A Formal Framework Approach for Ray-Scene Intersection Test Improvement

J. Revelles
Dpt. LSI
E.T.S.I. Informática
University of Granada
18071 Granada, Spain
jrevelle@ugr.es

M. Lastra
Dpt. LSI
E.T.S.I. Informática
University of Granada
18071 Granada, Spain
mlastral@ugr.es

R. Montes
Dpt. LSI
E.T.S.I. Informática
University of Granada
18071 Granada, Spain
rosana@ugr.es

P. Cano
Dpt. LSI
E.T.S.I. Informática
University of Granada
18071 Granada, Spain
pcano@ugr.es

ABSTRACT

The ray-scene intersection test is the most costly process when a scene is rendered. This process may be improved using any strategy to be able to speed-up it. Generally, any strategy used is based on the building of a spatial indexing in the scene domain or in the rays domain. However, there is no theory to formalize these techniques. In this paper, a formal framework approach for this technique is proposed.

Keywords

Graphics object theory, spatial indexing, ray casting, and acceleration techniques.

1. INTRODUCTION

Usually the programs based on ray tracing include acceleration techniques in order to improve the ray-scene intersection test. In theses terms, a lot of effort has been employed in the development of efficient solutions to the problem of ray-scene intersection test. However, there is no theory that formalizes the behavior of a generic optimizer and the techniques used in order to improve the intersection test process. In general, these techniques are based on some kind of spatial indexing.

In this paper we propose an abstract model of a generic optimizer as a function which selects a set of candidate objects for a given scene and a given ray. We also propose a model of optimizer based on spatial indexing. All of these concepts will be presented in order to establish theoretical premises to compare the efficiency of ray-scene intersection test when a spatial indexing technique is used.

WSCG POSTERS Proceedings
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

2. CONCEPTS, DEFINITIONS AND NOTATIONS

We focus our framework on optimizers that improve the ray-object intersection test using a spatial indexing scheme.

This scheme is a representation of the spatial distribution of scene objects. It is composed of a set of voxels, which will be called here *volumetric objects*. Each volumetric object may contain information of a subset of scene objects, which are contained in it. The ray-object intersection test performs the test with each volumetric object and returns a list of scene objects.

To formalize the optimizer behavior, we use the graphic objects theory [Tor92, Tor93] and a previous work about formalization of acceleration techniques [Rev01].

Graphic Object Theory

A graphic object o is a pair (μ,α), in which: μ is a function called presence function defined as $\mu : R^3 \rightarrow P$, where P is a presence domain, and α is a function called aspect function with domain in R^3 and range in T, where T is called aspect domain.

Where $P = \{0,1\} \subseteq R$. The aspect domain T is not defined because it is not necessary in the current framework. The set of all graphic objects is denoted by O.

For each graphic object $o \subseteq O$, we define the spatial region $Vol(o) \subseteq \mathbb{R}^3$ as the set of all points $p \subseteq \mathbb{R}^3$ such that $\mu(p)=1$. Formally it is:

$$\forall o \in O, \quad Vol(o) = \{\, p \in \mathbb{R}^3 \mid \mu(p) = 1 \,\} \tag{1}$$

Where μ is the presence function of o.

The null or empty graphic object, denoted by ϕ, is the unique graphic object satisfying the following property

$$\forall p \in \mathbb{R}^3 \ \mu(p) = 0$$

Where $\phi=(\mu,\alpha)$. This graphic object fulfils $Vol(\phi)=\Phi$ (Φ denotes the empty set of points). The presence domain satisfies the properties of a boolean algebra.

3. OPTIMIZER ABSTRACT CHARACTERIZATION

Rays

We can define a ray as a graphic object. A ray r is a graphic object (μ,α) such that exists a unique point $q \in \mathbb{R}^3$, and a unique direction vector $u \in \nabla$ such that:

$$\forall p \in \mathbb{R}^3 \quad \mu(p) = \begin{cases} 1 & \text{if } \exists t \in R^+ \mid p = q + tu \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Where $\mathbb{R}^+ \subseteq \mathbb{R}$ is the subset of real values strictly greater than zero, and ∇ is the set of unit length vectors in $\mathbb{R}^3$. The point q is the *origin* of the ray, and the vector u is the *direction* of the ray. The set of all rays is denoted by R.

