SOME APPLICATION ARCHITECTURES FOR REALISTIC IMAGE SYNTHESIS IN A DISTRIBUTED ENVIRONMENT

Marius Dorian Zaharia
POLITEHNICA University Bucharest
str. Splaiul Independentei, nr. 313, 77206 Bucharest
E-mail: zaharia@cs.pub.ro

Abstract*. The classical methods used to realistically render 3D scenes are "ray tracing" and "radiosity". These methods are known for the huge amount of processor time they require in order to obtain high resolution, true colour images. The paper describes three application architectures suitable to be used to accelerate the image synthesis process. This study will be used to build an adequate application architecture for a distributed realistic image synthesis system that will be developed at the Computer Science Department of POLITEHNICA University Bucharest.*
Keywords*: image synthesis, global illumination algorithms, ray tracing, radiosity, parallel algorithms, distributed computing.*

Introduction

The realistic image synthesis is a computing process which, starting from: a model of a scene containing 3D objects, the specification of the geometry and spatial positions of some light sources and of the viewing point, produces the scene image. The development of realistic rendering algorithms requires knowledge from various fields of computer science (computer graphics, computational geometry, computer vision, signal processing) as well as physics (optics, photometry, statistical physics) or mathematics (numerical methods).
A class of realistic rendering algorithms is known as "global illumination algorithms". They are taking into consideration the distribution of the lighting energy in the whole scene, at a given moment. The algorithms in that category determine the colour of an arbitrary point (P) from the surface of a given scene object, taking into account the radiation that reaches (P) directly from the light sources and the light that reaches (P) indirectly, through repeated reflections or transmissions across the surfaces of the other objects of the scene.
Other techniques heavily used for realistic scene rendering are texture mapping, atmospheric attenuation, volumetric modelling, fractal modelling.
The domain of the global illumination algorithms includes two major rendering methods: ray tracing and radiosity. Both methods try to solve the radiance equation ([10]) that is a Fredholm integral equation of the second kind.
The radiance equation describes quantitatively the energy transfer from a point (P) placed on a surface of a scene object to another point (P'):

$$I(P,P') = g(P,P')(e(P,P') + \int_S r(P,P',P'') * t(P,P',P'')dP'')$$

where:

- I(P,P') is the intensity of light radiation transmitted from P to P'
- g(P,P') is a factor depending on the scene geometry, given by:

$$g(P,P') = \begin{cases} 0, & if\_P,P'\_are\_mutually\_invisible \\ \dfrac{1}{dist(P,P')}, & if\_P\_is\_visible\_from\_P' \end{cases}$$

- e(P,P') is the intensity of the light radiation emitted from P' to P
- the integral is defined over all the surfaces S of the scene objects
- r(P,P',P") is the intensity of the specularly or diffusely reflected radiation from P" to P and passing through P'

The radiance equation is deduced from the general light transport equation eliminating the terms for polarisation, phosphorescence and fluorescence.
These approximations ensure that:

- the solution of the radiance equation at a given wavelength λ is independent of the solution at other arbitrary wavelength λ'. So,

in case of a colour image, the radiance equation could be solved several times at different wavelengths and the final solution is found combining the previous results.
- the radiance equation is time invariant
- the scene objects are placed in vacuum

The radiosity method builds explicitly a function that approximates the radiation distribution. This is performed by solving the above equation using numerical methods. The ray-tracing method computes samples of the solution function. One variant of the method computes ray trajectories from the light sources to the viewing point (so called photon-tracing method). Another variant treats separately each ray starting from the viewing point and then passing through an arbitrary pixel of the scene image until the ray intersects the surface of a most closely placed object. At each such primary intersection, the colour of the corresponding pixel is computed using a local illumination model and the algorithm is recursively applied for the reflected and respectively the transmitted ray ([5]).
Both methods make different assumptions about the interaction between the light radiation and the surfaces of the objects. Therefore, the radiosity method considers the surfaces of the scene objects as being opaque and perfectly diffuse reflectors. The ray tracing method considers that the scene objects are illuminated by radiation coming directly from the light sources or indirectly from other intermediate surfaces that reflect or transmit the light specularly.

Methods to accelerate the global illumination algorithms

The both previously mentioned methods are requiring large amounts of computer resources, especially computing time. That is why one of the associated research directions consists in finding techniques to accelerate the realistic rendering process. The most known examples are:

- The technique of the bounding volumes ([13]) that accelerates the process of finding the intersections between rays and object surfaces. This method has variants based on building hierarchies of bounding volumes.
- Methods based on spatial decomposition techniques. For example the space is partitioned in parallelepiped cells that have associated the lists of objects they intersect. A first approach of this type of rendering acceleration was given by Glassner ([6]) but a lot of variants exist ([2]). These methods are particularly adequate for scenes containing many objects.
- Hierarchy based techniques are used for adaptive sub-structuring of the surface patches that model the boundaries of the scene objects, in order to increase the efficiency of the radiosity algorithm ([5]).
- Methods that find the correlation between adjacent pixels. This type of methods consider that these pixels have similar intensities as well as similar histories of the (ray-object) intersections on the path from the viewing point to each of the adjacent image pixels ([1]).
- In the case of 3D scenes that include surfaces modelled through polygonal meshes, a popular acceleration method is the Level of Detail (LOD) rendering. It uses for modelling a larger number of polygons for the surfaces placed near the viewing point ([12]).

