Online ISSN 1848-3380 , Print ISSN 0005-1144
ATKAFF 52(4), 339–352(2011)

Marjan Sikora, Ivo Mateljan, Nikola Bogunović

Beam Division in Acoustic Simulation of Non-Homogenous Environments

UDK 534.86:534-14
IFAC 4.2.1.6

Original scientific paper

In order to expand the area of use of the beam tracing method, the beam tracing with the refraction method (BTR) was developed. The BTR is best suited for acoustic and hydro-acoustic simulation of non-homogenous environments. The BTR can trace the refraction as well as the reflection of the sound wave, using triangular beams. The geometry of the scene in the BTR is based on triangle meshes rather than polygons. This enables the BTR to simulate complex, irregular shaped objects, including non-convex volumes. Furthermore, the BTR traces beams through several entities filled with different media. This paper presents algorithms and data structures used to divide beams during the interaction of a beam with the complex, non-convex environment. This paper also brings measurements of the implemented beam division code and the comparison of measured results with results of other methods.

Key words: Acoustics, Beam tracing, Hidden surface removal, Hydro-acoustics, Simulation

Dijeljenje snopova kod akustičke simulacije nehomogenih sredina. Proširena metoda praćenja snopova (PMPS) razvijena je kako bi se proširilo područje primjene metode praćenja snopova. PMPS je prilagođena akustičkim i hidroakustičkim simulacijama nehomogenih sredina. Geometrija scene se u PMPS-u temelji na nepravilnim trokutastim mrežama, a ne na poligonima, tako da je moguće simulirati složene, nepravilne objekte. Osim toga, PMPS prati širenje snopova kroz više nekonveksnih entiteta, ispunjenih sa različitim medijima. U ovom radu predstavljeni su algoritmi i strukture podataka pomoću kojih se vrši dijeljenje snopova u PMPS-u. Dijeljenje snopova se događa prilikom interakcije snopa sa složenom, nekonveksnom sredinom. U radu su također prikazana mjerenja vremenske složenosti koda PMPS-a, te usporedba sa drugim metodama.

Ključne riječi: akustika, hidroakustika, praćenje snopova, simulacija, skrivanje poligona

1 INTRODUCTION

The area of the application of the beam tracing implementation presented in this paper is the simulation of sound and ultrasound waves. Other beam tracing implementations exist not only in acoustics [1, 3, 5], but also in other fields such as visualization [8] and calculation of radio wave propagation [13].

As far as authors know, the beam tracing method was never used to simulate the sound wave propagation in non-homogenous environments, where the sound wave propagates through several different media. Such environments can often be found in the real world. Examples of such environments are the propagation of the sound in the sea water and ocean layers, the use of acoustic lenses, the medical ultrasound, the propagation of the sound in geology and seismology. Currently, the model of choice for such environments is the numerical simulation (FEM, BEM) [16]. But numerical methods have limitations that make them unsuitable for situations that have large dimension/wavelength ratio.

The beam tracing with the refraction method (BTR) was developed [12] for environments that have complex, non-convex geometry. In the BTR, when the beam encounters a discontinuum, it has to be accurately divided. This is not an easy task, because the geometry can be non-convex, and because the geometry it is defined with large number of triangles. So the central issue for the accuracy and the performance of the BTR is the process of beam division, which is presented in this paper.

This paper is organized in a following way: this section gives an introduction and analyses the previous work in this field; the second section presents algorithms and data structures used in the beam division process of the BTR, and the physics of the BTR; the fourth section presents results of the implementation of the beam division as a standalone computer program, and the comparison of the BTR simulation of a complex environment with the FEM simulation; the last section gives the conclusion and the future

work. Two appendices are included to clarify some details mentioned in the paper.

1.1 Previous work

The beam tracing method was introduced by Heckbert and Hanrahan [6], as well as Walsh et al. [15] to overcome limitations of the virtual-source and the ray-tracing methods. The aim was to achieve the spatial coherence which is lacking in the ray tracing, as well as the acceptable performance, which is the problem with the virtual-source method. The use and the development of the beam tracing started in the field of visualization. The beam tracing was used for the auralization later, as Farina developed Ramsette in 1995 [3]. In the beginning, he didn't use the beam division, but instead he traced only one reflected beam - the strongest one. Analyzing the simulation results he concluded that in order to enhance the accuracy of the simulation, beams would have to be split when they fell on two different planes. Such an adaptive beam tracing algorithm was developed by Drumm [1]. Funkhouser at al. further enhanced the speed of the algorithm by using preprocessed spatial subdivision structures [5], and they produced a real-time auralization of the architectural environment.

Since the scope of this paper is the algorithm of the beam division let us now discuss some important details by comparing mentioned implementations of the beam tracing with the BTR. Generally, authors of sound simulations use beams with polygonal section [1, 5]. Since these simulation target primarily architectural environments, the geometry of models used in these simulations is defined with polygons. So in order to perform the beam division they have to test for polygon/polygon intersections.

In the BTR all beams have triangular section, and the geometry is defined by triangles (or more precisely by irregular triangle meshes). So in the BTR, all geometric operations during the process of the beam division are done between two triangles. We believe that in this case geometric operations can be simpler and better optimized, than in the case of polygon/polygon or polygon/triangle operations. The other reason why triangles were selected is because the BTR was designed primarily for complex, irregular environments.

2 BEAM DIVISION IN BTR

In the first part of this section the scene composition and the topology is described briefly in order to clarify some phases of the process of the beam division. The second part of this section brings the description of beams. The third part gives the detailed explanation of the beam division process. In-depth explanation of some algorithms used in the division of beams is given in appendices. The fourth part of this section describes briefly the physics of the BTR simulation.

2.1 Scene composition and topology

The scene in the BTR is organized in such a way to enable the tracing of the wave through several volumes filled with different media. These volumes can be non-convex, and a single volume can have several interior islands.

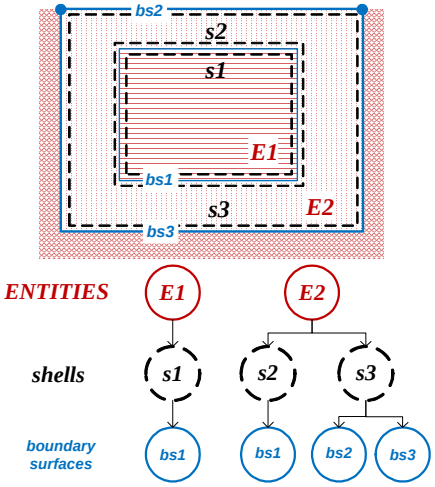


Fig. 1. Scene composition

The hierarchy of the scene structure has tree levels: the entity, the shell and the boundary surface (Fig. 1). An entity represents a volume filled with single media. It is defined by one or several shells. The simple, convex entity is defined by a single shell (E1 in Fig. 1), while the complex, non-convex entity has several shells (E2 in Fig. 1). A shell is the closed triangle mesh, which defines either the outer boundary of the entity, or some of the inner islands in the entity. A boundary surface is the part of the shell, which divides two entities. If the entity is surrounded with only one entity its outer shell is composed of only one boundary surface (s1 in Fig. 1). If the entity is surrounded with two or more entities (volumes filled with the different medium), then its shell would be composed of several boundary surfaces (s3 in Fig.1). Thus, a boundary surface is not always made of the closed mesh, but the set of boundary surfaces that make one shell must form the closed mesh.

The topology in the BTR is used to speed-up the computation. The topology of the scene has two levels. The first level is a surface topology of triangle meshes (that define the boundary surfaces and shells), and the second level is a volume topology that defines the space relationship of entities in the scene. The mesh topology is composed of vertexes, edges and triangles. It is based on a winged-edge structure, and serves to speed up geometric tests that check nearby triangles. The mesh topology is calculated by the simulation during the preprocessing phase.

