EG UK Theory and Practice of Computer Graphics (2010)
John Collomosse, Ian Grimstead (Editors)

Screen Space Re-Rendering for the Simulation of Concert Lighting†

Ian Stephenson‡[1] and Liam Scanlan[2]

[1]National Centre for Computer Animation,Bournemouth University,UK
[2]Arts University College, Bournemouth, UK

Abstract

The visualisation of modern concert lighting requires complex illumination models to be calculated at interactive frame rates. Each light may have 20 or more parameters which are all changing in real time. Here we present a technique which allows static scenes with rapidly changing illumination to be re-rendered quickly in graphics hardware, with support for arbitrary geometry, complex BRDFs, shadowing and volumetric fogging. The rendering algorithm is re-factored to allow changing lighting to be applied to scenes which are otherwise fully rendered, minimising the calculation required when lights are moved.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications

Modern concert lighting systems contain large numbers of "intelligent" lights. While the total number of lights used may be less than found in theatrical lighting, or even traditional concert lighting systems the complexity of each light makes controlling such a show a challenging task. Whereas traditional theatrical lamps allow control over their brightness, modern lights use the DMX [ANS04] serial protocol to control virtually every possible parameter of the light, including:

- Intensity
- Orientation
- Colour
- Cone Angle
- Focus
- Gobo Pattern
- Gobo Orientaton

For example the Varilte VL2500 Spot [Var04a] uses 22 control channels.

Programming a lighting desk to control these lights to produce a specific visual effect is a highly demanding task. It is made more challenging as it is rarely possible to obtain access to venues in advance of the shows. It is down to the lighting designers' skill to imagine what a lighting configuration will look like, with only limited time for adjustments once the equipment is installed. Even once the equipment is operational, it is likely that the stage will be required by the performers, rather than being available for lighting development.

1. Previous Work

A number of commercial systems exist which attempt to assist the lighting designer in visualising stage sets. These include "Capture","SunLite" and "ESP Vision", which connect a PC to the lighting desk and display a rendered image of the stage on screen. However these are based upon OpenGL style renders, and the results are generally of low quality. They require high end graphics hardware, and can only maintain interactive frame rates when relatively simple scene geometry is used.

We have previously tackled the problem of visualising theatrical lighting [Ste09] through the use of relight-

† This work was supported by Mark Cunniffe Lighting Design.
‡ e-mail: ian@dctsystems.co.uk

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG10/001-007

Figure 1: *Beauty Pass rendered offline in Angel [Ste04]*

ing techniques [DWT*02, Deb06] allowing real images of stage lighting to be rapidly captured, and re-combined using graphics hardware. Each light was represented by a single image, showing the scene as it would appear if that light was fully illuminated. These were then scaled and summed, using Apple's "Core Image" framework [App07] to perform these calculations on the GPU. However this is only practical for lights with a low degree of freedom. While this is appropriate for the fixed lights used in traditional theatrical productions, the high performance lights used at modern concerts are capable of producing far more lighting configurations than can be realistically recorded and stored.

The complex lighting effects required can be synthesised by off-line rendering, but the capabilities of real time graphics hardware is insufficient to reproduce them at the required speed. [DAG95] combined pre-rendered images to achieve real time results, in a similar way to our previous work. However to support "intelligent" lights by the simple linear recombination of images would require thousands of renders per light, which is clearly impractical.

Within the digital effects and animation industry more advanced relighting techniques have been developed which enable a compositor operating on only 2D images to make adjustments to the lighting and texturing of scenes after they have been rendered [GH00, PVL*05]. Rather than simply rendering a final image (beauty pass), surface parameters, such as the position and orientation of visible points in the scene are recorded. These can be used to quickly recalculate the illumination model of visible surfaces. However such systems cannot calculate accurate shadows, and it is difficult to support complex BRDFs.

