A graph based process for easy design of refinement oracles in hierarchical radiosity

Jérémie Turbet (JeT) - François X. Sillion

iMAGIS – GRAVIR/IMAG INRIA
655 avenue de l'Europe (ZIRST)
38330 Montbonnot Saint Martin, France.
[Jeremie.Turbet|Francois.Sillion]@imag.fr

ABSTRACT

Refinement is the part of the hierarchical radiosity algorithm that decides the best subdivision of the scene geometry to meet user goals using minimum resources. The refinement oracle is a central component of the radiosity algorithm, because it affects the computation time and the radiosity computation error. Hierarchical radiosity refinement remains a research topic today because of the variety of the geometric and radiometric configurations encountered: currently there does not exist a universal oracle that works well in all the different scene geometries and lighting configurations. It is therefore highly desirable to develop flexible tools for the generation of appropriate oracles suited to different tasks. In this paper we propose a graph structure for the refinement process and a classification of the elementary problems the oracle can handle during the refinement. This representation clarifies the complex refinement process by reducing it to the composition of simple tools. New refiners can easily be created or modified with a marginal increase of the computation time, and many advantages in terms of automatic checking and performance analysis.

Keywords: radiosity, hierarchical refinement, lighting, global illumination, refinement oracle.

1 Introduction

The radiosity equations were introduced in the 1930s, but were applied to computer graphics only in the mid-1980s. The radiosity algorithm has quickly evolved: the first versions dealt only with very simple scene geometry, and required heavy computation times. With the creation of the progressive and then the hierarchical radiosity algorithms, those limitations have been seriously diminished. A few commercial packages nowadays actually use radiosity to produce realistic images. Hierarchical radiosity requires an oracle to decide at which level of the scene hierarchy the energy exchanges have to be established. The oracle takes each pair of hierarchical elements in the scene and decides wether the elements should be linked together to represent the energy exchange, or be subdivided. Despite the apparent simplicity of the decision, existing oracles only work correctly in limited configurations and are hardly manageable. This is probably the main reason of the slow development of radiosity-based tools. New, application-specific oracles are needed to give radiosity the popularity of other global illumination methods.

The generation of refinement criteria is complex due to the wide range of domains covered. Oracles have to deal with geometric, energetic and visibility issues, in a tightly inter-related manner. We propose a classification of all simple actions occurring during the refinement process into a set of elementary tools. Each tool is able to answer queries about its area of competence, and provides a unique answer from a specified set of possible ones. The refinement process can then be seen as a "discussion" between the refinement oracle and a set of such tools. We use a graph representation to describe the oracle: each node of the graph is an atomic task answered by a tool, and the last node is the answer of the oracle. This structure provides a number of advantages. Evolutivity and modularity are intrinsic, providing maximum flexibility and ease for tool manipulation and oracle generation. Furthermore, a number of operations can be applied generically to the graph, such as automatic consistency checks or statistics gathering. We validate the structure by re-creating an existing oracle using a graph and tools, comparing computation times and results, and discussing the treatment of visibility queries using different tools.

1.1 The radiosity algorithm

The radiosity algorithm computes an estimation of all energy exchanges in a scene, limited to diffuse reflectors. The wavelength used for the energy depends on the application. Radiosity was first used to compute thermal exchanges, especially for heat distribution [Hotte67]. The method has been extended to the visible part of the wavelength domain for light simulation [Goral84, Nishi85].

1.2 Mathematical definition

The radiosity equation describes the energy balance in a scene as an integral equation [Yamau26, Buckl27] :

$$B(x) = E(x) + \rho(x) \int_y B(y) F(x,y) V(x,y) \delta y \quad (1)$$

• E is the natural exitance at point x
• ρ is the diffuse reflectance at point x
• F the relative orientation and distance factor between x and y
• V the visibility between x and y

The first use of finite elements to propose a global illumination calculation method dates back to 1934 [Higbi34] but was not used due to the lack of computation resources at the time.