The volume topology is used during the tracing of

beams, to speed up the traversing of beams between neighbor entities. It ensures that no matter what is the total number of triangles in scene, only the local number of triangles determines the performance of the algorithm. The volume topology is defined by the user, during the definition of the scene structure.

During the preprocessing phase, the binary space partitioning (BSP) tree is calculated for each entity, to speed up the triangle hiding & dividing stage of the beam division. Using the BSP, the algorithm of the beam division efficiently determines the correct order of the visibility for illuminated triangles, even in the case of cyclic hiding of triangles.

2.2 Beams

Beams in the BTR are defined by three edge rays, and consequently have the triangular section. Also, shells are made of triangle meshes, so all geometric operations during the process of beam division are done between two triangles. Since triangles are the simplest polygons, these operations are faster than in the case of triangle-polygon clipping (Overback [8]) or polygon-polygon clipping (Drumm and Funkhouser [1, 5]). The special attention was given to the design of triangle-triangle clipping algorithms (see appendix A for details).

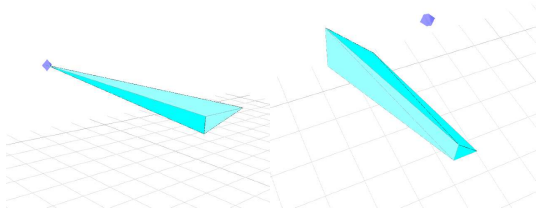


Fig. 2. Left - the initial beam, right - the reflected/refracted beam

Initial beams (Fig. 2 – left) have the form of a triangular pyramid and are composed of four triangles – three side ones, and one closing triangle. Beams that result from reflections and refractions have the form of a clipped pyramid (Fig. 2 – right). They are composed of eight triangles.

The beam tracing in the BTR starts with the generation of initial beams, which results with 20 initial beams. Initial beams are generated using the icosahedron shown in Fig. 3 – left. Each initial beam has its apex in the location of the sound source (in the center of the icosahedron). Corner rays of each initial beam pass through vertexes of one triangle of icosahedron. Thus all beams have the section in the form of an equilateral triangle.

Further tessellation of each beam is shown in Fig. 3 – right. It results in the greater number of narrower initial

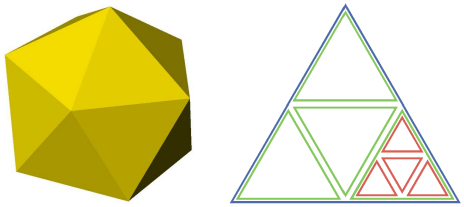


Fig. 3. Left - icosahedron, right - tessellation

beams (80 beams), which can improve the performance of the process of beam division, as will be shown in Section 4. If the scene requires more narrow beams, further tessellations can be applied resulting in $20 \cdot 4^n$ beams, where n is the number of subsequent initial beam tessellations applied.

2.3 Beam division

Each beam propagates inside an entity. The beam eventually encounters the shell of the entity which represents the boundary of the entity. In the simplest case the beam falls inside only one triangle of the shell. In such case the beam is closed with the plane of the triangle. In this case no beam division is necessary. But in most cases, the beam falls on several triangles of the shell mesh. In this case the beam has to be divided into smaller beams, one for each illuminated triangle of the shell. Also, since entities in the BTR are non-convex, some of illuminated triangles could obscure each other. In this case the precise region that is not obscured has to be determined to generate the accurate division of the beam. The correct beam division is of the utmost importance, because it ensures the space coherence of the beam tracing method. The volume of divided beams is used for the calculation of the intensity of sound of each beam, which is in the end used to render the distribution of the level of sound intensity in the scene.

The process of beam division implemented in the BTR is done in the following order:

- find illuminated triangles & backface culling
- transform & clip triangles
- project triangles
- clip projected triangles with the beam
- hide & dive triangles
- create divided beams.

This process is illustrated in Fig. 4, starting with the initial geometry of the scene (first row), and ending with divided beams (last row). The scene consists of the source

(blue) and one entity (magenta) in which one beam (blue rays) is traced/divided. The entity is composed of one outer shell and two inside shells that partially occlude each other.

2.3.1 Finding illuminated triangles and backface culling

The process of the beam division starts with determining the order of visibility of triangles, which is done by traversing the pre-calculated BSP tree. The importance of determining the order of visibility will be explained during the hiding & dividing phase.

Next comes the phase of finding triangles that are completely or partially inside the beam volume - illuminated triangles. These triangles are candidates for the beam division. When this stage is completed, all illuminated triangles are put in a queue named *IT*.

Illuminated triangles in the queue *IT* are then filtered with the backface-culling process in order to remove triangles with the normal in the opposite direction of the beam propagation.

Illuminating & backface culling is illustrated in the 2nd row of Fig. 4. Left picture displays all triangles that form the entity (magenta). The middle picture displays the result of illuminating and backface culling – candidate triangles in the queue *IT* are displayed in cyan. The traced beam is outlined by three blue corner rays.

2.3.2 Transformation and projection of triangles

All triangles in the queue *IT* that remain after the backface-culling are then transformed from the world space to the beam space. (Fig. 4, 2nd row, right). The beam space is a coordinate space, with the origin in the apex of the beam, and whose z coordinate is parallel with the direction of the beam propagation. The matrix used for this transformation is saved for later use. After projection, all triangles are clipped with the plane $z=0$ to avoid the distortion of parts of triangles that are beyond the focal point.

Transformed and clipped triangles are then projected onto the projection plane orthogonal to the z axis (the direction of the beam propagation), in order to accelerate computation since ray/triangle operations in 3D would be slower than in 2D. Projected triangles are then added to the binary sorted tree *PT*. Such data structure is chosen since during hiding & dividing phase the ordered traversal of the projected triangle set will be performed, and this structure ensures that it would be done efficiently. The order in which *PT* is sorted is determined by the order of the visibility of triangles. The closer the triangle is to the sound source, the higher order of visibility it has.

