(3.1) REAL-TIME VERSUS OFFLINE RENDERING TECHNIQUES

Rendering techniques can be broken down into two distinct categories: real-time and offline. Offline rendering techniques require anywhere from seconds to many hours to render a single image. Current offline rendering techniques are able to render a ultra-realistic image that take into account many light sources as well as many types of light principles such as reflections, refraction, sub-surface light scatter, and more in very complicated scenes consisting of millions of triangles and at fairly high resolutions. Real-time rendering techniques render images at a fast enough rate to support multiple frames a second and vary greatly. These algorithms can be broken down into dynamic and static. Dynamic scenes allow a user to interact with the scene and actively change the scene such as moving the geometry, the camera, or the light source whereas static scene do not. As opposed to offline algorithms, real-time algorithms often have to make sacrifices when rendering the scene and therefore can't be as scientifically accurate as offline algorithms. Regardless of whether the algorithm is offline or real-time, the algorithm can trace it's roots back to a single equation, *The Rendering Equation*.

(3.2) THE RENDERING EQUATION

When discussing light transport in computer graphics, the most significant paper is *The Rendering Equation* [Kajiya 1986]. In it Kajiya presents an equation that generalizes most rendering algorithms. Such a statement can be confirmed by the fact that all rendering equations try to recreate the scattering of light off of different types of surfaces and materials. The rendering equation is an integral that is adapted from the study of radiative heat transfer for use in computer graphics with an aim at balancing the energy flow between surfaces. The equation, however, is still an approximation because it does not take into account diffraction and it assumes that the space between objects, such as air, is of homogeneous refractive index meaning that light won't refract due to particles in the air.

$$I(x,x') = g(x,x')[\epsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''] \quad (3.1)$$

The rendering equation is broken down into 4 parts. First, I(x,x') is the intensity of light or energy of radiation passing from point x' to point x measured in energy of radiation per unit time per unit area. Second, the geometry term, g(x,x'), indicates the occlusion of objects by other objects. This term is either 0, if x and x' are not visible from one another, or $1/r^2$ where r is the distance between x and x' if x and x' are visible from one another. Third, the emittance term, ε(x,x'), measures the energy emitted from point x' that reaches point x. Lastly, the scattering term, ρ(x,x',x''), is the intensity of energy

scattered by a surface point x' that originated from point x'' and then ends at point x. As mentioned in the previous section, illumination can be calculated using the Neumann series. A Neumann series is a mathematical series of the form:

$$\sum_{k=0}^{\infty} T^k \quad (3.2)$$

where T is an operator and therefore T^k is a notation for k consecutive operations of operator T.

Furthermore, the rendering equation can also be approximated using the Neumann series. This is done by rewriting the rendering equation above (equation 3.1) as:

$$I = g\varepsilon + gMI \quad (3.3)$$

where M is the linear operator given by the integral in the rendering equation. Next, we rewrite equation 3.3 as:

$$(1 - gM)\, I = g\varepsilon \quad (3.3)$$

so that we can invert it to get:

$$I = g\varepsilon + gMg\varepsilon + gMgMg\varepsilon + g(Mg)^3\varepsilon + \ldots \quad (3.4)$$

Equation 3.4 is a Neumann series of the form:

$$I = g\varepsilon \sum_{k=0}^{\infty} (Mg)^k \quad (3.5)$$

Equation 3.5 indicates that the rendering equation (equation 3.1) is the final intensity of radiation transfer as a sum of a direct term, a once scattered term, a twice scattered term, and so on. Therefore, as mentioned in the previous section, indirect illumination can be calculated by summing light incident on a surface due to the reflection of light *n*-times all the way up to infinity. The more scattered terms we include in the calculation, the better the approximation but worse performance. Therefore, in real-time applications, this needs to be avoided.

Next, we show how the rendering equation can be seen has a generalization of most rendering algorithms, but first we must cover some other rendering equations. A good place to start is with offline rendering techniques.

(3.3) OFFLINE RENDERING TECHNIQUES

Key examples of offline rendering techniques are ray tracing, radiosity, and photon maps. We begin with ray tracing.

