UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

FABIO DE ALMEIDA AQUOTTE

Real-time Diffuse Indirect Illumination with Virtual Light Meshes

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Marcelo Walter

Porto Alegre
May 2017

CIP — CATALOGING-IN-PUBLICATION

Aquotte, Fabio de Almeida

Real-time Diffuse Indirect Illumination with Virtual Light Meshes / Fabio de Almeida Aquotte. – Porto Alegre: PPGC da UFRGS, 2017.

60 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2017. Advisor: Marcelo Walter.

1. 3D. 2. Graphics. 3. Rendering. 4. Real-time. 5. Global Illumination. I. Walter, Marcelo. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Rui Vicente Oppermann
Vice-Reitora: Prof[a]. Jane Fraga Tutikian
Pró-Reitor de Pós-Graduação: Prof. Celso Giannetti Loureiro Chaves
Diretora do Instituto de Informática: Prof[a]. Carla Maria Dal Sasso Freitas
Coordenador do PPGC: Prof. João Comba
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“In the beginning the Universe was created. This has made a lot of people very angry and been widely regarded as a bad move.”

— Douglas Adams

ACKNOWLEDGEMENTS

First I wish to thank my advisor, Marcelo Walter, for the opportunity to work with him and the kindness, guidance, and generosity he has given me through this two-year journey. I would also like to thank all other professors with whom I have interacted during this time. I've had nothing but positive constructive experiences.

I thank the Federal University of Rio Grande do Sul (UFRGS) and the Institute of Informatics (INF) for the infrastructure they have provided to aid in this research and the National Counsel of Technological and Scientific Development (CNPq) for the funding that has allowed me to focus full time on research work. I also thank Aquiris Game Studio for their generosity in providing their time to discuss this research and provide valuable feedback.

Finally, I extend the greatest gratitude to my parents Eldo and Darci, to my sister Sheli and to my girlfriend Lilian for their love and continual support of my endeavors.

ABSTRACT

Indirect illumination on a rendered scene can add a great deal to its visual quality, but it is also a costly operation. Therefore, a lot of research targets how to render indirect illumination in real-time. While powerful techniques for real-time indirect illumination currently exist, they provide only coarse-grained artistic control over the trade-off between quality and speed. We propose a *Virtual Light Mesh* to compute the scene's diffuse indirect illumination, inspired by the use of other current auxiliary meshes such as Navigation Meshes and Collision Meshes. A Virtual Light Mesh (VLM) is a simplified mesh of polygonal lights used to approximate the light bounced by the real geometry. Together with the VLM, we design an acceleration data structure for efficient indirect illumination performance with a complex VLM. The use of VLM presents some positive properties: greater artistic control of the indirect illumination characteristics; the possibility of integration with existing techniques such as skeletal animation and procedural generation; and simple integration into existing asset production tools and pipelines. Our experimental results show that artist controlled indirect illumination is a viable alternative to existing methods.

Keywords: 3D. Graphics. Rendering. Real-time. Global Illumination.

Iluminação Indireta Difusa em Tempo Real usando Malhas de Luzes Virtuais

RESUMO

A iluminação indireta é capaz de elevar consideravelmente a qualidade visual de cenas renderizadas, mas é também uma operação custosa. Por este motivo, há muito esforço de pesquisa voltado para a renderização de iluminação indireta em tempo real. Apesar de atualmente existirem técnicas poderosas para a iluminação indireta em tempo real, elas fornecem ao artista apenas um controle grosseiro do equilíbrio entre qualidade e desempenho. Nós propomos uma *Malha de Luzes Virtuais* para calcular a iluminação indireta difusa numa cena, inspirados pelo uso de outras malhas auxiliares, como Malhas de Navegação e Malhas de Colisão. Uma Malha de Luzes Virtuais (MLV) é uma malha simplificada de luzes poligonais usadas para aproximar a luz refletida pela geometria real. Juntamente com a MLV, nós projetamos uma estrutura de dados de aceleração para atingir um desempenho eficiente com iluminação indireta usando uma MLV complexa. O uso da MLV apresenta algumas características vantajosas: maior controle artístico dos atributos da iluminação indireta; a possibilidade de integração com técnicas existentes como animação esquelética e geração procedural; e integração simples com ferramentas e processos de produção de arte existentes. Nossos resultados experimentais mostram que a iluminação indireta controlada por artistas é uma alternativa viável a métodos existentes.

Palavras-chave: 3D. Gráficos. Renderização. Tempo Real. Iluminação Global.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

PRT Precomputed Radiance Transfer

VLM Virtual Light Mesh

CONTENTS

1 INTRODUCTION

Computer Graphics research always strives to improve the visual quality of synthesized images. Global illumination effects such as indirect illumination, where the light that illuminates a given point reflects one or more times through the scene before reaching that point, can significantly improve the visual quality of a rendered scene. While commonplace in off-line rendering, this effect is still quite challenging to achieve in interactive applications. There is a substantial increase in complexity required to compute light that travels through arbitrarily complex paths and that must be gathered from many different directions at each shaded fragment.

There are currently many techniques to tackle the problem of real-time indirect illumination, using different strategies and being subject to particular limitations. These methods focus on achieving physical accuracy as much as possible within the confines of real-time constraints. This focus leads to very coarse grained control of the quality versus speed trade-off. There are usually only a few parameters whose values can be tuned to configure the resolution, and consequently the performance cost, used across the entire scene to evaluate the indirect illumination.

In this thesis, we present a technique that delivers a greater amount of artistic control for computing real-time diffuse indirect illumination. According to production needs and artistic vision, artists can fine-tune the quality of indirect illumination for any portion of the scene. We also provide control for local changes in indirect illumination characteristics to better suit the needs of the application. Additionally, our technique provides this control in a way that is both familiar and easy to integrate with existing tools and asset pipelines.

Our inspiration comes from the current usage of "non-visual" meshes to control different systems in game engines, such as Navigation Meshes (SNOOK, 2000) and Collision Meshes (UNREAL, 2017). Similarly to techniques based on Virtual Lights, we apply artist authored Virtual Light Meshes (VLMs), composed of polygonal area light sources that are used to approximate the diffuse light that would bounce off the real geometry. Our use of polygonal area lights lets us use a smaller number of virtual lights and provide the above-cited advantages.

1.1 Overview

In Chapter 2 we provide a review of existing real-time indirect illumination techniques. The detailed description of our VLM technique is presented in Chapter 3. Chapter 4 describes an acceleration data structure that makes our technique applicable to scenes with complex meshes. Our results are shown and discussed in Chapter 5. Finally, our conclusions and ideas for future work on this technique are presented in Chapter 6.

2 BACKGROUND AND RELATED WORK

As our work is focused on real-time indirect illumination, we present in Section 2.1 a short review of the background on this topic, followed by a review of existing competing techniques in Sections 2.2 through 2.7. Finally, in Section 2.8, we present a discussion of the limitations of the presented methods and how our work relates to them.

2.1 Indirect Illumination

In 3D Rendering, we seek to synthesize images that represent a view through a virtual camera into a scene described by a 3D model. To do that, we must simulate the way light travels from the scene's light sources through the scene and into the camera. As precisely simulating the real world behavior of light transport is computationally intractable, we must choose simplified models of light transport to evaluate.

One widely used model is given by the Rendering Equation introduced by Kajiya (1986). It models the amount of light leaving a given point in a given direction as a function of the light being emitted by the point plus the incoming light being reflected by that point. It is given by

$$L_o(\vec{x}, \vec{\omega_o}) = L_e(\vec{x}, \vec{\omega_o}) + \int_\Omega f_r(\vec{x}, \vec{\omega_i}, \vec{\omega_o})\, L_i(\vec{x}, \vec{\omega_i})\, (\vec{\omega_i} \cdot \vec{n})\, d\vec{\omega_i} \tag{2.1}$$

where $\vec{x}$ is the point position, $\vec{n}$ is the point surface normal, $\vec{\omega_o}$ is the outgoing direction, L_o is the amount of outgoing light, L_e is the amount of emitted light, Ω is the hemisphere on the point centered around its normal, f_r is a function that describes the proportion of incoming light at the position that is reflected in the outgoing direction, and L_i is the amount of incoming light.

As computing the solution of light transport with the Rendering Equation is quite costly, given that it involves finding its value at every surface point in the scene, further simplifications are usually used. The main one is the separation of light transport into two types: local illumination and global illumination. In local illumination, the only light that is taken into consideration when illuminating a surface point is the light that comes directly from the light source. It's called local as it only depends on the local characteristics of the surface point, the light sources and the camera. For this reason, it is far cheaper to compute and is widely used in real-time rendering.

Global illumination, on the other hand, comprises the illumination that takes into account the entirety of the scene and how light interacts with it before reaching a given surface point. In real-time rendering, specific global illumination effects, such as shadows, reflections, ambient occlusion and indirect illumination, are usually individually added onto local illumination with specific techniques.

Indirect illumination is the particular global illumination effect where light is reflected by one or more surfaces before reaching the illuminated surface point. It is responsible for the illumination of surfaces that are not directly reached by light from the light sources and color bleed effects, where the color of a surface tints nearby objects due to the light being reflected off of it.

2.2 Light Transport Simulation Methods

The traditional approach to computing indirect illumination in offline rendering is by simulating the way light travels through the scene according to the rendering equation proposed by Kajiya.

Illumination in a scene can be computed by numerically solving the rendering equation using Monte Carlo integration techniques (KAJIYA, 1986). This method is called Path Tracing and is accomplished by sampling a large number of light paths through the scene, evaluating the rendering function for each light path and averaging their contribution. The accumulation of samples causes the computed illumination value to converge toward the correct solution. Sampling a light path is done through ray tracing: rays are cast from the camera into the scene and recursively reflected whenever they intersect the rendered geometry until they reach a light source.

Simply casting rays in random directions leads to a lot of wasted computation for those rays that generate paths with very little contribution or even those that fail to reach a light source at all. Importance sampling can be used to generate rays with a greater probability in directions that are more likely to produce significant lighting contributions, by taking into account the reflexive characteristics of the geometry surface at each intersection point.

To increase the likelihood of generated rays reaching the light source, Bidirectional Path Tracing (LAFORTUNE; WILLEMS, 1993; VEACH; GUIBAS, 1995) casts rays both from the camera and from the light sources into the scene up to a bound reflection depth and then tries to directly connect the camera originated paths with the light

originated ones. Metropolis Light Transport (VEACH; GUIBAS, 1997) tries to generate new useful paths from previously found ones by perturbing their trajectories.

Introduced by Jensen (1996), Photon Mapping divides the illumination computation into two phases. In the first phase, photons are emitted from the light sources in the scene and deposited on the geometry surface points wherever they bounce. This is done by casting and recursively reflecting rays from the light sources and, for each intersection point, storing a photon in the photon map. The second phase computes the illumination at each point by estimating the photon density around that point. To do that, it queries the photon map for the photons nearest to the point and aggregates their contributions.

A lot of research effort has been spent in trying to bring these light transport simulation techniques to interactive applications. Using the power of GPUs for tracing rays (ZHOU et al., 2008; AILA; LAINE, 2009; PARKER et al., 2010) can greatly increase ray casting rates. Rays can also be grouped and rearranged (BOULOS et al., 2007) to improve parallel computation coherency and performance. Methods that combine multiple techniques to improve performance have also been proposed (WANG et al., 2009). Finally, high quality real-time filters (GASTAL; OLIVEIRA, 2012) can be used to process noisy light transport simulation results generated quickly with small sample counts to produce noise free results.

