Computer Graphics in 3D WaveFront Construction

Lucas Freitas, Jorge Stolfi and Martin Tygel – UNICAMP, Brazil

Abstract

We describe an algorithm for simulating the propagation of a seismic wawefront, modeled by a mesh of triangles, through a general three-dimensional geophysical model, with automatic control of sampling density over the expanding front. We show that wavefront modeling is considerably simplified by using the Quad-edge mesh data structure. We also describe a control mechanism to eliminate samples when the front contracts.

1 Introduction

Simulation of a seismic wavefront propagating through a general three-dimensional geophysical model has several applications in geophysics and petroleum engineering, such as the validation of seismic inversion software. The wavefront must be represented by some kind of mesh, whose vertices move with time. Since the front may expand (and sometimes contract) a lot during the simulation, the simulator must insert or delete vertices as needed, in order to maintain a fairly uniform mesh density at all times.

The code that handles the changes in the mesh topology is usually the most complex, expensive, and error-prone part of the simulator. We show here that a particular representation of two-dimensional meshes, the *Quad-edge* data structure, leads to considerable simplifications of the mesh maintenance code.

The principles of wavefront-based seismic simulation are presented in section 2. In section 3 we describe the Quad-edge data structure and its basic topological operators. The WFC algorithm, including the vertex density control mechanism, is detailed in secton 4, in terms of the Quad-edge operators. Section 5 shows some examples generated with a preliminary implementation of this method (still without density control).

2 The WaveFront Construction method

The classical wavefront-based method is a graphical procedure based on *Huygens's principle* (7): the wavefront W_{t+h} for a traveltime $t + h$ is calculated from a previous one, W_t, as the enveloping surface of spheres with their centers distributed along W_t and with radii hv, where v is the local speed of propagation at the center of the sphere.

This method assumes that the geophysical model is locally homogenous over each sphere, and is practical only for isotropic structures with two-dimensional variation—not for the general three-dimensional models which are most needed in practical applications.

2.1 Sample nodes and the wavefront mesh

The *WaveFront Construction* (WFC) method of Vinje et al. (8) can be thought as a hybrid of seismic ray tracing and a computer implementation of the classical sphere-envelope method. It uses a discrete representation of the wavefront, consisting of a set of records (*nodes*), representing sample points on the moving wavefront. Each node s contains the coordinates $s.c$ of the corresponding sample point, its velocity vector $s.v$ (which specifies the local speed and direction of propagation), its oscillation mode $s.m$ (see section 2.4), and possibly other local properties of the wavefront, such as intensity and polarization. Finally, the node contains a count $s.k$ of interactions with reflectors already simulated along the ray traced by the point $s.c$.

In many situations, the set of sample points is not enough: one must also link neighboring nodes in some way, e.g. so as to form a triangular mesh (9; 10). Uses of such a mesh include visualization of the wavefront, and detecting its arrival at a specified point (3).

Given a set N_t of wavefront nodes belonging to W_t, the set N_{t+h} of the next wavefront (W_{t+h}) is obtained by *initial-value ray tracing* from each node of N_t. See figure 1.

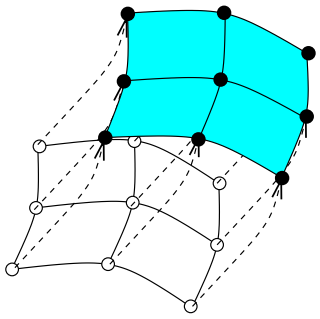


Figure 1: Basic step of the WFC method: The initial wavefront W_t and its nodes N_t (white dots), node propagation (dashed arrows), the new nodes N_{t+h} (black dots) and the new wavefront W_{t+h}.

The procedure described above is merely a parallel version of ordinary ray tracing. A distinguishing feature of WFC is the introduction of a *node density control* mechanism in order to ensure a sufficiently dense set of sample points

throughout the simulation. As described by Červený (7), whenever the nodes in some part of the wavefront fail to satisfy some local density criterion, a new node is created. Its initial conditions are obtained from the neighboring nodes in some sort of interpolation.

