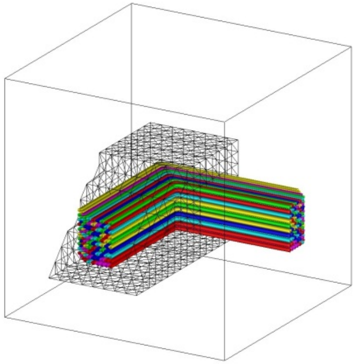
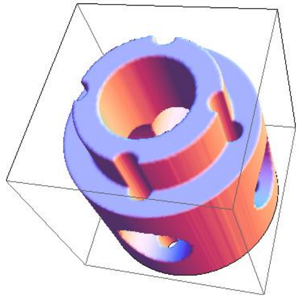
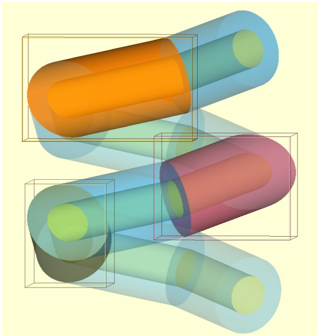
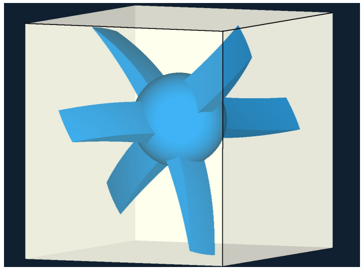




How to Use CS to Become a Nuclear Physicist:
Applying Computational Geometry to Reactor Physics



David L. Millman



Motivation/Background

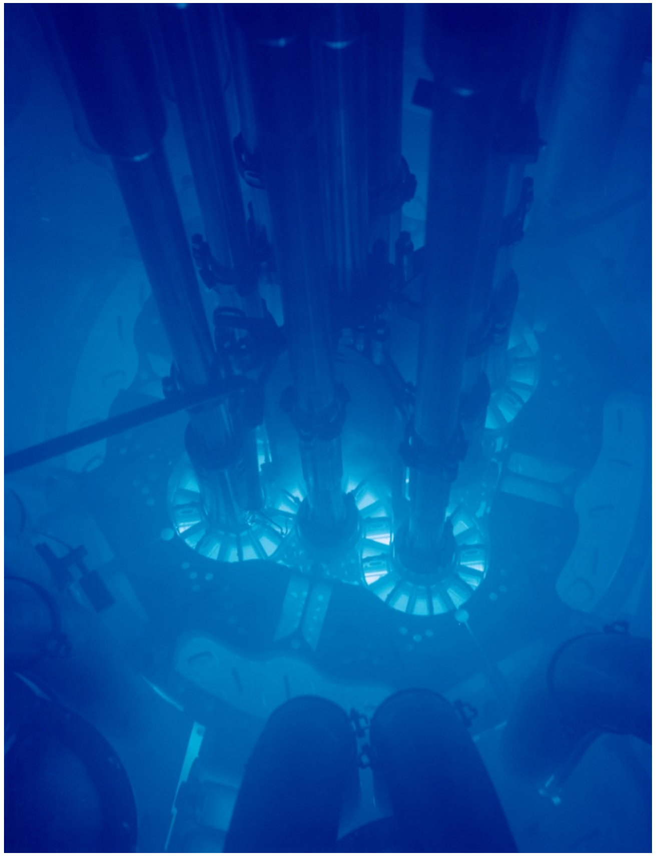
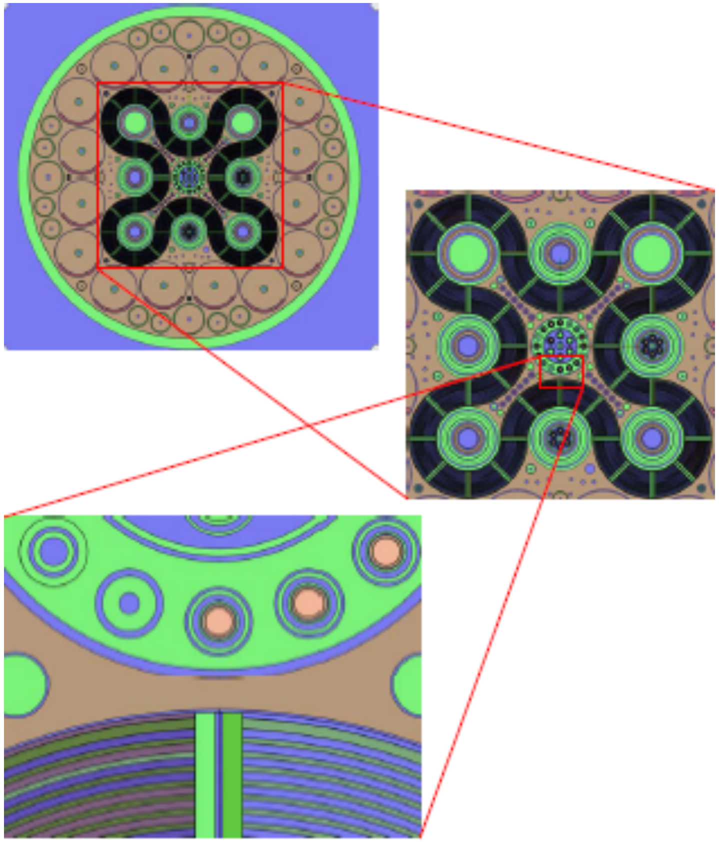


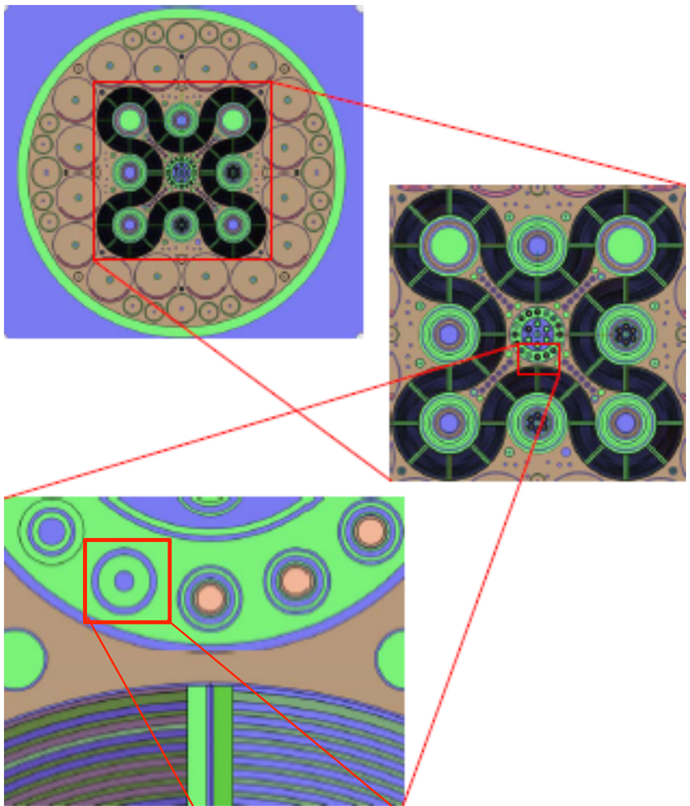
Image from Idaho National Lap, Flickr



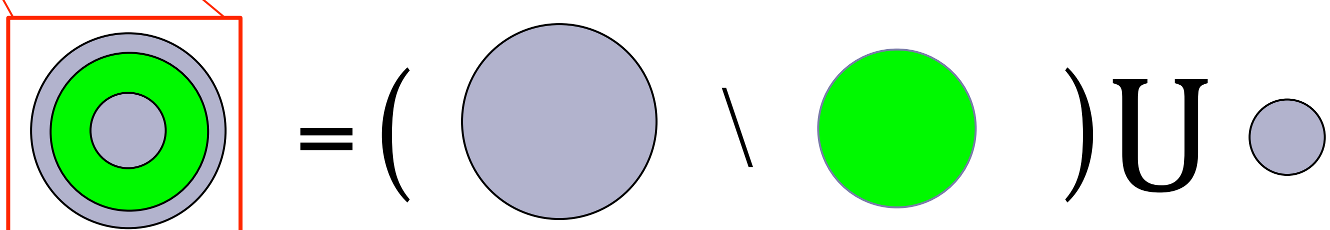
T.M. Sutton, et. al., *The MC21 Monte Carlo Transport Code*, Proceedings of the Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007), Monterey, CA (2007)

Motivation/Background

T.M. Sutton, et. al., *The MC21 Monte Carlo Transport Code*, M&C+SNA 2007

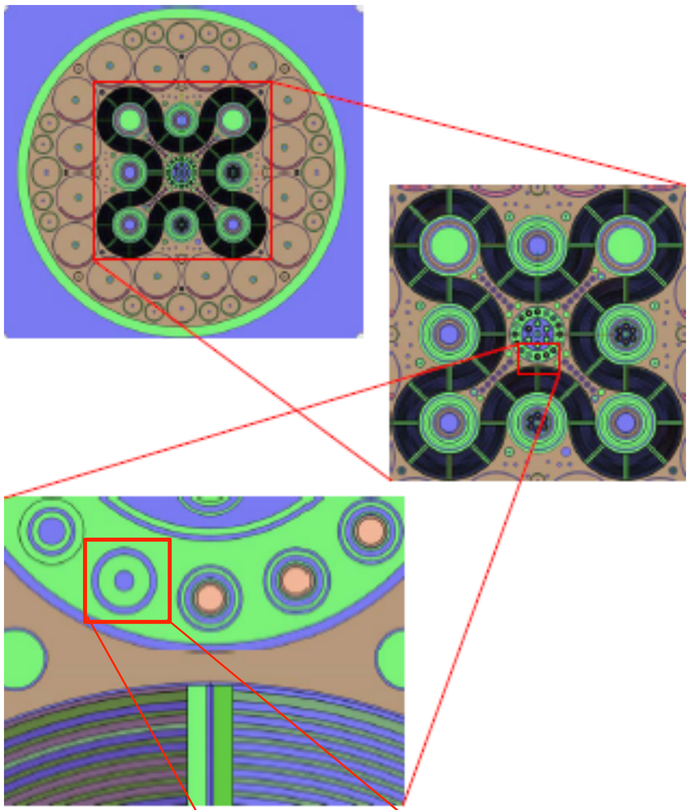


Constructive solid geometry (CSG) is used to define geometric objects in Monte Carlo transport calculations.



Motivation/Background

T.M. Sutton, et. al., *The MC21 Monte Carlo Transport Code*, M&C+SNA 2007

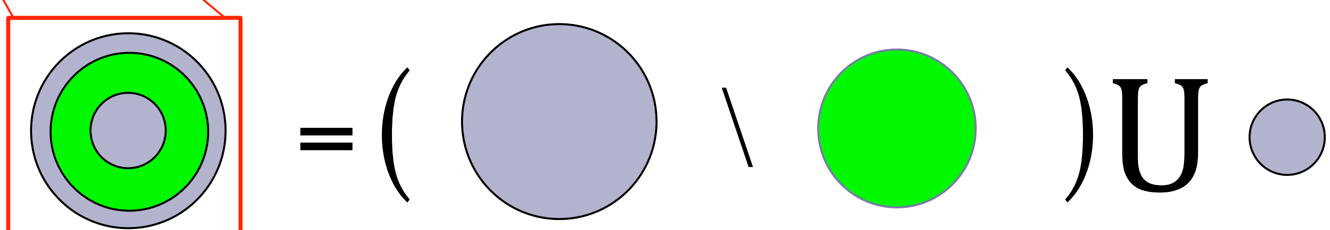


Constructive solid geometry (CSG) is used to define geometric objects in Monte Carlo transport calculations.

CSG provides an exact representation of an objects boundary.

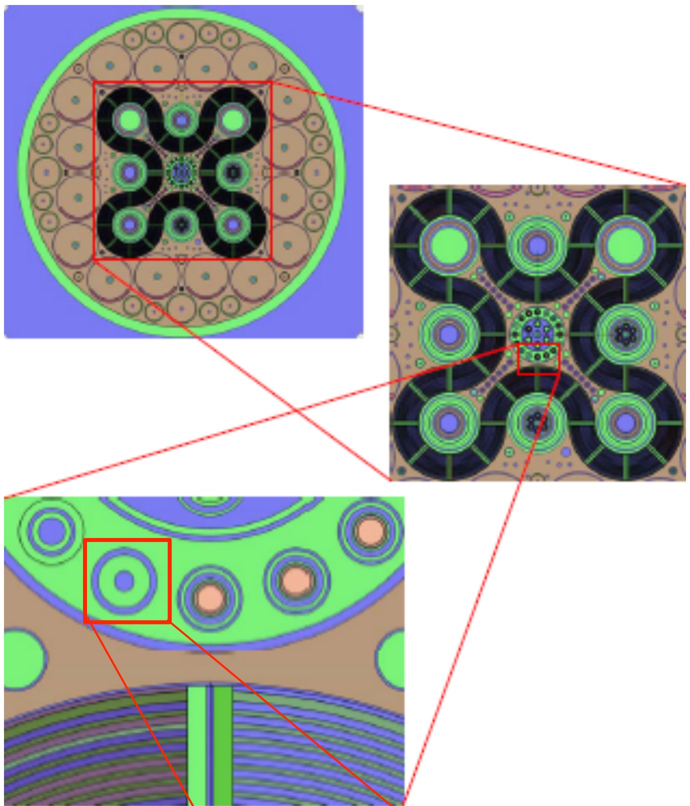
CSG allow nearly unlimited flexibility for creating complex models for:

- Criticality analysis
- Reactor analysis



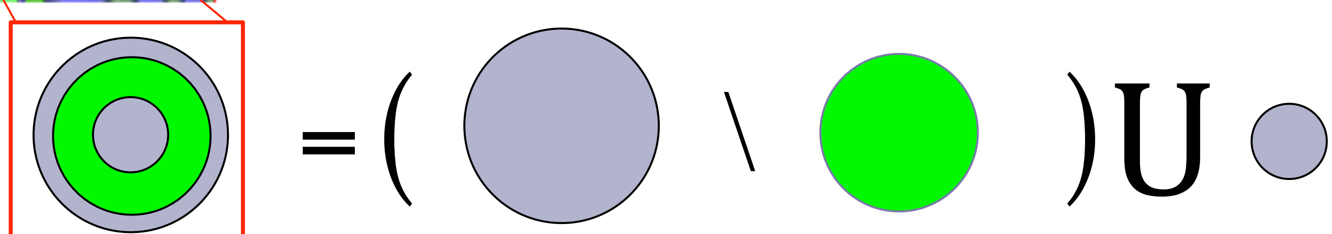
Motivation/Background

T.M. Sutton, et. al., *The MC21 Monte Carlo Transport Code*, M&C+SNA 2007



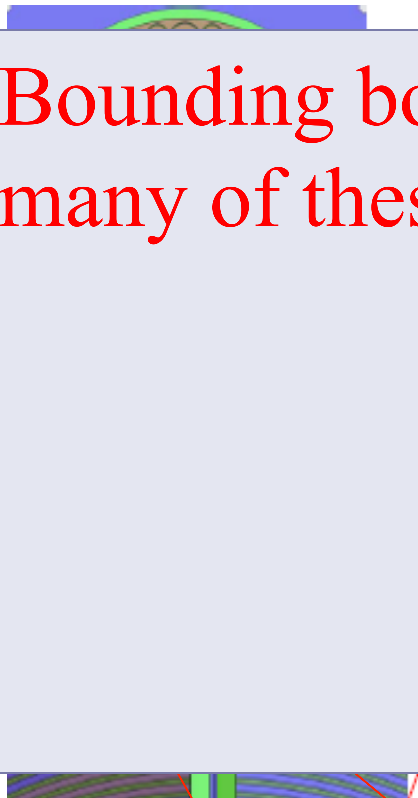
CSG components can be difficult to process. Compared to other representations, for CSG components:

- Particle tracking is slower
- Sampling is more resource intensive
- Properties (such as volume) are difficult to compute



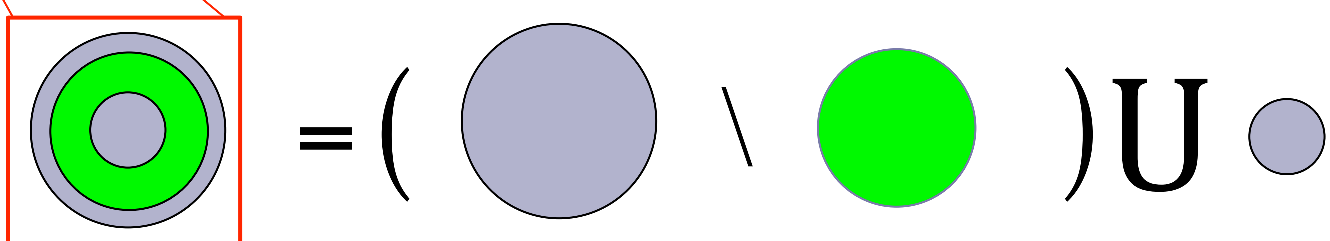
Motivation/Background

T.M. Sutton, et. al., *The MC21 Monte Carlo Transport Code*, M&C+SNA 2007



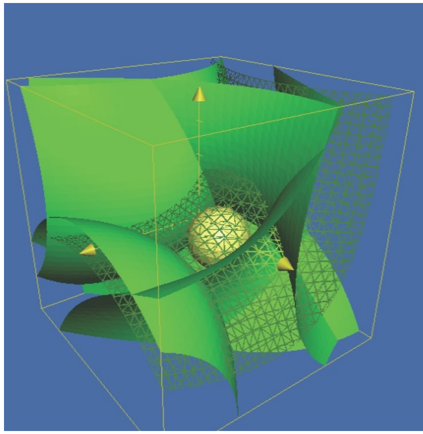
Bounding boxes help solve many of these difficulties….

…unfortunately, computing bounding boxes for CSG components is non-trivial.



Difficulty: Finding the domain.

Basic idea: *Divide-and-conquer*.

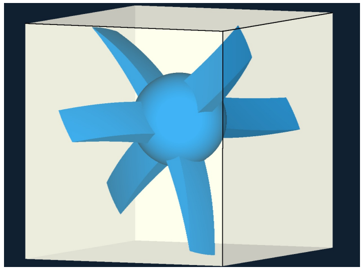


Recursively decompose space into boxes,
determining the surfaces affecting each box,
stopping when the box is small enough
or surfaces are simple enough
that we can approximate a property accurately.

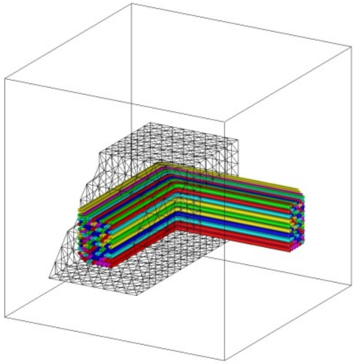
Our contribution: Framework for computing props of each component in a multi-comp. CSG models.

Based on a minimal, extensible set of predicates that handles any model & is very efficient on common cases.

Outline



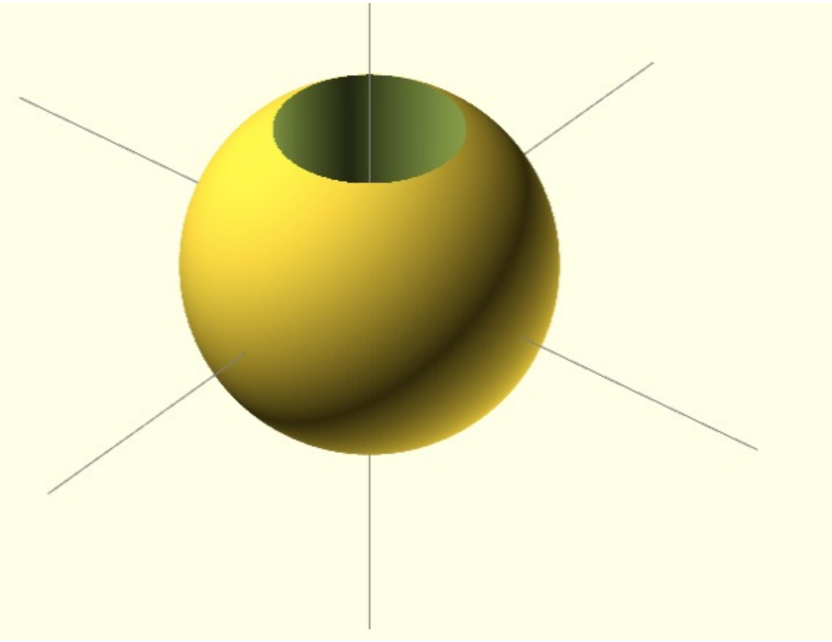
Framework applied to bounding boxes



Framework applied to volumes

What People See

Let D be the region left after drilling a radius r hole through the center of a radius R sphere centered at the origin.



What is the optimal axis-aligned box bounding of D?

