Hierarchical Space Partitioning of a World of Triangles using a Fixed Resolution Octree

Paul Nathan 17/01/2015

The standard octree used for storing and efficiently searching points in space in $O(\log(N))$ time will fail to work with triangles since the node subdivision loop will run indefinitely, trying to split the continuous region of the triangle's interior into distinct entities. Thus the subdivision loop must be ended by specifying a minimum spatial resolution beyond which no subdivision will occur. Instead, any triangles remaining in the last level of subdivision will be returned during a search. This is unlike the octree for points, where only a single point at most will ever be returned from a spatial query. Thus the distinct features of the fixed resolution octree for triangles are

1. Finite level of subdivisions, determined by specifying a minimum spatial resolution and the World spatial extent

2. Multiple objects may be returned by a query, some of which may be returned multiple times from different nodes

TREE CONSTRUCTION OUTLINE

- Determine the extent of the World box and specify the minimum node resolution, then evaluate the maximum number of subdivisions. A subdivision is the division of a parent node into eight octants (child nodes), each with linear dimension half that of the parent node. For simplicity, the World box will be an axis-aligned cube of side lengths *D*. The minimum side length shall be donated *d*. The maximum number of subdivision $L_{\max}$ rounded up to the next integer is given by

$$L_{\max} = \left\lceil \frac{\ln\left(\frac{D}{d}\right)}{\ln 2} \right\rceil = \left\lceil \log_2\left(\frac{D}{d}\right) \right\rceil$$

  If precise minimum resolution is important, then the World extent *D* can be enlarged such that $D = 2^{L_{\max}} d$. The rounding up of $L_{\max}$ ensures that *D* will always be greater than or equal to the original world extent.

- Prepare a dynamic array for holding the tree nodes. A tree node custom data type contains the following:

  - Subdivision level ($L = 0$ represents the root node)
  - Box corner points $\mathbf{P}_{\max}$ and $\mathbf{P}_{\min}$ (for ease of creating subdivisions)
  - Box face unit normals, edge vectors and edge unit normals
  - Index of parent node and indices of child nodes (if any)
  - Indices of triangles colliding with node (i.e. contained in or intersecting node box)
  - Boolean indicating whether the node is a leaf node (no children). This would be due either to containing no triangles, or to having reached the maximum subdivision level $L = L_{\max}$

- Initialise a root node containing the indices of all triangles, and box corner points corresponding to the World box dimensions. The root node's parent index is set to zero (itself). Push the index of this root node onto an empty stack of integers

- Begin while loop
    - Pop one index from the stack and obtain the corresponding node
    - Loop through all triangles contained in the node's parent triangle index list[1]
        - Check for collision or containment of triangle with node box. Add any positive results to the node's list of triangles
    - If no collisions or $L = L_{\max}$, mark node as leaf node and continue
    - Else subdivide this node into octants of half linear scale. Assign the present node's index as the parent index of the new child nodes, and store the child node indices in the present node's child node index list. Increment the value of L for the child nodes. The child nodes are appended onto the end of the tree node array in its current state, the indices of the child nodes therefore range from the tree node array length to this value plus seven. Push the indices of these eight new nodes onto the node index stack and continue
    - Loop until node index stack is empty

TREE QUERY OUTLINE (for triangle – triangle intersection)

- Prepare a dynamic array of indices of candidate triangles for full collision detection test

- Push the octree root node index 0 onto an empty stack of integers

- Begin while loop
    - Pop one index from the stack and obtain the corresponding node
    - Check query triangle for collision with node box
    - If collision, then
        - if this is a leaf node then add the node's contained triangle index list (unless empty) to the dynamic array of indices of candidate triangles to test fully, else add this node's eight child node indices to the node index stack and continue
    - Loop until node index stack is empty