(3.3.1) RAY TRACING

Ray tracing is a technique for rendering an image of a three-dimensional scene by casting rays from a camera positioned somewhere in the scene. These rays are shot into the scene and register the first surface it hits. From this surface point, additional rays goes to each of the light sources to determine occlusion from the light sources as well as to other surfaces to calculate reflections. These rays can also be used to calculate other lighting phenomena such as refractions. The rays from the camera can be cast into the scene using different sampling patterns and techniques such as 1 per pixel or many per pixel. Also, the rays can be cast through the center of each pixel or through the use of stochastic sampling can be cast through non-uniformly spaced location in each pixel to avoid aliasing artifacts or "jaggies." Ray tracing is able to recreate ultra-realistic scenes but at a high cost. Examples of ray tracing techniques include [Whitted 1980], [Cook 1986], and [Ward 1988]. With adaptations to ray tracing techniques and advances in technology, there now exist some interactive ray-tracing techniques mentioned in section 3.4.

Ray tracing can also be related to the rendering equation. [Whitted 1980] describes a new approximation for ray tracing by rewriting the Phong illumination model in order to improve the quality of specular reflections. The Phong illumination model is a way of calculating lighting on a surface through the combination of three components ambient, diffuse, and specular. Diffuse is the reflection of light from rough surfaces, specular is the reflection of light on shiny surfaces, and the ambient component accounts for the amount of light that is scattered throughout the scene. The ambient term is most similar to indirect lighting, but is a user-specified amount to avoid any actual calculations. The improved model from [Whitted 1980] is written:

$$I = I_a + k_d \sum_{j=1}^{j=ls} (\bar{N} \cdot \bar{L}_j) + k_s S + k_t T \quad (3.6)$$

where S is the intensity of light incident from the specular reflection direction, k_t is the transmission coefficient, and T is the intensity of light from the transmitted light direction. k_s and k_t are coefficients that are to be used to try to accurately model the Fresnel reflection law. Equation 3.6 is in the form of equation 3.4 from [Kajiya 1986] as I = gε +gMogεo +gMogMogεo + … with Mo as a scattering model which is the sum of reflection and refraction as well as a cosine term that is the diffuse component. The term gεo has shadows with point radiators and the ambient term can be interpreted as the ε term. Lastly, M is approximated by summing over all the light sources rather than using integration.

(3.3.2) RADIOSITY

Radiosity is a type of rendering algorithm that was adapted for use in computer graphics from thermal engineering techniques. The method is based on the fundamental Law of Conservation of Energy within a closed area. It provides a global solution for the intensity of light incident on each surface by solving a system of linear equations that describes the transfer of energy between each surface in the scene. Examples of radiosity are seen in [Immel et al. 1986] and [Goral et al. 1984].

Radiosity is a natural extension from the rendering equation (equation 3.1) since its focus is on the balancing the energy flow. The only difference is that radiosity makes assumptions about the reflectance characteristics of the surface material. Radiosity is found by taking the hemispherical integral of the energy leaving the surface called flux which can be found using the following from [Goral et al. 1984]:

$$B_j = E_j + \rho_j H_j \quad (3.7)$$

where B_j is the rate of energy leaving the surface j measured in energy per unit time per unit area, E_j is the rate of direct energy emission, ρ_j is the reflectivity of surface j, and H_j is the incident radiant energy arriving at surface j per unit time per unit area. Equation 3.7 can be derived using our rendering equation (equation 3.1) [Kajiya 1986] by integrating over all surfaces in the scene to calculate the hemispherical quantities, calculating the contribution of the emittance and reflectance terms by checking for occlusions, and using those calculations the rendering equation becomes:

$$dB(x') = \pi[\epsilon_0 + \rho_0 H(x')]dx' \quad (3.8)$$

where ε_0 is the hemispherical emittance of the surface element dx', ρ_0 comes from the reflectance term, and H is the hemispherical incident energy per unit time per unit area. This adaptation of the rendering equation (equation 3.8) is the same as the radiosity equation shown above (equation 3.7).