In our implementation we will develop these techniques to show how interactive performance and realism can however be achieved by a combination of "baking" information into pre-rendered images, and re-factoring the rendering solution, allowing lights to be adjusted after most of the complexity of the scene has been pre-calculated.

2. Implementation

In high quality rendering the most time consuming tasks are the calculation of occlusion (hidden surfaces, and shadows), and the shading of surfaces (through the execution of potentially complex shaders). We currently ignore global illumination effects though they could be included within our system.

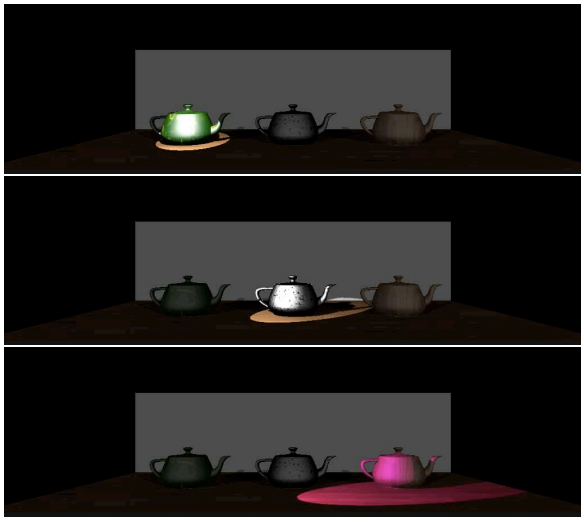

Figure 2: *Spotlights, Rendered by Masking Figure 1*

If we make the assumptions that:

- The camera does not move.
- the scene geometry is not going to change.
- the position (though not the orientation) of the light is fixed.
- the light is sufficiently far from the illuminated surfaces that its area is negligible.

then both occlusion and surface properties can be pre-calculated in any high quality off-line rendering system by rendering the scene illuminated by each light in turn, treating the light as a point source, as in figure 1.

This image (which is stored in the EXR format to support the high dynamic range) holds all the required information about the direct light paths from a single light to the camera, via each surface point. The complexity of surface geometry, texturing and shading is fully encoded within this pre-calculated image, so the interactive part of the simulation is independent of the original scene.

For this application the use of a point light beauty pass is more effective than recording surface parameters, as it requires less storage, less calculation at re-render time, is easier to generate, and allows the user to texture and shade the surface freely within their modelling and rendering package. Most importantly it includes shadowing information and compared to calculating *all* lighting at re-render time, the results will be of higher quality.

The primary limitations of the approach are the restrictions on movement which can be relaxed if necessary at the cost of additional pre-baking and storage. For example

Figure 3: *Including Textures Within the Image Mask to Create Gobo Effects*

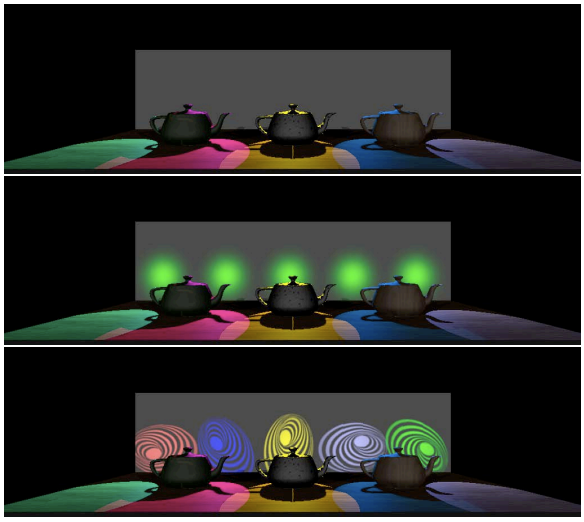

Figure 4: *Summing Multiple Re-Rendered Images to Support Multiple Lights*

a light which can change position (by moving along a track), can be rendered in multiple positions, and the appropriate pre-calculated image selected at final render time. Similarly multiple camera angles can be pre-calculated. Render times for these images may only be a few seconds (given a suitable rendering system), so it does not preclude lights being moved within an application. However once calculated the user will be able to interact with the scene with fully interactive response times.