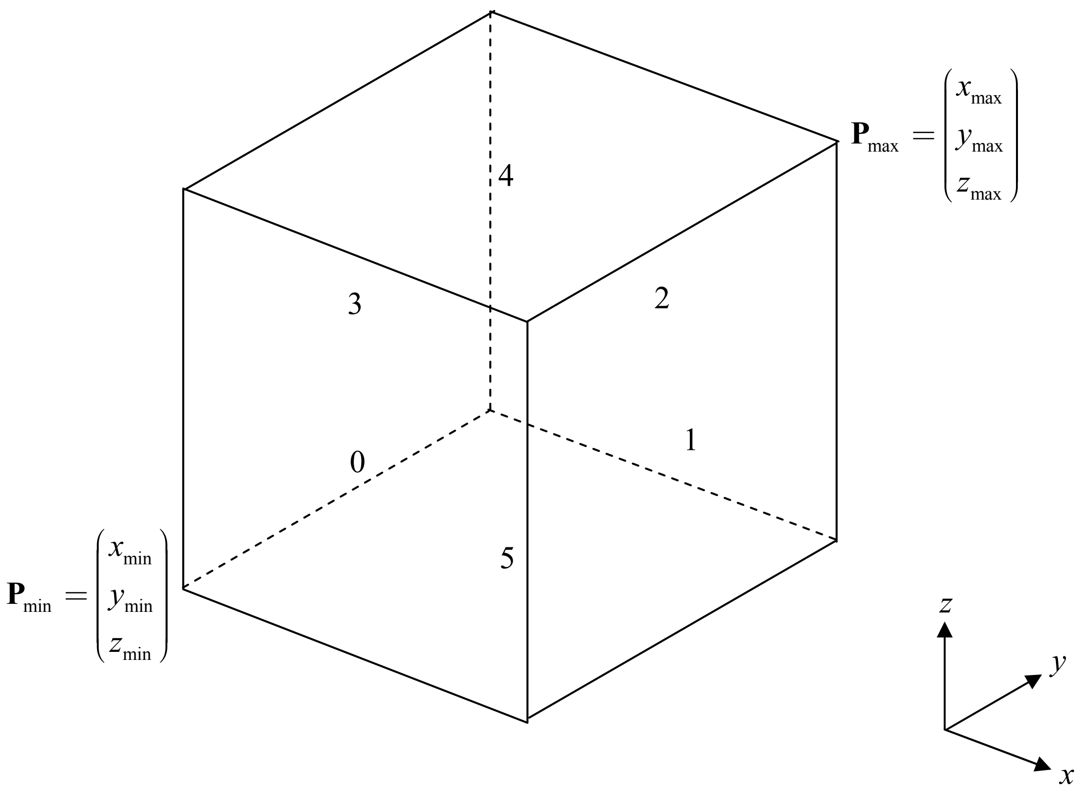
- Remove any duplicate index entries in the returned list of triangles, and then perform detailed triangle – triangle intersection test between query triangle and returned triangles.

Note that this same query procedure could be used to check for ray – triangle intersections, the only difference being the simpler collision detection required (ray – box-plane tests)

[1] If a triangle is not in a child node's parent triangle index list, it cannot possibly intersect one of the child nodes as it is spatially separated from the entire octant of space occupied by the parent, and therefore also all its child nodes. Thus it is not necessary to test the entire array of triangles.