Some parallel/distributed realistic image synthesis applications

An important method to solve the huge resource request of the realistic image synthesis applications consists in designing parallel or distributed algorithms that should benefit from the intrinsic parallelism of ray tracing and radiosity.
Many research groups have developed hardware architectures particularly suitable to accelerate image synthesis. Representative examples are the WARP architecture (developed at Carnegie

Melon University) and the Pixel-Plane architecture of H. Fuchs ([4]).
In the absence of such dedicated architectures, the objective of this paper is to study, select and present the most adequate type of application architectures suitable to be used to develop a distributed realistic image synthesis application using “loosely coupled” hosts (general purpose processors) connected in a local or wide area network.
The available communication mechanisms are sockets and the distribution of the tasks could be performed using RPC mechanisms.

A spatial decomposition technique, suitable to be used by a ray tracing application running in a distributed environment was proposed by Dippé and Swensen [3]. They divide the object space in non-intersecting adjacent cells and assign each spatial region (or more such regions) to one processing element (PE). The spatial regions could have parallelepiped or tetrahedral shapes.
One central processing element determines the load factor of each host and eventually modifies the spatial distribution between processors in order to balance the load factors of the processing elements. Processor load factors could be computed as the product between the number of the objects (or a measure of the objects “complexity”) in the corresponding spatial region and the number of rays entering the region in a given amount of time.
The rendering process begins when the spatial region that contains the viewing point fires rays at a specified resolution. Each ray has an associated pixel (home pixel) that will be coloured at the end of the ray tracing process. When a ray enters a spatial sub-region (SSR), the associated PE computes the intersections with the objects contained in the spatial region. When the ray exits (SSR) it will eventually enter one region adjacent to (SSR) and the corresponding PE will receive an adequate message.
Some global functions of the system as: the initial distribution of the symbolic description of the scene, the initial build of the processor map and the global monitoring of the load factors of the processors could be accomplished by separate processes.
If the scene is too complicate to be copied in the local memory of each computer that participates at the synthesis application, an increase of the communication overhead is to be expected.
In a preliminary phase of the application development, the split of the object space could be done statically. In a subsequent phase, a separate process should dynamically and periodically modify spatial regions (changing the position of some vertices of the regions boundaries) in order to balance the corresponding load factors.
The communication between different PE-s could be done by messages that contain:

- rendering information (having geometric or physical nature) as: the ray direction, the co-ordinates of the intersection point where ray enters the spatial cell, the associated pixel co-ordinates, and the ray colour.
- control information as: the “begin rendering” message sent to the process that manages the spatial region containing the viewing point, load factors values, information describing the topology of the processing elements mesh, write pixel value in frame buffer.

The implementation could use independent computers. Each computer shelters one or more “virtual processing elements” (VPE) (each one is a separate process running under the local operating system) each of them communicates through messages with a set of “neighbour” VPE-s residing on the same host or on a different one. One VPE manages only one spatial region. The processors from the boundary of the “processor mesh” have associated unbounded regions. Only these processors should be able to modify the common “frame-buffer”. This buffer could be implemented using a file or a shared memory segment that must be accessed mutually exclusive. The final gathering of the partial rendering results could be realised by a separate process.
In the case of the radiosity method, the activities that could be spatially decomposed for parallel processing are the meshing process (the progressive refinement of distinct areas of the

global polygonal mesh), the form factors computations and the final polygonal shading ([9]).

Another interesting distributed application architecture (MISTRAL) was developed at University of Leeds to accomplish real-time visualisation of 3D scenes represented by CSG models. The MISTRAL system uses a computation model named by its authors “processor farm”. A computation model is an abstraction level placed between the application and the underlying hardware. The same machine could support more associated computation models. These are representing in fact patterns for using that machine in an application.
The “processor farm model” consists in three independent modules:

- A queue that include the processes waiting to run (the actually running processes do not communicate each other);
- A pool of servers which are able to run any process from the queue;
- A scheduler which assigns processes to servers. That algorithm could be distributed or centralised.

In order to emphasise the parallelism of the rendering algorithms, the image space will be structured as a set of rectangular regions with vertices placed on a square grid. The regions should be separately treated, by processes named “rendering servers”, that do not intercommunicate. Therefore, this kind of problem decomposition respects the processor farm model requirements. Each server contains one copy of the rendering software and one copy of the scene description that will allow the process to build any sub-region of the image. The generated pixels are routed to a final visualisation server. A separate process could compile the scene description language, assign regions to servers, generate the interconnection tables (process – image sub-region – host) and broadcast the control information and the 3D scene description to the rendering servers.