(3.3.3) PHOTON MAPS

Photon maps originally introduced in [Jensen 1996] is a two pass global illumination method. As mentioned in the Background section, Einstein coined the term photons as the particles present in the energy of light waves. In the method of photon mapping, the term photon is used in a similar context. The first pass of the method consists of making two photon maps by emitting packets of energy called photons from the light sources and storing where they hit surfaces in the scene. The second pass of the method calls for the use of a distribution ray tracer that is optimized using the data gathered in the photon maps. Photon maps are able to render complex lighting principles such as caustics.

Photon maps are an extension to the rendering equation (equation 3.1) as well. During the second pass of the method, the scene is rendered by calculating the radiance by tracing a ray from the eye through the pixel and into the scene using ray tracing, and the radiance is computed at the first surface that the ray hits. The surface radiance leaving the point of intersection x in some direction is computed using the equation from [Jensen 1996]:

$$L_s(x, \Psi_r) = L_e(x, \psi_r) + \int_{\Omega} (f_r(x, \psi_i; \psi_r) L_i(x, \psi_i) \cos(\theta_i) d\omega_i) \quad (3.9)$$

where L_e is the radiance emitted by the surface, L_i is the incoming radiance in the direction Ψ_i, and f_r is the BRDF or bidirectional reflectance distribution function, which is a four-dimensional function that describes how light is reflected at a surface point. Lastly, Ω is the sphere of incoming directions. This can be broken down into a sum of four components:

$$L_r = \int_{\Omega} (f_r L_{i,l} \cos(\theta_i) d\omega_i) + \int_{\Omega} (f_{r,s} (L_{i,c} + L_{i,d}) \cos(\theta_i) d\omega_i)$$
$$+ \int_{\Omega} (f_{r,d} L_{i,c} \cos(\theta_i) d\omega_i) + \int_{\Omega} (f_{r,d} L_{i,d} \cos(\theta_i) d\omega_i) \quad (3.10)$$

where the first term of equation 3.10 is the contribution by direct illumination, the second term is the contribution by specular reflection, the third term is the contribution by caustics, and the fourth term is the contribution by soft indirect illumination. Both equations 3.9 and 3.10 are direct adaptations from

the rendering equation in [Kajiya 1986] (equation 3.1).

(3.3.4) OTHER OFFLINE RENDERING TECHNIQUES

Many recent offline techniques have been influenced by real-time techniques most notably the idea of using virtual point lights or VPL's as introduced in *Instant Radiosity* [Keller 1997]. This technique will be discussed in detail in the real-time rendering techniques section, but the main idea is that we can render indirect light through the use of a set of VPL's where we accumulate the contributions of each of these lights in multiple rendering passes. Examples of using this technique are used for rendering illumination from area lights, high dynamic range (HDR) environment maps or sun/sky models, single/multiple subsurface light scattering in participating media, and most importantly for our purposes indirect illumination. For offline purposes, techniques typically call for the use of upwards of millions of VPL's to achieve higher quality renderings, however, many techniques attempt to mitigate the cost of using such a large number of lights.

VIRTUAL POINT LIGHTS

In *Lightcuts: A Scalable Approach to Illumination* [Walter et al. 2005], it is discussed that when using many lights, as in the VPL method, the cost to render the scene scales linearly with the number of lights used. This limits the number of VPL's we can use without serious performance impact. Therefore, the Lightcuts method is introduced as a way to reduce the rendering cost of using VPL methods by making it “strongly sub-linear” with the number of lights used without noticeable impact on quality. Using this method, hundreds of thousands of point lights can be accurately rendered using only a few hundred shadow rays. Lightcuts does this by clustering a group of VPL's together to form a single brighter light thereby reducing the cost of rendering from the number of lights present in the cluster group to a single light. This is done by using a global light tree which is a binary tree that has individual VPL's as the leaves and the interior nodes are the light clusters that contain the individual light below it in the tree. A “cut” of this tree then is “a set of nodes such that every path from the root of the tree to a leaf will contain exactly one node from the cut” and will represent a valid clustering of the lights. This approach was further advanced in *Multidimensional Lightcuts* [Walter et al. 2006] for use with visual effects such as motion blur, participating media, depth of field, and spatial anti-aliasing in complex scenes and again in *Bidirectional Lightcuts* [Walter et al. 2012] to support low noise rendering of complicated scenes with glossy surfaces, subsurface BSSRDF's, and anisotropic volumetric models.