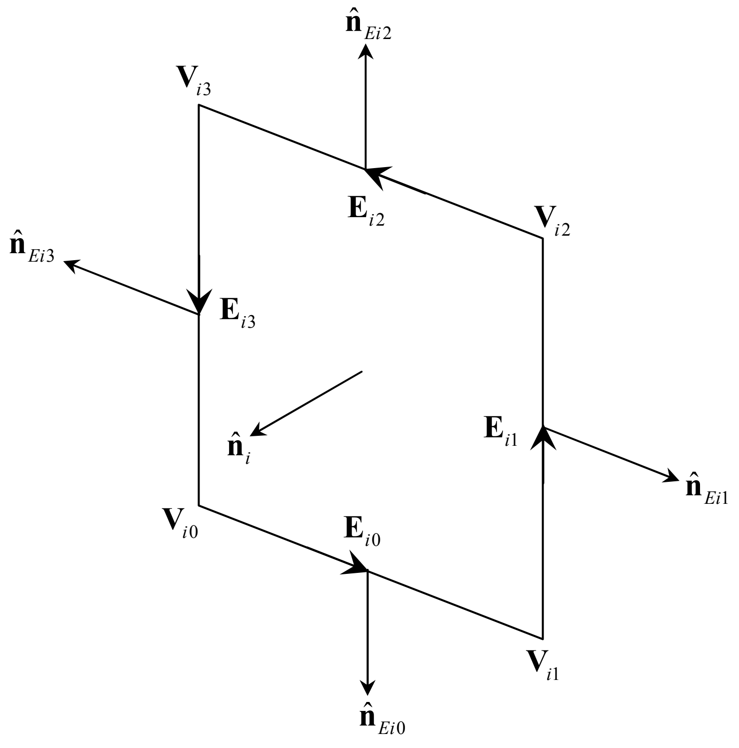
NODE BOX GEOMETRY

Face numbers:



4
3
2
1
0
5
$\mathbf{P}_{\max} = \begin{pmatrix} x_{\max} \\ y_{\max} \\ z_{\max} \end{pmatrix}$
$\mathbf{P}_{\min} = \begin{pmatrix} x_{\min} \\ y_{\min} \\ z_{\min} \end{pmatrix}$
z
y
x

All faces are wound anti-clockwise looking towards the outward facing surface (looking into the normal vector). For box face *i*:



The equations describing all the vertices are listed below

$$\begin{aligned}
\mathbf{V}_{00} &= \mathbf{P}_{\min} \\
\mathbf{V}_{01} &= \left(x_{\max} \quad y_{\min} \quad z_{\min}\right) \\
\mathbf{V}_{02} &= \left(x_{\max} \quad y_{\min} \quad z_{\max}\right) \\
\mathbf{V}_{03} &= \left(x_{\min} \quad y_{\min} \quad z_{\max}\right)
\end{aligned}$$

$$\begin{aligned}
\mathbf{V}_{10} &= \mathbf{V}_{01} \\
\mathbf{V}_{11} &= \left(x_{\max} \quad y_{\max} \quad z_{\min}\right) \\
\mathbf{V}_{12} &= \mathbf{P}_{\max} \\
\mathbf{V}_{13} &= \mathbf{V}_{02}
\end{aligned}$$

$$\begin{aligned}
\mathbf{V}_{20} &= \mathbf{V}_{11} \\
\mathbf{V}_{21} &= \left(x_{\min} \quad y_{\max} \quad z_{\min}\right) \\
\mathbf{V}_{22} &= \left(x_{\min} \quad y_{\max} \quad z_{\max}\right) \\
\mathbf{V}_{23} &= \mathbf{V}_{12}
\end{aligned}$$

$$\begin{aligned}
\mathbf{V}_{30} &= \mathbf{V}_{21} \\
\mathbf{V}_{31} &= \mathbf{V}_{00} \\
\mathbf{V}_{32} &= \mathbf{V}_{03} \\
\mathbf{V}_{33} &= \mathbf{V}_{22}
\end{aligned}$$

$$\begin{aligned}
\mathbf{V}_{40} &= \mathbf{V}_{03} \\
\mathbf{V}_{41} &= \mathbf{V}_{02} \\
\mathbf{V}_{42} &= \mathbf{V}_{12} \\
\mathbf{V}_{43} &= \mathbf{V}_{22}
\end{aligned}$$

$$\begin{aligned}
\mathbf{V}_{50} &= \mathbf{V}_{00} \\
\mathbf{V}_{51} &= \mathbf{V}_{21} \\
\mathbf{V}_{52} &= \mathbf{V}_{11} \\
\mathbf{V}_{53} &= \mathbf{V}_{01}
\end{aligned}$$

