An investigation of real-time deep G-Buffer applications in screen-space rendering

Michael Daltrey
School of Arts, Media and Computer Games
University of Abertay Dundee
DUNDEE, DD1 1HG, UK

ABSTRACT

Context/Background
Screen-space rendering is very common in modern games as it allows for complex effects to be rendered independent of the scene geometry. However, they can suffer from visual artefacts due to a lack of depth complexity. One way of reducing the effect of these artefacts is with the use of deep G-Buffers.

Aim
To develop and evaluate potential real-time applications of deep G-Buffers in screen-space rendering.

Method
A DirectX 11 powered rendering engine will be developed to test the viability of deep G-Buffer use in real-time rendering. The renderer will demonstrate common screen-space rendering effects using both a standard single-layer G-Buffer and a two-layer deep G-Buffer.

Visual Studio 2015's integrated profiling tools will be used to gather performance measurements for both CPU and GPU. To ensure realistic results, measurements will be taken using a complex scene of static and dynamic objects.

Results
A comparison of the visual results will showcase the expected improvements afforded by deep G-Buffers. The performance measurements will be evaluated to determine the viability of deep G-Buffer usage in modern games. It is expected that the quality of the image will improve substantially, and in most situations, the additional performance costs will be an acceptable trade-off for this improvement in visual quality.

Conclusion
This project will discuss whether the additional performance cost of using deep G-Buffers is worth the visual improvement that they bring. If this is the case it could foreshadow some major visual improvements in games over the next couple of years.

Keywords
Rendering, Screen-Space, Deferred Shading, Geometry Buffer, Ambient Occlusion, Performance, Optimisation, Global Illumination, Direct X, Layered Depth Images, Depth of Field, Deep G-Buffers.

1. INTRODUCTION

Recent trends in the game industry has seen the adoption of multi-pass rendering in many commercial engines (Anon 2009, Vosburgh 2010, Magnusson 2011). Most common of these techniques is deferred shading, pioneered by Deering et al back in 1988. Deferred shading separates the shading of the geometry into two separate passes. First object and material information is gathered into a set of render targets commonly known as the Geometry Buffer (G-Buffer) (Saito and Takahashi 1990). In a second pass the shading is calculated taking the G-Buffer as input.

Although the information stored in the G-Buffer is predominantly used for shading the scene, it can also be useful for producing screen-space post processing effects. This is advantageous as it doesn't require additional passes over the geometry. Common screen-space rendering effects such as ray-traced reflections and ambient occlusion can produce appealing results with minimal additional cost. Since the G-Buffer only stores depth and colour information for visible objects, parts of the image can flicker as the camera moves and objects come in or out of view. The additional occluded information is often needed to provide a temporally stable result.

Multiple views of the scene can provide enough information to improve the stability of screen-space rendering effects as shown by (Vardis, Papaioannou and Gaitatzes 2013). Layered Depth Images (LDIs) (Shade et al 1998), provided a method to store depth and material information for the primary visible layer of objects and all layers of hidden objects. There are many multi-pass methods available for generating LDIs (Everitt 2001, Myers and Bavoil 2007, Bavoil and Myers 2008) most seeing use in order-independent transparency techniques. As the scene complexity of modern games increases it becomes more expensive to perform scene traversal multiple times. Possibly applying pre-rasterization steps such as tessellation and occlusion culling unnecessarily.

Deep G-Buffers (Mara, McGuire and Luebke 2013), a recent variation on Layered Depth Images shows that it is possible to generate a two-level LDI in a single pass over scene geometry. They showed that deep G-Buffers can be applied to screen-space global illumination effects, improving stability compared to single layer G-Buffer methods. Deep G-Buffers are also applicable to many other screen-space rendering effects as mentioned in the 2013 paper "There are many potential applications of a deep G-buffer, including global illumination, stereo image reprojection, depth of field, transparency, and motion blur" (Mara, McGuire and Luebke 2013, p. 4).

Deep G-Buffers have proven to be an effective tool to improve the quality and stability of screen-space rendering effects. This year (Mara et al 2016) presented new techniques to reduce the time it takes to generate deep G-Buffers substantially, making it plausible for real-time applications. This project aims to build on the current research by investigating additional applications of the technique. This investigation will help to demonstrate the visual improvements that can be achieved using deep G-Buffers, performance comparisons will serve as evidence for their viability as a technique in modern games.

The project aims to develop an application to demonstrate the effectiveness of deep G-Buffers to improve various screen-space rendering effects, assessing its use in depth of field, motion blur, transparency or stereo image reprojection. Giving good evidence to support their use in modern games.

Key objectives

- Review current trending screen-space rendering techniques
- Identify effects that could benefit from the use of deep G-Buffers
- Implement chosen effects using both a single layer and a two-layer deep G-Buffer
- Critically analyse the performance results and make recommendations of the commercial viability of deep G-Buffers in modern games.