Provided $R > r$, a box with:

- minimal point $(-R, -R, -R+f(R,r))$
- maximal point $(R, R, R+f(R,r))$

What the Computer Sees

Let *D* be the intersection of 10 quadratics:

$0 > 0.74742x^2 + 0.93022y^2 + 0.32256z^2 + 0.26590xy +-0.82750xz + 0.43517yz + 2.47974x +26.97936y + 7.15111z +171.27254$

$0 > 0.00487x^2 + 0.00638y^2 + 0.00212z^2 + 0.00181xy +-0.00537xz + 0.00299yz + 0.51989x +-0.07938y + 0.87196z + 36.54138$

$0 <-0.00469x^2 + 0.00617y^2 +-0.00134z^2 + 0.00116xy + 0.00609xz + 0.00326yz + 0.52845x +-0.08488y + 0.86497z +-11.92745$

$0 > 0.00180x^2 + 0.00647y^2 + 0.00497z^2 +-0.00039xy + 0.00597xz + 0.00003yz + 0.59729x +-0.12904y + 0.98774z + 37.27755$

$0 > 0.00173x^2 + 0.00681y^2 + 0.00479z^2 +-0.00022xy + 0.00574xz + 0.00034yz +-0.76442x + 0.12037y + 0.67647z + 27.71845$

$0 > 0.00180x^2 + 0.00657y^2 + 0.00498z^2 +-0.00037xy + 0.00599xz + 0.00008yz +-0.76185x + 0.11119y + 0.68028z + 27.63880$

$0 <-0.00156x^2 + 0.00591y^2 +-0.00403z^2 + 0.00324xy +-0.00503xz + 0.00601yz +-0.90629x + 0.19555y + 0.44420z +-24.48200$

$0 > 0.00643x^2 + 0.00046y^2 + 0.00614z^2 +-0.00143xy +-0.00036xz +-0.00301yz +-0.04751x +-1.00153y +-0.12108z + 11.02481$

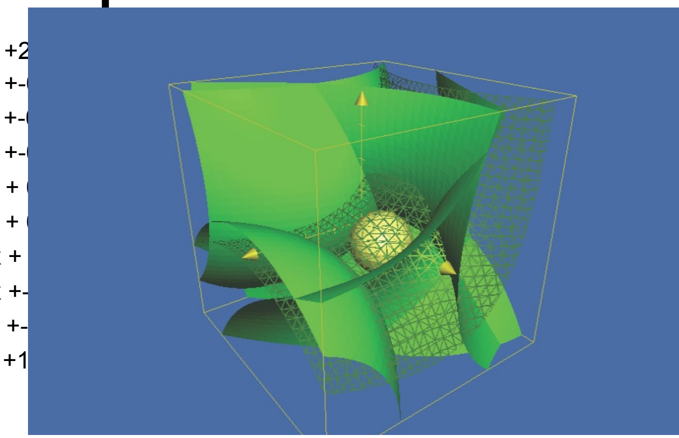
$0 > 0.00323x^2 +-0.00046y^2 +-0.00276z^2 + 0.00209xy +-0.01145xz + 0.00273yz +-0.19156x +-0.92584y +-0.35667z +-40.49961$

$0 < 0.50007x^2 + 0.50004y^2 + 0.50003z^2 + 0.00009xy + 0.00002xz + 0.00004yz + 6.69291x +10.62269y +12.50413z +106.97040$

What the Computer Sees

Let *D* be the intersection of 10 quadratics:

0 > 0.74742x^2 + 0.93022y^2 + 0.32256z^2 + 0.26590xy +-0.82750xz + 0.43517yz + 2.47974x +2
0 > 0.00487x^2 + 0.00638y^2 + 0.00212z^2 + 0.00181xy +-0.00537xz + 0.00299yz + 0.51989x +-
0 <-0.00469x^2 + 0.00617y^2 +-0.00134z^2 + 0.00116xy + 0.00609xz + 0.00326yz + 0.52845x +-
0 > 0.00180x^2 + 0.00647y^2 + 0.00497z^2 +-0.00039xy + 0.00597xz + 0.00003yz + 0.59729x +-
0 > 0.00173x^2 + 0.00681y^2 + 0.00479z^2 +-0.00022xy + 0.00574xz + 0.00034yz +-0.76442x +
0 > 0.00180x^2 + 0.00657y^2 + 0.00498z^2 +-0.00037xy + 0.00599xz + 0.00008yz +-0.76185x +
0 <-0.00156x^2 + 0.00591y^2 +-0.00403z^2 + 0.00324xy +-0.00503xz + 0.00601yz +-0.90629x +
0 > 0.00643x^2 + 0.00046y^2 + 0.00614z^2 +-0.00143xy +-0.00036xz +-0.00301yz +-0.04751x +-
0 > 0.00323x^2 +-0.00046y^2 +-0.00276z^2 + 0.00209xy +-0.01145xz + 0.00273yz +-0.19156x +-
0 < 0.50007x^2 + 0.50004y^2 + 0.50003z^2 + 0.00009xy + 0.00002xz + 0.00004yz + 6.69291x +1



From a picture, we can determine the bounding box without trouble.

Not so easy for a collection of polynomials.

Computing Bounding Boxes Is Difficult

Alg 1: Apply set operations to the bounding boxes of primitives.

"for difference and intersection operations this will hardly ever lead to an optimal bounding box."

–POV-Ray documentation

Alg 2: Convert CSG to boundary rep.

"efficient, accurate, and robust computation of the boundary remains a hard problem for CSG model described using curved primitives."

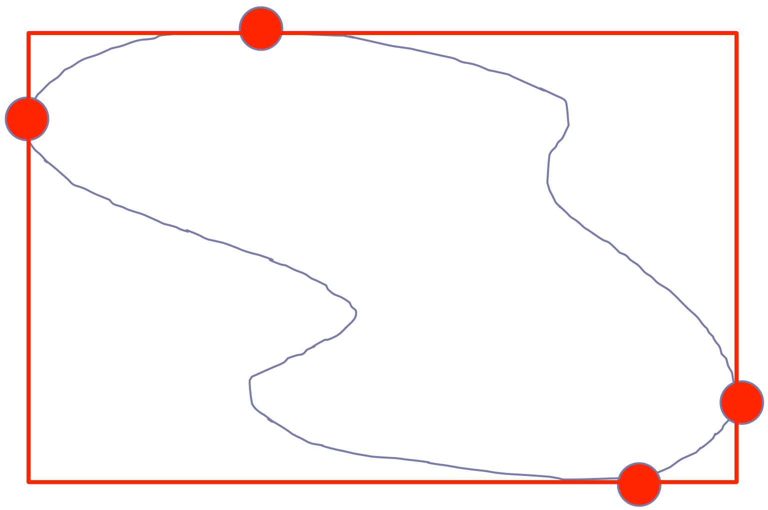
– Lin & Gottschalk [SG98]

More recent work indicates converting CSG to boundary rep is still hard. [K00], [MTT05], [SW06], [DLL+08]

Computing an AABB

Question: Given a domain *D*, compute the optimal axis-aligned bounding box (AABB) of *D*?

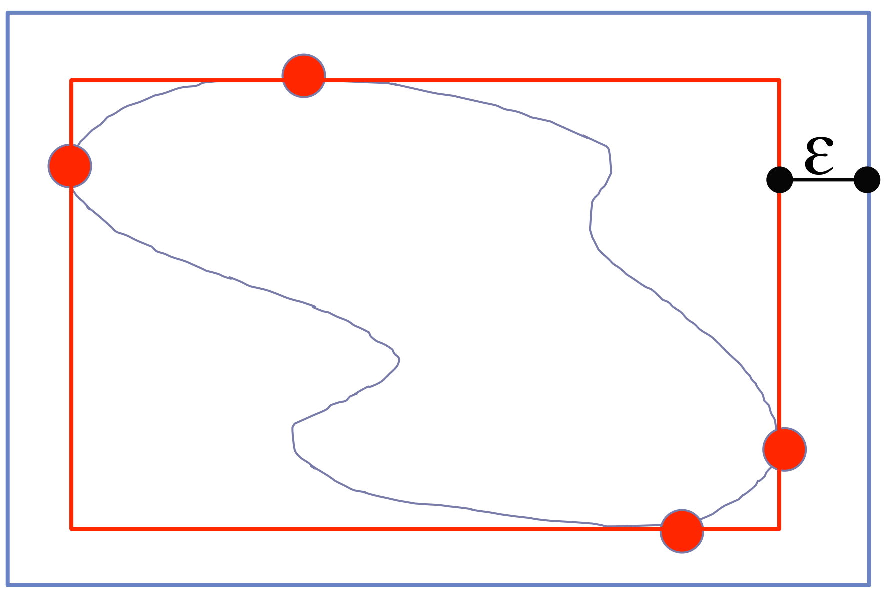
Observation: To compute optimal AABB we compute extremal points in each direction.



Ask an easier question

Given ε > 0, *ε-box* for *D* is an axis-aligned bounding box that is at most ε larger in each direction than the optimal axis aligned bounding box for *D.*

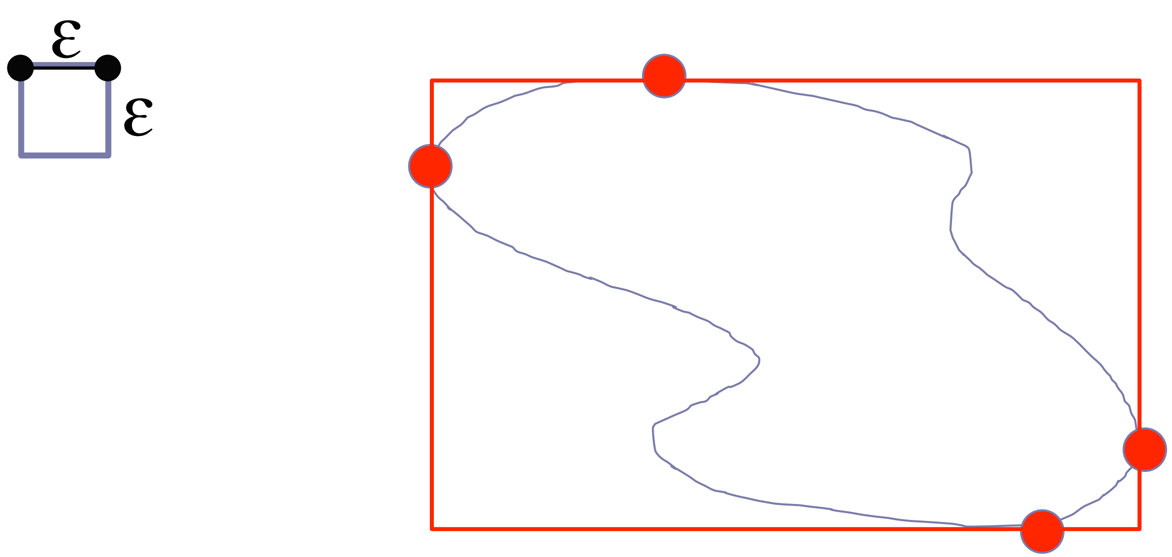
We call an *ε-box* a *numerically-optimal* bounding box.



Ask an easier question

Question: Given a domain *D*, compute the *ε-box* of *D*?

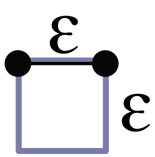
Observation: To compute *ε-box* for *D* we must identify boxes of size *ε* containing an extremal point.

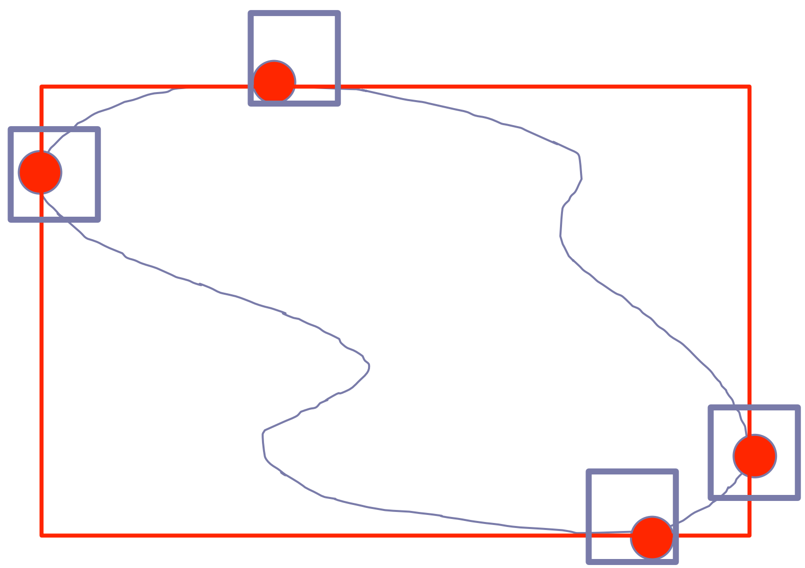


Ask an easier question

Question: Given a domain *D*, compute the *ε-box* of *D*?

Observation: To compute *ε-box* for *D* we must identify boxes of size *ε* containing an extremal point.

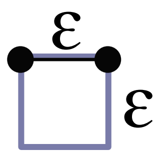


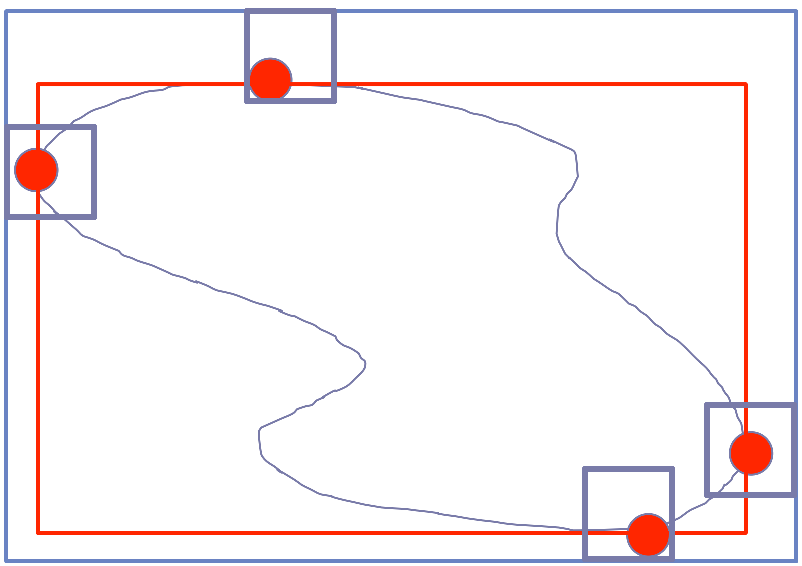


Ask an easier question

Question: Given a domain *D*, compute the *ε-box* of *D*?

Observation: To compute *ε-box* for *D* we must identify boxes of size ε containing an extremal point.





Ask an easier question

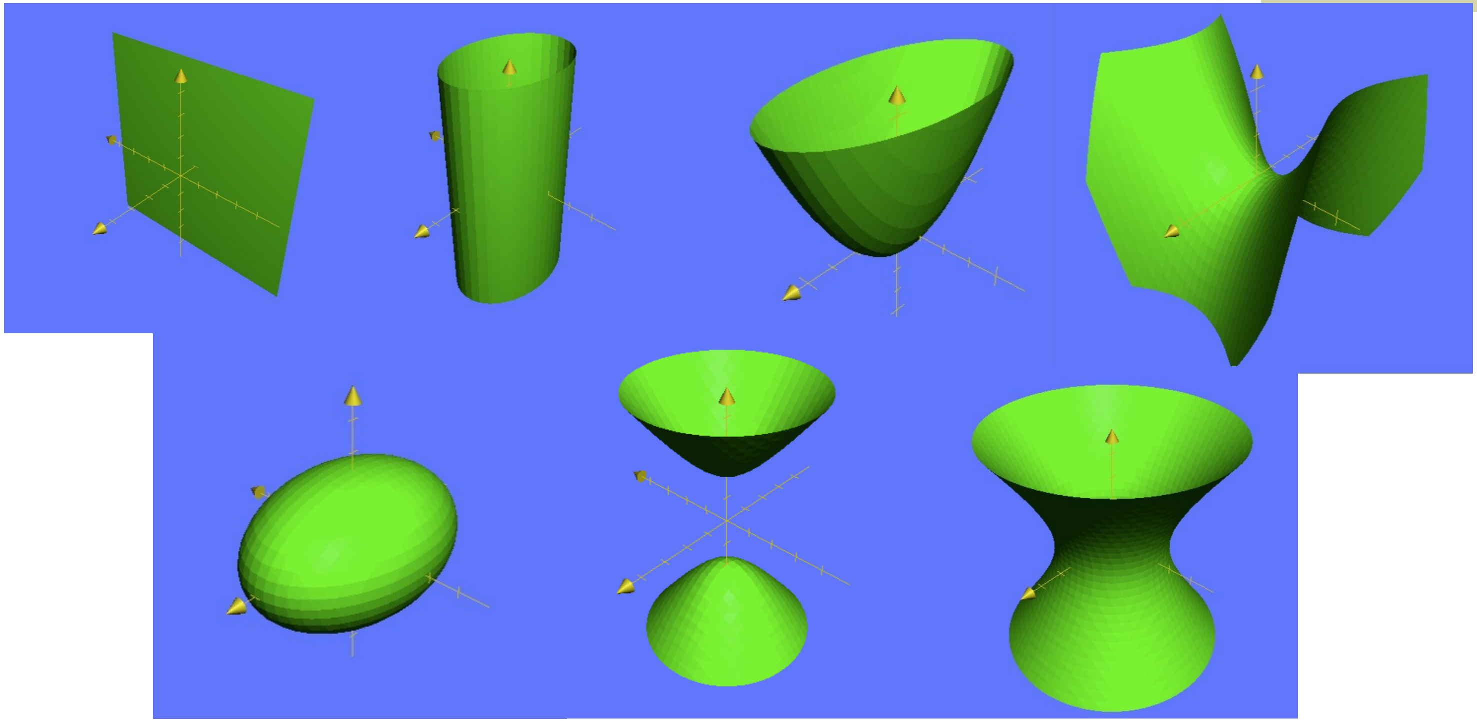
Question: Given a domain *D*, compute the *ε-box* of *D*?

Observation: To compute *ε-box* for *D* we must identify boxes of size *ε* containing an extremal point.

How do we identify boxes of size *ε* containing an extremal point?**

**In this talk, I describe a simpler version of our algorithm that is good in practice but does not have provably tight bounds. See our paper for the gory details of the full algorithm and the proofs!

Primitives: Signed Quadratic Surfaces

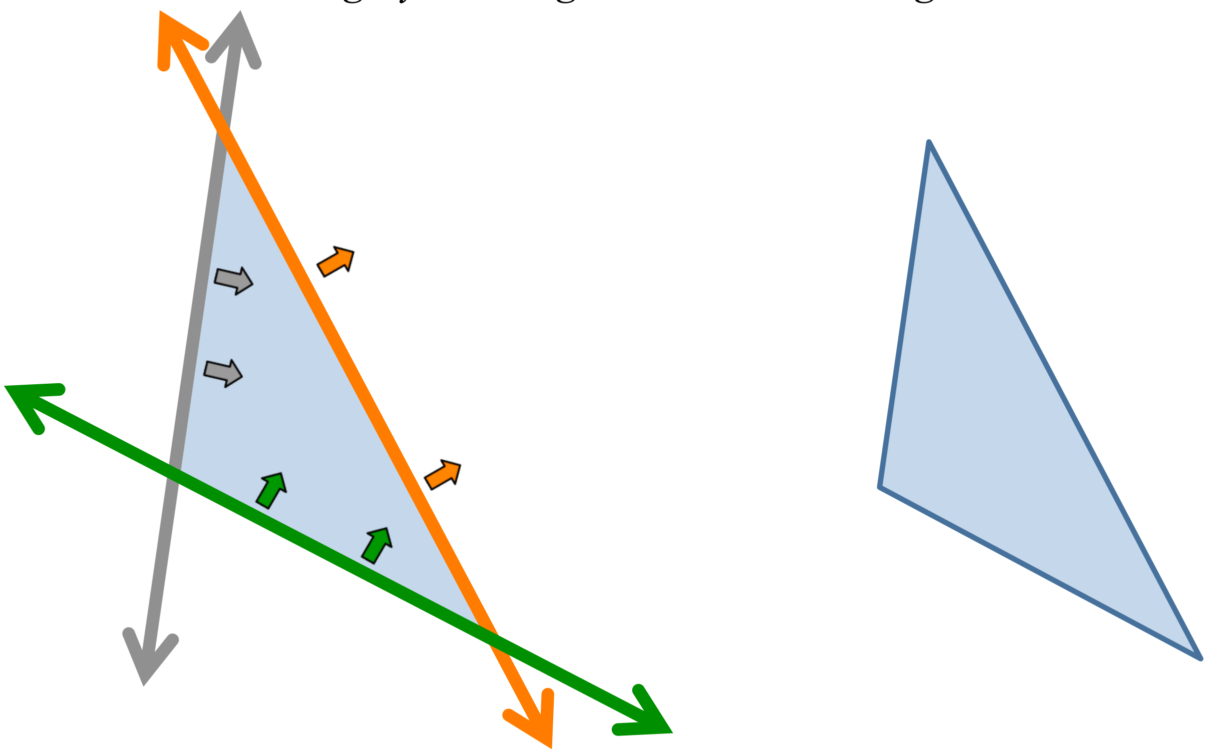


$$f(x, y, z) = Ax^2 + By^2 + Cz^2 + Dxy + Exz + Fyz + Gx + Hy + Iz + J$$

Model Representation Component: Boolean Formula

A *component* defined by intersections and unions of signed surfaces

$$S_{grey} \cap S_{green} \cap -S_{orange}$$



Algorithm Overview (D&C)

(1) Given an initial (very large) bounding box

(2) Traverse an octree:

(a) Subdivide initial box into sub-boxes

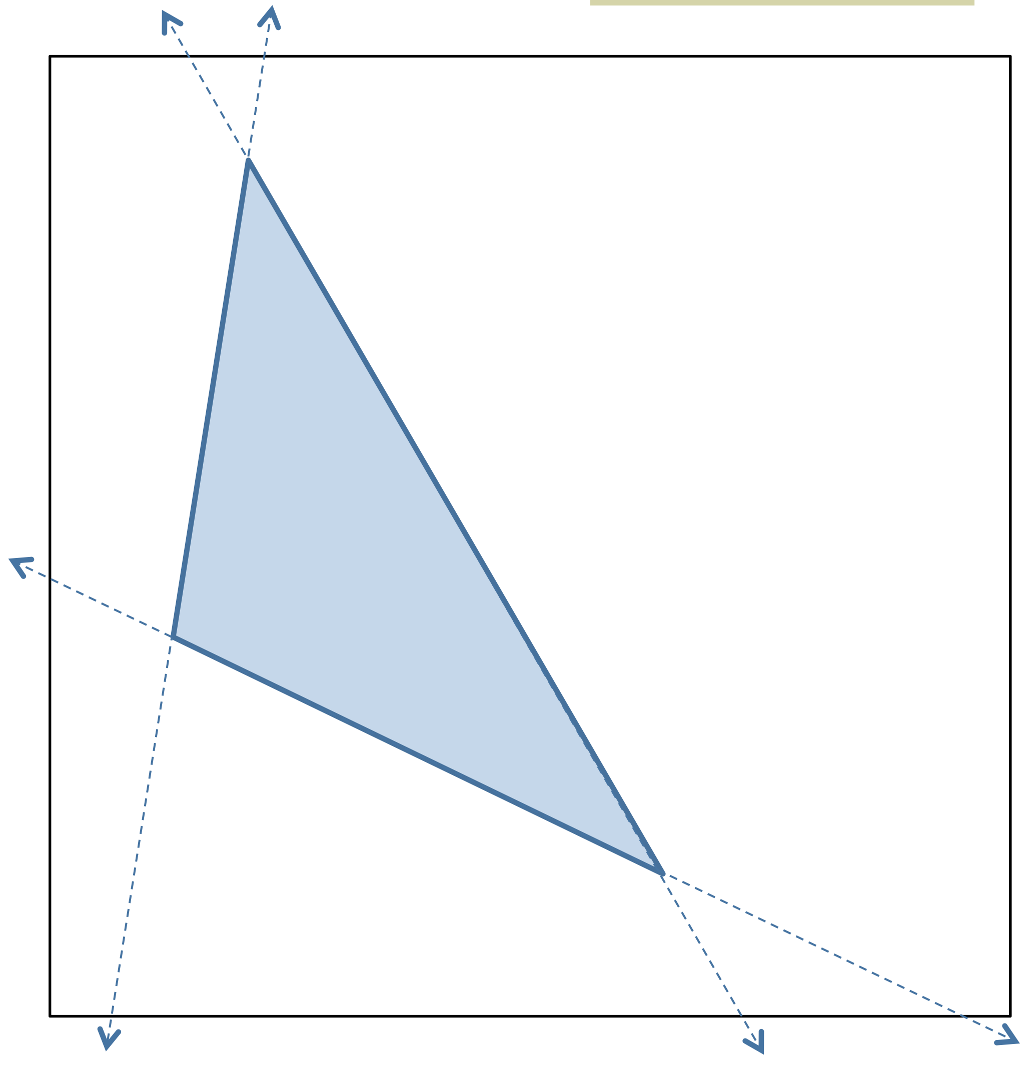
(b) For each sub-box:

