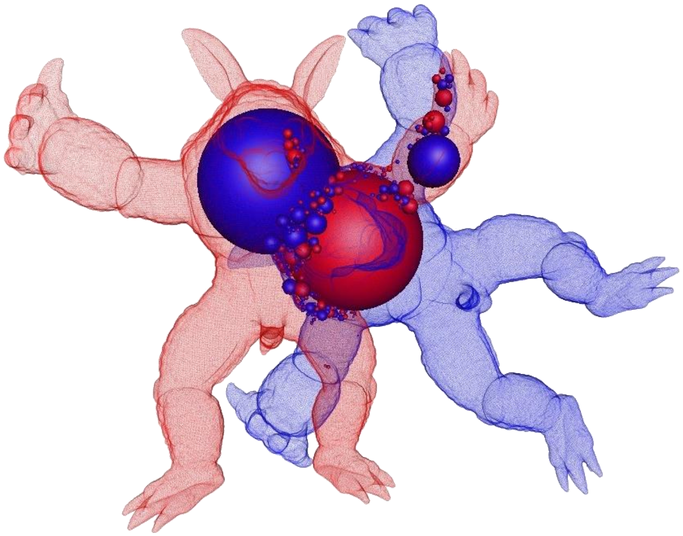
Bremen

Computational Geometry

Of Collisions and Spheres

and their Application to Computer Graphics and Beyond



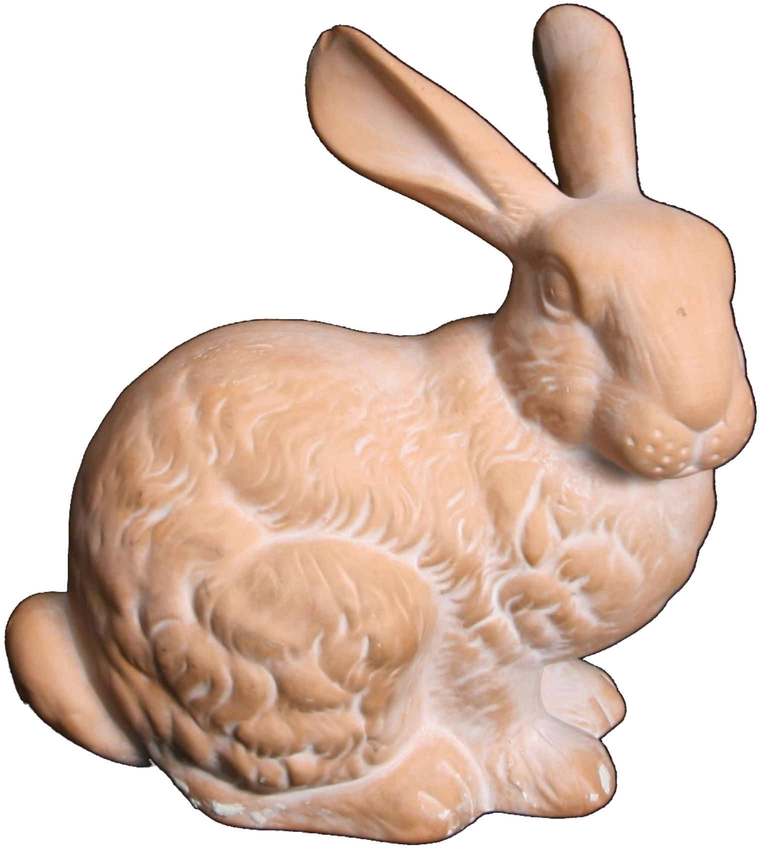
Motivation

[Rise of the Tomb Raider]

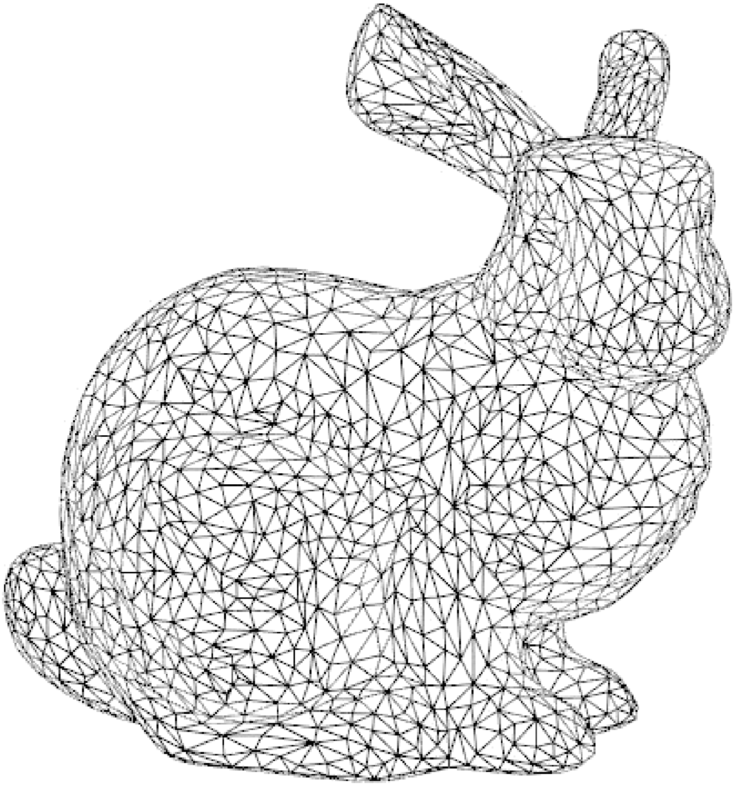
[RTT Deltagen 12]

