Energy-Efficient Photon Mapping

By

Brandon Wesley Light

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

May 2007

APPROVED:

Dr. Emmanuel Agu, Advisor _____

Dr. Robert Kinicki, Reader _____

Dr. Michael Gennert, Head of Department _____

Abstract

Mobile devices such as cell phones, personal digital assistants (PDAs), and laptops continue to increase in memory and processor speed at a rapid pace. In recent years it has become common for users to check their email, browse the internet, or play music and movies while traveling. The performance gains are also making mobile graphics renderers more viable applications. However, the underlying battery technology that powers mobile devices has only tripled in capacity in the past 15 years whereas processor speeds have seen a 100-fold increase in the same period.

Photon mapping, an extension of ray-tracing, is a robust global illumination algorithm used to produce photorealistic images. Photon mapping, like ray-tracing, can render high-quality specular highlights, transparent and reflective materials, and soft shadows. Complex effects such as caustics, participating media, and subsurface scattering can be rendered more efficiently using photon mapping.

This work profiles the energy use of a photon-mapping based renderer to first establish what aspects require the most energy. Second, the effect several photon mapping settings have on image quality is measured. Reasonable tradeoffs between energy savings and moderately diminished image quality can then be recommended, making photon mapping more viable on mobile devices.

Our results show that image quality is affected the least as settings corresponding to final gather computations are adjusted. This implies that a user can trade a modest decrease in image quality for significant gains in energy efficiency. Suggestions are made for using energy more efficiently when rendering caustics. Results also show that, although overall energy use is higher with larger image resolutions, per-pixel energy costs are cheaper.

Acknowledgements

I wish to first thank my advisor, Professor Emmanuel Agu, for all the time and effort he contributed to my thesis. When the work was daunting, he always provided the guidance needed to tackle each obstacle. His understanding and flexibility when problems did arise were always greatly appreciated. Beyond this, Professor Agu has furthered my interest in computer graphics and instilled in me an interest for future scientific research.

I would also like to recognize Zack Waters for all the time he saved me as I was learning photon mapping. He went above and beyond answering a few questions. Zack consistently surprised me each time he set aside time to meet and discuss photon mapping.

Professor Robert Lindeman shared his enthusiasm with us early in the process. He helped define our vision and was always a source for unique avenues of inquiry.

I owe much of my success to Peter Lohrmann and Chen-Hao (Jason) Chang. The ENCORE software used in this thesis was borne from our combined efforts. As my colleagues they were a valuable sounding board for any of my questions and thoughts.

Finally, I cannot thank my friends and family enough for their encouragement in the final stages of my work. This is especially true of both my mother and my sister whose support I will always be grateful to have.

Table of Contents

List of Figures

List of Tables

List of Equations

1 Introduction

Mobile devices have become increasingly powerful over the years allowing users access to common desktop applications such as email clients, web browsers, and media players in any environment. Laptops are powerful enough to replace desktop computers for those users who prefer convenience and portability. Personal digital assistants (PDAs) are indispensable for business professionals with busy schedules. Cell phone usage cuts across all demographics and is becoming ubiquitous.

These trends are fueled by the continuing improvement of processor speed, memory size, and storage devices. Figure 1 shows laptop technology trends between 1990 and 2000. Since 1990, hard disk capacity has increased 1,000 fold. Just as impressive, processor speed has increased 400 fold and available memory has increased 100 times. However, even with this rapid advancement, battery capacity has only increased roughly 3 times since 1990.

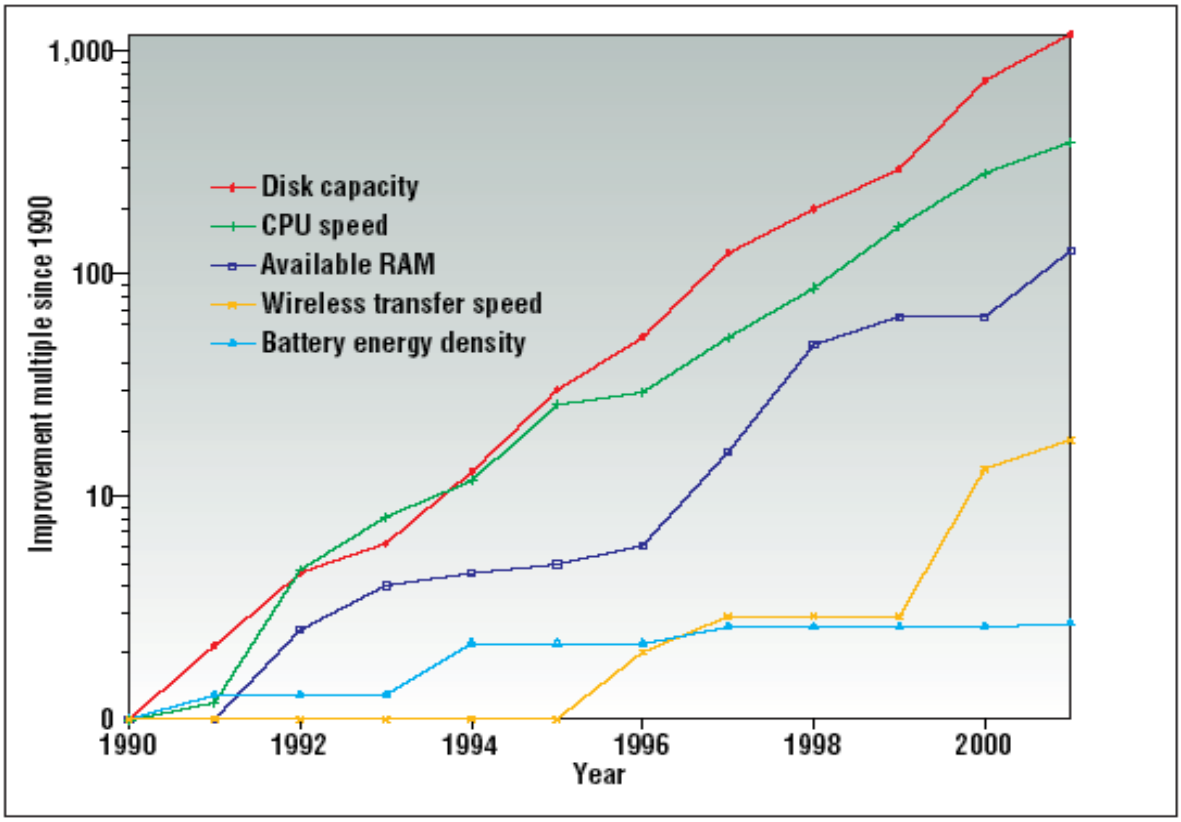

Figure 1: Improvement of computer components during the 1990s [1].

This illustrates how precious the energy stored in a battery is as applications on mobile devices become more demanding. Software developed for mobile devices needs to consider how efficiently the application uses energy in mobile applications. Work has been done by several researchers to help improve energy efficiency [2,3,5]. Graphics algorithms are typically evaluated on how long it takes to render a high quality image. Chang and Lohrmann contrasted the energy efficiency of several data structures used in ray tracing on a Central Processing Unit (CPU) and a Graphics Processing Unit (GPU) [9,10]. Their work focused on selecting the most efficient acceleration structure based on scene complexity, screen resolution, and processor unit type. Their final images rendered in all cases were of the same quality. Additional energy savings could be achieved if the user is willing to sacrifice image quality slightly.

1.1 Goal of the Thesis

This thesis was motivated by interest in investigating whether there may be reasonable tradeoffs between image quality and energy savings. Photon mapping was chosen as the rendering method of choice. Not only does photon mapping generate photorealistic images, it can efficiently render advanced effects like caustics, subsurface scattering and participating media. Some examples of these advanced effects are shown in Figures 2 and 3. Photon mapping has a variety of settings that can be analyzed in terms of their impact on execution time and energy efficiency. The photon mapping algorithm was implemented in ENCORE, a global illumination renderer used for the tests.

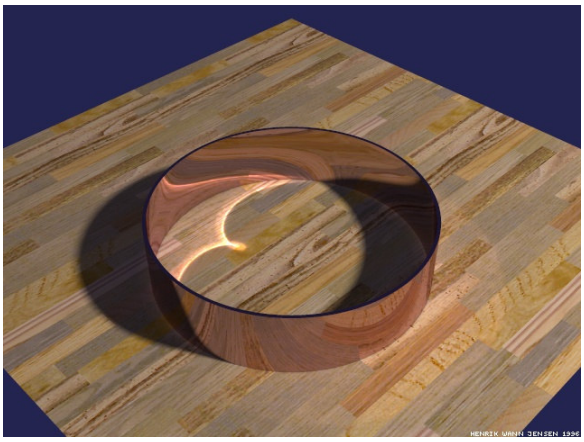

Figure 2: An example of a caustic rendered by photon mapping [4].

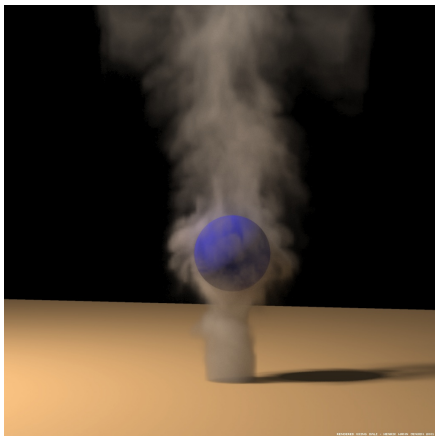

Figure 3: An example of a participating media (smoke) rendered by photon mapping [4].

The first step was to identify the main settings that most adversely affect energy consumption. The second step was to compare how changing these settings impacted image quality. Finally a comparison is made between settings to determine what has the most favorable tradeoff between image quality and energy savings. Armed with this information, developers writing a renderer using photon mapping for a laptop (and other energy-limited mobile devices) will be able to select a satisfactory image quality while conserving energy.

1.2 Organization of the Thesis

An understanding of the fundamental concepts of global illumination is necessary for the reader to interpret the results and conclusions of this thesis. Chapter 2 outlines these core principles and discusses the advantages and disadvantages of several global illumination techniques. Chapter 2 also outlines why photon mapping is a good choice given the alternatives. Photon mapping is described in Chapter 3, along with an overview of the classic ray-tracing rendering algorithm it extends.

The metrics used in evaluating the time and energy efficiency are presented in Chapter 4. This chapter also describes the scenes used in the tests. Experimental results are shown in Chapter 5 along with a detailed analysis. The general conclusions that are supported by the results are described in Chapter 6. Finally, recommendations for future work are found in Chapter 7.

2 Related Work

Energy consumption research has focused either on general-purpose mobile applications or hardware. Farkas et al. profiled the energy consumption of a Java Virtual machine on a mobile device called the Itsy [5]. Two software profilers have been designed for measuring the energy consumption of applications. PowerScope monitors energy use at the application and process level [2]. It uses a modified version of the Linux kernel and an external multimeter to capture energy measurements. PowerSpy is a software-only profiling tool that measured energy use at the thread level [3]. The energy consumption of network cards has also been studied [6,7]. Tscheblockov measured the energy consumption of desktop graphics cards from different manufacturers using pass-through shunts and a multimieter [8].

Only recently has graphics research focused on optimizing rendering algorithms for energy consumption. Lorhmann explored the energy consumption of several acceleration structures on both the GPU and the CPU for static scenes [9]. Chang looked at how suitable these same acceleration structures were for dynamic scenes [10].

3 Global Illumination Rendering

Global illumination is a lighting concept where light traveling to a point directly from a light source is combined with light reflected from one or more other points to estimate the total light reaching that point. The most realistic images a computer simulation can generate must implement some global illumination technique. Namely, the simulation must accurately model the physical process of light scattering throughout a scene. Considering that an image is a two-dimensional grid of pixels, each representing the color of light reaching that point on the image plane, lighting is arguably the most important aspect of image rendering.

3.1 Representing Light

Light travels in a particle form called a photon. These photons originate from light sources that emit them into the scene. A light bulb, a television, and the Sun are common examples of light sources.

A *point light* is a simple model of a light source that is represented as a point in space. This type of light has no analog in reality but is frequently used in graphics as a simplistic abstraction. An *area light* is a planar surface where every point on the surface emits light. It is comparable to a fluorescent ceiling light with a translucent glass covering.

In its most basic form, a photon in a simulation is represented by a point in Cartesian space (X, Y, and Z components) and a triplet of color values (R, G, and B components) representing the power the photon's light. Further details on representing light within the photon mapping algorithm can be found in Chapter 4.

3.2 Surface Properties

When a photon hits a surface some amount of the photon's power is absorbed by the surface. The photon then splinters into several new photons, each of which hold some fraction of the remaining, unabsorbed power. These new photons are then reflected to different regions of the scene. The travel vectors and the proportion of light reflected with the new photons are dependent on the type of surface.

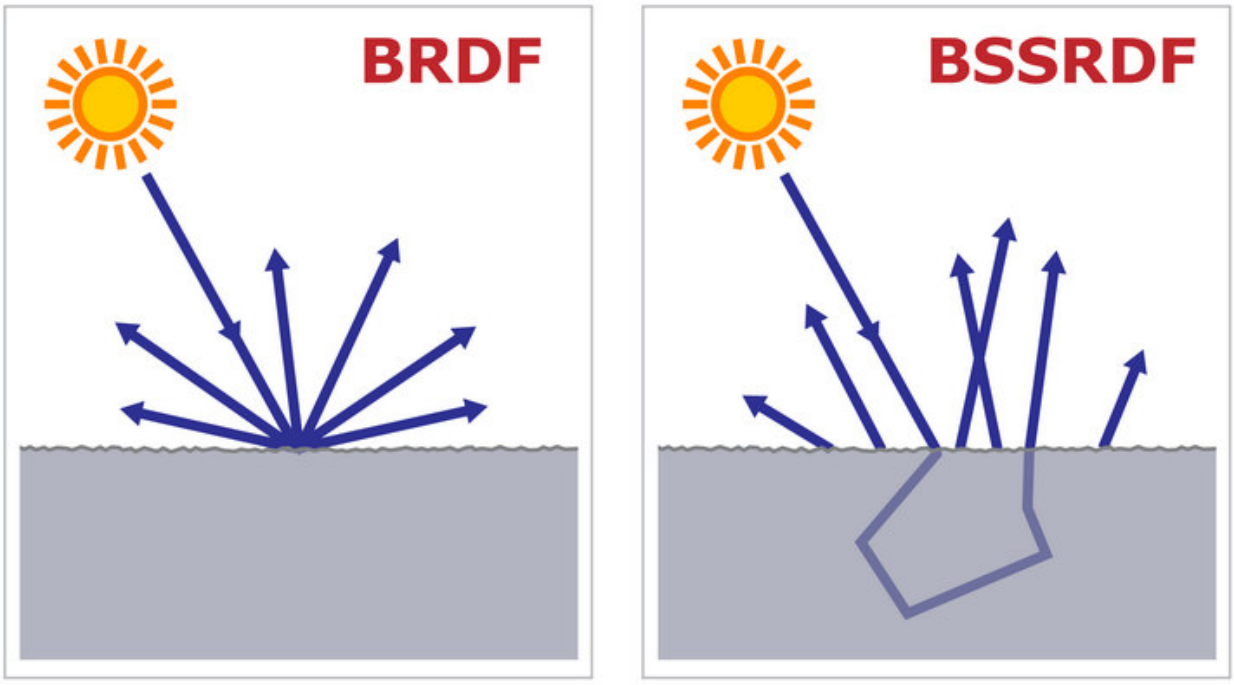

Figure 4: A visual example of light reflecting according to a BRDF and a BSSRDF [11].

Photon-surface interactions can be much more complex. Some object surfaces allow photons to enter and scatter within before finally exiting the object as newly reflected photons. This phenomenon is called subsurface scattering. [12,13] are two papers that discuss this concept. Human skin, a surface that otherwise looks like dull flesh-colored plastic, can be modeled realistically with subsurface scattering. Subsurface scattering is a complex technique that could not be investigated adequately within the bounds of this thesis.

The entire process of photon-surface interaction can be described by a Bidirectional Scattering Surface Reflectance Distribution Function (BSSRDF) [14]. A Bidirectional Reflectance Distribution Function (BRDF) is a more specific BSSRDF where it is assumed that light does not enter the surface and is scattered at the point of the original interaction [14]. Figure 4 is a diagram how a photon interacts with a surface when using a BRDF or a BSSRDF.

There are two different kinds of reflection: diffuse reflection and specular reflection. Most surfaces in the world reflect light both diffusely and specularly, although to varying degrees. Diffuse and specular surfaces are described in sections 3.2.1 and 3.2.2 respectively. Some surfaces allow photons to enter and pass through without being reflected within the object. Such a surface is transparent and its properties are discussed in section 3.2.3.

3.2.1 Diffuse Surfaces

Incoming light is reflected by a diffuse surface in all directions, regardless of the incident vector (the direction prior to the surface interaction). A Lambertian surface is one that scatters light equally in all directions. Figure 5 shows two Lambertian spheres. Notice the smooth shading across the sphere—this is the result of reflecting light evenly.

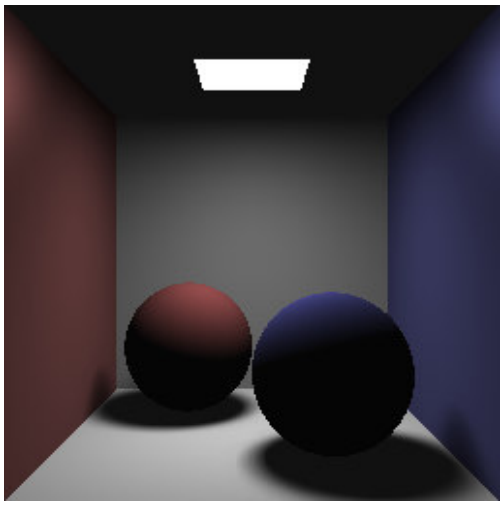

Figure 5: A scene with two spheres, each with a Lambertian surface.

3.2.2 Specular Surfaces

Specular surfaces reflect incoming light at an angle equal to the incident angle. A mirror is a perfectly specular object in that it reflects all light in this manner. Many specular surfaces have imperfections so the angle of reflection tends to vary slightly. These variations show up on specular surfaces as a small circular highlight, such as the two spheres in Figure 6.

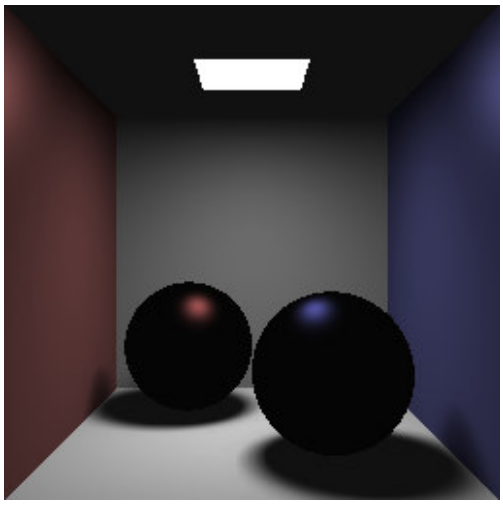

Figure 6: A scene with two specular spheres.

3.2.3 Transparent Surfaces

Transparent materials allow light to pass through and depending on the degree of transparency can distort the direction of the light. This distortion is called refraction.

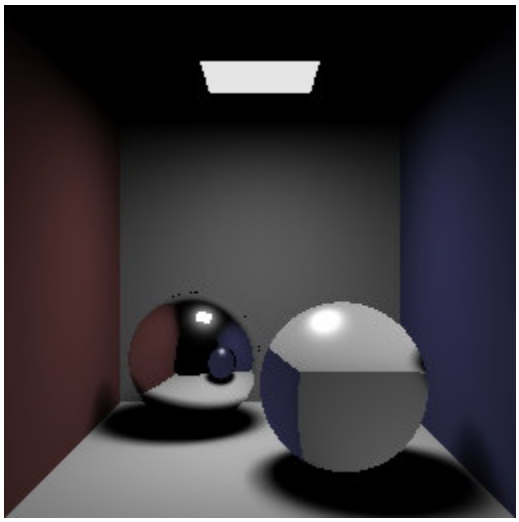

Figure 7: The right sphere has a transparent material.

In Figure 7 there is a glass sphere that has bent light passing through it in a way that the blue wall on its right is actually visible on the left side of the sphere. The degree that light is refracted is determined by *Snell's Law*, shown in Equation 1.

$$\frac{\sin(\theta_2)}{C_2} = \frac{\sin(\theta_1)}{C_1}$$

Equation 1: Snell's Law

Snell's Law states that if the angle of incidence is θ_1, then the angle of refraction will be θ_2, given C_1 and C_2 which are the speeds of light through the first medium and the second medium, respectively [15]. Figure 8 depicts Snell's Law as a light ray is refracted by entering a medium where the speed of light is slower.

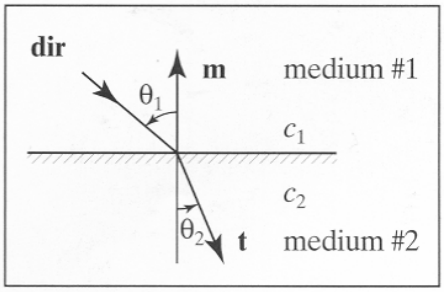

Figure 8: Light refracted as it leaves medium #1 and enters medium #2 [15].

3.3 Approximating Global Illumination

In reality, far too many photons are emitted from any single source that makes it impractical to simulate every photon. Thus global illumination algorithms approximate reality to generate an image. The success of a global illumination technique is determined in part by how accurately it approximates reality. A common way to begin approximating the illumination in a scene is to separate the calculation into two parts: direct illumination and indirect illumination.

3.3.1 Direct Illumination

A point on a surface is considered directly illuminated by a light source if that point has unobstructed line-of-sight to a point on the light source. If the line-of-sight is obstructed, then the surface point is shadowed. The degree that a surface point is shadowed is dependent on the type and number of light sources in the scene.

For instance, if the line-of-sight from a surface point to a *point light* source is obstructed, that light source contributes no direct light to the surface point. On the other hand, if some (but not all) points on an area light have an unobstructed path to the surface point then that surface point is partially shadowed. Figure 9 demonstrates a scene lit by direct lighting where the light source is outside a window.

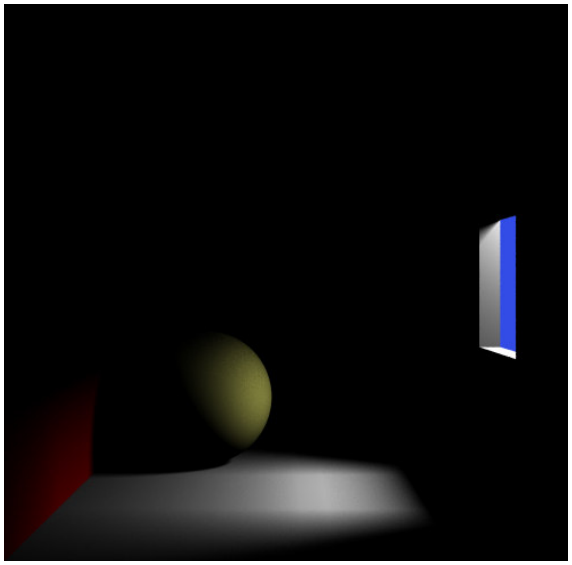

Figure 9: An image rendered with only direct lighting using photon mapping [16].

Soft shadows are rendered during the direct illumination stage of rendering. Soft shadows are partially lit regions on a surface that go from stark-black shadow to a region that is

completely lit by an area light. The soft shadows in Figure 9 are on the floor and the red wall on the left.

