Programming of 3D Boolean Algebra on Yin Set

Yan Tan

2019 年 12 月 25 日

Contents

What?

- A Yin set $\mathbb{Y} \subseteq \mathbb{R}^3$ is a regular open semianalytic set whose boundary is bounded. The class of all such Yin sets form the Yin space $\mathbb{Y}$.
- Boolean algebra
  - $\widehat{0}$ and $\widehat{1}$ respectively get $\emptyset$ and $\mathbb{R}^3$
  - Complementation $'$.
  - Meet operation $\wedge$.
  - Join operation $\vee$ can be realized by $'$ and $\wedge$.

How?

- Represent Yin sets $\mathbb{Y}$ by their oriented boundary $\mathbb{J}$.
- The boundary of every connected component of a Yin set is an orientable compact surface.
- A partial order exists between these surfaces like the inclusion relation in 2D.
- $\mathbb{J}$ can be approximately represented by a set of oriented triangles $\mathbb{T}$.
- An isomorphism ρ between $\mathbb{T}$ and $\mathbb{Y}$
  - Reduce a 3-dimensional problem into 2-dimensional

Translating Mathematic Concepts to Class

- Point → Class **Point**
- Vector → Class **Direction**
- Straight Line → Class **Line**
- Segment → Class **Segment** : public **Line**
- Flat → Class **Flat**
- Planar → Class **Planar** : public **Flat**
- Oriented and connected face formed by a set of triangles → Class **Face**
- Yin set's boundary → Class **Spadjor**
- Yin set → Class **Object**

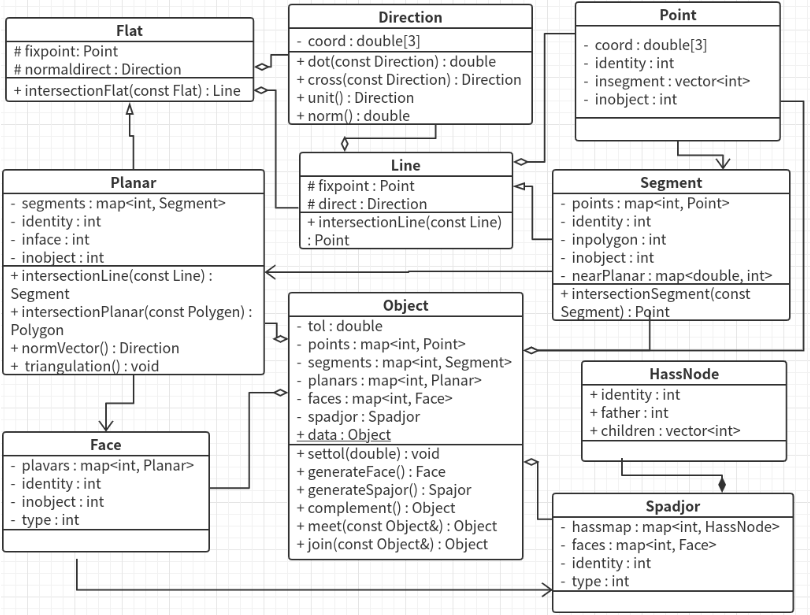
Relations Between These Classes

"has a" :

- **Segment** has two **Point**s that are its endpoint.
- **Planar** has at least three **Segment**s as its boundary.
- **Face** has at least four **Planar**s.
- **Spadjor** has at least one **Face**.

"is a":

- **Segment** is a **Line** and has two **Point**s as endpoints.
- **Planar** is a **Flat** contains some **Segment**s as edges.



Flat
- # fixpoint: Point
- # normaldirect : Direction
- + intersectionFlat(const Flat) : Line

Direction
- - coord : double[3]
- + dot(const Direction) : double
- + cross(const Direction) : Direction
- + unit() : Direction
- + norm() : double

Point
- - coord : double[3]
- - identity : int
- - insegment : vector<int>
- - inobject : int

Planar
- - segments : map<int, Segment>
- - identity : int
- - inface : int
- - inobject : int
- + intersectionLine(const Line) : Segment
- + intersectionPlanar(const Polygen) : Polygon
- + normVector() : Direction
- + triangulation() : void

Line
- # fixpoint : Point
- # direct : Direction
- + intersectionLine(const Line) : Point

Segment
- - points : map<int, Point>
- - identity : int
- - inpolygon : int
- - inobject : int
- - nearPlanar : map<double, int>
- + intersectionSegment(const Segment) : Point

Object
- - tol : double
- - points : map<int, Point>
- - segments : map<int, Segment>
- - planars : map<int, Planar>
- - faces : map<int, Face>
- - spadjor : Spadjor
- + data : Object
- + settol(double) : void
- + generateFace() : Face
- + generateSpajor() : Spajor
- + complement() : Object
- + meet(const Object&) : Object
- + join(const Object&) : Object

HassNode
- + identity : int
- + father : int
- + children : vector<int>

Face
- - plavars : map<int, Planar>
- - identity : int
- - inobject : int
- - type : int

Spadjor
- - hassmap : map<int, HassNode>
- - faces : map<int, Face>
- - identity : int
- - type : int

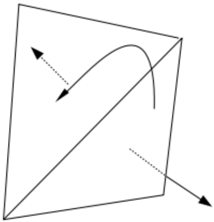
How to Realize Boolean Operations

- Complementation $'$ can be accomplished by changing orientation of boundary of Yin sets $\mathbb{Y}$, we can do that by change planar's orientation.
- Meet operation $\wedge$ can be done by taking the following five operations.
  1. Computing triangles intersection generates some segments and some triangles change into some planars.
  2. Do triangulation for planar.
  3. Determine triangles if still part of the Yin set's boundary after meet operation.
  4. Pasting every triangles to some faces represent Yin set's boundary.
  5. Create a new hassmap represent faces' inclusion relation.

Pasting operation

- Using a Stack st.
  - st's element is Planar that have finished triangulation. So it contains triangles.
- St is empty at first. Choose a triangle hasn't been pasted push in and save it in a empty map<int, Planar> m.
  1. pop the triangle t in the Top of st.
  2. then find triangles that share an edge with t if it hasn't been pasted and should be pasted with t.
  3. if triangle t1 should be pasted, push t1 into st and save it in the map m.
  4. if no triangles should be pasted. Back to first step and continue.
  5. if st becomes empty again, using the map m create a new face.Then break the loop.

- How to determine whether a triangle should be pasted with t.
    - choose an edge e of t.
    - fix e, rotate t in the opposite direction to the normal vector of t until finding the first triangle t1 that has edge e and has normal vector that is opposite to t's. t1 is the only triangle share edge e should be pasted with t.



regular open semianalytic boundary is bounded

Test Work

Test data :

- spheres, torus, n-ple torus.
- two cylinders coincide in a line.
- two tetrahedrons have a triangle planar contain another or coincide in a point.

Test way :

- input data in Obj format.
- output data in Obj format.
- render with ray tracing.