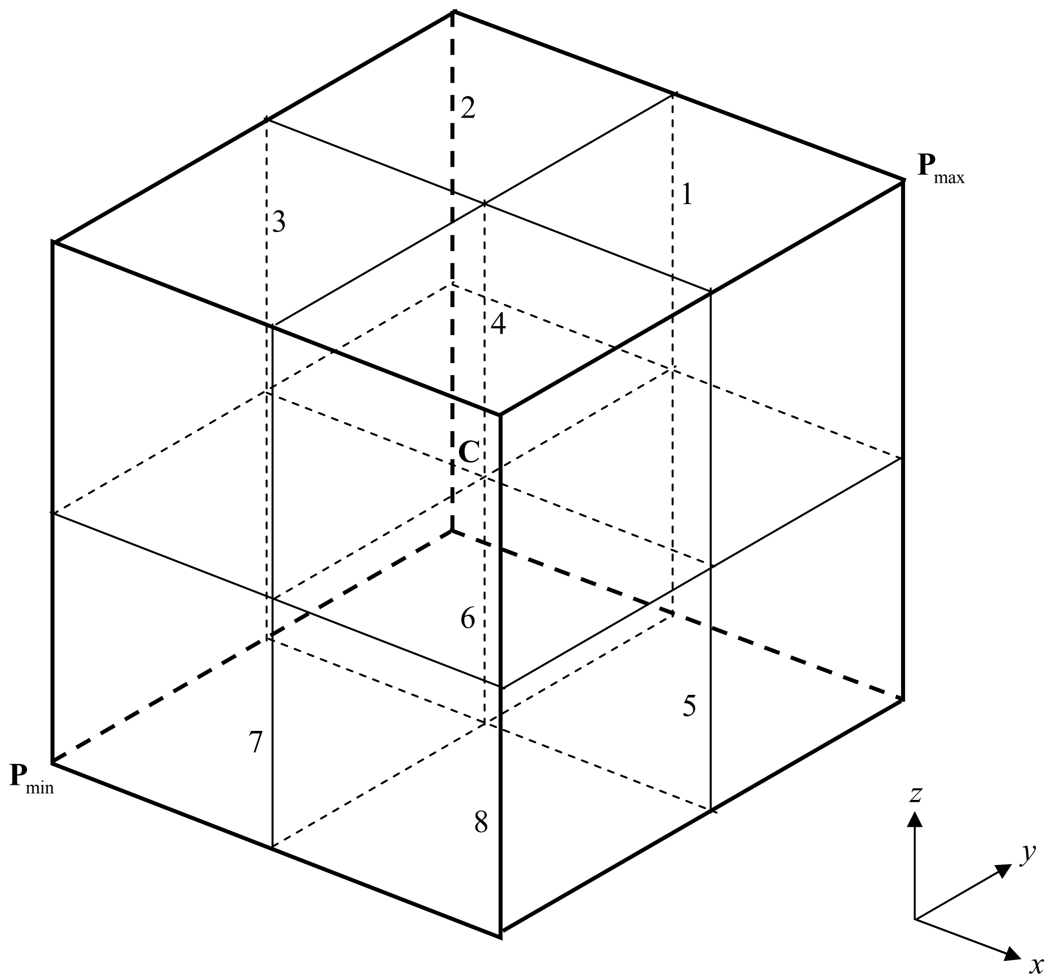
The equations describing all the edges and normals are listed below. The first subscript represents the box face number ($i = 0 \ldots 5$), the second subscript represents the edge number ($j = 0 \ldots 3$)