3.3.2 Indirect Illumination

The majority of light that enters our eyes is from indirect illumination, sometimes called ambient light. Indirect illumination is the result of photons scattered by surfaces many times after being emitted from a source. Global illumination techniques are largely separated by the method used to calculate indirect lighting.

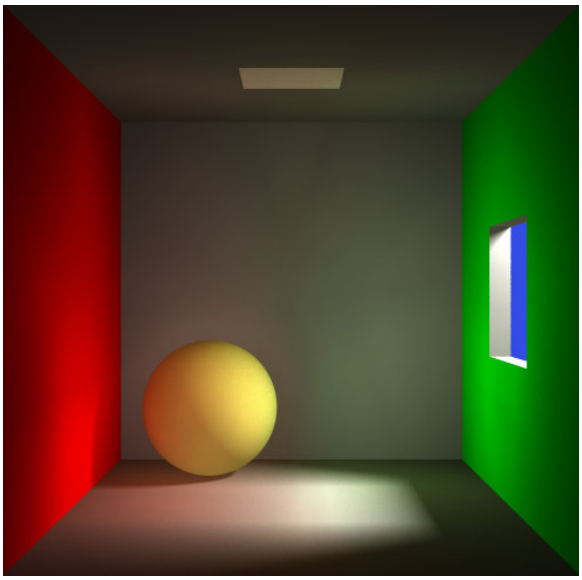

Figure 10: An image rendered with direct and indirect lighting using photon mapping [16].

Figure 10 demonstrates indirect lighting brightening the scene. Note that the majority of the scene is lit indirectly. Also notice the red and green color "bleeding" onto the gray wall. Proper indirect illumination generates this desirable effect.

3.4 Global Illumination Techniques

Several global illumination techniques have been developed over the years to handle different reflection models (BRDFs) or improve rendering efficiency. This section briefly describes the most common approaches and the main limitations they have.

3.4.1 Monte-Carlo Ray Tracing

Monte-Carlo methods are sampling techniques used to estimate a solution to complex mathematical problems, such as an integral. A Monte-Carlo technique relies on the fact that a solution can be found with enough samples. Ray tracing can be extended to use Monte-Carlo sampling to estimate the amount of indirect light arriving at a point in space. Furthermore, a Monte-Carlo ray tracer can simulate an arbitrary BRDF and can render diffuse and specular reflections for any type of geometry [17].

Since Monte-Carlo ray tracing is a stochastic algorithm, images can have considerable variance from pixel to pixel. This noise can be reduced by increasing the sample size; however this also increases the runtime of the algorithm.

3.4.2 Radiosity

Modern radiosity methods are efficient at generating high quality images with a few limitations. When radiosity was originally developed it could only handle scenes with diffuse surfaces [18,19]. Since then radiosity has been extended to support different reflection models [20]. The general idea behind radiosity is to subdivide all surfaces in a scene into small patches and then solve a set of equations that represent the transfer of light between these patches.

One major hindrance to radiosity is that its performance is coupled to the complexity of the geometry in the scene. Subdividing complex models that may have tens of thousands of polygons (or more) is computationally expensive. Complex models approximate real geometry. Curved surfaces can be tessellated into triangle-strip representations. Care must be taken with the tessellation because lighting artifacts (visual incongruities) can occur around shadow boundaries [17]. Additionally, radiosity does not render specular reflections on curved surfaces well [17]. There have been several hybrid approaches that address this limitation by combing radiosity for indirect lighting with Monte-Carlo ray tracing [21]. However, these techniques are still limited by the tight coupling between radiosity and geometry.

4 Photon Mapping Overview

Photon mapping can be used in conjunction with Monte Carlo ray tracing to increase a renderer's efficiency while maintaining high image quality. Photon mapping models the emission and scattering of photons throughout a scene in a preprocessing step. These photons are stored in a data structure called the photon map. Rendering is done using a Monte Carlo ray tracer that uses the photon map to determine the illumination in regions of a scene.

The photon map is a special data structure that is decoupled from the geometry in the scene. This means that, unlike radiosity, photon mapping can be used with geometry of any complexity. Photon mapping techniques can also be used to implement effects like subsurface scattering [22] and participating media [23]. Shadows computed by direct lighting can also be made more efficient by applying photon mapping [24].

Photon mapping is a stochastic process in that each photon represents a single simulated light particle. Therefore as the number of photons stored in the photon map increase, the better the approximation of light distribution becomes. This also means that more memory and time is needed. However, photon mapping still significantly outperforms pure Monte-Carlo ray tracing.

The most common use for a photon map is to estimate the illumination due to indirect lighting in some region of space. The process to create a photon map is described in section 4.1. Specialized photon maps can reduce the cost of rendering expensive effects like caustics or soft shadows. Section 4.2 lists the most common types of photon maps. The following section describes an operation called the photon gather. A photon gather is the primary operation that uses the photon map. Finally, section 4.4 describes the basics of ray tracing and how a photon map can be elegantly combined with a Monte-Carlo ray tracer.

4.1 Photon Map Creation

Creation of the photon map is best explained in stages. The first stage is when every photon in the scene has been emitted, scattered and then stored in a list. In the next stage this list of photons is organized in a spatial data structure called a K-D tree.

4.1.1 Photon Scattering

Photon scattering simulates the transport of light throughout the scene. As mentioned previously in 3.1, the most basic photon representation is a point in space (X, Y, and Z coordinates) and a triplet of color values (R, G, and B components). Additionally, the photon may also store the normal of the surface that absorbs the photon, and its travel direction prior to hitting the absorbing surface. This information can be used in several parts of the renderer to further enhance an image.

All photons scattered in a scene originate at a light source. Photon emission is the process of choosing an initial direction and an initial position of the photon. Both the direction and position are determined by the type of light source. The position of a point light source is used as the initial position of all photons emitted from the source. A point light source also emits light equally in all directions. Hence, choosing an initial direction from this source is simply done by choosing a random direction.

The choice of a random direction is sufficient since many thousands, and possibly millions, of photons are emitted into a scene. Consider each photon as a sample of one way a light particle is scattered. As the number of samples increases the better the photon map represents the illumination in a scene.

Light sources other than a point source can be simulated by photon mapping. Figure 11 shows several light sources and the initial directions of emitted photons. Emitting a photon from a spherical light is done by first choosing a random point on the sphere's surface as the initial position. The initial direction is parallel to the path from the center of the sphere to the initial position. A square light, a type of area light, emits a photon from a random point on the surface and then a random direction within 90 degrees of the light's normal.

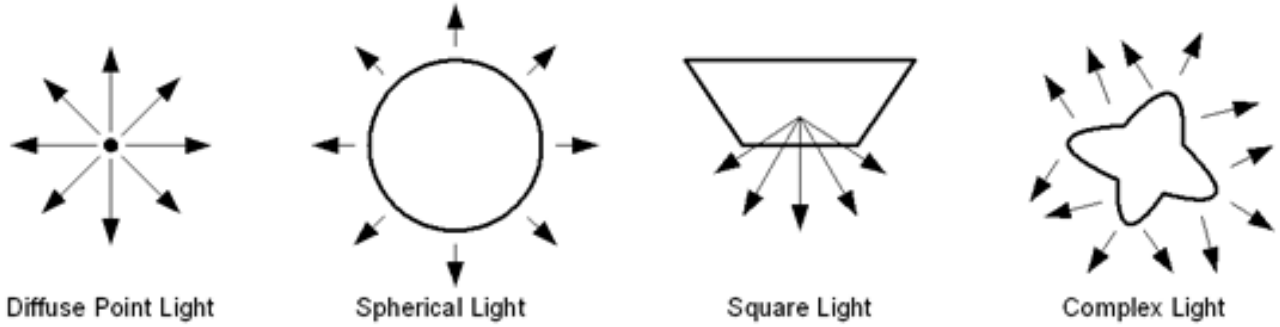

Figure 11: Several light sources and the directions they emit photons [16].

The amount of light a single photon transports is assumed to be the same regardless of which light source emits it. This simplification makes it much easier to estimate illumination later. However, this abstraction impacts the number of photons each light emits. In a scene with more than one light source, the number of photons emitted by a specific source must be proportional to the light's power relative to the other sources in the scene. For instance, if the brightness of each light source is the same, then each light source should emit about the same number of photons. If one light is brighter than the others by some percentage, then that light should emit more photons equal to that percentage.

Once an initial position and initial direction have been chosen for the photon, the simulation finds the first surface that the photon's path intersects. A technique called Russian Roulette is then applied at this surface interaction [17].

In reality when a photon interacts with a surface, the original photon is transformed in several ways at once. Some portion of the photon's power may be transformed into many other photons that are reflected in different directions. Another part of the photon may be transmitted through the object it hits if its surface is transparent. Modeling this behavior exactly would make this algorithm exponential in time complexity. Furthermore, every surface interaction would result in the creation of photons with reduced power. This would violate the assumption that every photon is of equal power. Russian Roulette allows us to do both: model a single photon path and maintain a constant photon power.

The standard Russian Roulette algorithm used by ENCORE is outlined in Figure 12. The main idea is to probabilistically determine whether a photon is reflected diffusely, specularly, transmitted through the object, or absorbed at the surface. The probability that any of these events happens is based on the surface's properties which determine a weight assigned to each event. A random number E is generated between 0 and the sum of those weights. Figure 12 demonstrates how E is compared to a range of those weights. For instance, if E is less than the specular weight, the photon is specularly reflected. If E is less than the sum of the diffuse and specular weight, yet larger than the specular weight it is diffusely reflected.

```
while( bounces < maxBounces )
{
   HitInfo surfaceHit =  FindFirstIntersection(photonRay);

   if( photon hit a surface )
   {
     float pS = surfaceHit->GetSpecularWeight();
     float pD = mat->GetDiffuseWeight();
     float pT = mat->GetTransmittanceWeight();
     float pTotal = pS + pD + pT;
.
     float E = randNumber(0, pTotal);

     if(E < pS)
        photonRay = ReflectPhotonSpecularly(...);

     else if(E < (pS + pD))
        photonRay = ReflectPhotonDiffusely(...);

     else if( E < pTotal)
        photonRay = TransmitPhoton(...);

     else
        // terminating loop

        // Note: photon will be absorbed if recent surface
        // is diffuse
        break;
   }
```

Figure 12: The Russian Roulette algorithm [17] as implemented in ENCORE.

After the reflection is determined a path is generated based on the reflection. Once the path is calculated the next surface interaction is found by finding the first intersection along the path. This continues until either the photon scattering is absorbed or an upper bound on the number of surface interactions is reached. In Figure 12 the upper bound is called `maxBounces`.

When a photon is absorbed by a surface it is stored in the photon map. One can also choose to store a photon in the photon map at each diffuse surface interaction as this will speed up photon scattering. It is important to note that photons can only be absorbed by surfaces with a diffuse component. If a photon's path ends at a surface that is purely specular then the photon is discarded instead of being stored in the photon map.

4.1.2 Photon Map Construction

Once all of the photons are emitted and stored in the photon map, the photon map must be constructed. Prior to construction, the photons are stored in an unsorted list. Since photons are searched for in the photon map based on their position in 3-dimensional space, a one dimensional list will be very slow. Instead, a data structure called a balanced K-D tree is used to increase the speed of searching the photon map [17].

A balanced K-D tree is a spatial data structure meaning that it organizes data based on a position in space. The photon map uses a form of the K-D tree that splits the data at a median point along one of the three Cartesian axes (X, Y, or Z).

Some implementations scale the power of all photons by a constant amount based on the total number of photons stored before constructing the photon map. This can be useful in maintaining a consistent level of stored illumination between photon maps that have stored a different number of photons [17].

4.2 Types of Photon Maps

Several different photon maps can be used to enhance different parts of the rendering step. The difference between these photon maps is the path the photons took before being stored. The basic photon mapping algorithm uses a single photon map called either the *global photon map* or the indirect photon map. The photons stored in the global photon map, sometimes referred to as global photons, have been absorbed by surfaces with a diffuse component.

The *caustics photon map* stores photons that have been reflected from a specular surface onto a diffuse surface. Any number of surface reflections can occur prior to the specular-to-diffuse series. The caustics photon map is used when rendering caustics.

A *shadow photon map* stores photons that represent regions of shadow. These ‘shadow’ photons are created any time a global photon hits a surface. Shadow photons are stored at the location the original photon would hit had it not been for its first intersection. The shadow photon map is used to speed up direct illumination.

A *direct photon map* stores photons at the first surface interaction after being emitted from a light source. Photons with two or more surface interactions are not stored in this photon map. The direct photon map can be used in conjunction with the global photon map when computing indirect illumination using final gathering.

4.3 Using the Photon Map

The primary operation on a photon map is called a *photon gather*. A photon gather estimates the irradiance (incoming light) at a point in space. Consider a point *x* in space where one wishes to estimate the irradiance as shown in Figure 13. A photon gather finds a set of photons within some maximum distance (called the *gather distance*) from *x*.

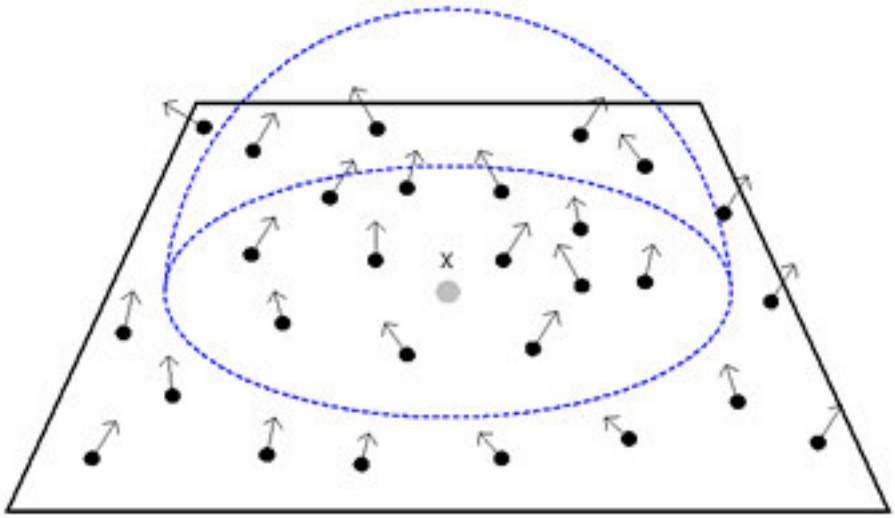

Figure 13: A typical photon gather. The blue hemisphere represents the volume of space that contains the photons that may be used [17].

Once the photons are collected, the power of the photons is summed. This sum is then multiplied by a value estimating the density of the photons within the volume defined by *x* and the *gather distance*. The product of the power sum and the density estimate is the value that represents the irradiance at *x*. Figure 14 is an example of the photon gather algorithm.

Precomputed irradiance is an optimization used to increase the speed of photon gathers during rendering [25]. When hundreds of thousands of photon gathers are performed and a photon map only has 100,000 photons, many photon gathers will be redundant calculations.

Precomputed irradiance works by choosing the position of every n^{th} photon as the location to perform a photon gather. The resulting estimate is stored as the power of a single photon which is then stored in a new photon map. After the new photon map is constructed, only the single closest photon is gathered. This photon's power is then used as the irradiance estimate.

```
List< Photons > photonList =
   GetNearestNPhotons(gatherCount, gatherDistance, surfaceHit, surfaceNormal);

if( photon list is empty )
{
   // no photons nearby, surfaceHit is not illuminated!
   return Color(0, 0, 0); // black
}

// get the radius to the furthest photon
farDistance = GetFarthestPhoton( photonList, surfaceHit );

R = 0, G = 0, B = 0;
for( each photon p in photonList )
{
   Color power = p->Power();

   R += power.R();
   G += power.G();
   B += power.B();
}

float densityEstimate =  (1/PI) / (farDistance  * farDistance);

R *= densityEstimate;
G *= densityEstimate;
B *= densityEstimate;

return Color(R,G,B);
```

Figure 14: Pseudocode of a *photon gather* operation based on the implementation in ENCORE..

4.4 Rendering

Rendering follows immediately after the photon map has been created. The color of each pixel is calculated by using a ray that samples the scene to be rendered. This process is called ray tracing and is explained in 4.4.1. The direct and indirect lighting calculations are discussed in 4.4.2 and 4.4.3, respectively. Section 4.4.4 briefly looks at how caustics are generated.

4.4.1 Basic Ray Tracing

Ray tracing is a point sampling technique where the color of each pixel is determined by a tree of rays that are used to sample a scene. A primary ray is the first ray in the tree. Each primary ray represents the reverse path light travels from a scene through a pixel on the image plane. The earliest form of ray tracing, known as ray casting, only traced rays from the eye into

the scene. Subsequent improvements, starting with Whitted ray tracing, spawned reflection, refraction, shadow and other kinds of rays at the intersection of the primary ray and scene geometry [26].

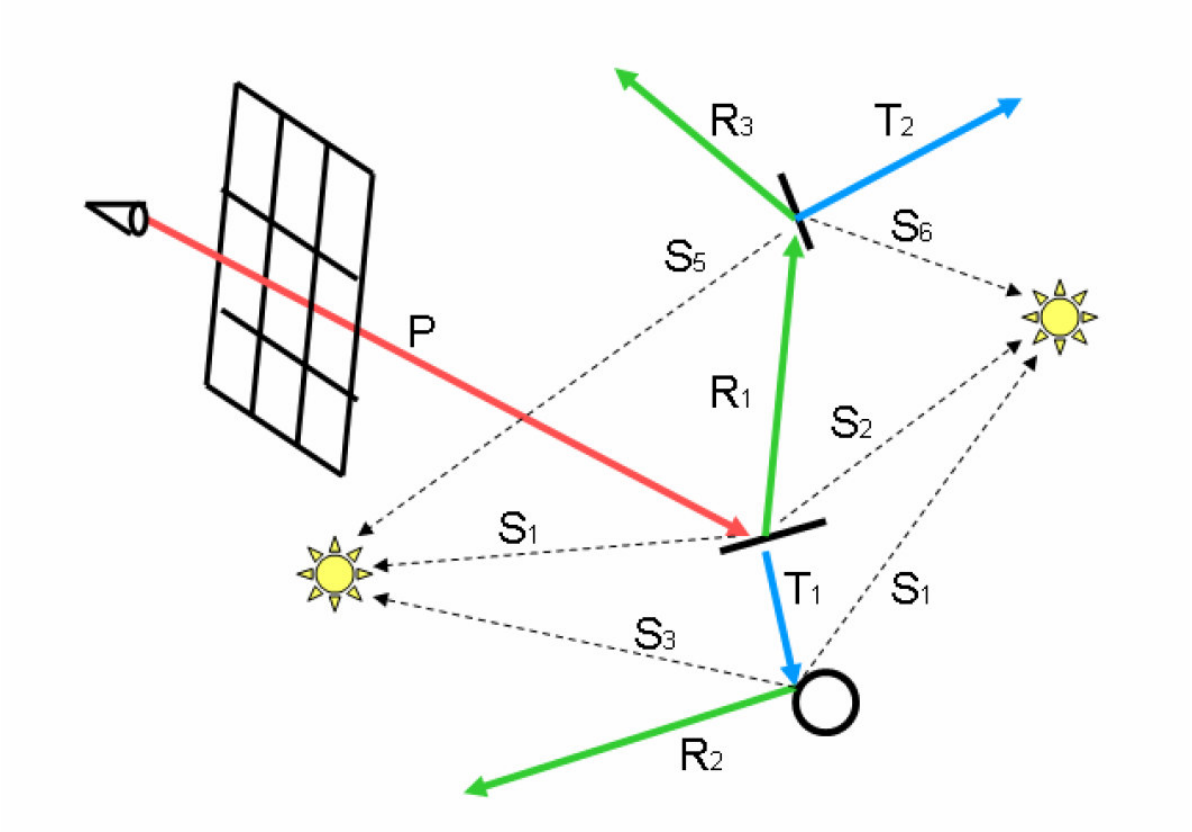

Figure 15: A diagram of the basic ray tracing algorithm [9].

Figure 15 is a diagram of ray tracing computing direct lighting. The red ray *P* is a primary ray. The origin of all primary rays is a point in space acting as the location of the image viewer's eye. The black grid in Figure 15 represents the image and each cell is an individual pixel. The primary ray is used to sample the first intersection point the ray has with scene geometry (represented by the black line at the end of ray P).

The rays that are spawned at the intersection with the primary ray are called secondary rays. These are used to further sample the scene in different ways. Shadow rays determine whether a light source directly illuminates that point. These lines are represented by dotted lines to the two light sources that look like yellow suns.

The green rays represent reflection rays. If a surface is reflective, a ray that follows the reverse path of light arriving at this surface is generated. R_1 in Figure 15 is one such reflection

ray. R_1 happens to intersect another surface in the scene. The light calculations at this intersection are identical to the ones that are performed for the intersection of the primary ray. The light calculated by tracing a reflection ray is then summed with the light calculations at the point in the scene that *P* intersects. Reflection rays are recursively traced to some user-configurable maximum depth. This helps avoid situations where many surfaces may be reflective and the recursion could continue indefinitely.

The blue rays represent transmitted rays. These rays are traced through surfaces if the surface is transparent and allows light to pass through. Usually these rays are refracted because of the difference in the speed of light through the surface's material as opposed to air or a vacuum. These rays terminate at the first intersection after exiting the material it is transmitted through. At this intersection point, the light is calculated in the same way as primary and reflection rays.

```
for( each row of pixels )
{
   for( each column of pixels )
   {
      Color pixelColor; // Note: pixelColor is initially black
      // get sample rays
      List< Rays > sampleEyeRays = GetEyeRaySamples( samplesPerPixel, jitter )

      for( each sample )
      {
         HitInfo hit = FindFirstIntersection( aSample)

         if( sample ray hit a surface )
         {
            pixelColor += CalculateRadiance( hitSurface );
         }
      }

     // average the pixel color of each sample
     pixelColor /= samplesPerPixel;

     // store pixel for image display later
     SavePixel( pixelColor, row, column )
     }
   }
}
```

Figure 16: Pseudocode of the ray tracing algorithm as implemented by ENCORE.

Figure 16 shows the pseudocode for generating primary rays and sampling the scene. Multiple primary rays can be spawned per pixel to improve image quality. This is known as

super-sampling. Each color sampled by a primary ray is averaged together to yield the final pixel color. So that each primary ray doesn't sample the same point in the scene, a small amount of noise, or jitter is added to this primary ray's direction.

The interested reader can find more details on ray tracing in [27] and [28].

4.4.2 Direct Lighting

The discussion of ray tracing above provides a high-level view of the direct lighting calculation. Direct lighting is explained in depth in this section for completeness.

The contribution to a pixel from direct lighting is equal to the sum of five different components. The diffuse component is the amount of light reflected diffusely by a surface. Similarly, the specular component is the amount of light reflected specularly by a surface. Both of these can be calculated at the same time.

```
// shadow photon optimization
bool completelyShadowed = QueryShadowPhotonMap();

if( completelyShadowed == false )
{
   for( each light in the scene )
   {
     samplePoints = light->GetSamplePoints();

     for( each sample point )
     {
        if(IsInShadow() == false)
        {
           diffuseColor  += CalculatePhongDiffuse(...);
           specularColor += CalculatePhongSpecular(...);
        }
     }

     // average the samples together
     diffuseColor  /=  samplePoints.size();
     specularColor /=  samplePoints.size();
   }
}
```

Figure 17: Pseudocode for diffuse and specular calculations for direct lighting.