Motivation

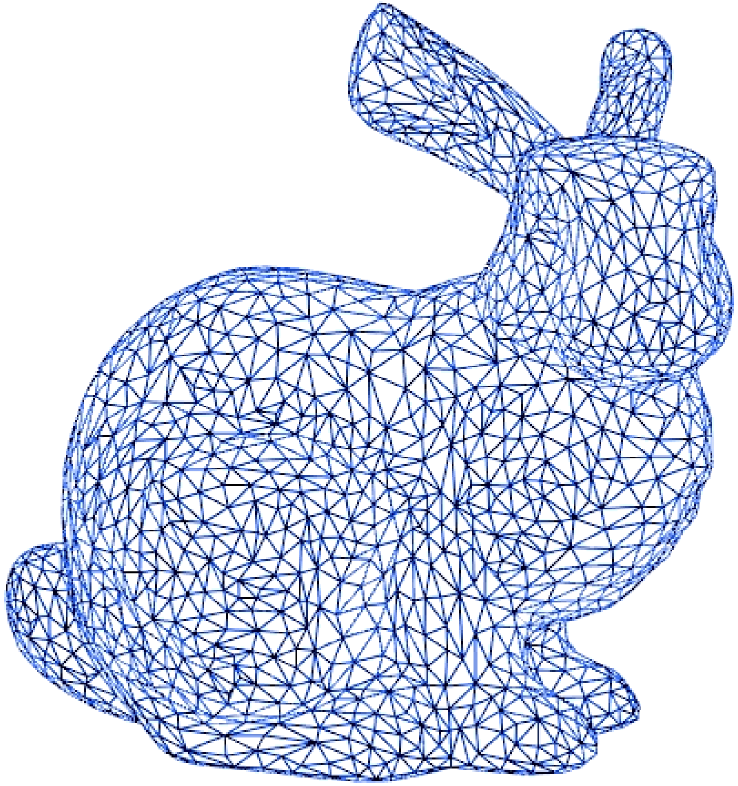
Collision Detection Basics

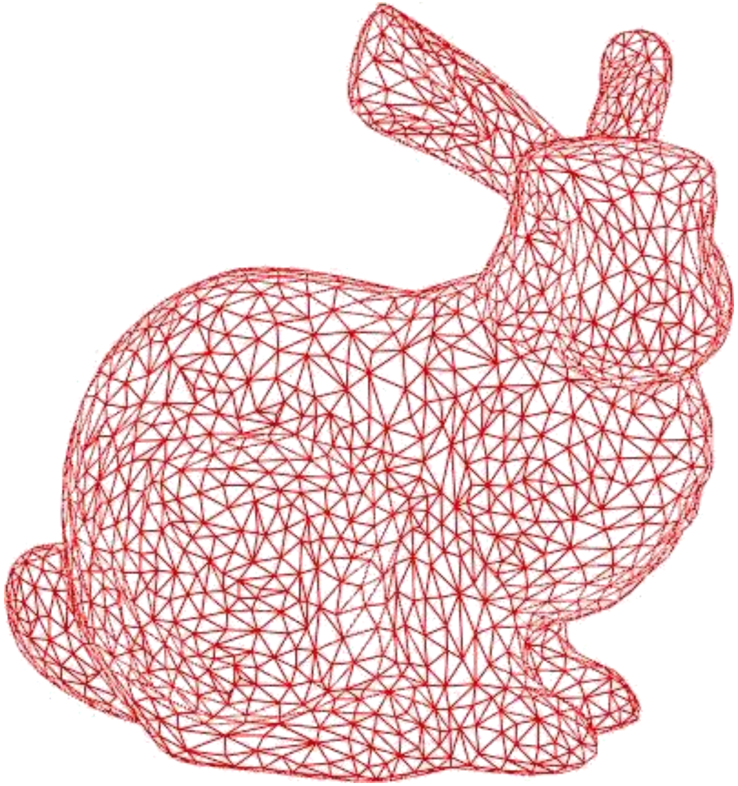


Collision Detection Basics

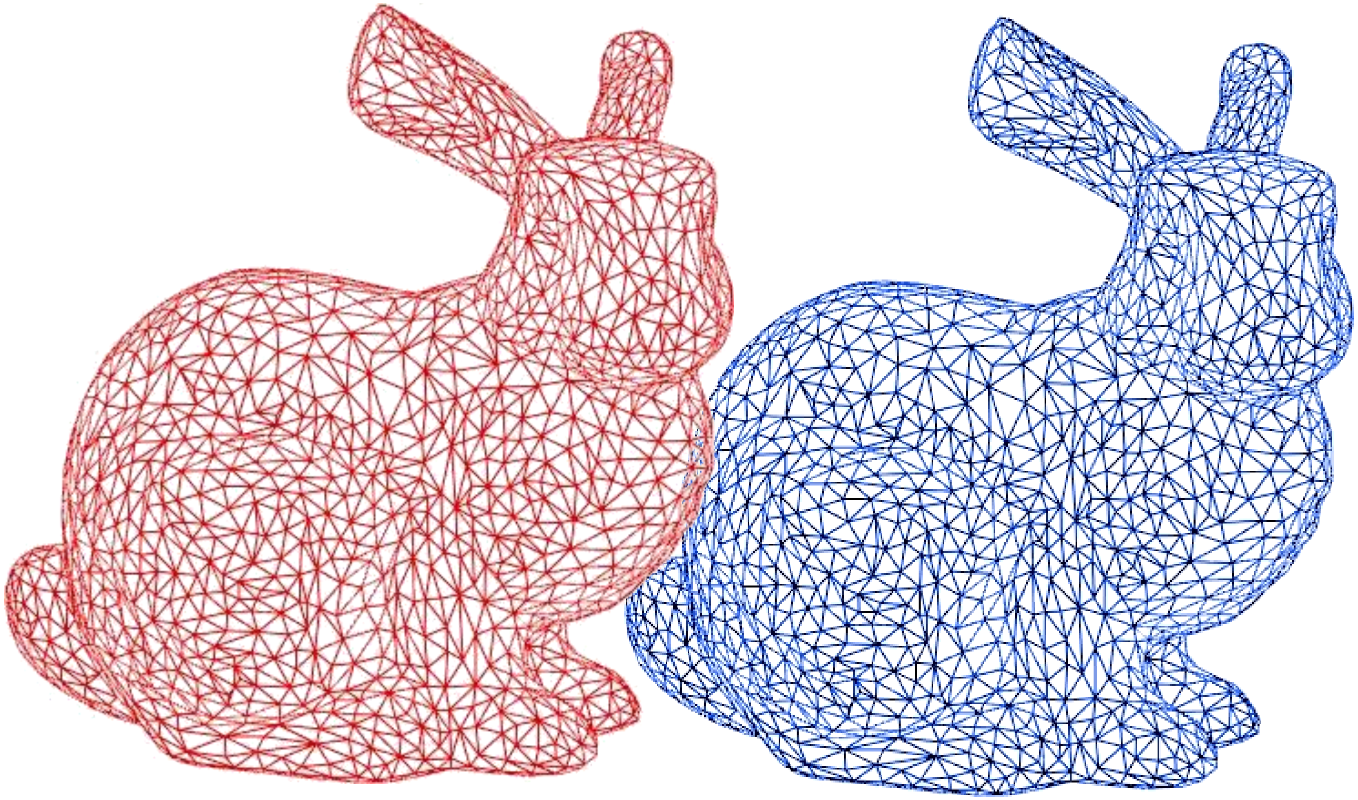


Collision Detection Basics

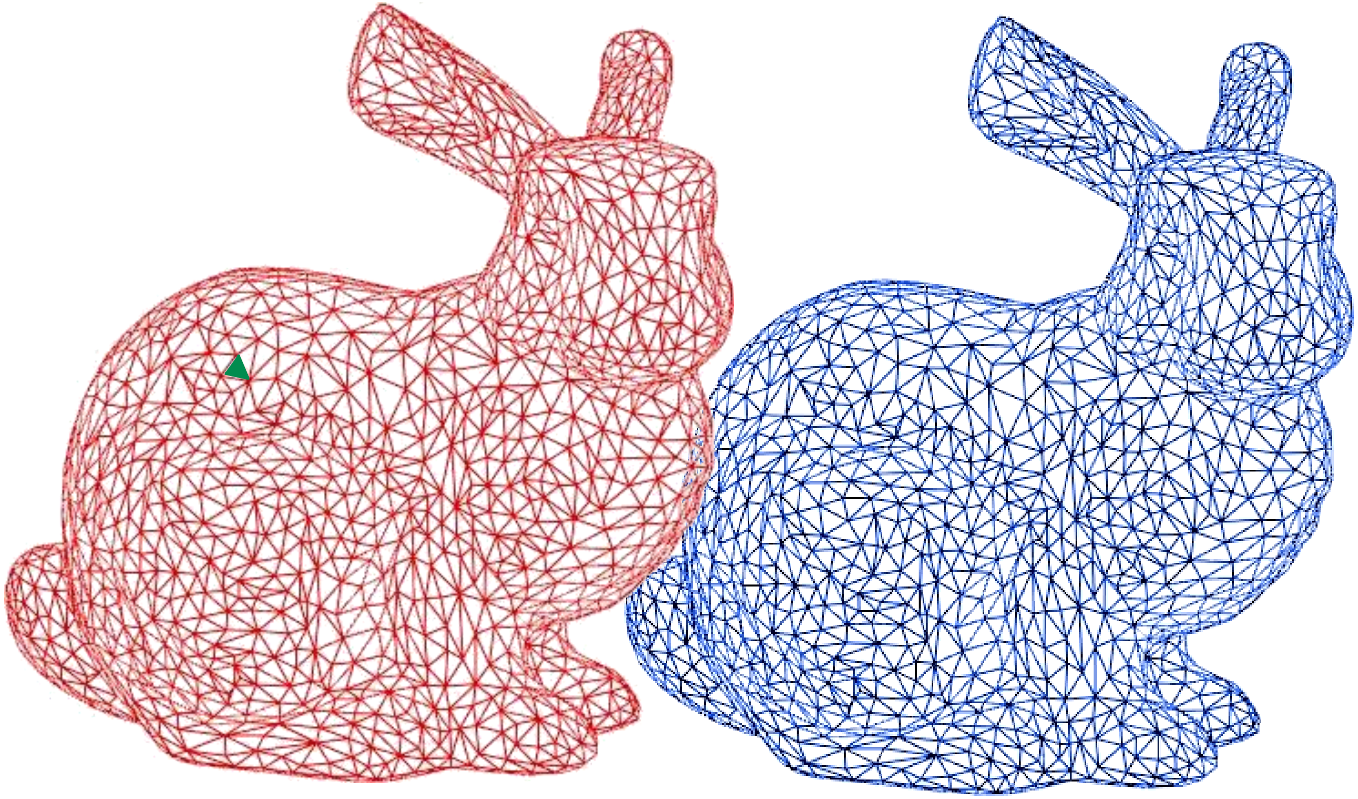




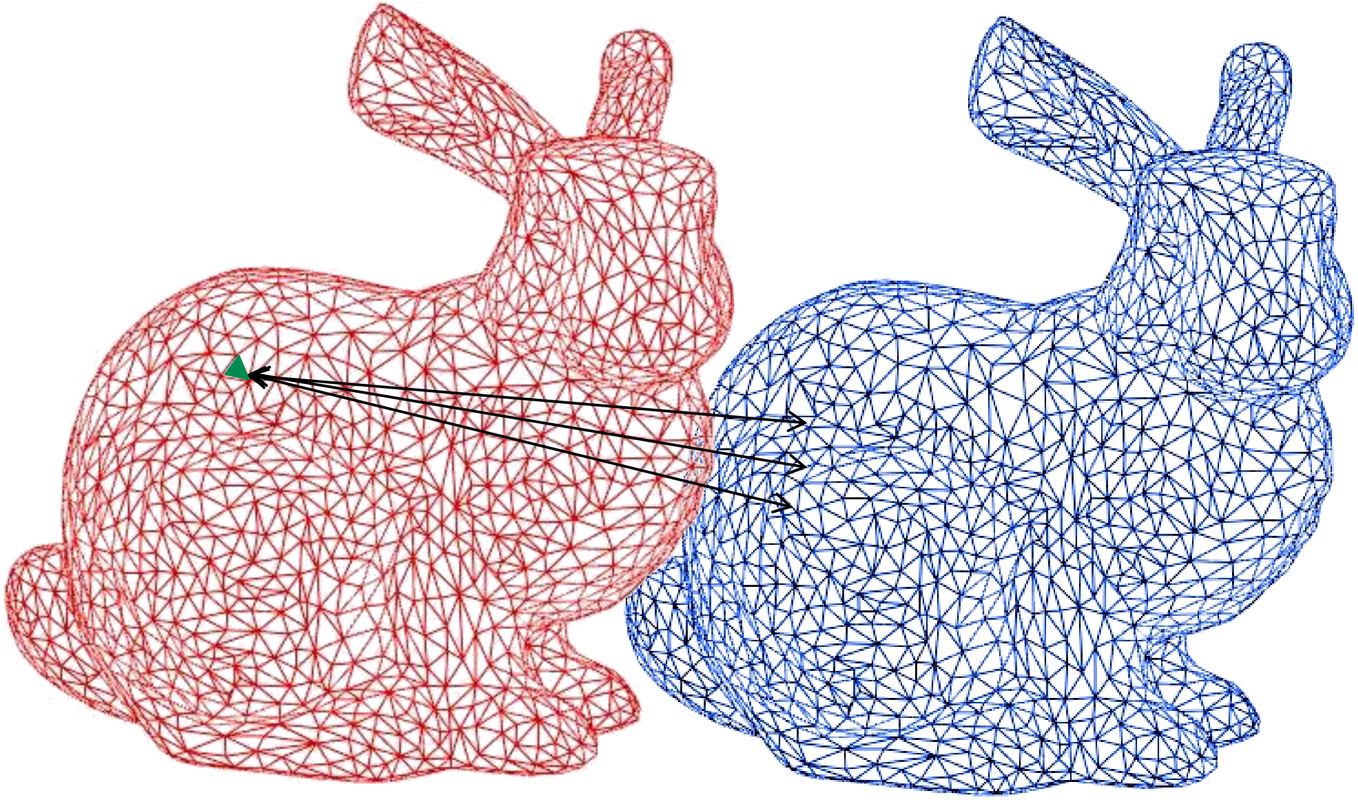
Collision Detection Basics



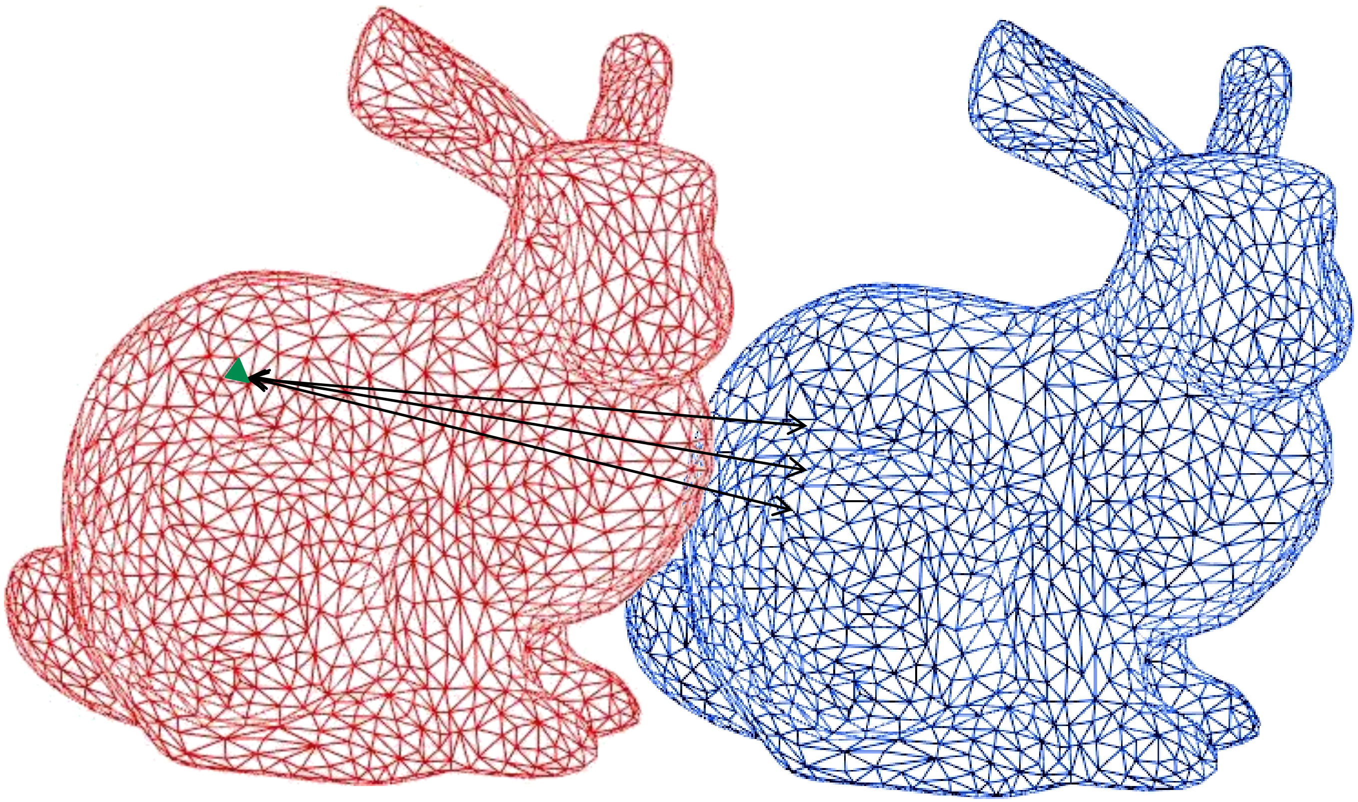