We define the set $\mathbb{R}^*$ as $\mathbb{R}^* = \mathbb{R} \cup \{\infty\}$, where ∞ is any element that it is not included in $\mathbb{R}$. This value is used to denote an infinity distance. By definition, any value $x \in \mathbb{R}$ holds $x < \infty$

Intersecting Rays and Objects

Let $r \in R$, be a ray, and let $o \in O$ be a graphic object, we define $S(r,o)$ as follows:

$$S(r,o) = \{\, t \in \mathbb{R} \mid \mu(q+tu) = 1 \,\} \tag{6}$$

Where is the presence function of o, q is the origin of r, and u is the direction vector of r. When $S(r,o) \neq \Phi$, an intersection occurs between the ray and this graphic object. Function S returns the set of distances from the origin to all points in the ray which belongs also to the volume of the object. In fact, we only need the lowest one of these real values.

We define the function I with the same domain of S and values in $\mathbb{R}^*$. For each $r \in R$, and $o \in O$ it holds:

$$I(r,o) = \begin{cases} \inf(S(r,o)) & \text{if } S(r,o) \neq \Phi \\ \infty & \text{if } S(r,o) = \Phi \end{cases} \tag{7}$$

Where *inf* denotes the *infimum* of a set of real values, which is always defined even for graphics object whose volume is not a closed region.

The main interest of the above definitions consists of determining which graphic objects in a given scene are intersected by a given ray. In what follows, we will use the symbol S to mean the set of all scenes.

Objects Intersected by a Ray

Let $r \in R$ be a ray, and let $s \in S$ be a scene, we define $C(r,s)$ as the set of graphic objects intersected by r, as follows:

$$C(r,s) = \{\, o \in s \mid I(r,o) \neq \infty \,\} \tag{8}$$

$C(r,s)$ will contain the graphic objects in s intersected by r. Therefore, the condition $C(r,s) \subseteq s$. We also want to know the nearest intersected graphic objects with respect to the ray origin. We also define $C_n(r,s)$ as the set of nearest graphic objects intersected by r as follows:

$$\begin{aligned} C_n(r,s) = \{ & o \in s \mid I(r,o) \neq \infty \\ & \wedge \nexists o' \in s \mid I(r,o') < I(r,o) \} \end{aligned} \tag{9}$$

The expression $C_n(r,s) \in s$ is also satisfied.

Strict Optimizer

Let A be a function with domain in $R \times S$ and values in S. A is a strict optimizer if and only if it fulfils the following condition:

$$\forall r \in R, \quad \forall s \in S, \quad C(r,s) \subseteq A(r,s) \subseteq s \tag{10}$$

In other words, the best optimizer is one which holds $C(r,s)=A(r,s)$, whereas the worse optimizer is one which always holds $A(r,s)=s$, that is, it always yields the whole scene.

Optimizer

Let A^+ be a function with domain in $R \times S$ and values in S. A^+ is an optimizer if and only if fulfils the following condition:

$$\forall r \in R, \quad \forall s \in S, \quad C_n(r,s) \subseteq A^+(r,s) \subseteq s \tag{11}$$

It is easy to prove that any strict optimizer is an optimizer by using the relation $C_n(r,s) \subseteq C(r,s)$ which always holds.

Spatial Representation

Let m be a function with domain in S and values in ε, this function m is a *spatial indexing method* (from now on SIM) if and only if for any given scene $s=\{o_1,o_2,\ldots,o_n\}$ and any given spatial representation $e=\{v_1,v_2,\ldots,v_k\}$ the following equality is satisfied:

$$\bigcup_{i=1}^{n} o_i \subseteq \bigcup_{j=1}^{k} v_j \tag{12}$$

Let $r \in R$ be a ray, let $s \in S$ be a scene, and let $e \in \varepsilon$ be a spatial representation, we consider that when an intersection occurs between an object $o \in s$ and a volumetric object, this volumetric object is also intersected by the ray r. This set is noted as $\Psi(r,s,e)$. This set is more formally defined as follows:

$$\Psi(r,s,e) = \{ o \in s \mid \exists v \in e / v \cap o \neq \phi \wedge v \cap r \neq \phi \} \quad (13)$$

Obviously, $\Psi(r,s,e) \in s$ is always satisfied.