Figure 17 is the pseudocode for these components. Note that this is where the shadow photon map is used to help optimize the calculation of soft shadows. Also note in Figure 17 that multiple sample points are used for each light source. This is necessary if the light source is not

a point light source where only a single sample point is needed. The Phong lighting model is used to calculate diffuse and specular surfaces in this thesis [29].

Two other direct lighting components are reflection and refraction. These components were adequately described for the purposes of this thesis in 3.3.1. Hill provides more specifics [15].

The final component captures a property of surfaces called emissiveness. Some properties add color to a scene without being a light source explicitly. The emissive color of a surface is simply added to the sum of the direct lighting components. This feature is implemented in the photon mapping renderer used by this thesis; however the scenes used for the tests do not contain any emissive surfaces.

4.4.3 Indirect Lighting

All indirect lighting using photon mapping involves the photon map. Indirect lighting using the photon map can be done in two different ways: direct visualization and final gathering.

In direct visualization, a photon gather is performed local to the point where one wants to calculate the indirect lighting. This is fast as only a single photon gather is used and no additional rays are necessary. The tradeoff is that variance is high between different photon map settings. Normally direct visualization requires a high number of photons stored in the map. The exact number is dependent on the scene, however 500,000 photons or more is not uncommon. If the photon map has too few photons the illumination across a flat surface may vary from too dark to too bright when there should be a smooth gradient.

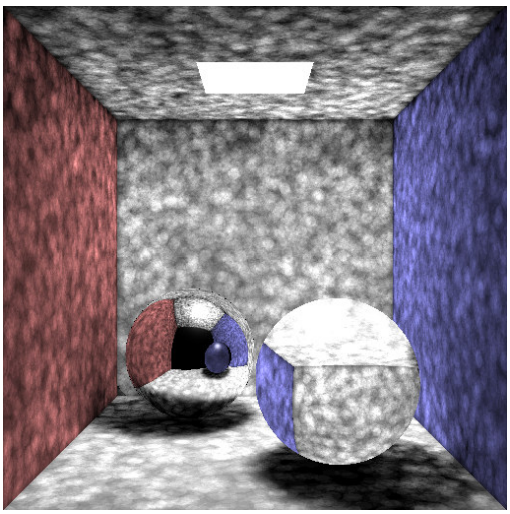

Figure 18: Direct Visualization using 50k photons

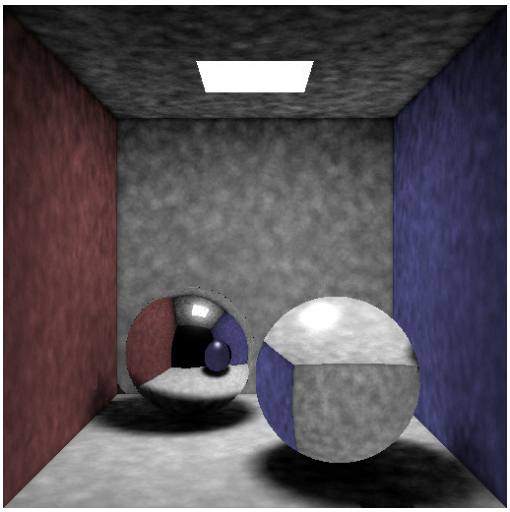

Figure 19: Direct visualization using 250k photons

Figure 18 and Figure 19 show the same scene rendered with a different number of photons stored in the global photon maps. There is a high amount of noise in Figure 18 when only 50,000 photons are used. Figure 19 increases the total number photons used by a factor of 5. The gradient of color across the walls in Figure 19 is still not very smooth but it is improved compared to Figure 18. The noise seen in Figure 18 can be reduced by increasing the *gather distance* or adjusting the *gather count.* However, this is still not going to yield images as good as those with higher photon counts.

Final gathering is the second method and generally yields the best results. It is a technique borrowed from Monte Carlo ray tracing. The main idea builds on the fact that the indirect illumination at a point *x* on a surface is the result of light bouncing off many surfaces. Instead of performing a photon gather local to *x*, photon gathers are performed at points above *x.* These points are found by sampling a hemisphere above x using rays. These rays are called final gather rays. Figure 20 shows from two angles a visualization of final gather rays. The point for which indirect illumination is being calculated is blue.

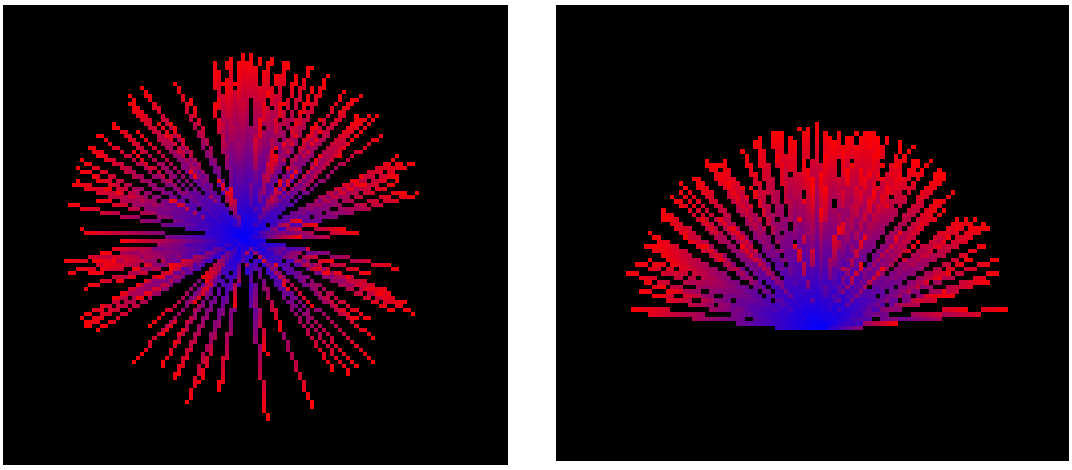

Figure 20: A visualization of final gather rays. Left image is a top view. The right image is a side view.

Many final gather rays need to be used for a satisfactory estimation of indirect lighting. Jensen explains that anywhere from 200 to 5000 final gather rays are needed [17]. Since a final gather needs to be computed at least once for every pixel, final gathering is very expensive. This cost can be reduced by using an optimization called irradiance caching that was developed for ray tracing [30]. The irradiance cache speeds up final gathering so significantly, it is usually required if one wishes to compute an image efficiently.

Irradiance caching exploits the fact that across a surface, the illumination usually changes gradually. The irradiance cache stores previously computed final gather values. Since illumination changes gradually across the surface, these cached values can be used to interpolate the illumination. Whether the values can be used depends on the value's associated weight. The weight is calculated based on, among other things, surface normals and the distance to the point where illumination is being calculated. This weight is compared to a user-configurable variable (named *tolerance* in this thesis) that corresponds to the amount of error one allows into the image. If the tolerance is high, more values can be reused and rendering speed increases. However, this performance gain comes at the cost of reduced image quality.

Irradiance gradients is another optimization technique used to improve image quality [31]. It is used in conjunction with irradiance caching. This feature is not currently implemented in the ENCORE renderer and therefore will not be tested.

4.4.4 Caustics

Caustics are effects generated by the focusing of light through one means or another. Typically objects that refract light create caustics. Light on a table after being refracted through a glass is a common caustic. Another common caustic is the light on the bottom of a pool after being refracted through the pool's water. Figure 21 shows a caustic generated by light refracted through a glass sphere. Caustics are rendered by directly visualizing a caustics photon map.

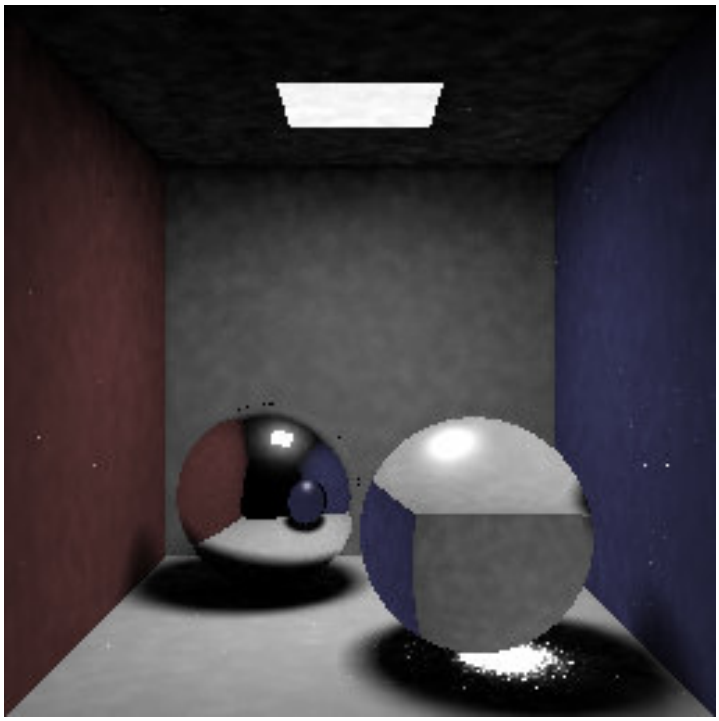

Figure 21: An image generated by ENCORE with a caustic present beneath the transparent sphere.

5 Experimental Setup

The goal of this thesis is to establish and rank tradeoffs between the energy consumed and image quality for renderers that use photon mapping. These tradeoffs can be used to increase the energy efficiency of similar photon mapping-based renderers (photon mappers) on mobile devices. The measurements used to establish these tradeoffs are discussed in section 5.1.

Five scenes were used to examine the impact surface types, total polygon count, and polygon distribution had on aspects of the photon mapping renderer. These scenes are presented in section 5.2.

The concluding section will provide a summary of the computer hardware used to run all of the tests. Knowing these details will not only increase confidence in the final results, but will provide a basis for comparison with future hardware.

5.1 Measurements

This thesis explores the possible tradeoffs by first measuring the time and energy consumed as photon count, irradiance cache tolerance, and other settings are varied. The highest quality image the renderer can calculate is used as a reference for comparing test images. The objective difference between the reference image and a given test image is found using *ltdiff*, an implementation of an image comparison metric developed by Lindstrom [32]. This difference is used to evaluate the degradation of image quality as renderer settings are changed. A relationship between image quality and energy use can then be established. This section explains the way time, energy, and image quality are measured and the role each has in analysis.

5.1.2 Energy

Energy consumption is measured using the Advanced Configuration and Power Interface (ACPI), a standard for querying battery information. ACPI is available to application developers on the Windows XP operating system. Windows updates power information stored by the operating system every couple of seconds. This information can be queried using the system procedure `callNtPowerInformation` [33]. It provides a data structure that contains the discharge rate since the last update.

Joules (J) is the unit of measurement for energy. The rate at which energy is used is measured by Joules per second, or more commonly Watts (W). We find the energy used by multiplying the average discharge rate over a period of time multiplied by the length of that period in seconds. ACPI reports a discharge rate in milliwatts (mW) so when the average discharge rate is multiplied by seconds the energy is in millijoules (mJ). This value is converted to Joules by dividing it by 1000 mJ (since 1000 mJ are in a single J).

An average discharge rate can be calculated by sampling the instantaneous discharge rate every second the photon mapper runs. The total energy for a given test is calculated by multiplying the average discharge rate by the total time elapsed. Confidence in the consistency of the average discharge rate was increased by performing several preliminary tests.

Each preliminary test was run on the test laptop (specified in 5.3) to check the variance of the discharge rate as reported by ACPI. The operating system on the laptop provided several energy-saving features. Some examples included dimming the display or turning off the hard disks when the system was idle. Since these features would alter the environment during a test, they were disabled.

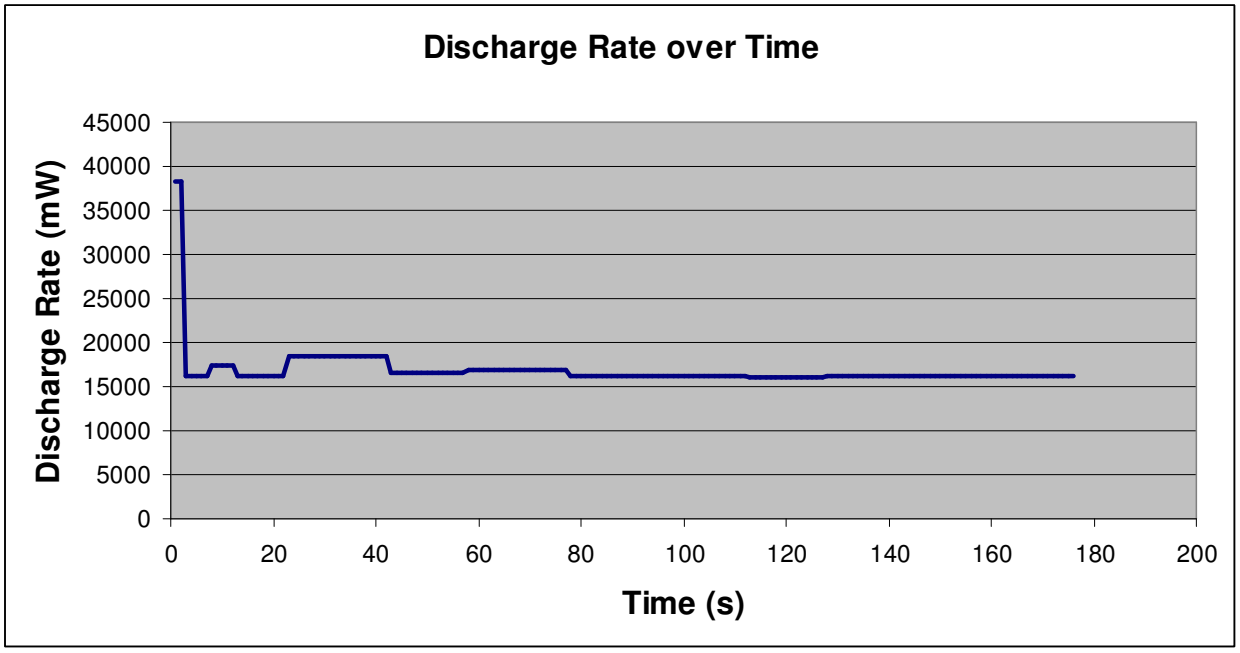

Figure 22: The discharge rate during image rendering.

Figure 22 is a graph of the discharge rate when an image was rendered using direct visualization of the photon map. The maximum reported discharge rate was 38.295 W and the minimum was 16.028 W. The standard deviation was 2.420 W with an average discharge rate

of 16.834 W. The first two samples reported a very high discharge rate and were the only statistical outliers. These high discharge rates may be caused by costs associated with the creation of a new process by the operating system as this behavior was noticed in all of the performed tests.

Excluding those two samples, the next maximum was 18.426 W at 2 seconds into the simulation. Every other sample was within one standard deviation of the average. All of the preliminary tests have demonstrated approximately the same consistency.

Windows updates its power data every 3 to 5 seconds. Given this resolution, one might wonder whether there are significant changes between each sample. This concern is reduced when one considers the total runtime of the typical test is anywhere from 3 minutes to 30 minutes in length. Over this lengthy period of time, any substantial variance in discharge rate would be averaged out.

5.1.1 Time

Execution time is a common metric used to evaluate the performance of applications. The execution time to generate an image is calculated by subtracting an end time from a start time. The start time is recorded before the first photon is emitted. The end time is recorded the moment all pixels have been assigned a color value. Time is recorded within ENCORE using the Windows system routine `GetTickCount` which returns the number milliseconds that have elapsed since the system booted.

Table 1 describes what aspects of the renderer are profiled. Direct lighting, indirect lighting, and caustics are different because they are calculated per pixel. In other words, the elapsed time can only be calculated for a specific pixel. The total time for these different aspects can easily be found by summing the elapsed time for every pixel.

What is measured?	Description
Image Time	The time to create the photon map plus the time to render a complete image.
Photon Map Creation	The time to emit and scatter photons, plus the time to construct the photon map. If precomputed irradiance is performed it is also added.
Precomputed Irradiance	The time to precompute irradiance
Direct Lighting	The time to calculate direct lighting for all pixels.
Indirect Lighting	The time to calculate indirect lighting for all pixels.
Caustics	The time to calculate caustics.

Table 1: Image creation stages profiled by time.

The execution time of a specific calculation used is very useful in conjunction with the energy measurements. Only the total energy consumed per generated image can be measured.

$$E_{DL} = \frac{T_{DL}}{T_{total}} \times E_{total}$$

Equation 2: The calculation to find the energy used by direct lighting.

However, the ratio of the time a specific calculation took versus the total time to generate the image to estimate how much energy that calculation consumed. For example, Equation 2 shows how the energy used by direct lighting can be calculated. T_{total} is the time to render the image and T_{DL} is the time used by direct lighting. E_{total} is the total energy used to render the image and E_{DL} is the energy used by direct lighting.

5.1.3 Image Difference

The quality of images is very subjective when judged by the human eye. For this reason an objective measure was sought. One common calculation to judge relative difference between images is the Root Mean Square (RMS) metric. This has been used mostly because it is intuitive and easy to compute. Unfortunately, RMS has trouble detecting some forms of distortion that the human eye can easily see [34].

A superior metric was implemented as a program called *ltdiff*. It was developed by Peter Lindstrom as a part of his PhD dissertation. Other researchers have found it quite useful with work on level of detail [35]. *Ltdiff* is based in part on human perception and psychology. It reports a unit-less value as a measure of the difference between two images. It also produces a delta image (see Figure 24) that helps a human viewer understand what regions *ltdiff* selected as different. The black regions in the delta image are regions identified as the same in input images. The white regions identify discrepancies between the input images. The brightest regions identify the location of the largest difference.

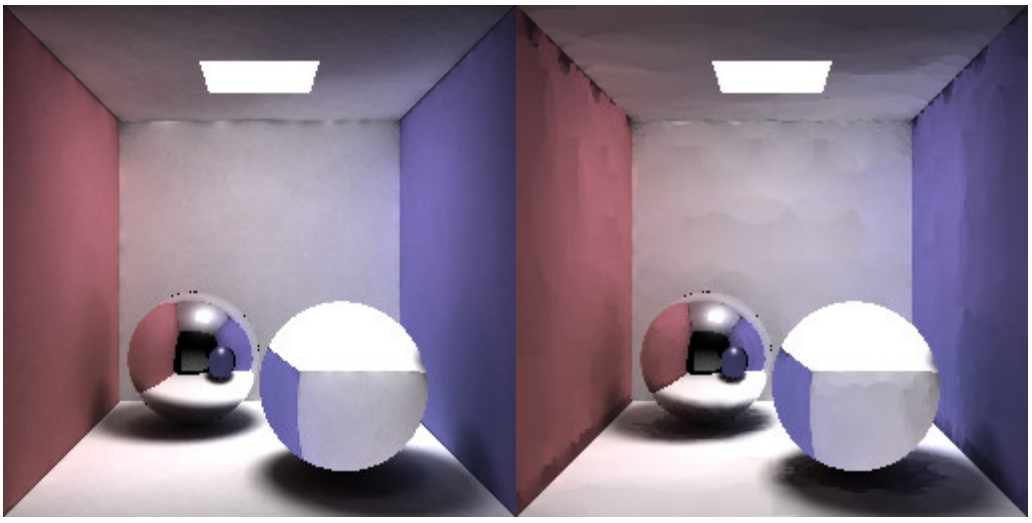

Figure 23: Two final gathered images rendered with ENCORE.
The reference image is on the left. The test image is on the right.

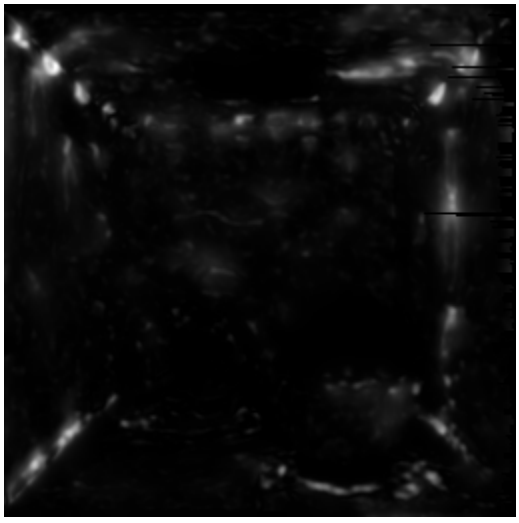

Figure 24: The delta image from *ltdiff* after comparing the images in Figure 23.

All images are compared to a reference image assumed to be the highest quality the renderer can generate. Differences measured by *ltdiff* are assumed to be of lesser quality. Any difference value reported by *ltdiff* could be considered a measure of the degradation in image quality. However, image quality has a subjective connotation so it may be more precise to refer to the measurement as just the image difference. *Image difference* will be the term used in the results section.

A maximum value of 100 was chosen as the limit for an acceptable difference between two images. Having an acceptable maximum provides a better context for discussing image quality. The specific value of 100 was chosen based on experience with numerous image

comparisons using *ltdiff*. The rightmost image in Figure 23 is an example of an image close to this maximum; *ltdiff* reported a difference of 90.86.

5.2 Scenes

All test scenes used a modified version of the Cornell Box. The Cornell Box is a scene with a long history in computer graphics and is commonly used to demonstrate global illumination. It is a square box with missing wall closest to the camera. The rear wall, ceiling and floor typically have the same color, with different colors for the left and right walls. Incarnations of the Cornell box vary by wall color, the models inside, or both. For the test scenes, the surface color of the walls remained the same; only the models within the box were changed.

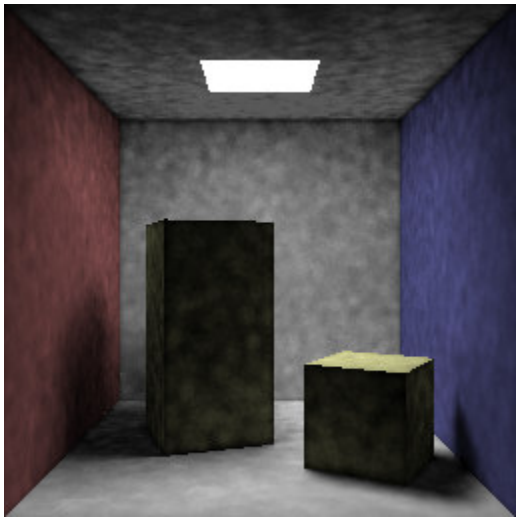

Figure 25: The Cornell Box with two diffuse boxes rendered by ENCORE.

Figure 25 and Figure 26 show the first two scenes. These two scenes account for every surface type (diffuse, specular, reflecting, or refracting) supported by ENCORE. The diffuse box scene, shown in Figure 25, contains two yellow diffuse boxes where one is taller than the other. All of the surfaces in this scene are diffuse and no surfaces are transparent or reflective. Direct illumination is stressed more here than other scenes because the boxes cast large shadows throughout the lower half of the image.

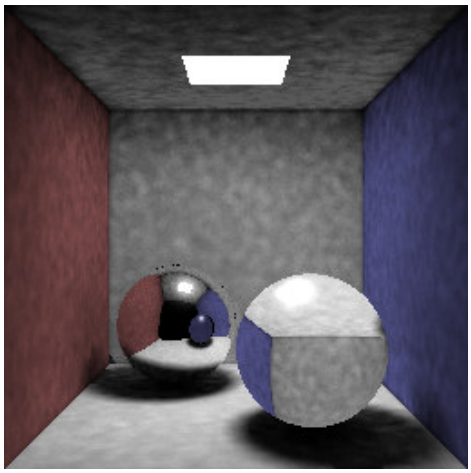

Figure 26: The Cornell box with two specular spheres rendered by ENCORE.