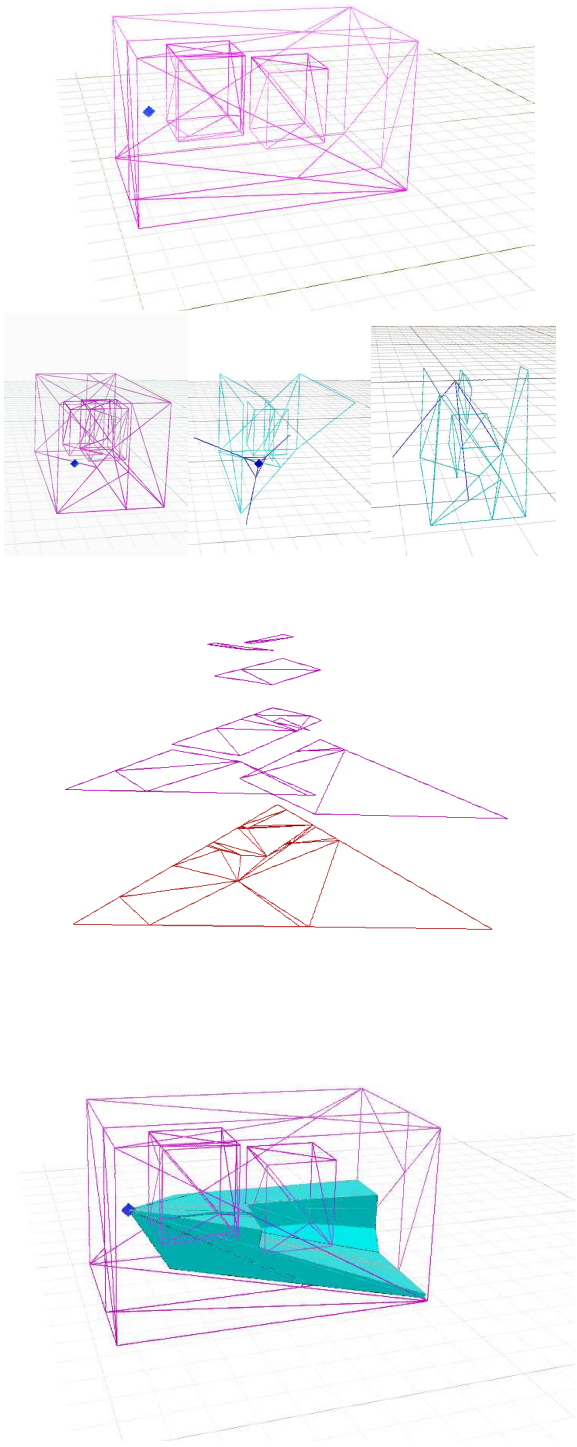


Fig. 4. Beam division process

2.3.3 Hiding and dividing algorithm

All triangles from *PT* (illuminated, transformed and projected) are clipped with the beam to process only that

part of triangles that are within the volume of the beam. The clipping is done in two steps: first the beam is transformed into the beam space, and the beam is then intersected with the projection plane, to obtain the section of the beam. The projected beam section (PBS) is a triangle. In the second step of the clipping process all triangles from *PT* are clipped with PBS. For each triangle from *PT* the intersection between the triangle and PBS has to be calculated. The intersection algorithm is designed with the special attention, wishing to achieve the accuracy and the performance. The algorithm is described in Appendix B. Clipped triangles are displayed in magenta, in the upper part of the third row of Fig. 4. Triangles that have higher order of the visibility (that are closer to the source) are displayed on the top. Triangles with the same order of the visibility are displayed coplanar.

In the next phase clipped triangles from *PT* are checked for the occlusion and divided accordingly. The result of this process is displayed in the bottom of the third row of Fig. 5, with red triangles.

These are data structures and algorithm for hiding & dividing process:

```
  PT - BST of projected triangles sorted in
order of visibility
  HT - BST of hidden, processed triangles
sorted in order of visibility
  CP - BST of pairs of triangles already
checked for occlusion

  determine the order O of first triangle in PT
  move all triangles from PT that have order O
   to the binary sorted tree of hidden triangles
   HT
  for each triangle t from PT
     for each triangle hT from HT
        if (t has not the same order as hT)
        and (t is not a neighbor of hT)
        and (t and hT are not in CP)
           add pair of (t, hT) to CP
           if hT obscures t
              subtract hT from t
              add all resulting triangles to PT
              add all resulting triangles (with
               parent information) to CP
              delete t from PT
              restart the loop
     move t from PT to HT
```

The algorithm traverses *PT*, in the order of the visibility. In the first step all foremost triangles from *PT* with the same order of the visibility are transferred to *HT*, because they cannot possibly occlude each other, and further checking is not necessary. After that, every triangle *t* from *PT* is checked for the occlusion with triangles (already processed) from *HT*. If the triangle *t* is not occluded by any triangle *hT* from *HT*, it is moved from *PT* to *HT*. Otherwise, if the triangle *hT* occludes *t*, they are subtracted (see Appendix A for details of the optimized subtraction procedure). Triangles that are result of the subtraction are added to *PT* and after that loops are restarted.

In the case of the occlusion, the loop is restarted several times. In order not to check triangle pairs twice, the information of already checked triangle pairs is recorded in *CP*. In the case of subtraction, the information of already checked pairs of triangles is copied from the parent to child triangles.

To avoid the overhead, two more tests are performed: if *t* and *hT* have the same order of the visibility they cannot occlude each other, and the occlusion test is skipped. Also if they are neighbors, since we have already done the backface-culling, they also cannot occlude each other, and the occlusion test is skipped.

Figure 5. provides an example of the hiding & dividing algorithm. The detailed step-by-step explanation of this example is provided in Appendix B. In this example four occlusion tests are performed, and four clippings occur. The process of hiding & dividing starts with four triangles that obscure each other and it ends with five triangles that do not obscure each other. Resulting triangles cover the whole area of the beam section.

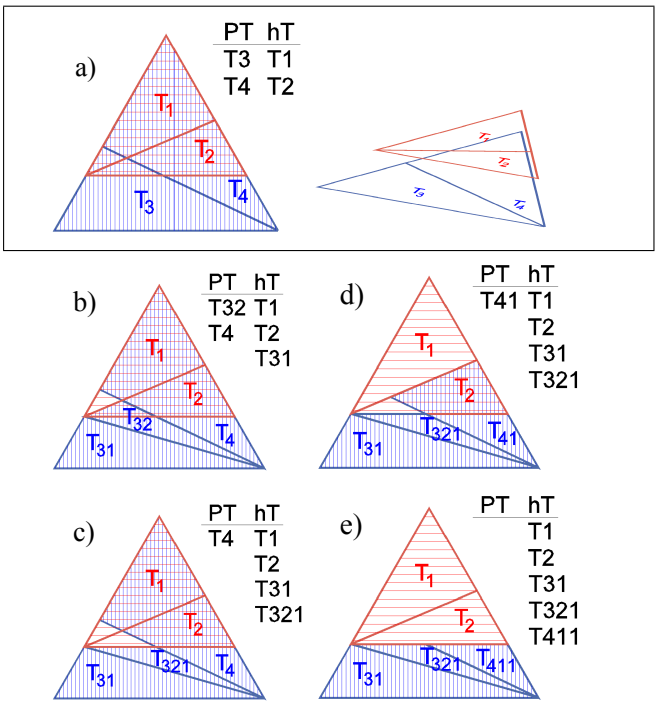


Fig. 5. Hiding and dividing algorithm example

2.3.4 Creation of divided beams

The process of the beam division in the BTR finishes with the stage of creating divided beams (Fig. 4, last row).

The initial beam, that is being divided, is removed from the queue, and new beams are generated using triangles from *HT*. Before new beams are formed, hidden triangles form *HT* are projected back from the beam space to the world space. Divided beams are then created using the same source point as the initial one, and with three edge rays that pass through three corners of the hidden triangle. Such divided beams are then closed with the plane of the triangle which was the parent (original) of the hidden triangle.

2.4 Physics of beam division

The BTR is designed to simulate the propagation of sound in non-convex spaces composed of different media. The BTR simulates two wave phenomena: the specular reflection and the refraction of sound.

The BTR traces beams with a triangular cross-section. When beams hit the boundary surface between two media, they are divided into smaller beams, to adjust to individual triangles that make the boundary surface. After the division, each divided beam is reflected or refracted, and such beams are further traced in the same manner. The tracing stops when the level of the intensity on the end of the beam is smaller than predefined threshold, or when the volume of the beam is smaller than predefined volume threshold.

The result of BTR simulation is the level of sound intensity at a single point or at points in a rectangular raster. Following the law of power averaging, the total sound intensity at the receiver is calculated by summing the individual intensities of each beam that contains the receiver. The individual beam that contains the receiver can be either a direct beam (coming directly from the source), or an indirect beam (that has been already reflected or refracted on the boundary surface between two media). The sound intensity at a point inside a direct beam is calculated using the following equation:

$$I = \frac{P_A}{4 \cdot \pi \cdot r^2} \cdot e^{-\gamma \cdot r}, \tag{1}$$

where P_A is the acoustical power of the source of the sound, r is the distance from the source and γ is the attenuation coefficient of the media within which the beam propagates. The first term in the equation describes the attenuation of the sound caused by the propagation of the spherical wave. The second term in the equation describes the attenuation caused by viscosity and other dissipative processes in the media [9].