4. SPATIAL INDEXING METHODS

Nowadays, it is possible to propose a SIM classification taking into account the strategy used to build them. In this purpose, we can propose three alternatives to describe a SIM.

Basic SIM

This strategy is useful for spatial indexing which can be built by a unique call to SIM procedure. That is, from a given scene, this procedure will return a set of volumetric objects for the whole scene satisfying the expression (12).

Recursive SIM

This strategy is proposed for scenes which have more complexity (many scene objects). A way to get the volumetric objects is applying some scene partitioning strategies. Thinking in it, two ways to get this purpose will be introduced. Both methods are based on a recursive technique to manage to do the purpose above mentioned. These methods are proposed lower.

4.2.1 Recursive SIM Using a Prefixed Bounding Box

This strategy is based on a recursive procedure. In this SIM, a termination criterion can be one of these: When no graphic object totally o partially is allocated in the prefixed bounding volume, when a known recursive depth level had been reached. This value is an integer number greater than zero. And when the number of graphic objects allocated into the prefixed bounding volume is less or just as known borderline value. This concept is called *maximum number of objects allocated*. This value is set greater than zero.

$$m(s) = \begin{cases} b(s) & \text{if } l = 0 \quad \vee \quad Card(s) \leq t \\ \bigcup_{i=1}^{n} m_i(s) & \text{otherwise} \end{cases} \quad (17)$$

Therefore, a recursive SIM of this category depends on four parameters:

- Prefixed bounding volume which has allocated the scene objects.
- Basic SIM used in order to partition the prefixed bounding volume.
- Maximum recursive depth level.
- Maximum number of objects allocated into a volumetric object.

In order to formalize it, and to minimize the notation, we introduce a function ρ. This function associates a SIM for each four parameters above defined. Therefore, the domain of ρ is $O \times M \times \mathbb{N} \times \mathbb{N}$ and its range is M.

The formal definition of function ρ is based on a condition. Let q be a basic SIM, let v be a graphic object, and let l and t be integer values satisfying $l \geq 0$ and $t > 0$. In these terms, $m = \rho(v, q, l, t)$ if and only if for any scene s the following equality is satisfied:

$$m(s) = \begin{cases} \{v\} & \text{if } l = 0 \quad \vee \quad Card(s') \leq t \\ \bigcup_{i=1}^{n} m_i(s) & \text{otherwise} \end{cases} \quad (14)$$

where:

- $s' = \{ o \in s \,/\, o \cap v \neq \phi \}$.
- $m_i = \rho(v_i, q, l\text{-}1, t)$.
- $\{ v_1, v_2, \ldots, v_n \} = q(s)$.

4.2.2 Recursive SIM Using a Classification Function

In this case, we start having into account the idea to subdivide the scene into simpler scenes (a less number of scene objects). In a first step, a classification of the scene into subscenes is done. For each subscene, an arbitrary bounding volume is set.

The subdivision of the scene into subscenes is carried out using a classification function. The domain of this function is S and its range is $P(S)$. Υ is a classification function if and only if the following equality is fulfilled:

$$\forall s \in S, \quad \Upsilon(s) = \{s_1, \ldots, s_n\} \Rightarrow \forall i = [1 \ldots n] \quad s_i \subseteq s \quad (15)$$

The set of all classification functions is denoted by C.

When a set of subscenes is computed by this kind of SIM, it also must returns a bounding volume which includes whole subscene it obtained. Therefore, this SIM must join a function to compute this bounding volume.

