Using Deep G-Buffers to Approximate Global Illumination

Trevor Morris
Student, University of California at Santa Barbara

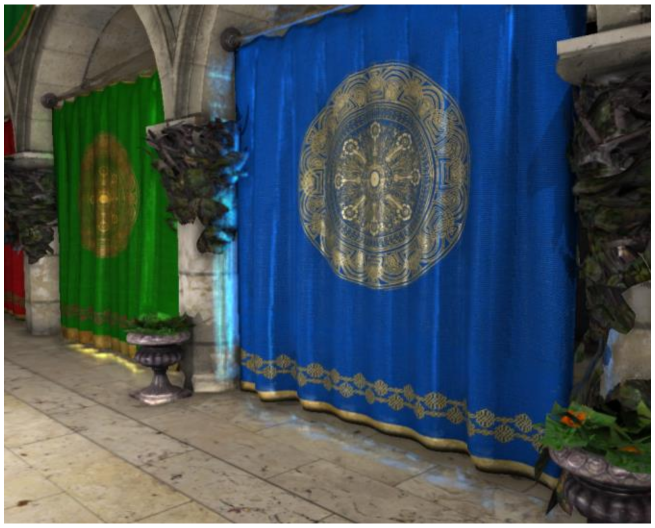

Figure 1: *Result of single-scatter radiosity algorithm + ambient occlusion both using a 2 layer deep g-buffer.*

Abstract

In this paper, I will describe my implementation of Deep G-Buffers as described by Mara et. al. [2016] and how I applied them to produce two approximate global illumination effects: ambient occlusion and single scatter radiosity.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display Algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Radiosity;

Keywords: radiosity, global illumination, ambient occlusion

1 Introduction

G-Buffers, or Geometry Buffers, are commonly used in real-time graphics applications for "screen-space" effects and to provide efficiency with deferred shading. Using G-Buffers, geometry is typically rendered to window-dimensioned textures which can store depth, normals, positions, and albedo color. These textures can then be used as inputs to other stages of shading, using only these fragments that will end up on the screen and without having to render the original geometry multiple times.

Mara et. al. [2016] proposes Deep G-Buffers, which are G-Buffers with multiple layers in a similar fashion to depth peeling. Each additional layer added is one step closer to camera space voxels (figure 2). Even adding one layer can greatly improve the quality of screen-space global illumination effects. In this paper, I focus on a 2-layer deep g-buffer.

*e-mail: trevormorris@umail.ucsb.edu

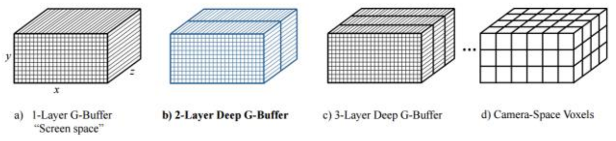

Figure 2: *Comparison to camera-space voxels [Mara et. al. 2016]*

2 Deep G-Buffer Implementation

A naïve approach to generating a 2-layered g-buffer would be to render the geometry to the first layer, and then render it again to the second layer if the depth exceeds the value in the first layer. Mara et. al. proposes a method to generate a 2-layer g-buffer in a single pass, using a geometry shader and an oracle to predict the first layer's depth buffer. Mara et. al. provided four variants to approximate the oracle.

I implemented the 'Previous' variant, which uses the first layer depth buffer of the previous frame as an oracle comparison depth buffer. Using this variant, errors arise with movement or camera movement, however the errors only effect the second layer of the g-buffer, which holds hidden surfaces.