2.3 Precomputed Light Transport Information

For scenes where all of the geometry is static, it is possible to precompute the parameters of light transport that are due to the structure of that geometry and use that information at runtime to approximately compute indirect illumination for dynamic lights. This is done by precomputing those parameters at many different probe points in space and using those values to interpolate for the remainder of the scene.

Precomputed Radiance Transfer (SLOAN; KAUTZ; SNYDER, 2002) stores at each probe location two functions: a lighting function that determines the incoming intensity of light at the probe point and a transfer function that determines how light behaves at that point. To allow for fast and compact storage, as a large number of probes must be stored, the functions are represented as spherical harmonics. This representation consists of storing a set of coefficients that are used to combine the equivalent set of basis functions to reconstruct an approximation of the original function. In addition to spherical harmonics, many other forms of representing the probe functions have been proposed,

such as wavelets (LIU et al., 2004; WANG; TRAN; LUEBKE, 2004; WANG; TRAN; LUEBKE, 2006), Gaussians (GREEN et al., 2006; GREEN; KAUTZ; DURAND, 2007), radial basis functions (TSAI; SHIH, 2006), tensors (SUN et al., 2007; SUN et al., 2008) and piecewise constant functions (XU et al., 2008).

Precomputed Light Field Probes (MCGUIRE et al., 2017) stores light field and visibility information at each probe location as an octahedral projection (CIGOLLE et al., 2014). These values are then used during rendering to compute an approximate ray tracing solution to indirect illumination.

2.4 Reflective Shadow Maps

Traditional Shadow Maps (WILLIAMS, 1978) store visibility information for the scene from the light source's point of view. Reflective Shadow Maps (DACHSBACHER; STAMMINGER, 2005) extend this representation to compute indirect illumination. In addition to the visibility information, represented by the depth value, a reflective shadow map also stores information about the reflection characteristics of the surface and about the intensity of light reaching it. This information effectively represents how the light from that source will be reflected by the scene.

During rendering, these values are used to compute the incoming light at each point. The incoming light is aggregated from each point in the reflective shadow map, using the stored information to calculate its final contribution. To achieve interactive frame rates, indirect illumination is computed at a lower resolution and interpolated for pixels at full resolution where the interpolated value is acceptable.

2.5 Voxel Cone Tracing

To simplify the computation of light bounces, Voxel Cone Tracing (CRASSIN et al., 2011) generates a simplified voxelized representation of the scene geometry that is stored in a sparse octree. This representation is then used to accumulate approximate light bouncing through the scene and finally compute the indirect illumination.

After generating the voxel octree representation, the method computes the incoming light intensity from each light source at the octree's leaves and stores them. Then, the light intensity values are filtered up into the higher levels of the octree, aggregating them

over bigger volumes. The final indirect illumination value evaluation at a given surface point is done by tracing cones from the point into the octree, aggregating the illumination of the volume encompassed by those cones.

To increase efficiency, the octree works as a MipMap Pyramid (WILLIAMS, 1983) during cone tracing. As the radius of aggregation increases along the cone, the lookups into the octree are done at the level corresponding to that radius. That way, each lookup accesses the precomputed aggregation that approximates the region covered by that slice of the cone.

2.6 Screen-space Methods

Rendering techniques that operate in screen-space have the advantage of their performance being independent of the complexity of the scene's geometry and depend only on the resolution of the rendered image.

For indirect illumination, the technique proposed by Ritschel, Grosch and Seidel (2009) can approximate indirect illumination in screen-space by rendering the scene to a G-Buffer and computing the final resulting illumination using its values. Instead of containing the resulting color as a regular render buffer, the g-buffer will store scene attributes such as surface color, position and normal at each pixel, which provides the screen-space technique with rich information about the scene.

To evaluate indirect illumination using the g-buffer data, the technique places small patches at each pixel positioned at the surface position and oriented according to its normal. These patches are then used to determine how light bounces between the pixels and these values are aggregated to arrive at the final illumination value for each pixel.

As the g-buffer stores information about a single surface point at each pixel, the indirect illumination is unable to take into account light bounced by points that are behind the visible surface. To circumvent this limit, techniques to obtain multiple layers of g-buffer information must be used. Depth peeling (EVERITT, 2001) addresses this issue by using multiple render passes and at each pass excluding surfaces that have been rendered in the previous passes, thus revealing deeper layers of the scene. Deep G-Buffers (MARA et al., 2016), on the other hand, use 3D G-Buffers capable of storing values for multiple surfaces at each pixel and separates the scene into multiple layers that are rendered to each slice of the Deep G-Buffer.

Additionally, as surfaces that reside outside the camera frustum are never present

in screen-space, their contribution can't be considered with such methods.

2.7 Virtual Lights

Indirect illumination may also be approximated through the use of virtual lights that simulate the light being bounced by the scene surfaces. The virtual lights are created to emit approximately the same amount of light that would be reflected by each region of the scene and are used as normal light sources to evaluate the final illumination.

Adapted from physics and engineering methods, Radiosity (GORAL et al., 1984) computes indirect diffuse illumination by dividing a scene's surfaces into small patches and calculating the view factors between these patches. The view factors encode how well the patches are able to see each other and, consequently, how well light will be transmited between them. These view factors form a system of rendering equations that can be solved to compute the final illumination across all surfaces.

Instant Radiosity (KELLER, 1997) builds on the idea of Radiosity by generating a large number of virtual point lights by tracing rays from the real light sources and placing them on the intersection points with the geometry surface. The large number of virtual lights needed to eliminate artifacts due to the approximation of surface reflections with point lights can penalize the performance of instant radiosity. To alleviate this issue, Light Cuts (WALTER et al., 2005) store the generated virtual lights in a tree data structure that can be simplified to reduce the number of virtual lights. This is done by finding sub-trees whose virtual lights can be combined into a single one performing this substitution, cutting the tree.

To reduce the cost of indirect illumination shadows incurred by using classic shadow maps for virtual lights, Imperfect Shadow Maps introduced by Ritschel et al. (2008) are low-resolution shadow maps generated from a simplified point-based representation of the scene. As this sparse representation may leave holes in the resulting shadow map, it must be filtered through a process of recursive down sampling by averaging valid values and up sampling by interpolating invalid values using the coarser level values.

2.8 Discussion

As presented in this chapter, existing real-time indirect illumination techniques focus on producing fast and physically accurate results for the effect but do not provide any mechanisms for arbitrary artistic manipulation of the end results. Our work seeks to explore this alternative approach to control indirect illumination through more powerful artistic input.

3 VIRTUAL LIGHT MESHES

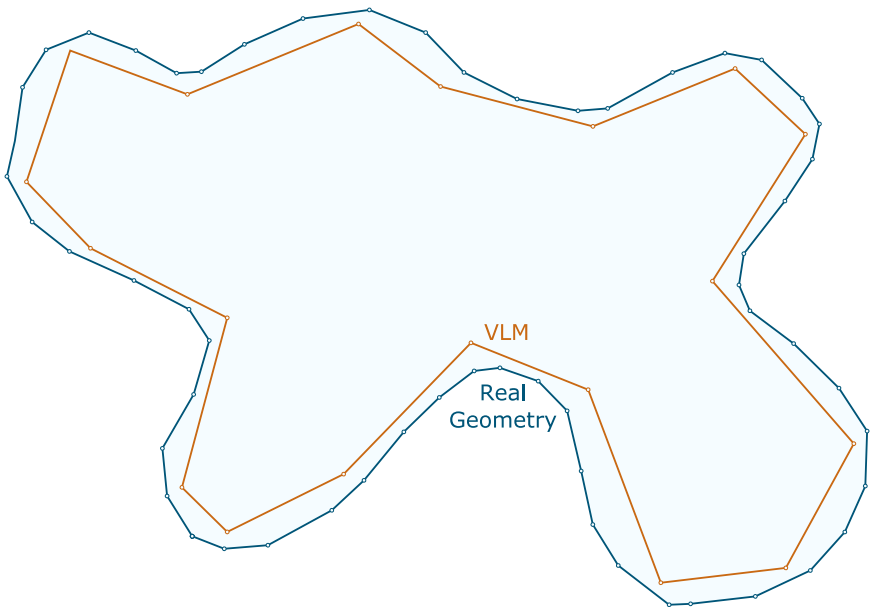
While physically-based simulation of indirect illumination with minimal manual input can generate realistic and physically accurate results, in some applications such as games and movies, artists will occasionally want to manipulate their results to achieve a certain look regardless of the correctness of those results.

To achieve that, we have designed a method that aims to evaluate real-time indirect illumination effect driven by granular artistic input. To increase the ease of use of that artistic input, we have based our method's control input on polygonal meshes, allowing artists to manipulate the results using a form with which they are already familiar and that can be integrated into their existing workflow.

3.1 Description

The Virtual Light Mesh is an artist authored mesh that is used to compute a scene's single-bounce diffuse indirect illumination in real-time. It represents a possibly simplified version of the visible geometry whose primitives are virtual polygonal lights used to simulate the light being reflected off of the visible geometry, as seen in Figure 3.1.

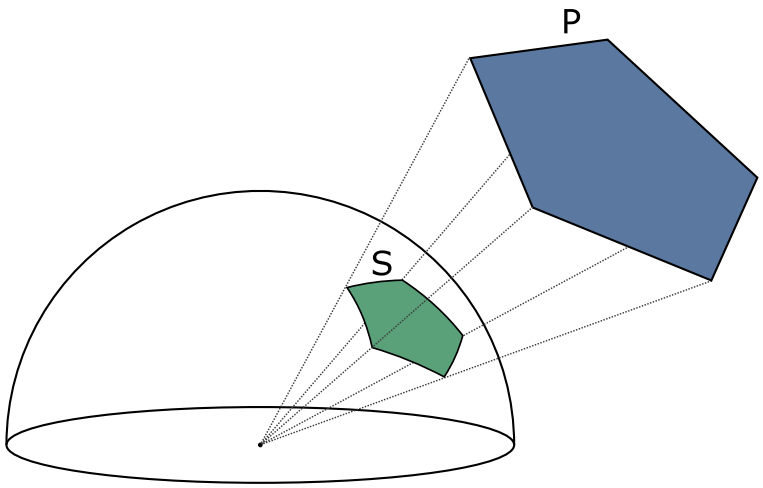
Figure 3.1: The VLM (orange line) corresponding to the real visible geometry (blue line)



Source: original image.

Each primitive in the VLM stores the positions of the virtual light's vertices and the light color. The light color value is used to influence the characteristics of the reflected

Figure 3.2: Polygon P projected onto the spherical polygon S



Source: original image.

light and is stored as a high dynamic range color value, that is, a color value whose components can be arbitrarily large. This makes it possible to use the light color to influence both the reflection hue and its intensity. While the method allows the use of any polygonal light sources, we implement triangular light sources as triangular meshes are most commonly used in rendering applications.

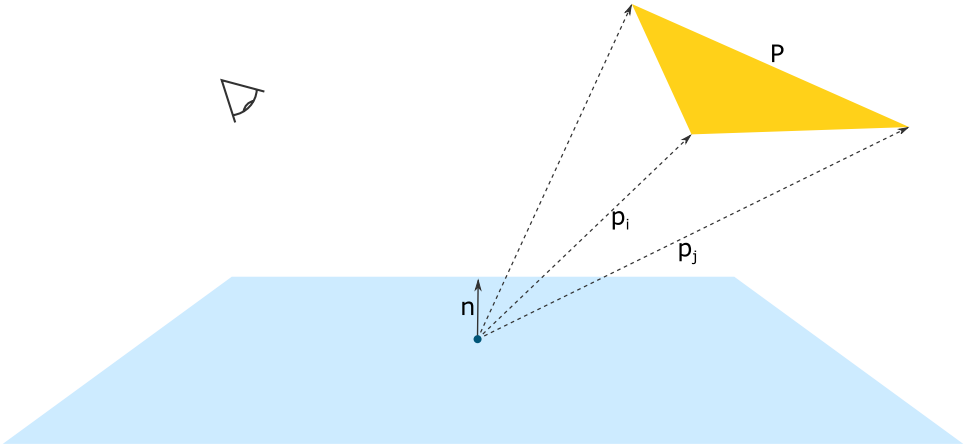
3.2 Diffuse Polygonal Light Sources

As the VLM consists of a mesh of polygonal light sources, in order to be able to simulate the effect of indirect illumination using them, we must be able to compute the illumination contributed from each of those polygonal light sources to any of the surface points being rendered.