Figure 26 shows the specular sphere scene. It replaces the boxes with two specular spheres. One sphere reflects light perfectly, similar to a mirror. The second sphere represents a glass ball that refracts rays and photons. The glass sphere will also generate a caustic if that effect is turned on.

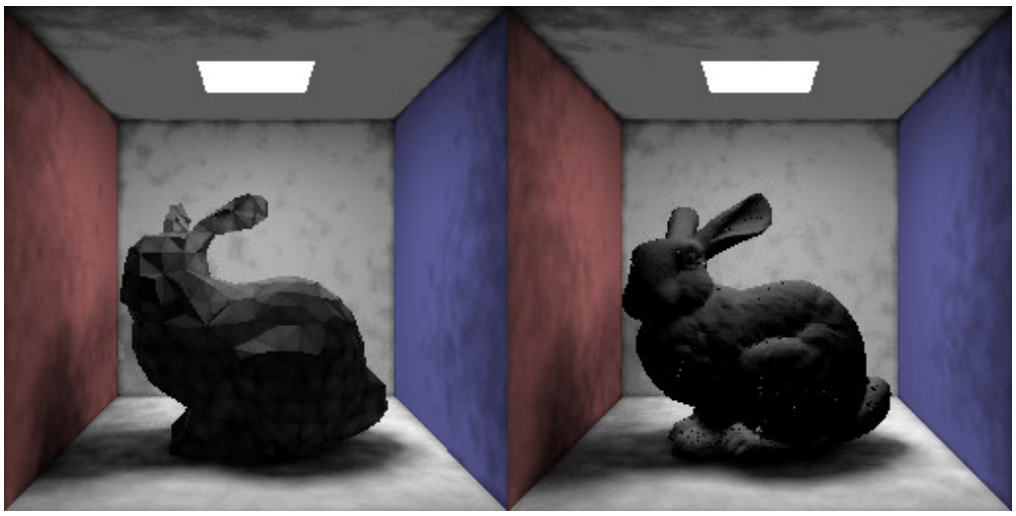

Figure 27: Two scenes with bunny models of different polygon counts. Images rendered by ENCORE.

Examples of the remaining scenes are shown as Figure 27 and Figure 28. The Stanford Bunny was used in all of these scenes as it comes in four levels of detail. The change in detail is useful for targeting rendering performance as a result of increased polygon counts. The low-detail bunny has fewer than 1,000 polygons. The high-detail bunny has over 60,000 polygons. Figure 27 compares two scenes where the polygon counts differ. Notice the lower detail in the bunny in the left scene. Figure 28 shows the three small, medium and large bunny scene

variations. These scenes were used to examine the impact different spatial distribution of polygons has on rendering performance.

Figure 28: Three scenes where the same bunny model at different scales. Images rendered by ENCORE.

5.3 Hardware

All tests were run on a Hewlett-Packard nc6230 consumer-grade laptop. Although more sophisticated laptops are available to consumers, this laptop was adequate for running the tests. The battery was listed on Hewlett-Packard's website as a 6-cell lithium ion battery with a capacity of 48 watt-hours [36]. The processor is an Intel Pentium M with a clock speed of 2.0 gigahertz with 1 gigabyte of system memory available. The operating system was Microsoft Windows XP with Service Pack 2 installed. No software was installed prior to or during the testing phase to ensure that the renderer was the primary application running on the laptop. During each test the wireless internet card was deactivated, the speakers were muted, and the display's brightness was reduced. No additional efforts were made to reduce energy consumption.

Standard system processes, such as Windows' services.exe and anti-virus software, were not disabled during the tests. These processes will consume energy but disabling them is not likely to save any noticeable amount. They largely run in the background with the idle process consuming over 99% of the CPU's time.

6 Results

The testing phase of the thesis was segmented into five batches of tests. Each batch contains a set of tests designed to examine the effects varying a rendering setting has on time, energy, and image difference. Both elapsed time and energy consumed were measured during each test as described earlier in section 4. The results presented in this section, however, are in terms of energy alone. This simplification is based on a correlation between rendering time and energy. Figure 29 demonstrates this correlation. It is a plot comparing time and energy for many of the tests presented in this section. The settings that were varied included the global photon count, the gather distance, the gather count, the irradiance cache tolerance, and the sample ray count for final gathering. A linear trend line was plotted to better identify the distribution of points. The fact that each data point is on or touching the trend line suggests a strong correlation between energy and time. Presenting both time and energy results would be redundant. Furthermore, this thesis presents solutions for reducing energy consumption on mobile devices. Thus, energy is the most appropriate metric.

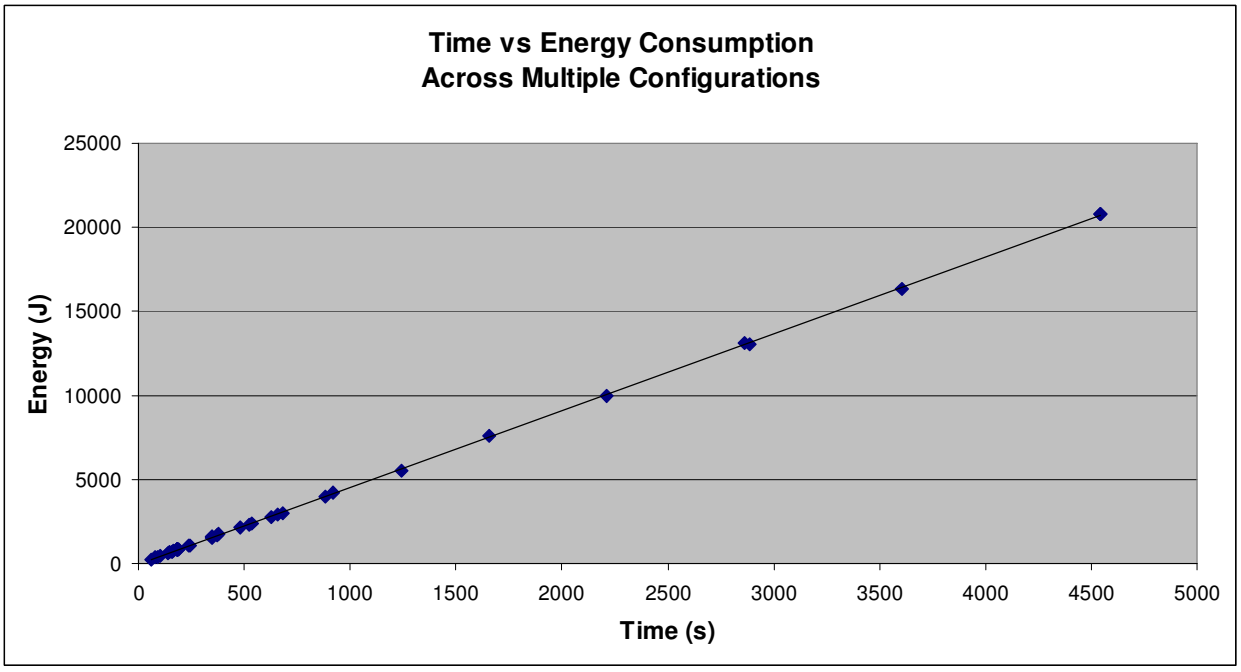

Figure 29: A scatter plot of time versus energy for many varied rendering configurations.

Two of the five batches examined indirect lighting calculations using direct visualization and final gathering. The majority of tests conducted were in these two batches. The

considerable focus on indirect lighting is due to the large percentage of the overall time and energy used by indirect lighting relative to other rendering calculations. Figure 30 shows the energy results of five different configurations. The results are separated by how much direct lighting, indirect lighting (using direct visualization), and photon map creation consume energy. Each configuration changes only by the number of photons stored in the global photon map. Notice how quickly the rate of energy consumption increases for indirect lighting as the number of global photons stored increases. Photon map creation and direct lighting consume far less energy.

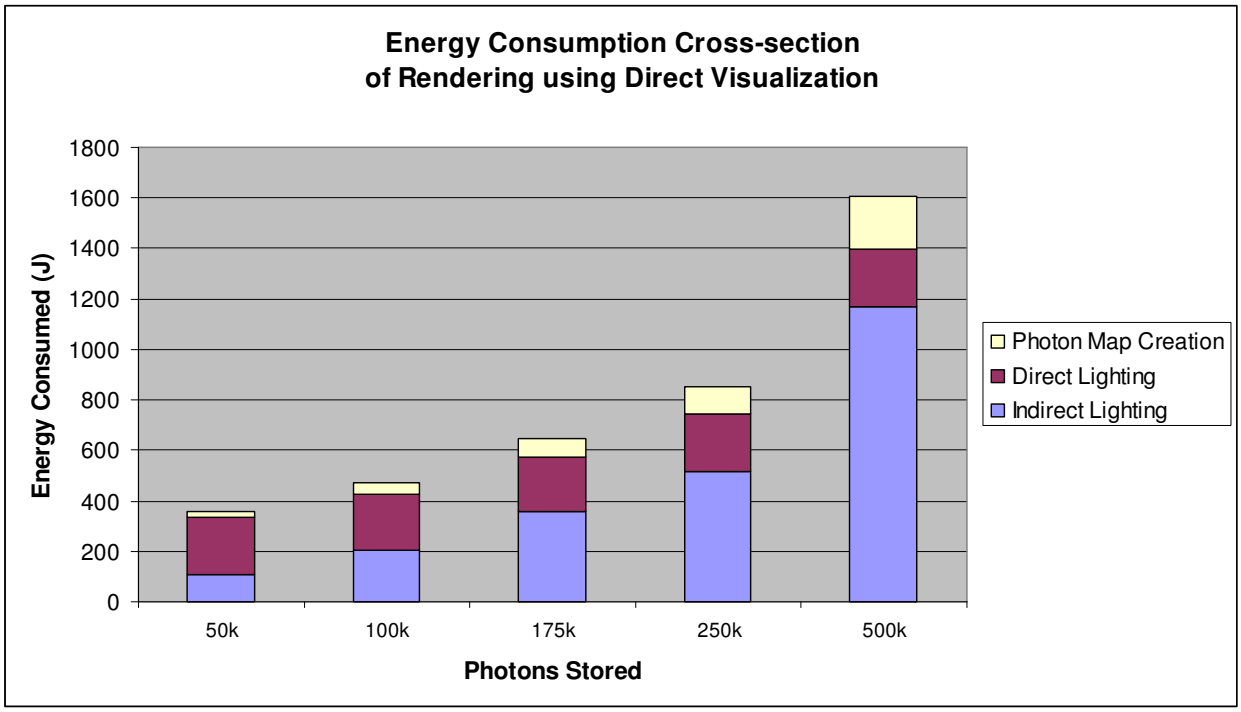

Figure 30: A comparison of energy use by direct lighting, indirect lighting using direct visualization, and photon map creation. Each test varies the number of stored global photons.

This trend is exaggerated when final gathering is used instead of direct visualization. Figure 31 shows data from four tests where indirect lighting was calculated using final gathering. Each test used 100,000 global photons and varied only in the number of sample rays used in each final gather calculation. It is hard to identify the amount of energy consumed by photon map creation in Figure 31 because in each case direct and indirect lighting combined to account for over 98% of the total energy used.

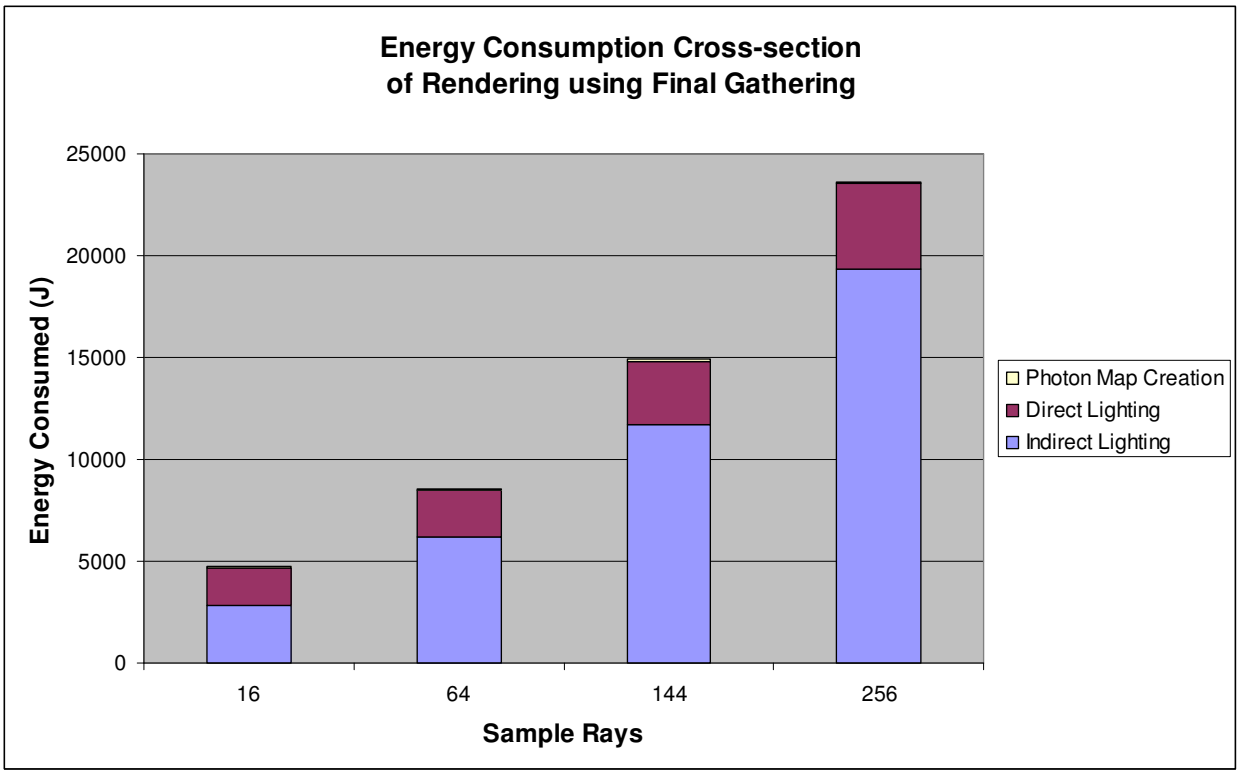

Figure 31: A comparison of energy used by direct lighting, indirect lighting using final gathering, and photon map creation. Each test varies the number of final gather sample rays.

The remaining test batches were small but no less thorough in evaluating other aspects of global illumination rendering. The third batch focused on caustics rendering. The change in energy consumption as the polygon count increases in a scene is evaluated in the fourth batch. The fourth batch also includes tests that considered the distribution of polygons within a scene. Rendering images of different resolution was investigated in the final batch. This is particularly relevant for rendering on mobile devices as screen resolution varies widely across different form-factors.

The results from evaluating indirect lighting and caustics (the first three batches) are discussed based on the stages of the photon mapping algorithm. This provides a more comprehensive view into the performance of the renderer than a disjointed discussion based on the data collection methods. Results from the high-polygon scene batch are presented where appropriate. The batch of tests investigating image resolution is discussed in section 6.4. All tests rendered an image with a resolution of 256x256 unless otherwise noted. This resolution

was chosen so tests finish in a reasonable time. It will be shown that the resolution will not affect the general trends.

6.1 Photon Map Creation

Photon map creation is a short stage that occurs before image rendering. Figure 30 and Figure 31 show that the energy consumed by this stage is small relative to direct and indirect lighting stages.

Figure 32 is a graph of the energy consumed during the photon map creation stage for two scenes across 5 configurations. Tests corresponding to the diffuse box scene are represented by blue data points. Tests corresponding to the specular sphere scene are shown in pink. Each of these scenes was previously described in section 4.2. The five configurations differ only in the number of global photons stored, ranging from 50,000 to 500,000. Each configuration used direct visualization for rendering indirect lighting. Note that the results would be the same had final gathering been selected for indirect lighting as photon map creation is an independent step.

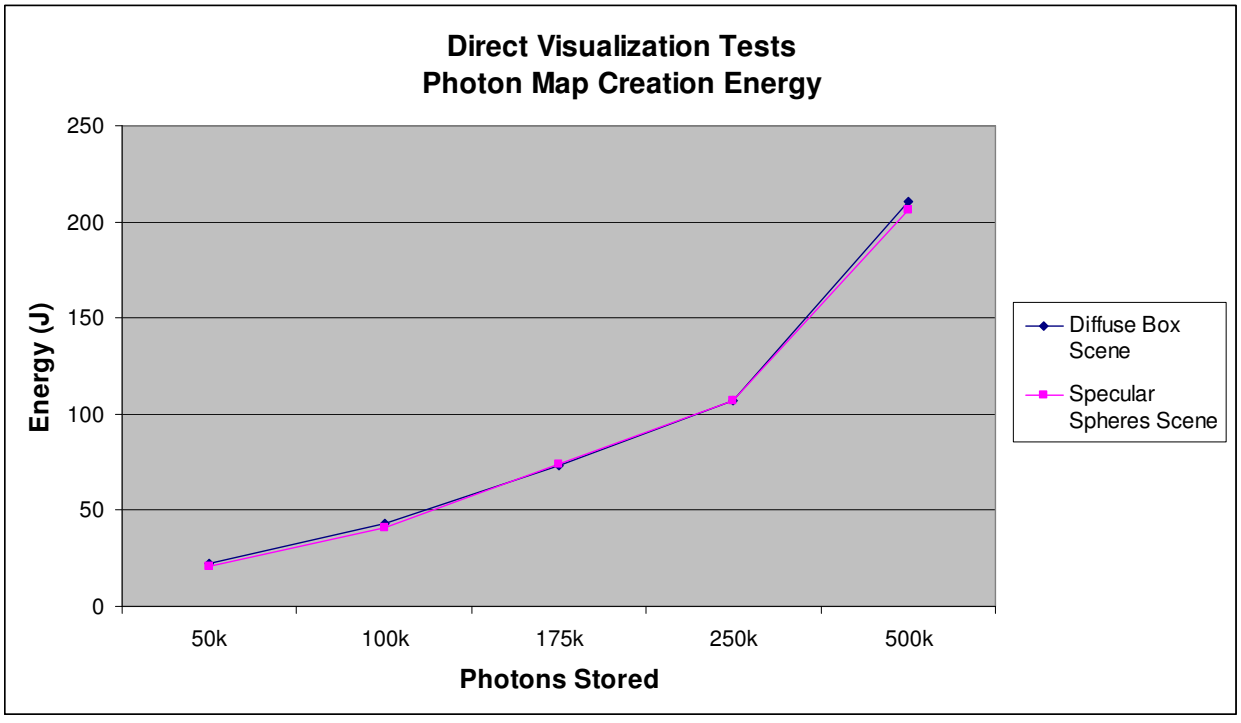

Figure 32: Energy consumed by five different rendering configurations.
Each configuration was used to render two different scenes.

Figure 32 demonstrates that as the photon count increases, so does the energy consumed. This is very intuitive as the total amount of processing and data required increases per added photon. Additionally, one can tell that both the diffuse box scene and the sphere scene have the exact same energy requirement given the same number of photons. This is not too surprising as global photons are only stored when they hit diffuse surfaces and the total area of diffuse surfaces in both scenes are relatively high.

A scene with a higher total area of pure specular surfaces than diffuse surfaces would likely have a higher energy consumption rate for a given photon count. This is evident if one considers that reducing the possible area of photon storage lowers the likelihood a photon will hit a diffuse surface after being reflected thus increasing the time before it is stored. On the other hand, this is not a particularly compelling test as the fact that it takes longer to scatter photons is just the nature of the scene being rendered.

While the energy characteristics of scattering global photons do not elicit any surprises, caustic photon scattering is a more interesting case. Figure 33 shows the energy consumed during the photon map creation stage when both global and caustic photons are scattered. The dark blue, pink, and yellow data points correspond to configurations where 30,000, 20,000, and 10,000 caustic photons were stored, respectively. The teal data points represent tests that no caustic photons were emitted or stored. The number of photons stored in the global photon map is identified on the x-axis.

Note that as the number of caustic photons stored increases, so does the energy consumed. This is as obvious as it was in the case of global photons discussed earlier. Looking deeper, Figure 33 demonstrates that creating a caustics photon map requires significantly more energy than creating a global photon map. A test using 50k global photons and 0 caustic photons (the left-most blue data point) used approximately 25 Joules. Another test that used the same configuration except with 10k caustic photons (the left-most yellow data point) used more than double the amount of energy. The high cost is related to the probability a caustic photon is created and stored.

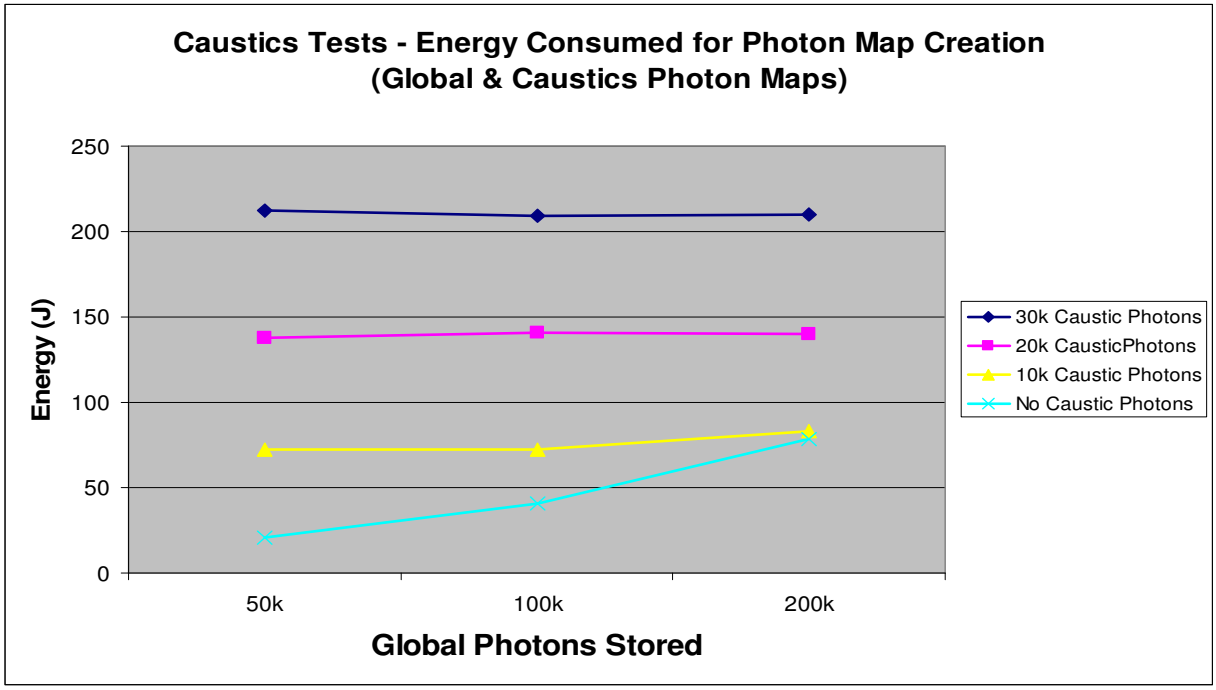

Figure 33: The total energy used by the renderer to store a different number of global and caustic photons. The specular sphere scene was used in all of the tests.

A caustic photon is stored when a photon is reflected by a specular surface onto a diffuse surface. In a scene with few specular surfaces, this specular-to-diffuse event is not as likely to happen as a diffuse-to-diffuse event. This fact could be attributed to the nature of the scene, similar to the nature of scenes with many specular surfaces and how they affect global photon scattering. However, since most scenes where caustics are present normally only have a few objects that generate caustics, the sparseness of specular-to-diffuse events is typically low. Jensen suggests that the scene could contain information for the renderer that indicates the location of caustic-generating objects [17].

