Lightcuts and Illumination

CS384G - Final Project

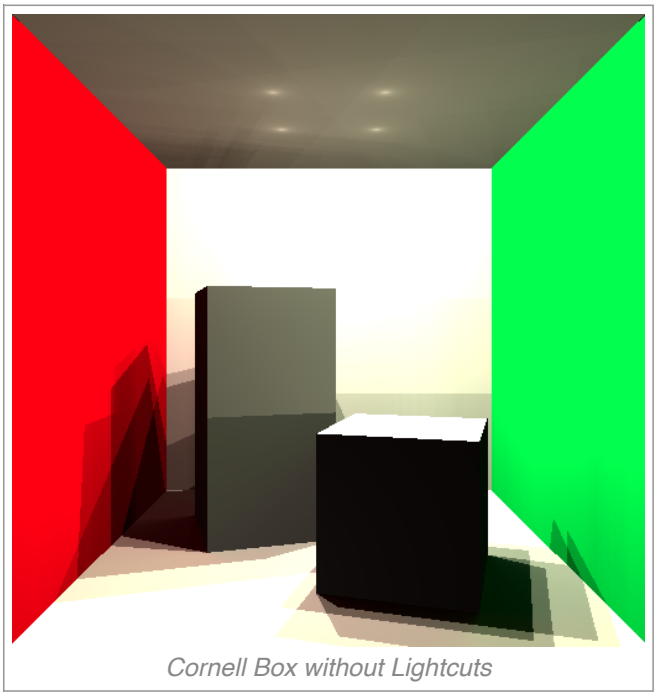
Nicholas Kelly, Saddam Quirem, Mike Verlinden

Introduction and Project Statement

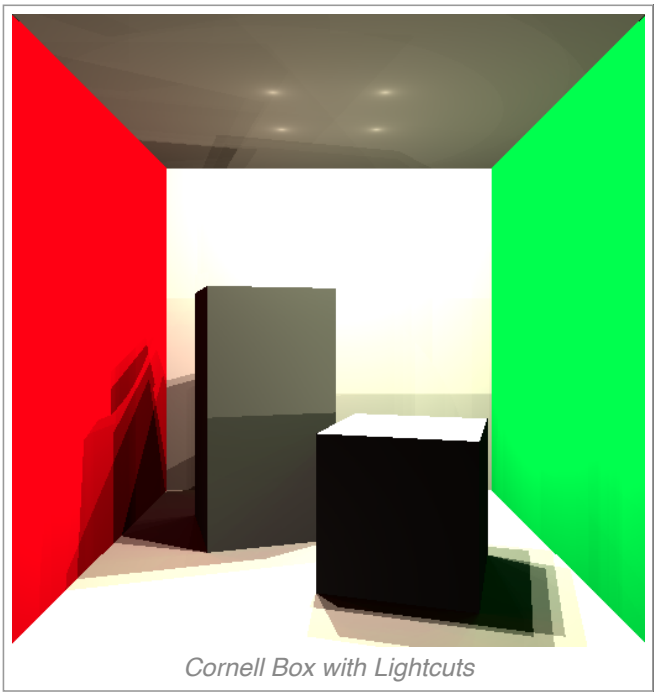
While acceleration structures such as k-d trees or bounding volume hierarchies greatly reduce the ray-object intersection complexity, every light source still needs to be assessed during the Phong shading. This can be troublesome for scene with large numbers of light sources, luminous objects, or global illumination. Lightcuts is a technique that seeks to reduce the overhead of iterating through a large number of light sources, by combining nearby light sources together. This of course comes with a certain amount of error, an error for which a threshold can be preset. This projects seeks to implement objects and methods which creates large amounts of light sources within the scene and demonstrate the acceleration in scene rendering with a large number of light sources using lightcuts.