Let b be a function to compute the bounding volume for a given set of graphic objects. This function has

domain in S and range in O. For all scenes $s \in S$, this function fulfils the following expression:

$$\forall o \in s,\ o \subseteq b(s) \qquad (16)$$

The set of all functions satisfying the above expression is called B. Due to this SIM is recursive, some parameters are necessaries in order to determine when the process must stop. Recursive depth has reached a known value. This value must be greater than zero. A recursive SIM of this category depends on the following four parameters:

- A classification function used to subdivide a given scene into subscenes.
- A function to compute the bounding volume for a given scene (set of graphic objects).
- Maximum recursive depth level.
- Maximum number of objects allocated into a volumetric object.

Using a similar formalization than the previous recursive SIM, we also introduce a function ρ'. This function associates a SIM for each four parameters above defined. Therefore, the domain of ρ' is $O \times B \times N \times N$ and its range is M..

The formal definition of function ρ' is based on a condition. Let Υ be a classification function, let b be a function to compute the bounding volume for a given scene, and let l and t be integer values satisfying $l \geq 0$ and $t > 0$. In this terms, $m = \rho'(\Upsilon, b, l, t)$ if and only if for any scene s the following equality is satisfied:
where:

- $m_i = \rho'(\Upsilon, b, l\text{-}1, t)$.
- $\{ s_1, s_2, \ldots, s_n \} = \Upsilon(s)$.

Optimizer Based On Spatial Indexing

There are many different classes of optimizers. Our attention will be focused on a sub-type or category. This sub-type will be called optimizers based on spatial indexing.

Let A be an optimizer. A is an optimizer based on spatial indexing if and only if the following property is fulfilled:

$$\exists! m \in M \ / \ \forall s \in S, r \in R, \quad A(r,s) = \Psi(r, s, m(s)) \qquad (18)$$

When an optimizer of this category is implemented the necessary algorithm to build $m(s)$ must be designed and implemented. After this, it is possible to process a wide set of rays. From this set of voxels we obtain the set of objects intersecting them. The function Ψ models this algorithm.

5. AN OPTIMIZATION EFFICIENCY MEASUREMENT

Using the above definitions and results, an optimization efficiency measurement can be defined.

A way to get this measurement is to use a random distribution of rays for a given optimizer, and for the whole scene, and after this to compute the relative gain in efficiency. This computation can be formally expressed by introducing:

$$M(A^+, s, P) = \int_{r \in R} \frac{Card(A^+(s,r))}{Card(s)} dP(r) \qquad (19)$$

where *Card* is the function which returns the number of elements which are in a set, and $P(r)$ is a probability measure function which models the probability distribution of the rays to be processed.

6. CONCLUSIONS AND FUTURE WORKS

In this work, a formal model of optimizer is proposed, and a model for optimizers based on spatial indexing has been proposed. A measure function to study the performances of any optimizer with respect to other one was proposed.

As future work, we are planning to produce definitions of concrete optimizers by applying this formal framework. This measurement can be useful to select the best optimizer. Moreover, it will also allow comparing the performances between any two optimizers.

7. ACKNOWLEDGMENTS

Special thanks to Carlos Ureña and Juan Carlos Torres for their contributions to this work. This work has been supported by a grant coded as TIC2001-2392-C03-03 (Spanish Commission for Science and Technology).

REFERENCES

[Rev00] J. Revelles, C. Ureña, M. Lastra. An Efficient Parametric Algorithm for Octree Traversal. Journal of WSCG 8(2), pp.212-219, 2000.

[Rev01] J. Revelles, C. Ureña. A Formalization of Ray Casting Optimization Techniques. 11th Spanish Conference on Computer Graphics (CEIG'2001), pp.85-98. Girona (Spain) 2001.

[Tor92] J.C. Torres. Abstract Representation of Graphics systems. Graphic Object theory. PhD thesis, Dpt. of LSI. U. of Granada (Spain), 1992.

[Tor93] J.C. Torres, B. Clares. Graphics Objects: A Mathematical Abstract Model for Computer Graphics. Computer Graphics Forum, 12(5), pp.311-328, 1993.