Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2835490 A1 2015/05/27

(21) 2 835 490

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2013/11/27

(41) Mise à la disp. pub./Open to Public Insp.: 2015/05/27

(51) Cl.Int./Int.Cl. *H04S 7/00* (2006.01),
G06T 15/06 (2011.01)

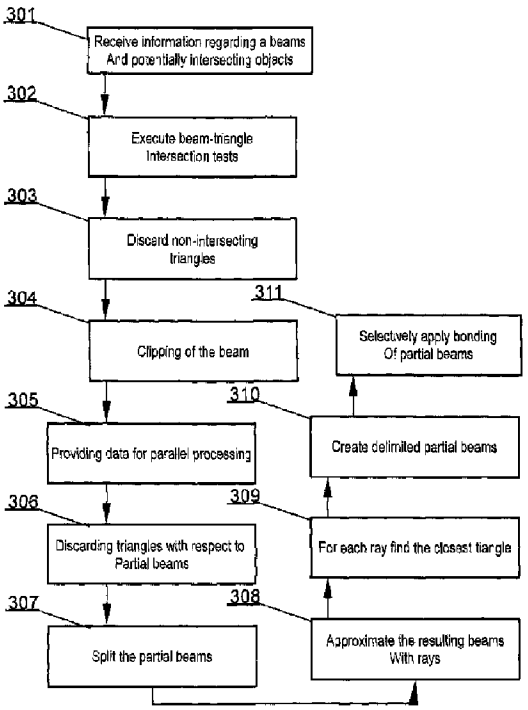
(71) Demandeur/Applicant:
AKADEMIA GORNICZO-HUTNICZA IM. STANISLAWA STASZICA W KRAKOWIE, PL

(72) Inventeurs/Inventors:
PALKA, SZYMON, PL;
PEDZIMAZ, TOMASZ, PL;
ZIOLKO, BARTOSZ, PL

(74) Agent: INTEGRAL IP

(54) Titre : SYSTEME ET PROCEDE PERMETTANT DE DETERMINER UN ENSEMBLE APPROXIMATIF D'OBJETS VISIBLES EN MATIERE DE TRACAGE PAR FAISCEAU

(54) Title: A SYSTEM AND A METHOD FOR DETERMINING APPROXIMATE SET OF VISIBLE OBJECTS IN BEAM TRACING



(57) Abrégé/Abstract:
A computer-implemented method for beam tracing, wherein a set of beams is a representation of a physical wave phenomenon, the method comprising: receiving information regarding a beam and potentially intersecting objects; executing beam-triangle intersection tests; discarding elements, for which an intersection test is negative; dividing the beam into smaller partial beams; providing data for parallel processing; discarding of triangles with respect to partial beams; dividing the partial beams into smaller partial beams; approximating the smaller partial beams with rays; for each ray finding the closest triangle if such a triangle exists; creating delimited smaller partial beams; selectively applying merging of smaller partial beams previously delimited.

ABSTRACT

A computer-implemented method for beam tracing, wherein a set of beams is a representation of a physical wave phenomenon, the method comprising: receiving information regarding a beam and potentially intersecting objects; executing beam-triangle intersection tests; discarding elements, for which an intersection test is negative; dividing the beam into smaller partial beams; providing data for parallel processing; discarding of triangles with respect to partial beams; dividing the partial beams into smaller partial beams; approximating the smaller partial beams with rays; for each ray finding the closest triangle if such a triangle exists; creating delimited smaller partial beams; selectively applying merging of smaller partial beams previously delimited.

APL011-1CA

SYSTEM AND METHOD FOR DETERMINING APPROXIMATE SET OF VISIBLE OBJECTS IN BEAM TRACING

BACKGROUND OF THE INVENTION

Field of the Invention

Embodiments of the invention generally relate to the field of beam tracing, in particular to sound tracing and thus simulating acoustic effects in real time.

Description of the Related Art

Presently, such solutions are mainly used in animated movies and/or cartoons, where calculations are time-consuming, because they do not require real-time regime. In turn, in video games industry there are usually geometry regions comprising activation triggers of various modification of sound.

A prior art Algorithm described in the literature and called Adaptive Frustum http://gamma.cs.unc.edu/SOUND/data/vis2008.pdf; (AD-Frustum: Adaptive Frustum Tracing for Interactive Sound Propagation; Anish Chandak, Christian Lauterbach, Micah Taylor, Zhimin Ren, Dinesh Manocha, Member, IEEE) also http://hal.inria.fr/docs/00/50/99/81/PDF/CG98.pdf (A Beam Tracing Method with Precise Antialiasing for Polyhedral Scenes; Djamchid GHAZANFARPOUR and Jean-Marc HASENFRATZ; Laboratoire MSI - Université de Limoges) use an adaptive beam subdivision for detecting polyhedra. In theory these methods allow to quickly find objects limiting (objects in range of the beam, not obstructed by others) the beam but in practice the structures used in the algorithm are too expensive (their usage is too time consuming) for its use in real-time solutions. Expensive intersection tests comprising specific cases, tree structures and a limited accuracy and slow beam merging algorithm does not allow to take full advantage of modern processors. The same problems make the algorithm difficult to implement and reduce its extension possibilities.

Another prior art publication US 8139780 B2 entitled "Using ray tracing for real time audio synthesis" discloses a sound engine may determine a final sound at a listener location by emulating sound waves within a three-dimensional scene. The

sound engine may emulate sound waves by issuing rays from a location of a sound event and tracing the rays through the three-dimensional scene. The rays may intersect objects within the three-dimensional scene which have sound modification factors. The sound modification factors and other factors (eg., distance traveled by the ray, angle of intersection with the object, etc.) may be applied to the sound event to determine a final sound which is heard by the listener.

Another prior art publication entitled "Interactive Sound Rendering in Complex and Dynamic Scenes using Frustum Tracing" by Christian Lauterbach, Anish Chandak, and Dinesh Manocha, published in IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 13, NO. 6, NOVEMBER/DECEMBER 2007 discloses an approach for real-time sound rendering in complex, virtual scenes with dynamic sources and objects. The approach combines the efficiency of interactive ray tracing with the accuracy of tracing a volumetric representation. There is used a four-sided convex frustum and perform clipping and intersection tests using ray packet tracing. A simple and efficient formulation is used to compute secondary frusta and perform hierarchical traversal. However the algorithm cannot perform real-time simulation and rendering on a high-end PC. This method has proven so complex that it is impossible to be executed in real time, in games on current hardware solutions.

The aim of the present invention is a computer implemented method for determining visibility in beam tracing that would be more efficient in terms of computing resources and lead to decreased time of obtaining the required results and thus enable creating real-time soundtracer.

Summary of the invention

The object of the invention is a computer-implemented method for beam tracing, wherein a set of beams is a representation of a physical wave phenomenon, the method comprising the steps of: receiving information regarding a beam and potentially intersecting objects; executing beam-triangle intersection tests; discarding elements, for which an intersection test is negative; dividing the beam into smaller partial beams; providing data for parallel processing; discarding of triangles with

respect to partial beams; dividing the partial beams into smaller partial beams; approximating the smaller partial beams with rays; for each ray, finding the closest triangle, if such a triangle exists; creating delimited smaller partial beams; selectively applying merging of smaller partial beams previously delimited.

Another object of the invention is a system for beam tracing, wherein a set of beams is a representation of a physical wave phenomenon, the system comprising: a hierarchical model memory for storing a scene representation; a dedicated signal processor for performing intersection tests of ray-triangle and beam-triangle type; a controller, coupled via a bus to the processor, configured to execute the steps of: receiving, via an I/O controller, information regarding a beam and potentially intersecting objects; executing beam-triangle intersection tests; discarding elements, for which an intersection test is negative; dividing the beam into smaller partial beams; providing data for parallel processing; discarding of triangles with respect to partial beams; dividing the partial beams into smaller partial beams; approximating the smaller partial beams with rays; for each ray finding the closest triangle if such a triangle exists; creating delimited smaller partial beams; selectively applying merging of smaller partial beams previously delimited.

Another object of the invention is a computer readable non-transitory storage medium storing computer-executable instructions performing all the steps of the computer-implemented method according to the invention when executed on a computer.

Preferably, the information regarding potentially intersecting objects is input as a hierarchical or not hierarchical representation of a scene composed of objects, preferably a scene tree.

Preferably, the method further comprises providing data for parallel processing partial beams and respective triangles in relation to the partial beams.

Preferably, the discarding of triangles with respect to partial beams is executed based on a result of a test on intersecting with the partial beams and those for which the intersection test is negative are discarded.

Preferably, the step of creating delimited smaller partial beams involves delimiting the beams with an object, which is located closest to the beginning of a ray approximating this particular smaller partial beam.

Preferably, the object located closest to the beginning of a ray approximating this particular smaller partial beam is determined based on a ray-triangle intersection test.

Preferably, the ray-triangle intersection test is the Ingo Wald's test.

Preferably, the ray-triangle intersection tests are executed in parallel with a use of SSE or AVX instructions.

Preferably, the merging of smaller partial beams is executed for smaller partial beams delimited by matching triangles.