Collision Detection Basics



Collision Detection Basics

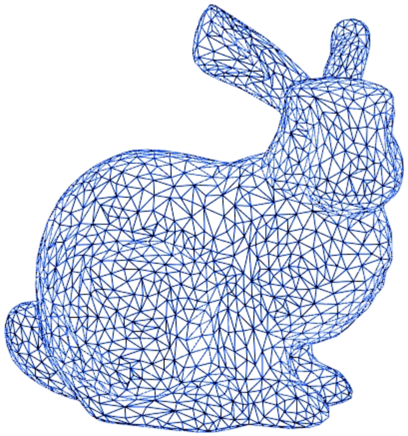


Collision Detection Basics

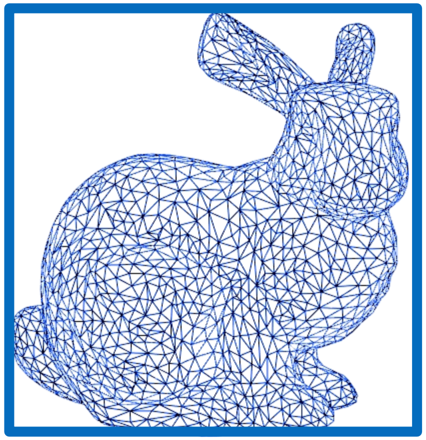


$$O(n^2)$$

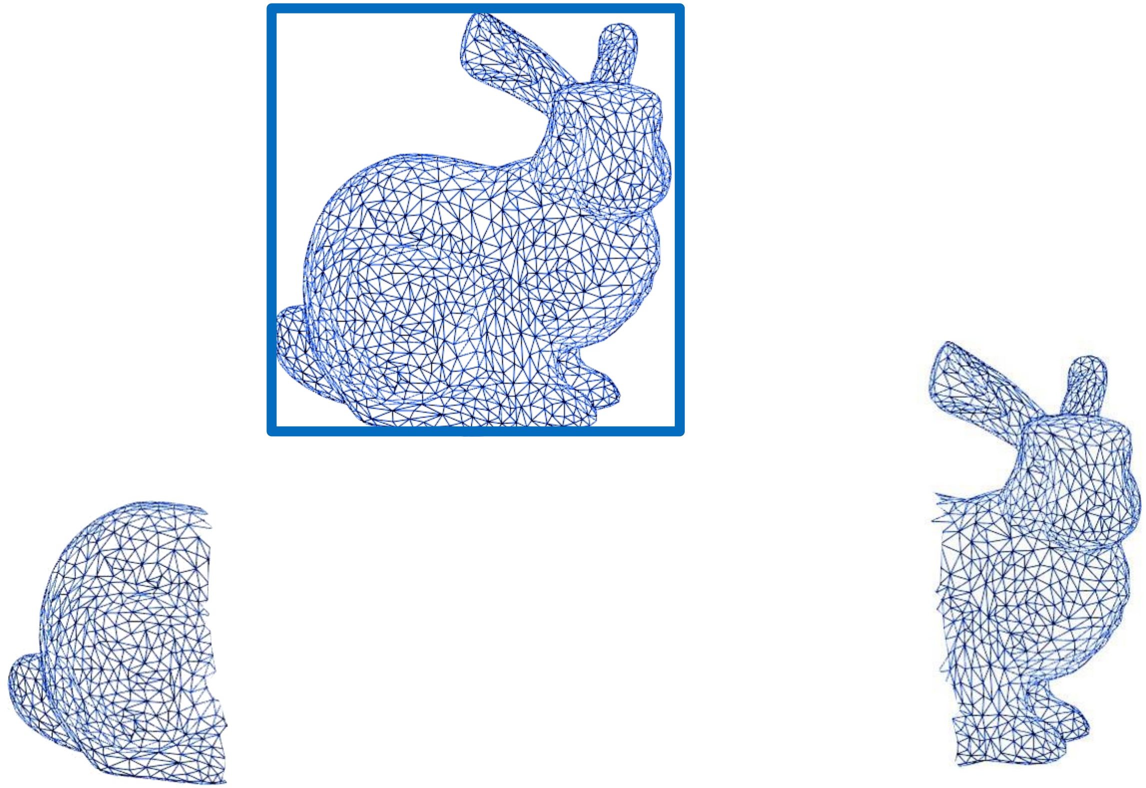
Bounding Volume Hierarchies



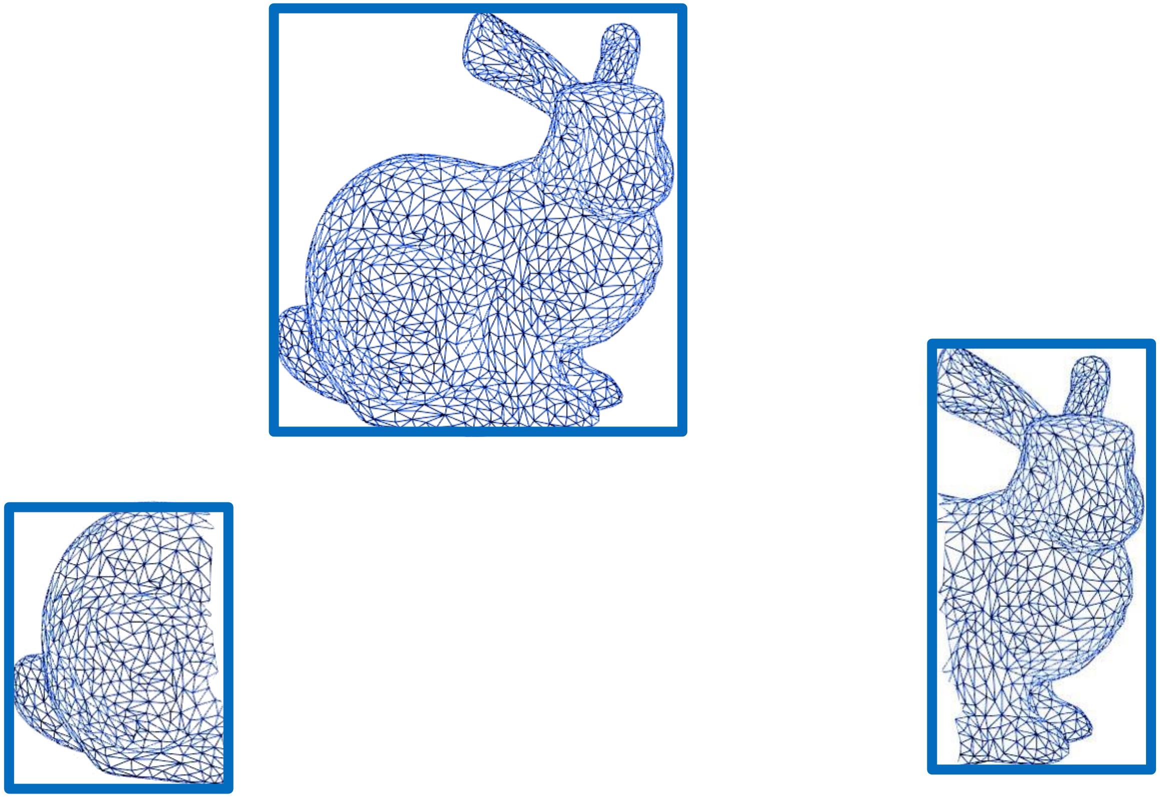
Bounding Volume Hierarchies



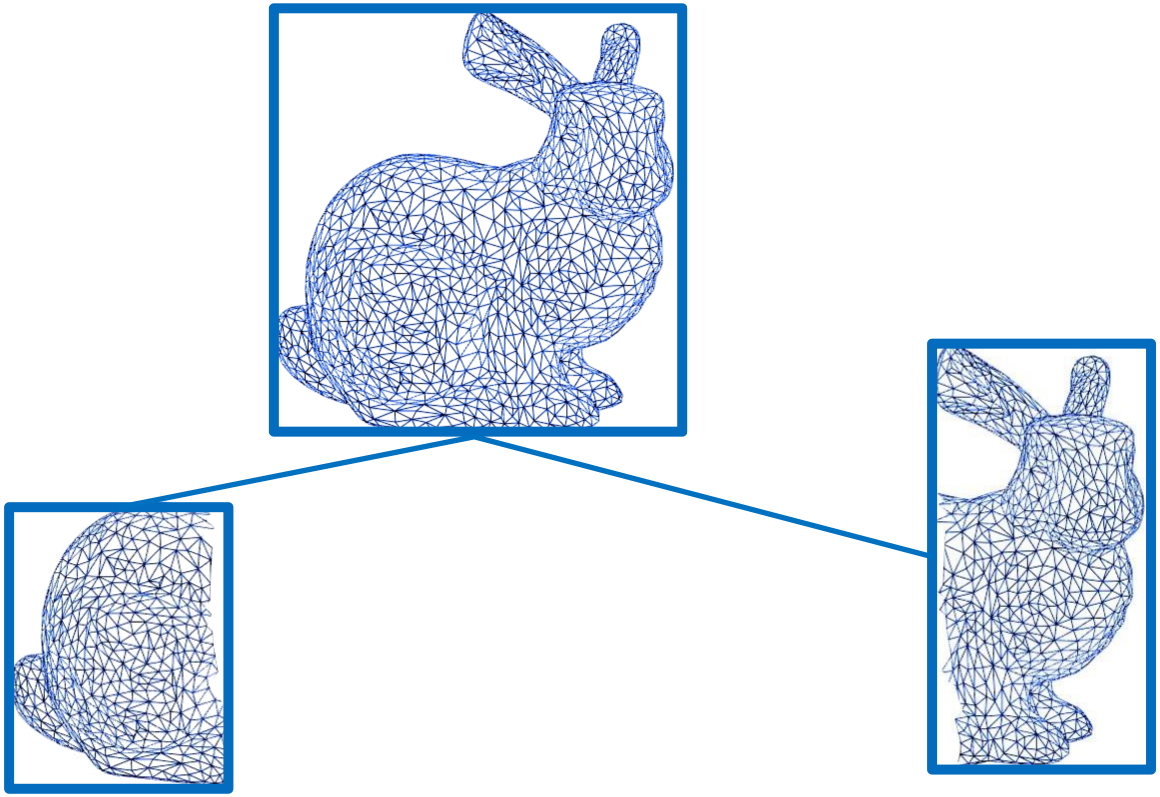