Evaluating the illumination from a general area light still is a very costly operation, due to the significant difficulty of computing the integration of the arbitrary spherical distribution that represents the way the surface reflects light over the arbitrary shape of the light source projected onto the hemisphere around the surface point. However, if we sufficiently restrict the surface light reflection distribution and the form of the area light's shape, it is possible to derive efficient solutions.

As we are only interested in computing diffuse illumination from polygonal light sources, we can use these restrictions. We use the clamped cosine spherical distribution to obtain a perfect diffuse reflection from the surface and the spherical polygon projected

Figure 3.3: Polygonal light source P (yellow) with corresponding vectors $\vec{p}$ (dotted arrows) illuminating a given surface point (blue) with its normal $\vec{n}$ (solid arrow)



Source: original image.

from our light source polygon onto the hemisphere, as shown in Figure 3.2. With these parameters, Baum, Rushmeier and Winget (1989) derived an analytical solution for the incoming light intensity at a given surface point due to the illumination from the polygonal light source, given by

$$E(P) = \frac{1}{2\pi} \sum_{\vec{p_i},\vec{p_j}} \vec{n} \cdot \left(\arccos(\vec{p_i} \cdot \vec{p_j}) \frac{\vec{p_i} \times \vec{p_j}}{\|\vec{p_i} \times \vec{p_j}\|} \right), \tag{3.1}$$

where $E(P)$ is the incoming light received from the polygonal light P, $\vec{p_i}$ and $\vec{p_j}$ are vectors from the surface point to adjacent vertices of P, $\vec{n}$ is the normal at the surface point being shaded, $\cdot$ is the dot product and $\times$ is the cross product, as can be seen in Figure 3.3.

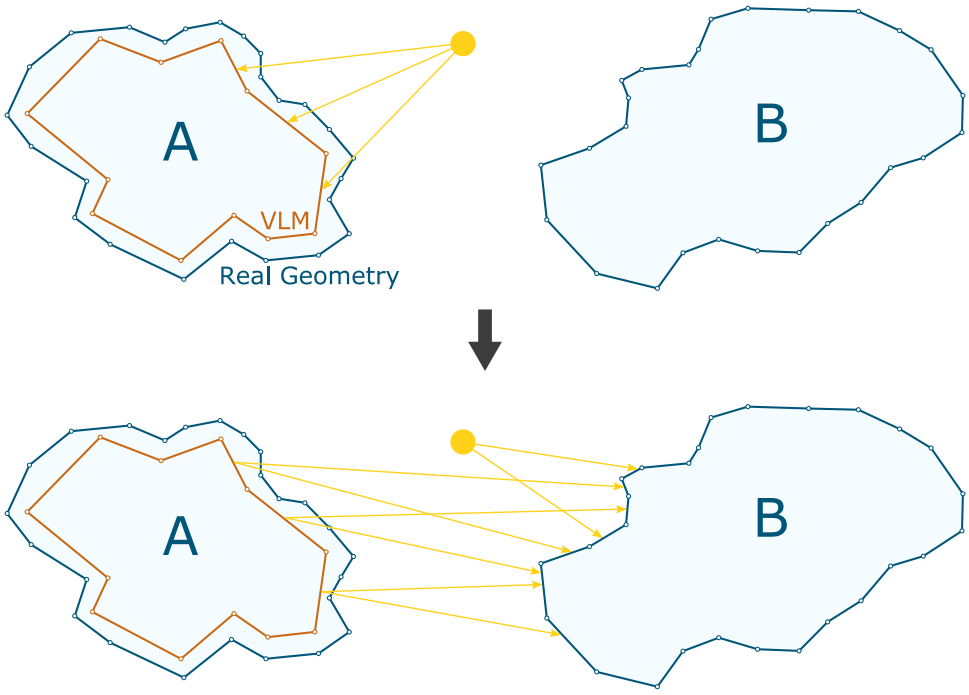
One important note is that, as the solution given in Equation 3.1 assumes a polygonal light that is entirely visible on the hemisphere centered around the surface point normal, we must clip the polygonal light source at the hemisphere's horizon whenever necessary.

3.3 Indirect Illumination with VLMs

We take advantage of the fact that diffuse indirect illumination is a low-frequency phenomenon to allow us to approximate the light bounced by the high-quality real geometry using the simplified geometry of the VLM.

To make the VLM geometry "reflect" light, we use a two-phase rendering process

Figure 3.4: Two-phase shading process for indirect illumination with VLM: in the first phase (top), the intensity of the virtual lights (orange) is calculated according to the real light (yellow); in the second phase (bottom), the visible geometry (blue) is illuminated with both real and virtual lights



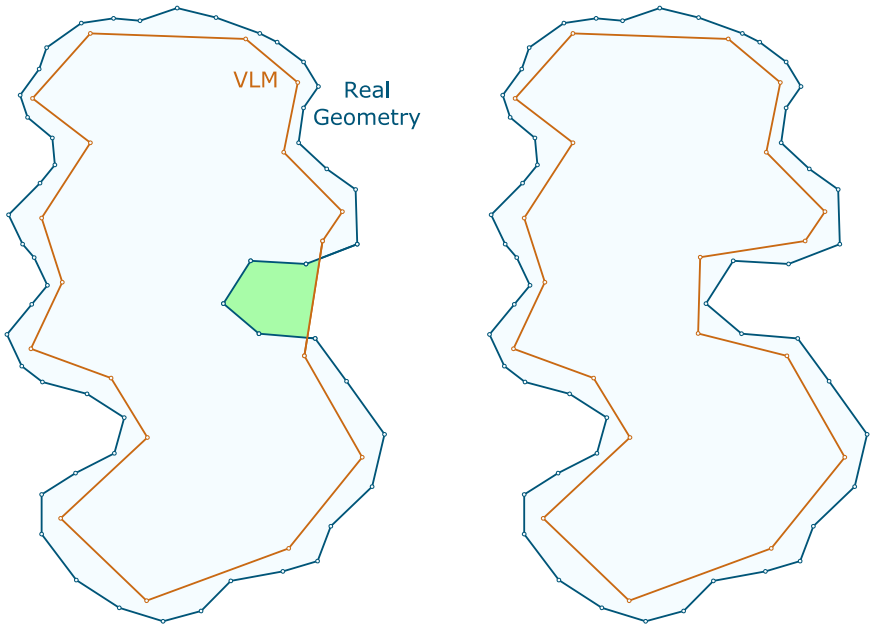
Source: original image.

that is similar to Virtual Point Light based methods. For each rendered frame, in the first phase, we use the scene's real lights to compute the amount of incoming light at each virtual light. This is done by calculating the light intensity at the virtual light due to all of the real lights and storing this value for each virtual light in the VLM. This computation is done using the existing illumination model for the real light sources, using the formulation for directional, spotlight or point light sources.

With that light intensity value, we can determine the light intensity that each virtual light must emit to approximate the diffuse reflection of incoming light. In the second phase, we use both the real lights and the virtual lights to shade the real geometry, obtaining the final result. Figure 3.4 illustrates the two-phase process.

This lets us compute a single bounce of diffuse indirect illumination reflected off the VLM geometry with its reflection characteristics.

Figure 3.5: A VLM (orange) that is placed in front of the visible geometry (blue), as shown on the left, will generate incorrect indirect illumination results inside the region marked in green, whereas restructuring the VLM so that it always remains behind the real geometry, as shown on the right, eliminates the issue



Source: original image.

3.4 Restrictions

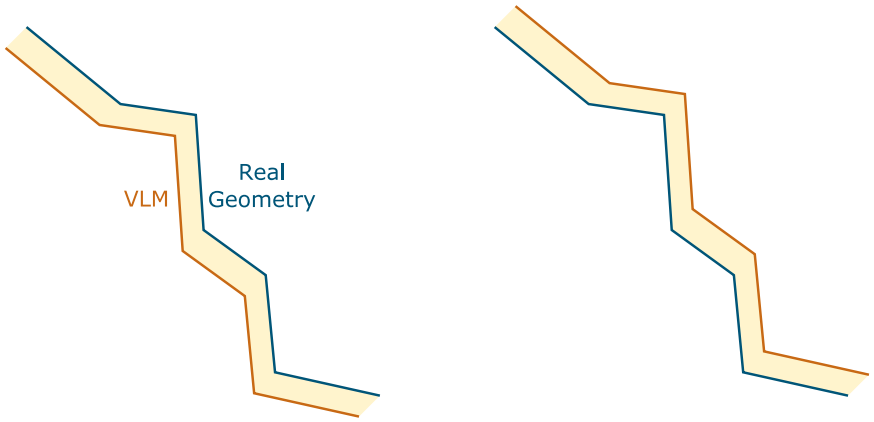
The nature of the VLM geometry's role in simulating the light reflected by the visible geometry places some restrictions on their relative positioning. Given that a particular point on the surface of the visible geometry should never be illuminated by the equivalent region in the virtual light geometry, each region in the virtual geometry should be placed behind the equivalent region in the real one, as shown in Figure 3.5.

While adhering to this restriction is not problematic in most objects, it can cause trouble with some types of geometry. In particular, two types of geometry are problematic: objects composed of two-sided primitives and thin curved objects.

For objects made out of two-sided primitives, as they do not have an internal volume, it is impossible to place the virtual geometry in a position that sits behind both sides of the primitive, as shown in Figure 3.6. In thin curved objects, as the simplified virtual geometry may not be able to closely match the curvature of the visible geometry, they may intersect causing portions of the virtual geometry to reside outside of the object, as seen in Figure 3.7.

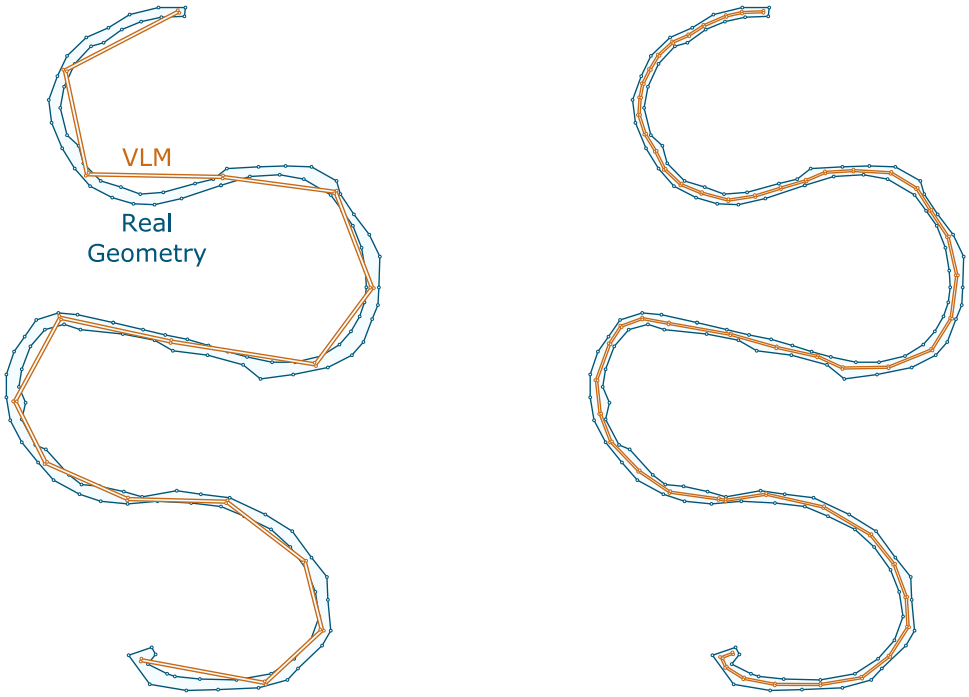
To solve this issue, we allow the virtual geometry to be associated with the equiv-

Figure 3.6: With two-sided visible geometry (blue), the virtual light (orange) that corresponds to the reflection off one visible side will incorrectly illuminate the other visible side



Source: original image.

