Radiosity

Kjell Isak Nyberg

September 8, 2019

Abstract

In the modern day of image and video rendering the essential goal is to reach an image which resembles reality as closely as possible. The search for rendering methods within the field of computer graphics is very extensive and a large number of techniques have been invented. Global Illumination Algorithms (GIA) are one of these techniques. GIAs works with the principle of indirect illuminations, incorporating light reflections in the lighting of a render. Radiosity is one type of GIA and is capable of rendering images which in some cases are indistinguishable from real life. This algorithm in particular will be further explained and compared with other means of image rendering, both GIA and other families of algorithms.

Keywords *Radiosity, Global Illumination Algorithms, Ray Tracing, Form Factor, Adaptive Meshing*

Contents

1 Introduction

The primary goal of image rendering has always been to create images which appear as realistic as possible. One attempt to reach this achievement was the invention of Global Illumination Algorithms (GIA) which includes algorithm such as radiosity and ray tracing. There are two main approaches to illumination within the field of computer graphics: direct illumination and indirect illumination. Direct illumination mainly concerns itself with light travelling directly from a light source, while indirect illumination also takes the light reflected of all the surfaces in the scene into account. This results in that indirect illumination is considered to be the superior when it comes to modelling realistic light behaviour. GIAs implement indirect illumination as the name Global Illumination suggests.

While radiosity came about in the field of computer graphics in the year 1984, the same algorithm had been used to model heat transfer in the field of thermodynamics many years before. This type of modelling was directly applicable to computer rendering, as heat and light act interact with surfaces in a very similar manner. [MFC86] The principle behind radiosity builds on there being a finite number of surfaces within a scene and a known total amount of light let into the environment, for example a light bulb with a known brightness. These surfaces are split into smaller patches that are allowed to reflect light onto each other. This light is propagated until the sum of the light of all the patches approaches the total amount of light initially let into scene.

2 Patches

As previously mentioned, the algorithm uses so called *patches* which are smaller sections of the surfaces of the objects in the scene. It is assumed that every point within a single patch interacts with the rest of the scene in the exact same way. Thus a midpoint is assigned to each patch that represents each point within the patch. It is thus assumed that the amount of light incident on the midpoint is equal to the amount on any other point in the patch. This reduces the computation needed.

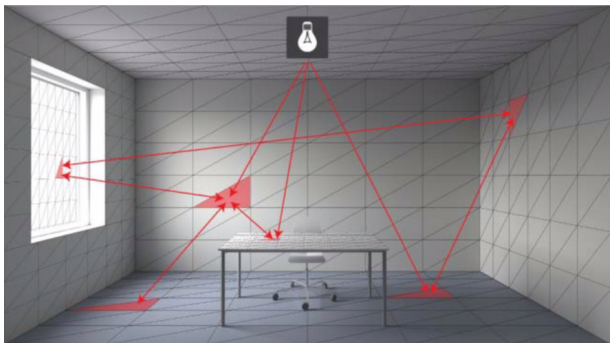


Figure 1: Radiosity illustration. [AI13]

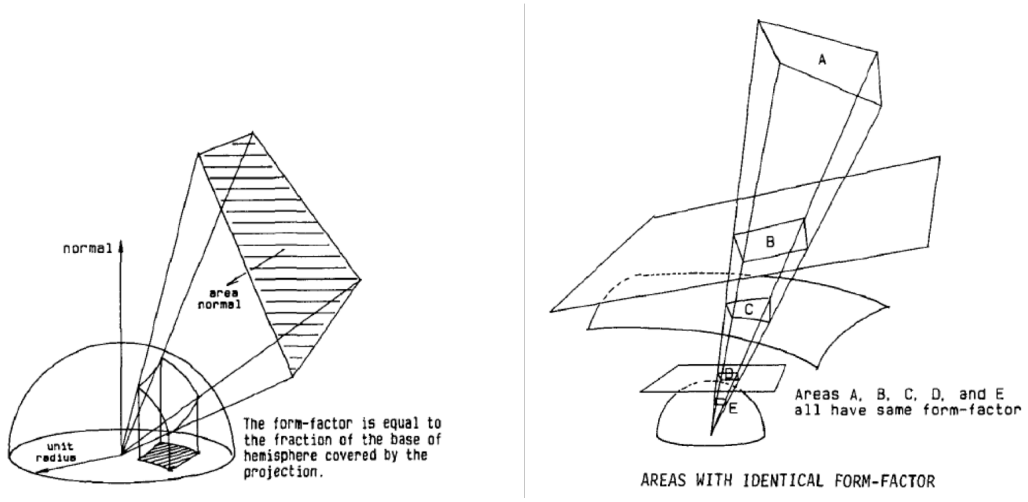
Figure 1 shows an environment with triangular patches in a room with a chair and a desk, that is lit up by a light bulb. The light bulb introduces the initial light into the scene shown by the red arrow coming out and hitting the different patches. These patches then reflect the light onto different patches in the scene. Note that all arrows coming out of a patch are double headed as light is reflected between the surfaces in both directions. With this approach, the light levels of a surface also becomes a property of that surface, rather than a property of the viewing angle of the observer. As a result, radiosity is viewpoint independent and the position of the observer does not affect the final outcome.