Unlike the sphere-envelope method, the WFC approach does not assume local homogeneity. Within a single time step, each ray may follow a curved path, or even hit the interface between very different media, where it may be reflected or refracted. See figure 7. As long as the time step and minimum sample density are appropriately chosen, neighboring nodes of N_t remain close to each other in N_{t+h}, so that the mesh structure is mostly unaffected by such interactions.

2.2 Geophysical model

We assume a geophysical model consisting of a finite number of *layers* where the relevant parameters of the medium (such as density and wave velocity) change smoothly with position. Layers are separated by surfaces where the parameters may change abruptly. These surfaces are customarily called *reflectors*, even when they reflect only a fraction (or none) of incident seismic signals.

2.3 Ray modes and mode conversion

Seismic signals propagating through isotropic media must be decomposed into two oscillation modes, pressure (P) and shear (S) waves, because each mode usually propagates at a different speed (7). A ray of either mode will follow a smooth path within a single layer. When it hits a reflector, the ray will usually split into four distinct secondary rays: either P or S, each either reflected (r) back into the same layer, or transmitted (t) into the next one with some refraction (7). These four rays have the same starting point but may have different directions and propagation speeds.

For homogeneous but anisotropic media, one must also separate the S mode into two principal polarized components, each with its own propagation speed; so the number of secondary rays increases to six. Whereas a ray in a homogeneous medium (isotropic or not) propagates along a straight line, in a smooth but non-homogeneous medium it follows a curved path defined by a differential equation (7). We will not discuss these model extensions here, since their implementation is relatively straightforward and independent of mesh representation and maintenance—which are the focus of this paper.

2.4 Ray signatures

In seismic ray tracing, one often wants to study only rays that have had a specific history of interactions with the reflectors: say, those rays that started in P mode, continued in S mode after crossing the the topmost reflector R_1, were reflected at the next reflector R_2, still as S, then transmitted again through R_1 and converted back to P. This history can be encoded by the string $\langle$P, 1tS, 2rS, 1tP$\rangle$, which is called the *signature* of the rays' final points.

In the same way, a wavefront of either S or P mode will usually split into four secondary wavefronts as it hits a reflector. The comments above, and the same encoding by signature, apply to wavefronts as well as to rays.

3 The Quad-edge data structure

A triangular mesh is a special case of a *two-dimensional subdivision*: a partition of an arbitrary surface into simpler parts — *vertices* (points), *edges* (arcs, straight or curved) and *faces* (polygons, flat or curved). The adjacency relations between these *elements* comprises the *topology* of the mesh; whereas the vertex coordinates and the shape of edges and faces constitute its *geometry*.

3.1 Mesh structures

In 1975, Baumgart observed that the topology of a subdivision is completely specified by the adjacencies of each edge, namely the two terminating vertices, the two adjacent faces, and the next edges around those vertices and faces (1). Indeed, the topology of a subdivision is determined by the *graph* (2) of its edges and vertices, plus the order of edges around vertices and faces. He implemented this insight in the *Winged Edge* data structure (1), where each edge is represented by a record with pointers to four adjacent edges.

After Baumgart's work, several other data structures have been proposed for the same goal. Here we consider the *Quad-edge* of Guibas and Stolfi (4). Its main advantages are its simplicity, and its ability to represent not only the subdivision but also its dual. The Quad-edge structure is abstractly defined by a set of *motion functions*, that allow "walking" from an element to the adjacent elements; and a pair of *topological operators*, that modify the data structure and hence the mesh topology.

3.2 Arcs and motion functions

The fundamental concept of the structure is the *arc* or *oriented edge*: an edge of the subdivision, taken with a specific orientation and seen from a specific side of the surface. Thus any arc e has a well-defined *origin vertex*, $e\,Org$, a *destination vertex*, $e\,Dest$, a *left face*, $e\,Left$ and a right face, $e\,Right$. The *symmetric* of e, $e\,Sym$, is the same undirected edge taken in the opposite direction.

The orientation of arcs also allows us to define the *ring of arcs of* a vertex v, as being the (circular) list of all arcs with $e\,Org = v$, ordered so that the left face of any of those arcs is the right face of the next arc in that list. The *next arc with same origin*, $e\,ONext$, is the arc that follows e in this ring. The *next arc with same left face*, $e\,LNext$, is obtained following boundary of the face $F = e\,Left$ in the counterclockwise direction.