(i) *classify* sub-box as *Inside*, *Outside*, *Boundary*, or *Unknown*

(ii) subdivide *Unknown* & *Bounday* sub-boxes

…

(3) Terminates once ε-box is found



Algorithm Overview (D&C)

(1) Given an initial (very large) bounding box

(2) Traverse an octree:

(a) Subdivide initial box into sub-boxes

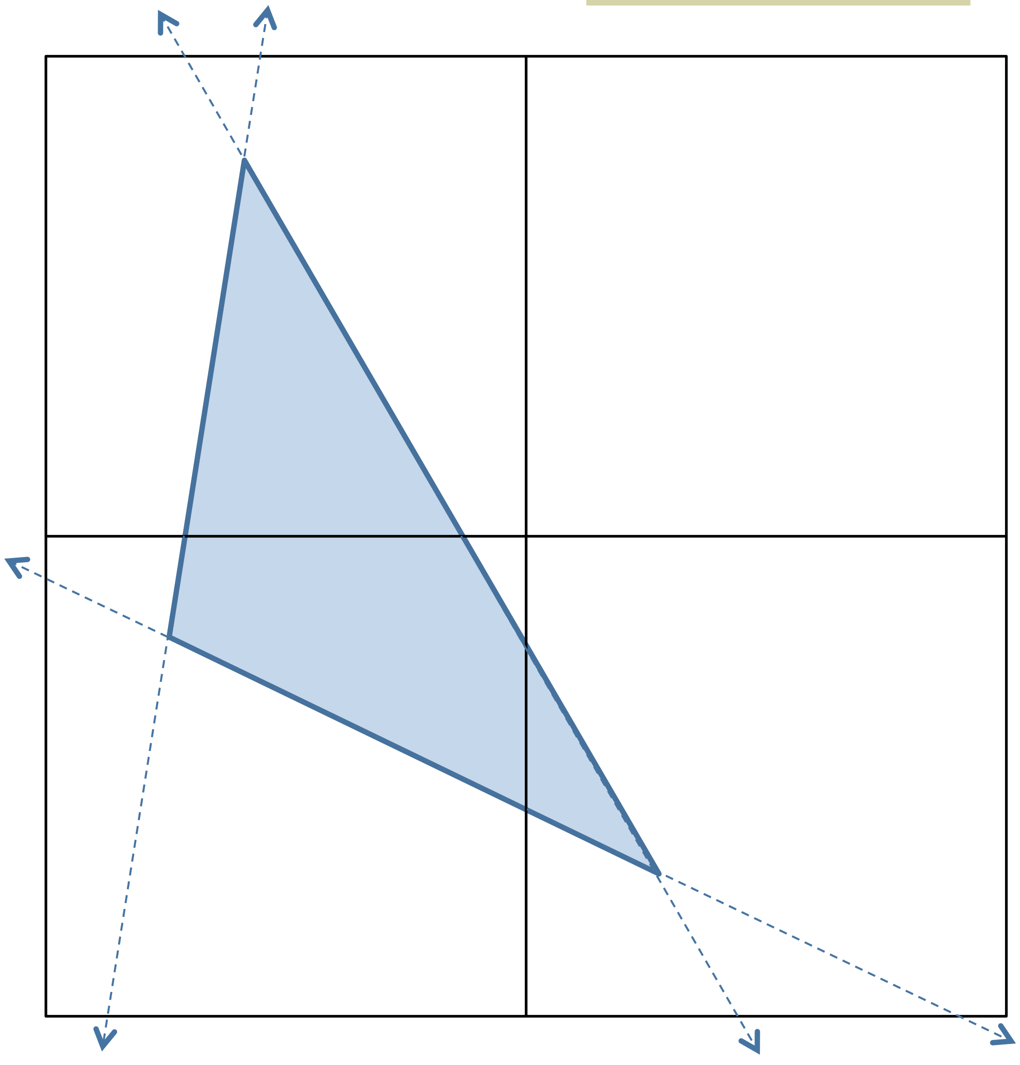
(b) For each sub-box:

(i) *classify* sub-box as *Inside*, *Outside*, *Boundary*, or *Unknown*

(ii) subdivide *Unknown* & *Bounday* sub-boxes

…

(3) Terminates once ε-box is found



Algorithm Overview (D&C)

(1) Given an initial (very large) bounding box

(2) Traverse an octree:

(a) Subdivide initial box into sub-boxes

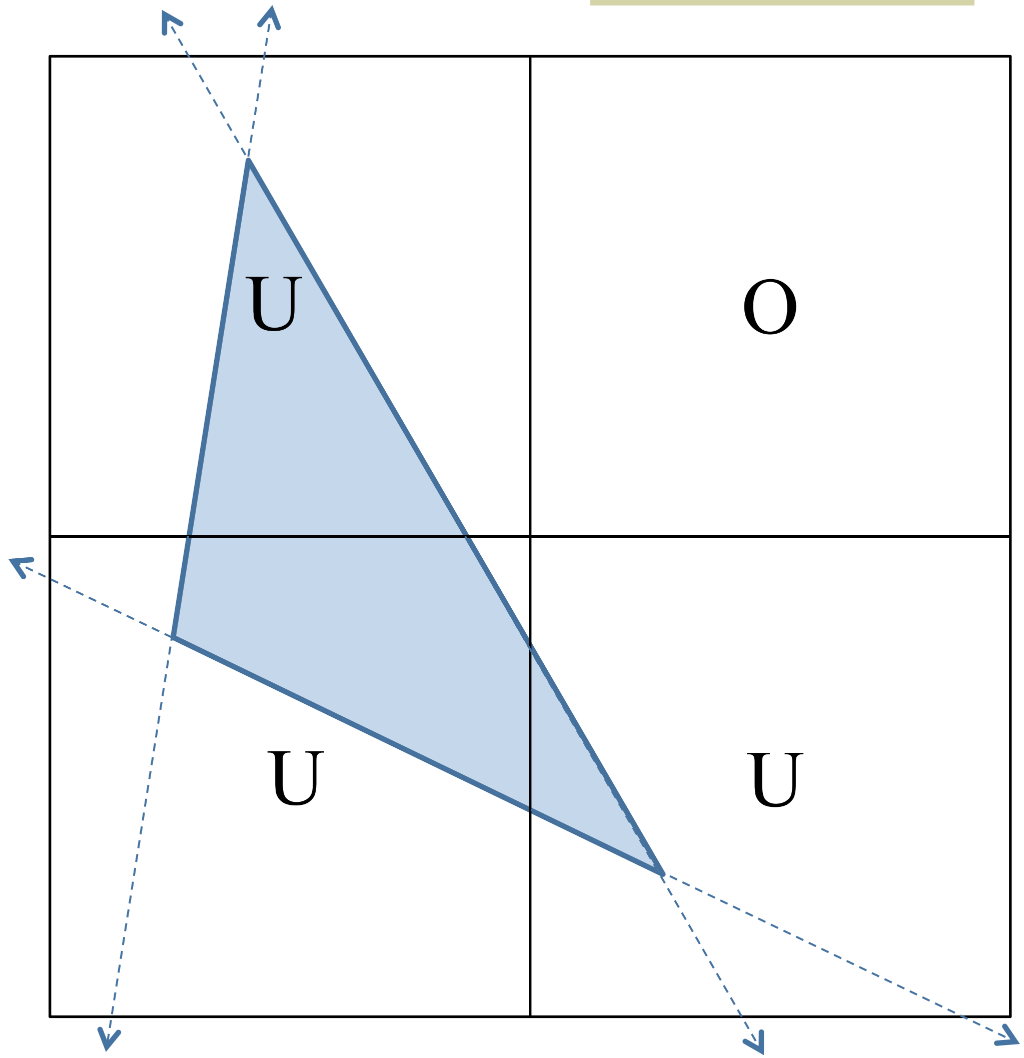
(b) For each sub-box:

(i) *classify* sub-box as *Inside*, *Outside*, *Boundary*, or *Unknown*

(ii) subdivide *Unknown* & *Bounday* sub-boxes

...

(3) Terminates once ε-box is found



Algorithm Overview (D&C)

(1) Given an initial (very large) bounding box

(2) Traverse an octree:

  (a) Subdivide initial box into sub-boxes

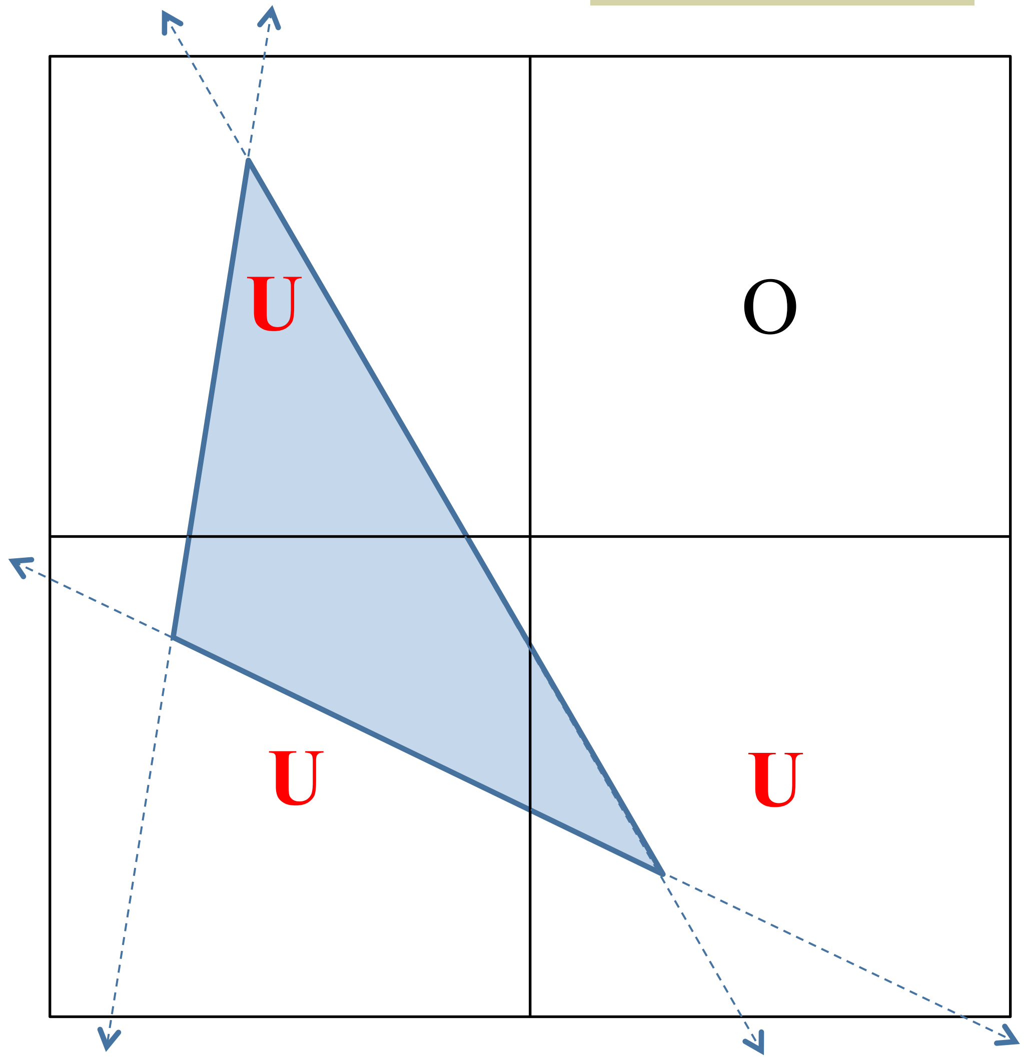
  (b) For each sub-box:

    (i) *classify* sub-box as *Inside*, *Outside*, *Boundary*, or *Unknown*

    (ii) subdivide *Unknown* & *Bounday* sub-boxes

…

(3) Terminates once ε-box is found



Algorithm Overview (D&C)

(1) Given an initial (very large) bounding box

(2) Traverse an octree:

(a) Subdivide initial box into sub-boxes

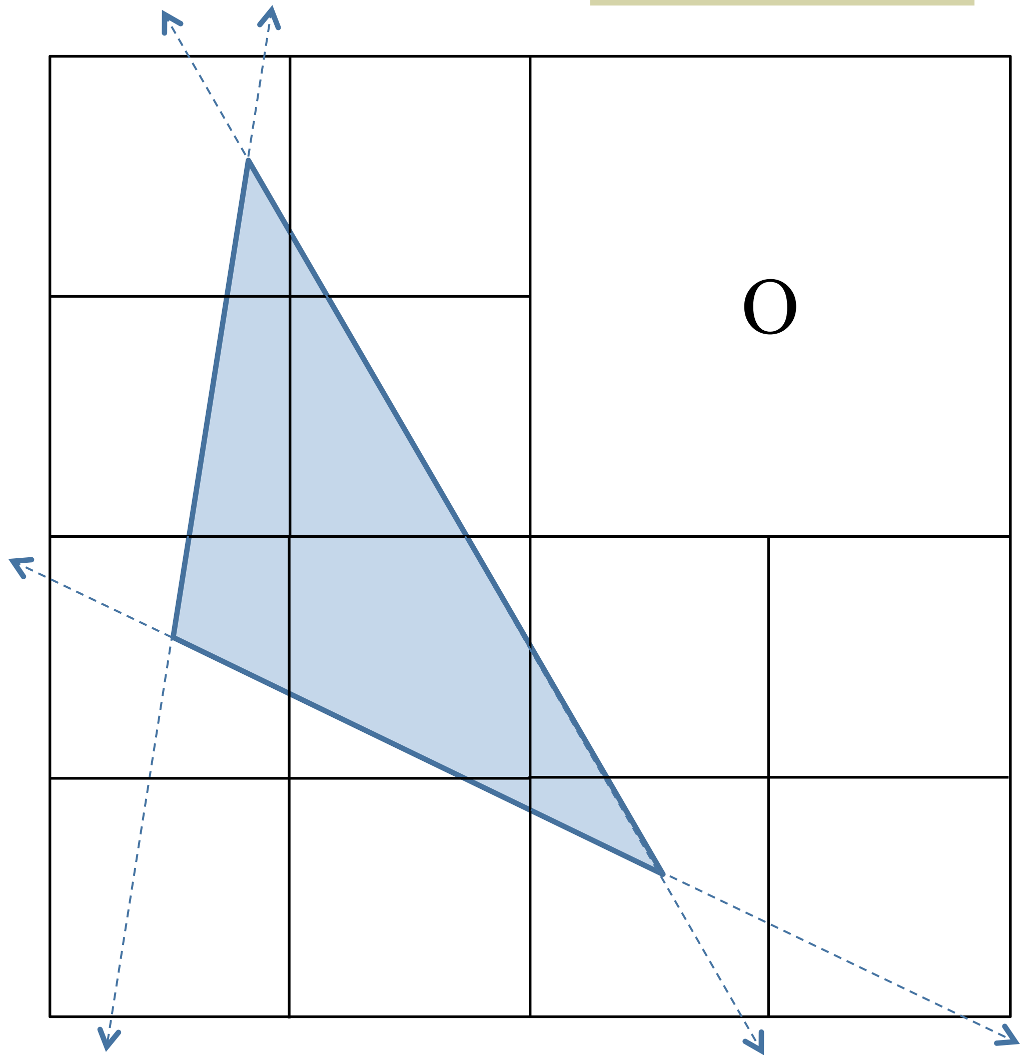
(b) For each sub-box:

(i) *classify* sub-box as *Inside*, *Outside*, *Boundary*, or *Unknown*

(ii) subdivide *Unknown* & *Bounday* sub-boxes

…

(3) Terminates once ε-box is found



Algorithm Overview (D&C)

(1) Given an initial (very large) bounding box

(2) Traverse an octree:

(a) Subdivide initial box into sub-boxes

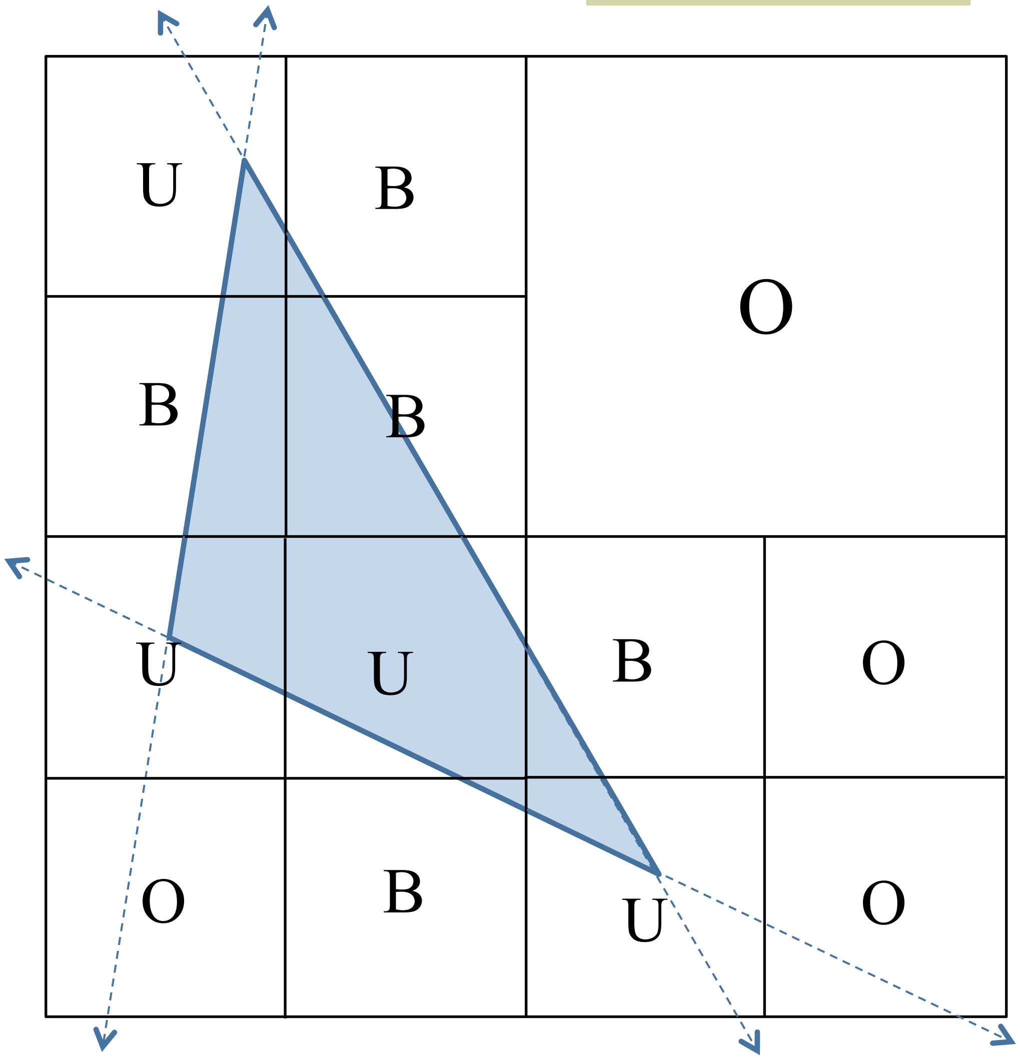
(b) For each sub-box:

(i) *classify* sub-box as *Inside*, *Outside*, *Boundary*, or *Unknown*

(ii) subdivide *Unknown* & *Bounday* sub-boxes

…

(3) Terminates once ε-box is found



Algorithm Overview (D&C)

(1) Given an initial (very large) bounding box

(2) Traverse an octree:

  (a) Subdivide initial box into sub-boxes

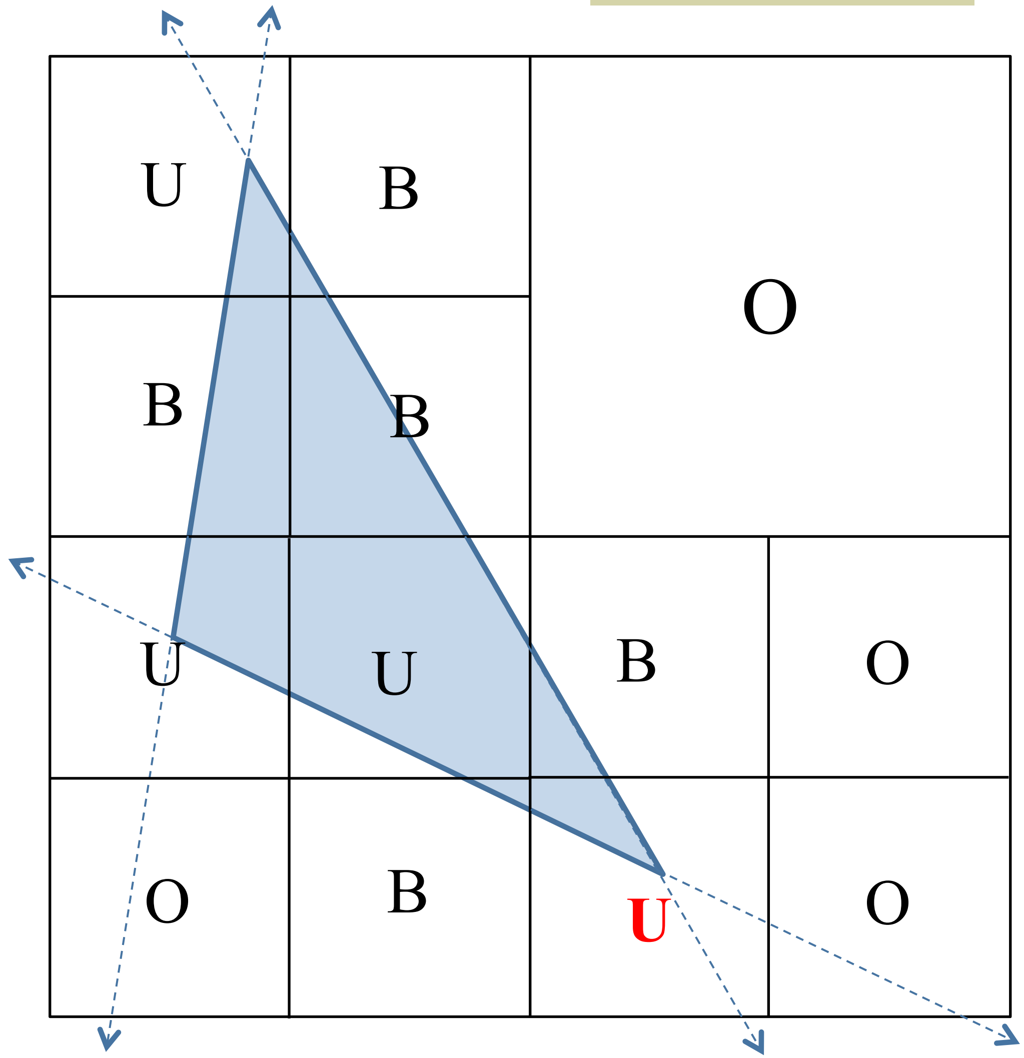
  (b) For each sub-box:

    (i) *classify* sub-box as *Inside*, *Outside*, *Boundary*, or *Unknown*

    (ii) subdivide *Unknown* & *Bounday* sub-boxes

…

(3) Terminates once ε-box is found



Algorithm Overview (D&C)

(1) Given an initial (very large) bounding box

(2) Traverse an octree:

(a) Subdivide initial box into sub-boxes

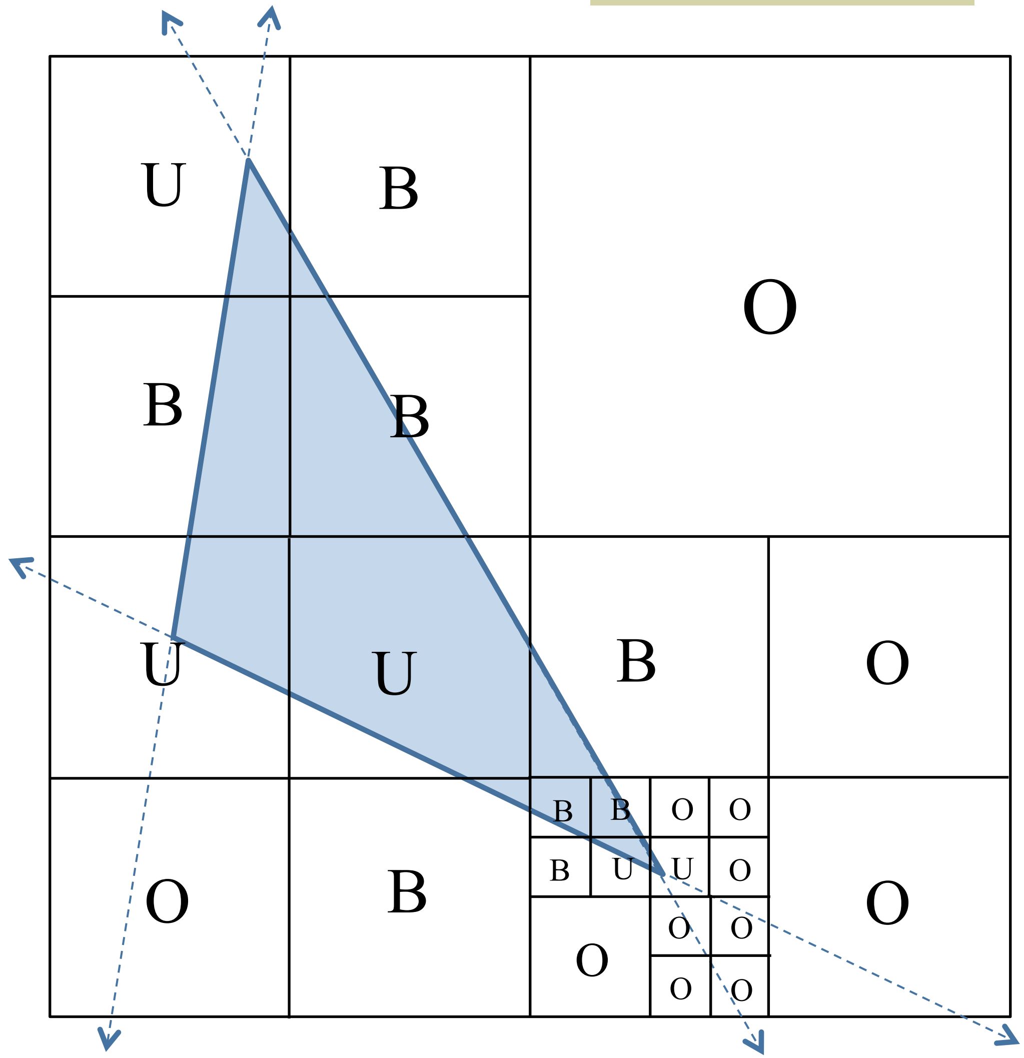
(b) For each sub-box:

(i) *classify* sub-box as *Inside*, *Outside*, *Boundary*, or *Unknown*

(ii) subdivide *Unknown* & *Bounday* sub-boxes

…

(3) Terminates once ε-box is found



Algorithm Overview (D&C)

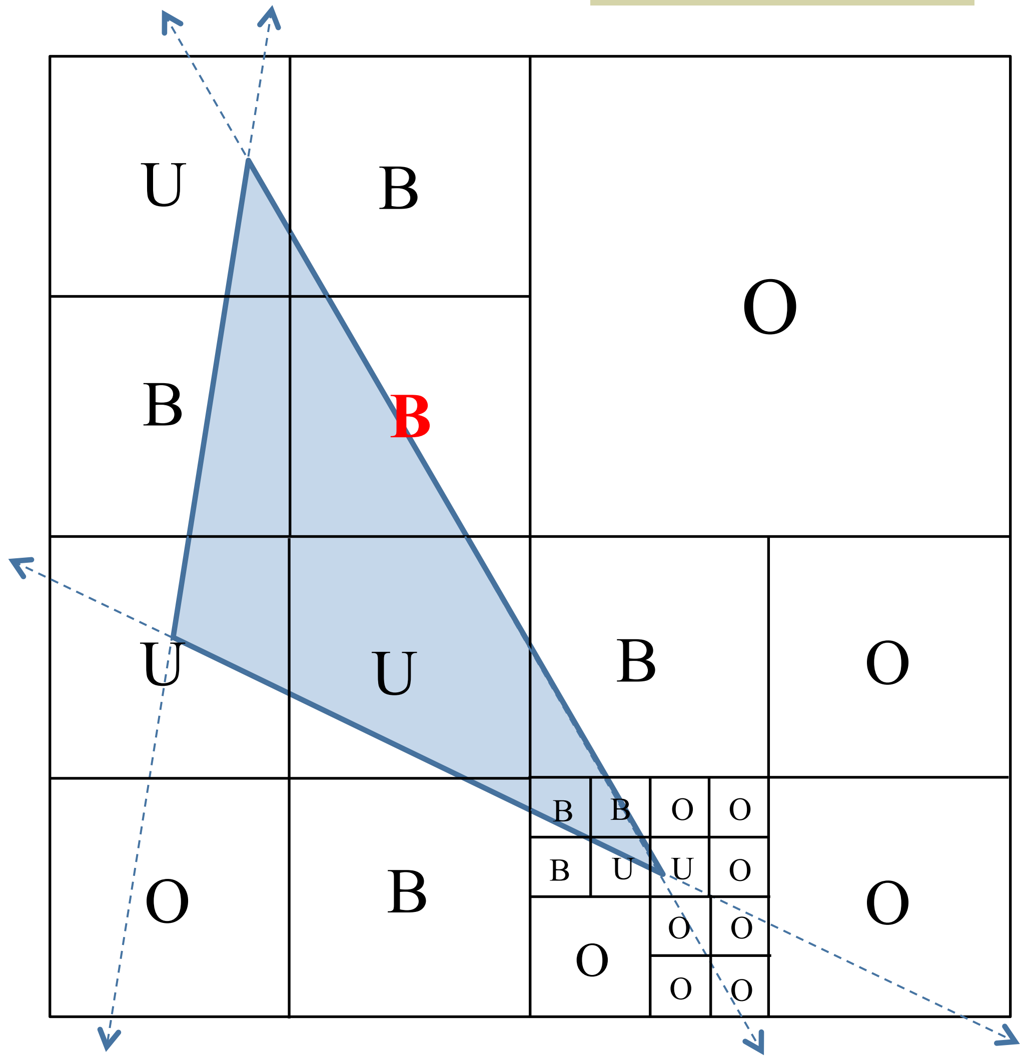
(1) Given an initial (very large) bounding box

(2) Traverse an octree:

- (a) Subdivide initial box into sub-boxes
- (b) For each sub-box:
  - (i) *classify* sub-box as *Inside*, *Outside*, *Boundary*, or *Unknown*
  - (ii) subdivide *Unknown* & *Bounday* sub-boxes

…

(3) Terminates once ε-box is found



Algorithm Overview (D&C)

(1) Given an initial (very large) bounding box

(2) Traverse an octree:

(a) Subdivide initial box into sub-boxes

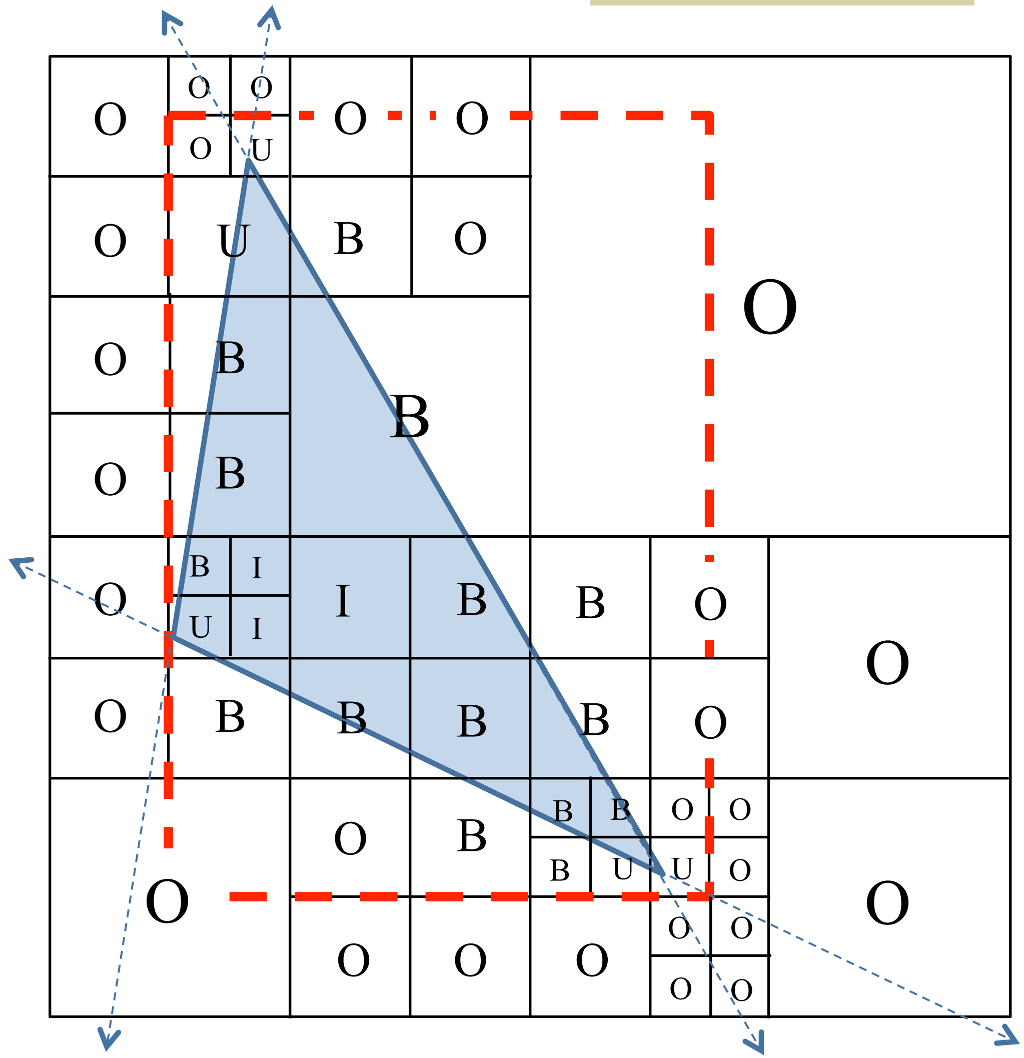
(b) For each sub-box:

(i) *classify* sub-box as *Inside*, *Outside*, *Boundary*, or *Unknown*

(ii) subdivide *Unknown* & *Bounday* sub-boxes

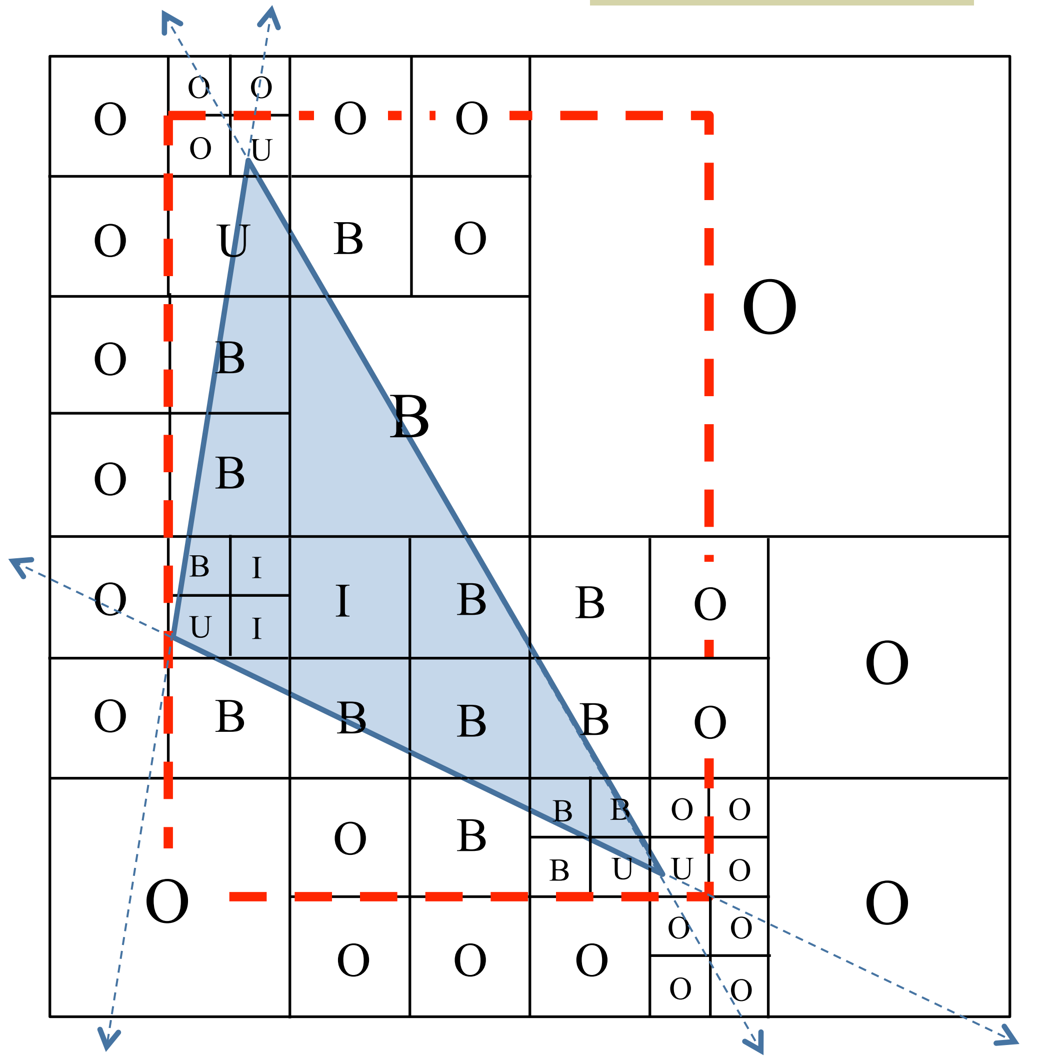
…

(3) Terminates once ε-box is found



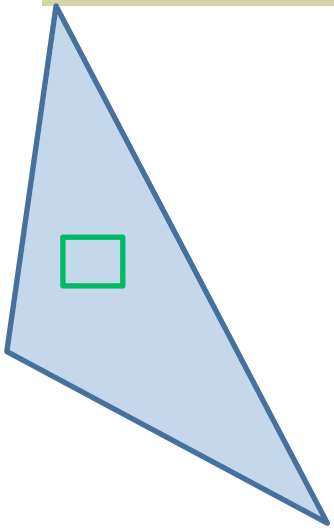
Algorithm Overview (D&C)

The crux
of this algorithm is
the *classify* operation



The *classify* Operation Overview

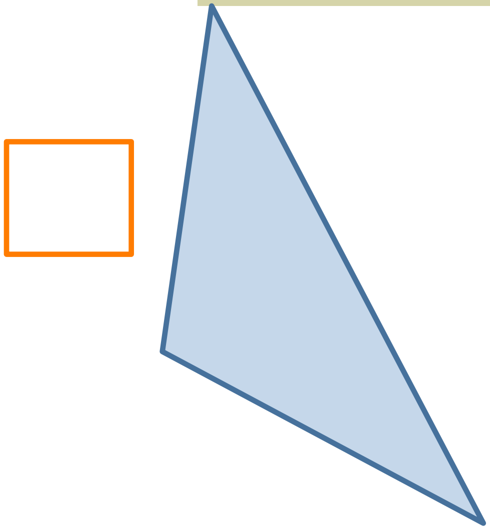
Given comp *C* and axis-aligned box *B*, *classify(C,B)*, returns:



- ***Inside***
- *Outside*
- *Boundary*
- *Unknown*

The *classify* Operation Overview

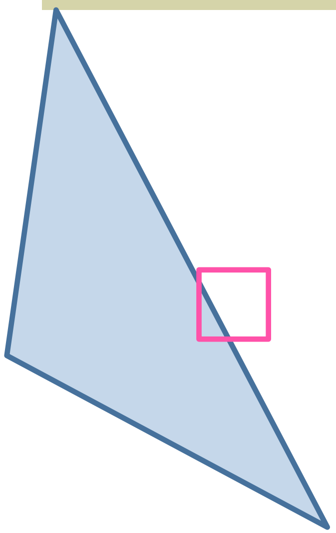
Given comp *C* and axis-aligned box *B*, *classify(C,B)*, returns:



- *Inside*
- ***Outside***
- *Boundary*
- *Unknown*

The *classify* Operation Overview

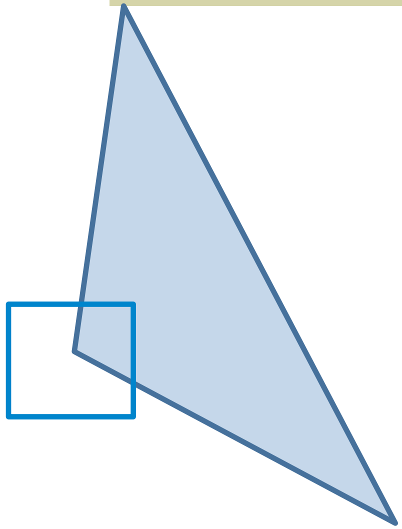
Given comp *C* and axis-aligned box *B*, *classify(C,B)*, returns:



- *Inside*
- *Outside*
- ***Boundary***
- *Unknown*

The *classify* Operation Overview

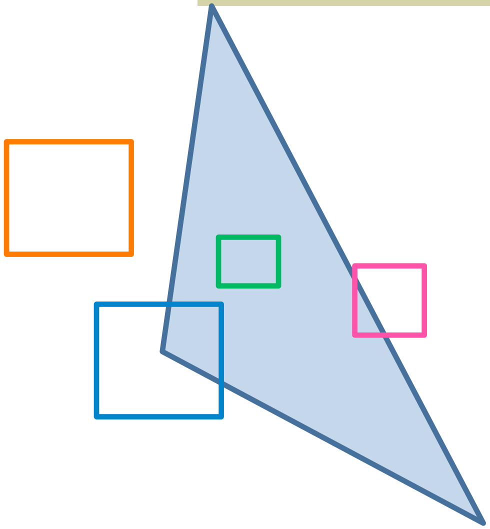
Given comp *C* and axis-aligned box *B*, *classify(C,B)*, returns:



- *Inside*
- *Outside*
- *Boundary*
- ***Unknown***

The *classify* Operation

Given comp *C* and axis-aligned box *B*, *classify(C,B)*, returns:

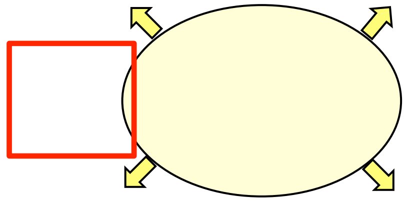


- *Inside* $\Rightarrow B \subseteq C$
- *Outside* $\Rightarrow B \cap C = \varnothing$
- *Boundary* $\Rightarrow \exists$ points $\mathrm{p}, \mathrm{q} \in B$ with $\mathrm{p} \in \mathrm{C}$ and $\mathrm{q} \notin \mathrm{C}$
- *Unknown* $\Rightarrow$ could not classify

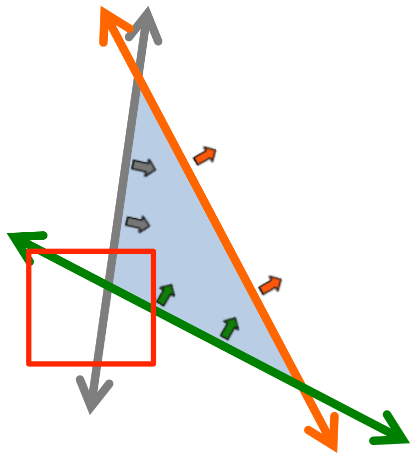
Operations used for *classify*

Let *b* be an axis aligned box:

- *boxLabel* – given a surface *S*, return if the points of *b* are inside, outside, or both with respect to *S*.

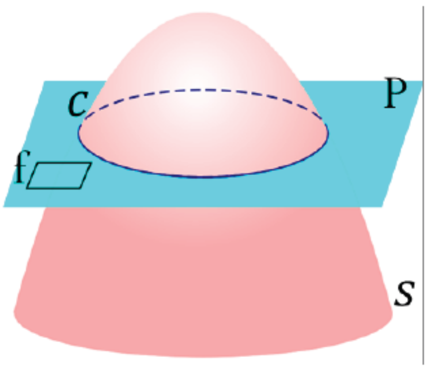


- *formulaRestriction* – given a Boolean formula *G* and the classification for all surfaces of *G* for *b*, replace all surfaces of *G* in which *b* is completely inside or outside with ***T*** or ***F*** and simplify.



The *boxLabel* Operation [M12]

Test if a face f intersects s.



Let c be the intersection curve of the plane P containing f and s.

$$c(x,y) = \begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} ① & ① & ② \\ ① & ① & ② \\ ② & ② & ③ \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

To determine if s intersects f, test properties of the matrix.

