to the final glow texture render target. The blurred glow is added onto the flip chain back buffer, resulting in the appearance of soft halos of light around each pixel originaly marked, using the alpha channel, as a source of glow.

This approach is compatible with full-scene anti-aliasing (FSAA), and you can enable FSAA in the sample through the "Change Device" GUI button. The FSAA compatability is made possible by the IDirect3DDevice9::StretchRect() function, which in this case is used to copy the anti-aliased flip chain back buffer to an aliased render-target texture for the post-processing operation. Alternatively, you could render the full scene to a render-target texture in the first place, but this would not allow for FSAA, since render-target textures cannot take advantage of hardware FSAA. Rendering to texture would also require additional memory to hold the full-resolution RTT and more texture samples to copy the full-resolution RTT to the flip chain back buffer. For these reasons, it is much better to render to the ordinary flip chain backbuffer in the first place and use StretchRect() to extract the contents.

The glow source information can be provided in a number of ways. This demo uses the diffuse texture alpha channel for some objects, and it uses a constant color alpha value for other objects. The alpha value could also be read from a per-vertex color value. For a complex game scene, you could also modulate the per-vertex or per-pixel alpha values according to some animated parameter or by the depth of objects in the scene so that the glow varies with time or distance.

Bibliography

1. Greg James, John O'Rorke, "Real-Time Glow" in "GPU Gems" ed. Randima Fernando, Addison Wesley : 2004, ISBN 0-321-22832-4, p 343 http://developer.nvidia.com/object/gpu_gems_home.html
2. Greg James, "Operations for Hardware-Accelerated Procedural Texture Animation" in "Game Programming Gems 2" ed. Mark DeLoura, Charles River Media : 2001, ISBN 1-58450-054-9, p 497
3. Masake Kawase, 2003 Personal Web site. http://www.daionet.gr.jp/~masa/column/2003-03-21.html
4. NVIDIA Developer Web site: 2003. http://developer.nvidia.com

Notice

Trademarks

Copyright

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com