The *dual* of a subdivision is another subdivision of the same surface, obtained by choosing a *dual vertex* inside every face, and connecting two dual vertices by a *dual edge* whenever the corresponding faces are adjacent. By definition, the Quad-edge represents simultaneously and uniformly both the subdivision and its dual. The *rotated* version of an arc e, denoted $e\,Rot$, is an arc of the dual edge

that crosses e, directed from $e\,Right$ to $e\,Left$ and oriented so that the ring of arcs around the face $e\,Right$ corresponds to the ring of arcs out of the dual vertex $e\,Rot\,Org$.

Here we assume that the surface is orientable, and that all arcs are seen from the same side of it. For subdivisions on non-orientable surfaces, we need another motion function, $e\,Flip$, that looks at the same directed edge from the other side of the surface. See the original article (4) for details.

3.3 The data structure

The Quad-edge data structure is a collection of *edge records* connected by pointers that encode the motion functions. Each edge record `e` represents the four arcs — two primal and two dual — associated with the same undirected edge e of the subdivision. The group comprises some arc $\bar{e}$ (the *canonical representative*) of e, and its rotated versions $\bar{e}\,Rot^r$, for $r \in 1, 2, 3$. Accordingly, the edge record `e` is divided into four *arc records* `e`$[0]$ through `e`$[3]$, where `e`$[r]$ represents the arc $\bar{e}\,Rot^r$. The pair $(\texttt{e}, r)$ is called the *reference* of arc $\bar{e}\,Rot^r$.

Each arc record `e`$[r]$ contains two fields, `Next` and `Data`. The former contains the reference to the arc $\bar{e}\,Rot^r\,ONext$ while the latter can be used to hold mesh geometry data and/or other application-specific information associated with the arc $\bar{e}\,Rot^r$. See figure 2.

Figure 2 shows that each edge record `e` belongs to four circular lists, defined by the `Next` links: two rings around the connected vertices and two around the separated faces. Given an arc reference $(\texttt{e}, r)$, the motion functions can be computed as follows:

$$\begin{array}{rlcl} & (\texttt{e}, r)\ Rot & = & (\texttt{e}, r+1) \\ & (\texttt{e}, r)\ ONext & = & \texttt{e}[r].\texttt{Next} \\ [!h] & (\texttt{e}, r)\ Sym & = & (\texttt{e}, r+2) \\ & (\texttt{e}, r)\ Rot^{-1} & = & (\texttt{e}, r+3) \\ & (\texttt{e}, r)\ OPrev & = & (\texttt{e}[r+1].\texttt{Next})\ Rot \end{array} \tag{1}$$

The basic Quad-edge structure does not have records explicitly representing vertices or faces. A vertex is implicitily defined as a ring of arcs, and it is referred by one of its outgoing arcs; and similarly for faces. In many applications, however, the structure is augmented with vertex and face records, whose addresses are stored in the `Data` fields of primal and dual arc records, respectively.

3.4 Basic topological operators

Another feature of the Quad-edge structure is that construction and modifications of arbitrary meshes can be reduced to just two basic topological operators—whereas most other mesh structures need half a dozen or more.

The `MakeEdge` operator takes no parameters and returns the reference e to the canonical arc of newly created edge record `e`. The `Next` fields of `e` are such that $e\,LNext = e\,RNext = e\,Sym$ and $e\,ONext = e\,OPrev = e$. The record is thus a complete Quad-edge structure by itself, representing the simplest valid subdivision of the sphere $\mathbb{S}^2$—with a single edge e, two vertices $e\,Org \neq e\,Dest$, and a single face $e\,Left = e\,Right$.