$$
\begin{aligned}
\mathbf{E}_{i0} &= \mathbf{V}_{i1} - \mathbf{V}_{i0} \\
\mathbf{E}_{i1} &= \mathbf{V}_{i2} - \mathbf{V}_{i1} \\
\mathbf{E}_{i2} &= \mathbf{V}_{i3} - \mathbf{V}_{i2} \\
\mathbf{E}_{i3} &= \mathbf{V}_{i0} - \mathbf{V}_{i3}
\end{aligned}
$$

$$
\begin{aligned}
\hat{\mathbf{n}}_i &= \frac{\mathbf{E}_{i3} \times \mathbf{E}_{i0}}{\left\| \mathbf{E}_{i3} \times \mathbf{E}_{i0} \right\|} \\
\hat{\mathbf{n}}_{Eij} &= \frac{\mathbf{E}_{ij}}{\left\| \mathbf{E}_{ij} \right\|} \times \hat{\mathbf{n}}_i \qquad i = 0 \ldots 5;\ j = 0 \ldots 3
\end{aligned}
$$

OCTANT GEOMETRY

Consider the division of a parent node box into its eight octants (child node boxes). The corner points of the child nodes can be determined from the corner points of the parent node, and the remaining vertices worked out using the equations listed above. The subscript after the corner point denotes the octant number, with the parent node having no subscript.



$$\mathbf{C} = \frac{1}{2}\left(\mathbf{P}_{\max} + \mathbf{P}_{\min}\right)$$

$$\begin{aligned}\mathbf{P}_{\max 1} &= \mathbf{P}_{\max}\\ \mathbf{P}_{\min 1} &= \mathbf{C}\end{aligned}$$

$$\begin{aligned}\mathbf{P}_{\max 2} &= \left(C_x \quad P_{\max y} \quad P_{\max z}\right)\\ \mathbf{P}_{\min 2} &= \left(P_{\min x} \quad C_y \quad C_z\right)\end{aligned}$$

$$\begin{aligned}\mathbf{P}_{\max 3} &= \left(C_x \quad C_y \quad P_{\max z}\right)\\ \mathbf{P}_{\min 3} &= \left(P_{\min x} \quad P_{\min y} \quad C_z\right)\end{aligned}$$

$$\begin{aligned}\mathbf{P}_{\max 4} &= \left(P_{\max x} \quad C_y \quad P_{\max z}\right)\\ \mathbf{P}_{\min 4} &= \left(C_x \quad P_{\min y} \quad C_z\right)\end{aligned}$$

$$\begin{aligned}\mathbf{P}_{\max 5} &= \left(P_{\max x} \quad P_{\max y} \quad C_z\right)\\ \mathbf{P}_{\min 5} &= \left(C_x \quad C_y \quad P_{\min z}\right)\end{aligned}$$

$$\begin{aligned}\mathbf{P}_{\max 6} &= \left(C_x \quad P_{\max y} \quad C_z\right)\\ \mathbf{P}_{\min 6} &= \left(P_{\min x} \quad C_y \quad P_{\min z}\right)\end{aligned}$$

$$\begin{aligned}\mathbf{P}_{\max 7} &= \mathbf{C}\\ \mathbf{P}_{\min 7} &= \mathbf{P}_{\min}\end{aligned}$$

$$\begin{aligned}\mathbf{P}_{\max 8} &= \left(P_{\max x} \quad C_y \quad C_z\right)\\ \mathbf{P}_{\min 8} &= \left(C_x \quad P_{\min y} \quad P_{\min z}\right)\end{aligned}$$

TRIANGLE – BOX COLLISION DETECTION

In order to determine whether a triangle should be included in a node's object list, it is necessary to perform a proper containment/intersection test between the triangle and the node box. This can be done in three steps, in order of increasing computational cost:

1. Triangle vertex contained in box:
   - all signed distances from box planes are negative (including zero)

2. Box edge – triangle intersection:
   - Edge – plane test followed by point-in-triangle test

3. Triangle edge – box-face intersection:
   - Edge – plane test followed by point-in-quadrilateral test

In all subsequent equations, the subscript T denotes 'triangle'. Quantities without a first subscript of T are taken to be those associated with the node box (plane or edge or vertex).

