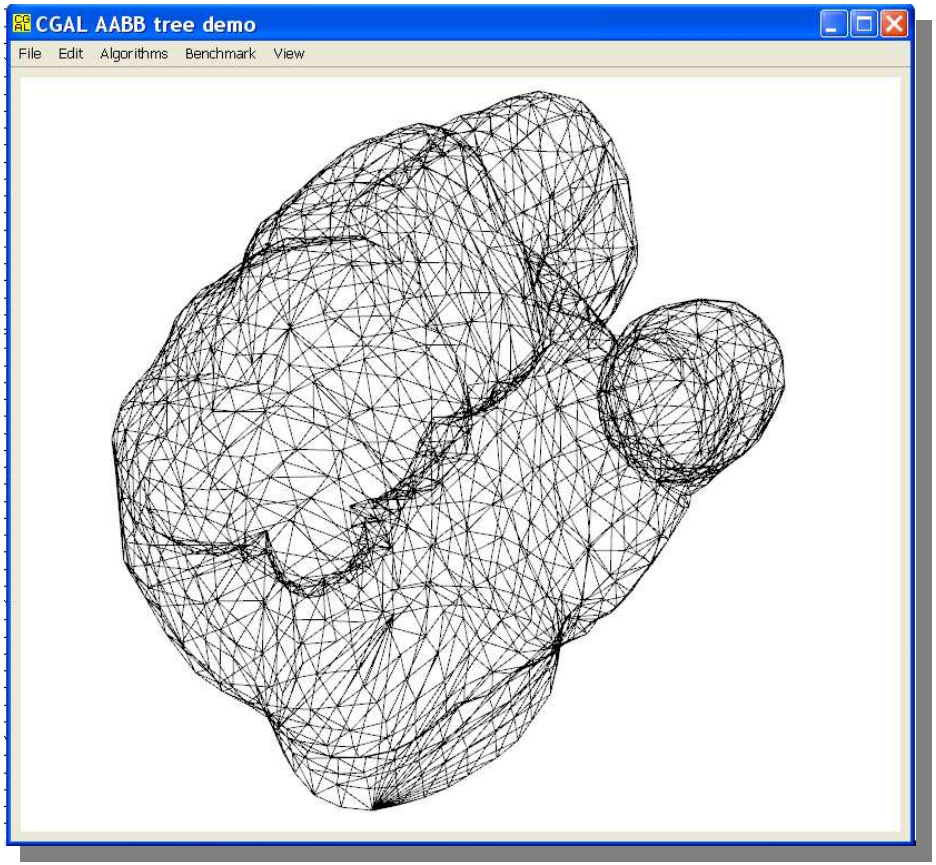
AABB Tree Demo

The AABB tree demo showcases several algorithms where the AABB tree is put at work with polyhedron facet and edge primitives. It also provides a menu for benchmarking so that a user can quickly get an order of the performances on her models for both intersection and distance queries.

The application accepts as input (and through drag & drop) OFF files which represent triangle surface meshes. The main window displays the input triangle surface mesh in wireframe mode and allows interaction through the QGLViewer arcball library.