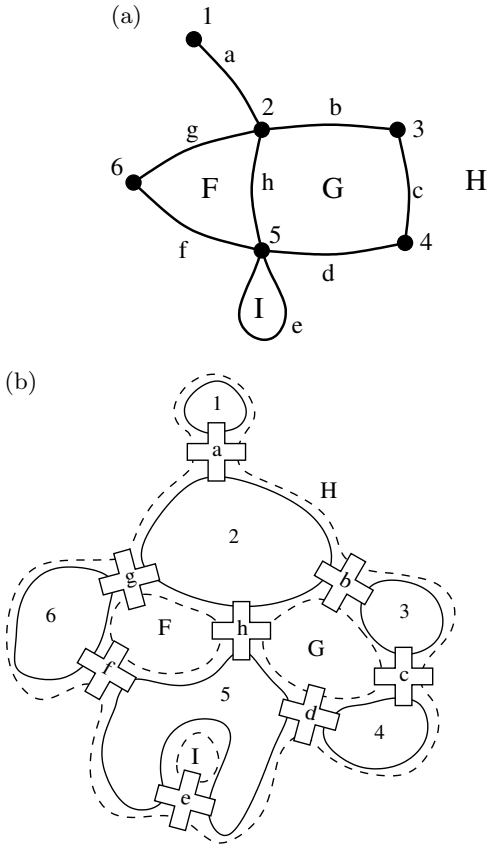


Figure 2: A simple subdivision of the sphere (a) and its Quad-edge representation (b).

The other Quad-edge operator is denoted by `Splice`(a, b) and takes two arc references a and b as parameters. If the vertices $a\,Org$ and $b\,Org$ are distinct, `Splice` will combine them into one vertex, by fusing the two arc rings into a single ring. Otherwise, `Splice` will break the vertex $a\,Org = b\,Org$ into two vertices, by splitting the ring of outgoing arcs into two separate rings. The operator also modifies the rings of dual arcs around faces $a\,Left$ and $b\,Left$ in the same way. See figure 3

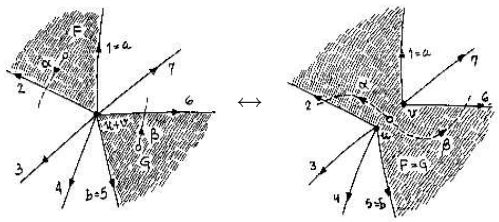


Figure 3: The effect of `Splice`(a, b) when either $a\,Org = b\,Org$ or $a\,Left = b\,Left$, but not both.

In both cases, `Splice` breaks the vertex rings just after the arcs a and b, and the operation amounts to

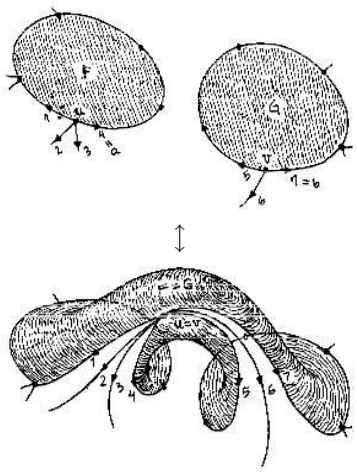


Figure 4: The effect of $\texttt{Splice}(a, b)$ when $a\ Org = b\ Org$ and $a\ Left = b\ Left$ are both true or both false.

swapping the contents of the `Next` fields of the corresponding arc records. The face rings are broken just *before* $a\ Rot$ and $b\ Rot$. Note that the `Splice` operation is its own inverse, i.e. the command sequence $\texttt{Splice}(a, b); \texttt{Splice}(a, b)$ leaves the structure unchanged.

Figure 3 illustrates this process for the two simplest cases where $a\ Org \neq b\ Org$ and $a\ Left = b\ Left$; or, conversely, where $a\ Org = b\ Org$ and $a\ Left \neq b\ Left$. In these two cases, the effect of `Splice` is to modify the subdivision, but not the topology of the underlying surface. In the two remaining cases—where $a\ Org \neq b\ Org$ and $a\ Left \neq b\ Left$, or $a\ Org = b\ Org$ and $a\ Left = b\ Left$—the `Splice` operation modifies not only the subdivision but also the topology of the underlying surface, by creating or removing a handle from face $a\ Left$ to face $b\ Left$, and combining or splitting the vertices $a\ Org$ and $b\ Org$ across that handle. See figure 4.