The most interesting information gleaned from Figure 33 is that the energy cost of emitting 10,000 caustic photons can be absorbed by emitting an increased number of global photons. The test configured to emit 200,000 global photons only took relatively the same amount of energy as the test that emitted 200,000 global photons in addition to 10,000 caustic photons. This is because in the course of emitting the 200,000 photons, the proper number of specular-to-diffuse photon interactions happened so that 10,000 caustic photons were stored. This makes a strong case that if one wishes to store more caustic photons one might as well increase the global photon count for more efficient use of time and energy.

Tests were run to evaluate the impact polygon count and polygon distribution has on energy use of the photon-mapper. Figure 34 shows the data from those tests corresponding to photon map creation. Polygon count is mapped to the x-axis and ranges from 1,000 polygons to 70,000 polygons. The distribution of the polygons is determined by the size of the bunny. The small bunny scene isolates polygons in the center of the scene. The large bunny has a much broader distribution although there is an empty volume of space within the bunny not occupied by polygons. Section 5.2 has examples of these scenes.

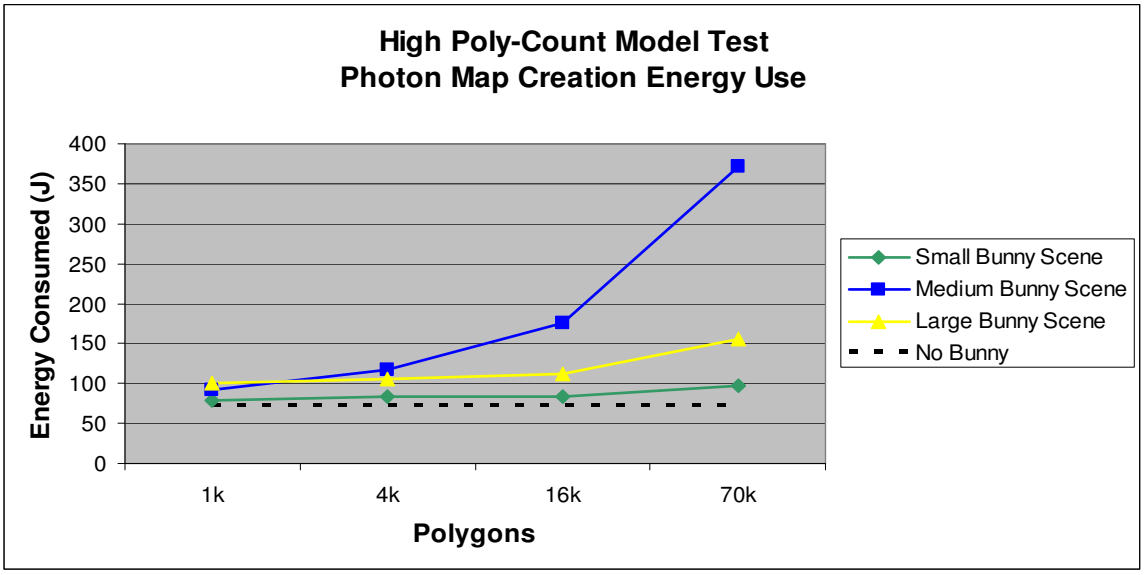

Figure 34: A graph showing the results of tests where the polygon count and distribution were varied.

An intuitive result easily seen in Figure 34 is that energy consumption increases as the polygon count increase. A more interesting feature of the data is that the medium bunny scene shows a significant increase in energy consumption compared to the small and large bunny scenes. The primary calculation during photon map creation is the intersection test between a photon's path and scene geometry. The medium bunny apparently distributes polygons in such a way that more intersection tests are performed than with the large and small bunnies.

6.2 Direct Lighting

Calculating direct lighting is the first step of three in the rendering phase. Specific tests were not conducted to investigate different configurations of direct lighting. In fact, settings that change the quality of direct lighting were kept constant in all tests. Those settings and their values are shown in Table 2.

Direct Lighting Settings	Value
Direct Photons Stored	10,000
Shadow Rays	144

Table 2: Direct lighting settings

Even though tests did not specifically investigate direct lighting, it serves to discuss this step briefly to understand how it impacts overall energy consumption of the renderer. Figure 35 shows the energy consumption reported in ten different tests (5 different configurations used to render two different scenes). The data used for Figure 35 comes from the same tests as those used for Figure 32. The most important thing to note is that the energy used for the diffuse box scene is constant, whereas the energy used to render the specular sphere scene is increasing as the global photon count increases. This can be attributed to the transparent sphere and the reflective sphere in the specular sphere scene.

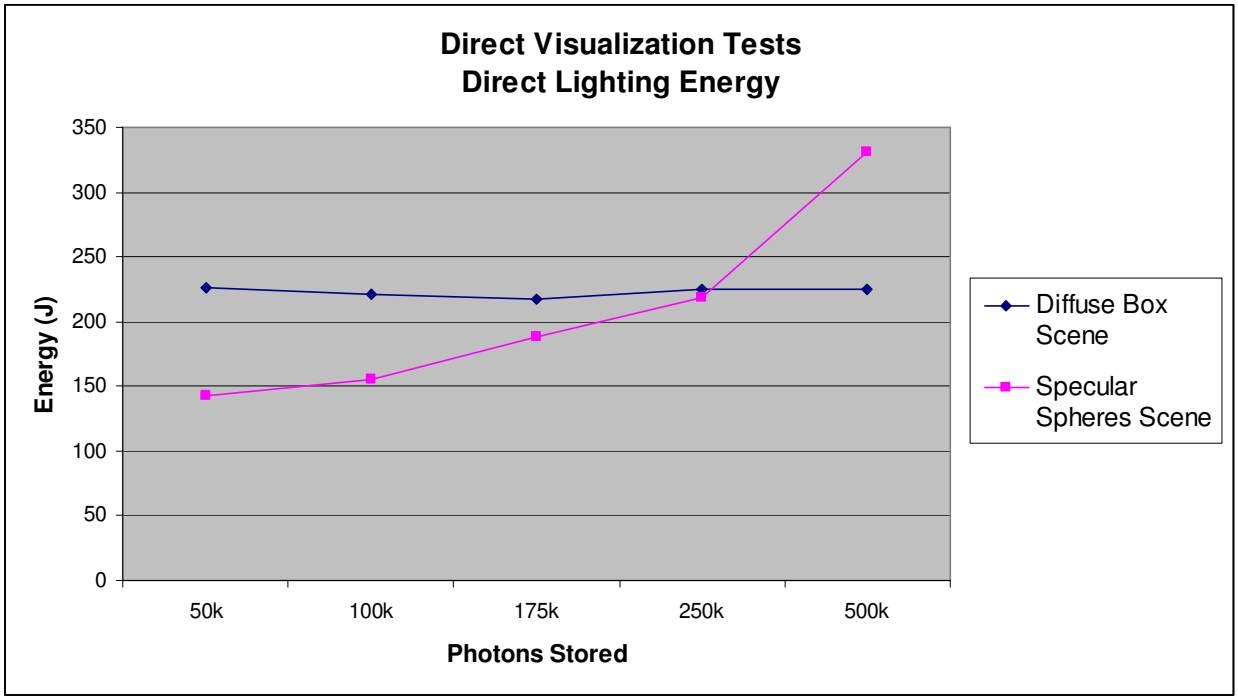

Figure 35: The energy consumption of multiple configurations where the number of global photons varies.

Recall that when a primary ray hits a reflective surface, reflection rays are generated. Likewise, when a primary ray hits a transparent surface, rays may be transmitted through the surface. These secondary rays are used to calculate the total light at the first surface they intersect. The total light calculation includes direct lighting, indirect lighting, and caustics (if

they are being rendered). The increase in energy use indicated by the pink line in Figure 35 is due to the increased cost of indirect illumination calculations because more global photons are stored. It makes sense to consider the cost of calculating the indirect illumination as a part of the cost of calculating the direct lighting because the results directly contribute to the color of the direct lighting.

Note that total energy used for direct lighting calculations for the specular sphere scene is lower than the diffuse box scene when 50,000 photons are stored in the global map. Excluding all indirect lighting calculations, it would be more expensive to calculate direct lighting for the diffuse box scene than for the specular sphere scene. This is because the diffuse scene has more regions of shadow in the scene and thus has more regions where shadow rays are used. Recall that this renderer implements soft shadows, which use shadow rays around the edges of the shadow regions to create a smooth transition from lit areas to dark areas.

The same tests described in reference to Figure 34 are used here for Figure 36. In this case, the data corresponds to the energy use of direct lighting. The results in Figure 36 are very similar in shape to those in Figure 34.

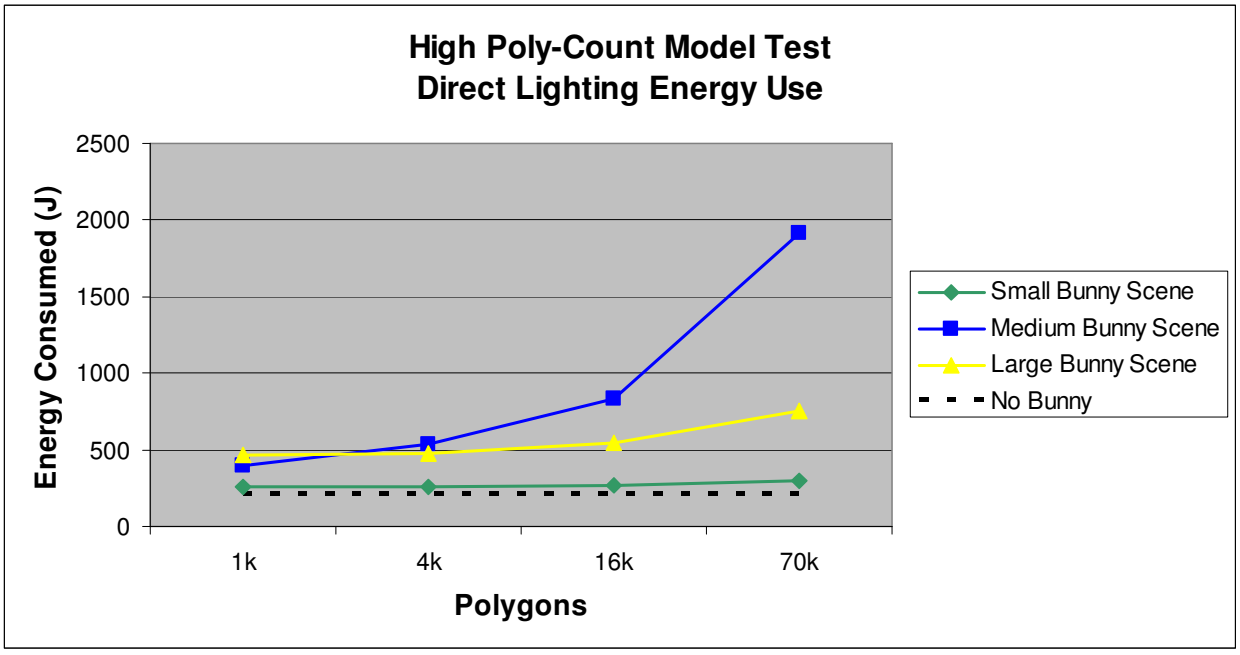

Figure 36: A graph showing the results of tests where the polygon count and polygon distribution was changed between scenes.

As one would expect the energy consumed increases as the polygon count increases. Also similar to Figure 34 is that the scene with the medium-sized bunny exhibited higher energy

consumption than the scene with the large bunny. This may be attributed to an increased number of triangle-ray intersection tests with the medium bunny scene. The main difference between Figure 34 and Figure 36 is the y-axis scale. Since both photon map creation and direct lighting performance is linked with the number of triangle-ray intersections, the difference in scales is due to the larger total calculation time for direct lighting.

Finally, unlike indirect lighting, rendering caustics have no significant impact on the energy used by direct lighting. The energy expended for direct lighting from tests in the caustics batch is graphed in Figure 37. The specular sphere scene was rendered for all of the tests. Note the tight grouping of the lines linking the data points. This suggests that while the total number of caustics photons used changes, the trend in energy use does not vary. This result is understandable because caustics tend to be isolated into a particular region of the scene. Photon gathers using the caustics photon map are very fast, frequently returning zero photons, because photons are concentrated in space. The concentration of photons makes sense as the definition of a caustic is light that has been focused into a region as a result of a reflecting surface.

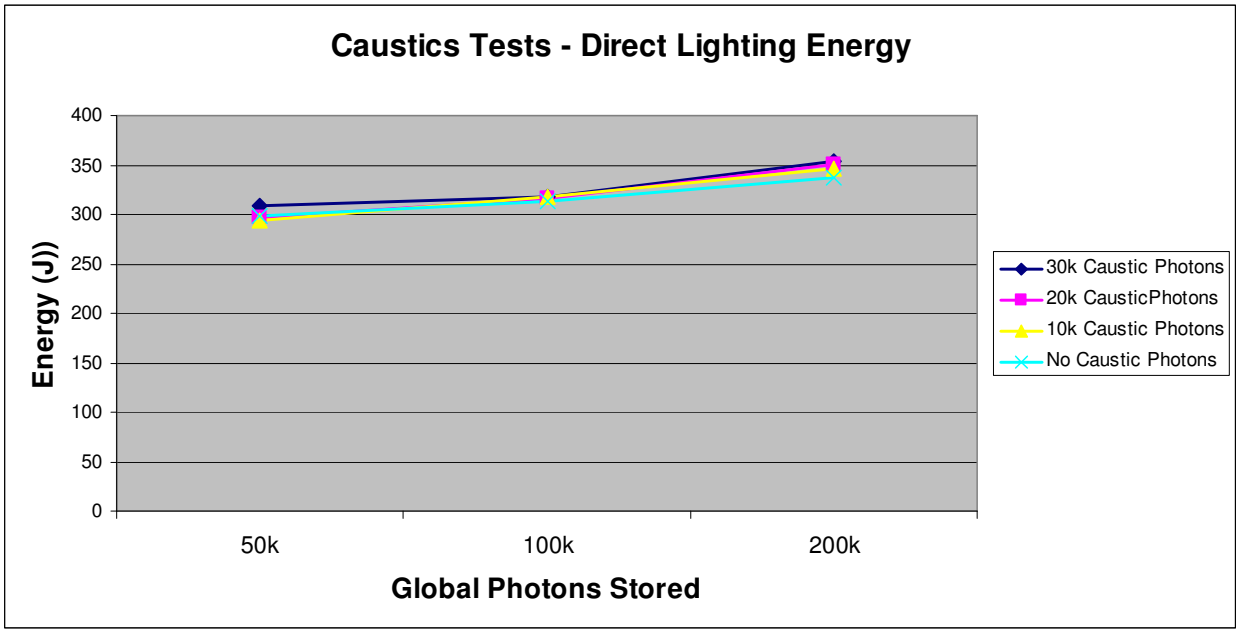

Figure 37: Energy consumption of direct lighting when caustics are rendered for the specular sphere scene.

6.3 Indirect Lighting

The main focus of this thesis has been on indirect lighting since it is the most costly calculation when rendering an image. Like the previous results sections, graphs showing the

change in energy consumption across different configurations are presented. Additionally, image comparisons using *ltdiff* will be used to relate how these different configurations affect the final rendered image.

6.3.1 Direct Visualization

The first batch of tests examined the sensitivity of the renderer to changing configurations when direct visualization is used to estimate indirect illumination. The settings changed in each configuration were the global photon count, gather distance, and the gather count. Figure 38 gives a high-level view of the effect varying these three settings has on energy consumption and image difference.

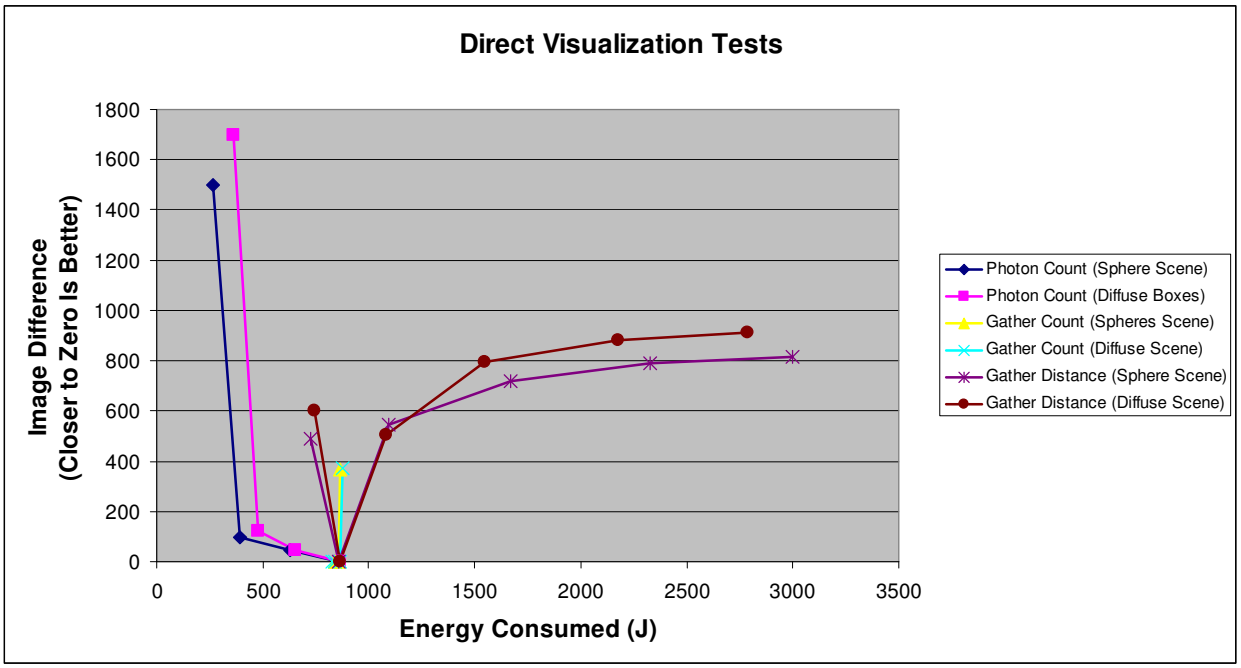

Figure 38: A comprehensive graph of all the tests run to evaluate direct visualization.

The two scenes used in these tests were the diffuse box scene and the specular sphere scene. Notice that dark blue and the pink lines correspond to the specular sphere scene and the diffuse box scene respectively. They both represent results from different configurations where only the total photon count was varied. The similar trend among these two lines illustrates that changing the photon count has the same impact on image quality and energy consumption in different scenes. Changing the gather count and gather distance have a different impact than the photon count. However, changes in each setting have the same effect across the two scenes.

The most striking feature of the data in Figure 38 is the image difference for most of the tests. The configuration for the reference image (the data point in Figure 38 that has an image difference of 0) and a handful of global photon count tests are the only tests that yielded differences of less than 200. Recall from 5.1.3 that image difference values of 100 or lower are considered acceptable. Almost all data points plotted in Figure 38 differ from the reference image by 300 or more units. This suggests that almost all changes to a configuration when using direct visualization will likely result in a strong deviation from the reference image. Put another way, it is not likely that a useful tradeoff exists between image difference and energy consumption when using direct visualization. At best, one can tweak the renderer's configuration to approach the best image possible.

The remaining discussion on direct visualization is concerned with the way settings should be adjusted and the effect those adjustments have on energy consumption and image difference. All of the results presented during this discussion will use results that were shown in Figure 38.

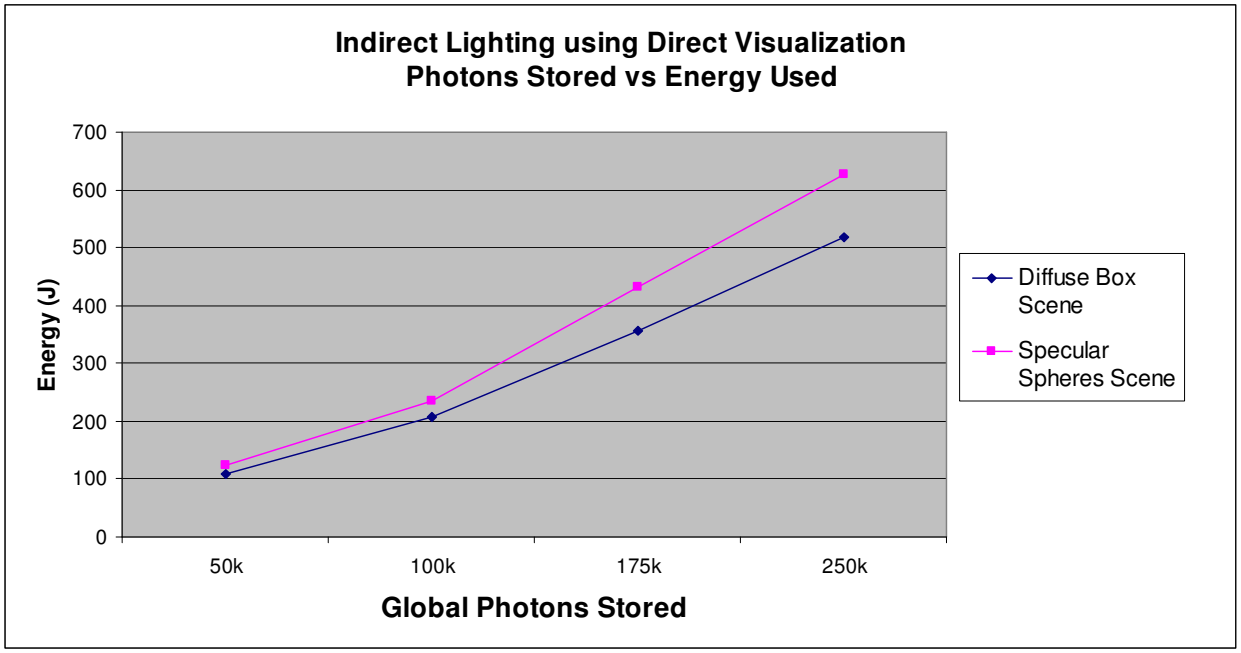

Figure 39: Change in energy use as global photons stored increases.

Figure 39 shows the increase in energy consumption of direct visualization as the number of global photons stored in the photon map increases. Figure 40 shows the image difference as reported by *ltdiff* of these same tests. The image references used for each scene were configured

to use 250,000 global photons. On the graph they are the rightmost data points. Figure 39 shows a smooth increase in energy consumption from 50,000 photons to 250,000 photons. Reading Figure 40 from the rightmost data point to the leftmost, the change in image difference is non-linear. There are over 1000 units in difference between the 50,000 photon image and the 100,000 photon image. Prior to this, the change in image difference is modest (between 0 and 100). This suggests that there is a minimum number of photons that can be emitted into the scene before one can approach a satisfactory estimate of indirect lighting.