When sound encounters the boundary surface that divides two media, the sound beam is reflected and refracted (Fig. 6). The sound intensity of such indirect beam is determined using the following equations:

$$I_{I'} = R^2 \cdot I_I, \tag{2}$$

$$I_{II} = \left(1 - R^2\right) \cdot I_I, \tag{3}$$

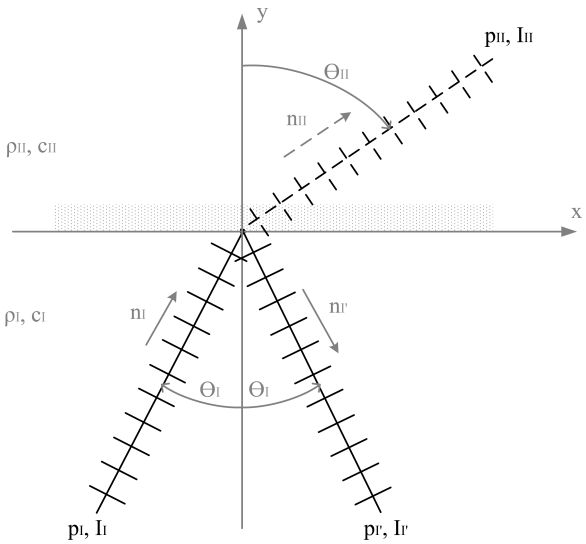


Fig. 6. The reflection and refraction of sound on a boundary surface

where I_I is the intensity of the incoming sound beam, $I_{I'}$ is the intensity of the reflected sound beam, I_{II} is the intensity of the refracted sound beam and R is the amplitude of sound reflection coefficient of the boundary [9, 14]. The reflection coefficient R is calculated using the following expression:

$$R = \frac{Z_{II} - Z_I}{Z_{II} + Z_I}, \tag{4}$$

where Z_I and Z_{II} are the acoustic impedances of the two media. The acoustic impedance is the function of the angle of incidence θ_I, as in [9].

In the BTR, the intensity of the sound at a point inside an indirect beam is calculated (Fig. 7) relative to the sound intensity at the barycenter of the starting triangle of the indirect beam (I_0) using the following equation:

$$I = I_0 \frac{r_1^2}{r_2^2} \cdot e^{-\gamma \cdot (r_2 - r_1)}, \tag{5}$$

where I_0 is the intensity of the sound at the barycenter of the starting triangle of the indirect beam, r_1 is the distance from the virtual source of the beam to the barycenter of the starting triangle of the beam, r_2 is the distance from the virtual source of the beam to the receiver and γ is the attenuation coefficient determined by the entity in which the beam propagates. I_0 is calculated as the intensity of sound of original beam, transformed with equation (2) for reflected, and equation (3) for refracted beam. It is stored in the data structure of the indirect beam, so all data required to calculate the intensity of sound is stored with the beam.

When a beam hits a boundary surface, the geometry of the beam is changed. The BTR generates one reflected and

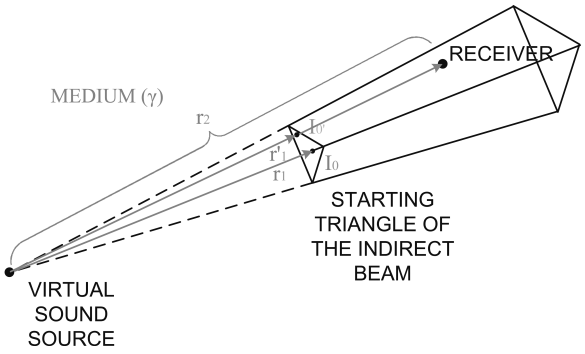


Fig. 7. Calculating the intensity of indirect sound in the BTR

one refracted beam for each incoming beam. Edge rays of reflected beam have the angle opposite to the angle of edge rays of the incoming beam – θ_I (Fig. 6). Edge rays of the refracted beam are generated according to Snell's law:

$$\frac{\sin(\theta_I)}{\sin(\theta_{II})} = \frac{c_I}{c_{II}}. \tag{6}$$

If the incoming angle θ_I is greater than the critical angle θ_{I-crit}:

$$\theta_{I-crit.} = \arcsin\left(\frac{c_I}{c_{II}}\right), \tag{7}$$

then only the reflected beam is generated.

Since the BTR works using the principles of the geometrical acoustics, for the simulation to work properly, dimensions of the geometry have to be equal or greater than the wavelength of sound. If this condition is not fulfilled, the principles of geometrical acoustics can't be used to calculate the wave propagation.

3 IMPLEMENTATION AND RESULTS

This section first describes the implementation of the method. Then, two kinds of results are presented - the first one is the measurement of the speed of the beam division process. The second one is the comparison of results and the speed of the complete BTR simulation with a commercial FEM simulation.

3.1 Software implementation

The BTR was coded in C++ language. The Standard template library (STL) was used for data containers, and all data variables were declared in double precision. Classes that represent the structure of the scene, geometric primitives and the BSP tree were created from the scratch, using published algorithms [2, 10, 7]. Special attention was given to the robustness of geometric operations. Every geometric operation was coded using the epsilon-error checking to avoid numerical problems. Both the relative and the absolute epsilon value were used when comparing equality and relation of two values. The double epsilon checking was done to provide the same accuracy over the whole range of double data type, and also to avoid problems with denormals.

User interface was programmed using the Microsoft foundation classes (MFC), and graphics was displayed with DirectX. DirectX and graphics card acceleration was not used for geometric calculations. Also neither multithreading, no SIMD instructions were used.

Tests were performed on a PC with Intel Core2Duo E8400 processor, with the clock of 3 GHz and 4 GB of RAM. The memory use during testing never exceeded 200 MB.

3.2 Measurements of the beam division process

The beam division was tested on three scenes, with the progressive complexity. Scenes are displayed in Fig. 8. The entity is displayed with magenta triangle mesh, traced beams are displayed in cyan with solid fill, and regions that are shadowed because of the occlusion are transparent. In upper left corner the index picture is presented with the geometry in magenta, and the source position in blue.

The first scene, named *s1*, is a simple room (24 triangles) with the v-shaped wall. This wall makes it a non-convex volume – the wall partially occludes other parts of the room. The second scene, named *s2*, differs from *s1* by an inner shell in the form of the sphere. The sphere occludes surrounding walls, and in combination with v-shaped wall causes multiple occlusions. This sphere has 224 triangles. The third scene, named *s3*, is similar to *s2*, but it has the second inner shell, in the shape of a box (12 triangles). The box is positioned between the source and the sphere, thus causing even more complex multiple occlusion of the scene geometry.

All three scenes have only one entity, but in the case of *s2* and *s3* the entity has multiple shells.

To determine how the total number of triangles in the scene influences the performance of the beam division, the scene geometry was tessellated several times. The geometry of *s1* was tessellated four times, resulting in versions with 24, 96, 384, 690 and 1536 triangles; *s2* was tessellated once, resulting in versions with 248 and 1280 triangles; *s3* was also tessellated once, resulting in versions with 260 and 1472 triangles. The tessellation was done in Autodesk 3D Studio VIZ 2008 software.