MATRIX SAMPLING

Another adaption of the VPL method in offline rendering techniques is with the use of mathematical matrix sampling to reduce the rendering cost. In *Matrix Row-Column Sampling for the Many-Light Problem* [Hašan et al. 2007], it is shown that many point lights can be seen as a large matrix of sample-light interactions. The final image is then the sum of all of the columns of the matrix. Therefore, by sampling this matrix and using a small number of the rows and columns, we can approximate the final image at a fraction of the rendering cost. The matrix used in this technique is comprised of the columns representing all of the surface points to be lit by each light and each row being comprised of all of the lights in the scene. Then it can often be shown that this matrix is of low rank meaning that the row and/or columns of the matrix can be linearly combined to form a smaller matrix that would approximate the original larger matrix. Therefore, this method approaches the problem in a similar way as Lightcuts, but instead of using a tree, we use a matrix. This method is comprised of four primary steps. First, we sample r randomly selected rows using shadow maps on the GPU. Information on shadows maps to follow. Next, we partition the reduced columns into clusters on the CPU. Third, we pick a representative from each cluster to be scaled appropriately to account for the entire cluster on the CPU. Last, we accumulate each of these representatives using shadows maps on the GPU. By using this method, the render time of shading m surface points using n VPL's is reduced from O(mn) to O(m+n). Additionally, *LightSlice: Matrix Slice Sampling for the Many-Lights Problem* [Ou and Pellacini 2011] uses a similar approach as in [Hašan et al. 2007] and [Walter et al. 2005], but found that neither technique optimally exploits the structure of the matrices in scenes with large environments and complex lighting so some modifications were made to reduce render time and improved overall quality.

SHADOW MAPS

Shadow maps is a technique typically used to render shadows, but as shown in the real-time rendering techniques section, they have additional uses. They are created in a render pass where the view of the scene is computed from the light source's point of view and the distances from the light to the nearest surface for each pixel is stored in a texture or buffer to be used for comparisons when rendering the scene from the camera's point of view to determine whether a surface point is in shadow [Williams 1978], [Reeves et al. 1987]. Further discussion on shadow maps will be done in the real-time rendering techniques section as well as in the implementation section.

VIRTUAL RAY LIGHTS

Lastly, [Novák et al. 2012] describes a technique to render scenes with single/multiple light scattering in participating media in *Virtual Ray Lights for Rendering Scenes with Participating Media*. The techniques modifies the idea of using VPL's by instead using virtual ray lights or VRL's inside the participating media. So instead of evaluating each VPL at discrete locations, we calculate the contribution of each VRL with an efficient Monte Carlo sampling technique. Monte Carlo sampling involves the use of randomly selecting point based off a probability distribution and is often used for sampling in a wide variety of techniques. The reason for using VRL's over VPL's is that VPL's can be negatively affected by singularities in the scene. These singularities can cause artifacts in the scene such as spikes of high intensity which can be eliminated through the use of clamping or blurring. But by using a VRL, we compute the contribution of the light by distributing the energy over a line segment reducing singularities. In this case, instead of having spikes of high intensity, we have a uniform noise distributed evenly over the image which would be more pleasing to the eye.