Bounding Volume Hierarchies



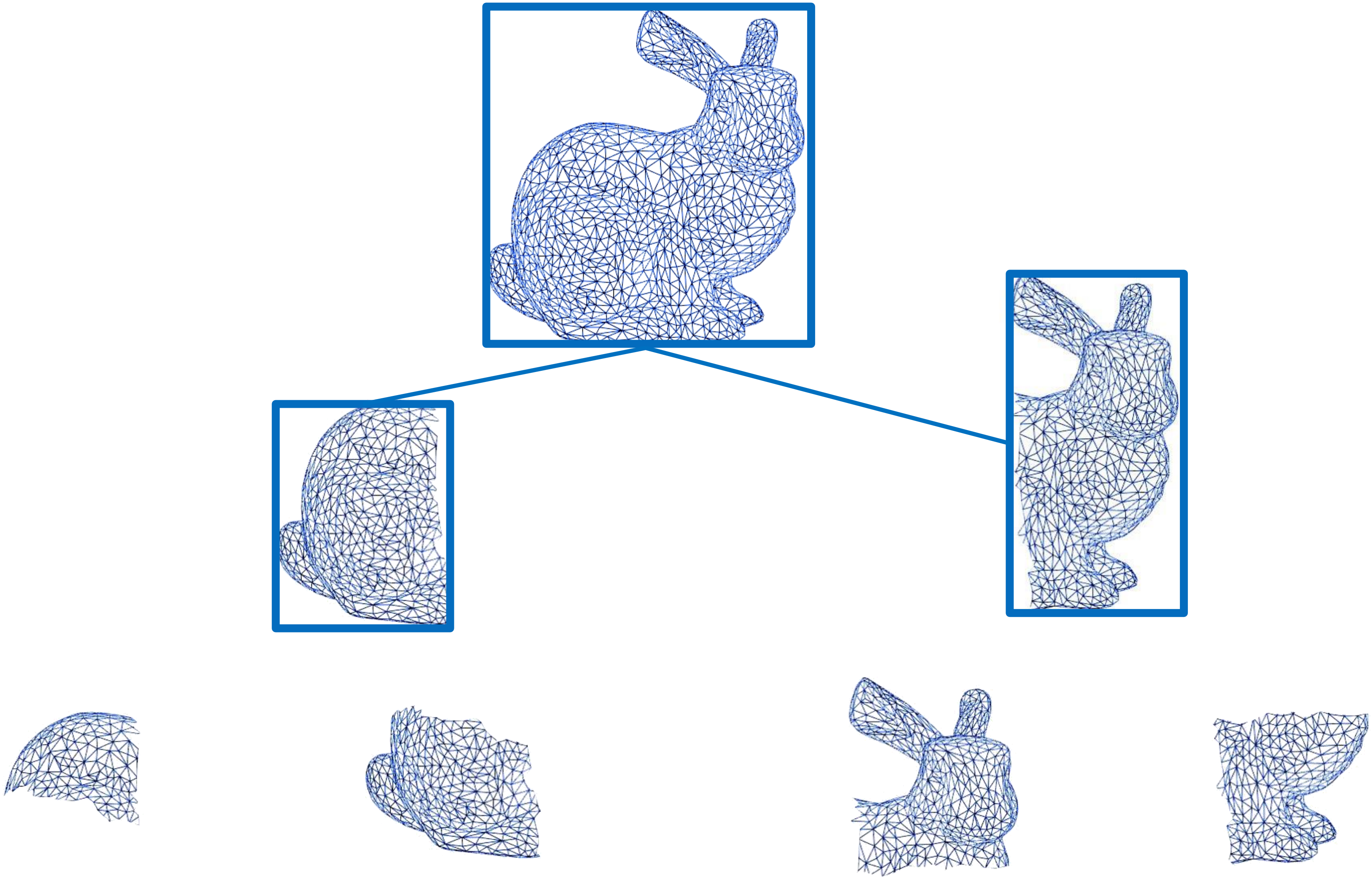
Bounding Volume Hierarchies



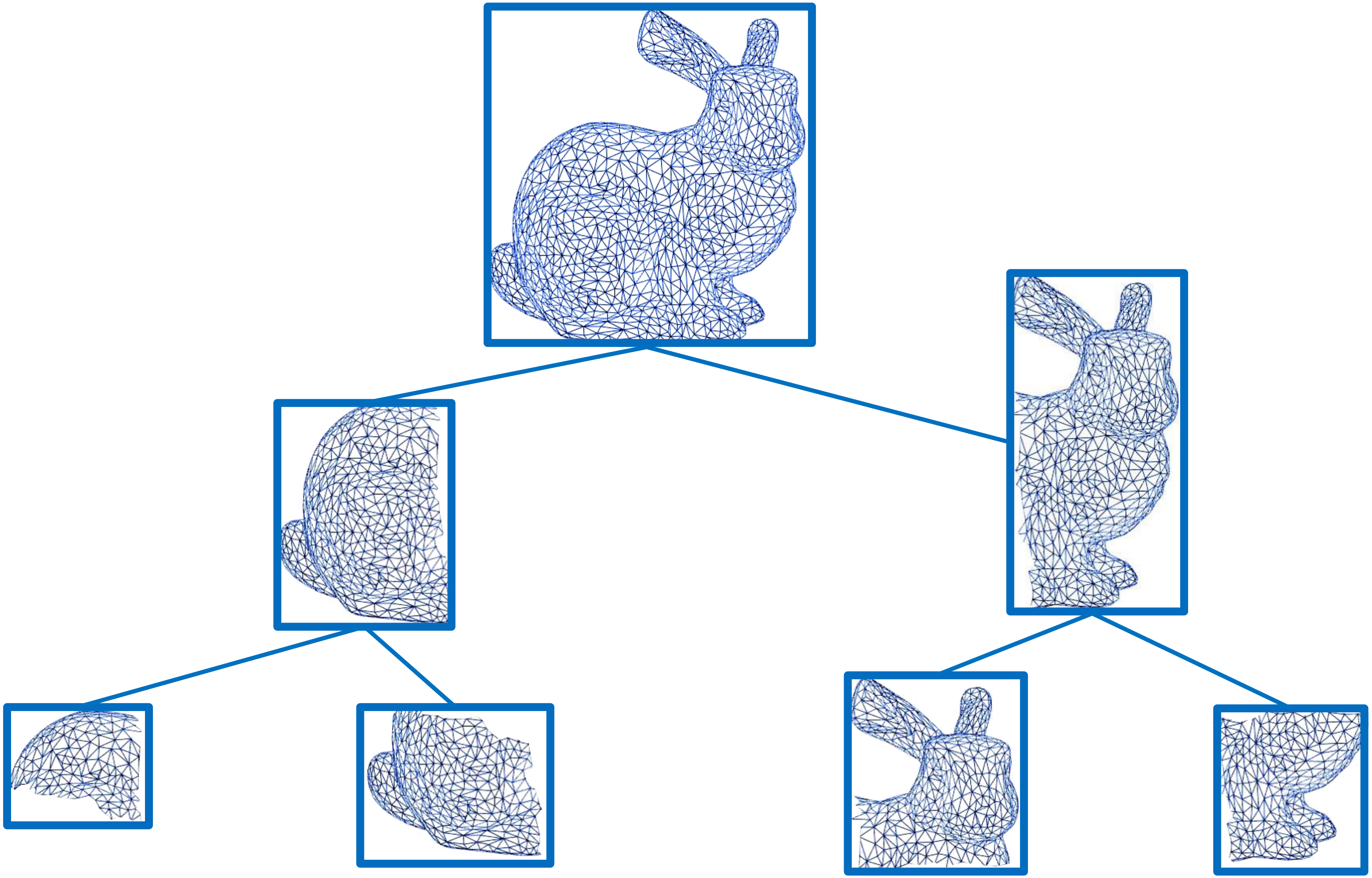
Bounding Volume Hierarchies



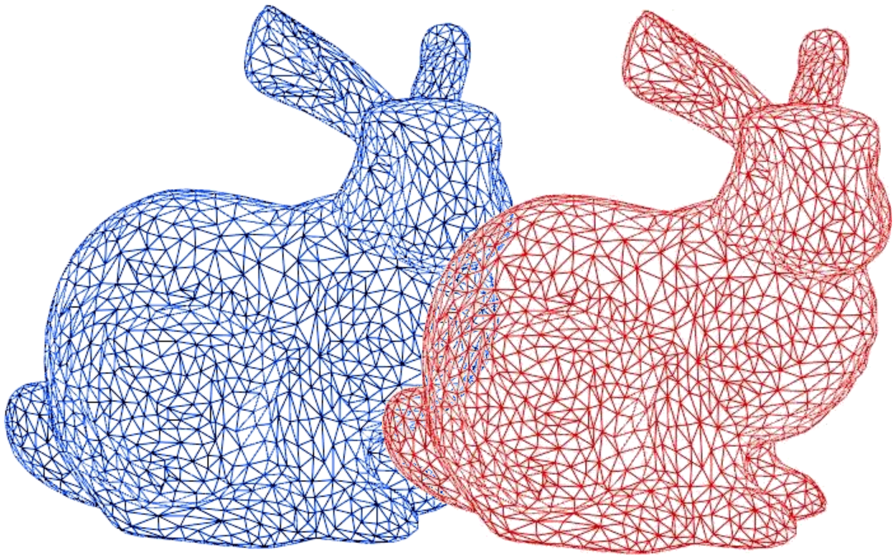
Bounding Volume Hierarchies



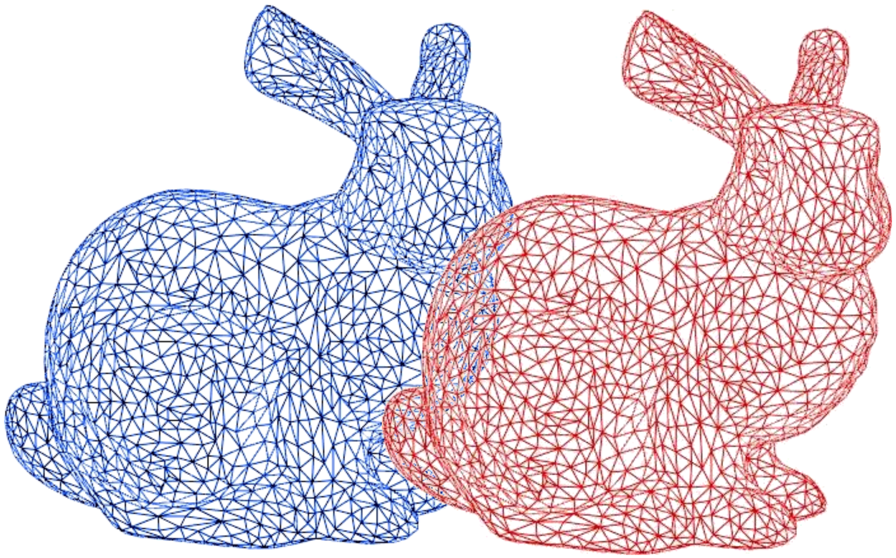
Bounding Volume Hierarchies

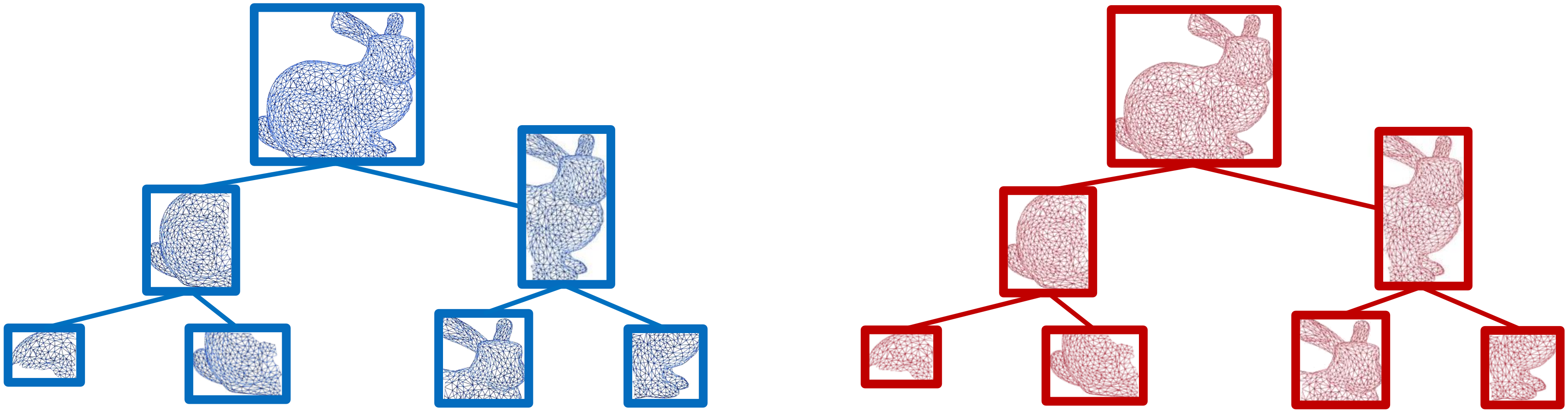


BVH Traversal