2.1. Basic Relighting

The complex illumination patterns of real stage lights can be calculated by simply masking the pre-rendered image. In addition to the illumination image for each light, we render a single P-pass which holds the position in 3D space of each pixel. The position of the light, and camera is also known.

Apple's Core Image framework is essentially a hardware accelerated compositing system, which applies SIMD operations to every pixel in an image. Using the P-pass, for each light we can calculate the illumination vector from the light to each pixel. By normalising this vector and comparing it to the light's axis we can identify if the pixel is within the beam angle of the light, and use this (along with the light's brightness and colour) to scale the illumination pass, as shown in figure 2.

If in addition to the light's principal axis, a second orthogonal axis representing the rotation of the light is passed to the shader, this can be used to calculate 2d coordinates for the position of the pixel within the light's projected beam. These can be used to a perform simple texture lookup, allowing gobo[†] effects to be simulated, as shown in figure 3. Changing the orthogonal axis rotates the gobo.

An image is calculated independently for each light source (which has its own illumination pass, but shares a common P-pass), and these are summed together to produce the final lit image, as shown in figure 4. If the parameters of a light, such as cone angle, direction, brightness or colour change then only the scaling of its illumination pass need be recalculated. The new output image for the modified light can then be summed with the cached images previously calculated for the other lights, so only lights which are changing contribute to the calculations cost. As this is hardware accelerated it can be performed at interactive frame rates.

2.2. Volumetrics

While the appearance of most rendered scenes is primarily dependent on the shading of surfaces, many forms of concert lighting make extensive use of smoke and haze to create volumetric effects. Visible beams of light are formed by the scattering of light by airborne particles. Unlike surface lighting it is not practical to pre-calculate this effect, as illumination must be considered for the entire line of points between a visible surface point and the camera. However we can consider the smoke to be homogeneous - of equal density and texture at all points in the scene, unlike surfaces which vary greatly. We also ignore volumetric shadowing, though this could be incorporated through the use of shadow maps.