(3.4) REAL-TIME RENDERING TECHNIQUES

With the advancement of technology and the expanded use of the GPU, many offline rendering techniques have been adapted to work on the GPU to run in real-time usually by making trade offs and approximations. This includes interactive ray tracing [Wald et al. 2002], approximate ray tracing on the GPU [Szirmay-Kalos et al. 2005], image space photon mapping [McGuire and Luebke 2009], and *Instant Radiosity* [Keller 1997]. The most significant advancement coming from the idea of virtual point lights as in *Instant Radiosity.*

(3.4.1) *INSTANT RADIOSITY*

As discussed in prior sections, *Instant Radiosity* introduces the technique of rendering indirect illumination approximated with the use of virtual points lights where each individual light acts independently as a producer of indirect illumination and behave like a normal point light source. In the original technique, these VPL's are generated using a quasi-random walk technique created at the hit points of the photons as they are traced into the scene from the primary light source. The contributions of these VPL's are the summed through the use of multiple rendering passes. This implementation had a few limitations such as requiring many rendering passes to support dynamic objects or lights as well as limited to simple environments due to the high cost of computing shadows for each of VPL's. These shadows had to be computed by using shadow volumes or shadow maps and were the bottleneck of the technique. There were also issues with low sampling rates which would result in weak singularities

when the VPL got increasingly close to the illuminated surface point as well as having high influence or contributions from each VPL on the overall color of the image. Also, temporal flickering can be seen if there are not enough VPL's being used. Despite these issues, real-time performance was attainable in 1997 through the use of this technique.

(3.4.2) ADAPTATIONS OF *INSTANT RADIOSITY*

In addition to the techniques already mentioned in the section 3.3.4 that adapted the idea of using VPL's for the offline rendering of multiple types of lighting phenomena, many real-time techniques also adapted the use of VPL's.

SHADOW MAP ALTERATIONS

As teased in section 3.3.4, in many real-time rendering techniques shadow maps have been expanded for use beyond just creating shadows. In *Translucent Shadow Maps* [Dachsbacher and Stamminger 2003] extended the binary shadow map look-up to a shadow map filter to assist in the implementation of real-time sub-surface scattering. Translucent Shadow Maps are this extension to shadow maps by having additional information store such as depth and incident light information. Therefore, each pixel in the Translucent Shadow Map stores the 3D position of the surface sample, the irradiance entering the object at that sample, and the surface normal. Although this implementation was for sub-surface scattering, it would lead to a development in rendering indirect illumination as well.

From this idea of extending shadow maps came further developments in *Reflective Shadow Maps* [Dachsbacher and Stamminger 2005]. This technique extended a shadow map such that each pixel in the shadow map would be thought of as an indirect light source. The reflective shadow map then stored information such as depth value, world space position, normal, and reflected radiant flux. With all this information available, we can then approximately calculate the indirect irradiance at a surface point by summing the illumination due to all pixel lights as seen in equations 3.11 and 3.12.

$$E_p(x,n) = \Phi_p \frac{max(0, n_p \cdot (x - x_p)) max(0, n_p \cdot (x_p - x))}{||x - x_p||^4} \quad (3.11)$$

$$E(x,n) = \sum_{pixels\ p} E_p(x,n) \quad (3.12)$$

where x is the surface point, n is the normal at x, n_p and x_p is the pixel light normal and position, and Φ_p is the reflected radiant flux of the visible surface point. This technique had the same singularity issue near the boundary of two adjoining walls. Limitations included ignoring occlusion and visibility for the indirect light sources as well as having to restrict the number of VPL's to around 400 meaning that sampling had to be done to chose the 400 best VPL's. Also, screen-space interpolation had to be performed by computing the indirect illumination using a low-resolution shadow map and interpolating using multiple low-res samples. With these restrictions in play, this technique renders approximate indirect illumination for dynamic scenes. Reflective shadow maps were then used in many real-time rendering techniques that followed. This included *Splatting Indirect Illumination* [Dachsbacher and Stamminger 2006] which rendered the VPL's contributions through a splatting technique in a deferred shading process which reduce the effect of scene complexity on the rendering time. It also allowed for efficient rendering of caustics and reduced scene artifacts.

Additionally, RSM's are used in other applications such as indirect illumination for area light sources as in *Direct Illumination from Dynamic Area Lights With Visibility* [Nichols et al. 2010]. Here we use RSM's to generate VPL's, but add visibility to the technique by computing occlusion by marching rays towards each VPL through a voxel buffer. The technique then checks the visibility for each VPL and adds in its illumination provided the VPL is visible.