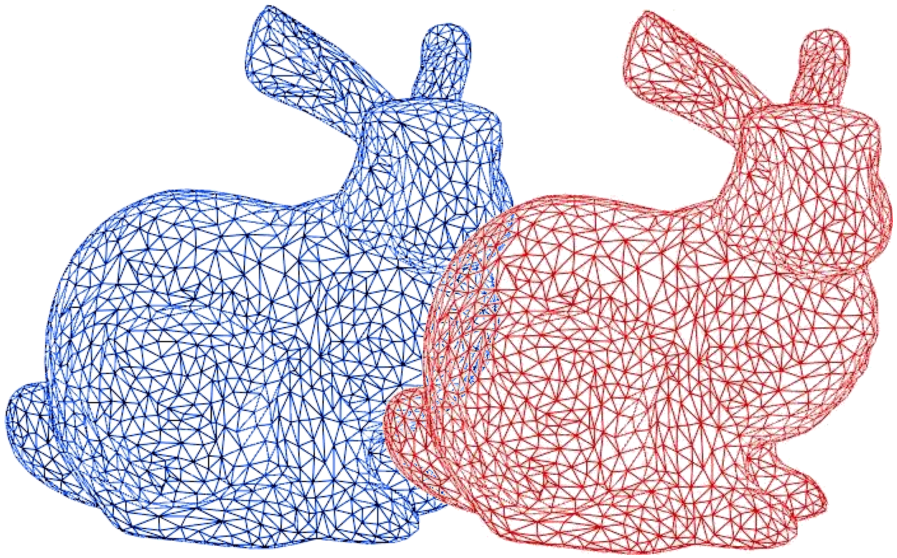


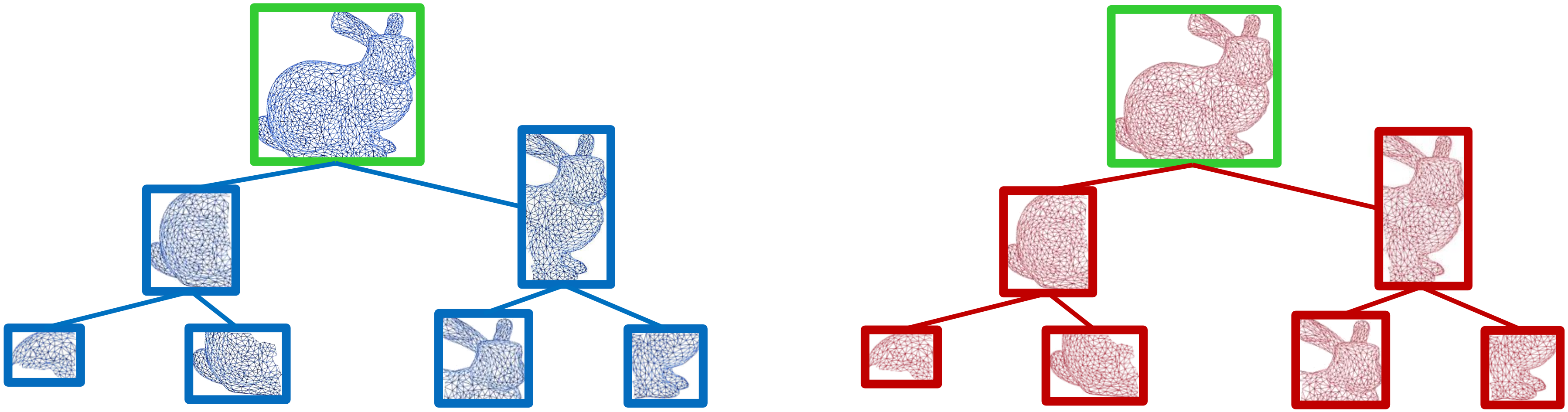
BVH Traversal



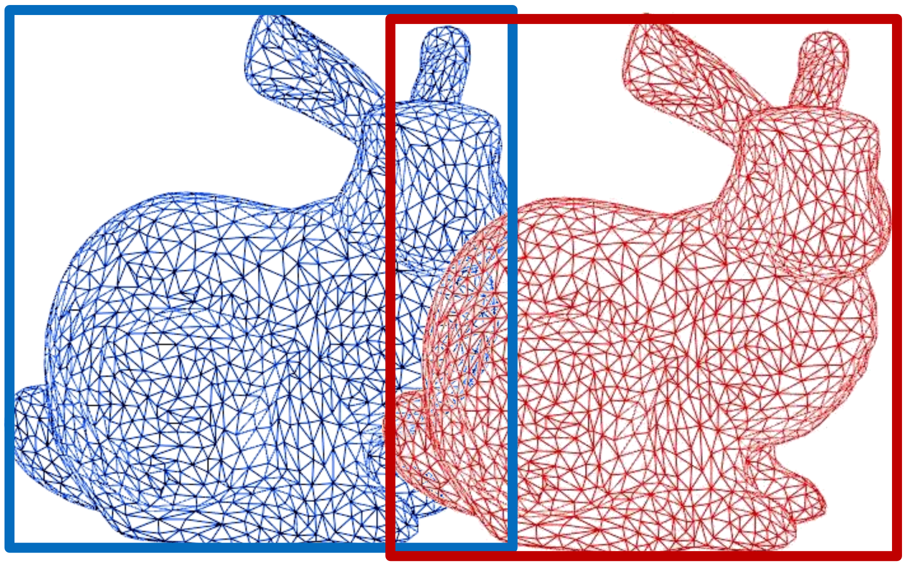


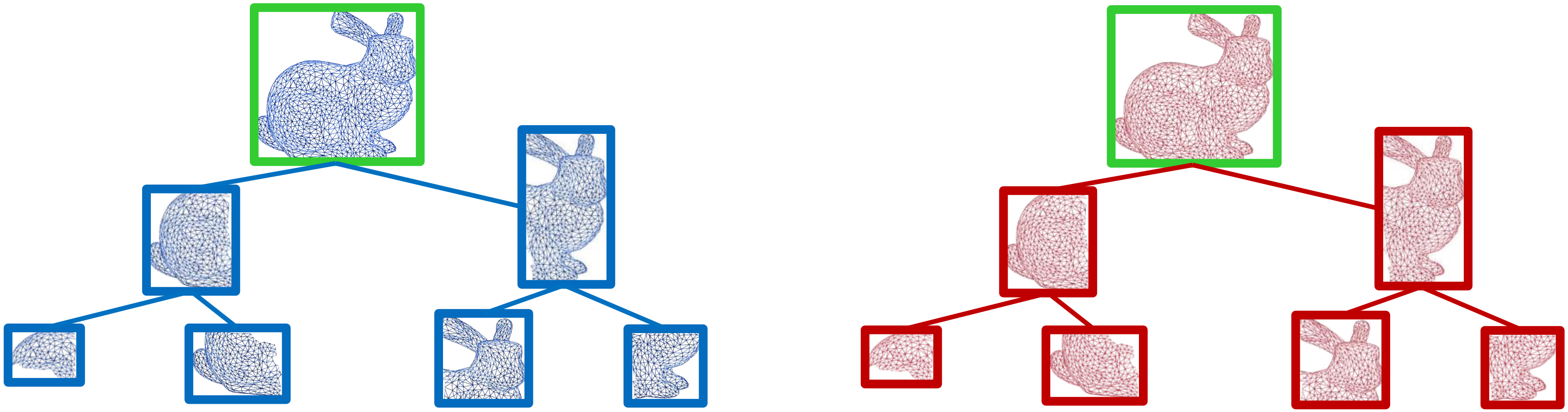
BVH Traversal



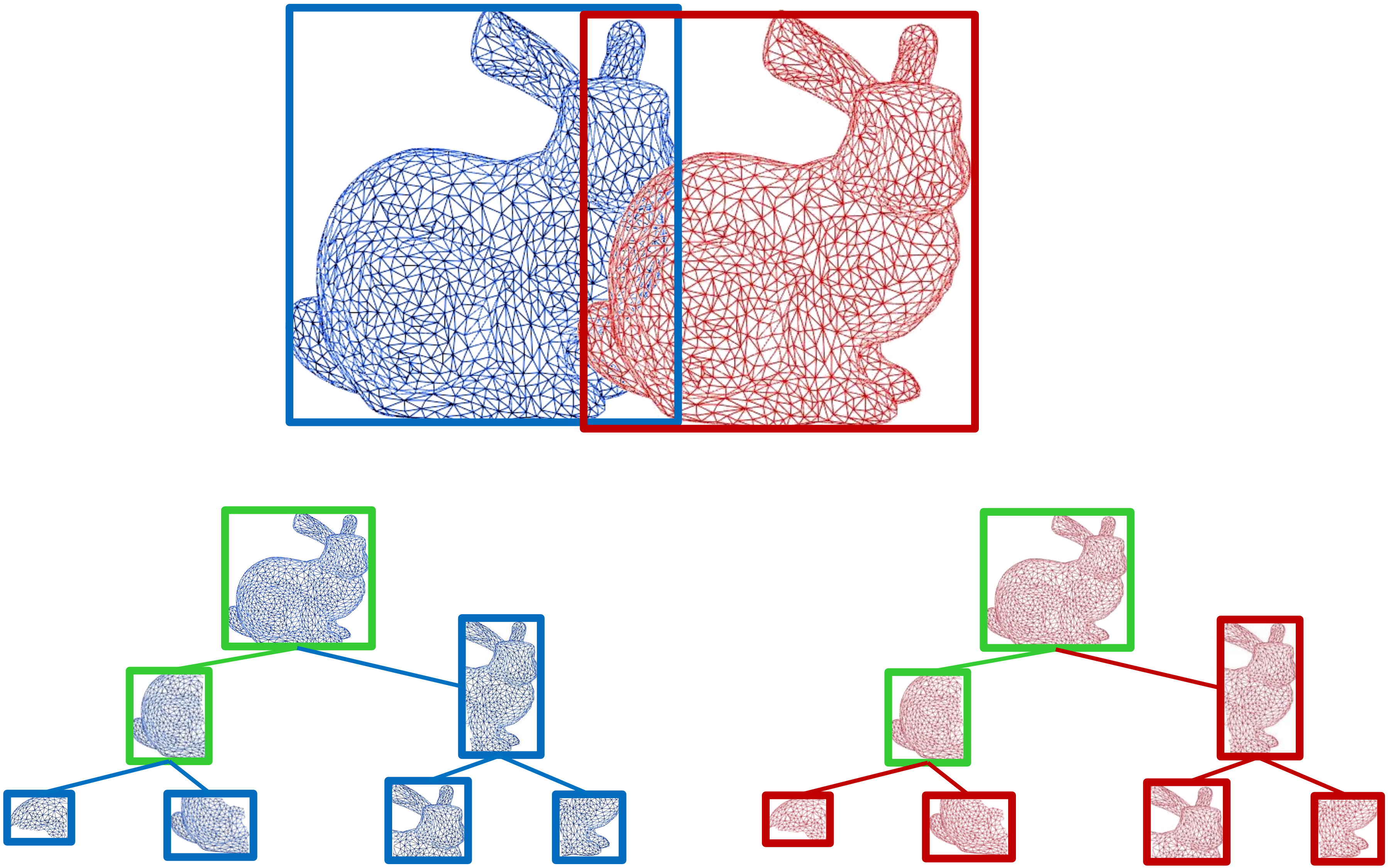


BVH Traversal

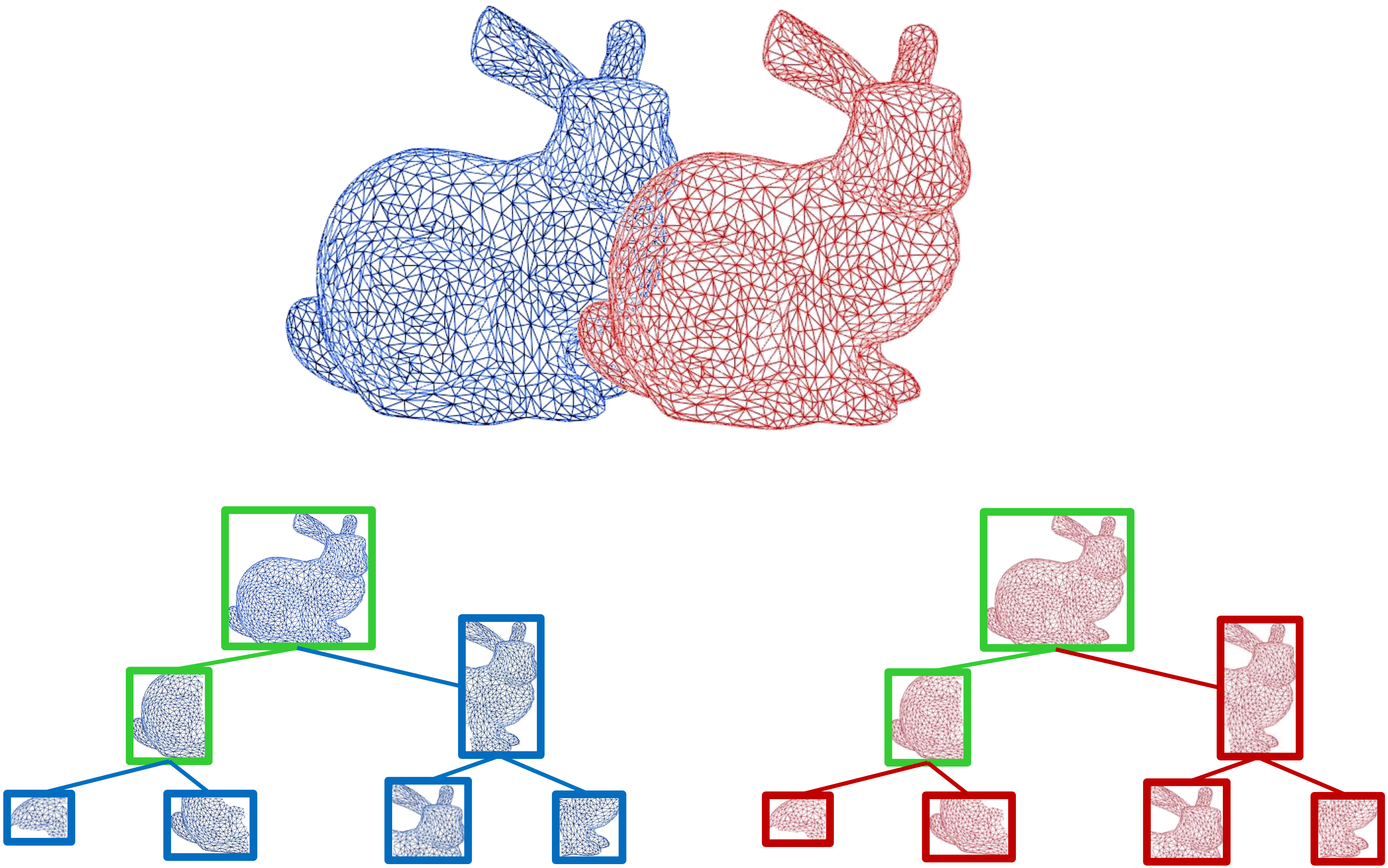




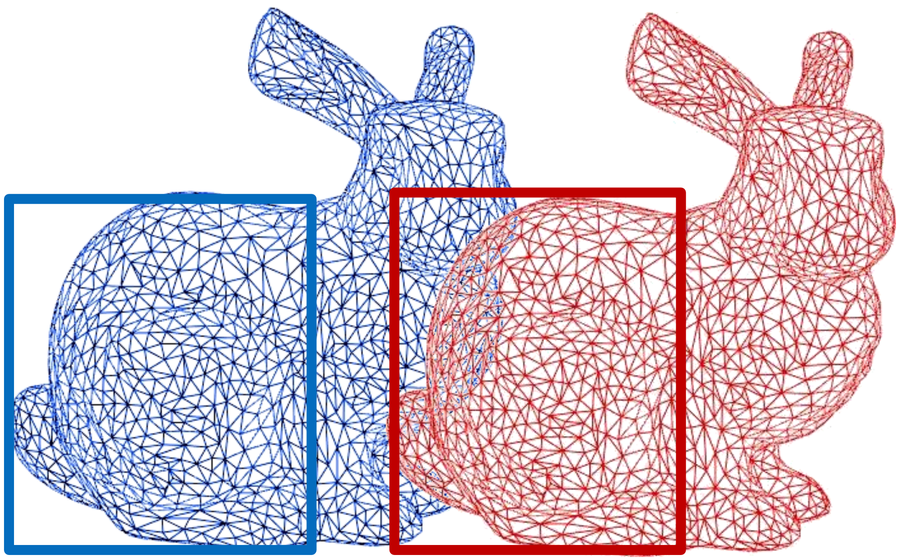
BVH Traversal

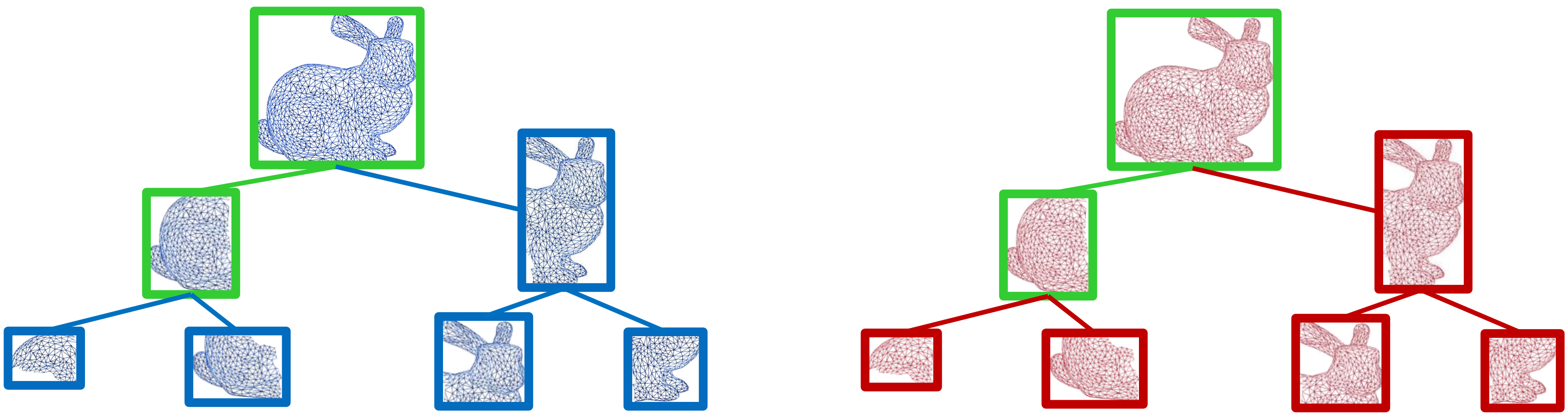