3.5 Enumerating vertices and edges

The enumeration of all vertices and/or all edges of a subdivision may be performed by a standard breadth-first traversal (2) of the Quad-edge structure:

```
PROCEDURE Enum(e, VVisit, AVisit)
    Q ← {e}; e Org .mark ← TRUE;
    while Q ≠ {} do
        e ← Dequeue(Q); VVisit(m);
        n ← e ONext Sym;
        while n ≠ e Sym do
            AVisit(n);
            if not n Org .mark then
                Enqueue(Q, n);
                n Org .mark ← TRUE;
            n ← n DNext;
```

This algorithm will call `VVisit` once for every vertex v of the mesh, giving as argument one of the arcs out of v. It will also call $\texttt{AVisit}(e)$ once for every primal arc e of the mesh. The algorithm uses a queue (2) Q of arc references, handled through the procedures $\texttt{Enqueue}(Q, e)$ and $\texttt{Dequeue}(Q)$; and a flag $e.\texttt{Data.mark}$ that tells whether a node has ever been inserted in the queue.

4 Three-dimensional WFC with the Quad-edge

According to Gjøystdal et al. (3), the main steps in WFC are: (i) generation of the initial wavefront, (ii) propagation of the front through one time step, (iii) adjustments of the ray field's density, and (iv) detection of arrivals at receivers. We will assume here that layers are homogeneous, so that ray paths are straight within each layer.

4.1 Wavefront model

The wavefront is modeled by a mesh of triangles, whose topology is represented by a Quad-edge structure. The `Data` field of a primal arc e points to a sample node s, denoted here by $e\ Org$ (see section 2.1), whose coordinates $s.c$ are those of the corresponding mesh vertex.

4.2 Generating the initial wavefront

The process is started with a very small spherical wavefront of specified mode (S or P) centered at a specified source, which is assumed to be a single point. This inital wavefront can be represented by a small number samples, displaced from the source by a small amount along a fixed set of directions, and connected into a mesh with a fixed topology. For instance, one could use 20 samples, placed and connected like the vertices of an icosahedron (3). We use points from an m by n spherical coordinate grid, with given m and n, with the topology of their *Delaunay triangulation* (DT), computed as described by Guibas and Stolfi (4).

4.3 Propagating the wavefront

In order to simulate the wavefront propagation over each time step, it suffices to traverse the whole data structure, and perform an initial-value ray tracing computation at every node s. This step is described by the procedure $\texttt{AdvanceNode}(s, h, \sigma)$ below, which updates the coordinates $s.c$ of node s by following the corresponding ray from the current time t to time $t + h$, with interactions specified by σ.

```
PROCEDURE AdvanceNode(s, h, σ)
    while TRUE do
      p ← s.c;
      s.c ← s.c + h s.v;
      (q, α, i) ← FirstHit(p, s.c);
      if α = +∞ then break;
      s.c ← q; h ← h − α h;
      Interact(s, i, σ[s.k]);
      s.k ← s.k + 1;
```

The procedure `FirstHit` is a purely geometric algorithm that, given the endpoints p and c of a line segment, intersects that segment with the surfaces of all reflectors in the model, and returns the intersection point q which lies closest to p. It also returns the ratio $\alpha = \text{dist}(p, q)/\text{dist}(p, c)$, and the index i of the corresponding reflector.

The signature of the desired wavefront is specified by a vector σ, where $\sigma[k]$ specifies which reflector should be involved in its kth interaction event, and which secondary ray should be followed after that event. Thus, for example, $\sigma[3] =$ 1rP specifies that the third interaction should be with refletor 1, and the simulator should then follow the reflected (r) ray with pressure (P) mode. The `Interact` procedure computes the velocity vector the appropriate secondary ray, using the geometric reflection law for r signatures, and Snell's law for t signatures.

It often happens that a simulated ray cannot be propagated past an interaction event; either because the reflector that was hit is not the one requested by the signature, or because the reflector and ray parameters are such that the requested secondary ray does not exist—e.g. due to total absorption or reflection. In such cases, the the sample record s is marked as *dead*. Dead nodes are ignored when processing or plotting the wavefront, but are retained in the wavefront model in order to preserve the mesh topology.

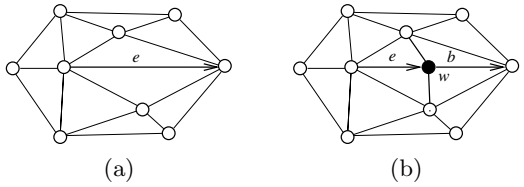
4.4 The density control mechanism

Node density control comprises two steps: (a) detect violations of the uniform density criteria, and (b) ajust the mesh so as to restore those criteria.