Lightcuts

Simple scenes and shaders will often use a smaller number of lights that are either supported natively with acceleration structures or with a small number array of sources. Summing up the contribution of all the lights in the scene is often not the bottleneck in the rendering process. However, more advanced lighting techniques can use thousands, tens of thousands or even more lights to illuminate a scene with various effects. Global illumination strategies often create thousands of virtual lights sources to create more realistic lighting. If rendering time is concern, then it is not practical to iterate over every light especially since the contribution from each sources varies greatly and most lights will not contribute enough to justify inclusion in every calculation. Lightcuts is a method for grouping lights together so that you can represent their total contribution with one coalesced light. Lights are arranged in a tree of nodes, similar to a KD tree, such that each node has a coalesced light representing the contribution of all lights contained in the node or all lights contained in its children (and their children on down). The node also includes an error term to describe how well it represents the lights under it and each successive level of the lightcuts tree has less error than its parents. Both the error and the representative light are meant to be as direction independent as possible. When illuminating a particular point in the scene, will start with the root node and compare its distance to the error term for that node. If the ratio is good enough then it will use the coalesced light only, if not it will drop down another level in the tree and make the same calculation. At some point it will either decide it no longer needs to traverse deeper past a node in the tree or will reach a leaf node and decide to use each individual light. Lightcuts are designed to give a disciplined approach to approximating the effects of a large number of lights in a much faster manor. Most errors with lightcuts are seen as harsher or overshadowed regions. due to the coalesced light being blocked more than the real set of lights would.



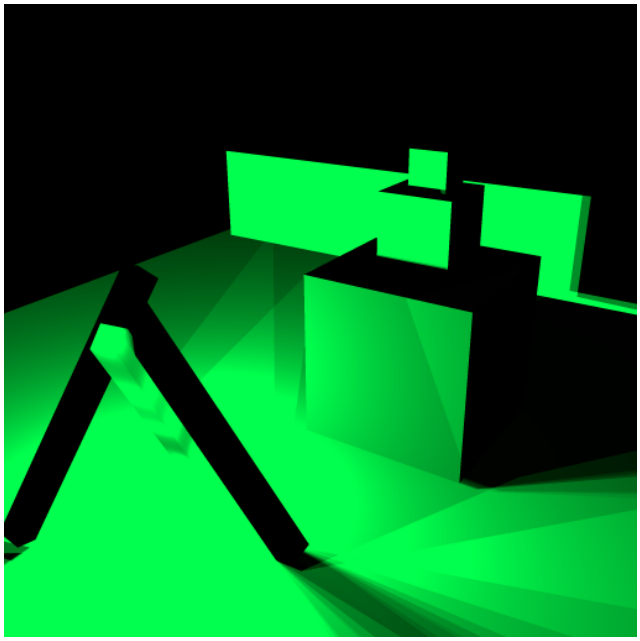
Cornell Box without Lightcuts



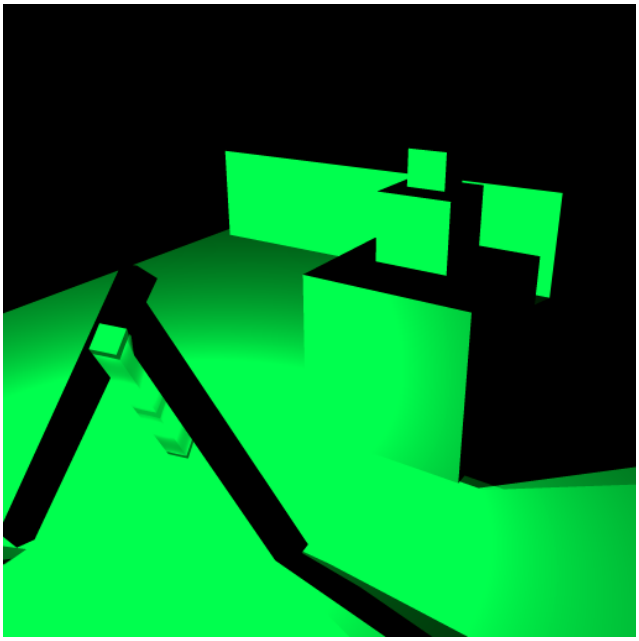
Cornell Box with Lightcuts

For our implementation we followed a similar approach to the Bruce Walter et al. [1] with a few differences in areas where it

was less clear what their exact algorithm is or we felt was better suited to our application. When creating their trees, they would use clustering and a similarity metric that wasn't fully described in the paper. We wanted to avoid clustering since it can scale poorly with larger numbers of lights and would adversely impact how long we could spend on testing. We instead use a similar method to k-d tree construction where we sort the lights by position about all three axises and look for the best place to divide by prioritizing even splits over lopsided splits and locations where two adjacent lights have a large divide in distance along that axis and/or color. Part of the assumption is that many of the scenes we want to render will have clusters of lights that would be easy to recognize in this way and partition separately. When selecting a coalesced light they take a weighted random sample from the lights in that node; where as, we create a representative light that combines the characteristics of each light weighed by their intensity except for the intensity of the light which is a sum of all components. It wasn't clear precisely what their error metric is, so for ours we find the average distance from each light to the centroid of those lights with each contribution weighted by intensity. We also scale the distance based off of the normalized color difference. When checking to see if we need to drop another level in the tree, we compare the ratio of the scaled error distance and distance to the point with an adjustable error term. Finally if a coalesced light is blocked by an opaque object before it lights the point we are testing we use both of its children's coalesced light sources instead (this is not recursive). The idea is that the coalesced light may be blocked when many of the other lights aren't or would contribute to a softer shadow so we drop one level which will have the largest separation and likelihood of contributing to the seen (each level afterwards not only contributes less to the scene but also is less likely to not be blocked as well.