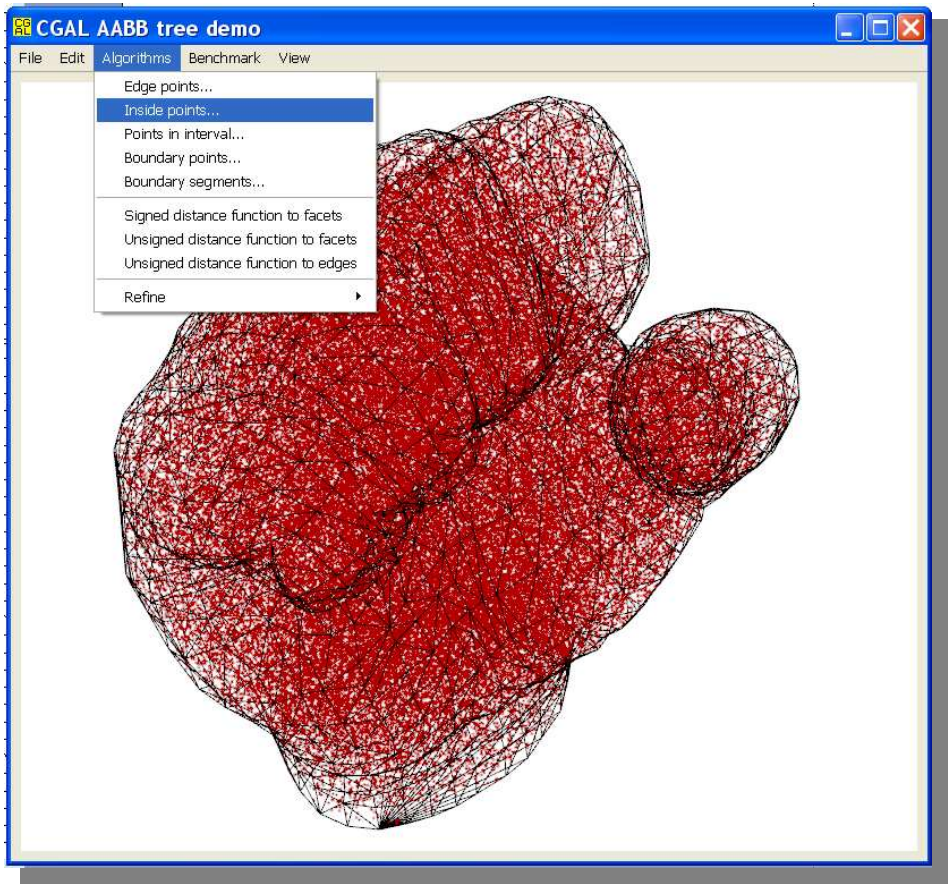


Inside Points

Menu Algorithms / inside points allows the user generating a specified number of points inside the input surface mesh (which is assumed to be intersection free and without boundary).

The demo constructs an AABB tree with the mesh facets and generates many intersection queries with randomly generated rays (the source of each ray is randomly chosen inside the mesh bounding box and the direction is arbitrarily chosen for all rays).

Each query counts the number of intersection between the ray and the input mesh in order to determine whether the ray source point is inside or outside (inside if odd, outside if even).