Preferably, the matching triangles are the same triangle or two different triangles located on the same plane and having the same material, or triangles located in planes having similar normal or having acoustic material of similar characteristics.

Preferably, the set of beams is a representation of a sound wave.

Brief description of drawings

The object of the invention has been presented in an exemplary embodiment in a drawing, in which:

Fig. 1 presents a set of four cases depicting usage of a ray – triangle approach;

Fig. 2 shows division of a beam into a predefined number of sections and approximation of each of these sections with a separate ray;

Fig. 3 presents the process of Adaptive Ray-Frustum;

Figs. 4A – 4I present the method according to the present invention on an exemplary scene; and

Fig. 5 presents a block diagram of a system according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

NOTATION AND NOMENCLATURE

Some portions of the detailed description which follows are presented in terms of data processing procedures, steps or other symbolic representations of operations

on data bits that can be performed on computer memory. Therefore, a computer executes such logical steps thus requiring physical manipulations of physical quantities.

Usually these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. For reasons of common usage, these signals are referred to as bits, packets, messages, values, elements, symbols, characters, terms, numbers, or the like.

Additionally, all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Terms such as "processing" or "creating" or "transferring" or "executing" or "determining" or "detecting" or "obtaining" or "selecting" or "calculating" or "generating" or the like, refer to the action and processes of a computer system that manipulates and transforms data represented as physical (electronic) quantities within the computer's registers and memories into other data similarly represented as physical quantities within the memories or registers or other such information storage.

A beam is considered herein to be a three dimensional object having a base that is a polygon (frequently located on a plane), having a defined range and edge vectors originating from vertices of the base (all such vectors should be directed at one half plane).

If extension of the edge vectors intersects at a certain point this point is called an apex. For example in case of a beam having a quadrangle base, two of the four vectors may intersect at one point and the other two in a second point. In such a case the beam virtually starts at the edge.

The most suitable bases are a triangle and a quadrangle, however, the present method may be applied to any polygon. A beam having a quadrangle base is frequently called a frustum.

The present invention is a new method called Adaptive Ray-Frustum created after iterative attempts to extend the Adaptive Frustum algorithm and testing how this extensions impact performance.

The method is based on an observation that it is advantageous to replace the small number of complex tests with an increased number of much simpler tests. In

practice, most of the quadtree structure used to manage beam subdivision in the Adaptive Frustum algorithm reaches the maximum level of subdivision after addition of a low number of triangles. It turns out, that in many cases the number of simple tests used in Adaptive Ray-Frustum can be lower than the number of complex tests in Adaptive Frustum.

The method according to the present invention is based on an assumption that a sufficiently small beam can be approximated with a ray. For example, as mentioned earlier, objects in a scene used for the simulation of sound are much less complex than the corresponding graphic objects in scenes. This means that when a beam of a small size is approximated with a ray, the likelihood of error is low. In addition, most of the beams are fully obstructed or not obstructed at all.

If it is assumed that a triangle tested for intersecting with a beam is sufficiently large that it is necessary to execute intersection tests with every part of the beam (after k-th clipping of the beam, the number of its parts is 4^k), there will be executed 4^{k+1}-1 tests.

The present invention introduces execution of up to 4^k simpler tests and 4^p-1 complex beam-triangle tests. An appropriate selection of p and k parameters allows for significant increase of processing speed and thereby significantly reduces execution time.

APPROXIMATION OF A BEAM WITH RAYS

Subdivision of a beam into parts is performed to allow binary determination whether a triangle intersects with appropriately small part of the beam or not. When an intersection occurs, it is assumed that a part of the triangle intersecting the beam, fully obstructs it. Indeed, this is true only for a very small beams.

Since it is decided to use such a simplification, it appears unnecessary to use an expensive intersection test between the beam and the triangle. This expensive test can be replaced with a much simpler check whether a ray routed through the center of the beam intersects with the triangle.

On the other hand, for large beams the approximation with a ray may introduce a high error and lead to a rejection of relevant triangles. The following description discusses the individual components of the solution.

The difference between a large and a small beam lies in its appropriate division taking into account a given scene. A division allowing to detect many triangles achieves a small beam while a division which does not allow to detect many objects in a scene results in a large beam. This is a quantitative approach. For example assuming there is a one point source of sound and 6 beams are emitted from the source, each originating at a different wall of a hexagon defined on the source.

If the beams are not split into partial beams, for each there may be detected 1 wall, which means that the sound may reflect from at most 6 different walls. If each beam is divided one time, there will be 24 partial beams which implies that the number of objects the sound may reflect off is at most 24. In case of two times division, there are up to 96 detected objects and in case of three times division there are up to 384 objects, while in case of four times division, there are up to 1536 objects detected.

Therefore referring to a large beam means a beam insufficiently divided with respect to the scene, that is not capable of detecting numerous objects (triangles). This problem is partially depicted in Fig. 1C, where a large triangle is not detected at all because the beam has not been divided and the ray has missed the triangle. As shown in Fig. 1D the smaller triangle obstructs the large triangle and because the division of the beam is insufficient (large beam) it will not detect the large triangle.

The above descriptive definition may also be quantitatively defined with an equation i.e. the method does not detect objects smaller than (2^(p-1))/h, wherein h is a distance to a given object and p is the number of divisions. For example: in case of a four times division and objects 10 meters away, the resolution of the method is 1.25 meter and only object larger than this number will certainly be detected. Smaller objects have only a certain probability of detection but there will never be a certainty.

USE OF RAY-TRIANGLE INTERSECTION TEST IN ADAPTIVE FRUSTUM

Adaptive Frustum algorithm performance tests show that its computationally most expensive part is the intersections tests. Such a test is executed frequently and

each of its execution requires not only a lot of calculations, but also loading into a memory information about given sub-beam.

The ideal way to speed up the intersection algorithm is to eliminate the need for mentioned test execution. However, it is not possible for all the triangles. Simplifying the intersection test can thus significantly speed up the processing of beams. A further improvement is to replace the "beam – triangle" intersection test with a "ray – triangle".

Unfortunately, the beam approximation with a ray is inaccurate for large beams. In this case, as potentially limiting, there would be taken into consideration only the objects intersecting with the ray passing through the center of the beam. This approach is flawed for large beams. The use of such a simplified test for large beams in conjunction with proper level of beam subdivision significantly reduces the accuracy of the algorithm.

Fig. 1 presents a set of four cases depicting use of a ray – triangle approach. Case (a) presents a ray (101), approximating a beam (102), which starts at the beam's starting plane (103).

Case (b) presents a ray intersecting with a triangle (104). The triangle (104) is thereby classified as intersecting with the beam.

Case (c) presents a large triangle (105) that intersects with the beam but the ray (101) approximating the beam does not intersect with the triangle (105). As a result, the triangle (105) is erroneously classified as not intersecting with the beam.

Case (d) presents two triangles (104, 106) intersecting with the beam (102). The approximating ray (101) intersects with both triangles, whereas the smaller triangle (106) is closer to the beginning of the ray. The smaller triangle (106) is classified as covering the whole plane of the beam (102) although the larger triangle (104) is a better candidate.

AN ALGORITHM USING A GRID OF RAYS

A solution of the problem of large beams approximation with a ray is a uniform division of the beam into a predefined number of sections and approximation of each of these sections with a separate ray as shown in Fig. 2.

Figure 2 shows, as case (a), a possible approximation of a beam with four rays. This approach allows for four times higher accuracy. It means that without division there could be one triangle detected of those that intersect with the beam. Division into four partial beams allows to detect four triangles. Further division will increase accuracy but also increase the computational effort.

A test of four rays can be effectively accelerated with a use of SSE (Streaming SIMD Extensions) instructions. Case (b) shows an approximation of a beam with sixteen rays, equivalent to a threefold division of the beam.

Additionally, AVX (Advanced Vector Extensions) instructions use make it feasible to execute the intersection test of all the rays with a given triangle with two test runs (instead of sixteen runs without the vector instructions), which gives further (in the range of 3 – 4 times depending on instructions used and number of accesses to memory etc.) increase of speed.

Using the SSE allows to test intersection of sixteen rays with a triangle in four runs, giving up to four times increase of speed. AVX differs from the SSE mainly in that AVX executes two times more computations during data processing.

After creating a grid of rays, for each of the rays a triangle must be found, for which the distance along the direction vector from the beginning of the ray to the plane of the triangle is minimal.

It is assumed that this is the triangle that limits the beam's section that is processed. In order to reduce the number of beams created by reflection from delimiting objects, there is introduced an additional step of merging parts of the beam limited by matching triangles (a compatible triangle is the same triangle or two different triangles located on the same plane and having the same material, or triangles located in planes having similar normal and plane constant and having acoustic material of similar characteristics.

In this approach, the adaptability of the algorithm during the step of finding limiting triangles is, effectively pushed to the final stage of the process.

The solution described above is much simpler than the one proposed in Adaptive Frustum algorithm. It does not require the use of tree structures, eliminates up to half of the retained beam parts and significantly reduces the size of the data needed for description of the beam part.