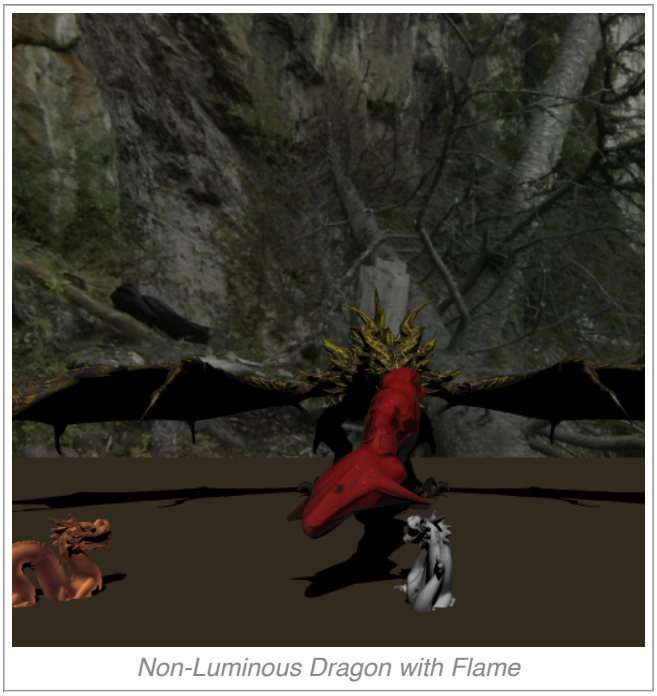
Emitting rod without Lightcuts



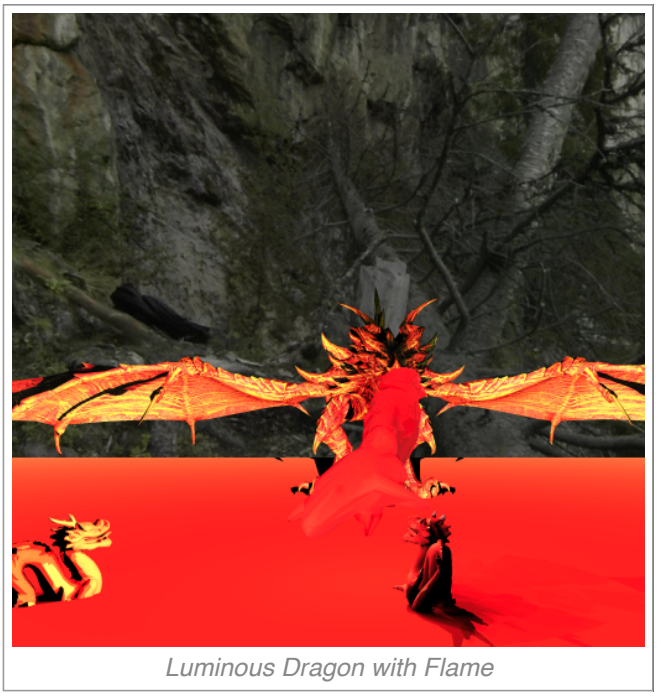
Emitting rod with Lightcuts

Object Lights

In order to utilize the light performance of Lightcuts, we provide a method of creating arbitrarily shaped light sources. To accomplish this, we associated a "luminosity" with a material which then can be assigned to any object. A luminosity contains information about the light source to be added as well as the density of the light sources. Upon loading the scene, each object which has a luminosity is parsed and object lights are generated. Lights are placed on the surface of the object and light density as determined by the surface area and the density scaling factor. Placement is done depending on the amount of lights for the surface (variable based upon surface area).