To determine how the number of initial beams influences the performance of the beam divisions, scenes were traced with various numbers of initial beams. They were

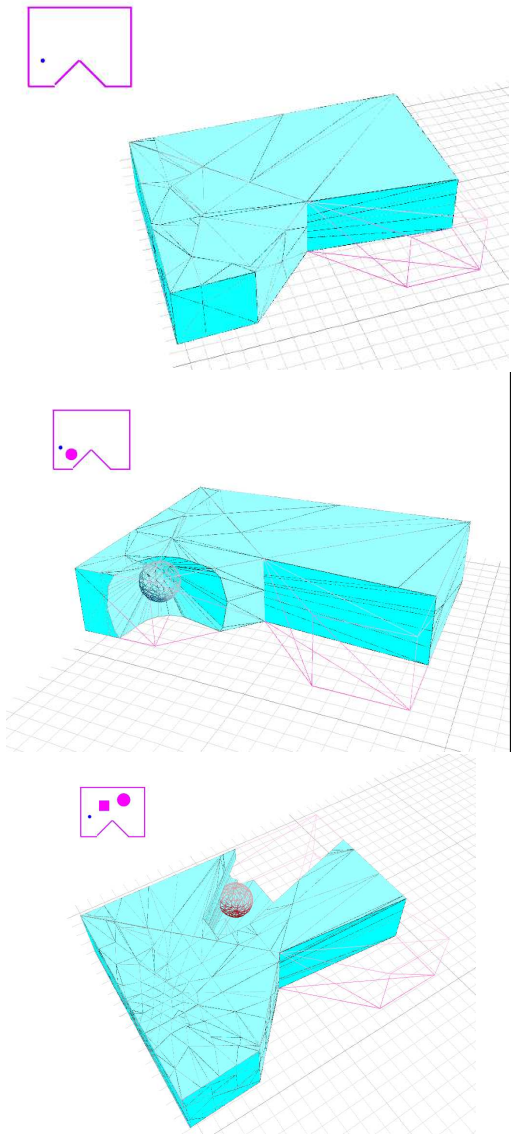


Fig. 8. Test scenes: s1 - top, s2 - middle, s3 - bottom

Table 1. Results for test scene s1

#triangles	# initial beams	wall clock time (s)	# divided beams	time per div. beam (ms)
24	20	0.01	227	0.03
	80	0.01	514	0.02
	320	0.03	1,628	0.02
	1,280	0.08	5,218	0.02
	5,120	0.28	18,309	0.02
96	20	0.02	409	0.06
	80	0.02	756	0.03
	320	0.04	2,097	0.02
	1,280	0.10	5,929	0.02
	5,120	0.35	20,013	0.02
384	20	0.20	850	0.23
	80	0.10	1,353	0.07
	320	0.13	3,073	0.04
	1,280	0.21	7,233	0.03
	5,120	0.62	23,151	0.03
690	20	0.44	1,258	0.35
	80	0.20	1,857	0.11
	320	0.17	3,792	0.04
	1,280	0.30	8,245	0.04
	5,120	0.86	24,526	0.03
1536	20	2.95	2,196	1.34
	80	0.88	3,126	0.28
	320	0.38	5,756	0.07
	1,280	0.60	10,874	0.06
	5,120	1.57	29,902	0.05

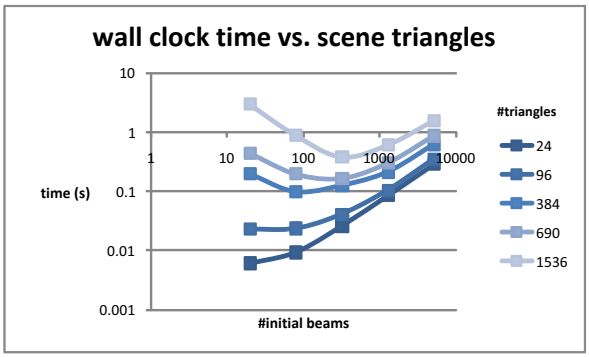


Fig. 9. Wall clock time vs. number of initial beams for different levels of geometry tessellation

first traced with 20 beams (resulting from the icosahedron), and subsequently with 80, 320, 1280 and 5120 initial beams (resulting from the tessellation of the icosahedron).

Let us first consider only the first test scene – *s1*, and the influence of the total number of triangles to the beam division process. The results of the test are in Table 1. Column 3 contains the wall clock time of the complete beam division, column 4 contains the number of beams after division, and column 5 contains the time needed to calculate one divided beam.

Figure 9. displays in logarithmic scale the wall clock time dependency on the number of initial beams for different numbers of initial triangles. For scenes with small number of triangles (24 and 96 triangles) the wall clock time increases continually when the number of initial beams is growing, so there is no reason to trace such a scene with the large number of initial beams. In other scenes, with more tessellated geometry the time changes in a different way. When the number of initial beams grows, the time first decreases, and then increases. For each level of tessellation there is the exact number of initial beams that gives the minimum time. This number is the optimum number of initial beams for the beam division of such a scene.

The optimum value exists because of the relationship between the geometry of beams and the geometry of the

entity. If there is only few initial beams, their section is bigger and they illuminate more triangles of the entity, thus causing more complex and time consuming division process. If the number of initial beams is increased, the size of the beam section decreases. So the beam illuminates less entity triangles, and the division process is simpler and faster. For the optimum beam division, the size of the beam section is near the size of the average entity triangle, so it illuminates only one entity triangle, thus achieving the simplest possible division process. If the number of initial beams is further increased, division process cannot be simpler and faster, because the beam cannot illuminate less than one entity triangle. In such case, the number of repeated division increases linearly with the number of initial beams, so the time also increases linearly.

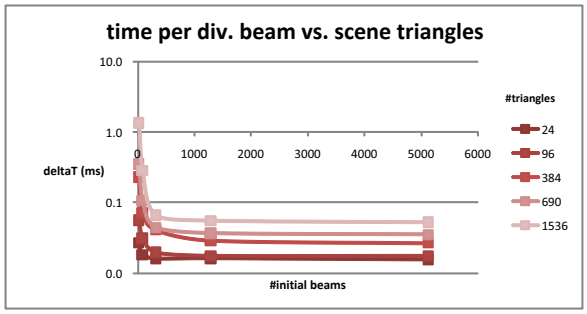


Fig. 10. Time per divided beam vs. number of triangles in scene, for different levels of geometry tessellation

The time required to calculate one divided beam is the wall clock time required for the whole process of beam division, divided by the number of divided beams. It increases with the scene complexity and decreases with the number of initial beams (Fig. 10), because with greater number of initial beams, every beam illuminates smaller number of geometry triangles, causing simpler and efficient beam division process. In the case of complex geometry, this decrease is drastic, so for the complex geometry it is clearly better to chose greater number of initial beams.

Finally, let us consider what happens if the number of initial beams is fixed (Fig. 11). For 320 initial beams, the total time of division increases linearly with the scene complexity. The number of beams that result from the beam division process also increases, but only logarithmically, so even complex scenes, with the large number of triangles, can be traced, without concern for the "explosion" of the number of beams.

Table 2. presents results of the beam division for test scenes *s2* and *s3*. Both scenes were tested in two versions: the first one with smaller and second one (tessellated) with greater number of triangles. As for the scene *s1*, the optimal number of initial beams exists for more complex

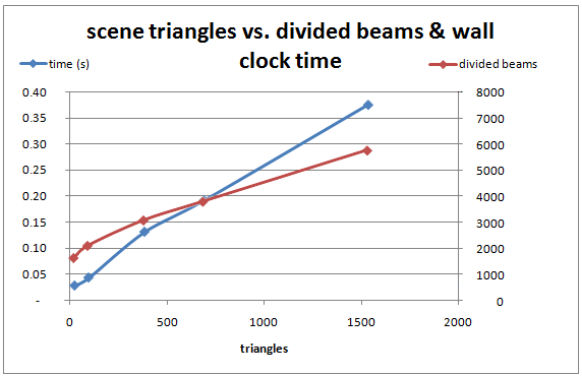


Fig. 11. Wall clock time and number of divided beams vs. number of triangles, for fixed number of initial beams (320)

Table 2. Results for s2 and s3

scene	#triangles	# initial beams	wall clock time (s)	# divided beams	time per div. beam (ms)
s2	248	20	0.1	374	0.30
vroom		80	0.1	802	0.11
+		320	0.1	1990	0.05
sphere		5120	0.8	19366	0.04
	1280	20	2.4	1163	2.07
		80	0.9	1791	0.52
		320	0.6	3690	0.16
		1280	0.9	9009	0.09
		5120	2.6	24648	0.11
s3	260	20	0.1	344	0.22
vroom		80	0.1	719	0.17
+		320	0.1	1808	0.06
sphere		1280	0.3	5629	0.06
+		5120	0.9	18730	0.05
box	1472	20	3.8	1171	3.23
		80	3.1	1793	1.73
		320	1.8	3516	0.50
		1280	2.0	8334	0.24
		5120	3.3	23981	0.14

scenes. For the simple geometry, lower number of initial beams is more suitable.