In practice, if for each ray method has found the closest triangle (by scanning all the triangles and executing the intersection test with the ray), it is possible to significantly reduce memory usage in this part of the algorithm.

The problem of the approach described herein is a large number of tested ray-triangle intersections. The number of triangles potentially intersecting with the beam is large, however, most of these triangles ultimately prove not to limit the beam. Nevertheless, for small values of maximum beam subdivision, the presented approach is more efficient than the approach based on Adaptive Frustum.

A SOLUTION COMBINING BEAM-TRIANGLE AND RAY-TRIANGLE TESTS

Algorithms operating directly on beams and operating only on rays, have major weaknesses. The first of them is constrained by the requirement for costly tests, but the hierarchical processing allows fast rejection of triangles. The second algorithm allows for much faster processing of a single triangle, but the number of verified triangles is too high.

The solution to this problem is to combine the advantages of both algorithms. The "beam-triangle" intersection tests are well suited for large beams, and therefore there a quick and inaccurate test for coarse rejection of triangles that certainly do not intersect with the beam can be used to improve efficiency. The number of triangles to be processed within a given scene, reduced by beam intersection test, in turn significantly reduces the number of simple "ray-triangle" intersection tests. This leads to better performance.

Another improvement of the aforementioned method is to use p-times division of the beam with application of the “beam-triangle” test for discarding the highest number of triangles. After execution of this step, a given section of the beam is divided k-p times and the obtained sections are approximated with rays.

The practical tests have shown that the p values giving the best results range between k/4 to k/2 depending on the applied „ray-triangle” intersection test and the type of vector instructions used in the implementation. As a selection of a value between k/4 to k/2 is scene dependent it is best to execute tests of a given scene. A default value may also be applied.

THE INGO WALD TEST

The test is based on the use of pre-calculated values for calculating barycentric coordinates of a point of intersection of a ray with the plane of a triangle. This algorithm is well suited to implementation using vector instructions, where several rays are tested at the same time in terms of existence of an intersection with a single triangle.

Fig. 4 presents that partial beams are approximated with a certain number of rays. Thereafter for each triangle that intersects with a given partial beam there are executed intersections with rays that approximate such partial beam. This is the moment when an intersection test, such as Ingo Wald's test, is required. This may be any intersection test such as Plucker coordinates test defined by Moller-Trumbore or other (such as

- Möller Tomas i Trumbore Ben. Fast, minimum storage ray-triangle intersection. *Journal of Graphics Tools 2(1).* 1997, s. 21-28;
- Shevtsov Maxim, Soupikov Alexei i Kapus Alexander; Ray-Triangle Intersection Algorithm for Modern CPU Architectures. *Proceedings of GraphiCon'2007 conference,* 2007;
- Wald Ingo; Realtime Ray Tracing and Interactive Global Illumination. PhD Thesis. Saarbrucken : Max-Planck-Institut fur Informatik, 2004.

There are three possible versions of the implementation of the algorithm:

- an implementation that uses no vector instructions;
- an implementation that uses SSE instructions - allows to perform an intersection test of four rays at one time against one triangle
- an implementation that uses AVX instructions - allows to perform an intersection test of eight rays at one time against one triangle.

Additionally, there may be implemented intersection of a plurality of triangles with one ray and acceleration of intersection of a single ray with a single triangle. Use of AVX and/or SSE and test of a plurality of rays at one time have proven most efficient on typical CPUs but other versions may be more efficient on other platforms or be easier to implement in designated hardware.

THE ADAPTIVE RAY-FRUSTUM ALGORITHM

The Adaptive Ray-Frustum Algorithm uses two levels of triangles processing in order to reduce the number of "ray-triangle" tests made in its last phase. The first level of processing is to discard these triangles, which certainly do not intersect with the beam or one of its parts. In addition, a scene tree is used to provide approximate set of triangles intersecting the beam. Such object can be defined as a hierarchical or non hierarchical structure and can provide spatial partitioning of the scene.

There is set a maximum depth of division in this part of the algorithm as p. From the list of triangles potentially intersecting with the beam there are chosen those, for which the "beam-triangle" test returns a positive result. Next, this list is processed in the same manner in each of the four parts of the beam. Subsequently, after p steps, the resulting list of potential triangles restricting the beam is passed to the second part of the algorithm.

The second level of processing in the Adaptive Ray-Frustum algorithm is a k-p fold division of the beam into parts and approximation of each part with a ray. For each of the triangles obtained from the first level, there is executed an intersection test with all rays approximating that part of the beam.

The state of each ray is described by a triangle, which intersects with this ray and a distance to the intersection point. If the tested triangle is closer than the recorded in the current state of the ray and the intersection test is positive, the state of the ray is updated.

The last step of the algorithm is to attempt merging of matching beams in order to reduce the number of beams stored in the algorithm.

The aforementioned method may be formulated in pseudocode as the following three methods:

[The Adaptive Ray-Frustum Method]

```
Adaptive Ray-Frustum(beam, maximum_beam_division,
                     pre_processing_depth, scene_tree)
{
1  Triangles = find_triangles(scene_tree)
```

```
2  If Triangles.count > 0:
   {
4    grid_of_rays = Create_Grid_Of_Rays(beam)
5    Process_The_Beam(beam, maximum_beam_division,
5                     pre_processing_depth, Triangles,
                      grid_of_rays)
6    Merge_partial_beams(grid_of_rays, maximum_beam_division)
   }
}
```

[The beam processing method]

```
Process_The_Beam(beam, maximum_beam_division,
                 pre_processing_depth, Triangles_To_Process,
                 grid_of_rays)
{
1  For each of partial beams:
   {
2    Triangles = Triangles_Intersecting_With_The_Beam(
                   beam_part, Triangles_To_Process)
3    If pre_processing_depth > 0:
     {
4      Process_Beam(partial_beam, maximum_beam_division - 1,
        pre_processing_depth - 1, Triangles, grid_of_rays)
     }
5    Else:
     {
6      Process_Ray_Set(partial_beam,
                       maximum_beam_division - 1,
                       Triangles, grid_of_rays)
     }
   }
}
```

[The method for processing a set of rays]

```
Process_Beams_Set(partial_beam,
                  maximum_beam_division - 1,
                  Triangles, grid_of_rays)
{
1  Rays_Count = maximum_beam_division * 2
2  For Each Ray Of The partial_beam:
   {
3    Triangle = Closest_Triangle_Intersecting_With_The_Ray(
                    Triangles)
4    Store the Ray and the Triangle in the grid_of_rays
   }
}
```

The process of finding the closest triangle may be as simple as scanning through the list of objects and selecting the one that intersects with a ray and the distance from the source of the ray to the intersection point is minimal (at this point the ray-triangle intersection test is applied).

Alternatively, one may sort the triangles according to distance of one vertex from the source plane of the beam (sometimes it allows for earlier completion of the search for the closest triangle because if one is found and is closer than the minimum distance of those yet to scan – the process may be stopped).

THE ALGORITHM FOR BEAMS MERGING

The fact of a uniform division of the beam into parts can significantly simplify merging together matching parts. Matching parts are considered parts that are adjacent and are limited by the same triangle or a triangle having the same material and positioned in the same plane. Sweep algorithm is used to browse the grid of partial beams and to find areas, which, are limited by the matching triangles (matching partial beams are limited by matching triangles). Such areas are then

merged in order to reduce the number of beams to be processed in the algorithm for tracking beams.

Beam merging is advantageous, because a beam is divided k times and has 4^k parts, which means that there will be created 4^k of reflected beams. Each of the reflected beams will then be divided k times into 4^k parts, which gives a maximum of $4^k * 4^k = 4^{2k}$ beams. For example, with an assumption that a source emits 6 beams in sound tracing engine in which only reflections are allowed, the final maximum number of beams (belonging to the same sound source) generated in the algorithm is $6 * (1 + 4^k + 4^{2k} + 4^{4k})$. For k=0 this means a maximum of 24 beams and for k=1 a maximum of 1662 beams, for k=2 a maximum of 394854 beams, for k=3 a maximum of 100688262 beams. Over 100 million beams in case of a division of each beam into 64 parts is an unacceptable number.

Limitation of the number of beams is of particular advantage at the first and the second reflection, because these beams will have to be further processed. For example in an exemplary test model there are created 1425383 beams in a conventional algorithm executed in 1.35s while after employing of the new algorithm of beams merging, their number is reduced to 4559 and the execution time is 0.04s.

Figure 3 presents the process of Adaptive Ray-Frustum. At step 301, there is received information regarding a sound beam and potentially intersecting objects (using a scene tree). In the Adaptive Ray-Frustum Method presented earlier in pseudocode, this step corresponds to line 1. Subsequently, at step 302 there are executed "beam-triangle" intersection tests. In the beam processing method presented earlier in pseudocode, this step corresponds to line 2. Those elements, for which an intersection test is negative are discarded at step 303.

After the first major part of the method is complete, the procedure advances at step 304 to divide the beam into smaller partial beams. In the beam processing method presented earlier in pseudocode, this step corresponds to lines 3-4. Subsequently, at step 305, there is provided data for parallel processing – for example two partial beams and respective triangles to further processing in relation to

the partial beams. In the beam processing method presented earlier in pseudocode, this step corresponds to lines 3-4.