To calculate the volumetric lighting we must ray march

[†] A mask placed in front of a light to project a pattern

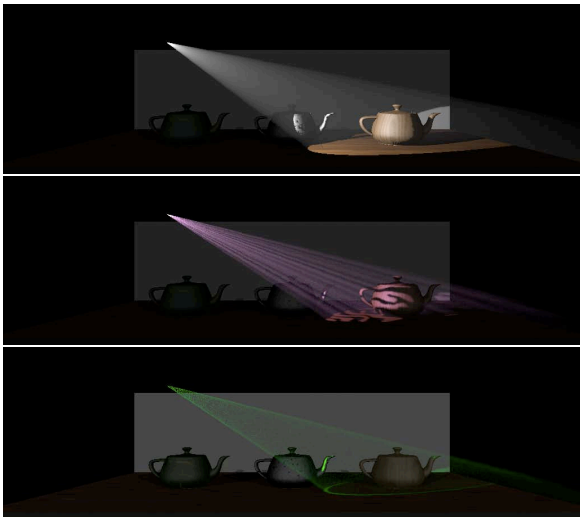

Figure 5: *Ray Marching to Create Volumetric Effects*

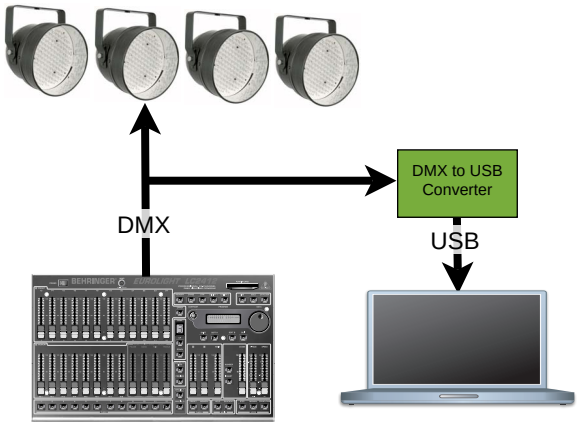

Figure 6: *Interfacing to a Typical Stage Lighting System*

from the surface to the camera. However once again this can be done on a per-pixel basis in Core Image. Ray marching would normally start at a surface and move towards the camera in small steps of a fixed size. At each step the fog density would be considered, and used to obscure the light from the surface. Any light falling at the current point would be considered to be scattered, and part of it would be added to the current light being carried towards the camera.

The limitations on the use of GLSL by Core Image make this slightly trickier than it would be in a more general programming language, as "if" statements, and data-dependent loops are not supported. However fixed loops are supported, so a a fixed number of steps from surface to camera can be used. This calculation is done immediately following the surface light calculation for each light, so for each light the resultant image includes both a volumetric and surface contribution. This means that if a light is adjusted, only its lighting contribution must be recalculated, rather than re-evaluating the volumetric effect of all lights.

To improve quality for a given number of steps, the ray from surface to camera is intersected with the light's cone of illumination. If there is no intersection, the same fixed number of samples must still be used, but if an intersection is found the samples can be moved into the illuminated area where they are most useful. Jittering is also used to improve image quality.

Both cone-intersection, and jittering calculation are difficult to implement in CI-GLSL, but allow the number of ray-march samples to be dramatically reduced. As few as two or four steps can be used for simple lights, while for lights with complex gobos (which increase the high frequency components in the lighting) ten steps is sufficient to produce complex volumetric illumination.

2.3. DMX Control

Stage lighting control systems can be immensely complex, with thousands of control channels — 20 or more channels for a single light is not uncommon. Modern lighting desks communicate with lights using the DMX [ANS04] standard: a digital serial protocol. Using a DMX to USB interface as in figure 6 the lighting simulation can be controlled from any standard lighting desk. Control channels of the real lights are mapped to parameters of the simulated lights. Specifically the Vari-lite VL2500 Spot [Var04a] and Wash [Var04b] Luminaires are currently modelled.

Scenes can be programmed on the desk, and previewed on screen. As the simulation operates at interactive speeds, the operator can work as if he was controlling a real lighting rig. Once the lighting director is satisfied with the images, the desk can be connected directly to the lighting rig and the programmed scenes used without modification.

3. Performance

The system was developed and tested using a MacBook Pro (2.4GHz Core 2 Duo) with an NVidia GeForce 8600M GT 256Mb. The CI framework allocates work to both the CPU and the GPU depending on their capabilities, but in this software and hardware configuration, the GPU was usually fully loaded, while the CPU was essentially unused. All of the images shown (other than the initial beauty pass) were rendered interactively on this system at a resolution of 1024x300.

The performance of the system is independent of scene geometry and surface lighting models, allowing the curved surfaces and complex shading models used in the scene to be handled without penalty.

The images for each each light must be summed to produce the final output, which in the worst case would take a time proportional to the number of lights. However the im-

	Per Sample Cost	Surface llumination Cost	March Steps	Volumentric Illumination Cost	Total Cost
Simple Light	1	1	4	4	5
Complex Light	1.5	1.5	10	15	16.5

Table 1: *Sampling Cost*

Figure 7: *Multiple Lights with Gobo and Volumetric Effects, Re-Rendered Interactivly on the GPU*

ages are summed in a binary tree, and the intermediate images cached by the CI framework. If a single light is moved, then only log2(n) summations need to be performed. The total number of lights within the scene is therefore not a limitation on performance, provided there is sufficient memory to cache the intermediate images. Even when this limit is exceeded the caching algorithm can often maintain good performance when only a subset of the lights are changing.