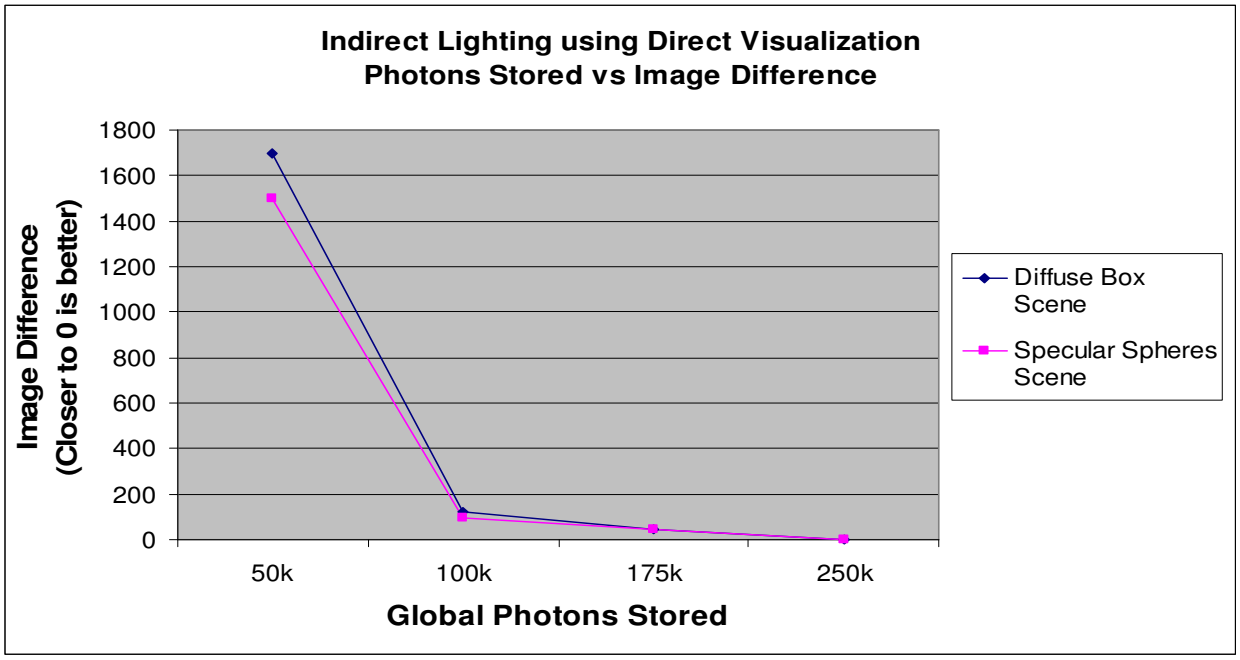

Figure 40: Change in image difference as global photons stored increases.

The modest change in image quality coupled with the noticeable decrease in energy consumption suggests that reducing the photon count from 250,000 to 175,000 would yield an acceptable image. This would save roughly 200 Joules during indirect lighting. If one were generating many images, one might accumulate a tidy sum of energy savings. On the other hand, since the savings are small one might choose to simply generate the best possible image.

The second variable tested for its impact on direct visualization was gather distance. Gather distance is the maximum distance photons can be stored from a point where the indirect lighting calculation is being performed. Any photon further from the gather distance is not considered a candidate for the lighting estimate. All of the tests here used photon maps with 250,000 stored photons and a gather count of 100 photons.

Figure 41 shows how increasing the gather distance increases the amount of energy consumed. This is intuitive as one is increasing the pool of photons considered for the lighting estimate. Figure 42 shows the change in image difference as gather distance increases. The reference image used a gather distance of 2 units, which is easily identified since it has a difference of 0. The image difference spikes both as one increases or decreases the gather distance. Notice in Figure 42 that as the gather distance increases, the image difference beings to plateau. This is because the maximum number of photons that can be collected (as determined by the gather count) is always reached. Continuing to increase the gather distance would only increase the energy costs without increasing image quality.

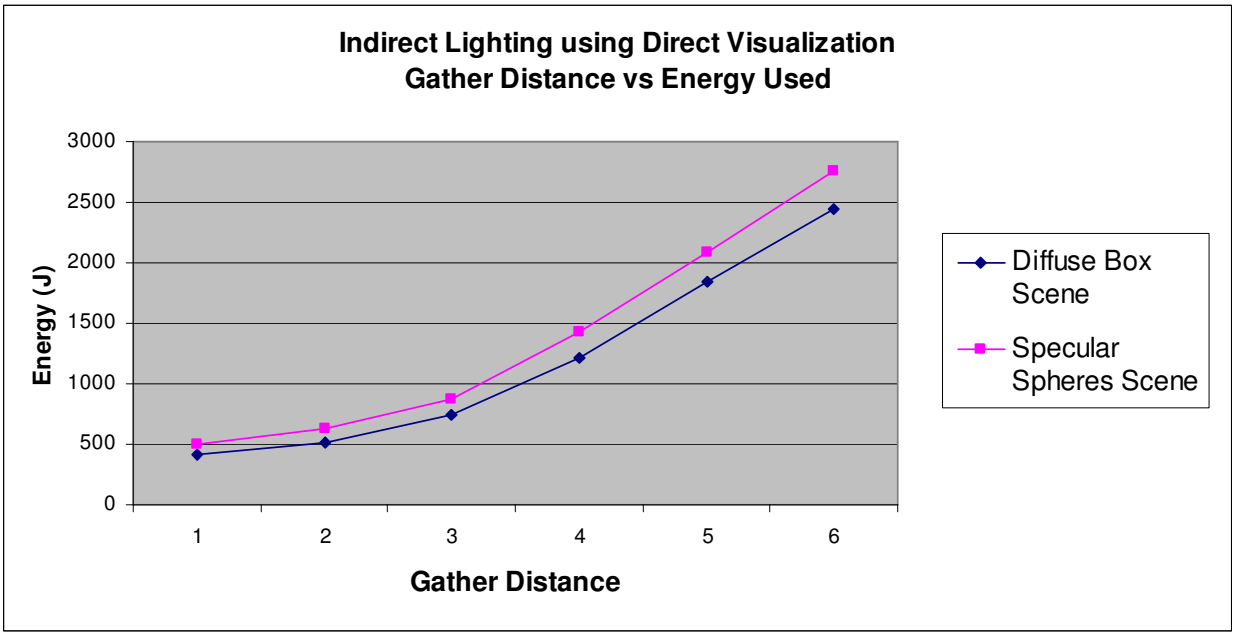

Figure 41: Change in energy use as gather distance increases.

Practical experience has shown that changing the gather distance does not yield better images as much as increasing the photon count or adjusting how lighting is estimated from the gathered photons. In other words, once a gather distance is chosen it should be kept constant. However, the lower the gather distance relative to the size of the world, the more energy one will save since it will eliminate photons from consideration much faster.

The final variable tested for direct visualization was the gather count. The gather count determines how many photons are collected from the photon map for use in an indirect lighting estimate.

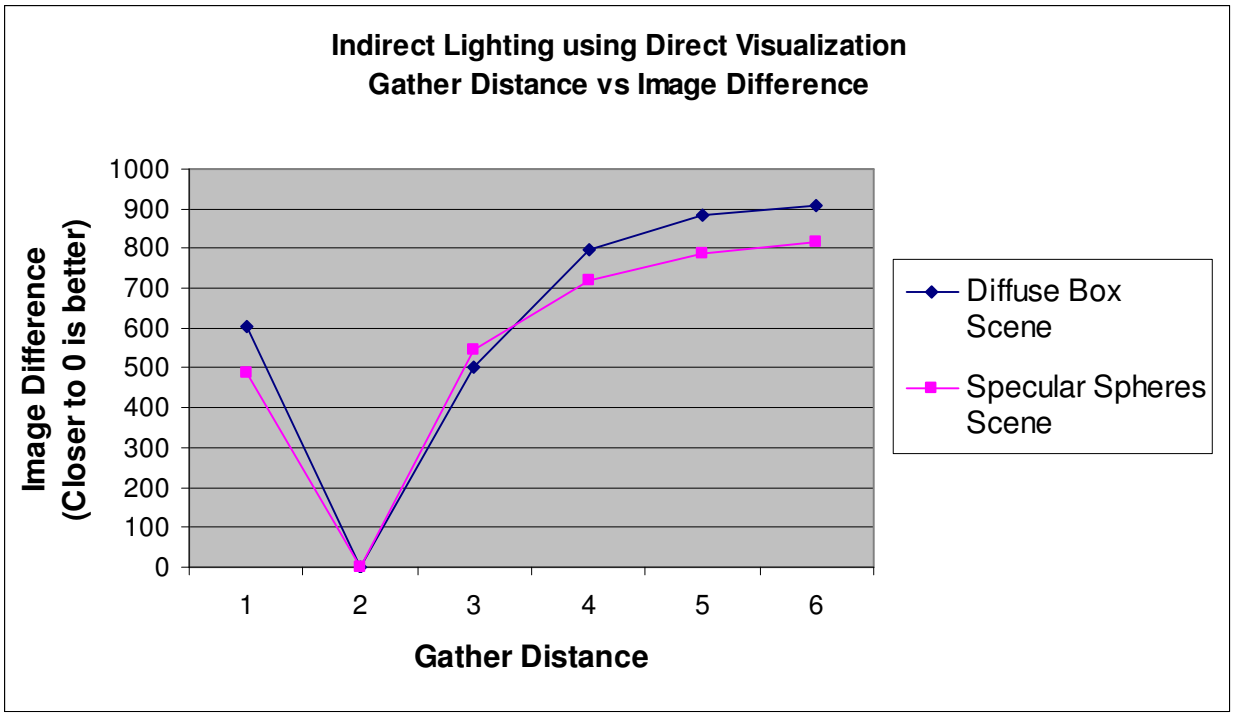

Figure 42: Change in image difference as gather distance increases.

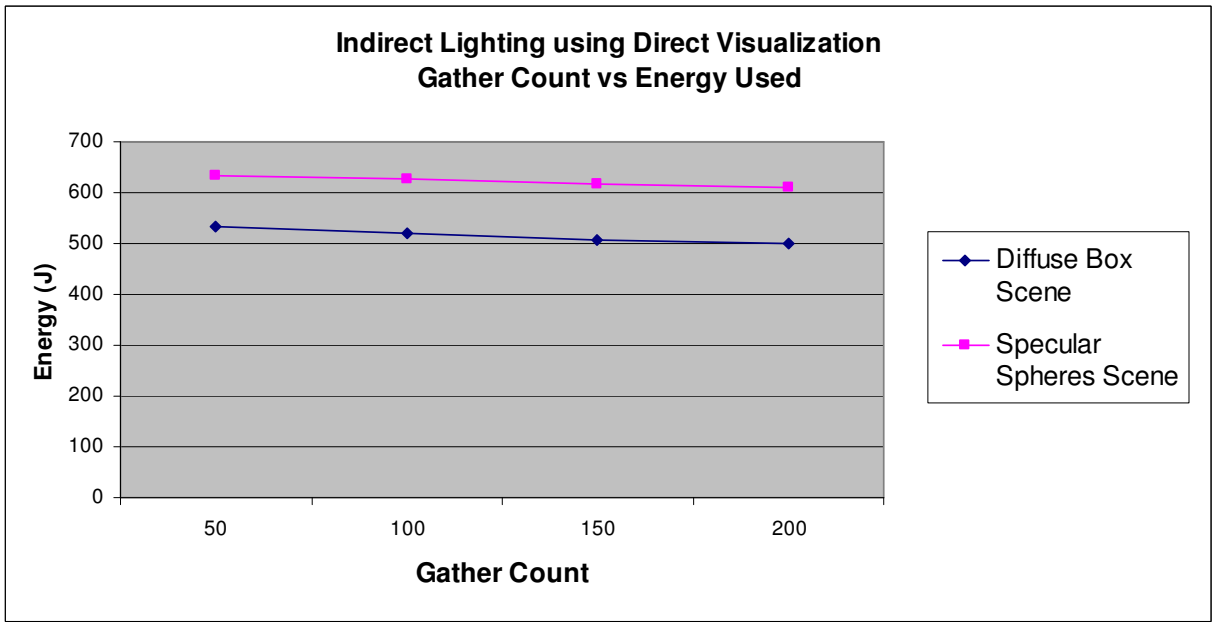

Figure 43: Change in energy use as gather count increases.

The gather count was varied from 50 to 200 by intervals of 50 photons. Figure 43 shows the results of four configurations across two different scenes. Unlike photon count and gather

distance, the energy savings by increasing the gather count is extremely small. The difference between 50 photons and 200 photons is less than 70 Joules.

Figure 44 shows how the images differ across the different configurations. The reference images used a gather count of 100. Much like how there is a minimum number of photons that should be stored, Figure 44 demonstrates that there is also a minimum gather count. There is a sizable difference between a gather count of 50 and 100. However, increasing the gather count did not change the image difference by more than 5 units.

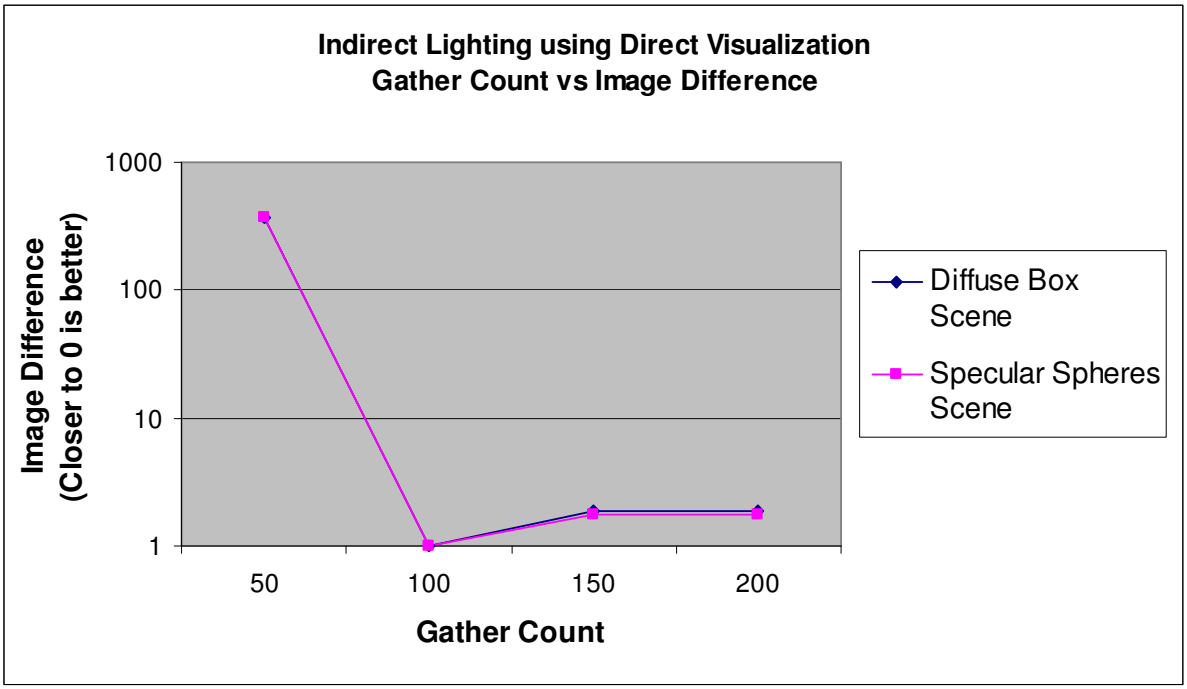

Figure 44: Change in image difference as gather count increases.

Since increasing the gather count reduced energy consumption with a low image difference, one might be lead to think increasing the gather count is a useful option. The subtle truth is that there is typically an ideal gather count that should be kept consistent. Notice that the difference between a gather count of 150 and 250 photons in Figure 44 are equal. The reason the difference is constant above 150 photons is that all of the photons within the spherical volume of space defined by the point of the indirect lighting estimate and the gather distance have been collected. If the gather count was unbounded, in every estimate 150 photons or fewer will be gathered. In simple terms, given a value for both the total number of photons and a gather distance, there is some maximum number of photons that can be collected and it makes no sense

to set a gather count higher than this maximum. One should choose a gather count not on how much energy one wishes to save, but based on what the gather distance and number of photons stored.

Finally, it is worth noting that increasing the polygon count of models in a scene has little impact on the energy used for indirect lighting. Notice in Figure 45 that for all sizes of the bunny model the energy consumed remains constant. This is because the photon map is a data structure decoupled from scene geometry.

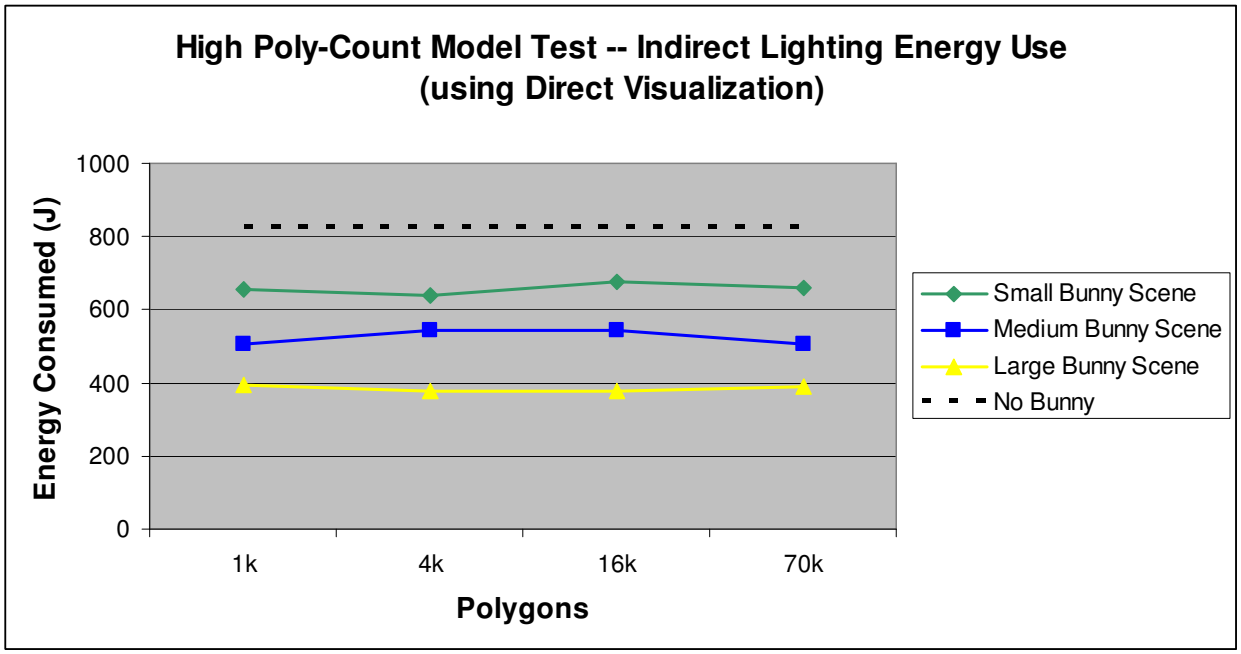

Figure 45: Impact on high polygon scenes have on indirect lighting with direct visualization.

One might find it surprising that increasing indirect lighting is cheaper in terms of energy when a high-polygon model is present than when it is not. Energy is also reduced for larger forms of the bunny. This is because when a final gather is calculated, photons are not only rejected by their distance from the estimate point, but by the surface normal of the absorbing surface. When a surface normal deviates 30 degrees or more from the photon's normal it is culled just like when it is farther than the gather distance. Since the high-polygon bunny model has many surfaces with many different normals, many photons are eliminated early. This strict culling will decrease the contribution of indirect lighting on high polygon models. A renderer can be configured so that it is less strict thereby increasing the amount of indirect light a model

receives. The resulting energy consumption is likely to be comparable to the energy characteristics of the scene without the bunny.

6.3.2 Final Gathering

Final gathering is the most computationally expensive calculation that is implemented in the renderer. The settings changed across the final gathering tests were the number of final gather rays, the irradiance cache tolerance, the number of photons used, and the photon spacing for precomputed irradiance.

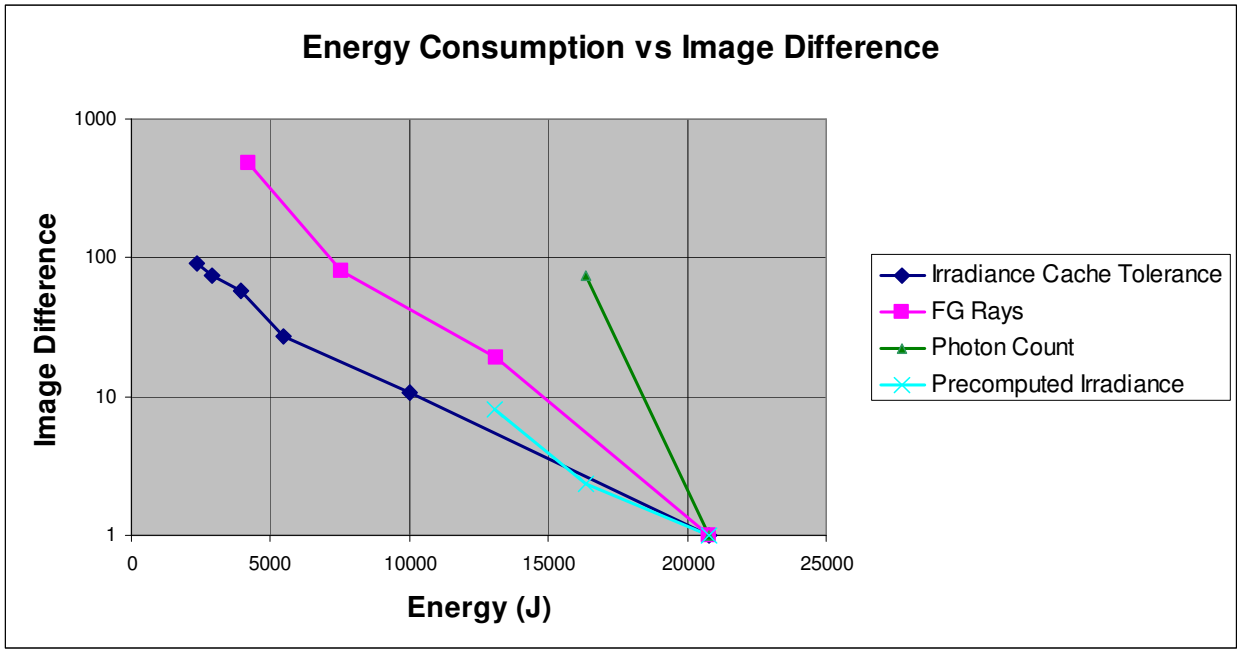

Figure 46: A comprehensive graph of tests run using final gathering for indirect lighting.

Figure 46 is a comprehensive graph of how photon mapping settings impact energy consumption versus image difference. It uses a logarithmic scale on the y-axis so that the single outlier (the left-most final gather ray test) does not obscure the overall picture.

Comparing this plot to Figure 38, its direct visualization counterpart, one will notice how much less the image difference varies as photon mapping settings are adjusted. Furthermore, many of the tests generated images with differences below 100 suggesting that several tradeoffs can be made.

Using the vertical gridline where energy equals 15,000 J as an intercept, each setting can be ranked in terms of what provides the best tradeoff between image quality and energy. The

best tradeoff can be made by adjusting the value of the irradiance cache tolerance as its slope is the closest to zero. This provides significant energy savings per unit image difference. The next best based on Figure 46 is precomputed irradiance. The third best tradeoff in a distant third can be achieved by modifying the number of final gather rays used. The worst performing potential tradeoff is the total global photon count.

The remaining discussion looks at how precomputed irradiance, image tolerance, and final gather rays can be configured to reduce energy use while maintaining a satisfactory image. Varying the number of photons stored in the global photon map is not considered here as it has the least favorable trend of energy savings per unit image difference. All the final gathering tests, unless otherwise noted, used the specular sphere scene.