Figure 12. shows how the total time of the beam division depends on the number of triangles in different scenes, when the number of initial beams is fixed to 320. For scene *s1*, the time rises linearly with the number of triangles in the scene. In the case of scenes *s2* and *s3*, where multiple occlusions exist, the angle of increase is steeper.

Let us compare results presented here with results of some implementations of the acoustical ray and beam tracing methods published recently. We have to stress that the direct comparison cannot be made, because of several reasons. The first reason is that the geometry of scenes of compared simulations was not the same. The second reason is that our implementation doesn't use hardware accel-

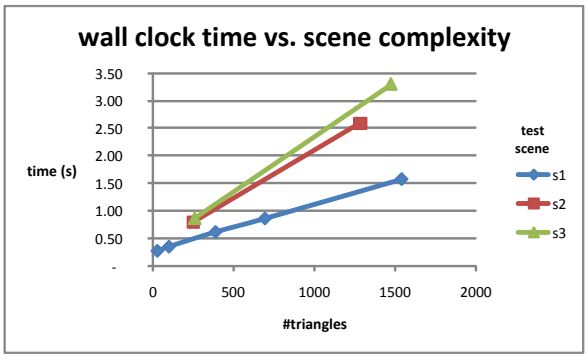


Fig. 12. Time dependency for fixed number of initial beams

eration, and it is optimized for propagation through several media, which in compared simulations is not the case. Also, tests were not performed on same computers, although all reports are fairly recent. Considering all mentioned, it is reasonable to say that this comparison could give an order of magnitude relation.

Feistel, Ahnert et al. report that classical linear ray tracing algorithm (EASE) for geometry such as *s1* (consisting of approx. 1,500 triangles) performs in about 70 seconds [4]. This paper presents a novel version of ray tracing, that is fully optimized for SIMD and graphics accelerator, and that uses advanced space division structures. This advanced ray tracing is reported to compute such geometry in 15 seconds. Our best result of comparable geometry is 0.38 seconds (Table 1). Since our algorithm uses the beam tracing, it is also more accurate, because the space coherence of the beam tracing eliminates aliasing errors to which the ray-tracing method is prune.

The interactive sound beam tracer from Funkhouser et al. [5] gives results for tracing the "Floor" scene. This scene represents several rooms that form one floor of the building (with 1,772 polygons). The tracing resulted in 18,239 beams, and the calculation was done in 2.1 seconds. Against such geometry we can compare our *s1* scene with 1,536 triangles. Our tracing resulted in 10,874 beams, and was completed in 0.38 seconds. We have to stress that in our case all 1,536 scene triangles had to be considered for division. On the other hand, compared "Floor" scene was subdivided in 814 cells, so only few polygons, that are visible inside one cell, had to be considered for the division of beams. In our simulation, since the simulation was done in one entity, all triangles had to be considered for the division of beams.

3.3 Comparison of the BTR and the FEM simulation of the ultrasound propagation

Tests presented in the previous subsection have checked the speed of the beam division. However, the beam division is only a part of the BTR. To check the whole BTR method the simulation was performed of a scene with an acoustic lens (Fig. 13). The scene was designed so that lens caused the refraction of ultrasound. This refraction resulted in the focusing of ultrasound inside the lens. If the BTR simulation produced the focusing of the ultrasound, that would mean that the simulation of the refraction in the BTR works. Furthermore, to check the accuracy of the BTR simulation and its performance the same scene was simulated with the well established FEM simulation, whose results were compared with the BTR.

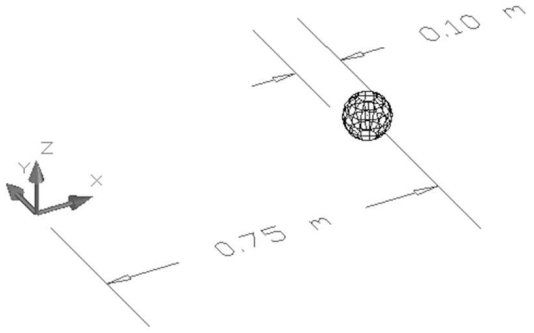


Fig. 13. The scene used to test refraction in the BTR

The acoustic lens scene consisted of a unidirectional sound source ($f = 100$ kHz; $P = 100$ W) emitting ultrasound into a space filled with glycerol ($c = 1920$ m/s; $\rho = 1260$ kg/m^3; $\gamma = 3 \cdot 10^{-6}$ m^{-1}). In the glycerol, there was an entity made of rubber ($c = 900$ m/s; $\rho = 930$ kg/m^3; $\gamma = 43 \cdot 10^{-3}$ m^{-1}). The rubber entity was a sphere with a diameter of 0.1 m, centered 0.75 m from the sound source (located at the origin of the coordinate system in Fig. 13).

The acoustic lens scene was simulated with the BTR, and with the well-established FEM simulation [17].

The BTR scene structure had two entities: the outer, non-convex entity filled with glycerol, and the inner, convex entity filled with rubber. Both entities had only one shell, in the form of a sphere. The geometry of the sphere consisted of 224 equilateral triangles. The BTR simulation traced direct sound and reflections/refractions up to 4$^{\text{th}}$ order.

Because simulating this scene with the FEM in full 3D would be too computationally complex to perform on a desktop computer, the FEM simulation was performed in 2.5D by utilizing the rotational symmetry of the acoustic lens scene. The rotational axis is the line from the source of the sound (the origin of coordinate system) through the center of the rubber sphere (Fig. 14). In this 2.5D setup, one finite element had dimensions of 0.36 x 0.36 mm, for a total of 2,222 x 833 = 1,850,926 elements. The simulation

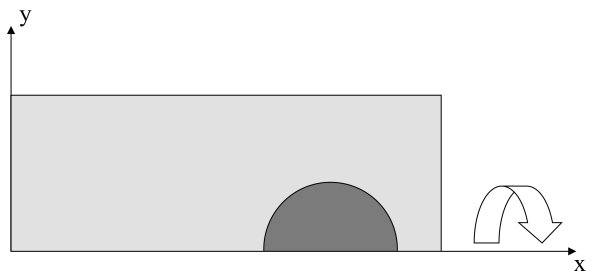
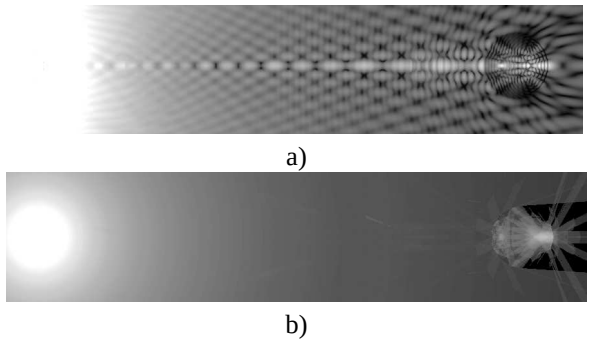


Fig. 14. 2.5D FEM simulation model - rotation around the x-axis

was done using a finite-element and explicit time-domain approach.