1.3 Hierarchical radiosity

The idea of the hierarchical radiosity is to let the computer manage the geometry depending on the radiosity function. [Hanra91] introduced a hierarchical definition of the surface geometry. This gives the algorithm the flexibility it needed to choose the right size of any surface element. The radiative exchanges are estimated between two surface elements, a criterion estimates wether the representation is adequate or not at this level of the surface hierarchy. If not, one or both elements have to be subdivided into a finer hierarchy. Energy exchanges are represented by links which transport the light from the emitter to the receiver.

The hierarchical radiosity reduces the modeling problem because there is no more need of taking into account the energy distribution in the scene. This formulation greatly limits the number of visibility factors to be computed, thereby decreasing the computation times. But the complexity is still quadratic in the number of input surfaces, which limits the use of the algorithm to small scenes. The notion of surface hierarchy has been extended to volumes (clusters) to achieve a O($nlog(n)$) complexity [Kok93, Silli94, Smits94], but the choice of grouping elements is not trivial and is still a research topic [Hasen99].

2 The issue of hierarchical refinement

Hierarchical radiosity allows to control the solution accuracy by the element subdivision choice. If the error of the estimated energy exchange on a hierarchical element (surface or cluster) is far from the desired representation (uniform for Haar wavelet basis functions, for instance), the refinement of the link (one or both elements) will reduce the error. Some current researches on radiosity focus on the heuristic of the refinement process, called refinement criterion or refinement oracle.

A first set of oracles assume that the error on a hierarchical element is proportional to the energy it receives. The most simple criterion in this oracle class is the ”BF criterion” [Hanra90, Hanra91]. It takes its decision of refining or not only with a threshold on the irradiance. The advantage of this oracle is its simplicity and so its rapidity, but it suffers of some important problems.
• The choice of which element should be split is not exactly defined.
• The threshold is absolute. Increasing the luminosity of the scene changes the refinement dramatically.
• The computation time and the solution accuracy do not depend simply on the threshold, despite simple asymptotic laws. This refiner is very hard to manipulate.
A second class of oracle consists of finding a bound of the radiosity error function on the elements [Smits92, Holzs98], those criteria are much more precise and gives better results. But there is a notable overhead of the computation time, and they often do not treat visibility very well. The third class estimates the impact of the refinement on the globality of the radiosity solution [Holzs94, Lisch94]. The time cost is however even greater.

2.1 Segmenting the problem

A refinement criterion is composed of different factors, of very different nature. Typically the subdivision should be finer along the shadow edges and where the radiosity function is quickly varying.

Those two factors are quite independent and can be treated separately. We identified the following factors as playing a key role in refinement :
• Visibility : Higher gradients of the solution are commonly found along shadow edges resulting from one or more occluders.
• Energy related : In a total visibility context, the energy distribution on a hierarchical element depends on the exitance distribution and the geometry relative to the two elements.

2.2 Refinement oracle

The refinement oracle is the heuristic which decides if the energy distribution on the receiver emitted from another hierarchical element is sufficiently well captured at the current hierarchical level. Depending on its answer, it establishes the link between the two elements, or splits the link to try to establish it at the directly lower level of the hierarchy. The oracle can use different refinement criteria according to the different configurations of the elements. In fact it can be seen as a selector of criteria, but the discrimination of such configurations is not trivial. Applications can have very different needs in global illumination: In the lighting engineering field, users will want to guarantee a minimal energy error in the scene, but probably do not care much for beautiful shadow boundaries. Conversely, creators of virtual reality environments may prefer beautiful shadows but not care about the reliability of the solution. The different criteria used and the order they are employed in an oracle define how the refiner will work, and can be guided to match the application needs.

To take into account all the possible oracles, we choose to represent the refinement process as a set of atomic (simple) questions and actions (each implemented in a *tool*), combined to obtain a complete decision process. Each tool will answer a question or perform a well-defined operation. The refinement process can then be seen as a discussion between the oracle and the set of tools. At the end of the discussion, the oracle will take its decision (Fig. 1). The possibles decisions are :
• Establish the link at this level.
• Split the link and reiterate its process with the elements children.

The user needs to influence the refinement in the direction he wants, he can do it using parameters plugged into the tools. Those parameters are the communication media through the refinement process. We differentiate between two different sort of tools, those that answer a question (question tool), and those that are performing a task (command tool).