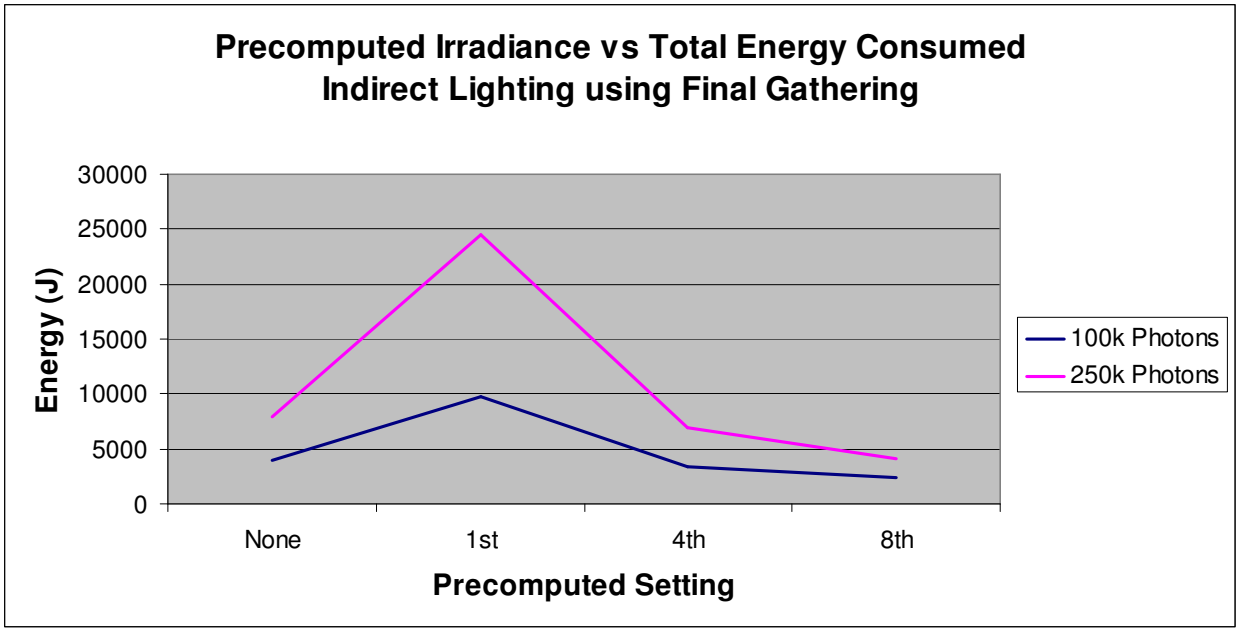

Figure 47: Energy use as the photon spacing increases for precomputed irradiance.

Figure 47 shows the energy trends of using precomputed irradiance with final gathering. Two sets of tests were run that changed the number of photons stored in the global photon map. The change in photon spacing is mapped onto the x-axis. The left-most data points ("None") do not use precomputed irradiance. The right-most data points precomputed irradiance at every 8th photon. There is approximately a 3,000 Joule decrease in energy use from no precomputed irradiance to precomputed irradiance at every 8th photon when 250,000 photons are stored.

Similarly, there is a 1,500 Joule decrease from no precomputed irradiance to every 8th when 100,000 photons are stored.

Energy use is considerably higher when irradiance is precomputed at every photon. Only the single closest photon needs to be found when a photon gather operation is performed when irradiance has been precomputed. Unlike the tests where it's computed every 4th and every 8th photon, precomputing at every single photon does not reduce the size of the photon map. Figure 47 shows that searching for a single photon out of 100,000 (or 250,000 as is the case in one of the test sets) is more expensive than collecting 100 photons as is done during a normal photon gather.

The energy results for these precomputed irradiance tests may be scene dependent. Since the volume of space within the specular sphere scene is a mostly empty, the distribution of photons is uniform. This means that the density of photons in an area of the scene is consistent with any other area in the scene of the same size. Scenes with more complex geometry or a light source obscured by a surface will distribute photons in a non-uniform manner. In other words, the density of photons across a surface will change significantly. Calculations in regions of higher density will consume more energy than those with a lower density. The extent varied densities has on the energy saved using precomputed irradiance was not evaluated.

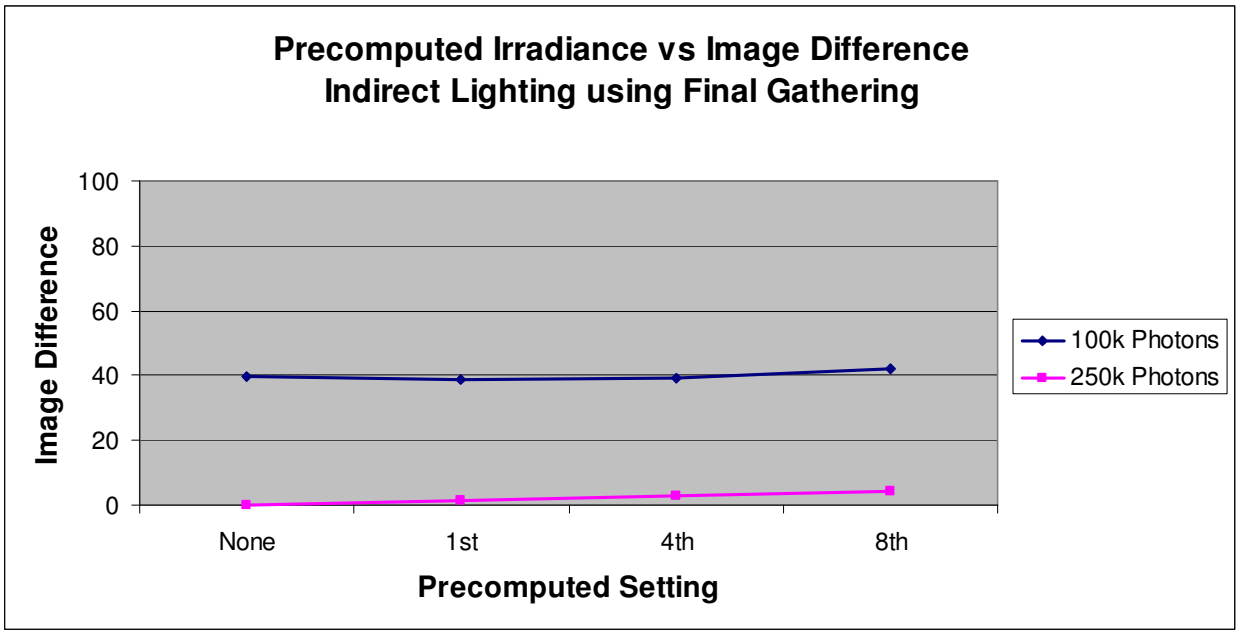

Figure 48: Image difference as the photon spacing increases for precomputed irradiance.

The reference image used in these tests was configured to have 250,000 photons in the global photon map and did not use precomputed irradiance. The results of the image comparisons are shown in Figure 48. All images generated where 250,000 photons were stored were very close to the reference image, differing by less than 5 units. The images that stored 100,000 photons in the global photon map differed from the reference by a constant amount of 40 units. The images using a lower number of photons showed more noise across surfaces which accounts for the constant difference.

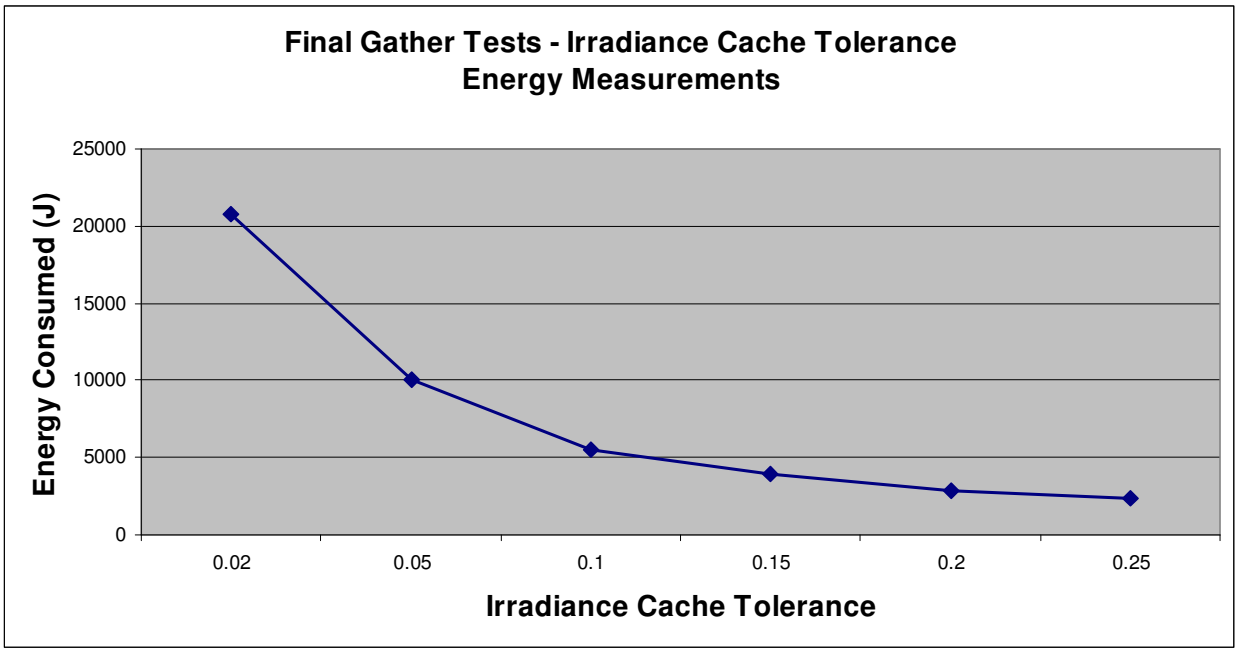

Figure 49: A plot of the change in energy consumption as the irradiance cache tolerance increases.

The irradiance cache tolerance is a parameter used to determine whether or not a previously computed lighting estimate stored in the cache can be used to interpolate a new lighting value. The higher the tolerance, the more the cached values can be used for interpolation reducing the number of expensive final gather calculations. Figure 49 shows how energy use decreases as the tolerance increases. There is a steep decline between 0.02 and 0.1. The trend approaches a horizontal asymptote after the tolerance reaches 0.15. Figure 50 shows the gradual increase in image difference as the cache tolerance increases. The reference image had a tolerance equal to 0.02. These results show that there is a highly favorable tradeoff

between image difference and energy savings. Changing the cache tolerance to 0.1 from 0.02 saves about 15,000 Joules and the image changes only by 30 units.

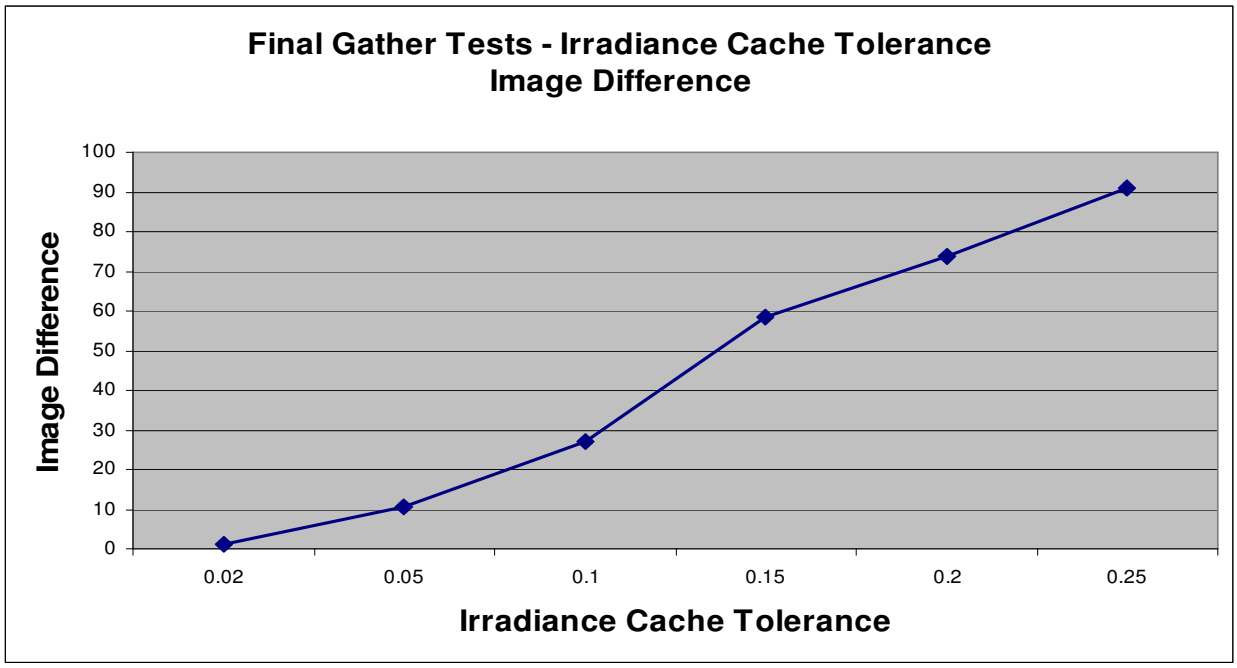

Figure 50: A plot of the change in image difference as the irradiance cache tolerance increases.

The last setting reviewed that affects final gathering is the number of final gather rays used to sample the scene. The ray count was varied from 16 rays to 256 rays. The reference image used 256 rays. Figure 51 shows that energy use increases smoothly as the number of sample rays are increased. The energy used by the configuration with 256 rays was four times as expensive as a similar configuration with 16 rays.

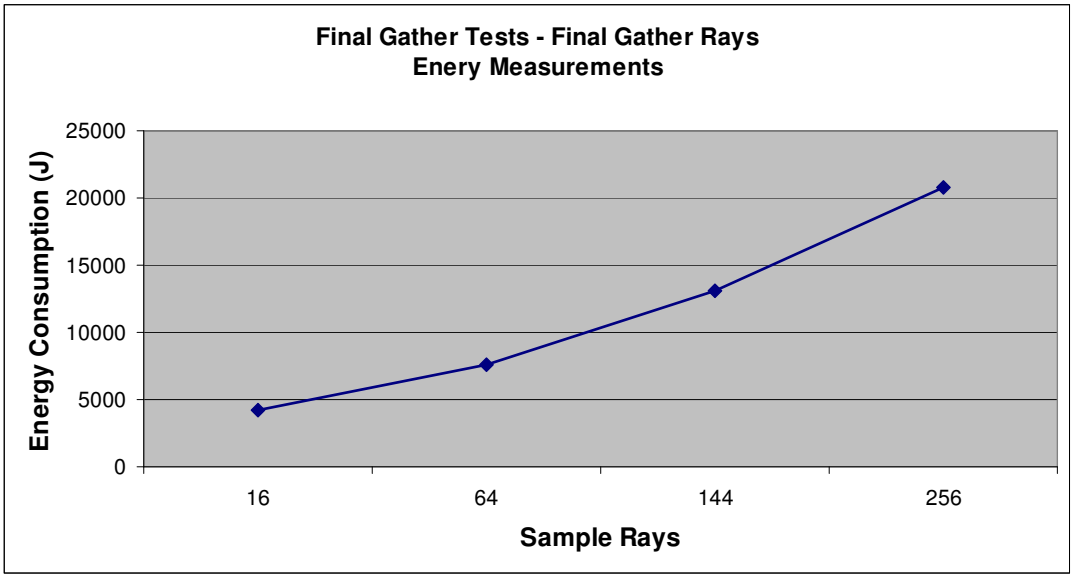

Figure 51: Energy use as the number of final gather sample rays increase.

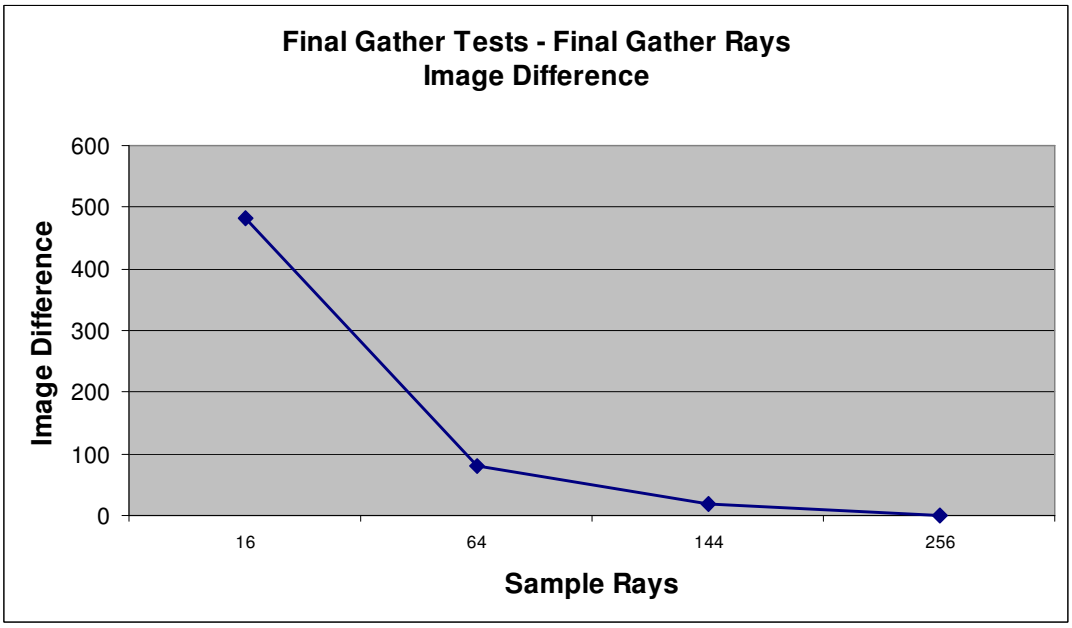

Figure 52: Image difference as the number of sample rays increase.

Image difference does not show as gradual a change as energy use. Figure 52 shows that when the number of sample rays drops below 64, the image difference spikes. There is a bare minimum of sample rays that should be used. The results in Figure 52 suggest that this minimum is between 16 and 64 final gather rays. The image difference for 64 rays was 80 units. If the maximum acceptable image difference is 100, then the number of rays used cannot be reduced too much beyond 64 rays.

Still, these results show that a promising tradeoff can be made. Over 6,000 J of energy is saved by reducing the number of final gather rays from 256 to 144. The resulting image is only 20 units different from the reference image.

The analysis of the impact high-polygon models have on energy consumption using direct visualization is the same for this final gathering method. Figure 53 demonstrates that a change in polygon count has no impact on energy consumption. The impact of polygon distribution on energy use is insignificant. However, it is hard to tell how insignificant as the total energy consumed by final gathering is so high.

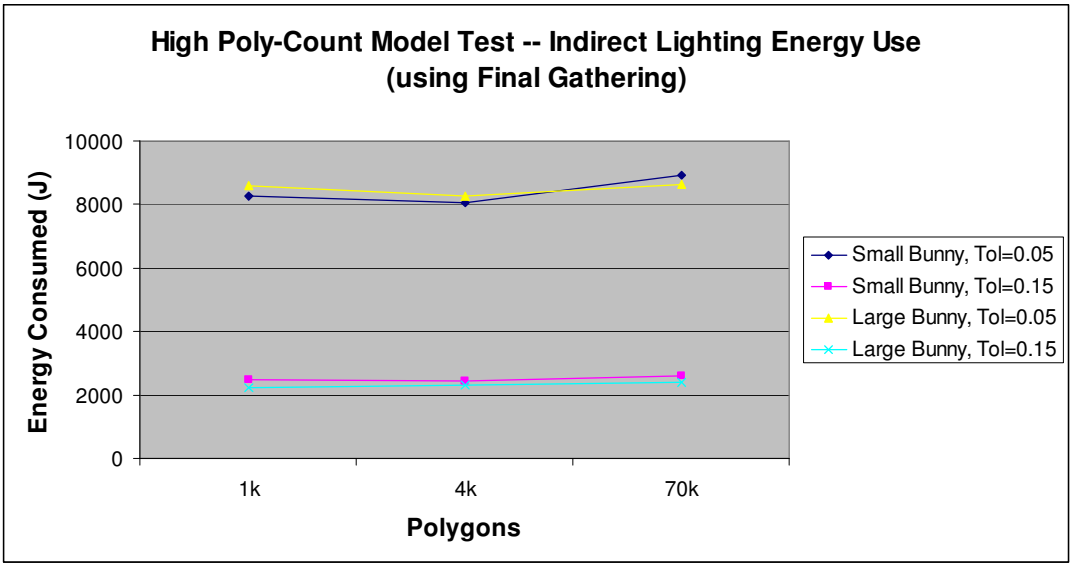

Figure 53: Impact high poly models have on indirect lighting using final gathering.

6.4 Image Resolution

Exploring how photon mapping settings change the energy characteristics seems logical. It is also useful to determine how well the tradeoffs discussed earlier will scale. Image resolution varies widely across the many mobile devices used by consumers. Some screens, like those for cell phones, are as small as 128x160 pixels. The PlayStation Portable game device has a larger resolution of 480x272 pixels. Laptop screens can have a resolution that rivals some desktop displays at 1440x900 pixels and larger. For these tests, three resolutions (128x128, 256x256, and 512x512) were chosen. Resolutions larger than 512x512 were not used as some configurations would likely run the battery down before completing the image.

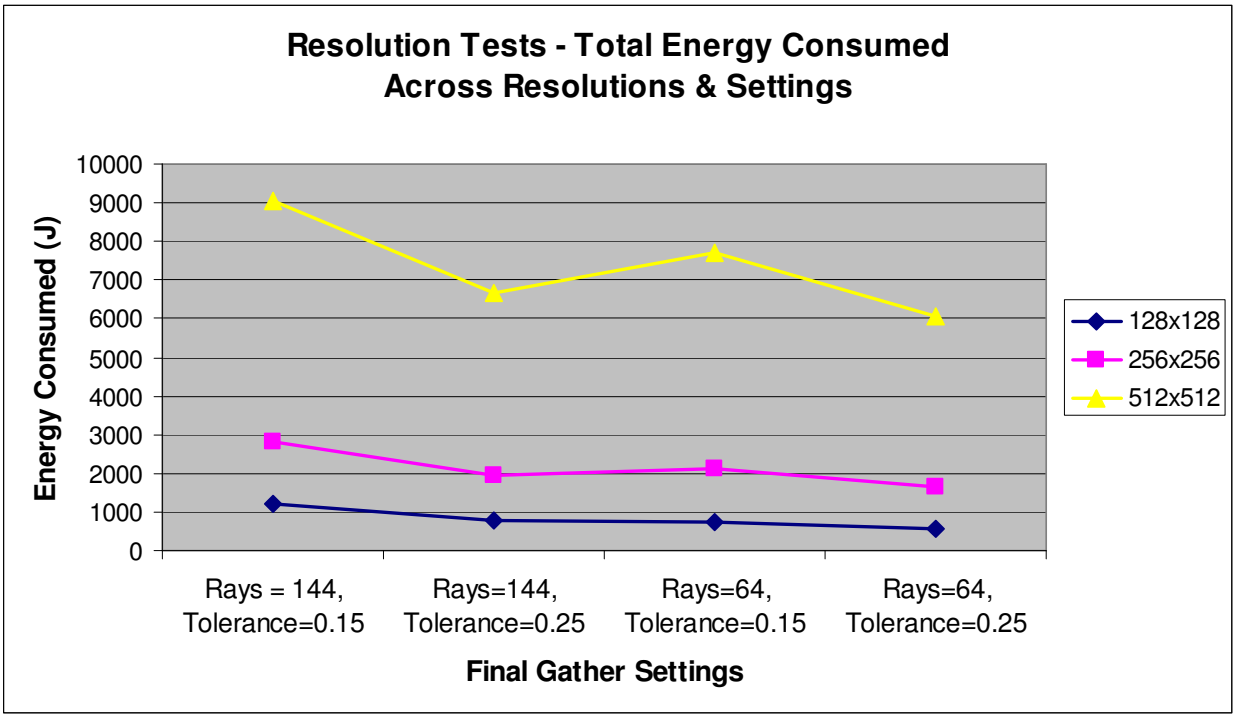

Figure 54: The total energy cost of four different configurations at three different resolutions.