Research question

Can deep G-Buffers be used to improve screen-space rendering effects and are they viable in modern games?

2. BACKGROUND

2.1 Layered depth images

Layered Depth Images (LDIs) (Shade et al 1998) were originally intended to improve the results of 3D image warping algorithms (Mark, McMillan and Bishop 1997). 3D image warping was used to increase the apparent frame rate of a conventional renderer by warping the image to appear as though it was viewed from a nearby viewport, creating smoother transitions between rendered frames. To deal with occlusion issues multiple views of the scene are warped to fill in the gaps. The choice of the locations of the additional views heavily affected the result.

LDIs achieved the same results using a single viewport composed of multiple layers of depth and material information. Removing the dependency on multiple filler views made LDIs a more robust method for 3D image warping. Three methods for generating LDIs were expressed in the paper. However, due to long generation times they are not suitable for games

Since the original paper hardware-friendly methods for generating LDIs have been developed. Depth peeling (Everitt 2001) and k-buffers (Myers and Bavoil 2007) demonstrate multi-pass methods of generating LDIs of any size. Most commonly used in order-independent transparency.

2.2 Deep G-Buffers

In 2013 (Mara, McGuire and Luebke 2013) Demonstrated their novel method for generating 2-level Layered Depth Images (Given the name "deep G-Buffers") in a single pass over the scene geometry. The aim of the paper was to "show that two layers with a minimum separation distance make a variety of screen-space illumination effects surprisingly robust" (Mara, McGuire and Luebke 2013, p. 1). Specifically looking at ambient occlusion, approximate radiosity and ray-traced mirror reflections.

They found that the "second-closest surface often provides little additional information about the scene" (Mara, McGuire and Luebke 2013, p. 2) and that for best results there needed to be a constant minimum separation between the layers of the deep G-Buffer. Current depth peeling methods could achieve this in two passes over the geometry but it was not possible to achieve in a single pass with current techniques. Deep G-Buffer generation in a single pass over the geometry was achieved by using an "oracle" to predict the depth of the next frame before it is rendered. They presented 4 different oracles, the most effective of which was a variation of reverse reprojection caching (Nehab et al 2007). This uses screen-space velocity vectors to calculate the current fragments position in the previous frame, sampling a predicted depth value from the previous depth buffer.

In the following three years, the same authors produced two more papers improving on the original techniques. One was released in 2014 and the latest was released in June of this year. The original paper demonstrated a deep G-Buffer generation time of 33ms for the San Miguel scene running on a GTX Titan. In the latest paper the times have been reduced substantially, down to just under 6ms running the same scene and resolution but on a newer GTX 980 card. As well as the use of a newer card a new hardware friendly variation of their depth reprojection technique was introduced. Improving second layer generation by up to 37% in some cases (Mara et al 2016).

This increase in speed demonstrated that real-time frame rates could be achieved. However, it does still add an additional overhead to the G-Buffer generation. It has already been shown that the additional cost is acceptable for global illumination effects. This project aims to investigate the other applications of the technique and the possible performance costs involved.

2.3 Screen-space rendering

(Mara et al 2016) Demonstrated improvements to Alchemy ambient occlusion (McGuire et al 2011), Ray Traced reflections (Sousa, Kasyan and Schulz 2011) and screen-space approximate radiosity (Soler, Hoel and Rochet 2010). Which all adapted well to a deep G-Buffer implementation.

The other suggested applications of deep G-Buffers given are "stereo image reprojection, depth of field, transparency, and motion blur" (Mara, McGuire and Luebke 2013, p. 4).

2.3.1 Stereo image reprojection
Stereo image reprojection is used to generate a stereoscopic image from a 2D image using the available depth information. This works a lot like the 3D image warping algorithms explained previously. Current implementations suffer from occlusion issues (Schulz 2010, Hoef and Zalmstra 2011) that could be reduced using the additional layer available in a deep G-Buffer. However, the cost of generating the deep G-Buffer would be close to the cost of rendering a proper stereoscopic image, which in the end would produce more accurate results.

2.3.2 Transparency
Order-independent transparency (OIT) allows for a scene of transparent object to be rendered without any prior depth sorting. Layered depth images has been the common solution for OIT algorithms. Everitt (2001) finds that accurate results can be achieved with 3 layers, while (Bavoil and Myers, 2008) demonstrate OIT using 4 layers. Deep G-Buffer generation would need to be extended to at least one additional layer to achieve accurate results. Performing depth prediction for another layer would likely be too costly to be beneficial over current OIT algorithms.

2.3.3 Motion blur
To make artificial images appear more realistic it is common practice to approximate effects seen when using real cameras. Most frequent of these are motion blur and depth of field. Motion blur is slightly less interesting as under movement minor artefacts and inaccuracies are harder to notice as demonstrated by (Suchow and Alvarez 2011).