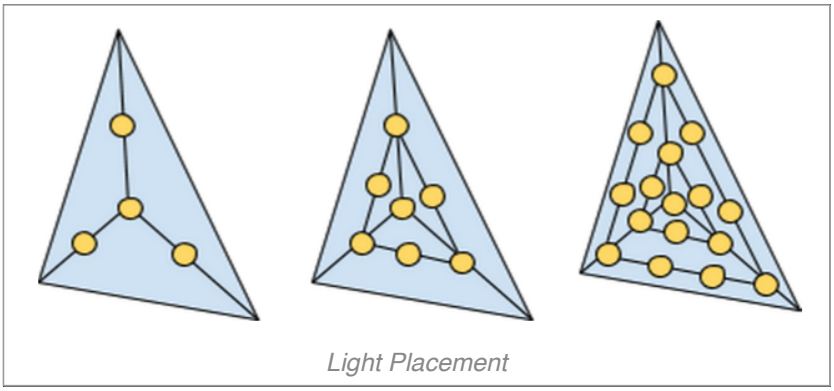


Non-Luminous Dragon with Flame



Luminous Dragon with Flame

The figure below depicts how the placement of the lights is done on a triangular surface. A light within the centroid of the triangle is always done regardless of density. Then depending on the surface area of the triangle with respect to the entire object and the density scalar, a number of lights are calculated for the surface. Lights are first assigned to the lines connecting the centroid and each vertex. These are done at equal dividers depending on the total number of lights to add. Then, lights are added on the lines between those light sources. Due to the scheme, placement will proceed in an iterative manner until the light budget for the surface is fulfilled. Thus, the overall density of the triangle may not be as balanced as the ideal configurations. The light sources are given the same properties as specified but the intensity of each light source is scaled by the overall amount added.



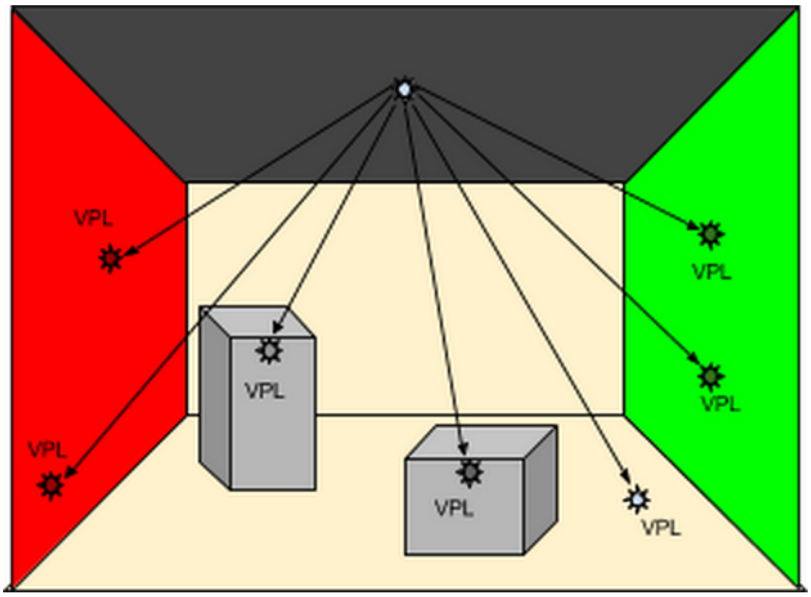
Light Placement

Textures

Additionally, textures were added to the Trimesh objects. In order to support textures, the parser was extended to accept texture coordinates for the polymesh. In terms of the actual mapping of texture coordinates to vertices, we created an OBJ to RAY converter which effectively allows us to use any publically available OBJ models and their associated textures (after converting to BMP). Not only do the textures apply to the diffuse term of the objects, but they can be used to change the color of the object lights. In this manner, the barycentric coordinates are calculated during light placement and the color for the light is taken from the texture.

Global Illumination & Instant Radiosity

For this project, we use a very simple implementation of instant radiosity[2] to create global illumination. Instant radiosity works by casting out a set number of light rays from direct light sources and creates virtual point lights with the diffuse color of the material that was intersected.