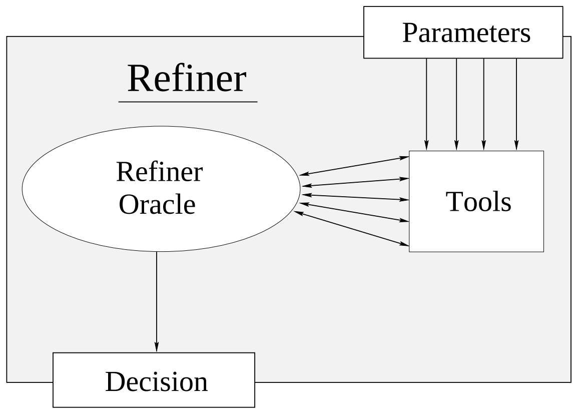

Figure 1: Refinement process structure as a discussion between tools and the oracle.

3 Tools

The tools are the minimal bricks essential to the construction of an oracle, they all have the same general structure so that they can be combined in a sort of “lego” construction.

3.1 Question tools

Question tools are those that answer a question, they constitute the body of the discussion, and guide the oracle into its decision. Three different types of answers can be distinguished :
• Precise answers : those answers are guaranteed to be true. For example, for a visibility classification tool, a precise answer is ”the visibility is null between the two elements”.
• Approximate answers : these are used when a precise answer can not be given. The answer is not guaranteed, it indicates a probability, such as ”the visibility is probably null between the too elements”.
• other answers : these do not answer the question asked, but rather indicate another form of information from the tool, which can result from a failure of the algorithm, or the realization that computing an answer would be too costly. A tool can use such answers to say ”I do not know what the visibility class is, but the situation is so complex that you should probably subdivide rather than attempt to get an answer”. This type of answer is very useful since it lets us model complex situations without forcing us to answer all questions exactly as they were asked.

We distinguish three different classes of questions, all of them are allowed to answer an answer taken from a set. The set of possible answers, precise and/or approximative, is defined once for each question tool.

3.1.1 Energy-related tools

Energy-related tools answer questions about radiative exchanges along the link during the refinement. For example, they can determine if there is enough energy on the link, or if the incoming energy will change significantly the receiver radiosity, etc.

3.1.2 Visibility tools

Visibility tools classify the visibility between the emitter and the receiver. This task is very specific, but it is typically the hardest and the most time consuming one of the refinement process. There is no limits on the number of answers, each tool has its own set. But 6 classes are frequently used, the three precise ones and their corresponding approximative answers.

- Visible : There is no occluder between the emitter and the receiver: no segments from one element to the other is occluded. The visibility factor is 1.
- Invisible : One or more occluder completely occlude the emitter from the receiver. All segments from one element to the other are blocked. The visibility factor is 0.
- Partial : There is one or more occluders between the emitter and the receiver, disposed such that at least one segment from one element to the other is blocked, while another one is not. The visibility factor lies between 0 and 1.

3.1.3 Geometric tools

Geometric tools and energy related tools are sometimes overlapping, because the energy distribution depends on the geometry. But geometric tools can be distinguished by their absence of notion of radiative flux. The treated problems include self intersecting surfaces, normal cones, etc. They are very useful to guarantee the proper functioning of other tools which require a specific geometric disposition. For example most of form factor computation algorithms cannot handle intersecting support plan surfaces (Fig. 2), the use of a geometric tool guarantee a valid answer.

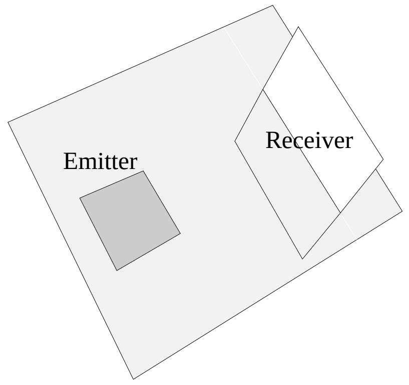

Figure 2: Example of a geometric tool which determine self intersection support plans of surfaces.