Imperfect Shadow Maps for Efficient Computation of Indirect Illumination [Ritschel et al. 2008] altered shadow maps for their purposes by making them imperfect shadow maps or ISM's. These ISM's were low resolution shadow maps with a simplified point-representation of the scene such that some of the depth values could be incorrect. This simplified scene is done by approximating the 3D scene by a set of points with a near uniform density which is then used to create an ISM. The point-based representation of the geometry is used to allow the creation of hundreds of ISM's in parallel in a single pass to support dynamic scenes. These hundreds of ISM's are stored in a single large texture and used as approximate visibility for the hundreds or thousands of VPL's used for indirect illumination. These VPL's are generated through the use of RSM's. ISM's can also be built on top of reflective shadow maps to create imperfect reflective shadow maps to allow for multiple bounces of light. With the use of ISM's, indirect illumination scales well with and increase in scene complexity, however, it has no effect on the increase of rendering costs for the direct illumination component. Limitations include the traditional VPL method issues as discussed in section 3.4.1 as well as that indirect shadows cannot be generated for smaller geometry. Although, this technique calls for inaccurate visibility, it is an upgrade

to the strategy of ignoring visibility as in RSM's and it results in a very minimal impact on the final scene but allows for good performance with real-time global illumination.

SCREEEN SPACE ALGORITHMS

Additional VPL variants include *Hierarchical Image-Space Radiosity for Interactive Global* Illumination [Nichols et al. 2009] where instant radiosity is combined with multiresolution techniques such as [Nichols and Wyman 2009]. This is done by accumulating indirect illumination at a variety of different resolutions in image space depending upon singularities. When no singularities are nearby, a lower resolution can be used whereas when there are singularities, a higher resolution should be used to avoid artifacts. Similar to many of the previous methods mentioned, this techniques ignores visibility for indirect light.

Lastly, VPL approaches can introduce bias. When a VPL is close to a surface, it will introduce a singularity that appears as a high intensity peak in the image. Most techniques solve this by clamping a VPL's contribution to a surface if it is nearby, however, this removes energy from the system and therefore introduces a bias. This bias results in darkening of the image near singularity areas such as wall boundaries and edges of objects. In order to prevent this, screen-space bias compensation [Novák et al. 2011] was introduced as a post-processing step to recover the clamped energy. This step involves applying a residual operator to the direct illumination and clamped indirect illumination as computed through the use of the rendering equation (equation 3.1) and a new residual operator.

(3.4.3) SPHERICAL HARMONICS AND LATTICE-BASED METHODS

Spherical harmonics are the angular portion of a set of solutions to Laplace's equation represented in a system of spherical coordinates. In [Nijasure and Pattanaik 2005] they are used as a way to represent the incident light at each sample point on a regular grid comprised of both direct and indirect light that arrives from all surface points visible to that sample point in a compact way. This compact representation is comprised of a small number of coefficients that are computed through the use of a spherical harmonics transformation. The incident radiance is approximated through the use of a cube map at each grid point, stored as spherical harmonics coefficients, and then the indirect illumination is calculated by interpolating the radiance at the nearest grid points. This techniques allows for indirect occlusion through the use of shadow cube maps, but is expensive for complicated dynamic scenes. [Papaioannou 2011] combined the grid-based radiance caching of [Nijasure and Pattanaik 2005] and use of spherical harmonics along with the reflective shadow maps discussed in the above section.

Instead of of using cube maps to sample the visibility, this technique sampled the RSM to increase performance. [Kaplanyan and Dachsbacher 2010] used a volume-based method with lattices and spherical harmonics to represent the spatial and angular distribution of light in the scene where VPL's from a RSM are inserted into a volume texture. This light propagation volume allows for iterative propagation of energy among voxels as well as accounting for fuzzy occlusion by storing depth information and RSM's in a separate occlusion volume to compute indirect shadows that is limited to surfaces larger than the grid size. These light propagation volumes allow for the use of many more VPL's than other methods due to not calculating the contribution of each of the VPL's individually. The occlusion calculations are also view-dependent which leads to popping artifacts.