The limit on performance is the GPU's ability to calculate the image for each light that changes. This calculation includes both the surface illumination and the volumetric illumination. The inclusion of gobos and other complex effects in a light source increases the complexity of surface illumination only slightly. However if the light source contains a lot of high frequency detail then the number of ray march steps must be increased. As a result lights with gobos are around three to fours times more costly than wash lights, as calculated in table 1.

The 8600M GT graphics card is capable of animating the test scene shown in figure 7, at interactive frame rates. This scene contains 5 simulated VL2500S spot lights with 2 gobos each, and 5 VL2500W wash lights at interactive frame rates. All lights can be updated simultaneously without noticeable delay. Up to 20 VL2500S lights can be animated on this hardware before the delay becomes significant.

Figure 8 shows 10 simulated video projectors, which support all of the features of the VL2500S, and were able to project both Quicktime movies, and camera input into the scene without significant lag. This compares favourably to commercial systems running on much more substantial hardware.

The 8600M GT is a modest card in almost all respects, its performance being one of the lowest in the NVidia 8000 range, which has itself been superseded by the NVidia 9000 series. It is also reported that the ATI Radeon series out performs NVidia hardware for Core Image compositing operations. That the 8600M GT supports the test scene comfortably indicates that there is sufficiently powerful hardware available to handle large concert productions with acceptable performance.

4. Conclusions

The approach presented here allows the interactive relighting of complex scenes by refactoring the rendering system to calculate the incident illumination at each pixel *after* all other surface calculations have been performed. The image is "re-rendered" in screen space, using pre-calculated images, so that the calculations required to simulate changing lighting conditions, including accurate shadows, is totally independent of the scene geometry. Surfaces may include complex procedural texturing and shading. The approach is extended to support accurate volumetrics.

This technique is particularly suited to the simulation of concert lighting, where "intelligent" lights project complex and constantly changing illumination patterns into the scene. Lighting designs can be visualised in real time under direct control of a standard lighting desk.

The responsiveness, and image quality produced by the prototype system running on moderate hardware compare favourably with commercial systems based on more conventional, hardware rendering techniques.

References

[ANS04] ANSI: *E1.11 - USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories*, 2004. 1, 4

[App07] APPLE INC: *Core Image Programming Guide*, 2007. 2

[DAG95] DORSEY J., ARVO J., GREENBERG D.: Interactive design of complex time-dependent lighting. *IEEE Comput. Graph. Appl. 15*, 2 (1995), 26–36. 2

[Deb06] DEBEVEC P.: Virtual cinematography: Relighting through computation. *Computer 39*, 8 (2006), 57–65. 2

[DWT*02] DEBEVEC P., WENGER A., TCHOU C., GARDNER A., WAESE J., HAWKINS T.: A lighting reproduction approach to live-action compositing. In *ACM Transactions on Graphics (SIGGRAPH 2002)* (2002), vol. 21(3), pp. 547–556. 2

[GH00] GERSHBEIN R., HANRAHAN P.: A fast relighting engine for interactive cinematic lighting design. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 353–358. 2

[PVL*05] PELLACINI F., VIDIMČE K., LEFOHN A., MOHR A., LEONE M., WARREN J.: Lpics: a hybrid hardware-accelerated relighting engine for computer cinematography. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 464–470. 2

[Ste04] STEPHENSON I. (Ed.): *Production Rendering*. SpringerVerlag, 2004. 2

[Ste09] STEPHENSON I.: Digital relighting for stage use. In *EG UK Theory and Practice of Computer Graphics 2009*, Tang W., Collomosse J., (Eds.). Eurographics Association, Cardiff, UK, July 2009. 1