Next, at step 306, there is executed discarding of triangles with respect to partial beams. The scene elements are verified against intersecting with the partial beams. Those for which the intersection test is negative are discarded. In the beam processing method presented earlier in pseudocode, this step corresponds to line 2.

Subsequently, at step 307, the partial beams are divided into smaller partial beams. In the method for processing a set of rays presented earlier in pseudocode, this step corresponds to line 1. Thereafter, at step 308, the partial beams of step 307 are approximated with rays. In the method for processing a set of rays presented earlier in pseudocode, this step corresponds to line 2. Next, at step 309, for each ray there is found the closest triangle if such a triangle exists. In the method for processing a set of rays presented earlier in pseudocode, this step corresponds to line 3. Thereafter, at step 310, there are created delimited partial beams. Each ray from the previous step corresponds to a single partial beam. The partial beam is delimited by an object, which is located closest to the beginning of a ray approximating this particular partial beam of step 307. In the method for processing a set of rays presented earlier in pseudocode, this step corresponds to line 4.

The final step of the method 311 is to selectively apply merging of partial beams previously delimited. This step is executed in order to limit the result of the processing. In the Adaptive Ray-Frustum Method presented earlier in pseudocode, this step corresponds to line 6.

Figures 4A – 4I present the method according to the present invention on an exemplary scene. The scene comprises a single beam and five obstacles P1 – P5 as shown in Fig. 4A also comprising one beam. Fig. 4B presents a result of executing method steps 302, 303 that is the whole P5 and part of P1 objects are discarded as non-intersecting with the main, initial beam. Fig. 4C presents the result of step 304 of the method wherein in this example the beam is divided into two partial beams. Fig. 4D presents the result of step 305 and 306 of the method. Data for parallel processing of the left partial beam and the right partial beam are provided and objects are discarded with respect to the previously defined partial beams.

Fig. 4E presents further division of the left partial beam and the right partial beam in line with the step 307 of the method. Fig. 4F presents step 308 of the method where partial beams are approximated with rays. Fig. 4G presents step 309 of the method where the closest triangles are found for respective beams. Fig. 4H presents step 310 of the method and the resulting delimited partial beams (not rays as at this stage the processing of partial beams is resumed).

Finally, Fig. 4I presents the final result of step 311 with partial beam merging applied. In this case it was possible to merge two partial beams delimited by the P3 object. Merging of partial beams of P2 was not possible because the walls of the P2 object delimiting the partial beams are located in different planes.

A comparison of the result of the method with an ideal result allows to determine that an appropriate selection of coefficients allows for obtaining an acceptable approximations of a precise result. The phase of merging partial beams allows in this case to decrease the number of final partial beams from 8 to 7. In practice there are frequently situations, in which the number of final beams decreases multiple times after the use of merging algorithm.

All the examples of Fig. 4 have been depicted in a two dimensional case. The objects shown are pillars in the three dimensional space (a front view instead of top view as in Fig. 4. Each edge of a perspective view of the pillar is in 3D represented by two triangles.

Fig. 5 presents a block diagram of a system according to the present invention. The system 501 according to the present invention may be also built as a dedicated hardware module, which will perform analogous role to existing GPU units. Such a module may be split into two core parts: a hierarchical model memory 507 for storing a beam and a scene tree and a dedicated signal processor 505 for performing intersection tests (both: ray-triangle and beam-triangle). The module can also be a part of hardware performing real-time sound tracing, which by using a scene representation, sources and receiver positions could render sound in a similar way to which GPU (Graphics Processing Unit) is used to render graphics.

Additionally there is a general purpose controller 502, for executing the steps of the aforementioned method for sound tracing in cooperation with an I/O controller 506, a memory 503.

The inner circuits will consist of hardware implementation of the aforementioned algorithms. In order to perform transformation from software to hardware one can use existing industry standard solutions like VHDL language.

The invention may also be implemented in a form of a dedicated hardware module, preferably an extension card, that will detect objects. This dedicated hardware module may be connected with other modules to create complete beam tracing or sound rendering module.

A processor of such extension card comprises a set of multithread logical units that using vector instructions process in parallel independent sets of tests of beam-triangle intersection.

Such processor also comprises a set of multithread logical units that using vector instructions process, in parallel, independent sets of tests of ray-triangle intersection.

The extension card also comprises a dedicated dispatcher circuit for assigning sets of tests to idle logical units.

The extension card also comprises a memory having a hierarchical model for storing a scene model. The hierarchical model memory is connected via a bus to the processors. A read/write buffer may comprise rays in a form of normalized directional vectors.

A scene tree is loaded into the hierarchical model memory at an initialization stage. Software using the extension card sends to the card a set of beams, for which visible objects have to be found. In response the software receives triangles intersecting with different rays into which the beam was divided.

Such extension card may be implemented in System on Chip technology (SoC) and may be a dedicated hardware for video gaming machines or computers or for other multimedia and wave phenomena simulation purposes.

A hardware implementation increases efficiency, which is an advantage as performance in gaming environments is always a priority.

The described algorithm significantly speeds up processing in applications for tracing beams in order to simulate the propagation of sound and other applications where beam tracing can be used. The use of a hybrid combination of processing beams and processing of rays eliminates the weaknesses of the Adaptive Frustum

algorithm. The resulting method for finding objects limiting the beam is much simpler to implement and allows the use of a more efficient algorithm for merging the beams. There has been gained a significant improvement in efficiency, while at the same time reducing the number of output beams.

It is a mixture of the Adaptive Frustum Tracing and Uniform Frustum Tracing with appropriate application of both of them. The first algorithm allows for quick discarding of triangles out of range and the other uses simple tests. The present solution allows for execution of some complex discarding tests and then execute as a set of quick simple tests on rays.

The present method due to multi-pass structure first extracts, from the scene tree, cells intersecting with the beam and from these cells extracts triangles subsequently passed to a discarding and beam division algorithm executed for a given time in order to finally approximate partial beams with rays and the remaining triangles are intersected with these rays. In case there is a need to do so, subsequent phases may be combined, for example extracting cells from the scene tree and obtaining triangles from them.

A different embodiment of the present invention may instead of emitting from the source/emitter beams in all directions, emit a predefined number of rays and perform reflections from full triangles (one time for each triangle):

- First, rays are emitted in all directions and there are found triangles with which they intersect; and
- Secondly, There are created reflected beams for all such triangles and subsequently typical processing is applied.

It can be easily recognized, by one skilled in the art, that the aforementioned method for sound tracing may be performed and/or controlled by one or more computer programs. Such computer programs are typically executed by utilizing the computing resources in a computing device such as personal computers, personal digital assistants, cellular telephones, receivers and decoders of digital television or the like. Applications are stored on a non-transitory medium. An example of a non-transitory medium is a non-volatile memory, for example a flash memory or volatile memory, for example RAM. The computer instructions and are executed by a processor. These

memories are exemplary recording media for storing computer programs comprising computer-executable instructions performing all the steps of the computer-implemented method according the technical concept presented herein.

While the invention presented herein has been depicted, described, and has been defined with reference to particular preferred embodiments, such references and examples of implementation in the foregoing specification do not imply any limitation on the invention. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader scope of the technical concept. The presented preferred embodiments are exemplary only, and are not exhaustive of the scope of the technical concept presented herein.

It can be easily recognized, by one skilled in the art, that the aforementioned method for beam tracing, in particular sound tracing, may be performed for simulating other physical wave phenomena like light, radio waves or shock waves. As a result the method can be used in multimedia and simulation systems other than sound tracer.

Accordingly, the scope of protection is not limited to the preferred embodiments described in the specification, but is only limited by the claims that follow.

What is claimed is:

1. A computer-implemented method for beam tracing, wherein a set of beams is a representation of a physical wave phenomenon, the method comprising the steps of:

- receiving (301) information regarding a beam and potentially intersecting objects;
- executing (302) beam-triangle intersection tests;
- discarding (303) elements, for which an intersection test is negative;
- dividing (304) the beam into smaller partial beams;
- providing (305) data for parallel processing;
- discarding (306) of triangles with respect to partial beams;
- dividing (307) the partial beams into smaller partial beams;
- approximating (308) the smaller partial beams with rays;
- for each ray, finding (309) the closest triangle, if such a triangle exists;
- creating (310) delimited smaller partial beams;
- selectively applying merging (311) of smaller partial beams previously delimited (310).

2. The method according to claim 1 wherein the information regarding potentially intersecting objects is input as a hierarchical or not hierarchical representation of a scene composed of objects, preferably a scene tree.

3. The method according to claim 1 further comprising providing (305) data for parallel processing partial beams and respective triangles in relation to the partial beams.

4. The method according to claim 1 wherein the discarding (306) of triangles with respect to partial beams is executed based on a result of a test on intersecting with the partial beams and those for which the intersection test is negative are discarded.

5. The method according to claim 1 wherein the step of creating (310) delimited smaller partial beams involves delimiting the beams with an object, which is located closest to the beginning of a ray approximating this particular smaller partial beam.