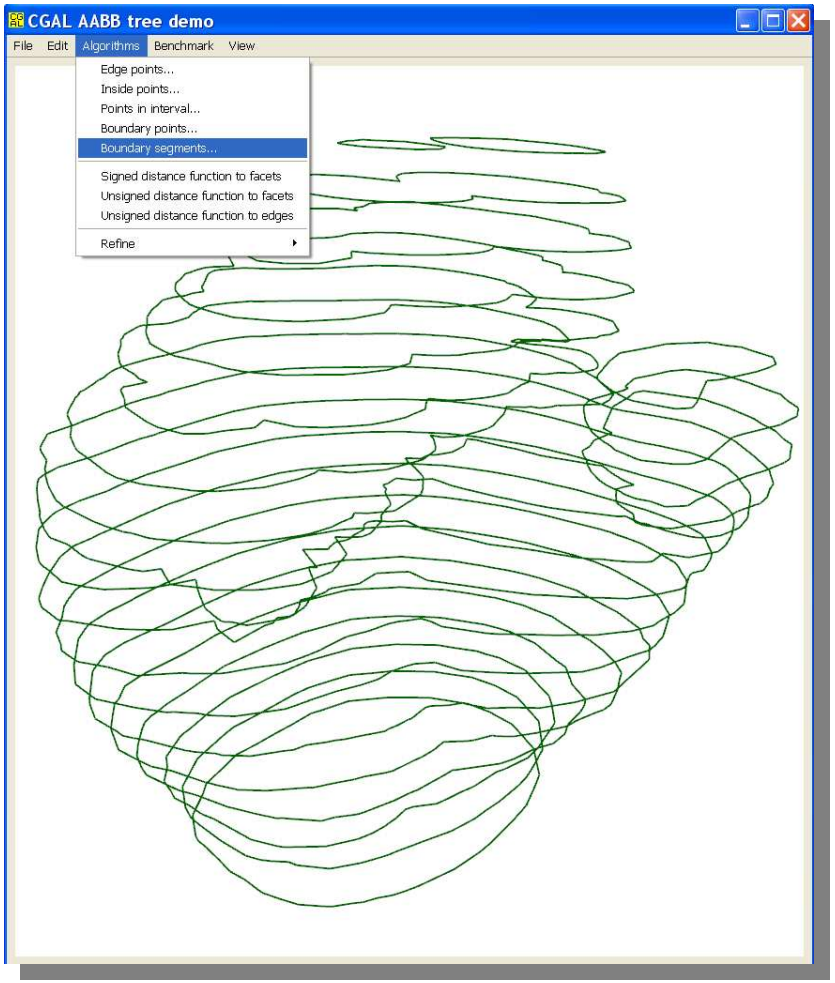
Boundary Segments

Menu Algorithms / boundary segments allows the user generating a specified number of horizontal slices from the input surface mesh.

The demo constructs an AABB tree with the mesh facets and generates many intersection queries with a set of horizontal planes. The objects returned (generally segments) are depicted in green.

The Algorithms / Edge points menu constructs an AABB tree with the mesh edge segments and generates random plane queries. The objects returned (generally points) are depicted in red.

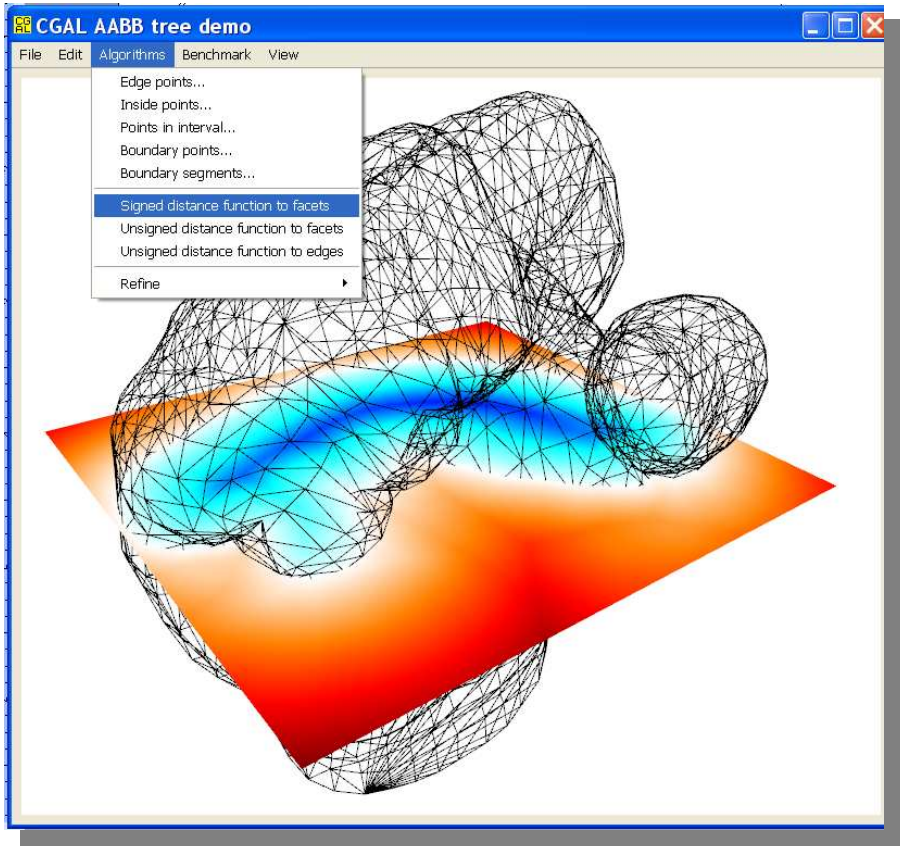
The Algorithms / boundary points menu constructs an AABB tree with the mesh facets and generates random line queries. The objects returned (generally points) are depicted in red.