The results of the FEM and the BTR simulations are shown in Fig. 15. The FEM calculates the pressure of the sound taking the phase into account, so its results exhibit the effects of sound wave interference, which can be seen in the line pattern in Fig. 15a. The BTR calculates the average intensity of sound, so there are no lines that represent the interference of the sound in Fig. 15b. The primary focus of the sound can be seen in the back half of the lens in both images. The position and the maximum intensity of the primary focus matches well in both simulations. The difference between locations of the maximum is 0.7 mm, and the difference between sound intensity levels is 1 dB.

The BTR creates a clear shadow region behind the lens, while the FEM simulation does not show a distinct shadow behind the lens. This discrepancy arises because the diffraction of sound is still not implemented in the BTR, while the FEM calculates diffraction as well as other wave phenomena.



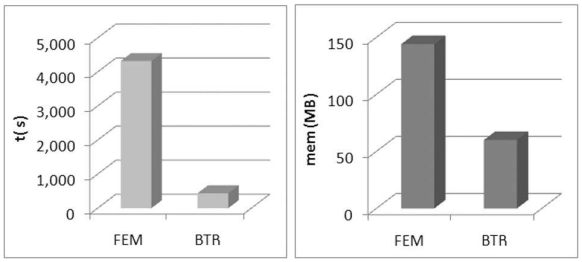
a)

b)

Fig. 15. The simulation of the acoustic lens scene with the FEM (a) and the BTR (b)

Let us consider the performance of the FEM and the BTR simulations. Both simulations were executed on the same hardware platform, which consisted of an Intel Core2Duo processor with a frequency of 2.4 GHz and 4 GB RAM.

Figure 16a shows that the BTR simulation executed approximately 10 times faster than the FEM simulation. The BTR also used less than half of the memory that the FEM used (Fig. 16b). Also the FEM could not perform this simulation in full 3D, but only in 2.5D. In addition, if the frequency of sound was higher, the performance of the BTR would stay the same, while the performance of the FEM would decrease. Because of the required wavelength/finite element size ratio, the number of finite elements would have to be increased, and the performance of the FEM would decrease significantly.



a) b)

Fig. 16. The simulation of a scene with an acoustic lens with the FEM (a) and the BTR (b)

4 CONCLUSION AND FUTURE WORK

This paper presents the process of the beam division implemented in the BTR simulation. The BTR is designed for the acoustical or the hydro-acoustical simulation of non-homogenous environments, where the sound wave propagates through several different media. The implementation of the BTR beam division, described in this paper, is designed for complex, non-convex geometry, composed of irregular triangle mesh boundaries.

In this paper, the detailed explanation of the beam division process is given, as well as results of tests the beam division for three scenes. Results show that for every scene there is the ideal number of initial beams that gives the best result. For scenes with less occlusion, the total tracing time was few seconds, even when the number of triangles in the scene was large. The most complex scene was traced in less than forty seconds. The total tracing time linearly grows with the number of triangles in the scene, and even for the most complex geometry the number of divided beams doesn't explode. Measured results compare favorably with published results of other methods and other implementations of the same method.

To check the complete process of the BTR, the simulation was tested on the non-homogenous scene with acoustic lens, containing two media. The results and performance of the BTR were compared with results of the commercial FEM simulation. This comparison showed that the

BTR is a good choice for situations in which the dimensions of the scene are large compared with the wavelength of ultrasound. In such cases, the calculation using numerical methods (FEM) is computationally intensive because the element of the mesh has to be small enough to satisfy minimum wavelength ratio requirements. In such cases, if the interference and diffraction of sound do not have to be simulated, the BTR shows the advantage over the FEM by providing the excellent performance and the acceptable accuracy.

While the accuracy and the performance of the BTR were found satisfying, they still leave the space for the improvement. To improve the performance, the simulation code is currently being rewritten for multicore processors, since they are now a common thing. Beside this, other advanced space structures are considered to be implemented in order to enhance the performance. Finally, in order to further estimate the ability of the BTR to simulate non-homogenous environments, a comparison with the real-world measurement is planned.

APPENDIX A TRIANGLE-TRIANGLE BOOLEAN OPERATIONS

Beams in the BTR have the triangular section, and the geometry of the scene is represented by triangular meshes, so regularized triangle-triangle Boolean geometric operations are of the special importance. Two of them are used in the process of the beam division: the first is the intersection of triangles *A* and *B*. It is performed during the clipping of illuminated triangles with the section of the beam (illustrated in Fig. 17 with light green polygons). The second one is the subtraction of the triangle *B* from the triangle *A*. It is performed during the hiding and dividing of triangles (Fig. 17 – light red polygons).

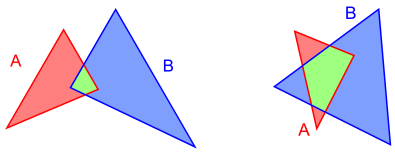


Fig. 17. Trivial triangle combinations

Figure 17. displays two trivial combinations of triangles. Beside these trivial combinations there exists boundary combinations that also have to be processed (Fig. 18). In the boundary combination two triangles have at least one collinear edge, and / or at least one collocated vertex.

For regularized triangle-triangle Boolean geometric operations other simulation use the Cohen-Sutherland polygon clipping algorithm [11]. In order to enhance the performance, instead of using the Cohen-Sutherland polygon clipping algorithm in the BTR a proprietary algorithm was developed. In this algorithm each case of triangle combinations is processed in the separate branch of code - clearly in expense to the length of the code.

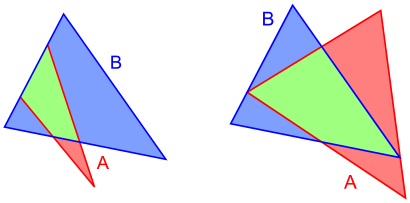


Fig. 18. Boundary triangle combinations

There exist 15 trivial triangle combinations, 100 boundary combinations with at least one collinear edge, and 143 boundary combinations where at least one vertex is collocated with an edge or a vertex of the other triangle. The algorithm is optimized for the speed, by using as little geometric operations as possible, and by treating separately every trivial and boundary case. The detailed explanation of the algorithm is outside the scope of this paper, so only short description will be given.

The BTR, the algorithm first checks if the combination of two triangles is the collinear, the collocated or the trivial one. Next, indexes of vertices and edges of triangles are rearranged, so that collinear or collocated vertices are moved to the front. After this, with few geometric tests the algorithm determines which triangle combination is being processed. Subsequently, a set of triangles that results from desired Boolean operation is created. In the case that the Boolean operation results with a polygon, the Delaunay triangulation of the polygon is performed.

APPENDIX B EXAMPLE OF BEAM DIVISION IN DETAIL

Fig. 5a shows one example of the triangle setup for the process of hiding, both in the perspective, and in the top view. Triangles *T1* and *T2* are in the front, with the order of the visibility 1. They only partially fill the section of the beam, and occlude remaining triangles. In the back there are two triangles *T3* and *T4*, with order of visibility 2. They fill the complete section of the beam and represent the occluded part of the mesh. At the start of the algorithm all triangles are in *PT*.

The algorithm starts with transferring *T1* and *T2* to *HT*, since they have the same order of visibility. Next *T3* (which is in *PT*) is checked against *T1*. Because *T1* partially occludes *T3*, *T3* is clipped. The clipping results with triangles *T31* and *T32* as shown in Fig. 5b.