Figure 3.7: Thin curved objects can make it impossible to place a simplified VLM (orange) in a position that remains completely inside the object (blue), as shown on the left, making it necessary to use a very detailed VLM, as shown on the right



Source: original image.

alent object in the visible geometry through the use of a manually created tag. When creating the VLM, the artist associates a numeric id with the real geometry and the corresponding virtual light geometry. This can be done either in a primitive level granularity or at more coarse objects for simplicity. During rendering, the surface point being shaded does not evaluate the illumination from virtual light sources that share its id, eliminating the incorrect self-illumination issue.

3.5 Properties

As discussed previously, the main advantage presented by VLMs is the greater amount of control that it provides artists over the visual result, by allowing them to precisely define the shapes and reflective properties of the simplified geometry used for indirect illumination independently from the characteristics of the visual geometry. They are able to add detail in certain parts of the scene that have greater visual importance or focus and reduce detail in less important regions, even completely excluding them from participating in indirect illumination if desired.

Additionally, the artist may choose to alter, with the same degree of granularity, the visual behavior of indirect illumination at a given part of the scene. For example, they are free to adjust the intensity or the color hue of the indirect light reflected in a certain area of the scene to evoke a specific mood without having to change the visible geometry or the real light source characteristics.

Since they are composed of polygonal meshes, VLMs can be integrated into existing asset production and management pipelines with ease. Their creation and manipulation can also be extended with small modifications to existing standard asset creation tools. Additionally, operations that can be applied to the visible geometry can also be applied to the corresponding virtual light geometry, allowing for effects such as animation and procedural generation.

Finally, due to the way the complexity of a scene's VLM can be arbitrarily scaled down (it is possible to construct a VLM comprised of a single primitive), it can be used in combination with other indirect illumination techniques, to complement them. For instance, one could use Precomputed Radiance Transfer (PRT) to compute the indirect illumination from the static scene geometry and apply VLMs only to specific objects with dynamic geometry as desired.

3.6 Limitations

The indirect illumination effect provided by VLMs is restricted to diffuse reflection, a limitation that is exploited both to allow for fast evaluation of area lights and for the simplification of the virtual light geometry. The method is also restricted to a single bounce of indirect illumination, due to the exponentially complex nature of evaluating lighting between virtual lights.

Finally, while the illumination model used for computing the intensity value for the virtual lights can use the real light solution for visibility, allowing the intensity of the reflected lights to take direct light shadows into account, we do not currently employ a visibility solution for the illumination generated by the virtual lights. This means that the indirect illumination computed by VLMs does not take occlusion into account and, thus, does not produce shadows.

3.7 VLM Creation

The creation of the VLM can be done by the artist either additively or subtractively. In additive creation, the artist will start from an empty VLM and manually add piece by piece the virtual light geometry to the regions of the scene where they feel the indirect illumination effect will be more impactful. The previously discussed restrictions must be taken into account during placement and the desired performance budget will limit the total complexity of the VLM.

To create it subtractively, the artist will start from a VLM containing the visible geometry. They then must simplify that mesh and possibly remove portions that they judge to be unimportant until they reach the desired performance budget. Finally, they must adjust the simplified mesh to make it adhere to the previously discussed restrictions.

4 ACCELERATION OF VLMS

The performance cost of evaluating indirect illumination with a VLM grows with the number of primitives that compose it. For that reason, while a simple VLM composed of a small number of primitives can be directly used for rendering indirect illumination while maintaining interactive frame-rates, any reasonably complex scene will require an equivalent VLM that is too large to be used in that fashion. Moreover, as scenes in production applications can grow arbitrarily complex, the cost of using the corresponding VLM would also grow indefinitely. This growth makes direct optimization of the per virtual light performance cost of indirect illumination insufficient to make it generally viable.

To solve that issue, we must be able to limit the number of virtual lights that need to be considered to evaluate the illumination at each visible surface point, independently of the total number of primitives present in the VLM. In this chapter, we present an acceleration data structure that aims to solve this problem by making it possible to query the subset of virtual lights that potentially illuminate a given point in space.

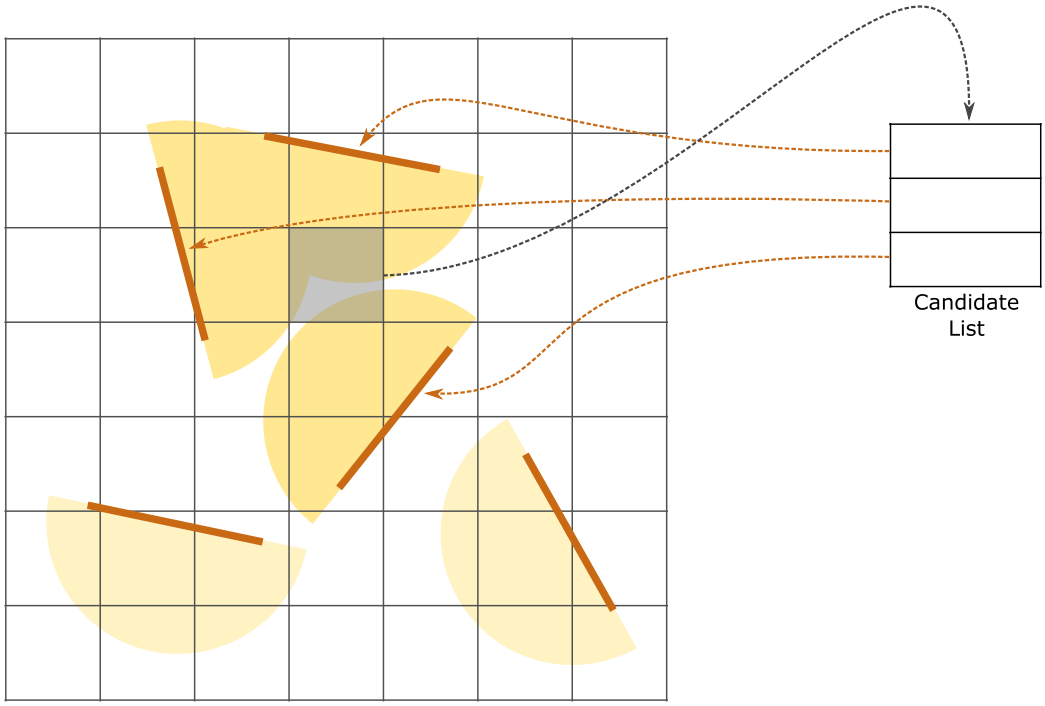
4.1 Data Structure

To be used in real time rendering, the data structure must allow us to quickly query the subset of virtual lights that potentially illuminate a given surface point. To that end, it must encode the influence of the virtual lights that compose the VLM across the scene's volume. With those query results, the renderer only has to consider the small subset of virtual lights that will actually contribute to the indirect illumination at each surface point, independent of the total number of virtual lights in the VLM.

In addition to that, this acceleration data structure must be efficient to create, or it will burden the technique with the requirement of precomputation and, with it, the restriction to static geometry. To be adaptable to a wider variety of applications, the data structure must also have configurable performance characteristics. Finally, to efficiently use the computing power of GPU hardware in its construction, it must be efficiently parallelizable.

To satisfy these requirements, our proposed data structure is based on a uniform 3D grid structure. The grid structure uniformly divides the scene's volume and stores at each grid cell a list of the virtual lights in the VLM that are candidates to be used when computing the indirect illumination at any surface point that resides inside that cell. The

Figure 4.1: Representation (in 2D) of the proposed VLM acceleration data structure where each grid cell stores a list of virtual lights (orange) that potentially illuminate (yellow region) surface points positioned within that cell's volume



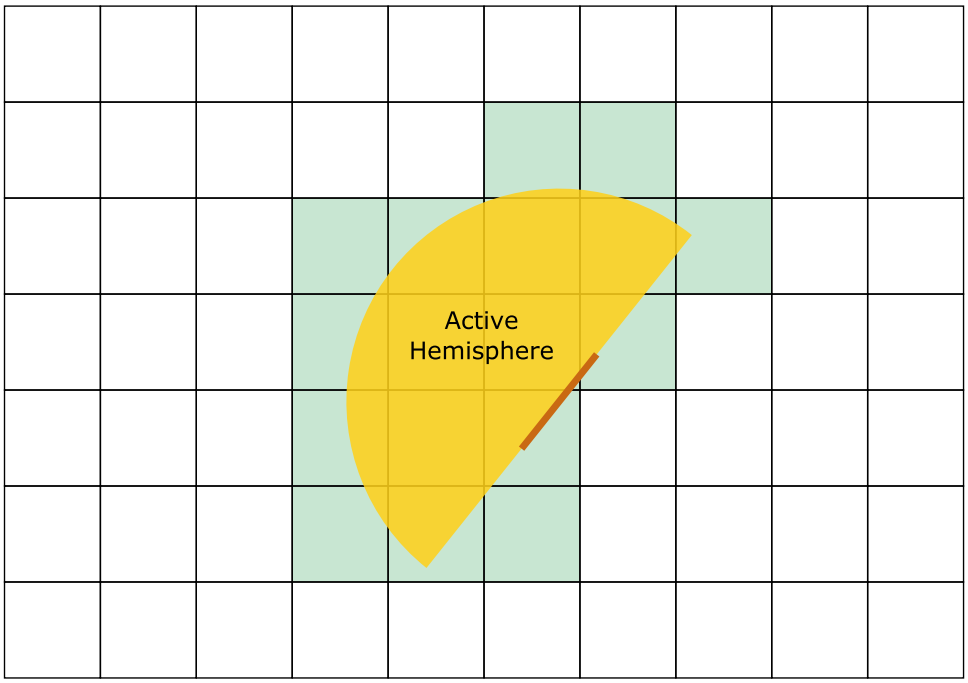
Source: original image.

grid's resolution and the maximum number of virtual lights to be stored in each cell are taken as configurable performance parameters. Figure 4.1 illustrates the organization of this structure.

As discarding virtual lights that contribute significantly to the illumination of a given cell from its candidate list would lead to incorrect results, the chosen maximum number of virtual lights per cell must be able fit the structure of the VLM that is being used, allowing for all influential virtual lights to be stored. For that reason, reducing this value may require that some portions of the VLM be altered to fit the limit. This means that the correct use of the acceleration structure does not impact the visual results, it only eliminates unnecessary work.

In memory, the cells' lists of candidate virtual lights are stored inlined as fixed size arrays of the same size, allowing for direct access using the grid cell index. Each position in the fixed size array stores an integer index into a separate array containing all of the VLM's virtual lights, avoiding the need to duplicate virtual light data. If the number of candidate lights for a cell is smaller than the maximum list size, the remaining entries are set to a special virtual light that generates no illumination. The total memory cost of the acceleration data structure is given by the number of cells multiplied by the number of

Figure 4.2: Representation (in 2D) of the heuristics used to select candidate virtual lights where the virtual light (orange) defines its active hemisphere (yellow region) that is used to determine which grid cells are potentially illuminated by it (green)



Source: original image.

candidate lights per cell and multiplied by the size of the integer index.