6. The method according to claim 5 wherein the object located closest to the beginning of a ray approximating this particular smaller partial beam is determined based on a ray-triangle intersection test.

7. The method according to claim 6 wherein the ray-triangle intersection test is the Ingo Wald's test.

8. The method according to claim 6 characterized in that the ray-triangle intersection tests are executed in parallel with a use of Streaming SIMD Extensions (SSE) or Advanced Vector Extensions (AVX) instructions.

9. The method according to claim 1 wherein the merging (311) of smaller partial beams is executed for smaller partial beams delimited by matching triangles.

10. The method according to claim 1 wherein the matching triangles are the same triangle or two different triangles located on the same plane and having the same material, or triangles located in planes having similar normal or having acoustic material of similar characteristics.

11. The method according to claim 1 wherein the set of beams is a representation of a sound wave.

12. A system for beam tracing, wherein a set of beams is a representation of a physical wave phenomenon, the system comprising:

- a hierarchical model memory (507) for storing a scene representation;
- a dedicated signal processor (505) for performing intersection tests of ray-triangle and beam-triangle type;

- a controller (502), coupled via a bus to the processor (505), configured to execute the steps of:
  - receiving (301), via an I/O controller (506), information regarding a beam and potentially intersecting objects;
  - executing (302) beam-triangle intersection tests;
  - discarding (303) elements, for which an intersection test is negative;
  - dividing (304) the beam into smaller partial beams;
  - providing (305) data for parallel processing;
  - discarding (306) of triangles with respect to partial beams;
  - dividing (307) the partial beams into smaller partial beams;
  - approximating (308) the smaller partial beams with rays;
  - for each ray finding (309) the closest triangle if such a triangle exists;
  - creating (310) delimited smaller partial beams;
  - selectively applying merging (311) of smaller partial beams previously delimited (310).

13. The system according to claim 12 wherein the information regarding potentially intersecting objects is input as a hierarchical or not hierarchical representation of a scene composed of objects, preferably a scene tree.

14. The system according to claim 12 further comprising providing (305) data for parallel processing partial beams and respective triangles in relation to the partial beams.

15. The system according to claim 12 wherein the discarding (306) of triangles with respect to partial beams is executed based on a result of a test on intersecting with the partial beams and those for which the intersection test is negative are discarded.

16. The system according to claim 12 wherein the step of creating (310) delimited smaller partial beams involves delimiting the beams with an object, which is

located closest to the beginning of a ray approximating this particular smaller partial beam.

17. The system according to claim 16 wherein the object located closest to the beginning of a ray approximating this particular smaller partial beam is determined based on a ray-triangle intersection test.

18. The system according to claim 17 wherein the ray-triangle intersection test is the Ingo Wald's test.

19. The system according to claim 17 wherein the ray-triangle intersection tests are executed in parallel with a use of Streaming SIMD Extensions (SSE) or Advanced Vector Extensions (AVX) instructions.

20. The system according to claim 12 wherein the merging (311) of smaller partial beams is executed for smaller partial beams delimited by matching triangles.

21. The system according to claim 12 wherein the matching triangles are the same triangle or two different triangles located on the same plane and having the same material, or triangles located in planes having similar normal or having material of similar characteristics.

22. The system according to claim 12 wherein the set of beams is a representation of a sound wave.

23. A computer readable non-transitory storage medium storing computer-executable instructions performing all the steps of the computer-implemented method according to any one of claims 1 to 11 when executed on a computer.