3 Form Factor

In simple, the *form factor* between two surfaces is the ratio of the light leaving one patch and arriving at the other patch. This is vital as the diffusion of light is the key aspect of radiosity as an algorithm. While this description is trivial to fathom, it is significantly harder to implement. Let the form factor of light leaving patch i and arriving at patch j be represented by F_{ij} and n be the number of patches in a scene. The form factor makes use of the property that, in a closed scene, the sum of all form factors leaving one patch is equal to 1. Which can be expressed as:

$$\sum_{j=1}^{n} F_{ij} = 1$$

The main properties that influence the form factor between two patches is the area of either patch, the angle formed between the two patches, and lastly the distance between the two patches. The explanation for this is quite trivial as a smaller area will have less light hitting it, two patches facing each other will exchange more light than two patches facing away from each other, and the further away something is the less light will reach the surface. [CG85]



(a) Nusselt Analogue Hemisphere

(b) Patches with Equal Form Factors

Figure 2: Form factor Hemisphere implementation [CG85]

These properties are all accounted for in the *Nusselt Analogue*, illustrated in figure 2 part a. This Analogue works by placing a unit hemisphere centred around the viewpoint. The patch is now projected onto the surface of the hemisphere, the hemisphere then in turn projects the same surface down onto the base of the hemisphere. Finally the form factor is equal to the area for the projection on the hemisphere base divided by the total area of the hemisphere's base. For the case of radiosity this is a simplified model of the Analogue, real implementations are different for efficiency. As shown in figure 2 part b, different patches may have the same form factor while being different in size, distance and orientation with relation to the viewpoint. This property of the hemisphere is used to improve the efficiency of the form factor calculation by creating a so-called *hemicube*

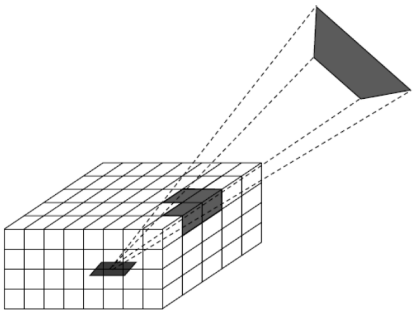


Figure 3: Hemicube [CG85]

Figure 3 provides a view of a so called *hemicube* which is constructed by placing a cube with its centre on the viewing point. The surfaces of the sides of the cube are divided into smaller discrete areas referred to as pixels. The form factor of each pixel is already pre-calculated. This means that rather than using the Nusselt analogue from figure 3, all the program has to do is to sum the form factors of each pixel that is projected onto. Like the number of patches, the size and number of pixels on the *hemicube* heavily influences the final outcome of the form factor. [CG85]

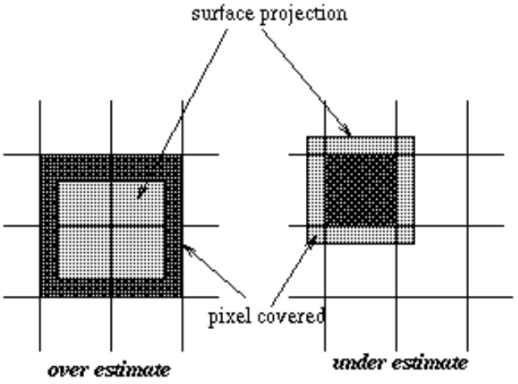


Figure 4: Hemicube Aliasing [RH90]