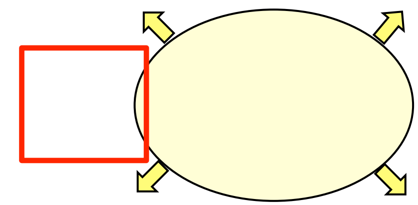
Test if c is an ellipse: $\operatorname{sign}\left(\begin{vmatrix} ① & ① \\ ① & ① \end{vmatrix} \right) = \operatorname{sign}(②)$

Test if c is real or img: $\operatorname{sign}\left(\begin{vmatrix} ① & ① & ② \\ ① & ① & ② \\ ② & ② & ③ \end{vmatrix} \right) = \operatorname{sign}(⑤)$

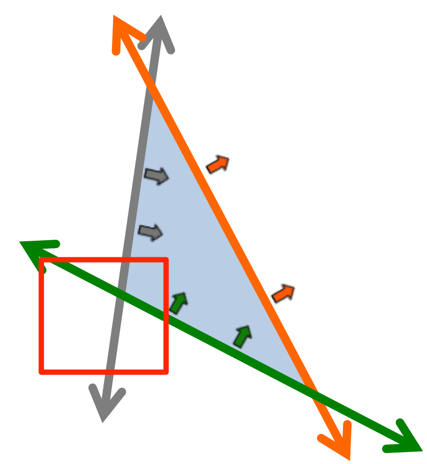
Operations used for *classify*

Let *b* be an axis aligned box:

- *boxLabel* – given a surface *S*, return if the points of *b* are inside, outside, or both with respect to *S*.

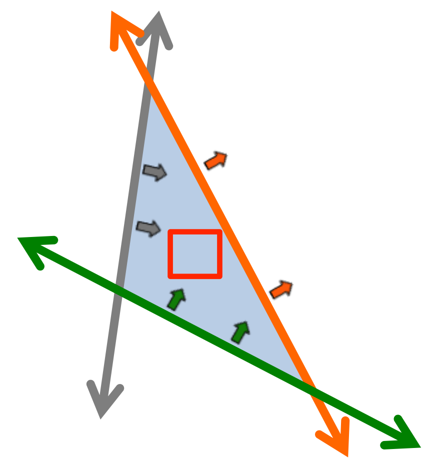


- *formulaRestriction* – given a Boolean formula *G* and the classification for all surfaces of *G* for *b*, replace all surfaces of *G* in which *b* is completely inside or outside with ***T*** or ***F*** and simplify.



The *formulaRestriction* Operation

$$S_{grey} \cap S_{green} \cap -S_{orange}$$



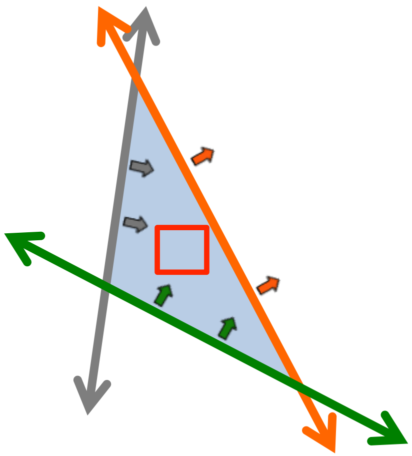
formulaRestriction –

Let *b* be an axis-aligned box,
given a Boolean formula *G* and
the classification for all surfaces of *G* for *b*,
replace all surfaces of *G* in which
b is completely inside or outside with ***T*** or ***F*** and simplify.

The *formulaRestriction* Operation

$$S_{grey} \cap S_{green} \cap -S_{orange}$$

$$\wedge \qquad \wedge$$



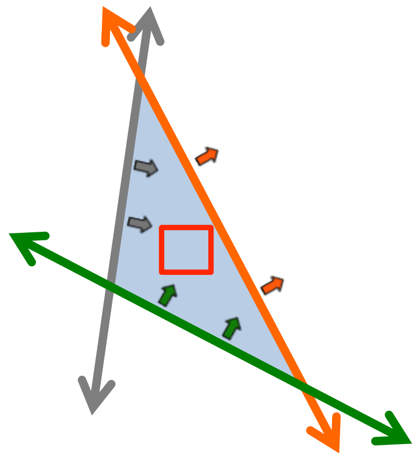
formulaRestriction –

Let *b* be an axis-aligned box,
given a Boolean formula *G* and
the classification for all surfaces of *G* for *b*,
replace all surfaces of *G* in which
b is completely inside or outside with ***T*** or ***F*** and simplify.

The *formulaRestriction* Operation

$$S_{grey} \cap S_{green} \cap -S_{orange}$$

$$T \quad \wedge \qquad\qquad \wedge$$



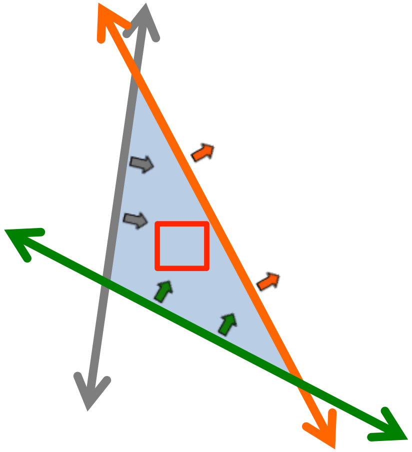
formulaRestriction –

Let *b* be an axis-aligned box,
given a Boolean formula *G* and
the classification for all surfaces of *G* for *b*,
replace all surfaces of *G* in which
b is completely inside or outside with ***T*** or ***F*** and simplify.

The *formulaRestriction* Operation

$$S_{grey} \cap S_{green} \cap -S_{orange}$$

$$T \quad \wedge \quad T \quad \wedge \quad T$$



formulaRestriction –

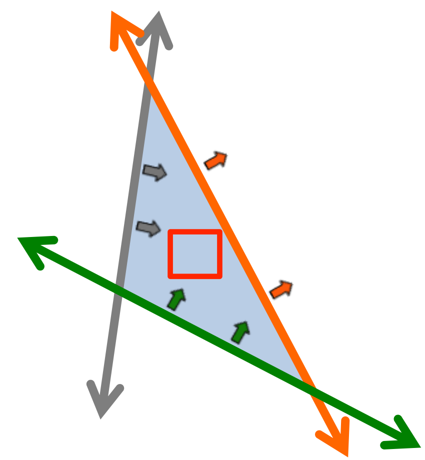
Let *b* be an axis-aligned box,
given a Boolean formula *G* and
the classification for all surfaces of *G* for *b*,
replace all surfaces of *G* in which
b is completely inside or outside with ***T*** or ***F*** and simplify.

The *formulaRestriction* Operation

$$S_{grey} \cap S_{green} \cap -S_{orange}$$

$$T \quad \wedge \quad T \quad \wedge \quad T$$

$$T$$

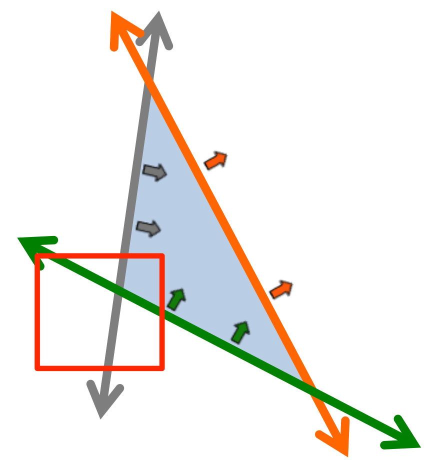


formulaRestriction –

Let *b* be an axis-aligned box,
given a Boolean formula *G* and
the classification for all surfaces of *G* for *b*,
replace all surfaces of *G* in which
b is completely inside or outside with ***T*** or ***F*** and simplify.

The *formulaRestriction* Operation

$$S_{grey} \cap S_{green} \cap -S_{orange}$$



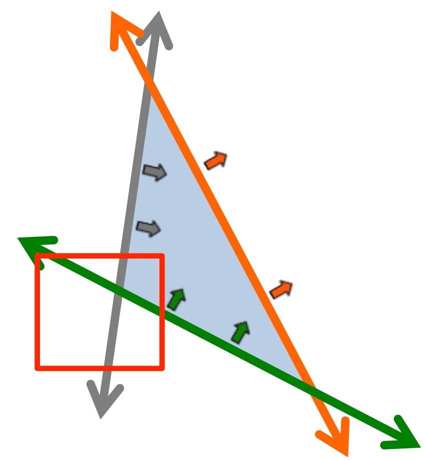
formulaRestriction –

Let *b* be an axis-aligned box,
given a Boolean formula *G* and
the classification for all surfaces of *G* for *b*,
replace all surfaces of *G* in which
b is completely inside or outside with ***T*** or ***F*** and simplify.

The *formulaRestriction* Operation

$$S_{grey} \cap S_{green} \cap -S_{orange}$$

$$S_{grey} \wedge S_{green} \wedge \quad T$$



formulaRestriction –

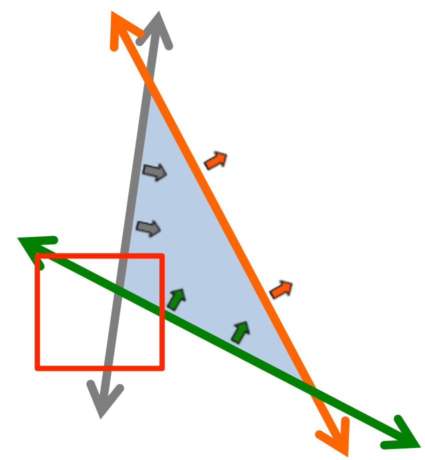
Let *b* be an axis-aligned box,
given a Boolean formula *G* and
the classification for all surfaces of *G* for *b*,
replace all surfaces of *G* in which
b is completely inside or outside with ***T*** or ***F*** and simplify.

The *formulaRestriction* Operation

$$S_{grey} \cap S_{green} \cap -S_{orange}$$

$$S_{grey} \wedge S_{green} \wedge \quad T$$

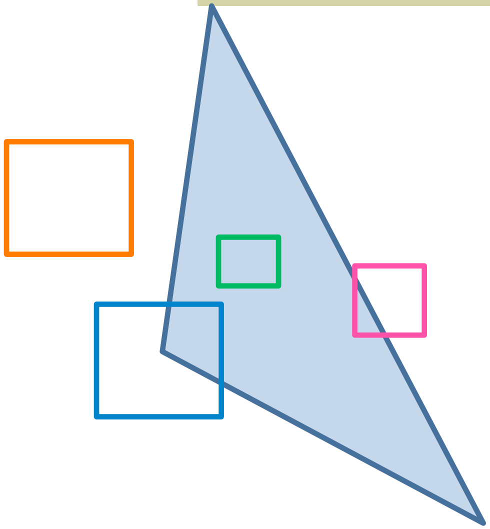
$$S_{grey} \cap S_{green}$$



formulaRestriction –

Let *b* be an axis-aligned box,
given a Boolean formula *G* and
the classification for all surfaces of *G* for *b*,
replace all surfaces of *G* in which
b is completely inside or outside with ***T*** or ***F*** and simplify.

The *classify* Operation



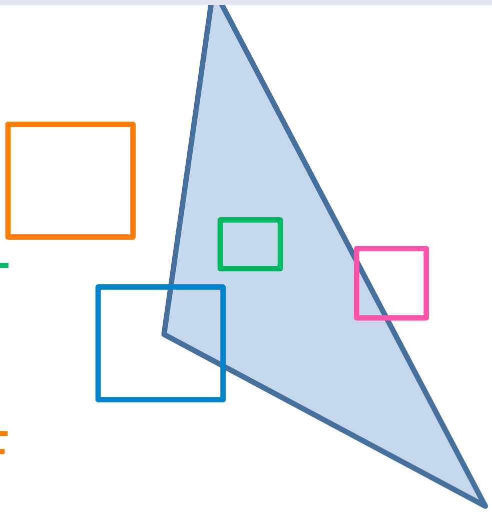
Given comp C and axis-aligned box B, *classify(C,B)*, returns:

- *Inside* $\Rightarrow B \subseteq C$
- *Outside* $\Rightarrow B \cap C = \varnothing$
- *Boundary* $\Rightarrow \exists$ points $\mathrm{p}, \mathrm{q} \in B$ with $\mathrm{p} \in \mathrm{C}$ and $\mathrm{q} \notin \mathrm{C}$
- *Unknown* $\Rightarrow$ could not classify

The *classify* Operation

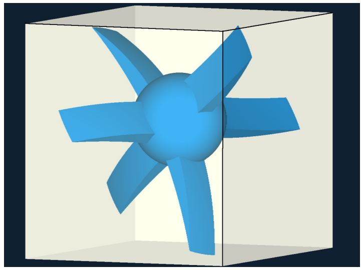
Using *boxLabel* and *formulaRestriction* we implement *classify* as:

Given comp *C* and axis-aligned box *B*, *classify(C,B)*, returns:

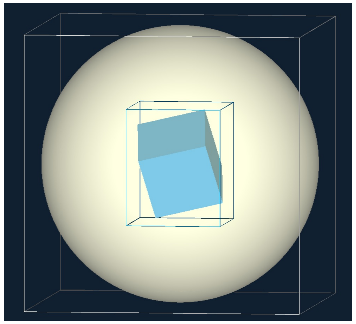


- *Inside* ⇔ Formula resolved to T
- *Outside* ⇔ Formula resolved to F
- *Boundary* ⇔ Formula resolved to 1 surface (or strengthen by cherry picking special cases that are commonly modeled [NMGG13])
- *Unknown* ⇔ could not classify

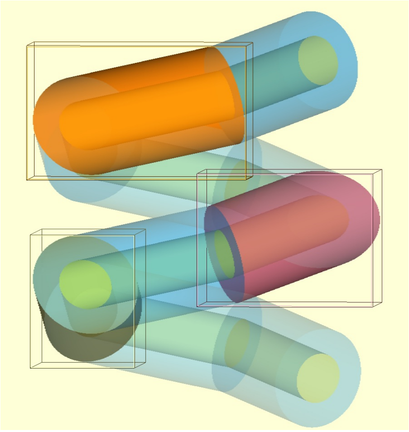
Experiment: Models



SpikeyBall – each spike is formed by the intersection of three planes and two paraboloids. The sharp features cause stochastic and sampling based algorithms produce a box that is too tight.

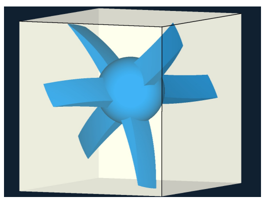


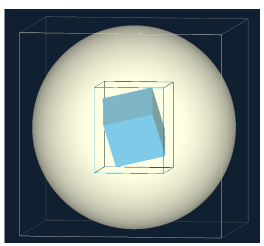
RotatedCube – a rotated cube inside of a sphere. It is easy to verify that the computed box is actually an epsilon box.



HelicalPipe20 – a helical section of piping. A model with multiple levels of hierarchy.

Experiment: 1. Compute AABB



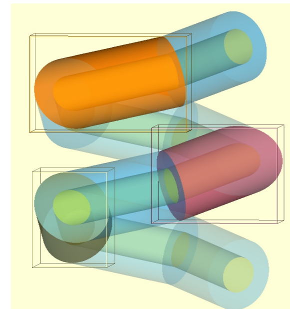


Comp ID			Time (s) for	
			ε = 0.5	ε = 0.05
SpikeyBall				
C0			0.60	1.67
RotatedCube				
C0			0.02	0.10
	C1		<.01	<.01
Total			*0.02*	*0.10*

Initial bounding box:
  Min point: (-1000, -1000, -1000)
  Max point: (1000, 1000, 1000)

Experiment: 1. Compute AABB

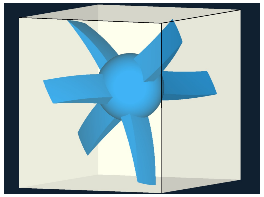
Comp ID			Time (s) for	
			ε = 0.5	ε = 0.05
HelicalPipe20				
C0			0.13	1.62
	C1		0.02	0.25
		C11	0.03	0.19
	C2		0.02	0.39
		C12	0.05	0.36
	C3		0.02	0.63
		C13	0.03	0.22
⋮			⋮	
Total			*0.75*	*8.53*

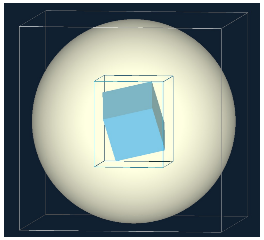


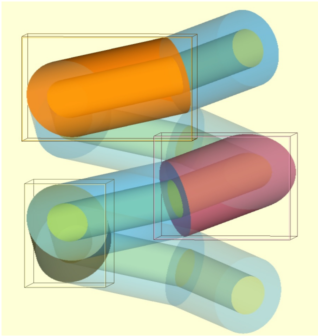
What is time consuming:

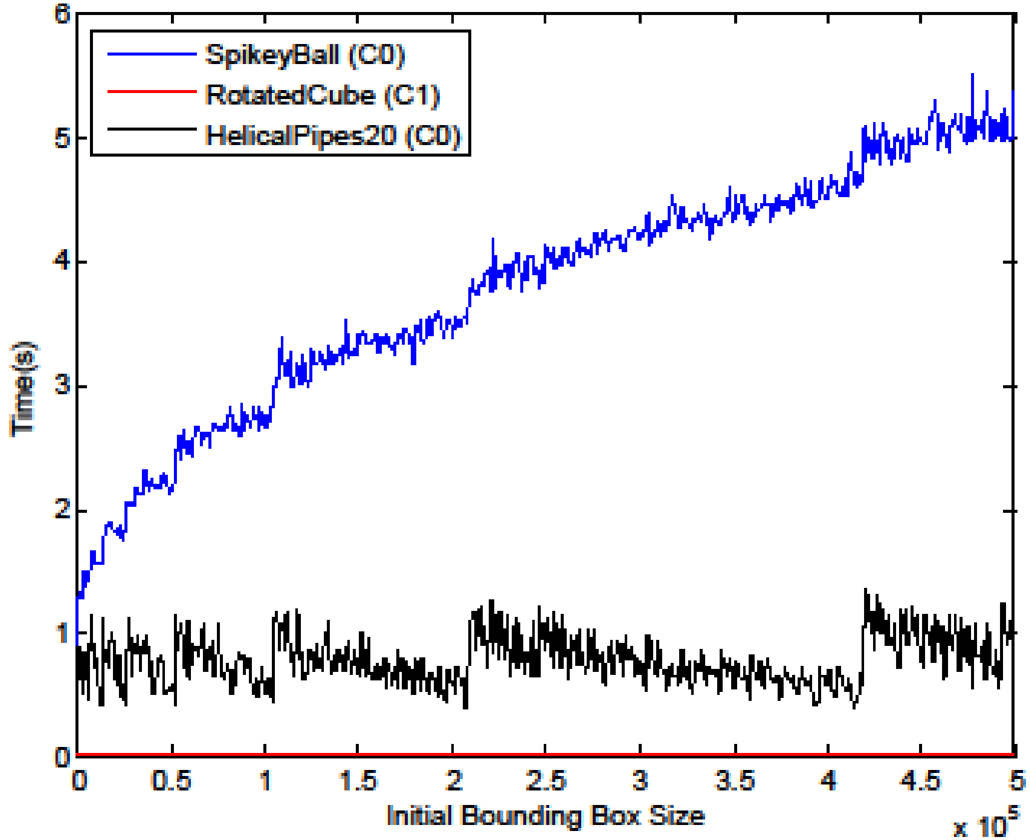
- The size of reducing from the initial bounding box to a tight bounding box.
- Tightening to a smaller ε.

Experiment: 2. Initial AABB



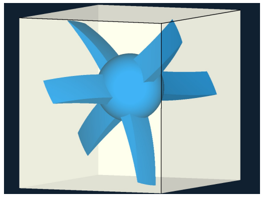


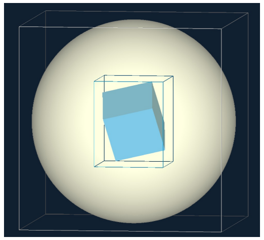


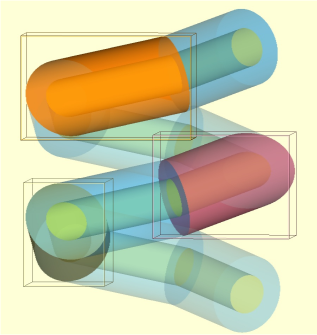


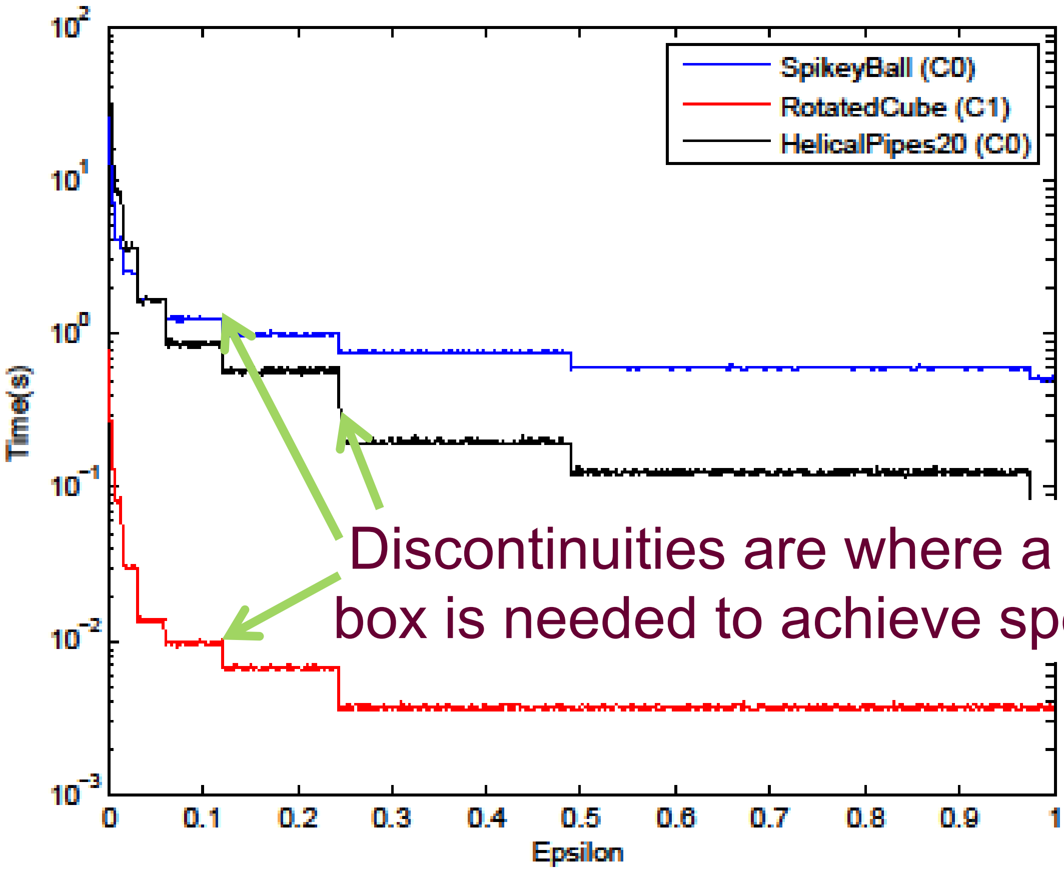
Computing the bounding box is practical
even if we must reduce by 4 orders of magnitude.