3.2 Command tools

Command tools are particular tools executing an action (an order). Their answers are only ”ok I have done it” or for some tools, ”No I cannot do it”. They can be used to compute a value using a special algorithm, mark an elements, etc. An example is the computation of the visibility factor (in case of partial classification). These tools are very useful to debug a refiner during its creation. The action the refiner has to do after the oracle gives its answer is a command tool. Those are called decision tools because they only appear at the end of the discussion when the oracle takes its decision.

The set of decision tools includes all possible actions after the oracle gives its answer. They only take place at the end of the refinement process for a pair of elements. Those tools are the only ones which do not have any answer, they are the conclusion of the discussion.

- Link establishment : when the oracle estimates that the radiosity function is sufficiently well represented, this tool establishes a link between the receiver and the emitter to symbolize the energy transfer.
- Subdivide : when the energy distribution seems not correct at this level of hierarchy, the refiner goes deeper in the hierarchy to try to link at lower levels. There is a choice for the subdivision to split the emitter, the receiver or both. This decision is a part of the oracle. The two subdivision tools just subdivide the emitter and subdivide the receiver.
- Stop : if the energy transfer doesn’t change anything to the solution, there is no need to link or to subdivide an element, because it will not change anything more. So we just stop the refinement process. For example, it occurs when the visibility is null.

4 Oracle graph structure

The oracle follows its procedure by asking questions to the tools, and taking appropriate action depending on the answers: it can either ask another question or perform an action. From question to question, it can finally express what it considers the best decision. We represent the set of possibles discussions by a graph structure. Each node contains a question tool or a command tool, each arc an answer of the tool-node and each leaf contains a decision tool (Fig. 3). We call these graph oracles ”decision graphs”. The node encapsulation of the tool is necessary for the graph structuration, moreover it allows many advantages (cf. section 4.3). There is a single tool for a node, and its answers are mapped by the node to other nodes.

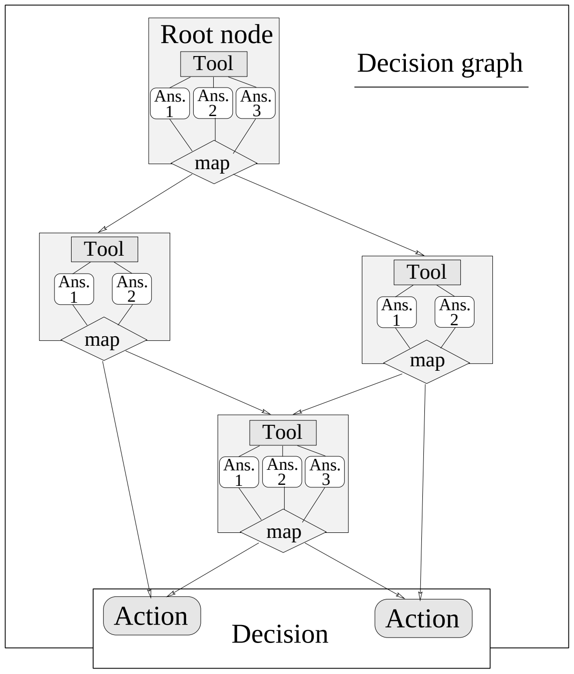

Figure 3: Tree structure of the decision graph.

The graph has to be acyclic, because this could result into a infinite refinement process, therefore it is a DAG (Directed Acyclic Graph).

4.1 Data manager

All along the discussion, the different tools often use common data. For example the principal axis between the receiver and the emitter is used by many geometric and visibility tools. But the modularity and the independence of the tools prohibits direct discussion between them, and so, the exchange of data. We have to use another actor in the discussion which is the ”memory” of what has been said, this is only a data manager. Each tool has to ask the data manager about the existence of the terms he needs to use in its execution. If a term has not already been computed, the tool, by its own, computes the value and tells the data manager it has done the work. This data manager stores all the values which are likely to be useful to the others.

A precision level is assigned to each value in the data manager. In this way, if a tool needs a precise term and the value has already been computed using a coarse estimation, it can recompute the precise term, and submit it back to the data manager. For example, the visibility term can be estimated during the visibility classification step but it can be insufficient to the linking tool which will recompute a more precise value of the visibility factor.