BVH Traversal



BVH Traversal



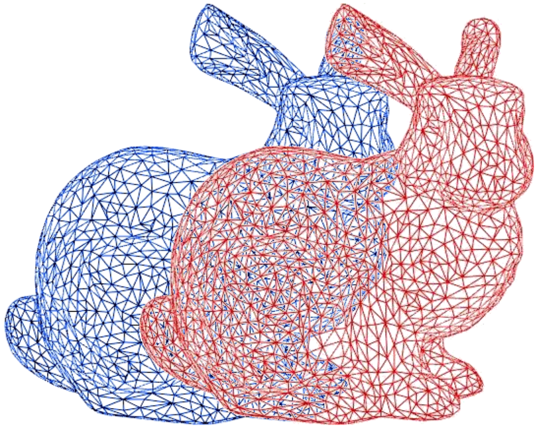


Simultaneous Recursive BVH Traversal

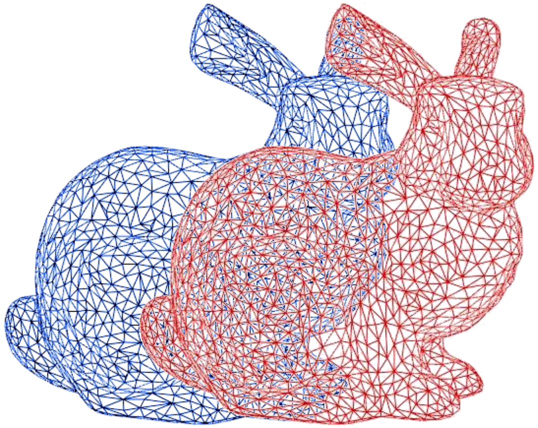
Simultaneous Recursive BVH Traversal

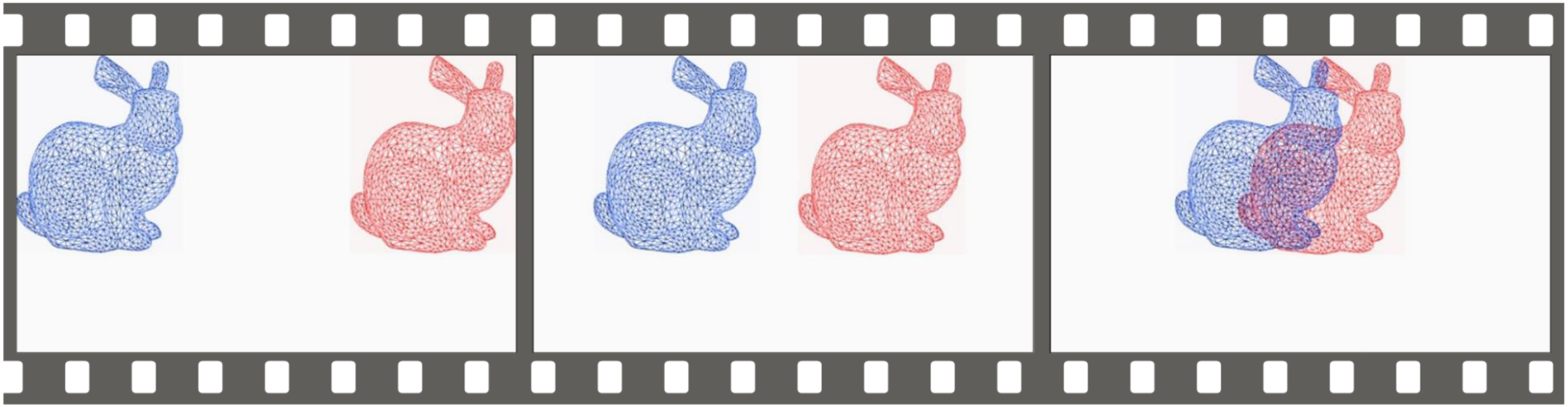
```
bool checkCollision( BV A, BV B )
  if A and B are Leaves then
    return checkPolygons(Polygon of A, Polygon of B)