4.2 Creation

In order to efficiently choose which virtual lights will be stored into each cell grid, we present a heuristic test that decides if a given virtual polygonal light is an interesting candidate for evaluation at a given surface point. For a certain virtual polygonal light source, we take the sphere centered on the virtual light's centroid and whose radius is equal to the distance of the farthest virtual light vertex to the centroid multiplied by a user-supplied range parameter. The half of this sphere on the light emitting side of the virtual light source is defined as its active hemisphere, as shown in Figure 4.2. A virtual light is said to be a candidate for illuminating a point if the point resides inside its active hemisphere. This heuristic is chosen due to its low evaluation cost.

The data structure creation is executed on the GPU using shaders and the structure is stored in GPU memory to minimize data transfer cost. Each grid cell is processed in parallel and evaluates the heuristic to choose its candidate virtual lights. The spatially coherent nature of the virtual lights' active hemispheres guarantees that a light that is a

candidate for a given grid position will likely also be a candidate for its neighbors. As the GPU simultaneously processes groups of close cells, each group of cells is likely to generate the same results, which works more efficiently on the GPU.

The performance configuration parameters must be tuned to the application's target platform as they are dependent on the available processing power and the performance budget allotted to indirect illumination. As a virtual light can be selected for a cell's candidate list even if it does not significantly illuminate most of that cell's volume, the grid's resolution will influence the quality of candidate selection. Increasing the grid's resolution, and consequently reducing the volume of each grid cell, decreases the likelihood that its cells will contain poor quality candidates, which can reduce the required maximum number of candidates per cell, at the cost of increasing the data structure's creation cost. The required maximum number of candidates per cell can also be lowered to improve rendering performance through reducing the range parameter or through modification of the VLM geometry.

4.3 Indirect Illumination with the Acceleration Data Structure

The computation of indirect illumination using the acceleration data structure is similar to the basic computation discussed in Section 3.3. Instead of using the entire set of virtual lights present in the VLM to evaluate the indirect illumination at each surface point, we must add an extra step into the shading function where the surface point's position is used to query the acceleration data structure and obtain the restricted set of candidate virtual lights to evaluate for that surface point. We then aggregate the illumination contribution from all candidate virtual lights to obtain the final indirect illumination result.

Since we only require the surface point's position to query the data structure, it can be used both in forward renderers, using a depth only render pre-pass to obtain the positions, or with a deferred renderer, using the position stored in the G-Buffer.

Given that the resulting acceleration data structure is only dependent on the geometry of the virtual lights that compose the VLM, it is possible for us to minimize the structure's per-frame management cost by keeping two separate data structures: one for the portion of the VLM corresponding to static geometry, that will not change between frames, and another for the portion of the VLM corresponding to dynamic geometry, that must be recreated at each frame. In this way, at each frame we only incur the cost of

recreating the data structure for the dynamic subset of the VLM, reducing the impact on our frame rate.

5 RESULTS

In this chapter we present the test results obtained from our indirect illumination method. We have run all tests on an Intel i7 4770 CPU with 16GB of RAM and a NVIDIA GeForce GTX 950 GPU with 2GB of VRAM. To evaluate our technique we have created two test scenes: one test scene inspired by the Cornell Box scene comprised of 5,144 triangles with a VLM containing 44 virtual lights and another scene using the Crytek Sponza (CRYTEK, 2010) model comprised of 262,209 triangles with a VLM containing 100,000 virtual lights. The VLM for the Cornell Box scene was created additively and the one for the Sponza scene was created subtractively by simplification of the original geometry. All image results from the Cornell Box scene have been rendered at 720×720 pixels and those from the Sponza scene at 1,280×720 pixels.

5.1 Performance

In order to evaluate the performance characteristics of our method, we have first measured the performance of directly using a VLM for indirect illumination without the use of an acceleration data structure. The results shown in Table 5.1 were generated using the Sponza test scene rendered at 1,280×720 pixels using test VLMs with different numbers of virtual lights. As can be seen, it is possible to directly apply VLMs that contain as many as 200 virtual lights while maintaining interactive frame rates.

Table 5.1: Rendering performance for VLMs with different numbers of virtual lights without using an acceleration data structure

Virtual lights in VLM	*Frame Time (ms)*	*Frames Per Second*
20	2.40	416.7
40	6.25	160.0
60	11.07	90.3
80	12.88	77.6
100	17.15	58.3
120	21.03	47.6
140	24.41	41.0
160	26.63	37.6
180	30.85	32.4
200	33.14	30.2
220	38.22	26.2
240	41.37	24.2

Source: original data.

For more complex VLMs, using the acceleration data structure becomes necessary. To that end, we have measured the cost of generating that structure. As there are two variables that govern the processing cost during creation of the data structure, we have measured both of their influences on the total cost. Table 5.2 shows the creation times for an acceleration data structure with different grid sizes based on VLMs of different sizes created by progressively simplifying the original geometry of the Sponza test scene.

Table 5.2: Creation time (ms) for acceleration data structure

Grid Size	*Virtual Light Count (Continues)*				
	10,000	*20,000*	*30,000*	*40,000*	*50,000*
10×10×10	1.18	2.58	3.77	4.86	6.15
20×20×20	9.82	20.11	30.19	39.68	49.71
30×30×30	33.73	67.58	101.46	135.43	168.72
40×40×40	80.24	159.53	240.63	319.92	400.41
50×50×50	155.63	313.28	467.55	623.61	782.03
60×60×60	269.67	540.71	810.61	1,080.84	1,346.55
70×70×70	428.74	858.15	1,286.98	1,713.08	2,143.82
80×80×80	638.51	1,278.62	1,918.07	2,559.05	3,195.69
90×90×90	908.59	1,820.62	2,733.54	3,640.62	4,547.87
100×100×100	1,247.95	2,500.49	3,749.59	4,990.72	6,240.62

Grid Size	*Virtual Light Count (Conclusion)*				
	60,000	*70,000*	*80,000*	*90,000*	*100,000*
10×10×10	7.37	8.88	9.89	11.27	12.56
20×20×20	59.82	69.91	79.41	90.23	99.38
30×30×30	203.09	236.03	269.02	302.78	337.15
40×40×40	480.56	559.17	640.57	719.61	800.26
50×50×50	935.03	1,092.83	1,248.53	1,405.72	1,560.68
60×60×60	1,618.38	1,890.37	2,158.08	2,428.02	2,699.22
70×70×70	2,567.63	2,999.58	3,428.62	3,851.54	4,281.91
80×80×80	3,833.15	4,472.84	5,113.12	5,758.82	6,389.94
90×90×90	5,462.31	6,375.11	7,280.95	8,192.07	9,109.06
100×100×100	7,488.04	8,737.14	9,994.31	11,244.98	12,494.19

Source: original data.

Finally, we have measured the performance cost of rendering indirect illumination using an acceleration data structure to restrict the number of virtual lights evaluated at each illuminated surface position. Given that, in this case, the performance depends only on the number of virtual lights per grid cell and that the number is the same for all of the scene's cells, Table 5.3 shows the rendering cost for various values of this variable on a 50×50×50 grid based on the Sponza test scene.

Table 5.3: Rendering performance using an acceleration data structure with various numbers of virtual lights per grid cell

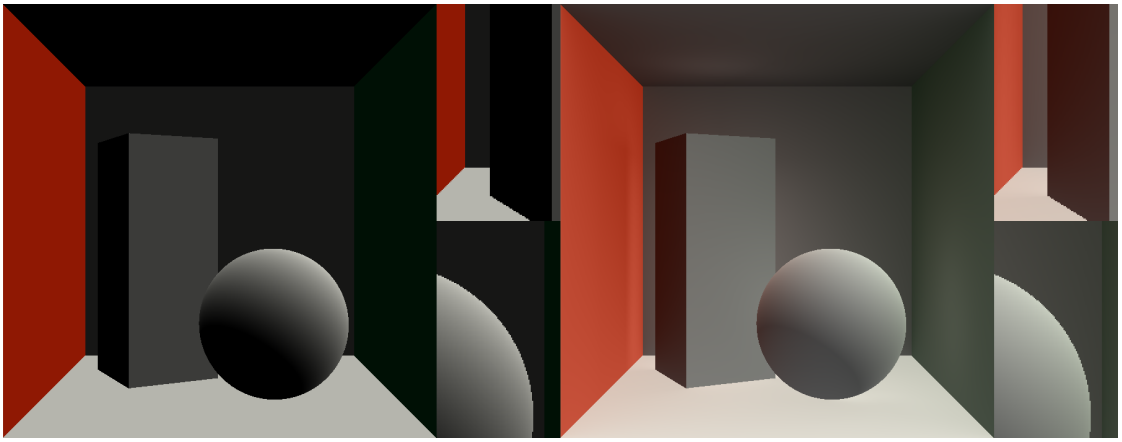
Cell List Size	*Frame Time (ms)*	*Frames Per Second*
10	12.96	77.2
20	15.77	63.4
30	16.54	60.5
40	27.03	37.0
50	29.77	33.6
60	33.57	29.8
70	51.60	19.4
80	79.55	12.6
90	86.09	11.6
100	89.86	11.1
110	96.75	10.3
120	115.26	8.7

Source: original data.

5.2 Visual

For the visual results presented in this section, we use our two prepared test scenes: the Cornell Box scene without the use of an acceleration data structure and the Sponza scene using an acceleration data structure with a grid size of $40 \times 40 \times 40$ and 30 candidate virtual lights per grid cell.

Figure 5.1: Comparison between direct illumination only (left) and direct plus indirect illumination (right) on the Cornell Box test scene with light coming from the upper back right corner showing close-up details of the color bleed in the box and the sphere