Vinje's *distance criterion* (8) consists of an upper bound on the distance between a sample point and its closest neighbor. Other criteria have been proposed by Vinje (9; 10), Lambaré (6), Kraaijpoel (5), and others. Our criteria consist of a lower bound $d_{\min}$ and an upper bound $d_{\max}$ on the *discrepancy* between adjacent mesh nodes r, s; which is defined as an appropriate combination of the spatial (Euclidean) distance between the positions $r.c, s.c$, and the angle between the velocity vectors $r.v, s.v$.

Thus, we solve the first part of the problem by checking, for each undirected edge e of the mesh, the discrepancy between the endpoints $r = e\,Org$ and $s = e\,Dest$. If the discrepancy exceeds the upper bound $d_{\max}$, then we split the edge in two by a new vertex w, and split the faces $e\,Left$ and $e\,Right$ by two new edges. Conversely, if the discrepancy between r and s is less than $d_{\min}$, we contract the edge, merging its endpoints into a new vertex w, and removing the adjacent faces. In either case, the position, velocity, and other properties of w are chosen by interpolation of those of r and s. These two cases are handled by the procedures `SplitEdge` and `ContractEdge` below:

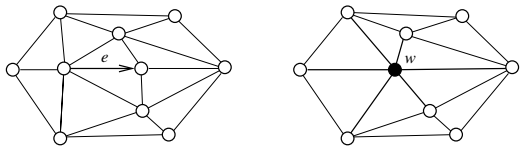
```
PROCEDURE SplitEdge(e)
    r ← e Org; s ← e Dest;
    w ← InterpolateNode(r, s);
    b ← MakeEdge();
    b Org ← w; b Dest ← e Dest;
    e Dest ← w;
    b Left ← e Left; b Right ← e Right;
    b LNext ← e LNext; b RNext ← e RNext;
    e LNext ← e RNext ← b;
    Connect(b LNext, b);
    Connect(e OPrev, e Sym);
```



(a) (b)

Figure 5: Increasing the sample density with `SplitEdge`(e): (a) before and (b) after the call.

```
PROCEDURE ContractEdge(e)
    r ← e Org; s ← e Dest;
    w ← InterpolateNode(r, s);
    a ← e ONext ONext;
    b ← e LNext;
    DeleteEdge(e ONext);
    DeleteEdge(e OPrev);
    DeleteEdge(e);
    Splice(a, b);
```



(a) (b)

Figure 6: Decreasing the sample density with `ContractEdge`(e): (a) before and (b) after the call.

The procedures `DeleteEdge` and `Connect` are topological operators defined in terms of `MakeEdge` and `Splice` (4). The call `DeleteEdge`(e) disconnects the edge e from the Quad-edge, by performing `Splice`$(e, e\,OPrev)$ and `Splice`$(e\,Sym, e\,Sym\,OPrev)$. This operation merges the two faces $e\,Left$ and $e\,Right$ into a four-sided face. The procedure `Connect`(a, b) is, to some extent, its inverse: it creates a new undirected edge e, with `MakeEdge`, and then adds it to the structure, connecting the vertices $a\,Dest$ and $b\,Org$ across the face $a\,Left$ (assumed equal to $b\,Left$).

5 Experiments

At present we have implemented just the first two steps of WFC: generation of the initial wavefront and its propagation. Figure 7 shows the propagation of a wavefront in a homogeneous layered medium with one reflector, according to the signature ⟨P, 1rP⟩. This prototype of the simulator was implemented in MATLAB, and is now being extended to more general models and converted to the C language.

6 Conclusions

The algorithms and experiments described above show that the Quad-edge structure simplifies considerably the implementation of WaveFront Construction method, especially the sample density control mechanism.