1. Triangle vertex contained in box:

The signed distance d_i of the triangle vertex $\mathbf{V}_T$ from the box face i is given by

$$d_i = \hat{\mathbf{n}}_i \bullet \left(\mathbf{V}_T - \mathbf{V}_{i0}\right)$$

The condition for vertex containment is

$$d_i < \varepsilon \qquad i = 0 \ldots 5$$

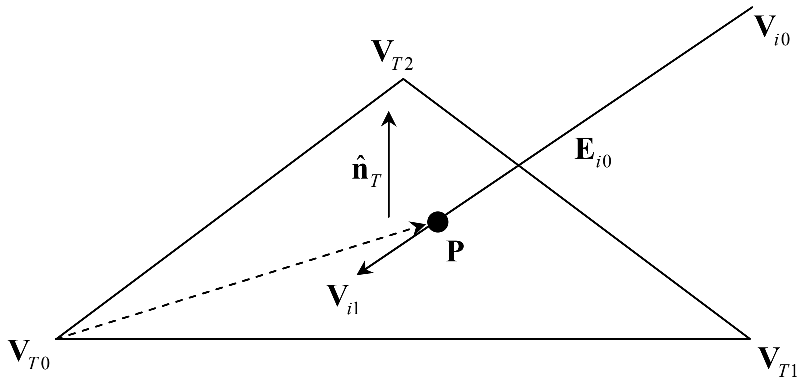
If the test returns true, then this particular triangle (via its index) is added to the node's object list. The test loop can exit as soon as any one of the conditions returns false, as if the signed distance is ever positive, then the point must be outside the box. If false, repeat the test for the other vertices of the triangle. If all tests return false, then continue onto the next step.

2. Box edge – triangle intersection:

This test is mathematically identical to the edge – triangle test carried out in step 2 of the triangle-triangle collision test, only here the edge is that of a box rather than another triangle.

a) Point-on-edge test:

Consider the example of edge 0 of box face i with the plane of the test triangle



The parametric vector equation of a point $\mathbf{P}$ on the edge is

$$\mathbf{P} = \mathbf{V}_{i0} + s\mathbf{E}_{i0}$$

The signed distance d of the point $\mathbf{P}$ from the plane of the triangle is given by

$$d = \hat{\mathbf{n}}_T \bullet \left(\mathbf{P} - \mathbf{V}_{T0}\right)$$

On intersection, $d = 0$. Substituting the equation for $\mathbf{P}$ and solving for s gives

$$\hat{\mathbf{n}}_T \bullet \left(\mathbf{V}_{i0} + s\mathbf{E}_{i0} - \mathbf{V}_{T0} \right) = 0$$

$$s = \frac{\hat{\mathbf{n}}_T \bullet \left(\mathbf{V}_{T0} - \mathbf{V}_{i0} \right)}{\hat{\mathbf{n}}_T \bullet \mathbf{E}_{i0}}$$

The edge intersects the plane if $0 \le s \le 1$, or, when carried out numerically