A major drawback of quantizing the surface of the hemisphere is the treatment of edge cases. Figure 4 gives an example of both over-, and underestimates. In figure 4 the light grey area is the actual image incident on the hemicube and the black is the pixels that are responding. The former side showcases a scenario in which the projection is overestimated as all four pixels are responding despite none of them being entirely filled up. This causes the projection to appear larger than initially intended. Similarly, in the latter side of the figure, the size of the projection is decreased due to the pixel being smaller than the projection resulting in that the projection becomes smaller despite the surrounding pixels being partially filled up. This problem can be tackled similar to pixels in a normal computer screen, where decreasing the size of the individual pixels while increasing the number of pixels the projections would fit better by minimizing these rounding errors. The major drawback is that increasing the number of pixels also increases the computation needed to find the final projection. Hence the amount of computing needed must be balanced with the necessity to approximate the form factor appropriately.

4 The Radiosity Equation

In a scene, the radiosity of a patch is given by the total light or illumination of a patch divided by the area of the patch. This can be computed by finding the sum of the direct and indirect illumination incident on the patch's surface and the light emitted by the surface itself. The self-emitted light is already a property of the surface. Let this value be assigned to the variable E for *emission*. Finding illumination incident on the paths' surface from other sources is a greater challenge and is quintessential to radiosity. As mentioned previously, the radiosity algorithm takes the diffusion of all other patches into account when calculating the final radiosity of a patch. This is done by first finding the radiosity of a different patch and multiplying that by the form-factor. Let the radiosity of a surface j be B_j and the form-factor of patch j to patch i be F_{ji}. This needs to be calculated for each patch in the scene since they all diffuse light, let n represent the number of patches in the scene. Hence the indirect illumination can be expressed as the sum of the product of the radiosity and form-factor of each other surface in the scene, lastly this sum needs to be multiplied by the reflectivity of the patch denoted by ρ. With all this in mind, the radiosity of patch i (R_i) can be mathematically expressed as:

$$R_i = \rho_i \sum_{j=1}^{n} B_j F_{ij}$$

After adding direct incident light, the final radiosity at patch i is: [MFC98]

$$B_i = E_i + R_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}$$

The problem with this formula is that in order to figure out the radiosity of one patch, you need to know the radiosity of a different patch, presenting a catch-22 situation. The way to get around this is to reconfigure the equation into the matrix form and calculate the final radiosity of all patches using the Gaussian Elimination algorithm, where B_1 to B_n are the unknowns, and the remaining variables are known.

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & \dots & \dots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & & & & -\rho_2 F_{2n} \\ \vdots & & \ddots & & & \vdots \\ -\rho_i F_{i1} & & & 1-\rho_i F_{ii} & & -\rho_i F_{in} \\ \vdots & & & & \ddots & \vdots \\ -\rho_n F_{n1} & \dots & \dots & \dots & \dots & 1-\rho_n F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_i \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_i \\ \vdots \\ E_n \end{bmatrix}$$

5 Efficiency

With the above equation in mind, it is easy to deduce that the radiosity algorithm has a complexity of $O(n^3)$ where n is the number of patches in the scene. This comes from the complexity of the Gaussian algorithm. In addition to a bad runtime, the space complexity also proportional to $O(n^2)$. While modern implementations have come with clever improvements to the algorithm, working the storage efficiency down to $O(n)$. [MFC98] This complexity being proportional to the number of patches creates a big problem for generating larger scenes, this makes it important to select the patches efficiently because otherwise the complexities of the scenes rise extremely fast. This is where techniques such as Adaptive meshing become a vital improvement to the Radiosity.

6 Adaptive meshing

With a complexity of $O(n^3)$ is it very apparent that using an excessive number of patches will drastically worsen the performance of the algorithm. The most simple approach would be to divide the scene up evenly with evenly spaced and sized patches.

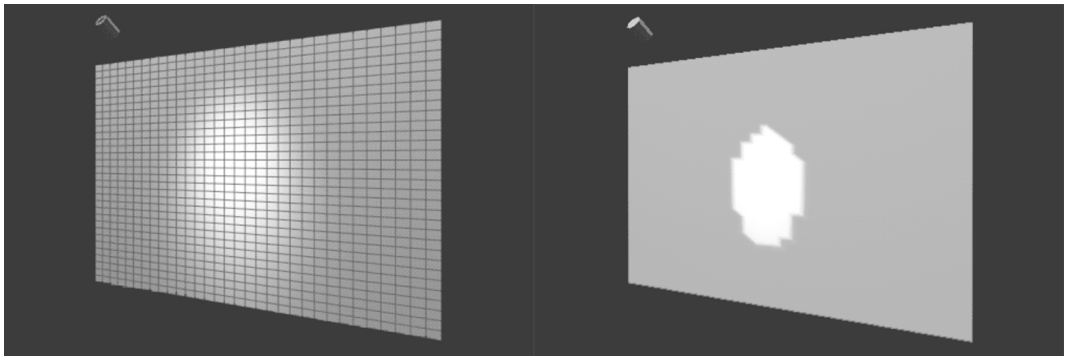


Figure 5: Non-adaptive Meshing [Lig]

