Supplemental Data:

1. Video results.
2. The full paper with brighter-than-sRGB gamma.
3. The sources of error (figure 10 in the paper) experiment repeated on the Cornell Box.
4. A figure showing the latency of radiosity through frames in the door sequence from the *Office* scene.
5. Screen-space ray trace listing.

1. Additional Figures

Figure 1: Four single-scattered radiosity estimates distinguish the impact of some underlying approximations in our fast method.

Figure 2: Screen-space radiosity (bottom) can capture simple dynamic illumination phenomena well compared to a static environment map probe (top). We show a completely unlit room with a door to a very bright adjacent door as the door closes through multiple frames of animation.

© 2014 NVIDIA Corporation and D. Nowrouzezahrai. All rights reserved.
2. Ray Cast Source Code

Listing 1 is our GLSL implementation of ray-scene intersection against the surface defined by the deep G-buffer’s depth layers, implemented by ray marching. The interesting parameters are:

- `depthBuffer01`: Camera-space linear Z floating-point buffer, with R = layer 0 and G = layer 1.
- `clipInfo`: Clipping plane information: \((z_f = \infty) \Rightarrow \vec{3}(z_n, -1.0f, 1.0f) : \vec{3}(z_n + z_f, z_n - z_f, z_f)\), where the near and far plane constants \(z_n\) and \(z_f\) are both negative values.
- `jitterFraction`: A number between 0 and 1 used to offset the ray randomly at each pixel.
- `distance`: The maximum distance to trace. On return, the actual distance traced.
- `hitTexCoord`: The \([0, 1]\) texture coordinate in the depth buffer of the hit location.
- `which`: Index of the layer hit.
- `guardBandFractionX/Y`: Guard band size relative to frame buffer size.

The function returns true if something was hit before the maximum distance or limiting number of iteration steps and false otherwise.

```
bool castRay2(vec3 csOrigin, vec3 csDirection, 
mat4 projectionMatrix, sampler2D depthBuffer01, 
vec2 depthBufferSize, vec3 clipInfo, 
float jitterFraction, int numSteps, 
const float layerThickness, inout float distance, 
out vec2 hitTexCoord, out int which) {
    // Current point on the reflection ray in camera space
    vec3 P = csOrigin;
    
    // Camera space distance for each ray—march step
    float stepDistance = distance / numSteps;
    float rayBumpDistance = 0.02;
    
    // Off screen
    hitTexCoord = vec2(-1, -1);
    
    // Amount that P increments by for every step
    vec3 PInc = csDirection * stepDistance;
    P += PInc * (jitterFraction + 0.5) + epsilon * csDirection;
    int s = 0;
    for (s = 0; s < numSteps; ++s) {
        // Project the trace point P into texture space.
        // Note that the projection matrix maps to \([-1, 1]\]
        // coordinates after homogeneous division
        vec4 temp = projectionMatrix * vec4(P, 1.0);
        
        // texture space P: Homogeneous div & remap to [0,1]
        vec2 tsP = (temp.xy / temp.w) * 0.5 + 0.5;
    }

    // Pixel space P
    ivec2 psP = int2(depthBufferSize * tsP);

    // Camera space z of the background at each layer
    vec2 sceneZ = texelFetch(depthBuffer01, psP, 0).rg;

    // The depth range that the ray covers at this pixel
    float intStartRayZ = P.z - PInc.z * 0.5;
    float intEndRayZ = P.z + PInc.z * 0.5;
    float rayZMin = min(intStartRayZ, intEndRayZ);
    float rayZMax = max(intStartRayZ, intEndRayZ);
    vec2 sceneZMin = sceneZ - layerThickness;
    vec2 sceneZMax = sceneZ;

    // Use a macro instead of a real FOR loop because we
    // need to use a BREAK statement below
    # for (int layer = 0; layer < 2; ++layer)
    // If the point on the ray is behind the scene point at the
    // same location... (As an optimization, break out of the
    // containing loop here, but don’t handle the result until
    // outside the loop)
    if ((rayZMax >= sceneZMin[(layer)]) &&
        (rayZMin <= sceneZMax[(layer)]) &&
        (sceneZ[(layer)] > -99999)) {
        // Hit...or off screen
        hitTexCoord = tsP;
        which = (layer);
        break;
    }
    # endfor

    P += PInc;
}
```

Listing 1: Screen-space ray tracing code in GLSL for two depth buffer layers.