2.7

A Simulation of Thermal Imaging

O'dell Hicks

Introduction

As forms of non-realistic rendering are becoming popular in graphics, so are simulations of technology-aided imaging, particularly in spy/military video games. Though the resulting images are aesthetically inferior to realistic rendering styles, such techniques can immerse players into the role of the spy they are controlling. This article presents a technique for a visual interpretation of temperature.

A Form of Lighting

While temperature is what is gauged, the reality is that objects with temperature emit infrared radiation (even ice cubes!), and they act as a kind of point light, emitting invisible light from everywhere on the surface. With a simple lighting algorithm, a heat reference for the object surface, and a color lookup texture, we can easily fake thermal imaging.

The Technique

The most time-consuming part of this technique is the surface temperature source. It could be done on a per-vertex basis, but that won't give much temperature detail unless the mesh is heavily tessellated. A better way is to take the base texture(s) and produce a grayscale temperature map, where pure black is the coldest and pure white is the hottest. Temperatures for non-living objects are pretty straightforward. For humans, exposed skin is generally the hottest area. Thin t-shirts should be cooler, and thicker clothing should be even cooler than that. If you desire, you may wish to blur out all distinguishable detail from the temperature map (such as facial features). In real life, imaging hardware does a variety of things, from featureless temperature gauging, to more detailed compositions.

The vertex shader first calculates the intensity of the infrared radiation reaching the eye/camera. This is the dot product between the eye's forward vector and the normal of the emitting object. In this sample, it's done on a per-vertex basis, but you may use normal maps to get a more detailed sampling. As the normal deviates from the direction of the eye, just like regular light, the infrared radiation is attenuated, resulting in a "cold" outline along the object edges. The shader passes this value (ranging

from 0 to 1) as a texture coordinate. The following snippet is the vertex shader for the technique:

```
vs_1_1
dcl_position  v0  ; vertex position
dcl_normal0   v1  ; vertex normal
dcl_texcoord0 v2  ; texture coordinate
m4x4 oPos, v0, c0 ; transform pos to homogenous clip space
m4x4 r0, v1, c4   ; Transform normal to world space
; Normalize the normal
dp3 r0.w, r0, r0
rsq r0.w, r0.w
mul r0, r0, r0.w
dp3 r1, c8, r0 ; Intensity of infrared emittance relative to the eye
 ; Texture coordinates
mov oT0.xy, v2 ; thermal map
mov oT1, r1    ; generated color lookup
```

This technique can be done with pixel shader version 1.4 and up; the sample provided uses version 1.4. In phase one of the pixel shader, we multiply the sampled temperature map (0 to 1, coldest to hottest) with the texture coordinate generated in the vertex shader (0 to 1, based on the intensity of the infrared radiation reaching the eye). The product of these values can be thought of as an overall temperature, and it coincides with the range in the thermal lookup texture. This texture is a gradient of colors, ranging from "coldest" to "warmest." In phase 2, we sample the thermal lookup texture with this coordinate. The following snippet is the pixel shader for the technique:

```
ps_1_4
texld r0, t0       ; thermal map
texcrd r1.rgb, t1  ; generated texture coordinate
mul r1.xyz, r0, r1 ; scale the thermal coordinate
phase
texld r1, r1       ; sample the color lookup with the coordinate in
r1
mov r0, r1         ; output the pixel
```

Conclusion

This article presents a simple way to produce the technique often referred to as "heat vision." As in real life, this technique can be modified to produce the green-hued "night vision." Using a different thermal color lookup and modulating it with the object's base texture would produce this effect. Finally, it is common to render a bit of noise or "snow" over the scene to further simulate imperfections in thermal imaging hardware. In this sample, the noise is achieved by drawing two triangles over the entire screen, with the texture coordinates randomly sampling an area of a static noise texture. The random change in texture coordinates gives the appearance of the noise being animated.