This approach is portrayed in figure 5. The left portrays a scene of a light source directed at a canvas with a grid-type mesh superimposed on the canvas surface. The right shows the final render of the same scene with the given patch mesh. It is very evident that the final product does not quite resemble the desired outcome with very jagged edges to the point of the circular shape not appearing circular at all. These pixel like shapes are called artifacts. [Lig] Needless to state, these artifacts are not desired. One solution could simply be to make the patches smaller, decreasing the size of the artifacts overall. However this would also result in that the number of patches would also increase, and as stated previously with a complexity of $O(n^3)$, that simply would just render the extra computation needed supererogatory. Instead, a clever way of circumventing this problem is the use of adaptive meshing.

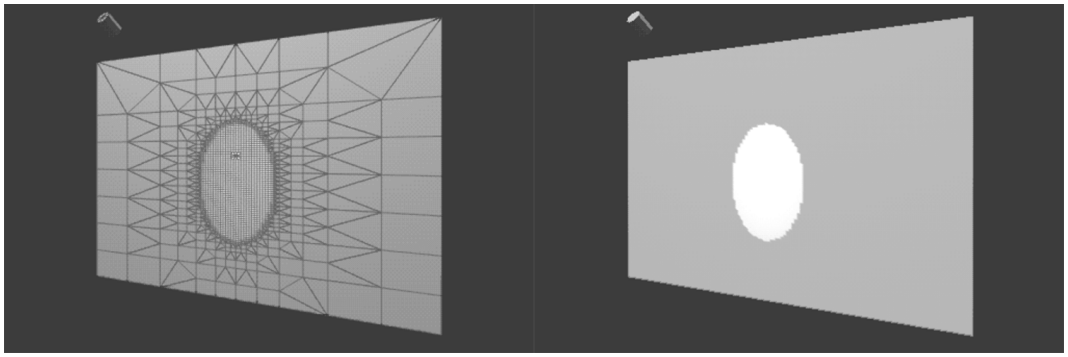


Figure 6: Adaptive Meshing [Lig]

Figure 6 is the same scenario as figure 5, however this time the mesh is different. The way adaptive meshing works is that rather than assigning an arbitrary number and size to the patches, the patches are assigned by finding the areas with similar radiosity levels and combining them into bigger patches, as seen on the outer most patches on the canvas on the left hand side. Likewise the patches that are subject to a steeper change of radiosity are made smaller in order to preserve detail. The efficiency of this strategy is very apparent on the right hand side render where in contrast to figure 5 the light shone onto the canvas is clearly rounder and the artifacts are far less conspicuous. Adaptive meshing enables the complexity to be allocated efficiently to the areas where it is needed.

7 Limitations of Radiosity

While radiosity is great for diffusing light, it has a number of limitations. As previously mentioned, the radiosity algorithm is based on thermal physics and modelling. This means that a major assumption is that the reflection of light follows a so called Lambert Reflection. This means that the light incident on a surface is redirected in all directions. This is great for rough surfaces such as walls of a room, however when it comes to glossy and reflective surfaces this assumption falls apart. A mirror for example is expected to only reflect light in a way so that the image emitted onto the mirror can be recaptured after being reflected. However radiosity bleeds and diffuses the light together from all angles, making it impossible to model a mirror-like reflection without additional implementations. In addition transparent surfaces are troublesome to model as the reflectivity of most transparent surfaces such as glass is very high, causing the light to go lost despite it propagating through the object. For example a render of a kitchen with lots of shiny surfaces such as a counter and reflective object like knives and trays will not have its features accurately portrayed. [HER90] While there are implementations of radiosity that make an attempt to accurately solve these problems, a better rendering method for this scenario would be ray tracing.