4.2 Parametrisation

Each tool can use some parameters, often threshold are used to influence their answer. Those parameters can be computed in two different ways :
• Fixed parameter : the value of the parameter is evaluated once at the beginning of the refinement process. The value can be fixed by the creator of the refiner (it is not noticed by the user). Another way is to give the control of the value to the user through a graphical user interface.
• Computed parameter : the parameters can be computed at each execution of the tool by a callback defined by the creator of the refiner. It allows a modification of the tools behavior all along the refinement process. The computation of the parameter values may need some values which are only accessible during the execution of the graph (through the data manager). Computed parameters are able to modify locally and automatically the refinement to the scene specificities. The parametrisation of all the tools is the parametrisation of the oracle, it gives the user great flexibility to control the refinement, even for a fixed graph layout.

4.3 Advantages of the DAG structure

The graph structure and the power of inheritance in object programming have many intrinsic advantages. The main ones are :
• Modularity : all the tools follow the same design, so they can be easily exchanged, added, replaced, deleted from the DAG. They all are independent of the context and only discuss with the data manager (which can be empty).
• Evolutivity : because the design scheme is the same for each tool and the number of requirements is limited to the minimum (the current treated link and the data manager), new features can be added with few limitations.
• Execution track : during the refinement process, each node can print its execution showing the branch in the DAG taken by a specific link refinement and the values it is composed or it returns

• Debugging : by defining manually an emitter and a receiver, and launching the execution track on them, we can determine very quickly what happens in the refinement process between two elements. Time consumption, answers of the tools, branch in the graph, ... This is very useful for debugging a refiner and understanding how it works.
• Graph validation : after the generation of the decision graph, it is possible to automatically test the validity of the graph : No cycle, no empty nodes, all tool answers are linked, etc.
• Visualization : the graph structure can be visualized with a drawing which is natural for the human eye. This representation is much more understandable than a source code.
• Interface generation : each tool owns its interface with the parameters manipulators. The collection of all tools interface can be packed together for an automatic graph interface generation.
• Branches reuse : it is easy to re-use branches of other refinement graphs without code duplication or complex flow control.
• Statistics : statistics are easily generated on the nodes of the graph, time computation, number of calls, unused branches, etc. Bottlenecks can be easily determined using this characteristic. Millions of links are refined during a refinement process, statistics generation is an important feature to understand this very complex task.

4.4 Computation times

In this section, we measure the overhead of time induced by the graph structure. We expect the refinement process using our DAG structure to be slightly more time consuming because at each node execution, the node has to research into its answer-mapping table the next node to execute. It is possible, once the refiner graph is designed, to create an iterative release of the refiner. This is done by a code generator which creates a new refiner source file. The main procedure contains all the branches of the graph using many switching instructions. The resulting refiner do not contains nodes and arcs anymore, only the tools and their relations are present.

We tested three similar refiners based on the well known BF criterion:
• BF : this is the standard refiner expressed in a single function.
• DAG release : this is the BF developed using the DAG refiner structure, it is composed of three main tools and six command tools.
• Flat release : this is the refiner self generated by the DAG release, it changes the DAG into a single function.

We tested these refiners on the following set of scenes:
• Room : Small scene, inside of a room with very few visibility problems.
• TD : Small scene with heterogeneous size of objects, creating some visibility problems.
• Maze : Labyrinth scene with an important visibility factor.
• Soda : Complex inside of a bar scene, with a medium visibility factor.
• Séjour : Complex scene with heterogeneous size of objects.

scene	BF release	DAG release	flat release
Room	0.70 s.	1.07 s. *+53%*	0.80 s. *+14%*
TD	2.30 s.	3.40 s. *+48%*	2.47 s. *+7%*
Maze	24.47 s.	43.50 s. *+78%*	26.47 s. *+8%*
Soda	83.70 s.	93.80 s. *+12%*	84.27 s. *+1%*
Séjour	164.23	183.90 s. *+12%*	167.49 s. *+2%*

We observe that the time overhead is inversely correlated to the scene complexity. The difference decreases progressively to 10% for the DAG release and almost zero for the flat release as the complexity of the scene increases. But for simple scenes the overhead induced by the structure is not negligible compared to the ”only-refinement” time. Since our goal is to work with realistic and very complex scenes, the refinement DAG structure seems a valid choice.