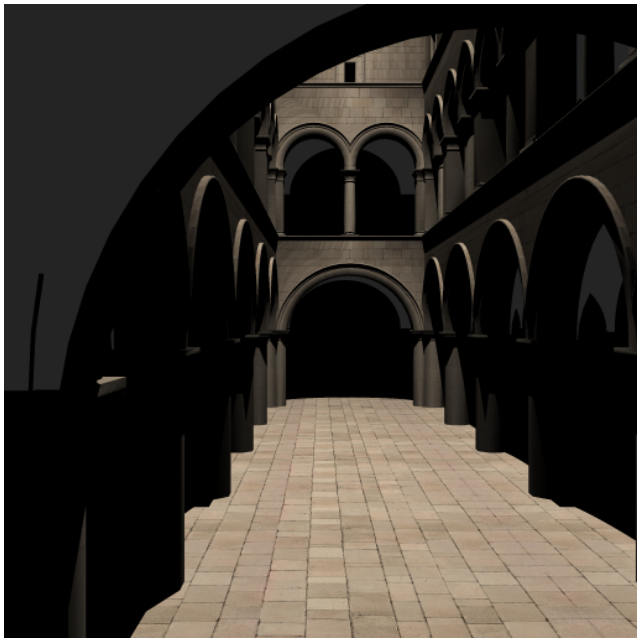


Global Illumination

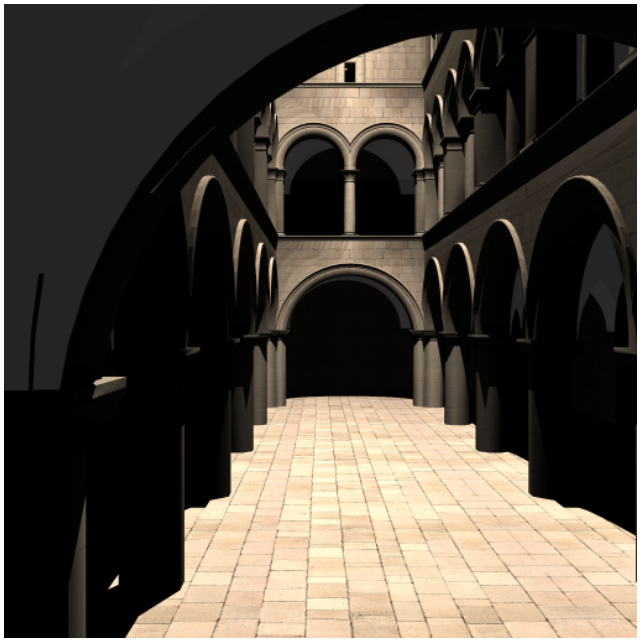
The results show that using lightcuts reduced the rendering time by an order of magnitude, while still creating a similar image. The instant radiosity model implemented does not include multiple bounces, but rather illuminations of objects in direct view of a direct light source. The image below contains ~10K virtual point lights, and shows the red and green illuminations on either side of the Cornell boxes and the ceiling. So that the scene is not saturated by the virtual point lights, the intensities of each VPL is attenuated by the number of VPLs present in the object the VPL is derived from.