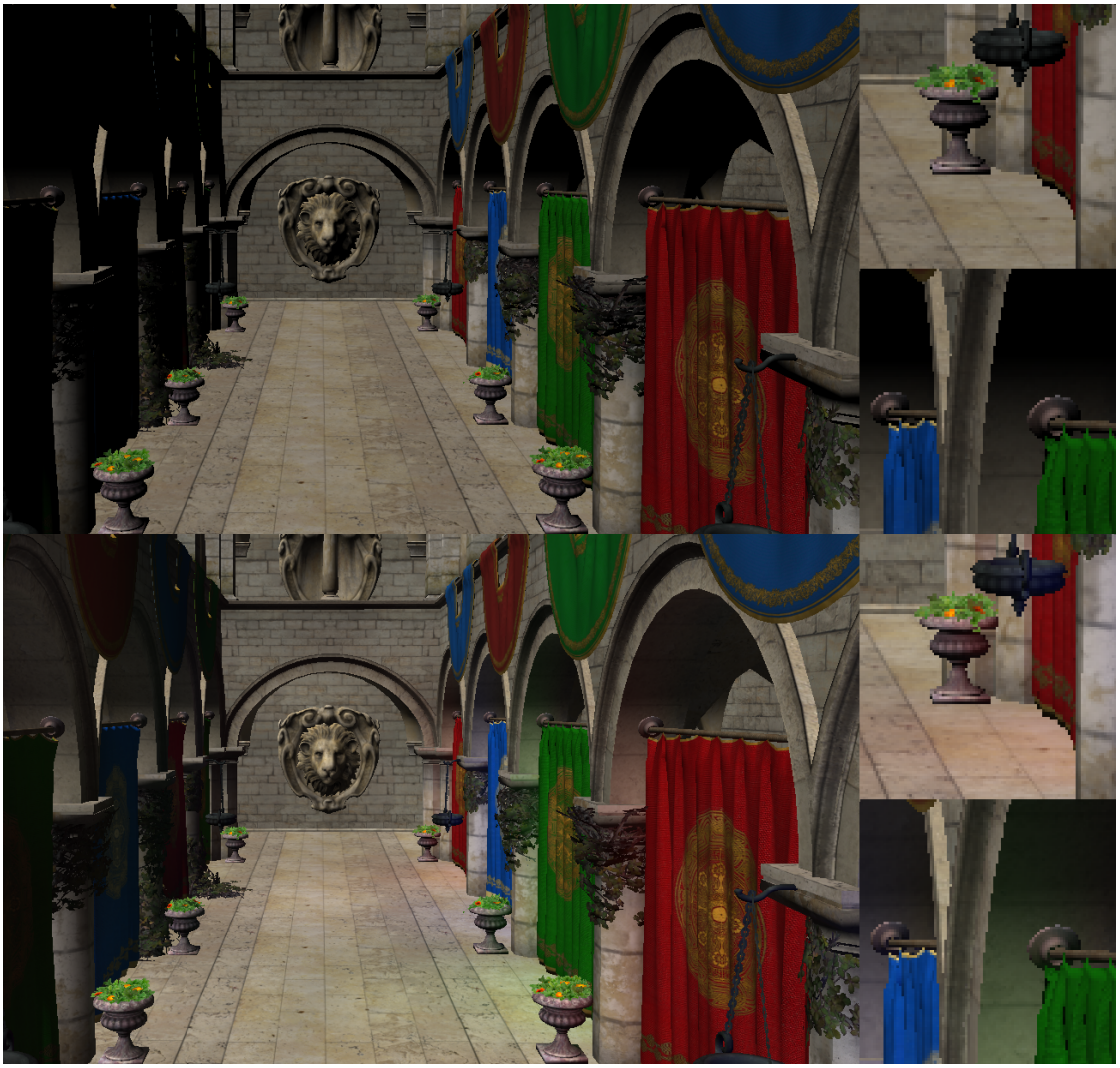


Source: original image.

We present comparisons between scenes rendered using only direct illumination and scenes rendered using both direct and indirect illumination for the Cornell Box test scene in Figure 5.1 and for the Sponza test scene in Figures 5.2 and 5.3. It is possible to

note the illumination of areas that are not reached by the real light source and the color bleed effect.

Figure 5.2: Comparison between direct illumination only (top) and direct plus indirect illumination (bottom) on the Sponza test scene showing close-up details of the color bleed in the scene



Source: original image.

Figure 5.4 shows the geometry of the virtual lights that compose the VLM used in each test scene. It is possible to see that the virtual light geometry can be simplified in comparison with the visible geometry and produce pleasant results.

As the main advantage of our method is the ability to arbitrarily control the visual results of indirect illumination, we present in Figure 5.5 the results of altering the VLM's characteristics in the Cornell Box test scene to change the intensity of indirect illumination generated by the red wall and even the hue of the reflected light. As the results show, that can be achieved without altering the visible characteristics of the scene's visible geometry

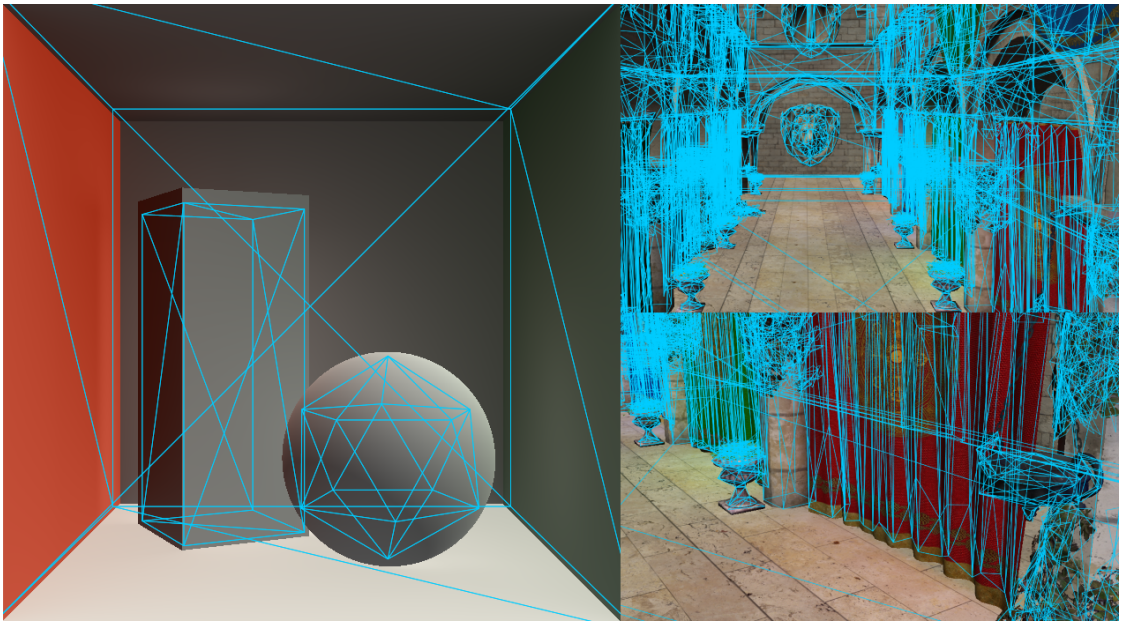
Figure 5.3: Comparison between direct illumination only (top) and direct plus indirect illumination (bottom) on the Sponza test scene showing close-up details of the color bleed in the scene (different position)

Source: original image.

or the characteristics of the real light sources.

Finally, we add an acceleration data structure with grid size $10\times10\times10$ and 20 candidate virtual lights per grid cell to the Cornell Box test scene. With that data structure, we show in Figure 5.6 the artifacts that can result from using a value for the range parameter in the acceleration data structure heuristic that is insufficient to properly approximate the virtual lights' illumination region. We can see that if the value is too low, surface points that should be illuminated by a given virtual light may be located in grid cells that are not located in that virtual light's active hemisphere, causing the visible abrupt blocky edge.

Figure 5.4: VLM geometry used for rendering indirect illumination in the test scenes



Source: original image.

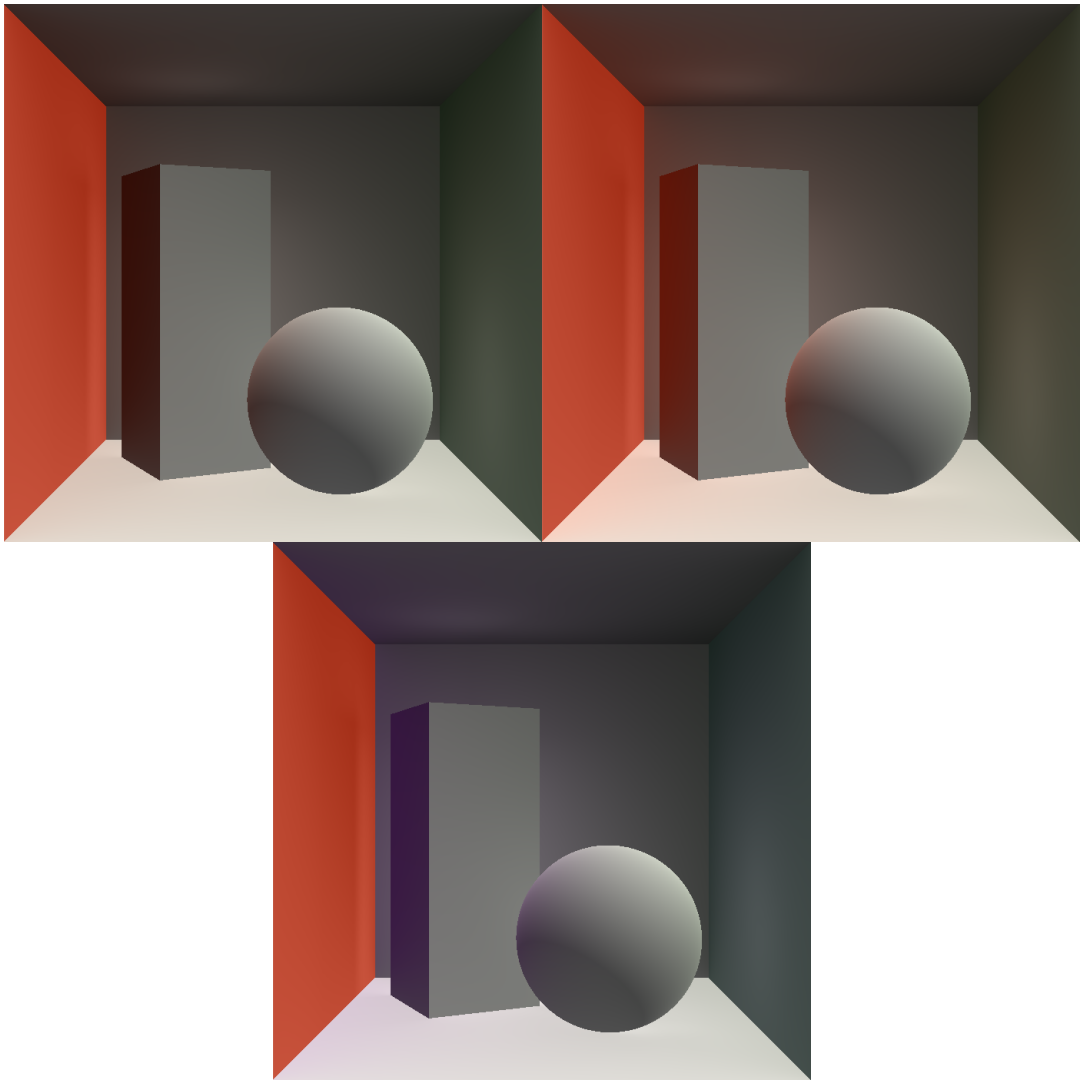
5.3 Qualitative Evaluation

Given that the main motivation behind our method is providing a greater level of control over indirect illumination effects to artists, we have consulted professional artists to present our results and obtain feedback on the validity of the approach we have taken to provide that control. To that end, the company Aquiris Game Studio has allowed us to meet with some of their team and discuss our method.

After becoming familiarized with our method's functioning and results, their response was markedly positive both to the general idea of artistic control of indirect illumination and to the specific method we presented to input this control using polygonal meshes. They have also appreciated the possibility of using our method to add indirect illumination effects such as color bleed to specific portions of a scene without having to pay the performance cost for indirect illumination in the entire scene.

Finally, they have recommended that we develop a production-ready implementation of our method integrated into commercial engines such as Unity 3D or Unreal Engine.