[Var04a] VARI-LITE LIMITED: *VL2500 Spot Luminaire Users Manual*, 2004. 1, 4

[Var04b] VARI-LITE LIMITED: *VL2500 Wash Luminaire Users Manual*, 2004. 4

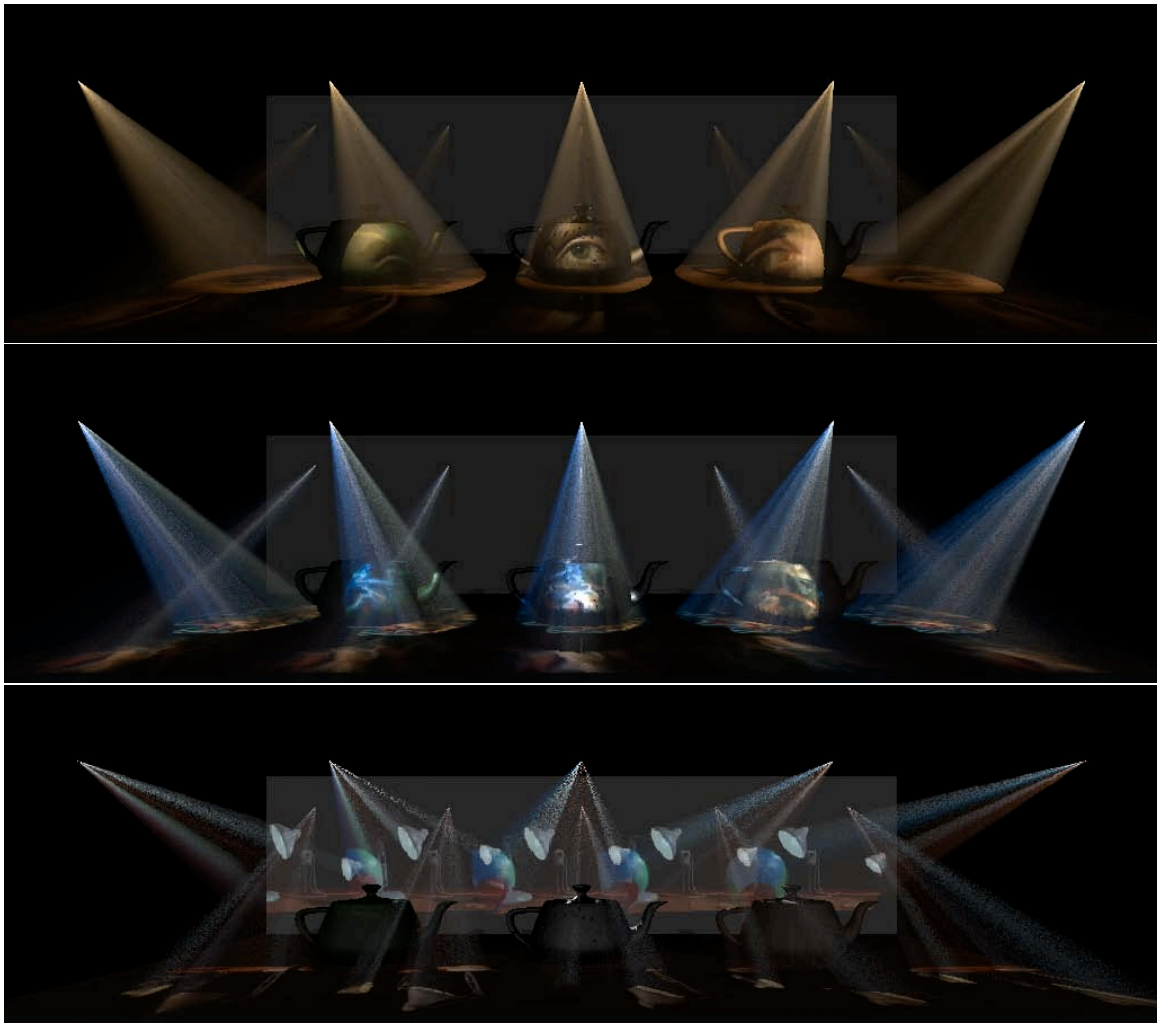

Figure 8: *Animating the Gobo Images to Support Video Projection*