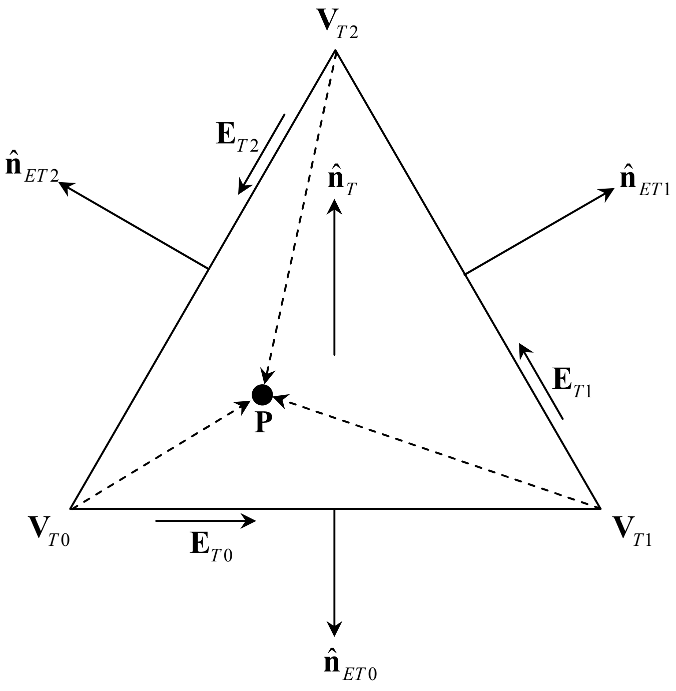
$$\left(s > -\varepsilon \right) \;\wedge\; \left(s < 1 + \varepsilon \right)$$

If false, move on to another edge. Note that if $\hat{\mathbf{n}}_T \bullet \mathbf{E}_{i0} = 0$ the edge is parallel to the plane, and if also $\hat{\mathbf{n}}_T \bullet \left(\mathbf{V}_{T0} - \mathbf{V}_{i0} \right) = 0$ then the edge is coplanar. To avoid a 0/0 ambiguity, ε can be added to the denominator.

Not all edges and faces need be tested when looping through the box faces. Testing all edges and faces would result in twelve duplicated tests. To avoid this unnecessary computational cost, it is only necessary to test faces 0, 1, 2, 3 (omit 4 and 5) with edges 0, 1, 2 (omit 3) of each face.

b) Point-in-triangle test:

If the point lies on the positive (outwards facing, or normal-directed) side of any edge, then it must be outside the triangle.



$$\hat{\mathbf{n}}_{ETj} \bullet \left(\mathbf{P} - \mathbf{V}_{Tj} \right) < \varepsilon \qquad j = 0,1,2$$

where

$$\hat{\mathbf{n}}_{ETj} = \frac{\mathbf{E}_{Tj}}{\left\| \mathbf{E}_{Tj} \right\|} \times \hat{\mathbf{n}}_T$$

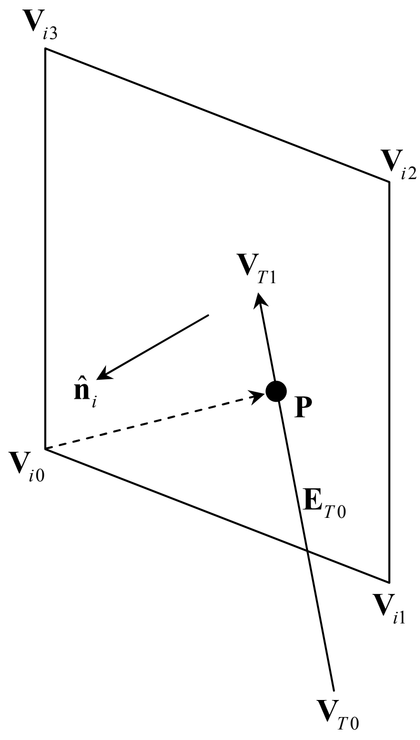
These edge unit normal vectors can be pre-calculated and stored with the triangle custom data type. This test is inclusive of the point lying on an edge. If the test returns true, then this particular triangle (via its index) is added to the node's object list, otherwise continue on to the final test below

3. Triangle edge – box-face intersection:

This test is similar to the test in step 3 above, just with the object interrogation roles reversed.

a) Point-on-edge test:

Consider the example of edge 0 of the triangle with the plane of the box face *i*