Experiment: 3. Tolerance





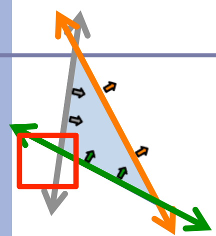




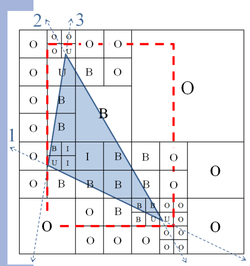
Discontinuities are where a smaller box is needed to achieve specified ε

Empirically, it takes O(1/ε) to compute an ε box.

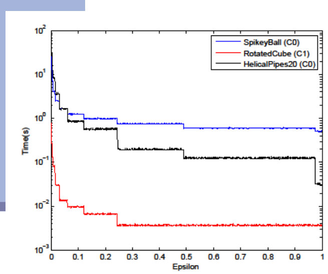
Summing Up Bounding Boxes



Described an operation for testing if an axis-aligned box contains the boundary of a component



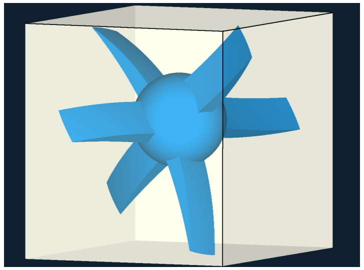
Described a divide-and-conquer framework for computing numerically-optimal bounding boxes



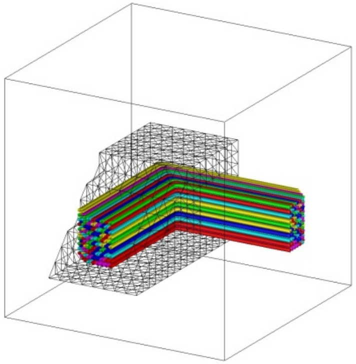
Experiments suggest that algorithm could be routine pre-processing for CSG components.

Extrapolating from experiments,
one million comps on 100 CPUs in about 5.5 min

Outline



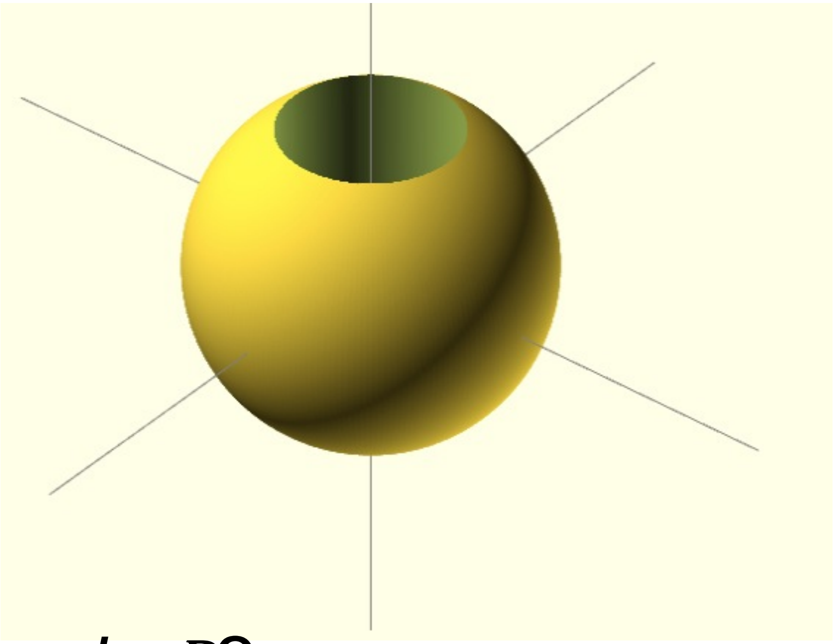
Framework applied to bounding boxes



Framework applied to volumes

Calculus Problem 1

Let D be the region left after drilling a radius r hole though the center of a radius R sphere.

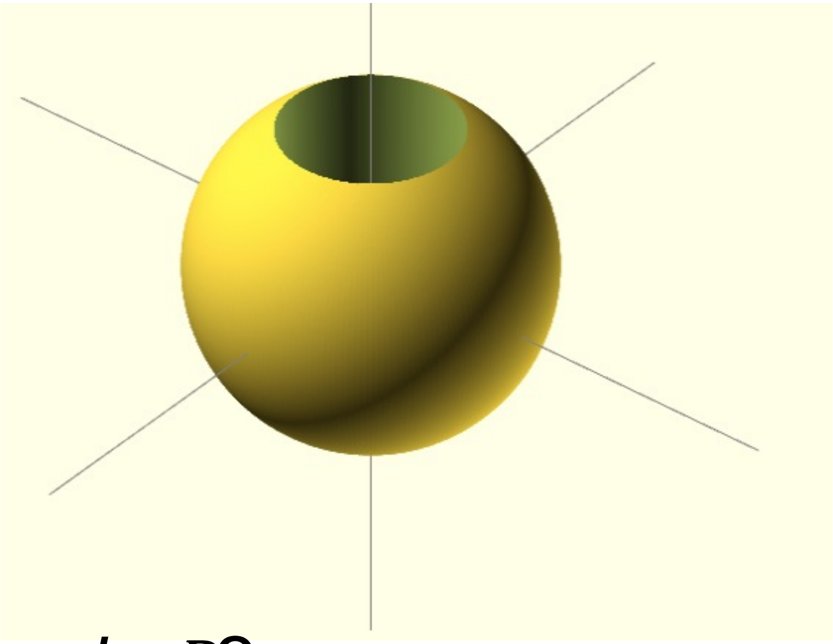


What is the volume of *domain* D?

$$\iiint_D 1\,dV$$

Calculus Problem 1

Let D be the region left after drilling a radius r hole though the center of a radius R sphere.



What is the volume of *domain* D?

$$\iiint_D 1\,dV = \frac{4}{3}\left(R^2 - r^2\right)^{\frac{3}{2}}$$

What the Computer Sees

Let *D* be the intersection of 10 quadratics:

$0 > 0.74742x^2 + 0.93022y^2 + 0.32256z^2 + 0.26590xy +-0.82750xz + 0.43517yz + 2.47974x +2$

$0 > 0.00487x^2 + 0.00638y^2 + 0.00212z^2 + 0.00181xy +-0.00537xz + 0.00299yz + 0.51989x +-$

$0 <-0.00469x^2 + 0.00617y^2 +-0.00134z^2 + 0.00116xy + 0.00609xz + 0.00326yz + 0.52845x +-$

$0 > 0.00180x^2 + 0.00647y^2 + 0.00497z^2 +-0.00039xy + 0.00597xz + 0.00003yz + 0.59729x +-$

$0 > 0.00173x^2 + 0.00681y^2 + 0.00479z^2 +-0.00022xy + 0.00574xz + 0.00034yz +-0.76442x +$

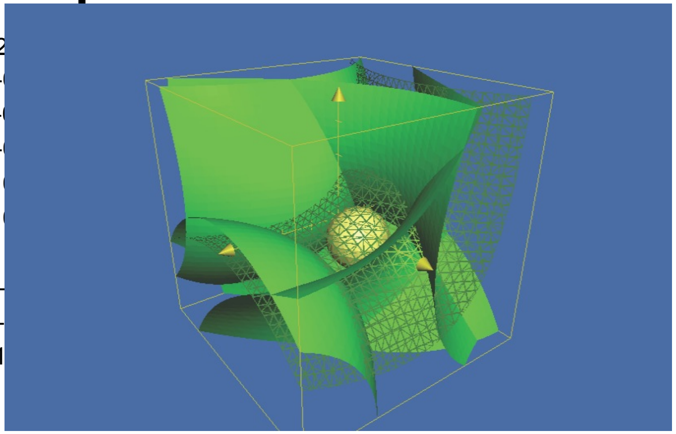
$0 > 0.00180x^2 + 0.00657y^2 + 0.00498z^2 +-0.00037xy + 0.00599xz + 0.00008yz +-0.76185x +$

$0 <-0.00156x^2 + 0.00591y^2 +-0.00403z^2 + 0.00324xy +-0.00503xz + 0.00601yz +-0.90629x +$

$0 > 0.00643x^2 + 0.00046y^2 + 0.00614z^2 +-0.00143xy +-0.00036xz +-0.00301yz +-0.04751x +-$

$0 > 0.00323x^2 +-0.00046y^2 +-0.00276z^2 + 0.00209xy +-0.01145xz + 0.00273yz +-0.19156x +-$

$0 < 0.50007x^2 + 0.50004y^2 + 0.50003z^2 + 0.00009xy + 0.00002xz + 0.00004yz + 6.69291x +1$

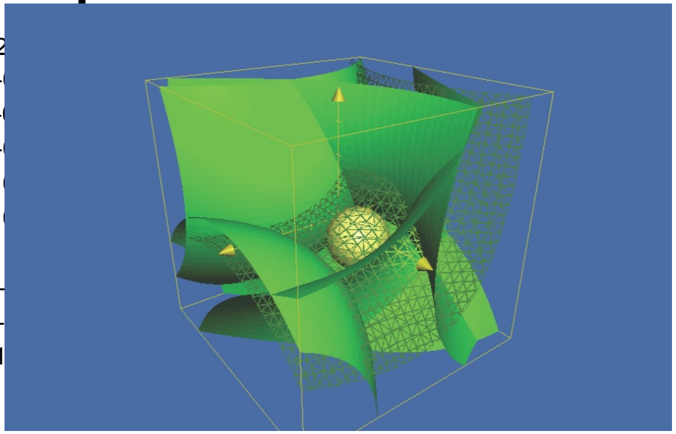


Even with a picture,
finding the limits of the integral is challenging

What the Computer Sees

Let *D* be the intersection of 10 quadratics:

0 > 0.74742x^2 + 0.93022y^2 + 0.32256z^2 + 0.26590xy +-0.82750xz + 0.43517yz + 2.47974x +2
0 > 0.00487x^2 + 0.00638y^2 + 0.00212z^2 + 0.00181xy +-0.00537xz + 0.00299yz + 0.51989x +-
0 <-0.00469x^2 + 0.00617y^2 +-0.00134z^2 + 0.00116xy + 0.00609xz + 0.00326yz + 0.52845x +-
0 > 0.00180x^2 + 0.00647y^2 + 0.00497z^2 +-0.00039xy + 0.00597xz + 0.00003yz + 0.59729x +-
0 > 0.00173x^2 + 0.00681y^2 + 0.00479z^2 +-0.00022xy + 0.00574xz + 0.00034yz +-0.76442x +
0 > 0.00180x^2 + 0.00657y^2 + 0.00498z^2 +-0.00037xy + 0.00599xz + 0.00008yz +-0.76185x +
0 <-0.00156x^2 + 0.00591y^2 +-0.00403z^2 + 0.00324xy +-0.00503xz + 0.00601yz +-0.90629x +
0 > 0.00643x^2 + 0.00046y^2 + 0.00614z^2 +-0.00143xy +-0.00036xz +-0.00301yz +-0.04751x +-
0 > 0.00323x^2 +-0.00046y^2 +-0.00276z^2 + 0.00209xy +-0.01145xz + 0.00273yz +-0.19156x +-
0 < 0.50007x^2 + 0.50004y^2 + 0.50003z^2 + 0.00009xy + 0.00002xz + 0.00004yz + 6.69291x +1



Even with a picture,
finding the limits of the integral is challenging

The difficulty is finding the domain

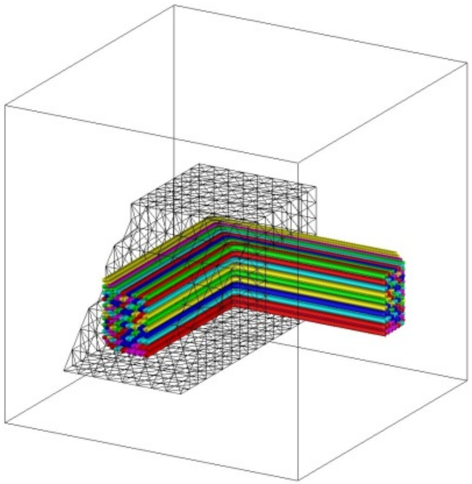
Overview

The framework uses analytic, stochastic and numerical integration, as appropriate.

Basic steps:

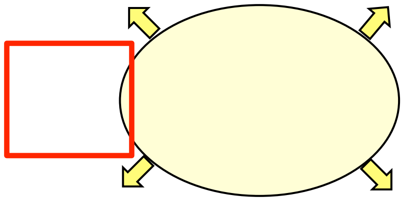
- Subdivide the model into boxes
- Identify boxes that are "easy" to integrate
  - difficult boxes are further subdivided
- Apply "best" integrator for each box.

Model Name	Alg	Time (sec)
cPiped100 tol: ±1.1e-04	**Old** **New**	**790.28** **1.41**



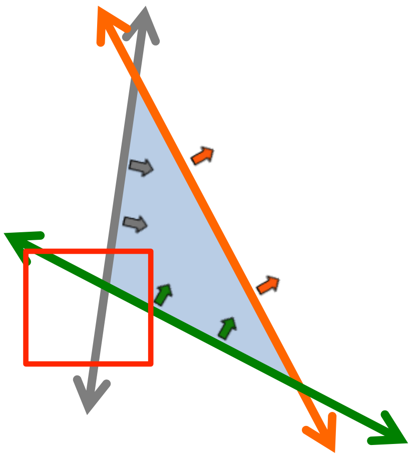
Recall: Ops for Bounding Box

Let *b* be an axis aligned box:



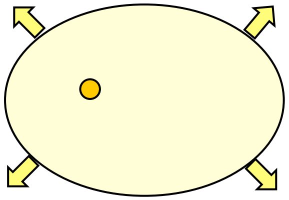
- *boxLabel* – given a surface *S,* return if the points of *b* are inside, outside, or both with respect to *S*.

- *formulaRestriction* – given a Boolean formula *G* and the classification for all surfaces of *G* for *b*, replace all surfaces of *G* in which *b* is completely inside or outside with ***T*** or ***F*** and simplify.

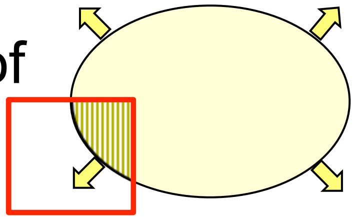


Operations on Primitives

- Point inside – return if query point is inside *S*



- *Integrator* – return the intersection volume of the interior of *S* with an axis-aligned box.

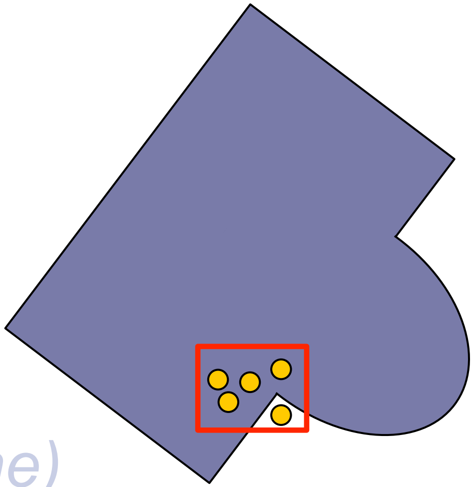


Surface-in-Box Integrators

Given a component *C*, axis-aligned box *b*,
a target error ε, and confidence δ,
an *integrator* either
computes volumes of *C* and *b's* intersection
or flags as "needs subdivision."

Basic integrators:

- *Monte Carlo Integrator (MC)*
- *Box Integrator (Box)*

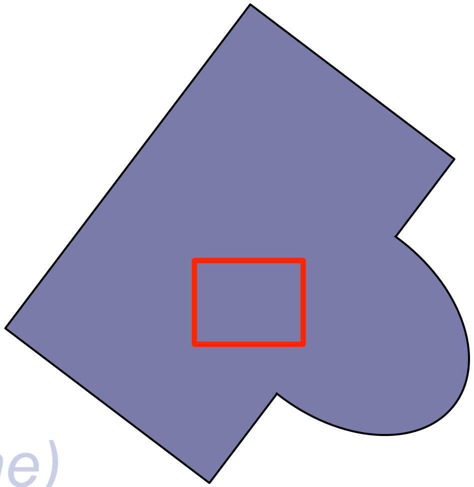


Advanced integrators:

- *Pair of Planes Integrator (2Plane)*
- *Bundle of Cylinders Integrator (BunCyl)*

Surface-in-Box Integrators

Given a component *C*, axis-aligned box *b*,
a target error *ε*, and confidence *δ*,
an *integrator* either
computes volumes of *C* and *b's* intersection
or flags as "needs subdivision."



Basic integrators:

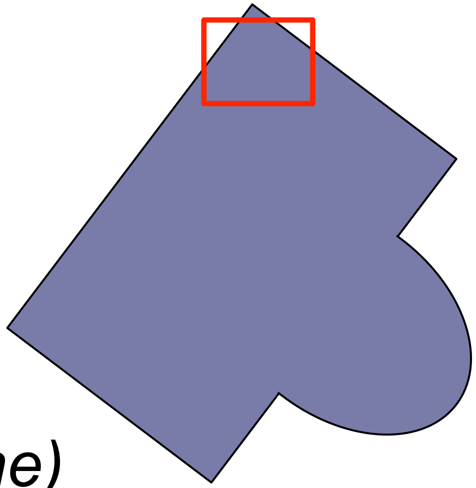
- *Monte Carlo Integrator (MC)*
- *Box Integrator (Box)*

Advanced integrators:

- *Pair of Planes Integrator (2Plane)*
- *Bundle of Cylinders Integrator (BunCyl)*

Surface-in-Box Integrators

Given a component *C*, axis-aligned box *b*,
a target error ε, and confidence δ,
an *integrator* either
computes volumes of *C* and *b's* intersection
or flags as “needs subdivision.”



Basic integrators:

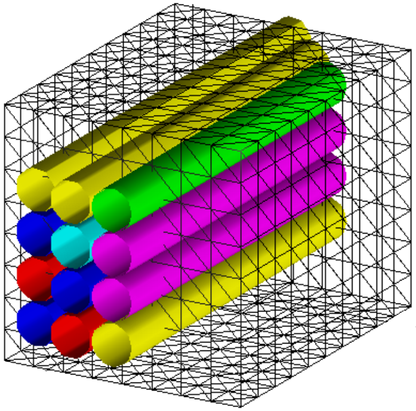
- *Monte Carlo Integrator (MC)*
- *Box Integrator (Box)*

Advanced integrators:

- *Pair of Planes Integrator (2Plane)*
- *Bundle of Cylinders Integrator (BunCyl)*

Surface-in-Box Integrators

Given a component *C*, axis-aligned box *b*,
a target error *ε*, and confidence *δ*,
an *integrator* either
computes volumes of *C* and *b's* intersection
or flags as "needs subdivision."



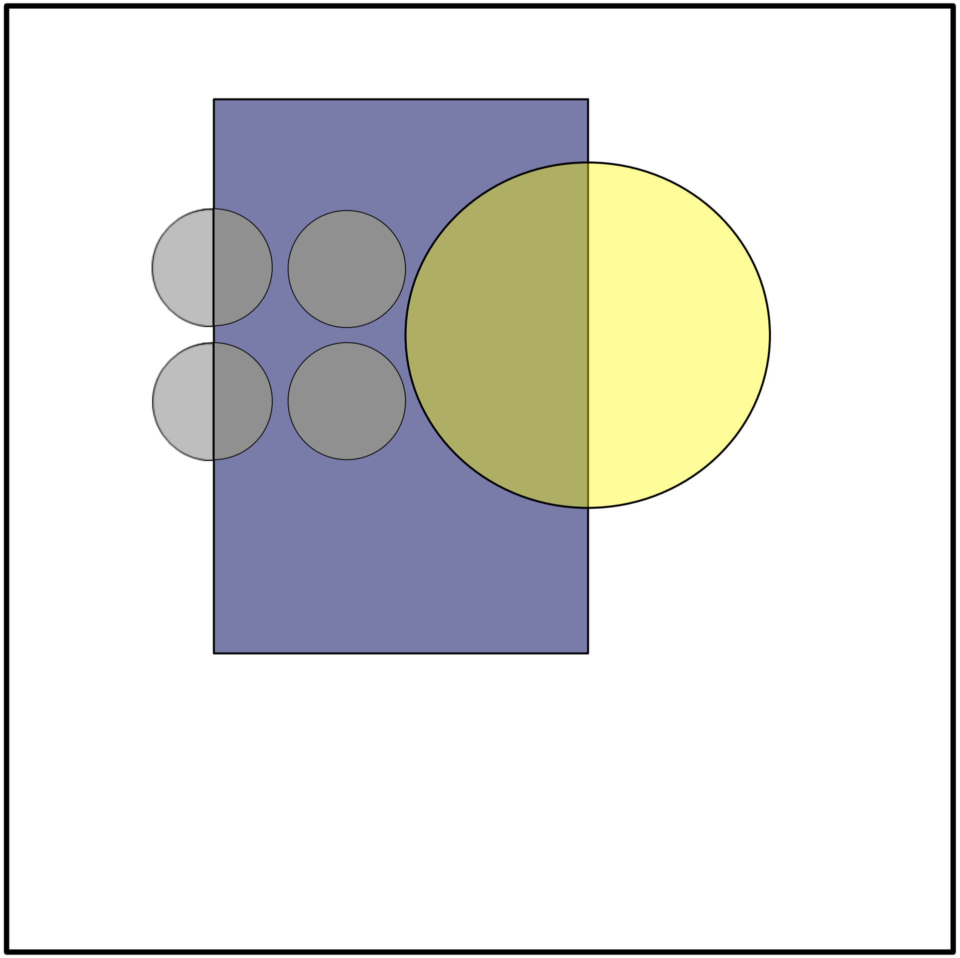
Basic integrators:

- *Monte Carlo Integrator (MC)*
- *Box Integrator (Box)*

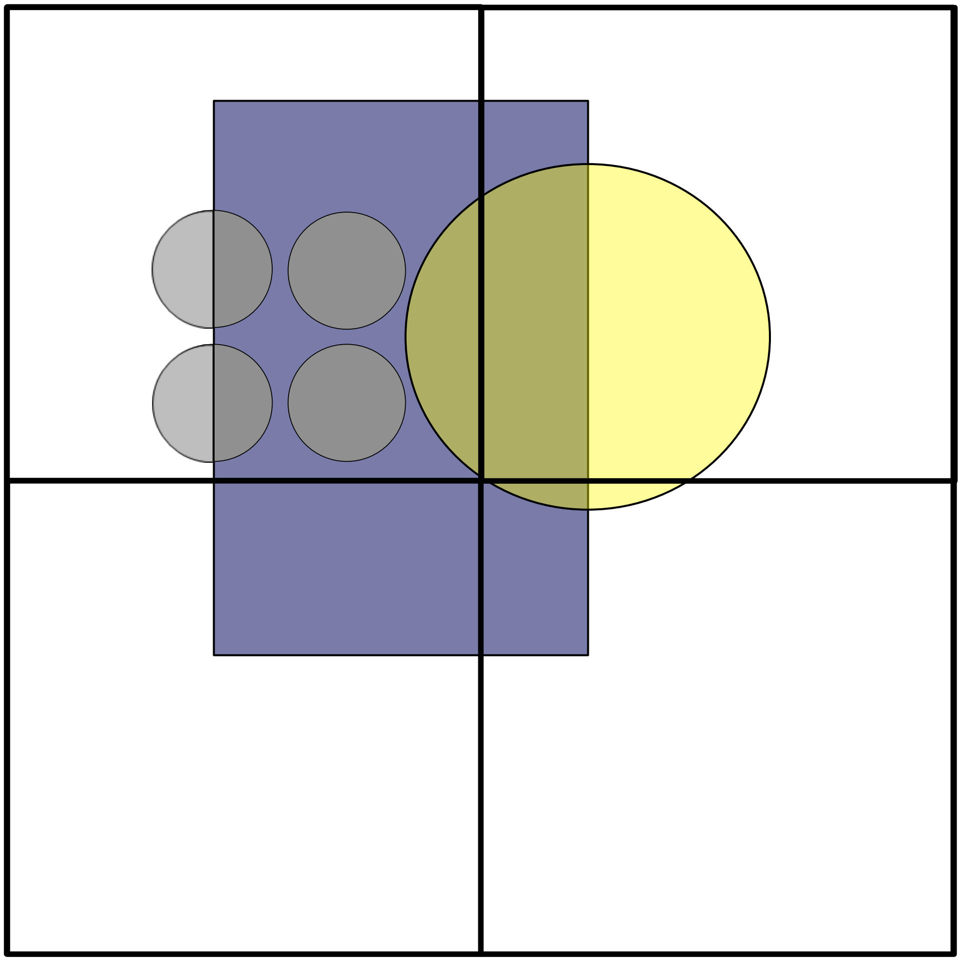
Advanced integrators:

- *Pair of Planes Integrator (2Plane)*
- *Bundle of Cylinders Integrator (BunCyl)*