The total energy used by four different configurations of the renderer at three different resolutions is shown in Figure 54. As one would expect, more energy is required to render images at higher resolutions. It is also expected that configurations with a lower tolerance consume more energy than those with a larger tolerance. Beyond those points, looking at the total energy used and the image difference alone does not yield any new information. Instead, comparisons made using energy consumption per-pixel and image difference per-pixel were found to be more informative. Energy use per-pixel is calculated by dividing the total energy used to render the image by the number of pixels in the image. Similarly, the image difference per-pixel is calculated by dividing the image difference from *ltdiff* and dividing it by the number of pixels in the image.

The first set of tests within the resolution batch used four different configurations of the direct visualization method. Figure 55 shows the energy used per-pixel from these tests. As one would suspect, increasing the number of photons stored increases the energy use. The more interesting point featured in Figure 55 is that while the energy cost per pixel at each resolution is the same, the lower resolutions have a higher cost per pixel. This is a result of the overhead of creating the photon map. Photon map creation takes a constant amount of time regardless of the

resolution of the image. Figure 56 shows that the difference per-pixel across different resolutions is not significant. This means that as resolutions change, the image difference scales as well.

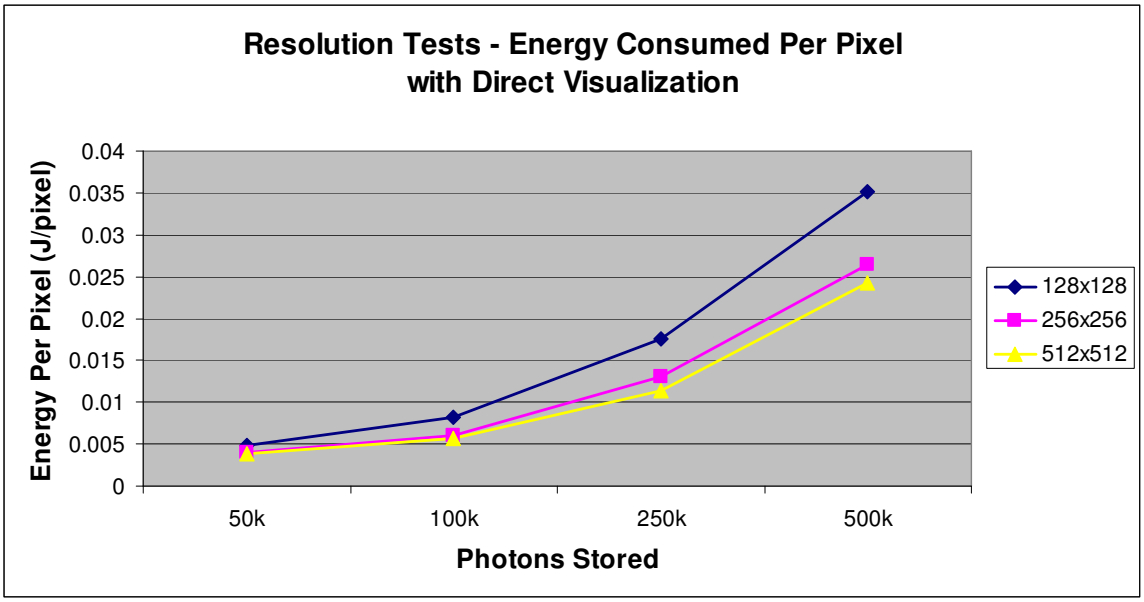

Figure 55: Energy used per-pixel for four different direct visualization configurations and three different resolutions.

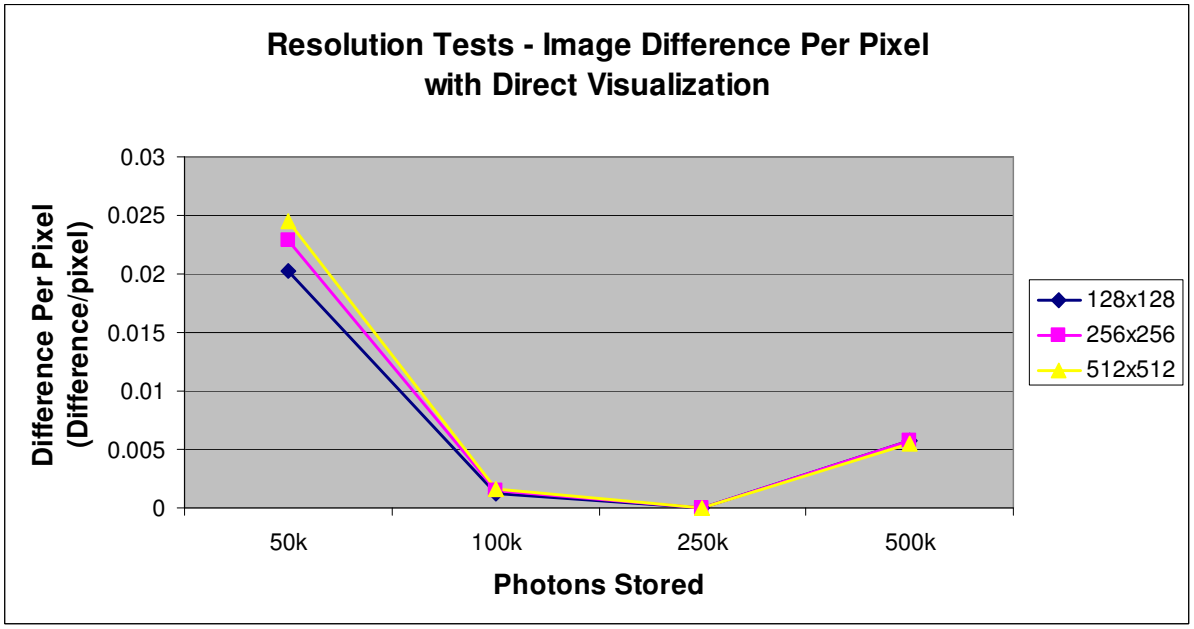

Figure 56: Image difference per-pixel for four different direct visualization configurations and three different resolutions.

It should be noted that before dividing the image difference values by the pixel count, the image difference values for images at 512x512 were several orders of magnitude higher than

128x128. For instance, the original image difference of the tests at 128x128 ranged between 20 and 350 units. The image difference for the tests at 512x512 ranged between 400 and 6500. The results presented in the previous section were all rendered using a 256x256 resolution. For those tests, an image difference below 100 was acceptable. Since image difference scales based on resolution, configurations yielding an acceptable image difference at a low resolution could render an unacceptable image at a higher resolution if the maximum image difference allowed was 100. A possible solution to this issue is to scale the acceptable image difference with the resolution, rather than use a constant. On the other hand, one might prefer to change the configuration to produce a higher quality image since artifacts are more noticeable at higher resolutions.

Several tests were also run using final gathering at different resolutions. The results from these tests are shown in Figure 57 and Figure 58. The conclusions drawn from the direct visualization tests remain the same for the final gather tests. The only minor difference is that the energy per-pixel deviates more across resolutions than in direct visualization.

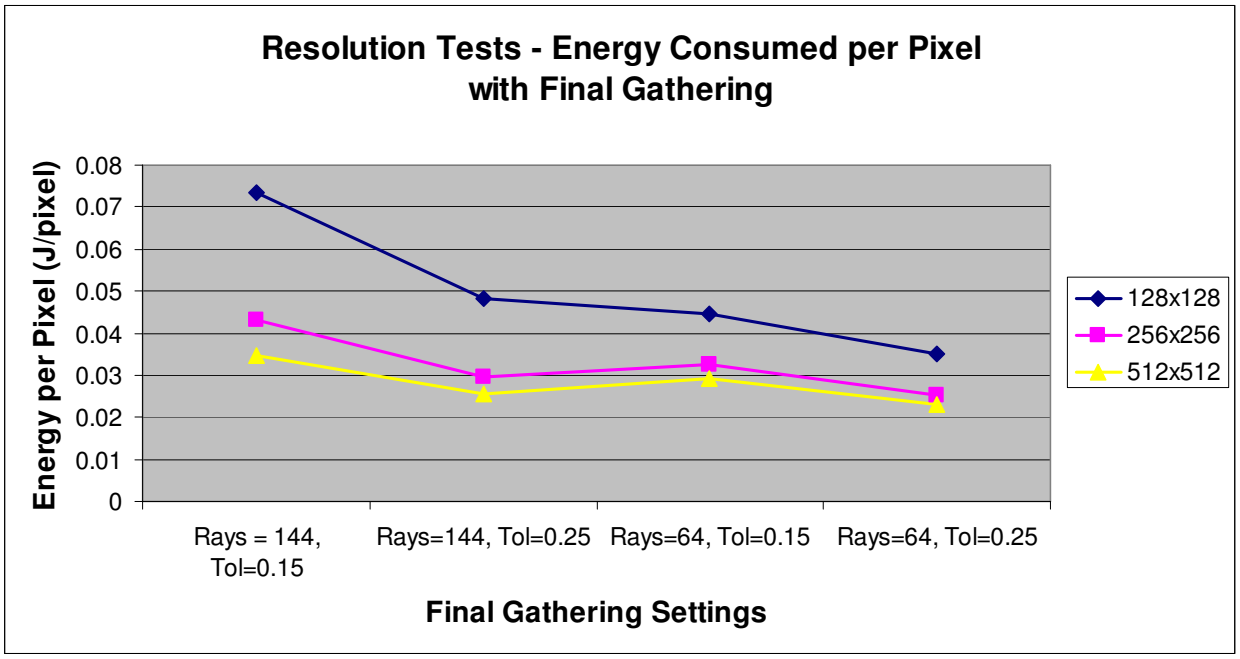

Figure 57: Energy used per-pixel for four different final gathering configurations across three different resolutions.

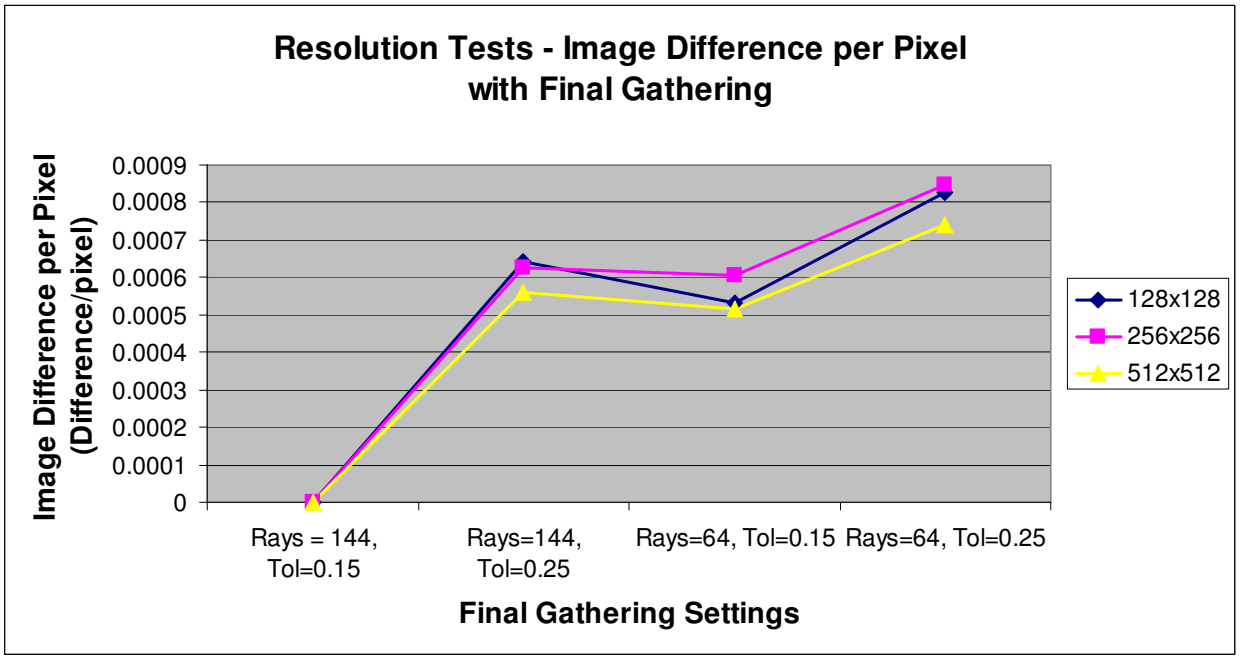

Figure 58: Image difference per-pixel for four different final gathering configurations and three different resolutions.

6.5 Results Summary

The results presented earlier in this section are condensed in Table 3 and Table 4. Both tables rate each setting by the impact they have on energy use and image quality. A low rating for energy impact implies that if the setting is varied significantly, the impact on energy use is minor. A high rating is the opposite where a small change in the setting has a great impact on energy use. A medium rating suggests that the effect on energy use is proportional to the degree the setting is changed.

Setting	Energy Impact	Image Sensitivity
Photon Count	Medium	Medium
Gather Count	Low	High
Gather Distance	High	High

Table 3: A quick summary of the energy impact and sensitivity of settings for direct visualization.

Setting	Energy Impact	Image Sensitivity
Precomputed Irradiance	High	Low
Irradiance Cache Tolerance	High	Low
Sample Rays per Final Gather	Medium	Medium
Photon Count	Low	Medium

Table 4: A quick summary of the energy impact and sensitivity of settings for final gathering.

Image sensitivity, also rated low to high, is based on the extent changing a setting had on the final image's deviation from a reference image as measured by *ltdiff*. Low sensitivity means that significant changes to a setting had little impact on image quality. For instance, changing the irradiance cache tolerance had a low sensitivity because images where tolerance was varied consistently differed less than 100 units from the reference image. A high sensitivity means that minor variations of a setting impacted the final image greatly. The gather count has a high sensitivity because tests where it was changed slightly reported over 300 units difference. Settings with a high energy impact and low image sensitivity are ideal for reducing energy consumption while maintaining an acceptable image quality. Note that photon count is rated differently in Table 4 than it is in Table 3 because photon count has a larger impact on total energy used for direct visualization than for final gathering.

Table 3 shows that an ideal tradeoff does not exist when using direct visualization for indirect lighting. Photon count is close with a medium energy impact and medium image sensitivity. However, as it is explained in 6.3.1, users should focus on generating the best possible image when using direct visualization.

Table 4 shows that precomputed irradiance and the irradiance cache tolerance were two settings that significantly reduced energy use without adversely diminishing image quality. The number of sample rays for a final gather was a more sensitive setting but it still offers a reasonable tradeoff.

7 Conclusions

The goal of this thesis was to establish reasonable tradeoffs between image quality and energy consumption for a photon-mapping based renderer. Most profiled aspects of the renderer did not yield such a tradeoff for two reasons: they were sensitive to changes in the configuration or accounted for such a small percentage of total energy use that modifications would have no impact. Those aspects that did yield a tradeoff were significant since they were the computations that required the most energy.

The most effective gains in energy reduction can be made when final gathering is used to render indirect illumination. Increasing the irradiance cache tolerance improves energy efficiency much faster while maintaining a higher degree of image quality than any other setting. Reducing the number of sample rays per final gather calculation also showed significantly reduced energy consumption while maintaining a high image quality.

Results have also shown that indirect lighting using direct visualization is very sensitive to deviations from the ideal configuration. Slightly reducing the total photon count in the global photon map will generate an image of a lesser quality faster than it would had final gathering been used. It is recommended that when using direct visualization the gather count and gather distance remain constant. Once a configuration yielding a high quality image is determined, the user can reduce the total number of photons slightly to save some energy.

It was noted that higher resolutions made artifacts between images easier to detect as evidenced by larger differences between high resolution images than low resolution images. However, increasing image resolution was shown to have little to no impact on per-pixel energy consumption and per-pixel image difference. This effect may be offset if the rendering configuration is set to produce higher quality images than lower resolution counterparts. It is safe to assume that mobile devices with higher resolution displays will also have more processing power available and can therefore generate higher quality images.

There are several optimizations that could be included in the renderer tested that would further increase energy efficiency without necessarily degrading image quality. Some of these include incorporating irradiance gradients, importance sampling, and user-specified hints for the renderer describing the location of caustic-generating objects in a scene. A comprehensive energy- efficient photon mapping renderer would be required to implement these features.

8 Future Work

Additional tradeoffs are likely to be found in more advanced areas of photon-mapping based rendering. Quasi-Monte Carlo sampling may help reduce the variance between images for direct visualization. Effects such as subsurface scattering and participating media have not been evaluated in terms of their energy efficiency.

Attention should be paid to direct lighting in an effort to discover other tradeoffs. One area of investigation would be how shadow rays and shadow photons can be better combined to improve energy efficiency.

Finally, research could be done on how well combining several tradeoffs impacts image quality. The results of such a study would summarize the proper way to minimize image degradation while maximizing energy efficiency.

9 References

[1] Starner, T. 2003. “Batteries and Possible Alternatives for the Mobile Market.” *IEEE Pervasive Computing*, pages 86-88.

[2] Flinn, J. and Satyanaranan, M. 1999. “PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications.” *Second IEEE Workshop on Mobile Computing Systems and Applications*.

[3] Banerjee, K. and Agu, E. 2005. “PowerSpy: Fine-Grained Software Energy Profiling for Mobile Devices.” IEEE WirelessComm Conference Proceedings.

[4] Jensen, H. W. Retrieved May 10, 2007 from http://graphics.ucsd.edu/~henrik/.

[5] Farkas, K. I., Flinn, J., Back, G., Grunwald, D. and Anderson, J. 2000. “Quantifying the Energy Consumption of a Pocket Computer and a Java Virtual Machine.” *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 252-263.

[6] Stemm, M., and Katz, R.H. 1997. “Measuring and Reducing Energy Consumption of Network Interfaces in Handheld Devices.” *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Science*, pages 1125-1131.

[7] Feeny, L., and Nilsson, M. 2001. “Investigating the Energy Consumption of a Wireless Network Interface in Add Hoc Networking.” *INFOCOM 01*, pages 1548-1557.

[8] Tscheblockov, T. 2004. Power Consumption of Contemporary Graphics Accelerators.” *X-bit Labs*. Retreived on May 9, 2007 from http://www.xbitlabs.com/articles/video/display/ati-powercons.html.

[9] Lohrmann, P. 2006. *Energy-Efficient Interactive Ray Tracing of Static Scenes on Programmable Mobile GPUs*. Master’s Thesis. Worcester Polytechnic Institute.

[10] Chang, C. H. 2006. *The Study of Energy Consumption of Acceleration Structures for Dynamic CPU and GPU Ray Tracing*. Master’s Thesis. Worcester Polytechnic Institute.

[11] “Bidirectional scattering distribution function.” Retrieved April 10, 2007 from http://en.wikipedia.org/wiki/Bidirectional_scattering_distribution_function.

[12] Hanrahan, P. and Krueger, W. 1993. ”Reflection from layered surfaces due to subsurface scattering.” *Proceedings of SIGGRAPH ’93, Computer Graphics Proceedings, Annual Conference Series*, pages 165-174.

[13] Jensen, H. W., Marschner, S., Levoy, M. and Hanrahan, P. 2001. "A practical model for subsurface light transport." *Proceedings of SIGGRAPH 2001, Computer Graphics Proceedings, Annual Conference Series,* pages 511-518.

[14] Nicodemus, F. E., Richmond, J. C., Hsia, J. J., Ginsberg, I. W., and Limperis, T. 1977. "Geometric considerations and nomenclature for reflectance." *Monograph 161*, National Bureau of Standards (US).

[15] Hill, F.S. 2001. *Computer Graphics using OpenGL* (2nd ed.). Prentice Hall, Upper Saddle River, NJ.

[16] Waters, Z. 2003. "Photon Mapping". Retrieved April 10, 2007 from http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/photon_mapping/PhotonMapping.html.

[17] Jensen, H. W. 2001. *Realistic Image Synthesis Using Photon Mapping*. A K Peters. Wellesley, MA.

[18] Goral, C. M., Torrance, K. E., Greenberg, D. P., and Battaile, B. 1984. "Modeling the interaction of light between diffuse surfaces." *Proceedings of the 11th annual conference on Computer Graphics and Interactive Techniques*, pages 213-222.

[19] Cohen, M. F., and Greenberg, D. P. 1985. "The hemi-cube: a radiosity solution for complex environments." *ACM SIGGRAPH Computer Graphics*, vol. 19 no. 3, pages 31-40.

[20] Immel, D. S., Cohen, M. F., Greenberg, D. P. 1986. "A radiosity method for non-diffuse environments." *ACM SIGGRAPH Computer Graphics*, vol. 20 no. 4, pages 133-142.

[21] Wallace, J. R, Cohen, M. F., and Greenberg, D. P. 1987. "A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods." *ACM SIGGRAPH Computer Graphics*, vol. 21 no. 4, pages 311-320.

[22] Dorsey, J., Edelman, A., Jensen, H. W., Legakis, J., and Pedersen, H. K. 1999. "Modeling and rendering of weathered stone." *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 225-234.

[23] Jensen, H. W. and Christensen, P. H. 1998. "Efficient simulation of light transport in scences with participating media using photon maps." *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 311-320.

[24] Jensen, H. W., Christensen, N. J. 1995. "Efficiently rendering shadows using the photon map." *Compugraphics*, pages 285-291.

[25] Christensen, P H. 1999. "Faster Photon Map Global Illumination." *Journal of Graphics Tools*, vol. 4, no. 3, pages 1-10.

[26] Whitted, T. 1980. “An improved illumination model for shaded display.” Communications of the ACM, vol. 23 no. 6, pages 343-349.

[27] Glassner, A., ed. 1989. *An Introduction to Ray Tracing*. Morgan Kaufmann. San Francisco, CA.

[28] Shirley, P. and Morley K. R. 2001. *Realistic Ray Tracing* (2nd ed.). A.K. Peters, Wellesley, MA.

[29] Phong, B. T. 1975. “Illumination for computer generated pictures.” *Communications of the ACM*, vol.18 no. 6, pages 311-317.

[30] Ward, G. J., Rubinstein, F. M., and Clear, R. D. 1988. “A Ray Tracing Solution for Diffuse Interreflection.” *Computer Graphics*, vol. 22, no. 4.

[31] Ward, G. J., Heckbert, P. 1992. “Irradiance gradients.” *Third Eurographics Workshop on Rendering*, pages 85-95..

[32] Lindstrom, P. 2000. *Model Simplification using Image and Geometry-Based Metrics*. Doctoral Dissertation. Georgia Institute of Technology.

[33] Microsoft. “CallNtPowerInformation”. *Microsoft Developer Network*. Retrieved May 7, 2007 from http://msdn2.microsoft.com/en-us/library/aa372675.aspx

[34] Teo, P. C. and Heeger, D. J. 1994. "Perceptual Image Distortion". *Proceedings of the International Conference on Image Processing*, pages 982-986.

[35] Williams, N., Luebke, D., Cohen, J., Kelley, M., and Schubert, B. 2003. “Perceptually Guided Simplification of Lit, Textured Meshes.” *Proceedings of the 2003 ACM SIGGRAPH Symposium on Interactive 3D Graphics*.

[36] Hewlett-Packard. Retrieved May 9, 2007 from http://h20000.www2.hp.com/bc/docs/support/SupportManual/c00622160/c00622160.pdf