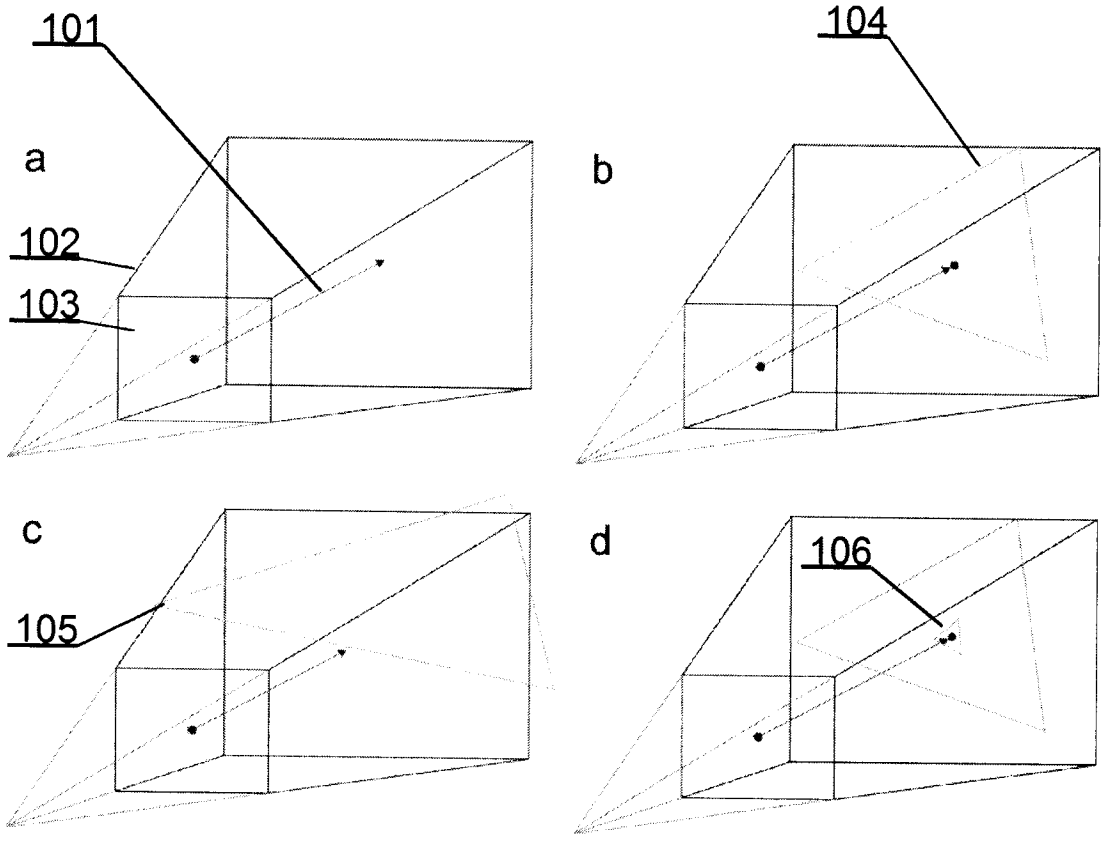


Fig. 1

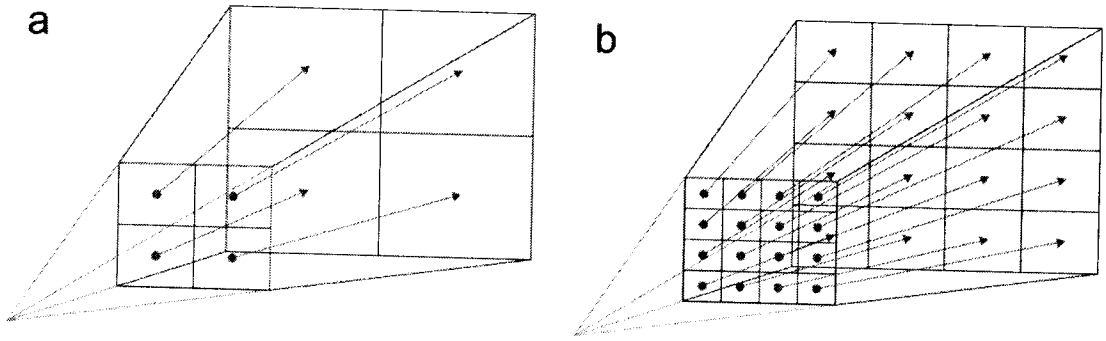


Fig. 2

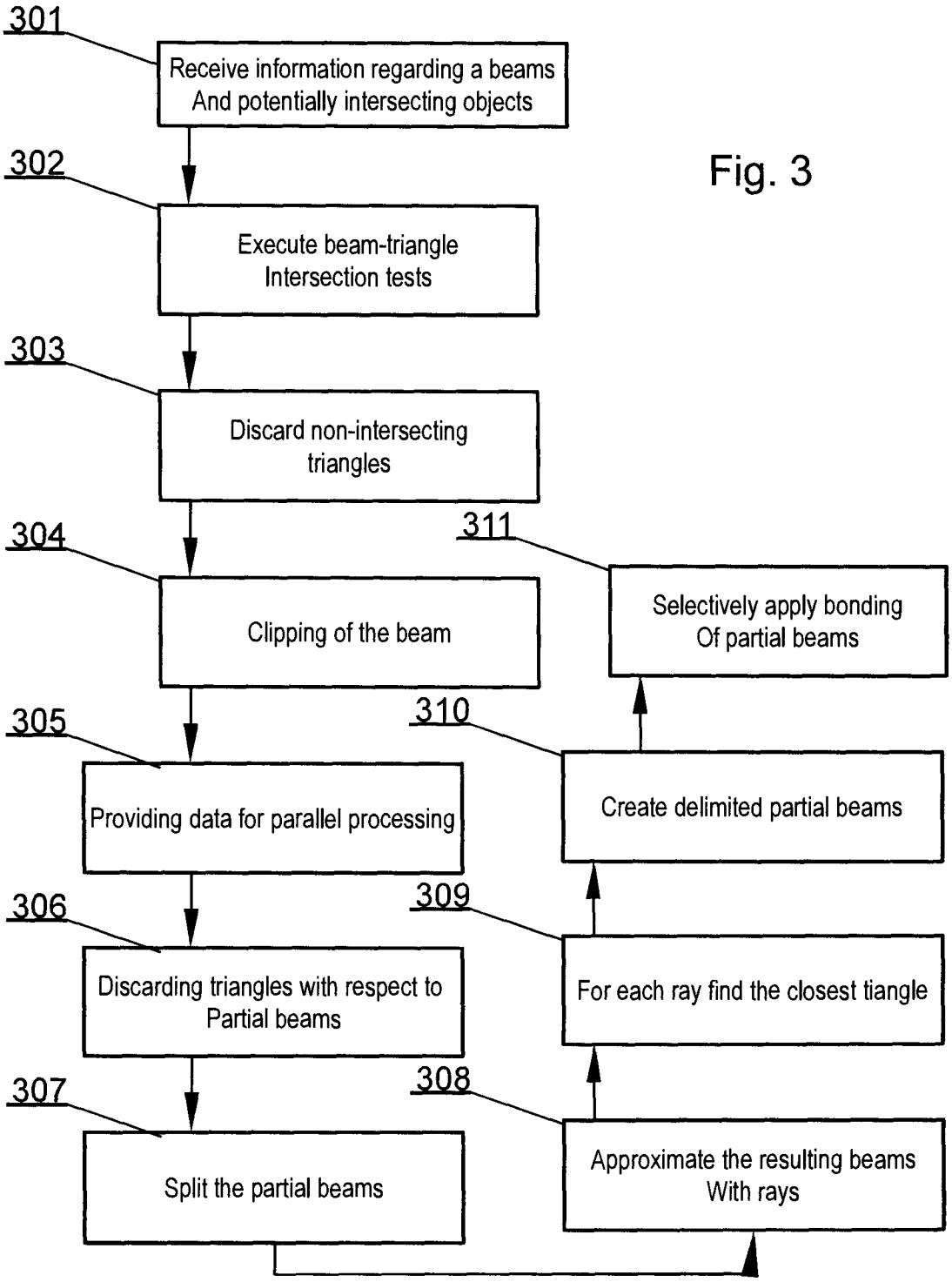

301
Receive information regarding a beams And potentially intersecting objects
302
Execute beam-triangle Intersection tests
303
Discard non-intersecting triangles
304
Clipping of the beam
305
Providing data for parallel processing
306
Discarding triangles with respect to Partial beams
307
Split the partial beams
308
Approximate the resulting beams With rays
309
For each ray find the closest tiangle
310
Create delimited partial beams
311
Selectively apply bonding Of partial beams