The parametric vector equation of a point **P** on the edge is

$$\mathbf{P} = \mathbf{V}_{T0} + s\mathbf{E}_{T0}$$

The signed distance *d* of the point **P** from the plane of the box face is given by

$$d = \hat{\mathbf{n}}_i \bullet \left(\mathbf{P} - \mathbf{V}_{i0}\right)$$

On intersection, $d = 0$. Substituting the equation for **P** and solving for *s* gives

$$\hat{\mathbf{n}}_i \bullet \left(\mathbf{V}_{T0} + s\mathbf{E}_{T0} - \mathbf{V}_{i0}\right) = 0$$

$$s = \frac{\hat{\mathbf{n}}_i \bullet \left(\mathbf{V}_{i0} - \mathbf{V}_{T0}\right)}{\hat{\mathbf{n}}_i \bullet \mathbf{E}_{T0}}$$

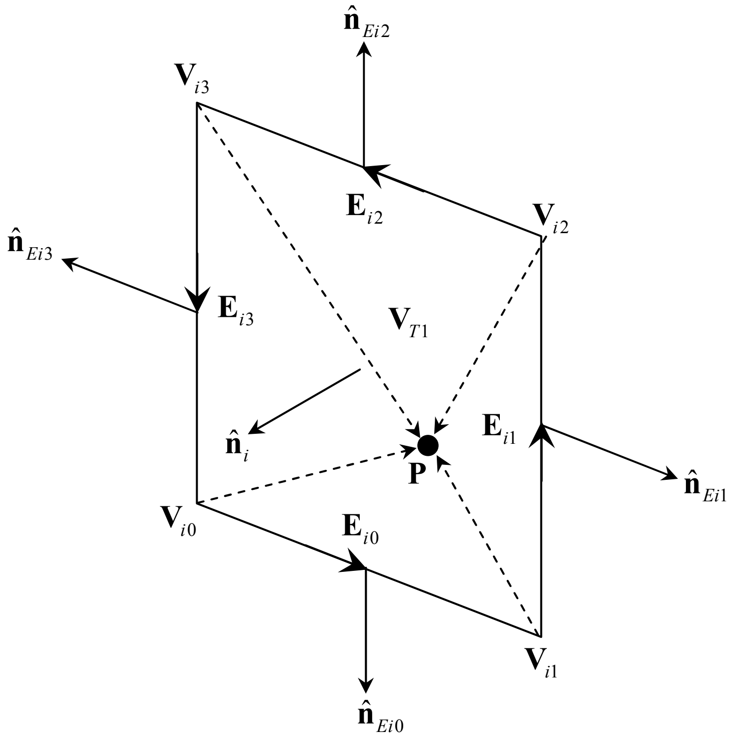
The edge intersects the plane if $0 \le s \le 1$, or, when carried out numerically

$$\left(s > -\varepsilon\right) \;\wedge\; \left(s < 1 + \varepsilon\right)$$

If false, move on to another edge. Note that if $\hat{\mathbf{n}}_i \bullet \mathbf{E}_{T0} = 0$ the edge is parallel to the plane, and if also $\hat{\mathbf{n}}_i \bullet \left(\mathbf{V}_{i0} - \mathbf{V}_{T0}\right) = 0$ then the edge is coplanar. To avoid a 0/0 ambiguity, ε can be added to the denominator.

b) Point-in-quadrilateral test:

If the point lies on the positive (outwards facing, or normal-directed) side of any edge, then it must be outside the quadrilateral.



$$\hat{\mathbf{n}}_{Eij} \bullet \left(\mathbf{P} - \mathbf{V}_{i0}\right) < \varepsilon \qquad j = 0,1,2,3$$

This test is inclusive of the point lying on an edge.

Additional Notes:

A sensible, working value for the Octree resolution can be programmatically obtained from the array of *N* triangles by finding the average of the areas of the parallelograms formed by each triangle and then taking the square root of this value to give the length of a side of a square of the same area

$$d = \sqrt{\frac{1}{N}\sum_{i}^{N}\left\|\mathbf{E}_{i2} \times \mathbf{E}_{i0}\right\|}$$