Figure 5.5: Original scene (top left) contrasted with manipulated intensity of the indirect illumination in the red wall (top right) and manipulated hue of the indirect illumination in the red wall (bottom)



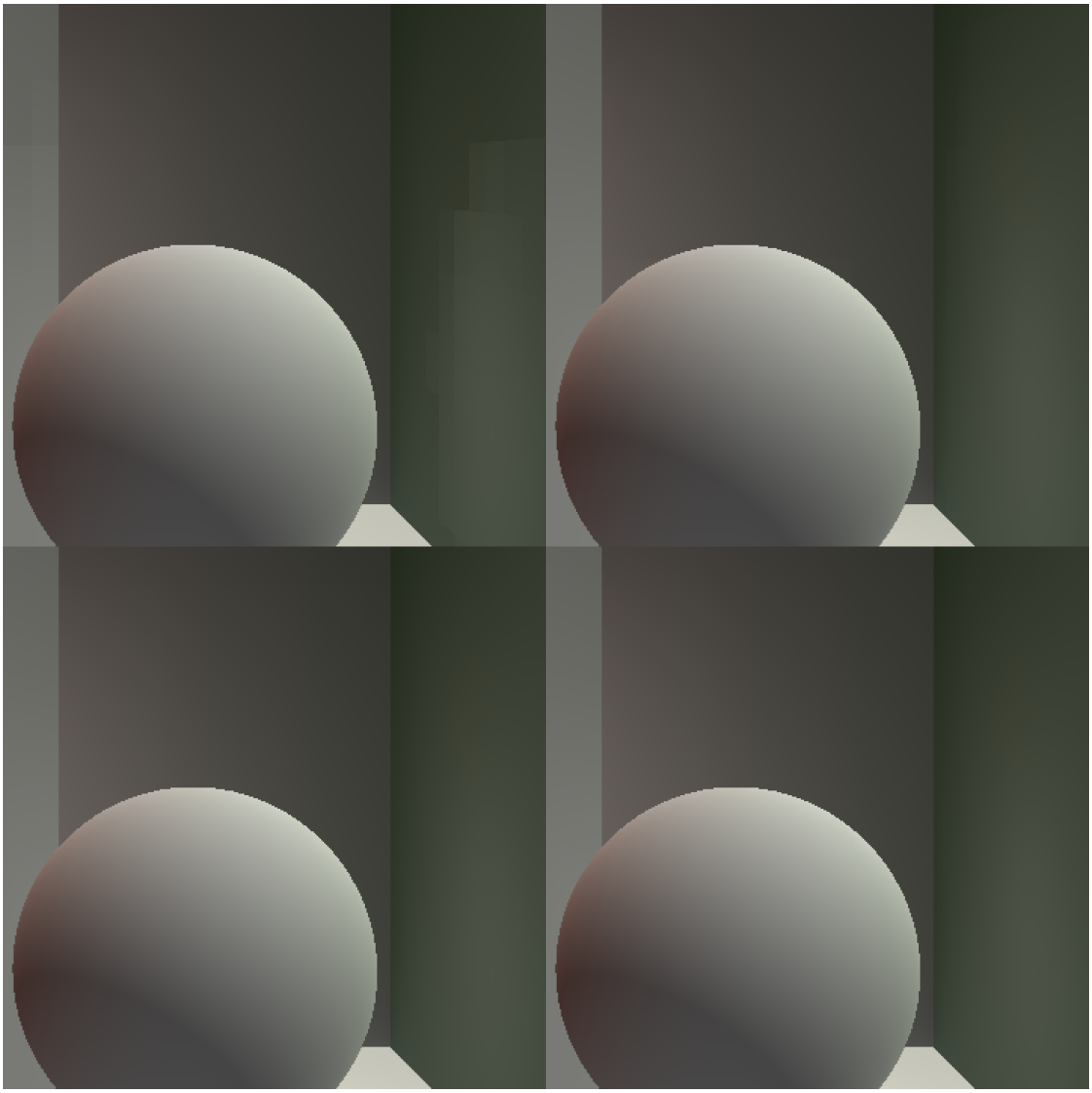
Source: original image.

5.4 Discussion

Our results show that not only is our method able to provide traditional diffuse indirect illumination effects in real-time, but it can also provide effects that can not be achieved directly with existing methods, such as arbitrary manipulation of indirect illumination characteristics and application to restricted portions of a given scene.

The positive response from professional game developers also supports our basic motivation and indicates the validity of this artistically controlled approach to methods advanced rendering effects.

Figure 5.6: Artifacts caused by an insufficient value for the range parameter in the acceleration data structure creation heuristic with values (from left to right and top to bottom): 1.5, 2.5, 3.5, and 4.5



Source: original image.

6 CONCLUSIONS

In this thesis we have presented the Virtual Light Mesh, an alternative technique for computing single-bounce diffuse indirect illumination at interactive frame rates with an innovative focus on precise artistic control of the appearance and quality of the indirect illumination, that can be directly applied to simple portions of scenes. We have shown the use of a rich input format to provide that fine-grained control to artists in a manner that is familiar to them.

Additionally, we have designed an acceleration data structure that makes it possible to apply that artistic control to more complex scenes, with VLMs with larger polygon counts, extending the applicability of the technique for general usage.

We have shown the viability of this alternative approach in producing visually pleasing results that can be arbitrarily manipulated without modification to the visual geometry or the lighting configuration.

Future research based on the VLMs presents many interesting avenues of exploration. Techniques such as Imperfect Shadow Maps, used in existing indirect illumination methods may be adaptable to add indirect illumination shadows to our method. Better optimization may also be achievable with the use of more intricate GPU tuned data structures and caching strategies. Tooling improvements for semi-automatic generation and manipulation of VLMs would also reduce the initial adoption cost of our method. Finally, there is active research in fast, high quality general polygonal light rendering (LECOCQ; SOURIMANT; MARVIE, 2015; HEITZ et al., 2016), which could potentially be adapted to use with VLMs, improving our visual results.

REFERENCES

AILA, T.; LAINE, S. Understanding the efficiency of ray traversal on gpus. In: **Proceedings of the Conference on High Performance Graphics 2009**. New York, NY, USA: ACM, 2009. (HPG '09), p. 145–149. ISBN 978-1-60558-603-8. Available from Internet: <http://doi.acm.org/10.1145/1572769.1572792>.

BAUM, D. R.; RUSHMEIER, H. E.; WINGET, J. M. Improving radiosity solutions through the use of analytically determined form-factors. **SIGGRAPH Comput. Graph.**, ACM, New York, NY, USA, v. 23, n. 3, p. 325–334, jul. 1989. ISSN 0097-8930. Available from Internet: <http://doi.acm.org/10.1145/74334.74367>.

BOULOS, S. et al. Packet-based whitted and distribution ray tracing. In: **Proceedings of Graphics Interface 2007**. New York, NY, USA: ACM, 2007. (GI '07), p. 177–184. ISBN 978-1-56881-337-0. Available from Internet: <http://doi.acm.org/10.1145/1268517.1268547>.

CIGOLLE, Z. H. et al. A survey of efficient representations for independent unit vectors. **Journal of Computer Graphics Techniques (JCGT)**, v. 3, n. 2, p. 1–30, April 2014. ISSN 2331-7418. Available from Internet: <http://jcgt.org/published/0003/02/01/>.

CRASSIN, C. et al. Interactive indirect illumination using voxel cone tracing: A preview. In: **Symposium on Interactive 3D Graphics and Games**. New York, NY, USA: ACM, 2011. (I3D '11), p. 207–207. ISBN 978-1-4503-0565-5. Available from Internet: <http://doi.acm.org/10.1145/1944745.1944787>.

CRYTEK. **Sponza Model**. 2010. Available from Internet: <http://www.crytek.com/cryengine/cryengine3/downloads>.

DACHSBACHER, C.; STAMMINGER, M. Reflective shadow maps. In: **Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games**. New York, NY, USA: ACM, 2005. (I3D '05), p. 203–231. ISBN 1-59593-013-2. Available from Internet: <http://doi.acm.org/10.1145/1053427.1053460>.

EVERITT, C. Interactive order-independent transparency. **White paper, nVIDIA**, v. 2, n. 6, p. 7, 2001.

GASTAL, E. S. L.; OLIVEIRA, M. M. Adaptive manifolds for real-time high-dimensional filtering. **ACM TOG**, v. 31, n. 4, p. 33:1–33:13, 2012.

GORAL, C. M. et al. Modeling the interaction of light between diffuse surfaces. **SIGGRAPH Comput. Graph.**, ACM, New York, NY, USA, v. 18, n. 3, p. 213–222, jan. 1984. ISSN 0097-8930. Available from Internet: <http://doi.acm.org/10.1145/964965.808601>.

GREEN, P.; KAUTZ, J.; DURAND, F. Efficient reflectance and visibility approximations for environment map rendering. **Computer Graphics Forum**, Blackwell Publishing Ltd, v. 26, n. 3, p. 495–502, 2007. ISSN 1467-8659. Available from Internet: <http://dx.doi.org/10.1111/j.1467-8659.2007.01072.x>.

GREEN, P. et al. View-dependent precomputed light transport using nonlinear gaussian function approximations. In: **Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games**. New York, NY, USA: ACM, 2006. (I3D '06), p. 7–14. ISBN 1-59593-295-X. Available from Internet: <http://doi.acm.org/10.1145/1111411.1111413>.

HEITZ, E. et al. Real-time polygonal-light shading with linearly transformed cosines. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 35, n. 4, p. 1–8, jul. 2016. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/2897824.2925895>.

JENSEN, H. W. Global illumination using photon maps. In: **Proceedings of the Eurographics Workshop on Rendering Techniques '96**. London, UK, UK: Springer-Verlag, 1996. p. 21–30. ISBN 3-211-82883-4. Available from Internet: <http://dl.acm.org/citation.cfm?id=275458.275461>.

KAJIYA, J. T. The rendering equation. **SIGGRAPH Comput. Graph.**, ACM, New York, NY, USA, v. 20, n. 4, p. 143–150, aug. 1986. ISSN 0097-8930. Available from Internet: <http://doi.acm.org/10.1145/15886.15902>.

KELLER, A. Instant radiosity. In: **Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques**. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997. (SIGGRAPH '97), p. 49–56. ISBN 0-89791-896-7. Available from Internet: <http://dx.doi.org/10.1145/258734.258769>.

LAFORTUNE, E. P.; WILLEMS, Y. D. Bi-directional path tracing. In: **Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)**. Alvor, Portugal: [s.n.], 1993. p. 145–153.

LECOCQ, P.; SOURIMANT, G.; MARVIE, J.-E. Accurate analytic approximations for real-time specular area lighting. In: **ACM SIGGRAPH 2015 Talks**. New York, NY, USA: ACM, 2015. (SIGGRAPH '15), p. 1–1. ISBN 978-1-4503-3636-9. Available from Internet: <http://doi.acm.org/10.1145/2775280.2792522>.

LIU, X. et al. All-frequency precomputed radiance transfer for glossy objects. In: **Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques**. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004. (EGSR'04), p. 337–344. ISBN 3-905673-12-6. Available from Internet: <http://dx.doi.org/10.2312/EGWR/EGSR04/337-344>.

MARA, M. et al. Deep g-buffers for stable global illumination approximation. In: **HPG**. [s.n.], 2016. Available from Internet: <http://graphics.cs.williams.edu/papers/DeepGBuffer16>.

MCGUIRE, M. et al. Real-time global illumination using precomputed light field probes. In: **I3D 2017**. [s.n.], 2017. p. 11. Available from Internet: <http://graphics.cs.williams.edu/papers/LightFieldI3D17>.

PARKER, S. G. et al. Optix: A general purpose ray tracing engine. In: **ACM SIGGRAPH 2010 Papers**. New York, NY, USA: ACM, 2010. (SIGGRAPH '10), p. 66:1–66:13. ISBN 978-1-4503-0210-4. Available from Internet: <http://doi.acm.org/10.1145/1833349.1778803>.

RITSCHEL, T. et al. Imperfect shadow maps for efficient computation of indirect illumination. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 27, n. 5, p. 1–8, dec. 2008. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/1409060.1409082>.

RITSCHEL, T.; GROSCH, T.; SEIDEL, H.-P. Approximating dynamic global illumination in image space. In: **Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games**. New York, NY, USA: ACM, 2009. (I3D '09), p. 75–82. ISBN 978-1-60558-429-4. Available from Internet: <http://doi.acm.org/10.1145/1507149.1507161>.