5 Example of the graph usage

There are many applications of the graph structure due to the appreciable number of advantages. We present in this section an example of the use of the DAG statistic generation advantage applied to a visibility study. The visibility study we chose deals with tools based on rays. We want to determine if there is a correlation between the size of the elements, the number of rays and the reliability of the tool's answer. We plugged into an existing refiner, a branch of tools that will execute our tests. It is composed of a geometric tool that computes the projected area of the elements along their center-center direction, a reference tool that will give us a reference visibility classification, and all the tools we want to test. Those tools are, in our case, only visibility tools based on rays with different sampling rates. We choose 3 sampling rates: 4, 16 and 64 rays per link to classify the visibility. The geometric tool is linked to the test branch if the emitter and the receiver projected areas are under a specified threshold, if not, nothing more than the ”normal” refinement is performed (Fig. 4).

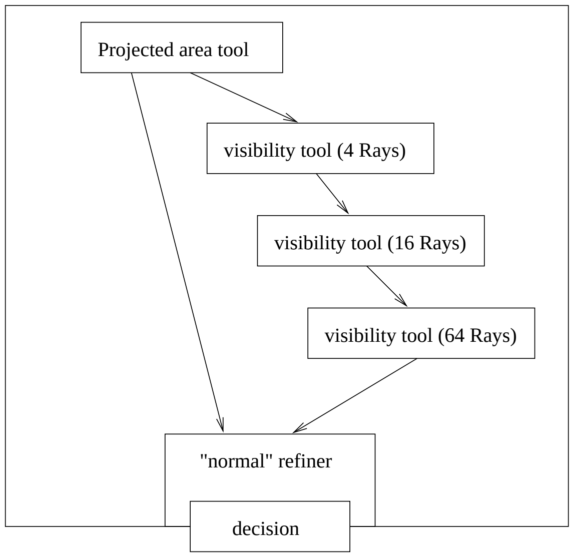

Figure 4: Graph scheme of the visibility tester refiner.

We used several values of threshold for the geometric tool to understand the behavior of each visibility tool in front of a specific geometric situation. By setting the generation of statistics in the refinement scheme, we collect after each refinement process in the all set of statistics, the number of each answer of the tools. We can compare the number of reference answer with the number of equivalent tool answer to compute a ”reliability” factor for each tool. The results are shown in figure 5.

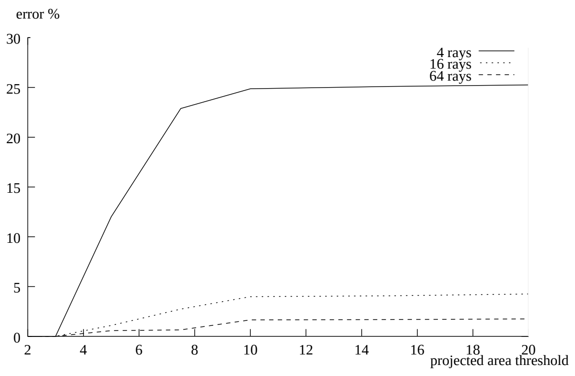

Figure 5: Error functions of the ray casting based visibility tools.

The reliability has been computed by summing all the erroneous classifications of each tools divided by the number of answers given (equal to the number of links refined). Values are presented in table 5. The threshold is expressed in area units, the minimum units in refinement process is 0,5 area units. The test scene is the ”TD” one previously discussed. Two iterations have been used to compute the values.

ϵ	# links	4 rays	16 rays	64 rays
2	*2266*	0 %	0 %	0 %
3	*3912*	0 %	0 %	0 %
5	*8040*	12,01 %	1,09 %	0,57 %
7,5	*14725*	22,89 %	2,74 %	0,65 %
10	*19025*	24,87 %	3,98 %	1,65 %
15	*19808*	25,02 %	4,07 %	1,68 %

The result is as expected: the reliability of the ray casting based answer tools depends on the size of the elements and the number of rays used. It gives a valuation of this reliability to guarantee percentage of error. The important part of this study do not focus on the visibility results, but on the usability and the contribution of the DAG structure: statistics are easily gathered, and the graph can then be easily modified to, for instance, use a different visibility tool based on the outcome of a geometric tool, with parameters extracted from the analysis of such experiments.