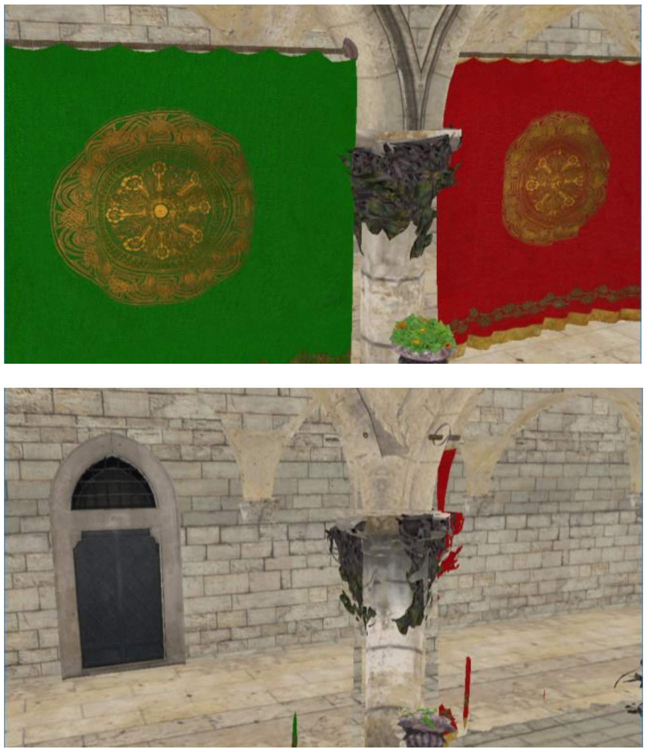

Figure 3: *Albedo of my Deep G-Buffer's first layer (top) and second layer (bottom), using a minimum separation of 1m.*

2.1 Minimum Separation

In order to get a useful second-layer, Mara et. al. enforces a *minimum separation* between layers. Fragments that are too close to the previous layer are not used for the second layer. This prevents complex geometry and fine detail from occluding the more useful secondary surfaces. I used a minimum separation of 1m for my implementation.

Since minimum separation is defined in world space, and depth is stored non-linearly, for layer 2's generation I had to linearize the depth from the comparison depth buffer, add the minimum separation, and then convert that number back to non-linear depth space in order to compare it to each fragment's depth.

3 Applications for Global Illumination

3.1 Ambient Occlusion

I first applied my deep g-buffer towards ambient occlusion, using the following formulae which Mara et. al. adapted from McGuire et. al.'s Scalable Ambient Obscurance [2012]:

$$AV(X) = \max\left(0, 1 - \sqrt{\frac{\pi}{N}\sum_{i=1}^{N}\max\left(0, A_{i,0}, A_{i,1}\right)}\right) \qquad (1)$$

$$A_{i,j} = AO(X, R(Z[j], i)) \qquad (2)$$

$$AO(X,Y) = \left(1 - \frac{\vec{v}\cdot\vec{v}}{r^2}\right)\cdot\max\left(\frac{\vec{v}\cdot\hat{n}_X - \beta}{\sqrt{\vec{v}\cdot\vec{v}+\varepsilon}}, 0\right), \qquad (3)$$

where *R(Z, i)* is the position of the *i*th sample reconstructed using depth buffer *Z*, and *X* is the position of the fragment. β is a bias constant. I used a radius *r* of 2m and a *β* of 0.05.

The algorithm tests for obscurance against both layers of the g-buffer and uses the highest of the two for the final value.

3.2 Single-Scatter Radiosity

Next, I implemented Mara. et. al.'s deep g-buffer extension of Soler et. al.'s screen space radiosity approximation [2009] given by the formula below:

$$E(X) \approx \frac{2\pi}{M}\sum_{\text{samples}} B(Y)\max(\hat{\omega}\cdot\hat{n}_X, 0),$$

where *B(Y)* is the radiance emitted from position *Y*, and *ω* is the normalized direction between *Y* and *X*. The incident irradiance *E(X)* at point *X*, is converted to outgoing radiance with the following formula:

$$B(X) = E(X)\cdot\rho_X\cdot\text{boost}(\rho_X),$$

$$\text{boost}(\rho) = \frac{\max_\lambda \rho[\lambda] - \min_\lambda \rho[\lambda]}{\max_\lambda \rho[\lambda]},$$

where ρ is the reflectance of the material. I used a reflectance of 1. The *boost* term amplifies scattering from highly saturated surfaces, where the highest R, G or B term is much larger than the lowest.

The input radiance comes from diffuse lighting which is calculated during the lighting pass. The output radiance from the radiosity pass is blurred and added back to the Phong shading from the lighting stage for the final output color.

The algorithm calls for N samples in both layers of the g-buffer, but only the M for which the surface normals face within 90 degrees of each other contribute to the final radiosity.