Church without Global Illumination

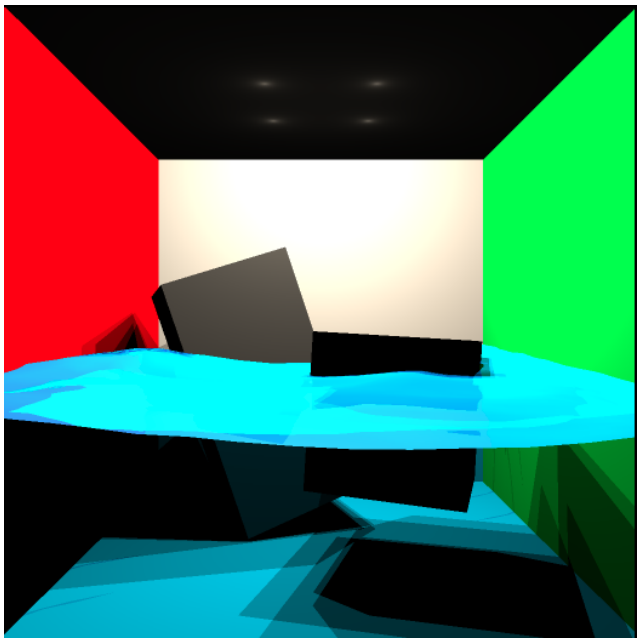
Church with Global Illumination



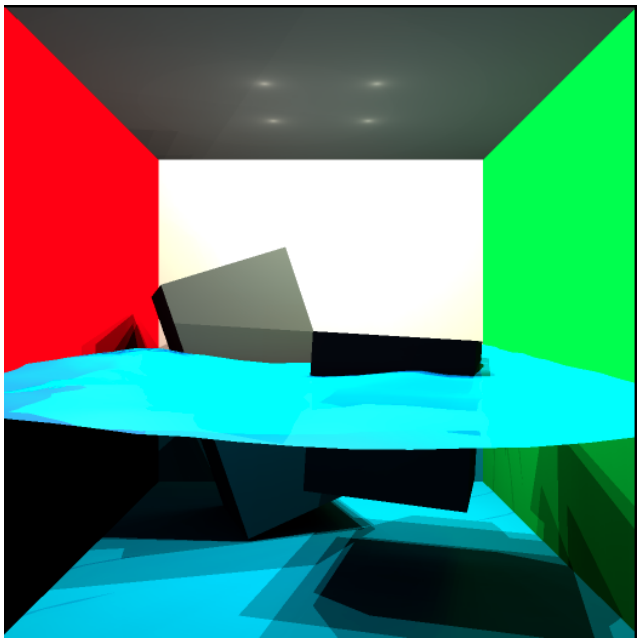
Roman Architecture without Global Illumination



Roman Architecture with Global Illumination



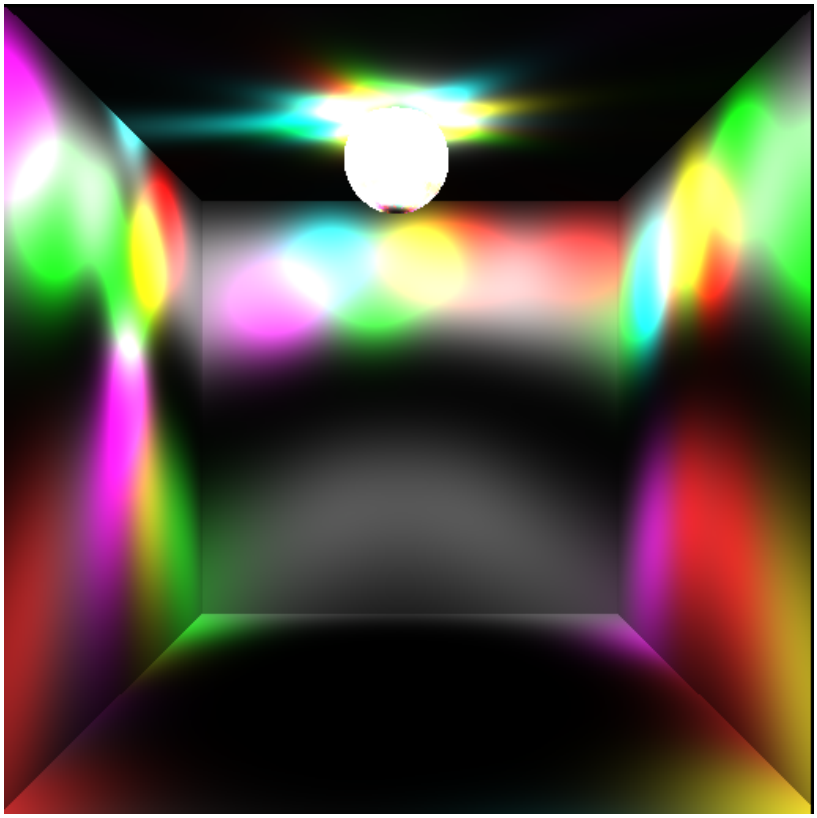
Cornell Water without Global Illumination



Cornell Water with Global Illumination

Spotlights

Support for spotlights was added in the parser and in the light sources. A spotlight is very similar to a pointlight, but the distance attenuation term is based upon the cosine (dot product) of the light source direction. In addition, a spread parameter is specified (exponent) which can be used to effectively widen or narrow the spotlight. Spotlights were initially added in order to approximate caustics in the global illumination. In this manner, spotlights would be inserted in areas of either light reflection or transmission. By inserting these sources, it would allow light to accumulate in certain areas, while being absent in other. However, we did not have time to finish this aspect.



Disco ball with Spot Lights

Performance Results

Test Case	No Extra Lights	No-Lightcuts	Lightcuts
Green Light Emitter Luminous Object	N/A	43s 11.8219 Lights/Pixel	7s 1.50233 Lights/Pixel
Cornell Box Instant Radiosity (8638 VPLs)	2s	2031s 8638 VPLs/Pixel	95s 273.64 VPLs/Pixel

References

1. Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. 2005. Lightcuts: a scalable approach to illumination. ACM Trans. Graph. 24, 3 (July 2005), 1098-1107. DOI=10.1145/1073204.1073318
2. Alexander Keller. 1997. Instant radiosity. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 49-56. DOI=10.1145/258734.258769
3. Dr. Fussell's class slides.