8 Ray Tracing

Ray tracing is a different GIA that primarily focuses on specular light rather than diffuse light. Ray tracing also only considers the light arriving at the viewer instead of the light arriving the surfaces of the scene. This is done by casting a ray through each pixel from the viewers perspective and letting this ray propagate through the scene, by either reflecting off object or propagation through objects. This gives the advantage of not having to deal with discrete sampling of an objects surface and also allowing the user to create images with realistic reflections and transparent objects. This proposes a solution for the already mentioned problem with radiosity. Ray tracing can also be used in conjunction with radiosity to minimize the artifacts that appear around edges of objects. [Lig]

Figure 7: Ray Tracing Render [Tra09]

Figure 7 is a perfect example of a render using ray tracing. The image features all kinds of complex lighting conditions of reflections, transparent and translucent and also glossy surfaces. While this scene would be a much greater challenge for a radiosity algorithm to render, ray tracing is very good at portraying this scene with incredible realism.

9 Limitations of Ray Tracing

Like radiosity, there are also some scenes that ray tracing does not preform as well, in particular scenes with a lot of diffuse light. This resulted in that so-called colour bleed cannot be modelled accurately. Colour bleed is when the colour of objects diffuses or spreads onto other surfaces in the scene. This can cause the shapes and objects in the scene to appear unusually sharp. In addition, while being computer intensive, ray tracing is also perspective dependent meaning that every time the viewing position and angle is altered the entire image needs to be rendered again. [Tob97]

10 Ray Tracing and Radiosity

Ray tracing and radiosity are quite fundamentally opposite in a way. While both are GIAs the way they render a scene is severely different. The fist difference between the two is the interpretation of light reflection.

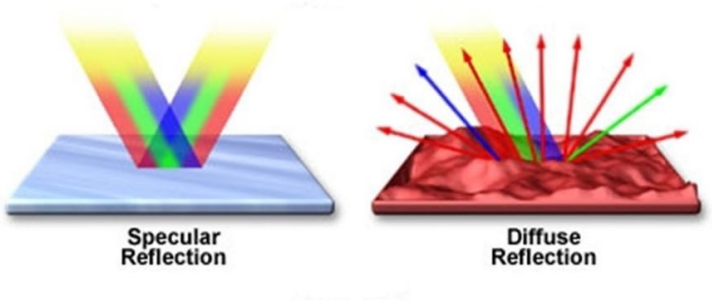


Figure 8: Diffuse and Specular light [Vir]

Figure 8 illustrates the key difference in light reflection between radiosity and ray tracing. The left hand side shows show specular light reflects while the right hand side shows diffuse light reflections. As previously written ray tracing uses specular reflection, the angle if incidence is assumed to be equal to the angle of reflection, which means that the image inbound to the surface will closely resemble the outbound (reflected) image. As shown in figure 8 this causes the surface to appear more reflective. Radiosity on the contrary uses diffuse reflections, which means that the incident light is scattered in all directions while also diffusing with the surfaces' light. This gives the surface colour while also propagating light everywhere else in the scene. In figure 8 is is to note that the right image reflects more red light than was incident on the surface. This results in the surface looking more matte and also causes in this case that the red light will diffuse and bleed to other surfaces in the scene.

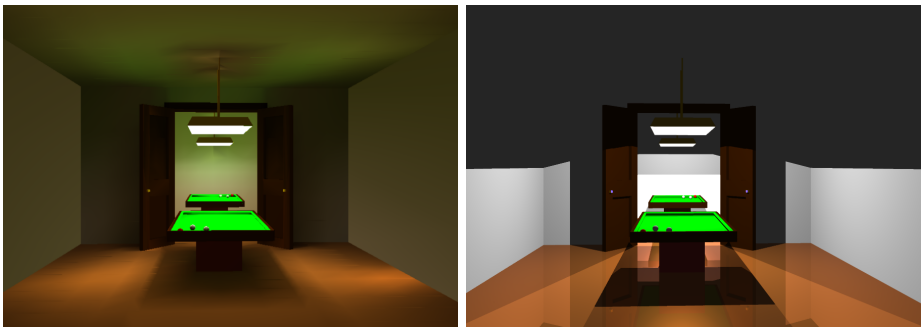


Figure 9: Same Scene with Different Renders [Tob97]

Figure 9 is another comparison of radiosity and ray tracing by placing the render of the same image side by side. The left render is the radiosity render and the right one is the ray tracing render. In the radiosity render it is easy to identify the characteristic colour bleeding of the green snookers table onto the ceiling causing the green tint. The shadow of the table is also very gradual and smooth, and the shadow cast by the door onto the back wall is barely visible at all.