2.3.4 Depth of field
Depth of field (DoF) on the other hand is visible while static and under movement.

One DoF effect seen in the real world that is yet to be seen in games is partial occlusion: Out of focus objects near the camera are semi-transparent resulting in partially visible background objects. This has been investigated by (Wimmer and Schedl 2013) using multiple layers generated from a k-buffer. The paper requires many layers to produce the effect, for most scenarios a 2-layer deep G-Buffer could be sufficient to produce an effect of similar quality at faster frame rates.

3. METHOD
3.1 Project development
To properly investigate the applications of deep G-Buffers a 3D rendering engine will be developed. The renderer will use the DirectX 11 API and be built for Windows 10. The deep G-Buffer implementation will use the standard reprojection technique as the hardware-friendly method presented in the latest paper requires functionality that is not available in DirectX 11. The performance impact of this should not be noticeable in most scenarios, possibly only exhibiting slower performance for very complex scenes.

The same ambient occlusion, approximate radiosity and ray-traced reflection effects demonstrated in the paper will be integrated into the engine. In parallel to this work, further research will be conducted to determine which previously mentioned screen-space rendering effect would benefit most from deep G-Buffer integration. A set period of two weeks will be allocated to attempt an implementation of the chosen effect. Any longer than two weeks would not leave enough time for evaluation.

To properly test the application, sufficiently complex assets will be required. The engine includes a 3D model converter using the Open Asset Import Library. The converter takes a common 3D file format and converts it to an engine friendly binary format. Storing 3D files in this manner vastly improves model load times, resulting in more time focussed on development and less time wasted waiting for models to load. On top of this the engine includes a comprehensive debug menu, including publicly accessible properties that can be modified at runtime. Improving the efficiency of development.

3.2 Evaluation
Performance measurements will be gathered for both CPU and GPU using the analysis tools available in Visual Studio 2015. The graphics analyser tool produces a very detailed report, including the time taken for each draw call in a single frame. This separates how long it takes to generate the deep G-Buffer from the rest of the shading. Multiple frames can be captured in a session allowing for direct comparison of different methods under the exact same circumstances. A single frame isn't always representative of the overall performance; thus, additional measurements will be taken of a live frame rate in the engine. Giving an overall performance measurement that can be used to compare effects.

The results available in the deep G-Buffer papers will be used as a reference to compare what effect the use of different hardware and graphics API has had on the speed of the deep G-Buffer generation. It is expected that due to the lack of the latest hardware friendly prediction method the generation times will be marginally slower.

3.3 Schedule
The engine is already in a functional state to begin implementing the deep G-Buffer techniques. Figure 1 displays the expected and latest allowable delivery dates for all stages of the project.

3.4 Risk analysis
It is possible that the project is too ambitious and not all features can be completed. If it appears any task is taking longer than the given maximum duration, then the focus of the project will change. Instead focusing on getting a single effect running using the deep G-Buffer, then comparing results with those found in the deep G-Buffer papers.

As this is an investigation it is possible that no benefits will be found for the chosen rendering effect. It is also possible that the result doesn't run at an interactive frame rate. If either of these issues should arise it still provides information useful for answering the research question.

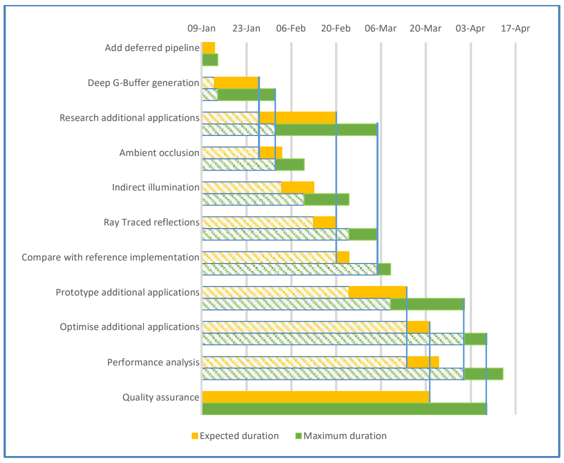

Figure 1– Project schedule Gantt chart. Vertical blue lines are used to show the beginning and end of tasks running in parallel

4. Summary

Current screen-space rendering techniques can produce reasonably accurate images at low cost. The lack of depth and material information for occluded objects in the scene has proven to have a negative impact on the quality of the effect. Layered depth images, and more recently deep G-Buffers, have proven that even with the inclusion of the first layer of occluded objects the quality and stability of screen-space effects can be improved. The applications of deep G-Buffers have so far been focused on global illumination techniques. This project aims to explore the benefits that deep G-Buffers can have on other common screen-space rendering effects.