  else
  forall the Children Ai of A do
    forall the Children Bi of B do
       if( overlap (Ai , Bi))
         return checkCollision(Ai , Bi)
  return false
```

Discrete Collision Detection

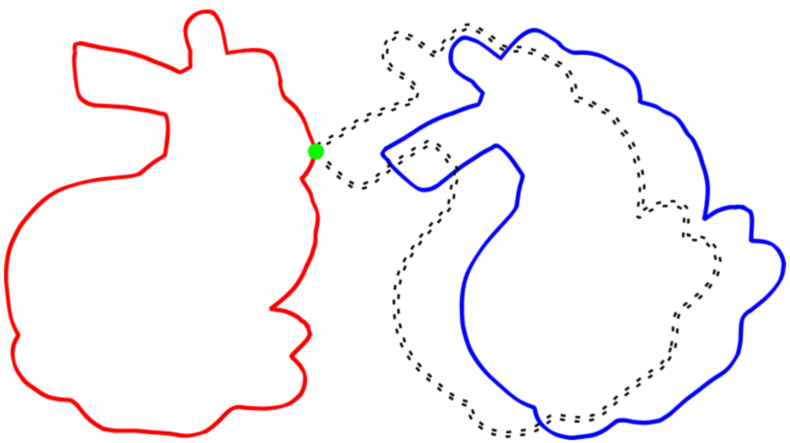


Discrete Collision Detection



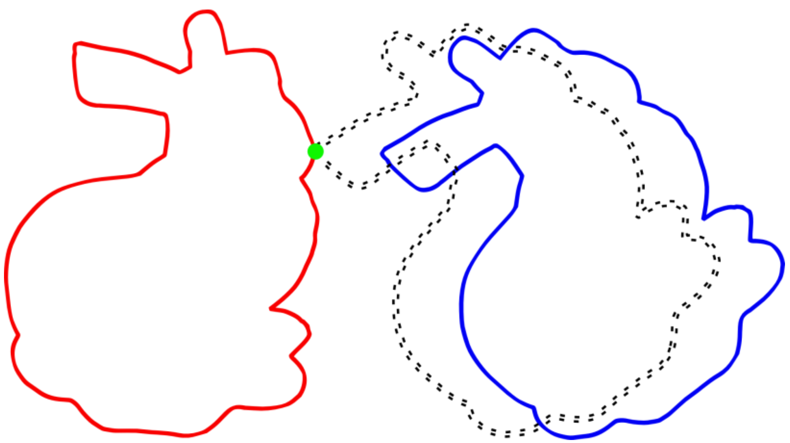


Penetration Measures

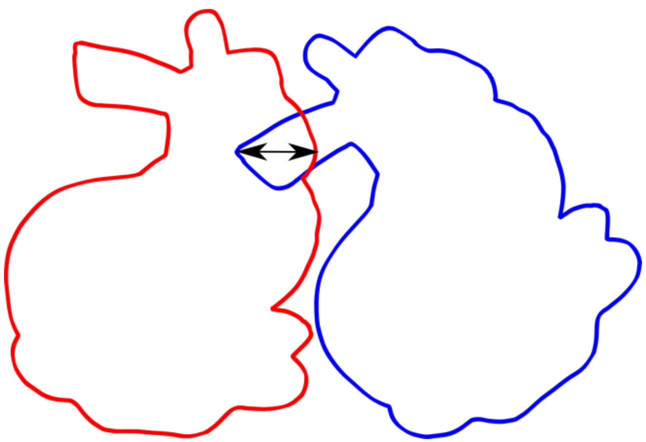


