Mesh Representations of VecGeom Solids

Murat Topak

Supervisors: Andrei Gheata
Evgueni Tcherniaev

August 26th, 2019
ISSUE

- Create a polygonal mesh for each solid (tube, cone, sphere, etc.) for the purpose of visualization and overlap detection.

- In case of detector geometry, neighbor detection can speed up navigation queries.
APPROACH

sm: SolidMesh

ph: Polyhedron

p1: Polygon
+flnd: vector<size_t> = {2,0,1}

p2: Polygon
+flnd: vector<size_t> = {2,3,1}

p3: Polygon
+flnd: vector<size_t> = {2,3,0}

p4: Polygon
+flnd: vector<size_t> = {1,3,0}

vertices: vector<Vector3D>
+vertices[0]: Vector3D = (0,0,0)
+vertices[1]: Vector3D = (0,1,0)
+vertices[2]: Vector3D = (1,0,0)
+vertices[3]: Vector3D = (0,0,1)
1st round

Box

Parallelepiped

Tet

Tet

SEextruded

Tet
3rd round

cone

hype

torus (horn)

torus (standard)

torus (spindle)
4th round

- Polyhedron
- Gentrap
- Sphere
- Polycone
- Cuttube
Decompose a convex or concave polygon into a set of triangles (ear clipping algorithm).

Algorithms working on convex polygons then can be made available to concave. e.g. SAT.
- Crossing number algorithm
- Shoot a ray in arbitrary direction and count the crossings. The point is outside if even number of crossings and inside otherwise.
FIXDEGENERATE()

- Degeneracies are not allowed in the Polygon class.
- Degeneracies are not allowed in the Polygon class.

- Repeated vertices:
  - Edge between v3-v4 has zero length.
**FIXDEGENERATE()**

- Degeneracies are not allowed in the Polygon class.

- Repeated vertices:
  - Edge between v3-v4 has zero length.

- Collinear vertices
ISSUE

- Compute polygon-polygon intersections in 3D. Two cases to consider depending on their normals.
ISSUE

- Compute polygon-polygon intersections in 3D. Two cases to consider depending on their normals.
ISSUE

- Compute polygon-polygon intersections in 3D. Two cases to consider depending on their normals.
ISSUE

- Compute polygon-polygon intersections in 3D. Two cases to consider depending on their normals.
ISSUE

- Compute polygon-polygon intersections in 3D. Two cases to consider depending on their normals.
APPROACH

Utilities3D

Polyhedron

- AddPolygons(poly: Polygon, triangulate: bool)

0..* 

+fPolys

Polygeon

Polygon

+ isValid: bool
+ IsPointInside(p: Vector3D): bool
+ IsPointInsideTriangle(p: Vector3D): bool
+ Triangulate(triangles: vector<Polygon>): void
+ CheckAndFixDegenerate(): void
+ Intersect(p2: Polygon): PolygonIntersection*

«struct»

PolygonIntersection

- fPoints: vector<Vector3D>
- fPolygons: vector<Polygon>
- fVertices: vector<Vector3D>

0..*

+fLines

Line

- fPts: Vector3D[2]
- Intersect(l2: Line): LineIntersection
- IsPointOnLine(p: Vector3D): bool

«struct»

LineIntersection

- fA: double
- fB: double

«enumeration»

LineIntersectionType

- fParallel
- fOverlap
- fIntersect
- fNoIntersect
CASE 1:
CASE 1:
CASE 2:
CASE 2:
FOLLOW-UP

● Visualization of complex meshes with OpenGL, when VecGeom is compiled with ROOT support.

● More accurate overlap detection and visualization.

● Detection of volume neighbors and implementation of specialized navigators aware of neighbors
THANK YOU