The chosen rendering effects will be implemented in a DirectX 11 powered 3D rendering engine, using both single and multi-layer G-Buffers. The visual quality of the techniques will be compared and performance measurements will be collected. The results will then be analysed to determine if the chosen effects have benefitted from deep G-Buffers and at what additional performance cost.

It is expected that the deep G-Buffer modifications to the effects will have had a positive effect on the quality and stability at an acceptable performance overhead. Conclusions will be drawn from the results giving suggested future work and any possible improvements to the chosen implementations.

5. REFERENCES

Anon. 2009. Build: Deferred rendering. [blog]. 26 February. Available from: http://www.develop-online.net/tools-and-tech/build-deferred-rendering/0116368 [Accessed 20 October 2016]

Bavoil, L. and Myers, K. 2008. Order independent transparency with dual depth peeling. Technical report. NVIDIA. Available from: http://developer.download.nvidia.com/SDK/10/opengl/screenshots/samples/dual_depth_peeling.html [Accessed 5 October 2016]

Deering, M. et al. 1988. The triangle processor and normal vector shader: a VLSI system for high performance graphics. In: ACM SIGGRAPH Computer Graphics. 22(4): pp. 21-30.

Everitt, C. 2001. Interactive order-independent transparency. Technical report. NVIDIA. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.9286 [Accessed 5 October 2016]

Hoef, M. van de. and Zalmstra, B. 2011. Fast gather-based construction of stereoscopic images using reprojection. Utrecht University.

Magnusson, K. 2011. Lighting You Up in BATTLEFIELD 3. In: Game Developer Conference 2011. San Francisco CA, 28 February – 4 March. Available from: http://www.dice.se/news/lighting-battlefield-3/ [Accessed 20 October 2016]

Mara, M., McGuire, M. and Luebke, D. 2013. Lighting deep G-Buffers: single-pass, layered depth images with minimum separation applied to indirect illumination. Technical report. NVIDIA. Available from: http://graphics.cs.williams.edu/papers/DeepGBuffer13/ [Accessed 19 September 2016]

Mara, M., et al. 2016. Deep G-Buffers for stable global illumination approximation. In: Proceedings of High Performance Graphics 2016. Dublin, June 20-22. Available from: http://graphics.cs.williams.edu/papers/DeepGBuffer16/ [Accessed 19 September 2016]

Mark, R. W., McMillan, L. and Bishop, G. 1997. Post-rendering 3D warping. In: Proceedings of 1997 Symposium on Interactive 3D Graphics, Providence, RI. April 27-30. pp. 7-16.

McGuire, M., et al. 2011. The Alchemy screen-space ambient obscurance algorithm. In: Proceedings of ACM SIGGRAPH Symposium on High Performance Graphics. Vancouver, BC. August 5-7. pp. 25-32.

Myers, K. and Bavoil, L. 2007. Stencil routed A-buffer. In: ACM SIGGRAPH 2007 Sketches. Article 21.

Nehab, D., et al. 2007. Accelerating real-time shading with reverse reprojection caching. In: Graphics Hardware 2007 Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware. Eurographics. pp. 23-35.

Saito, T. and Takahashi, T. 1990. Comprehensible rendering of 3-D shapes. In: ACM SIGGRAPH Computer Graphics. 24(4): pp. 197-206.

Schulz, N. 2010. Bringing stereo to consoles. In: Game Developer Conference Europe 2010. Cologne, August 16-18.

Shade, J., et al. 1998. Layered depth images. In: Siggraph 1998 proceedings of the 25th annual conference on Computer graphics and interactive techniques. pp. 231-241.

Soler, C., Hoel, O. and Rochet, F. 2010. A deferred shading algorithm for real-time indirect illumination. In:

ACM SIGGRAPH 2010 Talks. Los Angeles, CA. July 26-30. ACM. pp. 18.

Sousa, T., Kasyan, N. and Schulz, N. 2011. Secrets of CryENGINE 3 graphics technology. In: SIGGRAPH 2011 Advances in Real-Time Rendering in 3D Graphics and Games Course. Vancouver, BC. August 7-11.

Suchow, J. W. and Alvarez, G. A. 2011. Motion silences awareness of visual change. In: Current Biology. 21(2), pp. 140-143.

Vardis, K., Papaioannou, G. and Gaitatzes, A. 2013. Multi-view ambient occlusion with importance sampling. In: I3D '13 Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. pp. 111-118.

Vosburgh, E. 2010. Unity 3 Feature Preview – Deferred Rendering. [blog]. 9 September. Available from: https://blogs.unity3d.com/2010/09/09/unity-3-feature-preview-deferred-rendering/ [Accessed 20 October 2016]

Wimmer, M. and Schedl, D. 2013. Simulating partial occlusion in post-processing depth-of-field methods. In: W. Engel, ed. GPU Pro 4: Advanced Rendering Techniques. CRC Press. 2013, pp. 187-200.