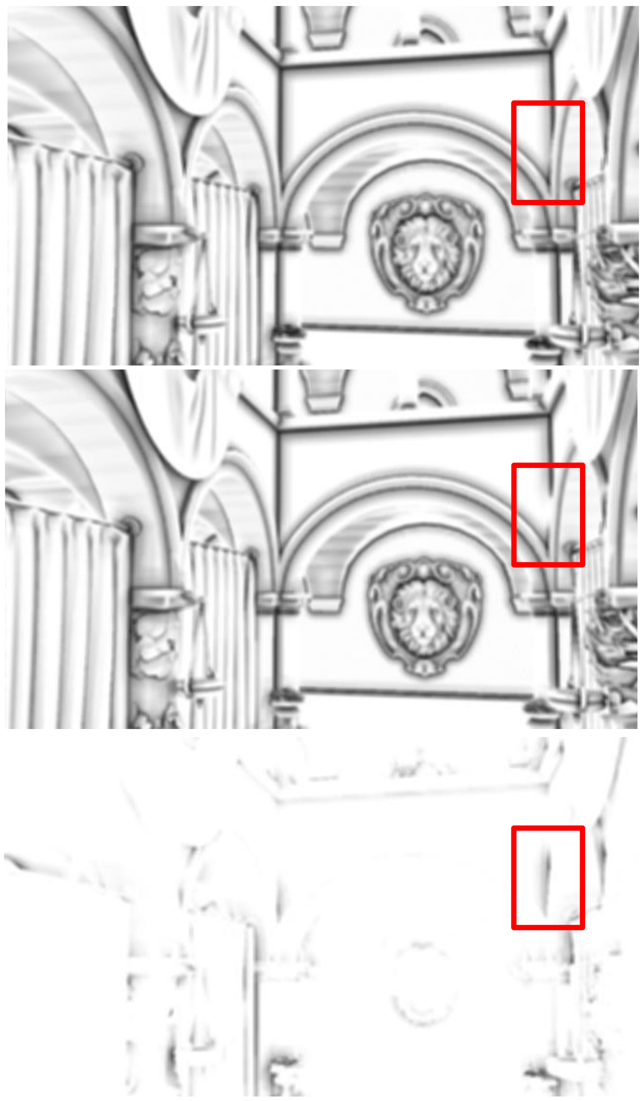

Figure 4: *Ambient Occlusion from both layers (top), only the first layer (middle) and only the second layer (bottom). The red box highlights an area where the second layer provided a very noticeable correction to an error in a single layer only approach.*

4 Analysis

For Ambient Occlusion, adding the second layer now correctly accounts for obscurance due to occluded objects, which is particularly noticeable between the wall and the arch highlighted in red (figure 4).

Similarly, for radiosity, we can see the contribution of the second layer by comparing figures 1 and 5. The second layer allows for the occluded portion of the green curtain to contribute color bleeding to the floor.

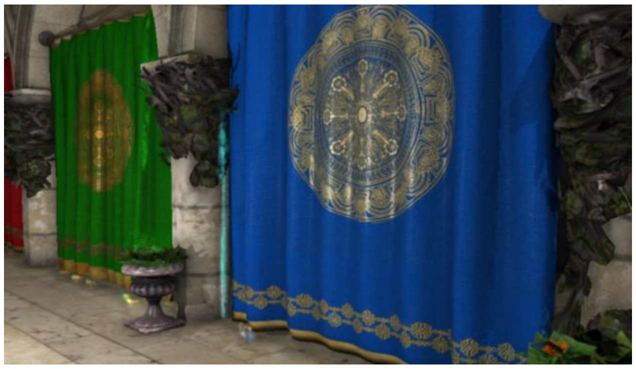
Figure 5: *Radiosity effects from the second G-Buffer layer only.*

Since the Deep G-Buffer can be generated in a single pass, it is fairly inexpensive, but some concerns may be memory usage and bandwidth for large display dimensions.

4.1 Issues & Improvements

Mara et. al. used a "Quasi-Monte Carlo" sampling method, while I used a normal-oriented hemisphere approach where the points are more densely clustered towards the origin. Using the QMC approach instead, I could improve quality and performance using fewer samples while reducing noise.

For use in a real application, I would explore using a different variant of the G-Buffer oracle instead of using the previous frames depth buffer. Other approaches introduce a frame of latency ('Delay'), use velocities to predict the future depth buffer ('Predict'), or reverse reproject to compute past visibility ('Reproject'). Using the 'Previous' variant, artifacts are very noticeable when the camera is in motion, although a higher framerate may help alleviate this issue.

5 Conclusion

The deep g-buffer is a relatively easy (when compared to voxels) and inexpensive improvement to screen space global illumination approximations. Having just one extra layer allows for most of the hidden surfaces to contribute to illumination effects, which greatly improves realism.

References

MARA M., MCGUIRE M., LUEBKE D.: Fast Global Illumination Approximations on Deep G-Buffers. NVIDIA Technical Report NVR-2014-001, 2014.

MCGUIRE M., MARA M., LUEBKE D.: Scalable ambient obscurance. In HPG (June 2012).

SOLER C., HOEL O., ROCHET F., HOLZSCHUCH N.: A Fast Deferred Shading Pipeline for Real Time Approximate Indirect Illumination. Tech. rep., Institut National de Recherche en Informatique et en Automatique, 2009.