The loop is restarted, but because of the information in *CP* and neighbor checking, all tests are skipped except for the testing of *T31* and *T2*. Since there is no occlusion, *T31* is moved to *HT*. *T32* is then checked against *T2*, and clipped to *T321*. The loop is restarted, but because of *CP* and neighbor filters, no checks have to be performed, and *T321* is finally moved to *HT* (Fig. 5c). T4 is then clipped with *T1*, resulting in *T41* as shown in Fig. 5d. Finally *T41* is clipped with *T2* resulting in *T411*. The final set of triangles is shown in Fig. 5e.

We tried to find an alternative to this highly iterative process, to optimize the performance. One solution has received a special attention: triangles are first triangulated with Delaunay triangulation [2], and after that, for every resulting triangle, it is be determined, which foremost original triangle it is part of, to be able to close the divided beam correctly afterwards. Since Delaunay triangulation is a well optimized technique, the first stage of this process would be very fast. But the second stage is an iterative process which slows down the algorithm. The BTR hiding algorithm in the same pass performs both stages – it divides triangles and determines the closing plane of the beam. The second drawback of the alternative method is that it results in higher number of divided beams.

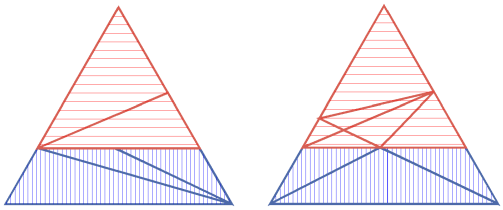


Fig. 19. Comparison of two algorithms

Figure 19. shows the comparison of the method implemented in the BTR (left) and the alternative method with Delaunay triangulation (right). The BTR method gives better result – it produces fewer divided triangles. Our method results in 5 divided triangles and alternative method in 9 triangles.

REFERENCES

[1] I. Drumm, "The adaptive beam tracing algorithm" in *Journal of the Acoustical Society of America*, vol. 107, pp. 1405 - 1412., 2000.

[2] M. de Berg, M. Van Kreveld, M. Overmars, O. Schwarkopf, *Computational Geometry – Algorithms and applications.* Berlin, Germany: Springer-Verlag, 2000.

[3] A. Farina, "Validation of the pyramid tracing algorithm for sound propagation outdoors: comparison with experimental measurements and with the ISO/DIS 9613 standards", *Advances in Engineering Software*, Elsevier Applied Science, vol. 31, pp. 241 - 250, 2000.

[4] S. Feistel, W. Ahnert, A. Miron, O. Schmitz, "Improved methods for calculating room impulse response with EASE 4.2 AURA", *Presented at 19th International congress on Acoustics*, Madrid, Spain, 2007.

[5] T. Fukhouser, N. Tsingos, P. Min, "A Beam Tracing Method for Interactive Architectural Acoustics", *Journal of the Acoustical Society of America*, vol. 115, pp. 739 - 756, 2004.

[6] P. S. Heckbert, P. Hanrahan, "Beam Tracing Polygonal Objects", *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1984.

[7] J. O'Rourke, *Computational Geometry in C.* Cambridge, UK: Cambridge University Press, 1997.

[8] R. Overback, R. Ramamoorthi, W.R. Mark A, "Real-time Beam Tracer with Application to Exact Soft Shadows", Proceedings of the Eurographics Symposium on Rendering, Grenoble, France, 2007.

[9] Pierce A.D.: Acoustics - An Introduction to its Physical Principles and Applications, McGraw-Hill (1981)

[10] P. J. Schneider, D. H. Eberly, *Geometric tools for computer graphics*, Morgan Kaufmann Publishers - Elsevier, San Francisco, 2003.

[11] I. E. Sutherland, G. W. Hodgman, "Reentrant polygon clipping", *Communications of the ACM*, vol. 17, pp. 32-42, 1974.

[12] M. Sikora, "Beam-tracing method in non homogenous environments", *Proceedings of the 2nd congress of Alps-adria acoustic society*, Opatija, Croatia, 2005.

[13] A. Schmitzy, T. Rick, T. Karolski, T. Kuhlen and L. Kobbelt, "Simulation of Radio-Wave Propagation by Beam Tracing", *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization*, Munchen, Germany, 2009.

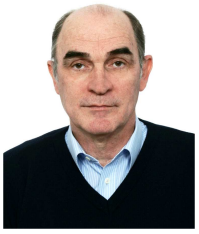
[14] Vorlander M.: Auralization, Springer (2008)

[15] J.P. Walsh, N. Dadoun, D.G. Kirkpatrick, "The geometry of beam tracing", *Proceedings of the 1st annual symposium on Computational geometry*, Baltimore, MD, USA, 1985.

[16] G. L. Wojcik, J. C. Mould Jr., L. M. Carcione, "Combined Transducer and Nonlinear Tissue Propagation Simulations", *Proceedings of the International Mechanical Engineering Congress & Exposition*, Nashville, TE, USA, 1999.

[17] G.L. Wojcik, D.K. Vaughan, N. Abboud, J. Mould, Jr.: Finite Element Modeling for Ultrasonic Transducers, Proceedings of Ultrasonic transducer engineering Conference, Vol. 3341 (1998), pp. 19 - 42

Marjan Sikora received his Electronics B.Sc. degree in 1995. and M.Sc. degree in 2000. from University of Zagreb, Faculty of electrical engineering and computing. In 2010., he obtained a PhD in Computer Science also from University of Zagreb, Faculty of electrical engineering and computing. He was first employed in ENTER d.o.o. Split, (1996 – 2004) as a GIS manager. He has then founded his own company „SIKORA“ (2004 – 2009), whose field of work was the IT with emphasis on the GIS. Since 2006, he is working as a teaching assistant at University of Split, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture. His research interests include geometric acoustical simulation methods and GIS.



Ivo Mateljan was born in Kastel Luksic, Croatia, in 1953. He received the B.S. degree in electrical engineering from University of Split, Croatia, in 1977, and M.S. degree and Ph.D. degree in technical sciences from University of Zagreb, Croatia, in 1983 and 1993, respectively. In 1995 he received a position of assistant professor, and in 2006 he received position of professor of electrical engineering at the the Faculty of eletrical engineering, University of Split, Croatia, where he is currently the head of the Electroacoustic deparment. His research interests include acoustics, signal processing and digital instrumentation. He is owner of firm Artalabs that develops software for virtual PC based instrumentation.

Nikola Bogunović received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Zagreb, Croatia, and his Ph.D. degree in computer science from the same university in 1963, 1971, and 1984 respectively. From 1968 to 1999., he was with R. Boskovic Institute, Zagreb, Croatia, directing the Division of Electronics, and Laboratory for Information Systems. From 1971 to 1972 he held the position of visiting scientist at Culham Laboratory, United Kingdom, and from 1985 to 1987 the position of visiting professor at Vanderbilt University, U.S.A. In 2000, he joined the Faculty of electrical Engineering and Computing, University of Zagreb, as full professor teaching courses on complex system design, digital system design, formal methods, knowledge representation paradigms and expert systems techniques. His current interests are focused on the application of formal methods in the design of computer systems and automated knowledge discovery in data and signal repositories, primarily in biomedical and financial domains. He is a senior member of IEEE, professional member of ACM and full member of the Croatian Academy of Engineering.

AUTHORS' ADDRESSES

Marjan Sikora, Ph.D.
Prof. Ivo Mateljan, Ph.D.
Department of Electronics,
Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture,
University of Split,
Ruđera Boškovića 32, HR-21000, Split, Croatia
email: sikora@fesb.hr, ivo.mateljan@fesb.hr

Prof. Nikola Bogunović, Ph.D.
Department of Electronics, Microelectronics, Computer and Intelligent Systems,
Faculty of Electrical Engineering and Computing,
University of Zagreb,
Unska 3, HR-10000, Zagreb, Croatia
email: nikolabo@zemris.fer.hr

Received: 2010-12-03
Accepted: 2011-12-05