Signed Distance Function

Menu Algorithms / signed distance function to facets allows the user generating a cross section image (100x100) of the signed distance function to the input surface mesh facets.

The demo constructs an AABB tree with the mesh facets and generates within the cross section a number of distance and intersection queries with points and rays. As for the "inside points" menu each ray query counts the number of intersection to determine the sign of the distance (inside versus inside) while each distance query determines its magnitude.

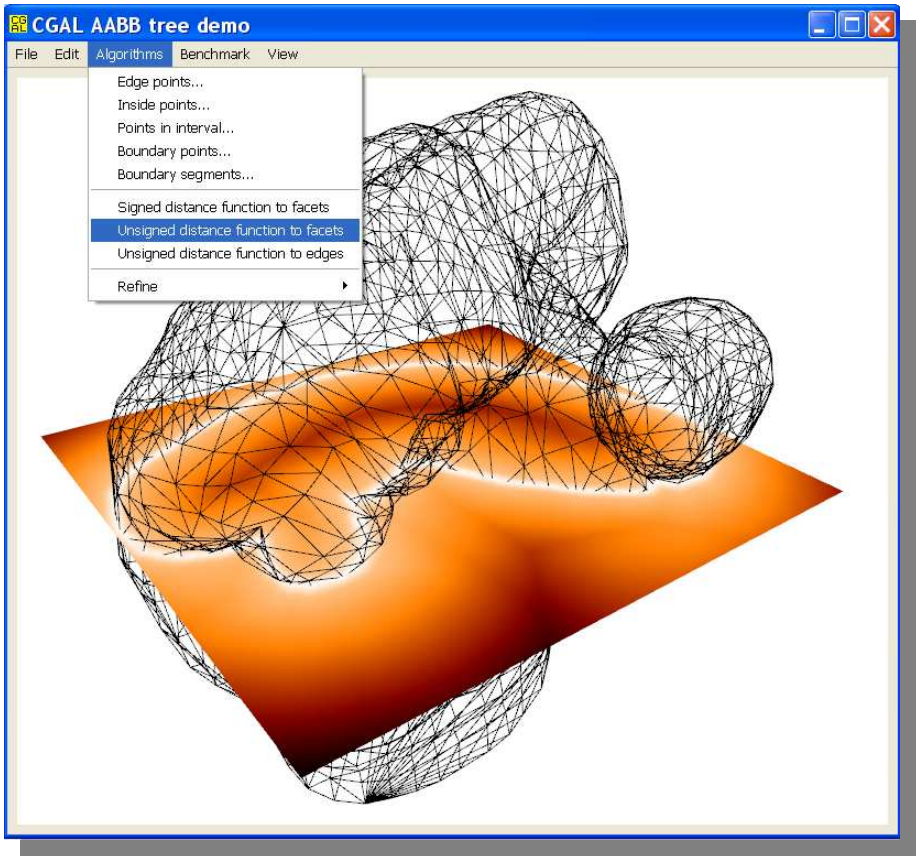


Unsigned Distance Function

Menu Algorithms / unsigned distance function to facets allows the user generating a cross section image (100x100) of the unsigned distance function to the input surface mesh facets.

The demo constructs an AABB tree with the mesh facets and generates within the cross section a number of distance queries with points.