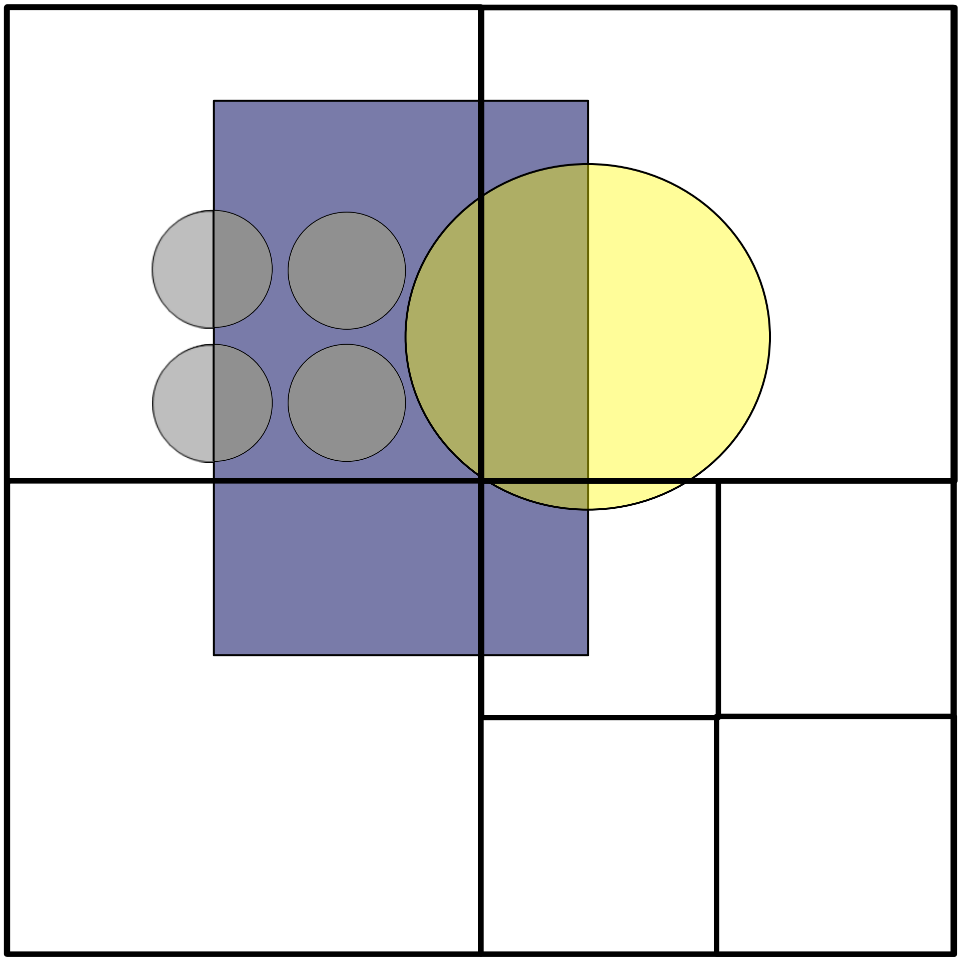
Algorithm Animation



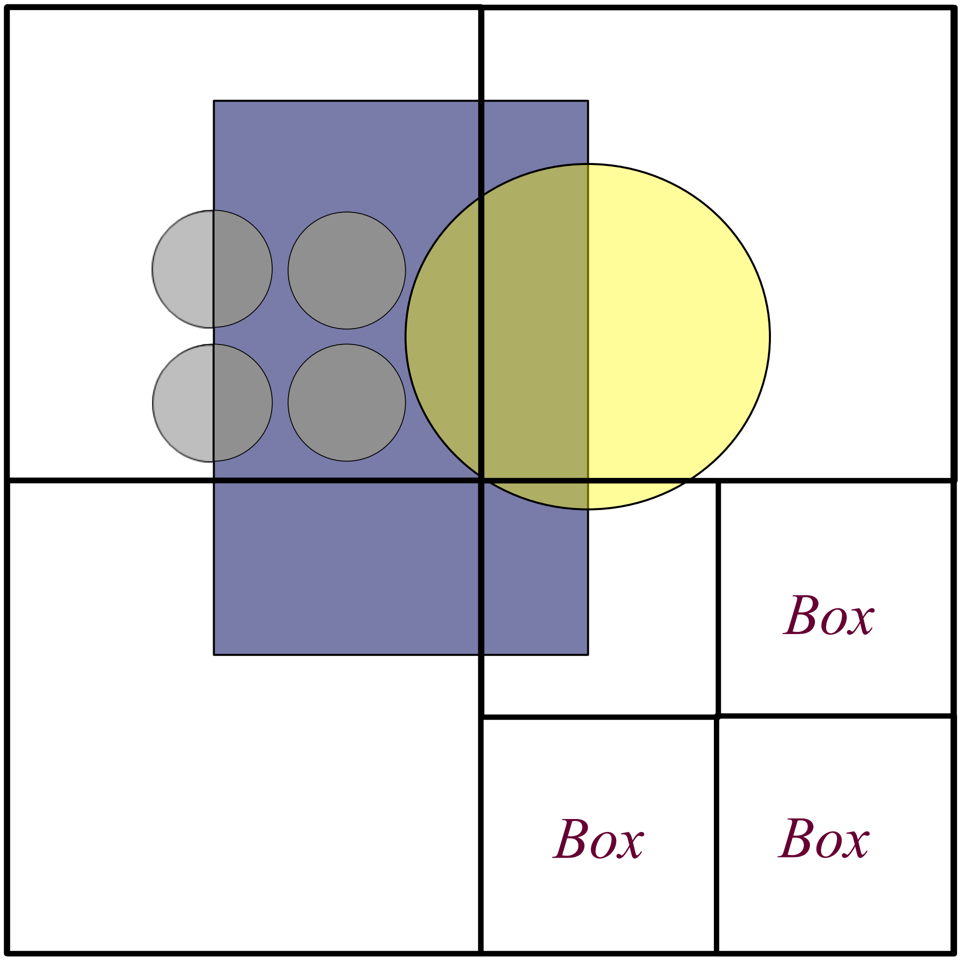
Algorithm Animation



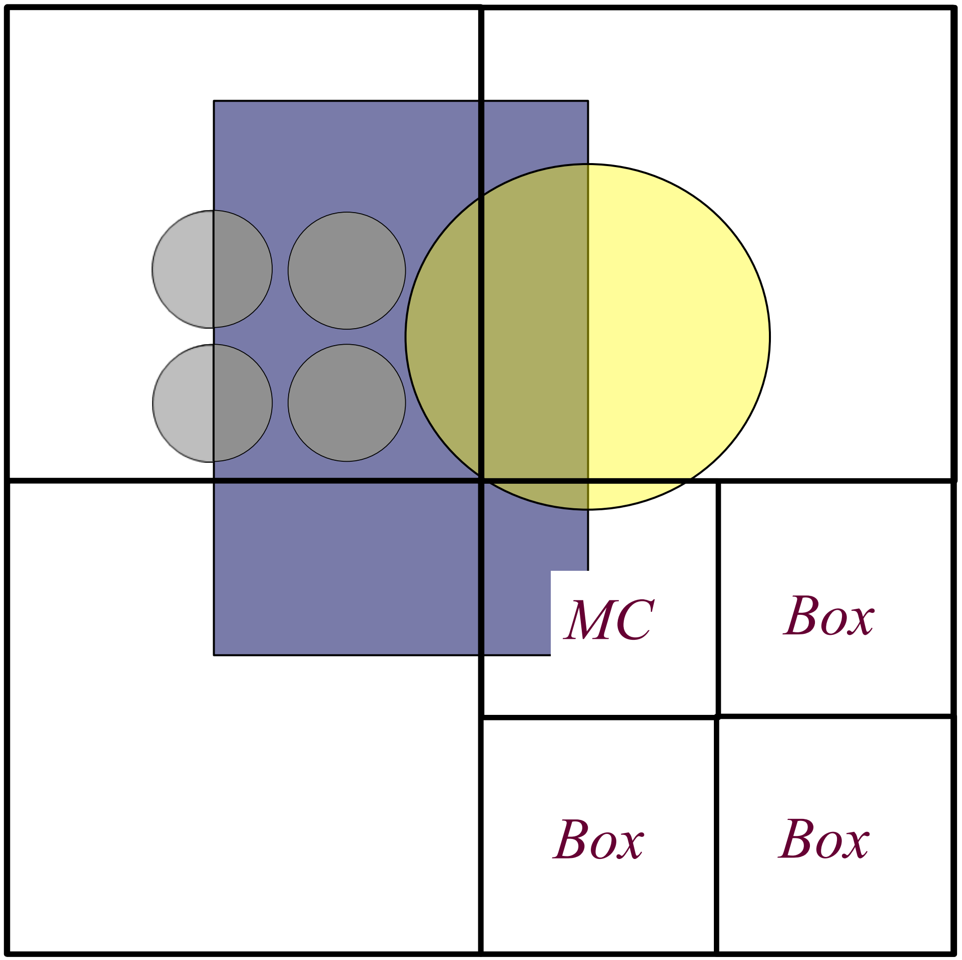
Algorithm Animation



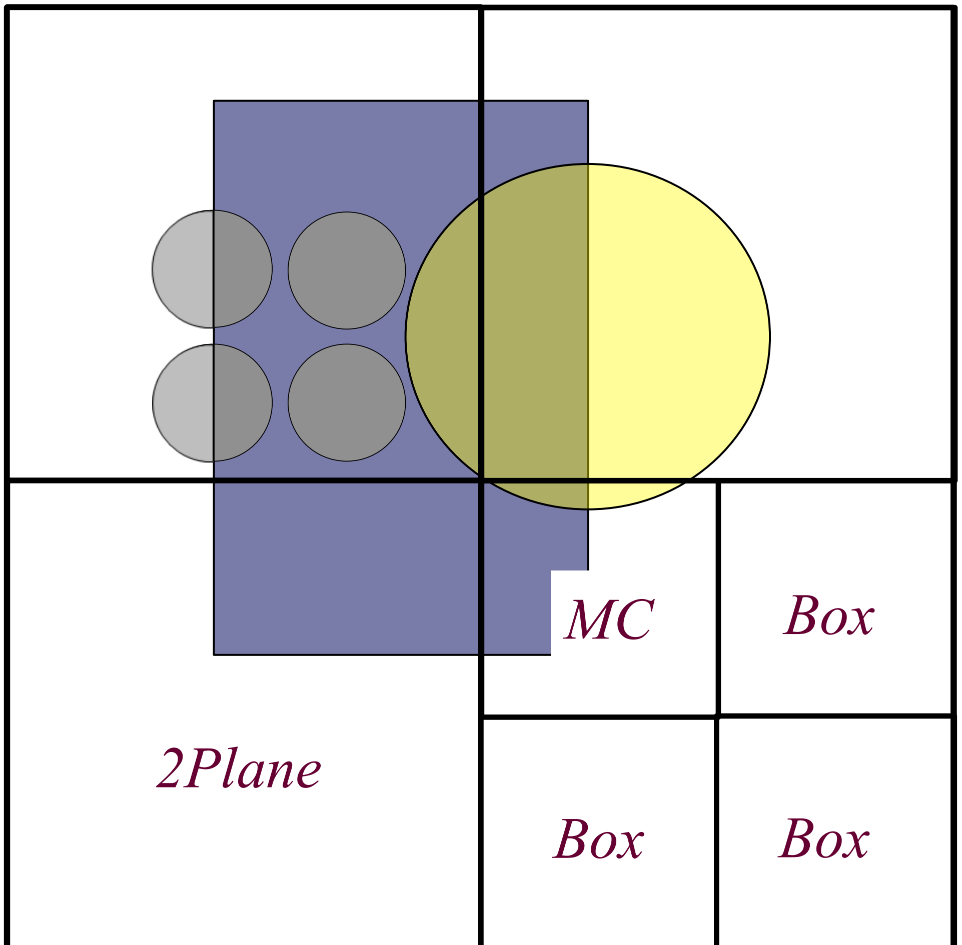
Algorithm Animation



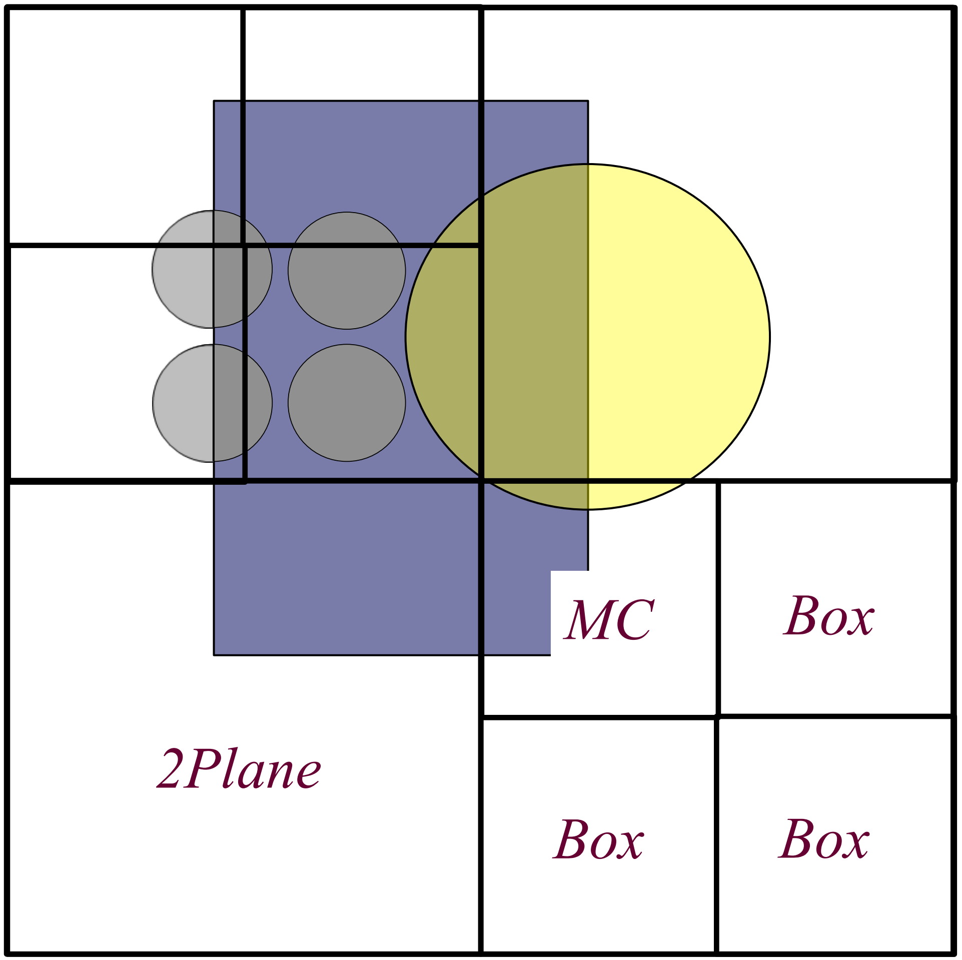
Algorithm Animation



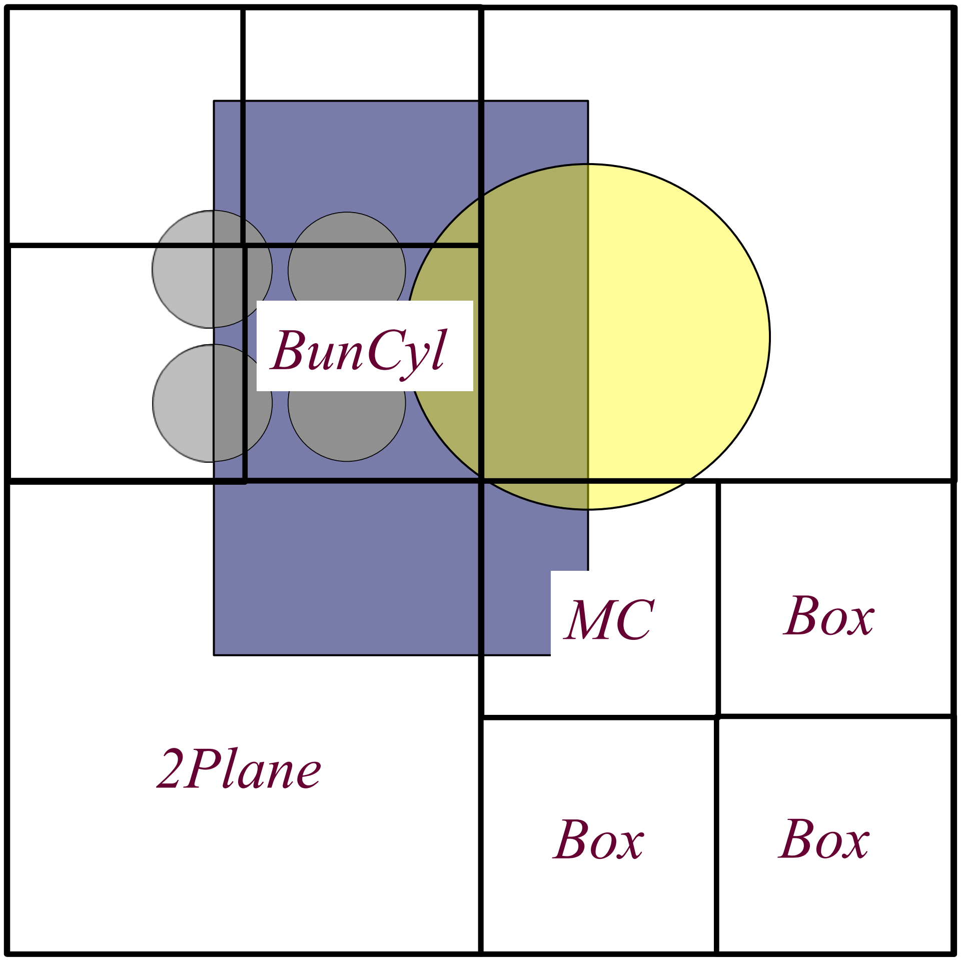
Algorithm Animation



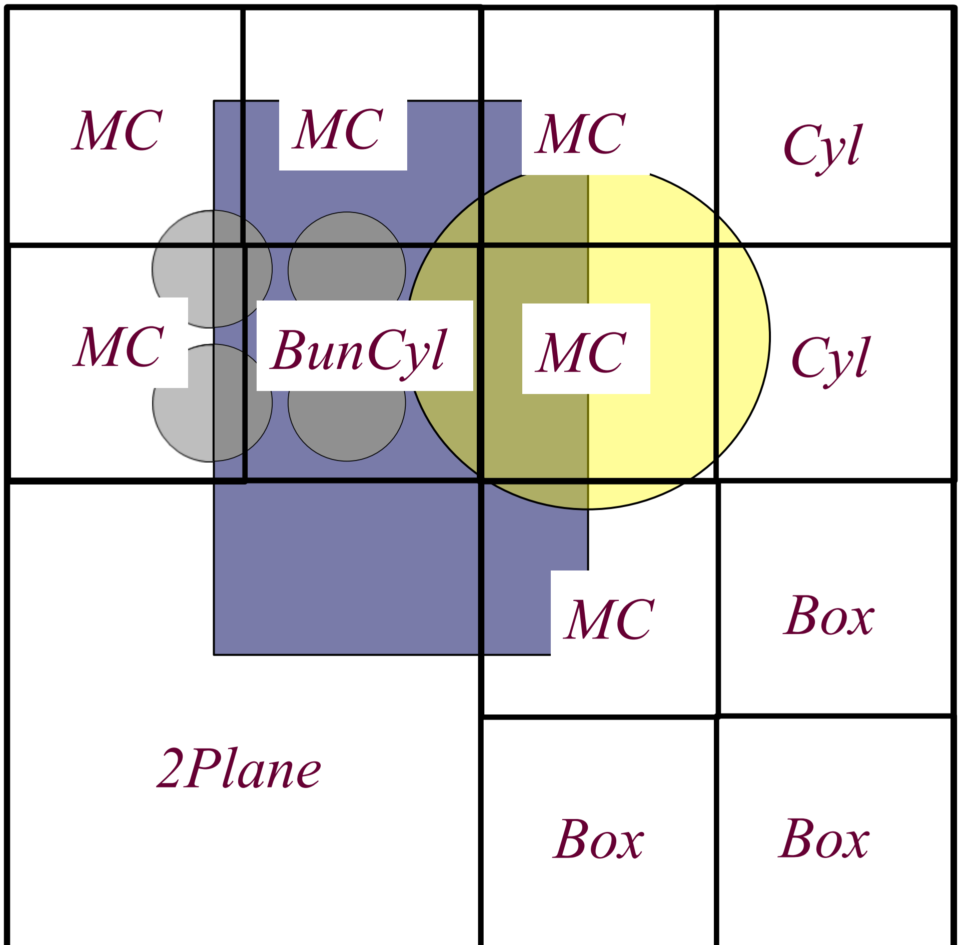
Algorithm Animation



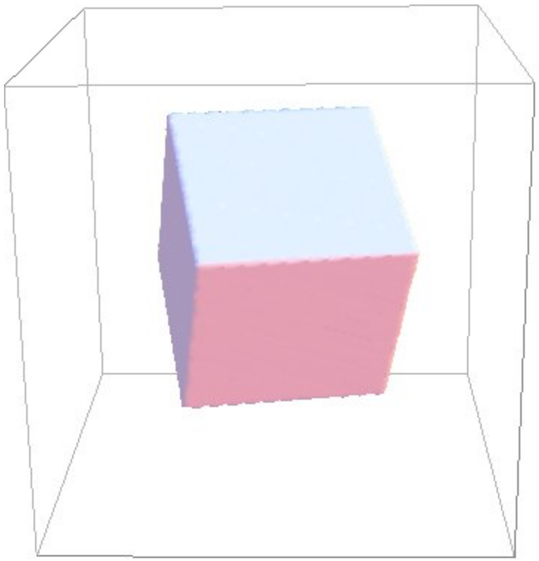
Algorithm Animation



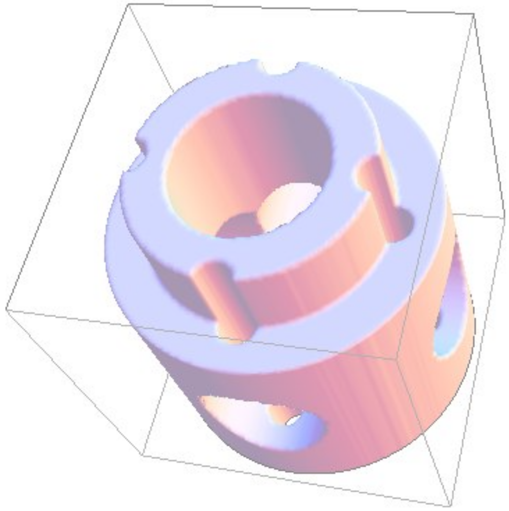
Algorithm Animation



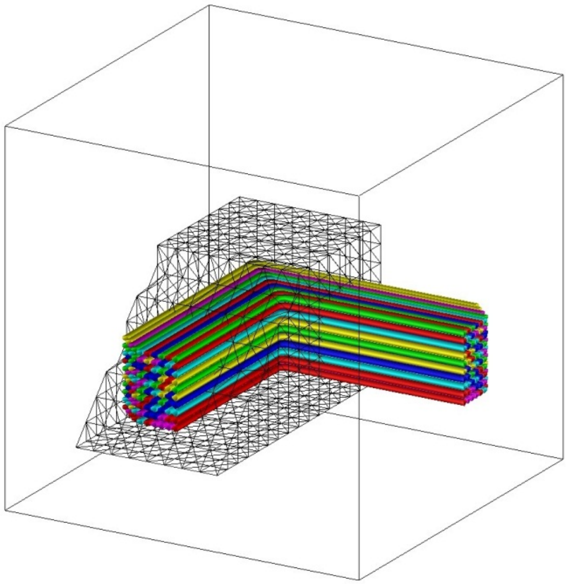
Experiment: Models



Cube



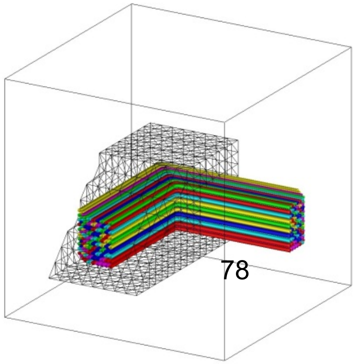
DrillCyl



cPiped12, cPiped100, and cPiped10000

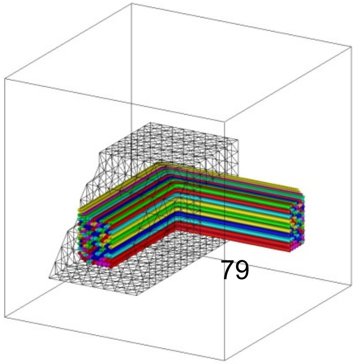
Experiment: Accuracy and Time

Model Name	Alg	Total Volume	Time (sec)
cPiped100 tol: ±1.1e-04	Analytic	**0.0731**920	-



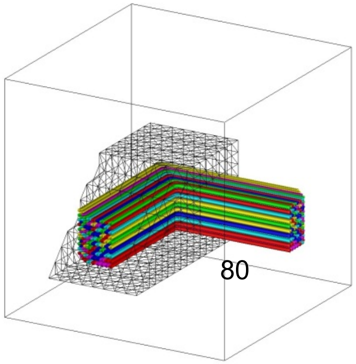
Experiment: Accuracy and Time

Model Name	Alg	Total Volume	Time (sec)
cPiped100 tol: ±1.1e-04	Analytic Monte Carlo (MC)	**0.0731**920 **0.0731**951	-



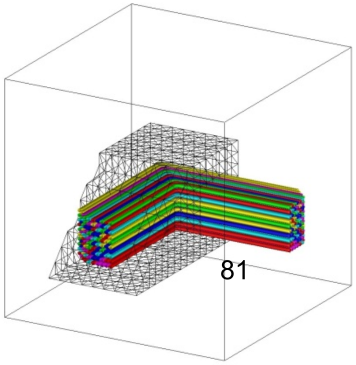
Experiment: Accuracy and Time

Model Name	Alg	Total Volume	Time (sec)
cPiped100 tol: ±1.1e-04	Analytic Monte Carlo (MC) +Subdivision & Box (Sdiv&Box)	**0.0731**920 **0.0731**951 **0.073192**1	-



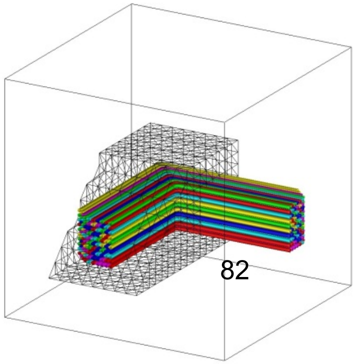
Experiment: Accuracy and Time

Model Name	Alg	Total Volume	Time (sec)
cPiped100 tol: ±1.1e-04	Analytic Monte Carlo (MC) +Subdivision & Box (Sdiv&Box) +Pair of Planes (2 Plane) +Bundle of Cylinders (BunCyl)	**0.0731**920 **0.0731**951 **0.073192**1 **0.073191**9 **0.073191**9	-



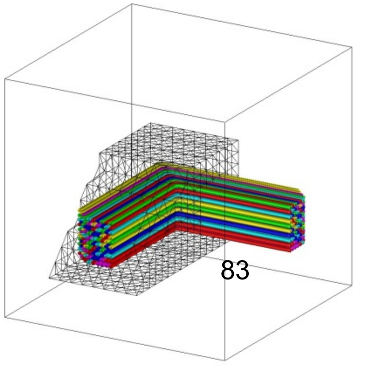
Experiment: Accuracy and Time

Model Name	Alg	Total Volume	Time (sec)
cPiped100 tol: ±1.1e-04	Analytic MC +Sdiv&Box +2Plane +BunCyl	0.0731920 0.0731951 0.0731921 0.0731919 0.0731919	-



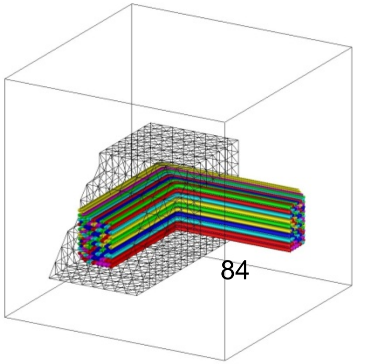
Experiment: Accuracy and Time

Model Name	Alg	Total Volume	Time (sec)
cPiped100 tol: ±1.1e-04	Analytic	0.0731920	-
	MC	0.0731951	790.28
	+Sdiv&Box	0.0731921	
	+2Plane	0.0731919	
	+BunCyl	0.0731919	



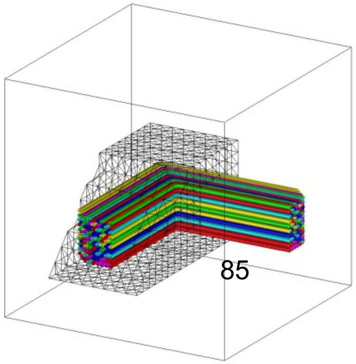
Experiment: Accuracy and Time

Model Name	Alg	Total Volume	Time (sec)
cPiped100	Analytic	0.0731920	-
tol: ±1.1e-04	MC	0.0731951	790.28
	+Sdiv&Box	0.0731921	63.96
	+2Plane	0.0731919	51.32
	+BunCyl	0.0731919	1.41



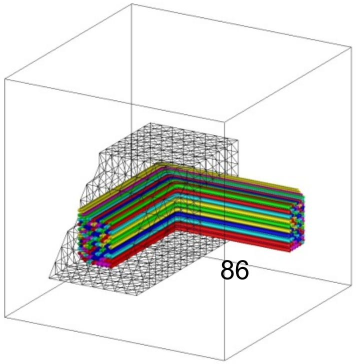
Experiment: Accuracy and Time

Model Name	Alg	Total Volume	Time (sec)
cPiped100 tol: ±1.1e-04	Analytic	0.0731920	-
	MC	**0.0731**951	**790.28**
	+Sdiv&Box	0.0731921	63.96
	+2Plane	0.0731919	51.32
	+BunCyl	**0.073191**9	**1.41**



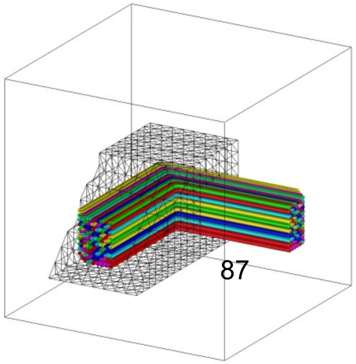
Experiment: # Boxes Impact Time

Model Name	Alg	Total Boxes	Integrators (% of total boxes)				Time (sec)
			MC	Box	2Plane	BunCyl	
cPiped100 tol: ±1.1e-04 vol: 0.0731920	MC	1	100.0	-	-	-	790.28



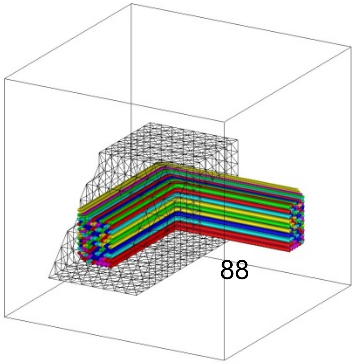
Experiment: # Boxes Impact Time

Model Name	Alg	Total Boxes	Integrators (% of total boxes) MC	Box	2Plane	BunCyl	Time (sec)
cPiped100 tol: ±1.1e-04 vol: 0.0731920	MC	1	100.0	-	-	-	790.28
	+Sdiv&Box	62,392,744	45.2	54.8	-	-	63.96



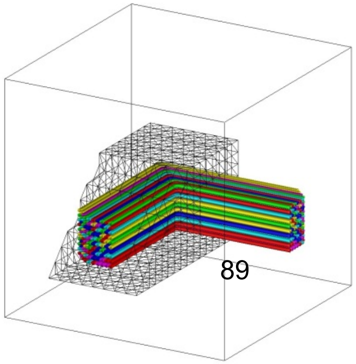
Experiment: # Boxes Impact Time

Model Name	Alg	Total Boxes	Integrators (% of total boxes) MC	Box	2Plane	BunCyl	Time (sec)
cPiped100	MC	1	100.0	-	-	-	790.28
tol: ±1.1e-04	+Sdiv&Box	62,392,744	45.2	54.8	-	-	63.96
vol: 0.0731920	+2 Plane	48,958,575	45.6	54.2	<0.1	-	51.32
	+BunCyl	482,756	16.8	49.6	11.8	3.3	1.41



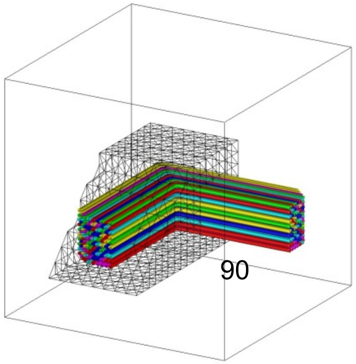
Experiment: # Boxes Impact Time

Model Name	Alg	Total Boxes	Integrators (% of total boxes) MC	Box	2Plane	BunCyl	Time (sec)
cPiped100	MC	1	100.0	-	-	-	790.28
tol: ±1.1e-04	**+Sdiv&Box**	**62,392,744**	45.2	54.8	-	-	**63.96**
vol: 0.0731920	+2 Plane	48,958,575	45.6	54.2	<0.1	-	51.32
	+BunCyl	**482,756**	16.8	49.6	11.8	3.3	**1.41**



Experiment: Integrators Impact Time

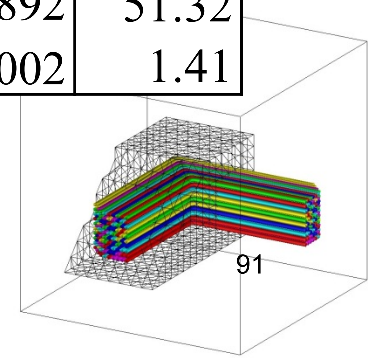
Model Name	Alg	Total Boxes	Integrators (% of total boxes) MC	Box	2Plane	BunCyl	Time (sec)
cPiped100	MC	1	100.0	-	-	-	790.28
tol: ±1.1e-04	+Sdiv&Box	62,392,744	45.2	54.8	-	-	63.96
vol: 0.0731920	+2 Plane	48,958,575	45.6	54.2	<0.1	-	51.32
	+BunCyl	482,756	16.8	49.6	11.8	3.3	1.41



Experiment: Integrators Impact Time

Model Name	Alg	Total Boxes	Integrators (% of total boxes)				Time (sec)
			MC	Box	2Plane	BunCyl	
cPiped100	MC	1	100.0	-	-	-	790.28
tol: ±1.1e-04	+Sdiv&Box	62,392,744	45.2	54.8	-	-	63.96
vol: 0.0731920	+2 Plane	48,958,575	45.6	54.2	<0.1	-	51.32
	+BunCyl	482,756	16.8	49.6	11.8	3.3	1.41

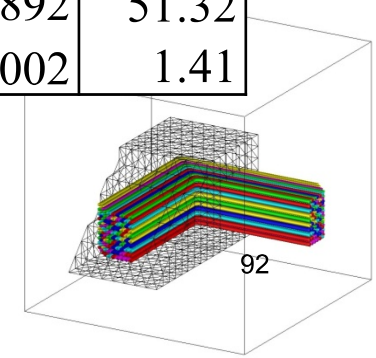
Model Name	Alg	Integrators (% of total vol)				Total Samples	Time (sec)
		MC	Box	2Plane	BunCyl		
cPiped100	MC	100.0	-	-	-	1,410,065,909	790.28
tol: ±1.1e-04	+Sdiv&Box	0.3	99.7	-	-	56,352,288	63.96
vol: 0.0731920	+2 Plane	0.3	75.3	24.4	-	44,694,892	51.32
	+BunCyl	<0.1	70.5	24.3	5.0	162,002	1.41



Experiment: Integrators Impact Time

Model Name	Alg	Total Boxes	Integrators (% of total boxes) MC	Box	2Plane	BunCyl	Time (sec)
cPiped100	MC	1	100.0	-	-	-	790.28
tol: ±1.1e-04	+Sdiv&Box	62,392,744	45.2	54.8	-	-	63.96
vol: 0.0731920	**+2 Plane**	48,958,575	45.6	54.2	**<0.1**	-	51.32
	+BunCyl	482,756	16.8	49.6	11.8	3.3	1.41

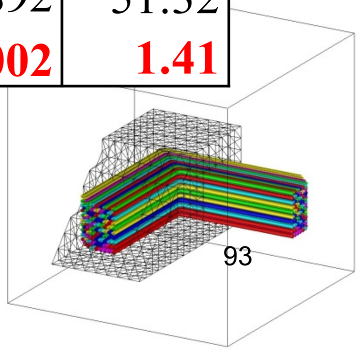
Model Name	Alg	Integrators (% of total vol) MC	Box	2Plane	BunCyl	Total Samples	Time (sec)
cPiped100	MC	100.0	-	-	-	1,410,065,909	790.28
tol: ±1.1e-04	+Sdiv&Box	0.3	99.7	-	-	56,352,288	63.96
vol: 0.0731920	**+2 Plane**	0.3	75.3	**24.4**	-	44,694,892	51.32
	+BunCyl	<0.1	70.5	24.3	5.0	162,002	1.41



Experiment: Integrators Impact Time

Model Name	Alg	Total Boxes	Integrators (% of total boxes) MC	Box	2Plane	BunCyl	Time (sec)
cPiped100	MC	1	100.0	-	-	-	790.28
tol: ±1.1e-04	+Sdiv&Box	62,392,744	45.2	54.8	-	-	63.96
vol: 0.0731920	+2 Plane	48,958,575	45.6	54.2	<0.1	-	51.32
	+BunCyl	482,756	16.8	49.6	11.8	3.3	1.41

Model Name	Alg	Integrators (% of total vol) MC	Box	2Plane	BunCyl	Total Samples	Time (sec)
cPiped100	**MC**	100.0	-	-	-	**1,410,065,909**	**790.28**
tol: ±1.1e-04	+Sdiv&Box	0.3	99.7	-	-	56,352,288	63.96
vol: 0.0731920	+2 Plane	0.3	75.3	24.4	-	44,694,892	51.32
	+BunCyl	<0.1	70.5	24.3	5.0	**162,002**	**1.41**



Experiment: Larger Model

Model Name	Alg	Integrators (% of total vol) MC	Box	2Plane	BunCyl	Total Samples	Time (sec)
cPiped10000	**MC**	-	-	-	-	-	**>12h***
tol: ±1.1e-04	+Sdiv&Box	1.6	98.4	-	-	279,088,846	358.09
vol: 0.0767715	+2 Plane	1.6	74.0	24.4	-	267,848,220	348.25
	+BunCyl	<0.1	70.5	24.3	5.1	931,534	**9.43**

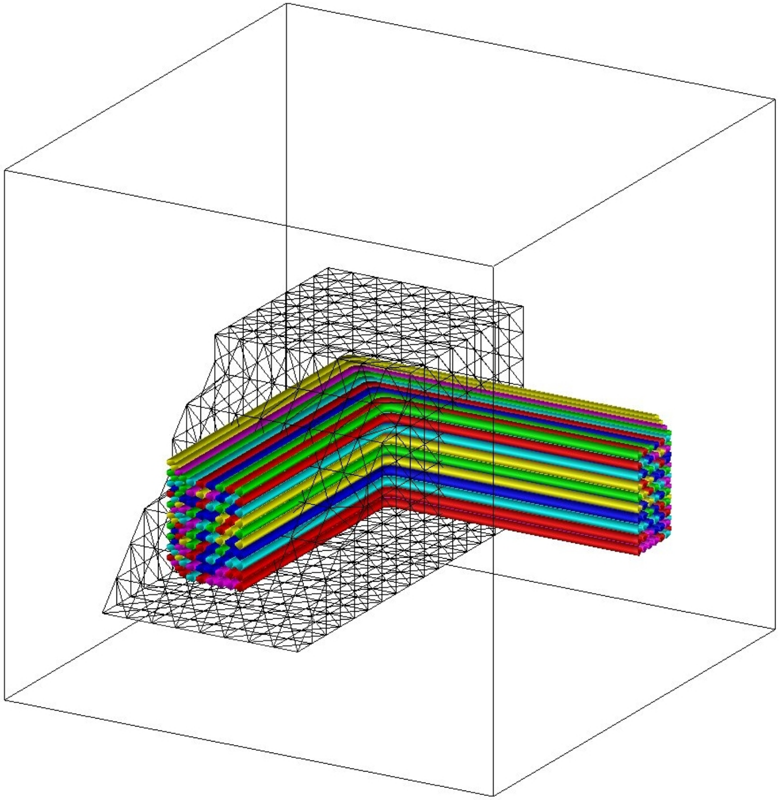
*Halted after 12 hours. Extrapolating from other experiments, ~76 hours.

cPiped10000 defined by over 40k surfaces.

Handle Common Cases (even if complex)

Often geometric models have repetitive structure.

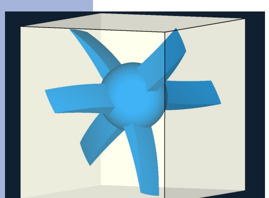
Use the repetition to decide how to process models more efficiently.



Conclusion

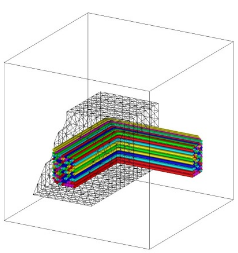
Framework that computes geometric properties for each comp of a model.

Based on a minimal, extensible set of predicates that handles any model & is very efficient on common cases.



Bounding Box

DLM, D. P. Griesheimer, B. R. Nease, and J. Snoeyink, "Computing Numerically-Optimal Bounding Boxes for Constructive Solid Geometry (CSG) Components in Monte Carlo Transport Calculations", SNA+MC 2013: Joint International Conference on Super Computing in Nuclear Applications + Monte Carlo, 2013



Volume

DLM, D. P. Griesheimer, B. R. Nease, J. Snoeyink, "Robust Volume Calculations for Constructive Solid Geometry Components (CSG) in Monte Carlo Transport Calculations", PHYSOR 2012: Advances in Reactor Physics, 2012

Contact: David L. Millman · dave@cs.unc.edu

http://cs.unc.edu/~dave

Bibliography

[LG98] M. C. Lin and S. Gottschalk, “Collision Detection Between Geometric Models: A Survey,” (1998).

[DLL+08] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean, “Near-optimal parameterization of the intersection of quadrics: I. The generic algorithm,” Journal of Symbolic Computation, 43, 3, 168-191 (2008).

[SW06] E. Schömer and N. Wolpert, “An exact and ecient approach for computing a cell in an arrangement of quadrics,” Computation Geometry Theory and Applications, 33, 1-2, 65-97 (2006).

[K00] J. Keyser, “Exact Boundary Evaluation for Curved Solids”, PhD thesis, University of North Carolina-Chapel Hill, 2000.

[MTT05] B. Mourrain, J.-P. Técourt, and M. Teillaud, “On the Computation of an Arrangement of Quadrics in 3d,” Computational Geometry Theory and Application, 30. 2, 145-164, 2005

[M12] D. L. Millman, “Degree-Driven Design of Geometric Algorithms for Point Location, Proximity, and Volume Calculation”, PhD thesis, University of North Carolina–Chapel Hill, 2012.

[NMGG13] B. R. Nease, D. L. Millman, D. P. Griesheimer, D. F. Gill, “Geometric Templates for Improved Tracking Performance in Monte Carlo Codes”, SNA+MC 2013