If this structure were not used, a refiner would have been ”hacked” to integrate the ray casting tools in it, the ray casting function should have been instrumented to store all their answers, values should have been packed together from various functions (rays, refiner) to compute statistics... In our method, nodes have been added without interfering with the ”normal” refiner, and statistics are given in a well formed shape at the end of the refinement.

6 Conclusions

The refinement oracle in hierarchical radiosity remains one of the difficult problems with this technique. Despite the vast body of research on this topic, there is not a valid answer working for all the different cases we can encounter in the radiosity process. Some criteria works well for a specific class of geometry, but are unusable in the next class. All the existing oracles are using their own structure specialized in the work they were created for. They all have common parts but it is impossible to reuse them because of their structure differences. In order to make radiosity usable for real-world applications, specific oracles must be designed to cope with all possible situations, and arranged appropriately depending on each application’s requirements. Our idea is to create a common open structure for all refiners. To achieve this goal we choose to design the refiners as a graph composed of elementary tools. Each tool is completely independent of the others and, answers a question or executes an action. The modularity of the DAG structure provides many advantages in the creation of new refiners and allows to reuse branches of the graph.

REFERENCES

[Buckl27] H. Buckley. On the radiation from the inside of a circular cylinder. *Philosophical Magazine Series 7*, 4(23):753–762, October 1927. one of the first papers on Fredholm integrals in radiative transfer theory.

[Goral84] Cindy M. Goral, Kenneth K. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the interaction of light between diffuse surfaces. *Computer Graphics*, 18(3):213–222, July 1984.

[Hanra90] Pat Hanrahan and David Salzman. A rapid hierarchical radiosity algorithm for unoccluded environments. In *Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, pages 151–71, Rennes, France, June 1990.

[Hanra91] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, July 1991.

[Hasen99] Jean-Marc Hasenfratz, Cyrille Damez, Francois Sillion, and George Drettakis. A practical analysis of clustering strategies for hierarchical radiosity. In *Computer Graphics Forum (Proc. Eurographics '99)*, volume 18, September 1999. To appear.

[Higbi34] H. H. Higbie. *Lighting Calculations*. John Wiley & Sons, New York, NY, 1934.

[Holzs94] Nicolas Holzschuch, Francois Sillion, and George Drettakis. An Efficient Progressive Refinement Strategy for Hierarchical Radiosity. In *Fifth Eurographics Workshop on Rendering*, pages 343–357, Darmstadt, Germany, June 1994.

[Holzs98] Nicholas Holzschuch and Francois. X. Sillion. An exhaustive error-bounding algorithm for hierarchical radiosity. *Computer Graphics Forum*, 17(4):197–218, December 1998.

[Hotte67] Hoyt C. Hottel and Adel F. Sarofim. *Radiative Transfer*. McGraw Hill, New York, NY, 1967.

[Kok93] Arjan J. F. Kok, Frederik W. Jansen, and C. Woodward. Efficient, Complete Radiosity Ray Tracing Using a Shadow-Coherence Method. *The Visual Computer*, 10(1):19–33, 1993.

[Lisch94] Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and error estimates for radiosity. *Computer Graphics*, 28(Annual Conference Series):67–74, July 1994.

[Nishi85] T. Nishita, I. Okamura, and E. Nakamae. Shading models for point and linear sources. *ACM Transactions on Graphics*, 4(2):124–146, April 1985.

[Silli94] Francois Sillion. Clustering and Volume Scattering for Hierarchical Radiosity Calculations. In *Fifth Eurographics Workshop on Rendering*, pages 105–117, Darmstadt, Germany, June 1994.

[Smits92] Brian E. Smits, James R. Arvo, and David H. Salesin. An importance-driven radiosity algorithm. *Computer Graphics*, 26(2):273–282, July 1992.

[Smits94] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 435–442. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[Yamau26] Zito Yamauti. The Light Flux Distribution of a System of Interreflecting Surfaces. *Journal of the Optical Society of America*, 13:561–571, November 1926.