SLOAN, P.-P.; KAUTZ, J.; SNYDER, J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 21, n. 3, p. 527–536, jul. 2002. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/566654.566612>.

SNOOK, G. Simplified 3d movement and pathfinding using navigation meshes. In: DELOURA, M. (Ed.). **Game Programming Gems**. [S.l.]: Charles River Media, 2000. p. 288–304.

SUN, X. et al. Interactive relighting with dynamic brdfs. In: **ACM SIGGRAPH 2007 Papers**. New York, NY, USA: ACM, 2007. (SIGGRAPH '07). Available from Internet: <http://doi.acm.org/10.1145/1275808.1276411>.

SUN, X. et al. Interactive relighting of dynamic refractive objects. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 27, n. 3, p. 35:1–35:9, aug. 2008. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/1360612.1360634>.

TSAI, Y.-T.; SHIH, Z.-C. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 25, n. 3, p. 967–976, jul. 2006. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/1141911.1141981>.

UNREAL. **Setting Up Collisions With Static Meshes**. 2017. Available from Internet: <https://docs.unrealengine.com/latest/INT/Engine/Content/Types/StaticMeshes/HowTo/SettingCollision/>.

VEACH, E.; GUIBAS, L. Bidirectional estimators for light transport. In: ____. **Photorealistic Rendering Techniques**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. p. 145–167. ISBN 978-3-642-87825-1. Available from Internet: <http://dx.doi.org/10.1007/978-3-642-87825-1_11>.

VEACH, E.; GUIBAS, L. J. Metropolis light transport. In: **Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques**. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997. (SIGGRAPH '97), p. 65–76. ISBN 0-89791-896-7. Available from Internet: <http://dx.doi.org/10.1145/258734.258775>.

WALTER, B. et al. Lightcuts: A scalable approach to illumination. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 24, n. 3, p. 1098–1107, jul. 2005. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/1073204.1073318>.

WANG, R.; TRAN, J.; LUEBKE, D. All-frequency relighting of non-diffuse objects using separable brdf approximation. In: **Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques**. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004. (EGSR'04), p. 345–354. ISBN 3-905673-12-6. Available from Internet: <http://dx.doi.org/10.2312/EGWR/EGSR04/345-354>.

WANG, R.; TRAN, J.; LUEBKE, D. All-frequency relighting of glossy objects. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 25, n. 2, p. 293–318, abr. 2006. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/1138450.1138456>.

WANG, R. et al. An efficient gpu-based approach for interactive global illumination. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 28, n. 3, p. 91:1–91:8, jul. 2009. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/1531326.1531397>.

WILLIAMS, L. Casting curved shadows on curved surfaces. **SIGGRAPH Comput. Graph.**, ACM, New York, NY, USA, v. 12, n. 3, p. 270–274, aug. 1978. ISSN 0097-8930. Available from Internet: <http://doi.acm.org/10.1145/965139.807402>.

WILLIAMS, L. Pyramidal parametrics. **SIGGRAPH Comput. Graph.**, ACM, New York, NY, USA, v. 17, n. 3, p. 1–11, jul. 1983. ISSN 0097-8930. Available from Internet: <http://doi.acm.org/10.1145/964967.801126>.

XU, K. et al. Spherical piecewise constant basis functions for all-frequency precomputed radiance transfer. **IEEE Transactions on Visualization and Computer Graphics**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 14, n. 2, p. 454–467, mar. 2008. ISSN 1077-2626. Available from Internet: <http://dx.doi.org/10.1109/TVCG.2007.70442>.

ZHOU, K. et al. Real-time kd-tree construction on graphics hardware. In: **ACM SIGGRAPH Asia 2008 Papers**. New York, NY, USA: ACM, 2008. (SIGGRAPH Asia '08), p. 126:1–126:11. ISBN 978-1-4503-1831-0. Available from Internet: <http://doi.acm.org/10.1145/1457515.1409079>.

APPENDIX A ILUMINAÇÃO INDIRETA DIFUSA EM TEMPO REAL USANDO MALHAS DE LUZES VIRTUAIS

Resumo da Dissertação em Português

A pesquisa em Computação Gráfica sempre busca melhorar a qualidade visual de imagens sintetizadas. Efeitos de Iluminação Global tais como Iluminação Indireta, onde a luz que ilumina um dado ponto é refletida uma ou mais vezes na cena antes de atingir aquele ponto, têm o potencial de melhorar significativamente a qualidade visual de uma cena renderizada. Enquanto este efeito é comum em renderização precomputada, ainda é um desafio utilizá-lo em aplicações interativas, devido à grande complexidade de se calcular as trajetórias complexas e numerosas que a luz pode tomar.

Existem diversas técnicas para atacar o problema de iluminação indireta em tempo real usando diferentes estratégias e sofrendo de diferentes limitações. Essas técnicas têm como foco serem o mais fisicamente corretas o possível, o que leva a um controle grosseiro de qualidade e performance e também a um pequeno número de variáveis que podem ser usadas para o controle dos resultados.

Esta dissertação apresenta uma técnica que proporciona um maior grau de controle artístico para o cálculo da iluminação indireta em tempo real. De acordo com as necessidades do produto e da sua visão artística, os artistas pode controlar de forma granular a qualidade da iluminação indireta em qualquer parte da cena. Nós também proporcionamos um controle para alterações locais nas características da iluminação indireta de uma forma familiar e simples de integrar em ferramentas e processos de criação de arte existentes.

Inspirados pelo uso de outras malhas não visuais tais como Malhas de Navegação e Malhas de Colisão, e de maneira similar às técnicas baseadas em Luzes Virtuais Pontuais, nós aplicamos Malhas de Luzes Virtuais criadas pelo artista e compostas de fontes de luz de área poligonais que são usadas para aproximar a iluminação difusa que seria refletida pela geometria real.

A.1 Malhas de Luzes Virtuais

A Malha de Luzes Virtuais (MLV) é uma malha poligonal criada pelo artista que representa uma versão simplificada da geometria da cena e que é usada para calcular uma

reflexão de iluminação indireta difusa. Durante a renderização, as primitivas da MLV funcionam como luzes virtuais que emitem a luz equivalente à que seria refletida por elas.

Enquanto o uso de luzes de área para o caso geral é muito custoso para aplicações interativas, adotando uma restrição de luzes poligonais com iluminação difusa, é possível usarmos uma solução analítica derivada por Baum, Rushmeier and Winget (1989) para realizar a iluminação em tempo real.

Usando esta formulação de luzes poligonais, a MLV simula a reflexão indireta da luz usando um processo de duas fases. Em cada quadro renderizado, na primeira fase o modelo de iluminação das fontes de luz reais é usado para determinar a intensidade da luz que deve ser emitida por cada luz virtual poligonal. Na segunda fase, tanto as luzes reais quanto as luzes virtuais são usadas para iluminar a geometria real, obtendo o resultado final de uma reflexão de iluminação indireta difusa.

A principal vantagem da MLV é o maior grau de controle proporcionado aos artistas, permitindo que eles definam de forma precisa a geometria simplificada responsável pela reflexão da luz indireta. Eles podem adicionar ou remover detalhes da MLV e também alterar as suas características de reflexão sem alterar a iluminação e a geometria reais, manipulando assim os resultados da iluminação indireta.

Além disto, o fato de serem compostas de malhas poligonais faz com que as MLVs possam ser facilmente integradas aos processos de criação de arte existentes. Ferramentas e procedimentos com suporte ao processamento de malhas poligonais podem ser facilmente adaptados para a inclusão das MLVs. Técnicas existentes baseadas na transformação de polígonos, como animação, também podem ser aplicadas à geometria da MLV.

Como o custo de renderização da iluminação indireta usando MLVs depende do número de primitivas que as compõem, é necessário que haja uma forma de limitar as primitivas processadas em cada ponto iluminado para que seja possível usarmos MLVs complexas com grandes números de primitivas. Para isso, projetamos uma estrutura de dados baseada em grades uniformes 3d para minimizar o processamento de fontes de luz virtuais que não contribuirão para o resultado final.

A estrutura de aceleração divide o espaço da cena em células uniformes e armazena em cada célula uma lista das fontes de luz virtuais que potencialmente iluminam pontos localizados dentro da célula. Durante a renderização, o cálculo da iluminação indireta é realizado usando somente as fontes de luz virtuais obtidas da consulta à estrutura de aceleração, fazendo com que o custo de renderização se torne independente do número total de primitivas na MLV.

Para criar esta estrutura de forma eficiente, nós definimos uma heurística para aproximar a influência de iluminação de uma fonte de luz virtual poligonal. Dada uma esfera centrada no ponto médio da fonte de luz poligonal e cujo raio é dado por um parâmetro de controle multiplicado pela distância daquele ponto médio até o vértice mais distante da fonte de luz, definimos o hemisfério ativo da fonte de luz como a porção desta esfera que se encontra na direção de emissão da luz. Usando esta heurística, consideramos uma fonte de luz como candidata para uma dada célula da grade uniforme se o seu hemisfério ativo intersecta o volume da célula.

A natureza da estrutura de aceleração permite a sua criação em GPU de forma eficiente, processando as células em paralelo e se aproveitando das características de distribuição espacial das regiões iluminadas por cada fonte de luz para garantir o processamento de forma coerente e melhorando a utilização do hardware paralelo.

A.2 Resultados e Conclusões

Para a avaliação dos resultados da nossa técnica, realizamos testes de performance e dos resultados visuais, além de uma avaliação qualitativa com artistas profissionais da área de jogos eletrônicos.

Nos testes de performance, verificamos a possibilidade do uso de MLVs com até 200 luzes virtuais mantendo taxas de atualização interativas, o que possibilita o seu uso para cenas simples sem a necessidade de adotar o uso da estrutura de aceleração. Para cenas mais complexas, no entanto, a necessidade do uso da estrutura fica claro. Com o uso da estrutura, é possível manter taxas de atualização interativas com até 60 luzes virtuais por célula da grade uniforme.

O tempo de criação da estrutura de aceleração varia de acordo com a resolução da grade uniforme e com a complexidade da MLV. Uma grade de $10 \times 10 \times 10$ células com uma MLV de 10.000 luzes virtuais leva 1,18ms enquanto uma grade de $100 \times 100 \times 100$ células com uma MLV de 100.000 luzes virtuais leva 12.494,19ms. Desta forma, enquanto a estrutura pode ser gerada em tempo real para cenas mais simples, para cenas complexas é necessário que ela seja gerada de antemão. Assim, é possível separar as partes estáticas e dinâmicas da cena em estruturas diferentes para minimizar o custo de criação em tempo real.

Nos testes visuais, mostramos os efeitos de iluminação indireta em duas cenas distintas, demonstrando os efeitos de iluminação de regiões não acessíveis diretamente às

fontes de luz reais e de transferência de cor entre superfícies próximas. Demonstramos também a habilidade de alterar os resultados de intensidade e de tom de cor da iluminação indireta sem alterar as características da iluminação e da geometria reais.

Finalmente, demonstramos os nossos resultados a artistas da empresa de jogos eletrônicos Aquiris Game Studio que responderam de forma positiva ao conceito geral de manipulação artística dos resultados de iluminação indireta, ao nosso uso proposto de malhas poligonais para realizar este controle e aos resultados obtidos.

Nós demonstramos, assim, que a técnica de Malhas de Luzes Virtuais é uma alternativa para a geração de uma reflexão de iluminação indireta difusa em temo real com um foco inovativo no controle artístico preciso dos resultados visuais de uma forma que é familiar aos artistas. Apresentamos também uma estrutura de aceleração que possibilita o uso desta técnica para cenas complexas, tornando-a aplicável para uso geral.