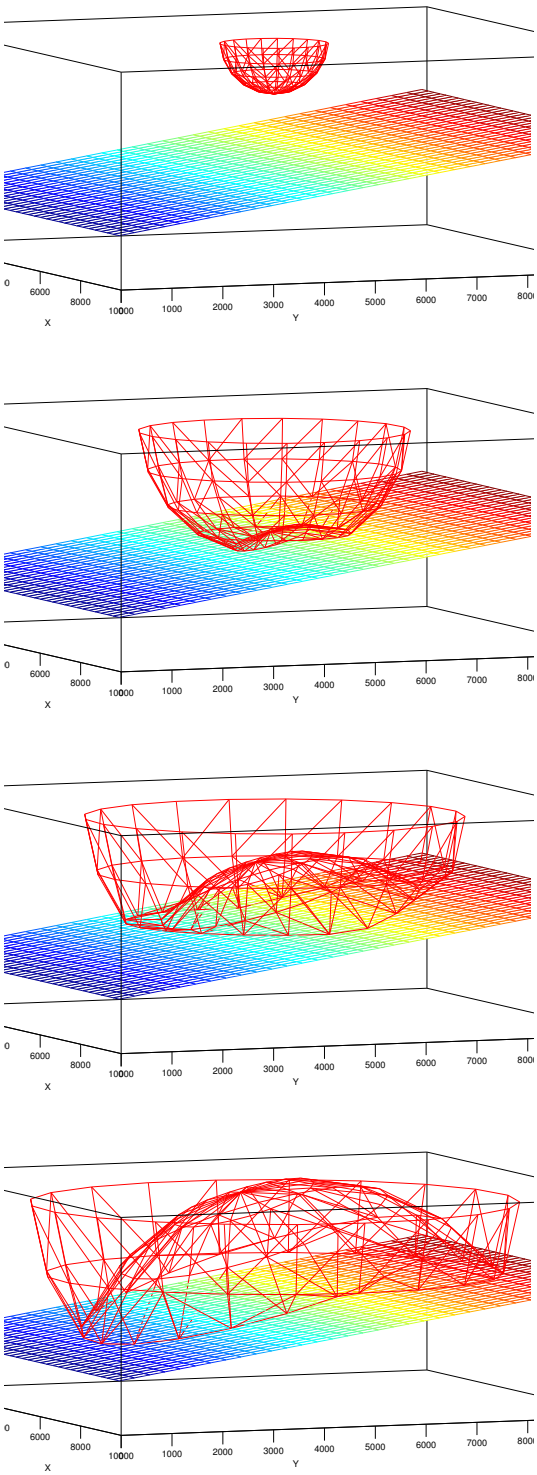


Figure 7: The reflection of a P-wave on a plain reflector.

Acknowledgements

We thank CNPq and the sponsors of the WIT – Wave Inversion Technology Consortium, Germany.

References

[1] B. Baumgart. A polyhedron representation for computer vision. In *Proc. 1975 AFIPS National Computer Conference*, volume 44, pages 589–596, 1975.

[2] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[3] H. Gjøystdal, E. Iversen, R. Laurain, I. Lecomte, V. Vinje, and K. Åstebøl. Computer representation of complex 3D geological structures using a new 'solid modeling' technique. *Geophys. Prospect.*, 33:1195–1211, 1985.

[4] L. J. Guibas and J. Stolfi. Primitives for the manipulations of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[5] D. Kraaijpoel. *Seismic Ray Fields and Ray Field Maps: Theory and Algorithms*. PhD thesis, Faculteit Geowetenschappen, University of Utrecht, 2003.

[6] G. Lambaré, P. Lucio, and A. Hanyga. Two-dimensional multivalue traveltime and amplitude maps by uniform sampling of a ray field. *Geophys. J. Int.*, 125:584–598, 1996.

[7] V. Červený. *Seismic Ray Theory*. Cambridge University Press, 2001.

[8] V. Vinje, E. Iversen, and H. Gjøystdal. Traveltime and amplitude estimation using wavefront construction. *Geophysics*, 58:1157–1166, 1993.

[9] V. Vinje, E. Iversen, H. Gjøystdal, and K. Åstebøl. Estimation of multivalued arrivals using wavefront construction, Part I. *Geophys. Prospect.*, 44:819–842, 1996.

[10] V. Vinje, E. Iversen, H. Gjøystdal, and K. Åstebøl. Estimation of multivalued arrivals using wavefront construction, Part II: Tracing and interpolation. *Geophys. Prospect.*, 44:843–858, 1996.