Fig. 3

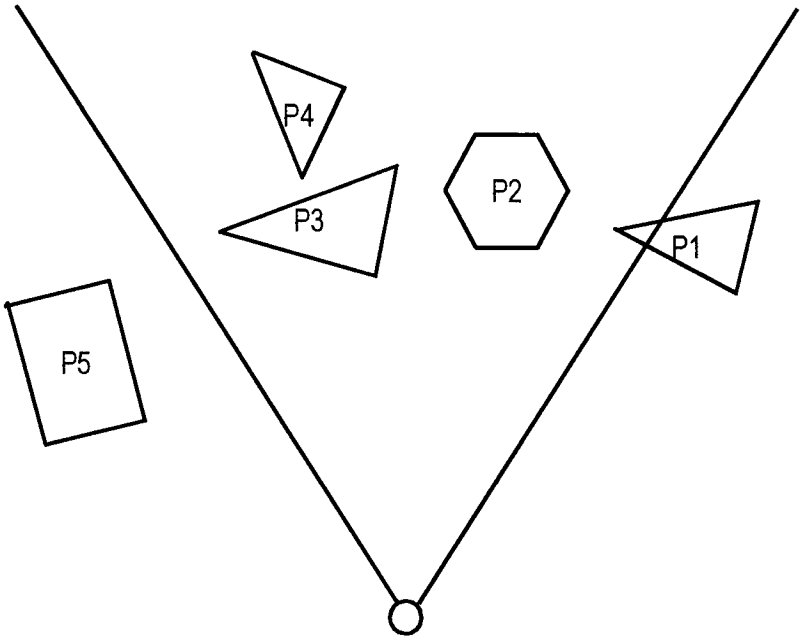


Fig. 4A

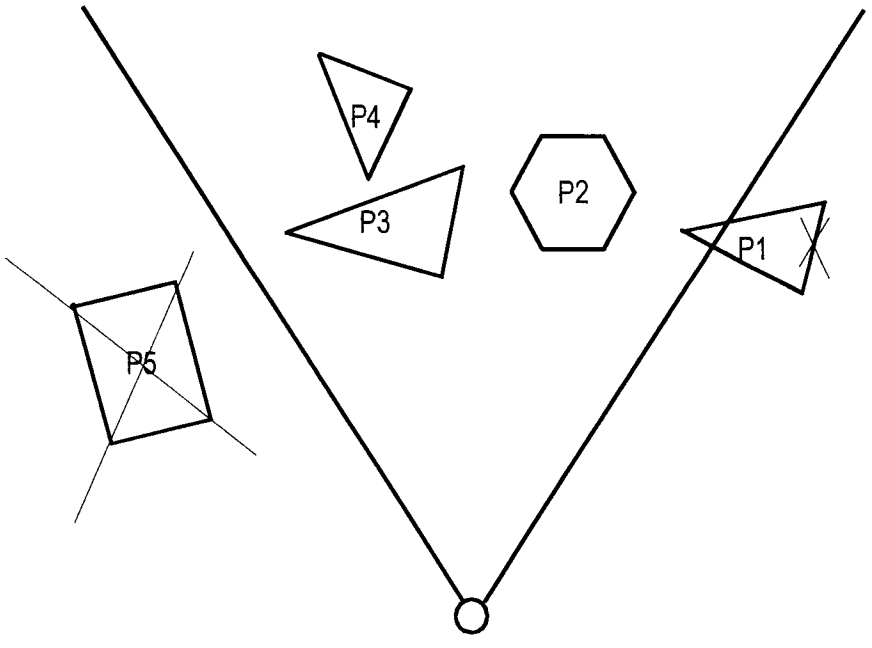


Fig. 4B

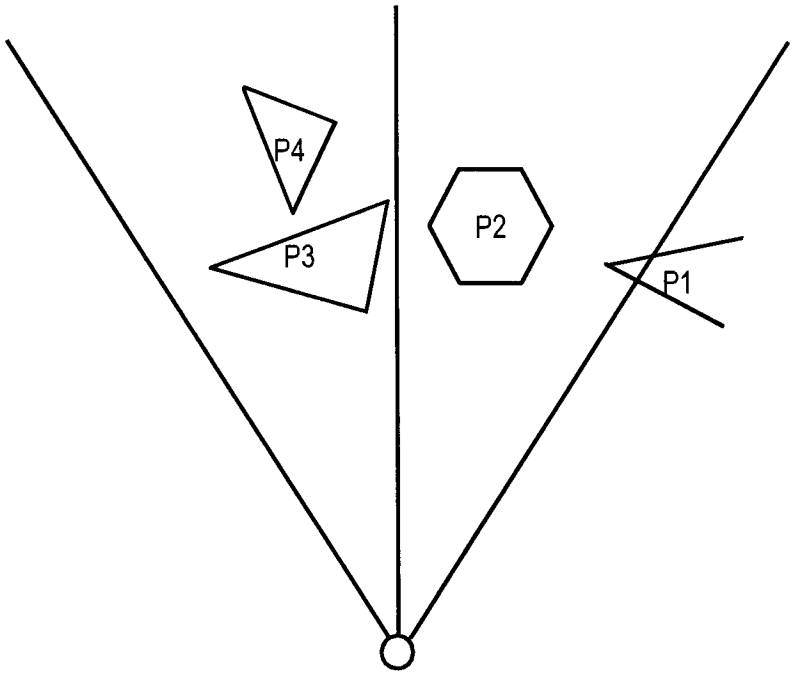


Fig. 4C

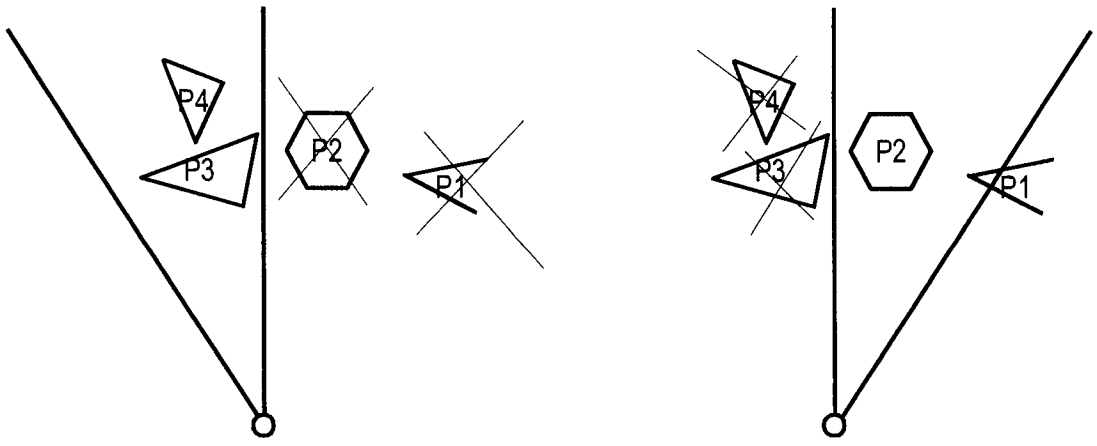


Fig. 4D

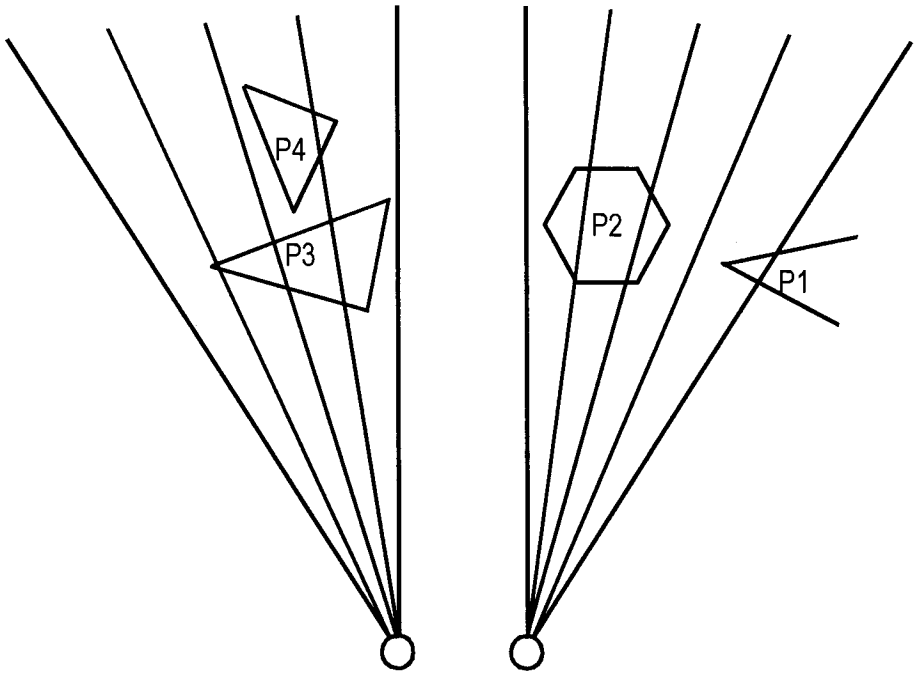


Fig. 4E

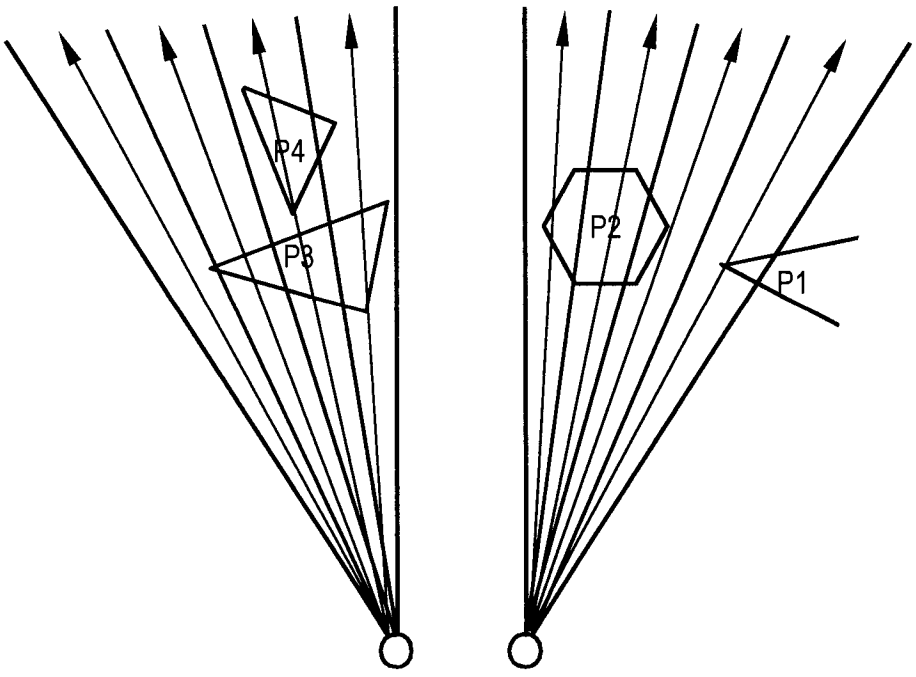


Fig. 4F

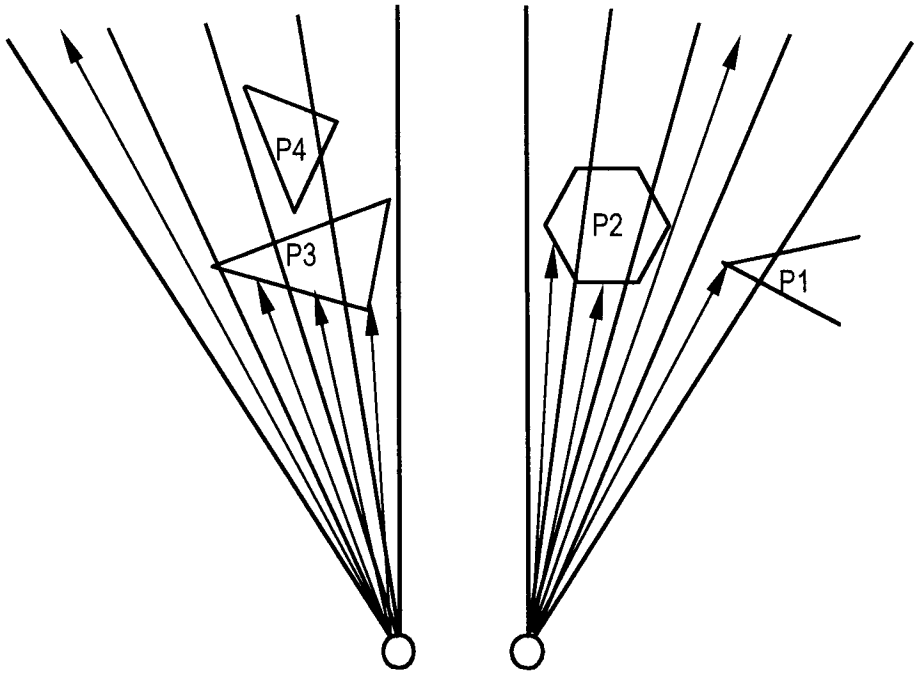


Fig. 4G

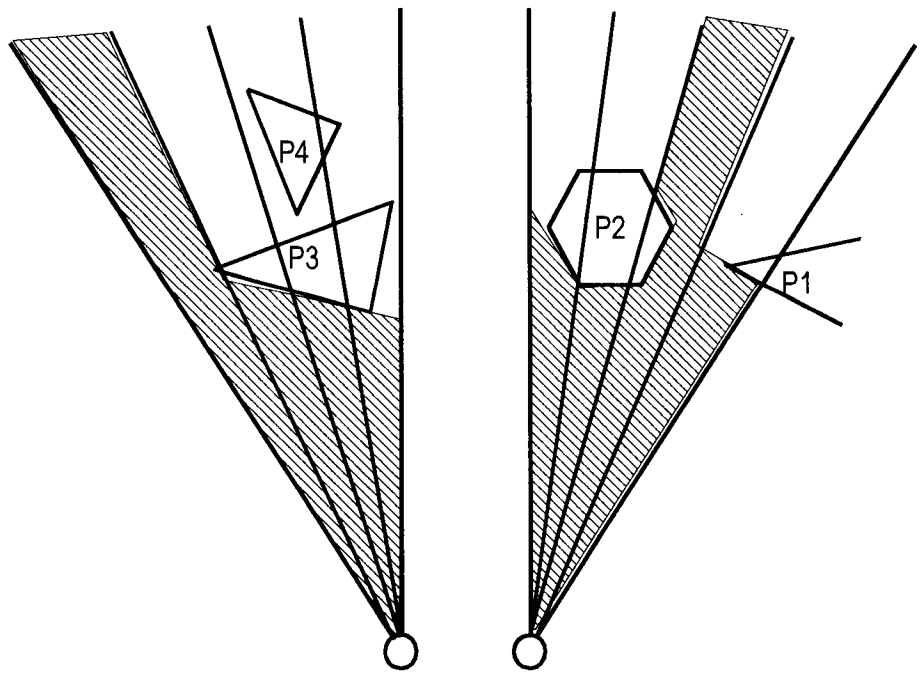


Fig. 4H

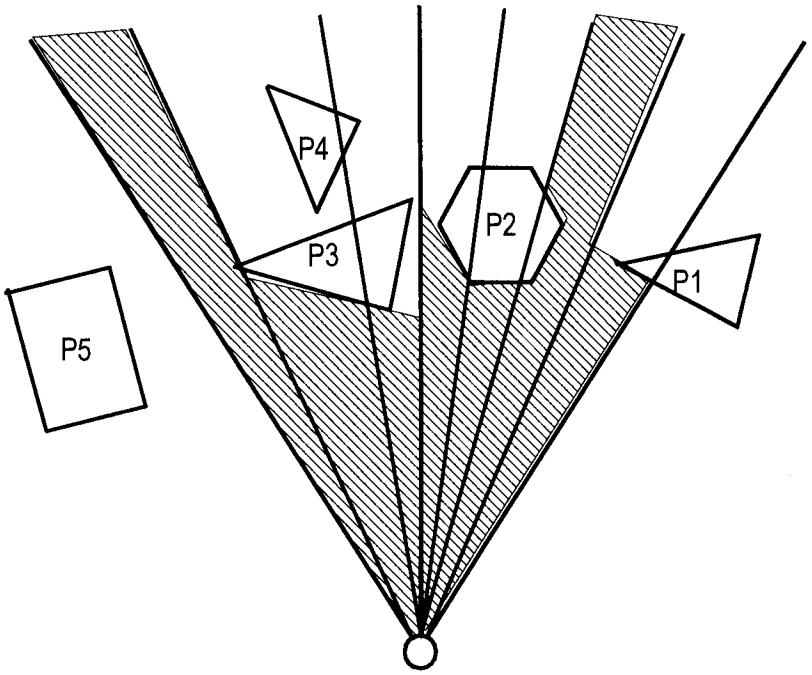


Fig. 4I

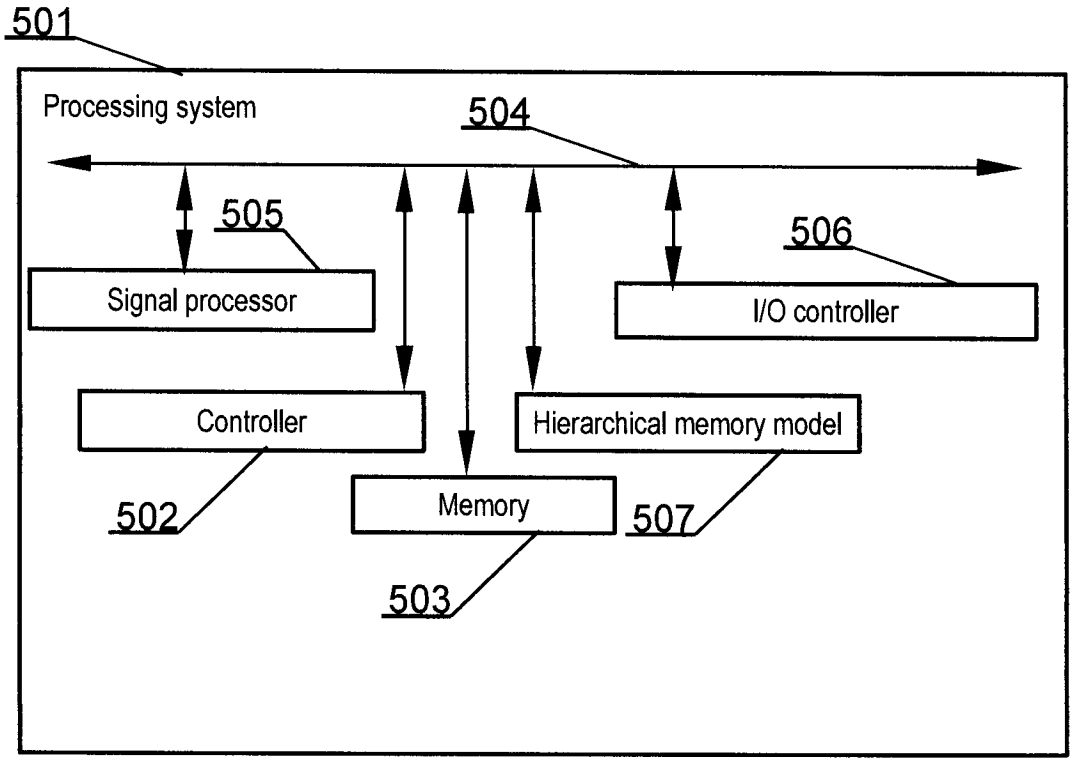
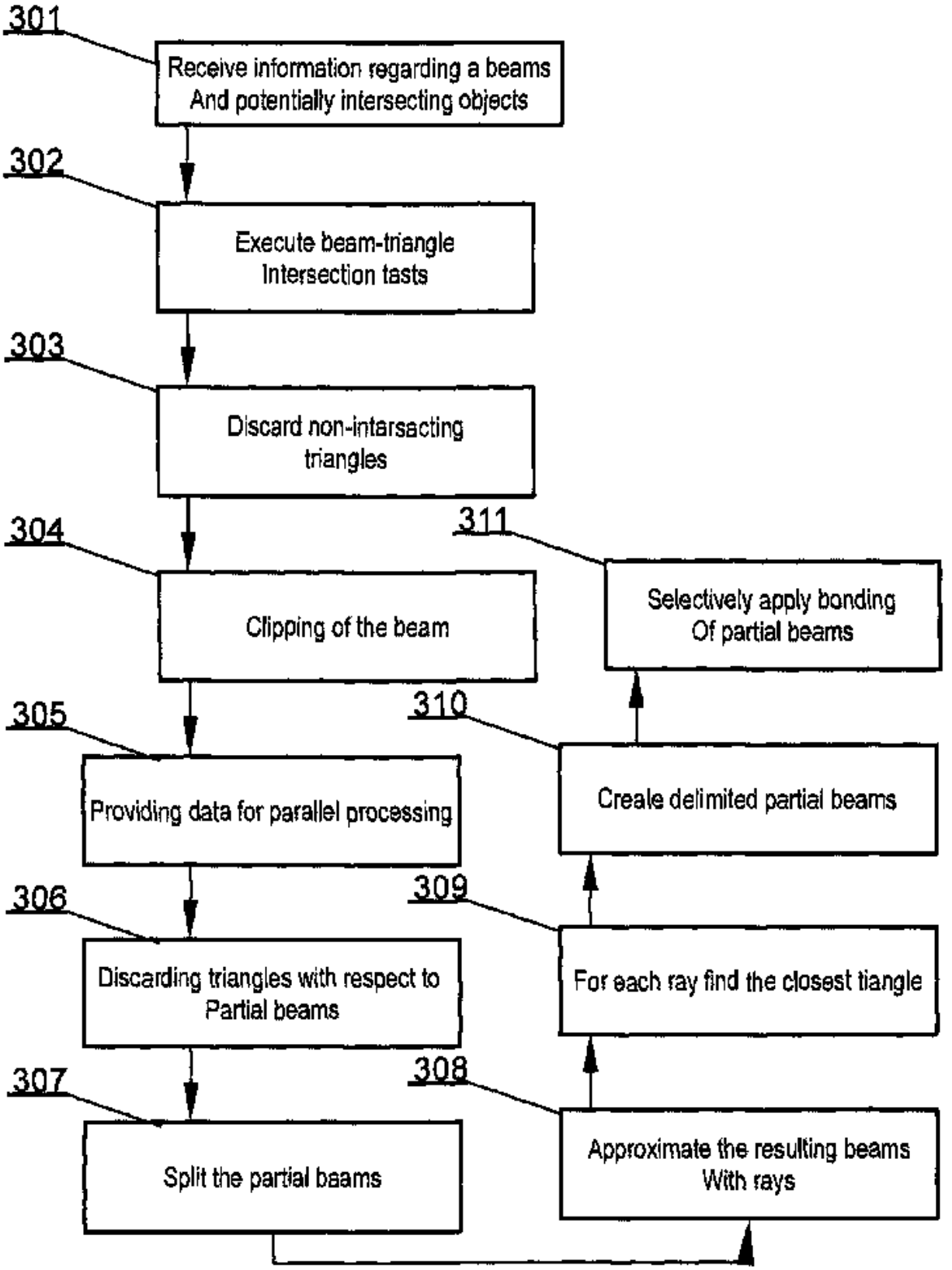

501
Processing system
504
505
506
Signal processor
I/O controller
Controller
Hierarchical memory model
502
Memory
507
503

Fig. 5


301
Receive information regarding a beams And potentially intersecting objects
302
Execute beam-triangle Intersection tasts
303
Discard non-intarsacting triangles
304
Clipping of the beam
305
Providing data for parallel processing
306
Discarding triangles with respect to Partial beams
307
Split the partial baams
308
Approximate the resulting beams With rays
309
For each ray find the closest tiangle
310
Creale delimited partial beams
311
Selectively apply bonding Of partial beams