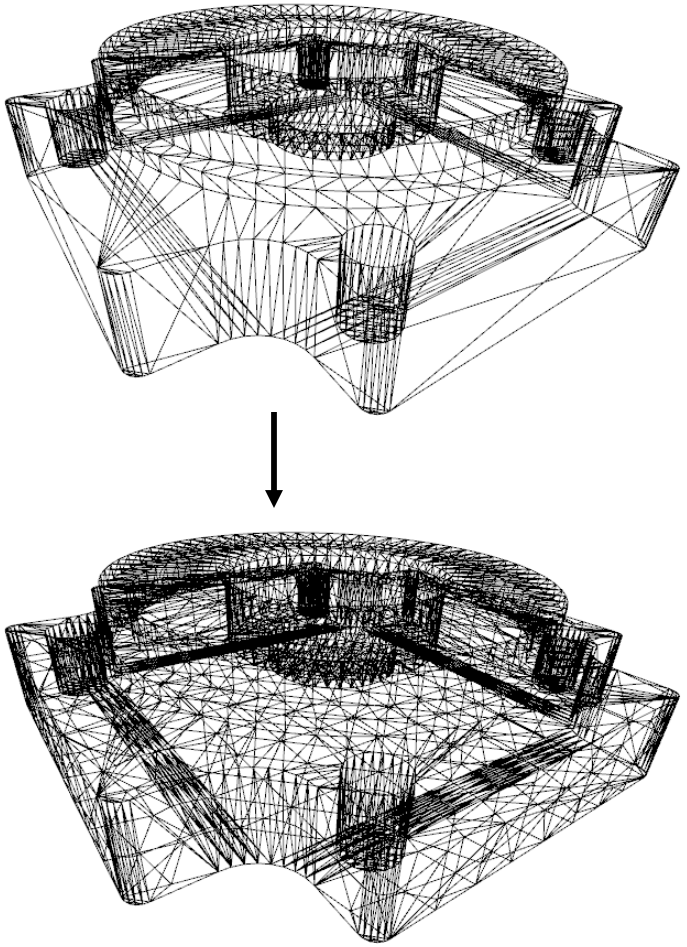
Another menu provides the same functionality but for the mesh edges.



Refinement

Menu Algorithms / Refine allows the user refining the current mesh through

- recursive longest edge bisection until a maximum specified edge length is matched (shown). The goal is to check the influence of long primitives spanning the whole bounding box of the input mesh over the performances of distance and intersection queries (accessible through the benchmark menu).
- Loop subdivision. The surface is made smoother through 1-to-4 triangle splits and vertex relocations (not shown).



Benchmarks

Menu Benchmarks constructs an AABB tree with the mesh facets and allows benchmarking the number of intersection and distance queries per second with various random primitives. The user is invited to specify the duration of each test.

The sub-menu "against #triangles" benchmarks various performance indicators (memory occupancy, tree construction time, number of intersection and distance queries per second) against an increasing number of triangles of the current mesh. The latter is obtained through recursive longest bisection until the mesh complexity reaches 1M triangles. All results are output in ASCII form to the console window.

```
Benchmark intersections
Construct AABB tree...done (0.157 s)
Generates random queries...done (1.468 s)
Benchmark do_intersect()
198294 queries/s with segment
197106 queries/s with ray
217081 queries/s with line
396296 queries/s with plane
Benchmark any_intersected_primitive()
195489 queries/s with segment
193271 queries/s with ray
214415 queries/s with line
386114 queries/s with plane
Benchmark any_intersection()
146888 queries/s with segment
144476 queries/s with ray
156757 queries/s with line
243704 queries/s with plane
Benchmark number_of_intersected_primitives()
67802 queries/s with segment
57004 queries/s with ray
55505 queries/s with line
8676 queries/s with plane
Benchmark all_intersected_primitives()
65249 queries/s with segment
53711 queries/s with ray
52235 queries/s with line
6064 queries/s with plane
Benchmark all_intersections()
47624 queries/s with segment
38483 queries/s with ray
35678 queries/s with line
2687 queries/s with plane
```