The ray tracing render on the right side is noticeably different. The shadows are incredibly sharp and the light emitted from the ceiling light is chalk-white. There is absolutely no colour bleed of the snookers table and the colour green is not present anywhere else than the pool table. Lastly the most important feature seen is the reflection of the back door onto the floor. The floor looks very polished and reflective in the ray tracing render. Figure 9 should be enough for anyone to conclude the vast difference in appearance of ray tracing and radiosity. However deciding between the two can be a challenge as they deal with different environments with vastly differences. The example from figure 7 does present a very good ray tracing render. However a scene is not going to have the render method in mind when it is constructed, that would limit the whole purpose of creating realistic images when the user is restricted on which lights and surfaces are allowed in the scene to cater to the strengths of an algorithm. This is when combined rendering comes into play.

Figure 10: Combined rendering [Tob97]

Figure 10 has introduced a so-called combined render in the central image. This render uses a combined render of radiosity and ray tracing. This method takes advantage of the fact that the two algorithms are so opposite in their ideal environments. For example, the colour bleed on the ceiling which originates from the radiosity render which was not produced by the ray tracing can be incorporated with the reflective floor which originates from the ray tracing render. This creates a combined render which looks superior to either individual render. The combined method can also improve the artifacts after the adaptive meshing of the radiosity algorithm.

11 Conclusion

Radiosity is a rendering algorithm that divides a scene's surfaces into so-called patches and then letting light diffuse over all surfaces in the environment with a complexity of $O(n^3)$, where n is the number of patches. The algorithm is viewpoint independent and once a scene has been rendered it can be viewed from any angle in the scene without needing to re-render the scene. Radiosity excels at modelling environments with matte surfaces that diffuse a lot of light, while it struggles with reflective or transparent surfaces.
Ray tracing is another rendering algorithm that works by tracing the light from the viewer to a light source by reflecting a ray though the environment. In contrast to radiosity it is not viewpoint independent and needs to be rendered every time the perspective changes. Ray tracing cannot model diffuse light very well, but on the other hand is very effective at rendering transparent and reflective surfaces.
The two algorithms can be combined to make a rendering algorithm that plays at the strengths of both algorithms in order to make a multi functional rendering algorithm.

References

[AI13] Mette Hvass Michael Jørgensen Jens Christoffersen Werner Osterhaus Kjeld Johnsen Anne Iversen Nicolas Roy. *Daylight calculations in practice*. English. Version 1. DANISH BUILDING RESEARCH INSTITUTE. 2013. 1 p. 2013.

[CG85] Michael F. Cohen and Donald P. Greenberg. "The hemi-cube: a radiosity solution for complex environments". In: *SIGGRAPH*. 1985. DOI: 10.1145/325334.325171. URL: http://doi.acm.org/10.1145/1671970.1921702.

[HER90] Kenneth E. Torrance Holly E. Rushmeier. "Extending the Radiosity Method to Include Specularly Reflecting and Translucent Materials". In: *ACM transactions on graphics* 9 (1990).

[Lig] *Lightscape User's Guide*. English. Version 1.0. Autodesk, Inc. 1999. 1 p. April, 1999.

[MFC86] David S. Immel Philip J. Brock Michael F. Cohen Donald P. Greenberg. "An Efficient Radiosity Approach for Realistic Image Synthesi". In: *ComputerGraphics* (1986).

[MFC98] John R. Wallace Donald P. Greenberg Michael F. Cohen Shenchang Eric Chen. "A Progressive Refinement Approach to Fast Radiosity Image Generation". In: *ComputerGraphics* 22 (1998).

[RH90] Joanne Ng Rik Harris A. Marriott. *CG351-551 Radiosity: Concerning Form Factor*. 1990.

[Vir] *The Role Of Reflectivity In Glass Selection*. English. [Online; accessed 27, July, 2019]. Viracon, 2012. URL: https://www.viracon.com/pdf/TTReflectivity.pdf. 2012.

[Tob97] Robert F Tobler. *Radiosity - Ray Tracing*. 1997.

[Tra09] Gilles Tran. *Glasses, pitcher, ashtray and dice*. [Online; accessed 27, July, 2019]. 2009. URL: http://www.oyonale.com/images/3D/glasses.jpg.