Another representative distributed application architecture was described by Liu and Chen ([11]). It is the DISCOVER system (Distributed Interactive Scientific Computing and Visualisation Environment) developed at National Chang Kung University Taiwan. This application emphasises the manipulation and visualisation of medical images and facilitates co-operative analysis of medical images by some groups of specialists. It uses a client-server architecture whose elements could fit well the control requirements of a distributed image synthesis application.
First of all, the model of the underlying hardware architecture consists of a virtual host and a processor pool. Theoretically, the virtual host could be formed from an unlimited number of PC’s (user interface processors) and the processor pool is made up of workstations which should be able to provide computational power to the virtual host.
From the software point of view:

- the server side includes the servers themselves (that are running on the processor pool) and the communication module which manages message passing channels between the computer servers and the user interface hosts
- the client side has three types of processes: session manager, system queue manager and group manager. Each PC from the virtual host could shelter several windows and each window is a client of one or more workstation servers.

The communication module (similar to the Parallel Virtual Machine) hides the heterogeneity of the underlying operating systems and assures the interaction between the distributed processes.
The session manager provides a uniform communication program interface; it has the same role as the communication module in the case of workstations.
The system queue manager has the following responsibilities:

- allocates computer server processes for user-interface hosts. (For that purpose it uses a simple load-balancing algorithm based on Huffman trees)
- assigns processes to the servers. The scheduling algorithm could be distributed or centralised.

Conclusions

The two above mentioned application models (processor farm and DISCOVERY model) have many similarities and could constitute the basis of new distributed architectures that should ensure superior performances relatively to single control flow applications.
However, the respected running times to render a given scene using such distributed applications will be far behind the corresponding rendering times obtained in case of applications running on dedicated hardware. This loss of efficiency is due to the considerable communication overhead in case of a distributed application running over a collection of computers connected through local or wide area networks.
Better performances could be realised by optimal assignment of tasks to processing elements and by carefully splitting the global activity into smaller tasks. The paper described two possible modalities to assign the image synthesis activities to processors. The first one ([3]) is based on a spatial decomposition of the object space and the second one considers the splitting of the image space. These two types of spatial decomposition (and the corresponding task assignments) could be combined, eventually in a hierarchical manner where the image space decomposition should be placed at a higher level in the hierarchy.

References

[1] Akimoto, T., Mase, K. and Suenaga, Y. (1991) Pixel-Selected Ray Tracing, *IEEE Computer Graphics & Applications*, pp. 14-22, July
[2] Arvo, J. (1987) Fast Ray Tracing by Ray Classification, *Computer Graphics (Proc. Siggraph)*, vol. 21, No. 4, pp. 55-64, July
[3] Dippe, M. and Swensen, J. (1984) An Adaptive Subdivision Algorithm for Realistic Image Synthesis, *Computer Graphics*, July
[4] Dew, P. (1989) *Parallel Processing for Computer Vision and Display*, Addison Wesley Publishing Co.
[5] Foley, J., van Damm, A., Feiner, S. K. and Hughes J. F. (1992) *Computer Graphics Principles and Practice*, Addison Wesley Publishing Company, Inc.
[6] Glassner, A.S. (1984) Space Subdivision for Fast Ray Tracing, *IEEE Computer Graphics and Applications*, vol. 10, No. 4, pp. 15-22, April
[7] Glassner, A.S. (1995) *Principles of Digital Image Synthesis*, vol. 1-2, Morgan Kaufmann Publishers Inc., San Francisco CA
[8] Gu, X., Gortler, S. J., Hoppe, H., McMillan, L., Brown, B. J. and Stone, A.D. (1999) Silhouette mapping. *Computer Science Technical Report TR-1-99*, Harvard University, March
[9] Hanrahan, P., Salzman, D. and Aupperle, L. A. (1991) Rapid Hierarchical Radiosity Algorithm, *SIGGRAPH ’91 Conference Proceedings*, vol. 25, No. 4, pp. 197-206, Addison Wesley Publishing Co., July
[10] Kajiya, J. T. (1986) The Rendering Equation, *SIGGRAPH ’86*, pp. 143-150
[11] Liu, P., Chen, L. S., Chen, S. C., Chen, J. P., Liu, F. Y. and Hwang, S. S. (1996) Distributed Computing: New Power for Scientific Visualization, *IEEE Computer Graphics and Applications*, May
[12] Puppo, E. and Scopigno, R. (1997) Simplification, LOD and Multiresolution, Principles and applications. *EUROGRAPHICS ’97*, pp. 16-72
[13] Whitted, P., (1980) An Improved Model for Shaded Display, *Communications of the ACM*, vol. 23, No. 8, pp. 343-349, June
[14] Zaharia, M. (1996) Algoritmi paraleli/ distribuiti în domeniul graficii de sintezã si al prelucrãrilor de imagini, Raport de Cercetare in cadrul proiectului TEMPUS S-JEP 07101/94, apãrut în: *Aspecte ale elaborãrii algoritmilor distribuiti*, pag. 370-493, Univ. POLITEHNICA Bucuresti, Politecnico di Torino
[15] Zaharia, M. (1997) Organization of Geometric Information for Parallel Processing, *Workshop on Parallel and Distributed Systems*, Bucharest, Commission of the European Communities, Task Force Human Resources, Education, Training and Youth, Structural Joint European Project S_JEP-07101-94 (DISCO), 26-28 May