Continuous collision detection

Penetration Measures

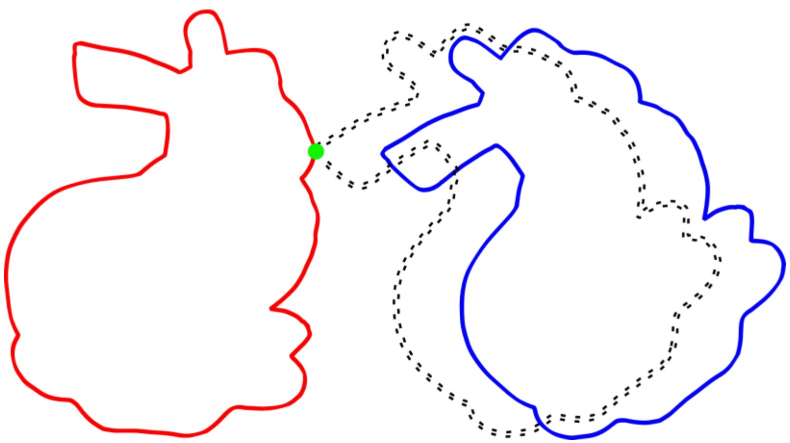


Continuous collision detection

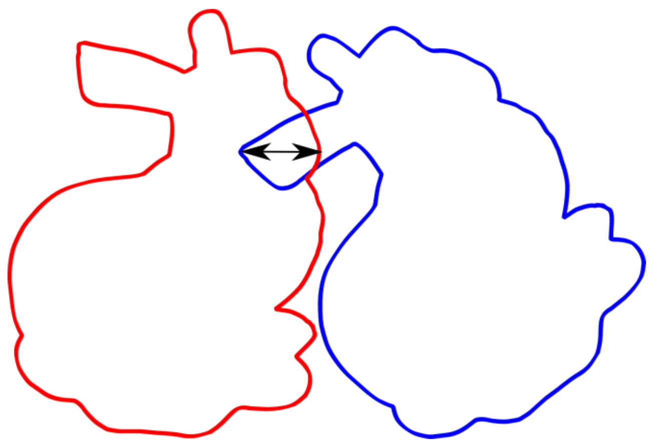


Translational penetration depth

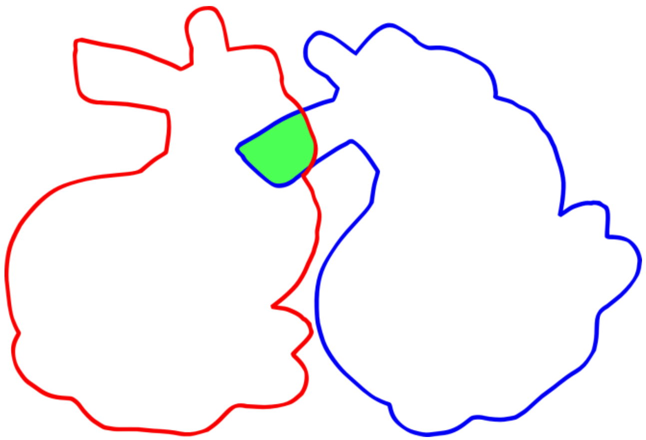
Penetration Measures



Continuous collision detection

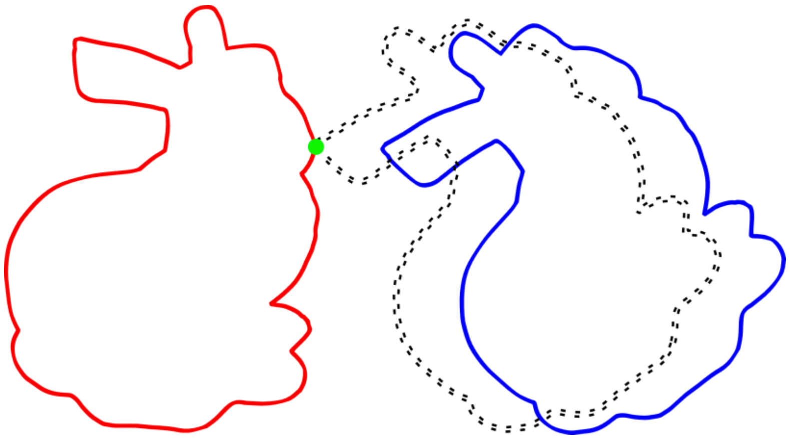


Translational penetration depth

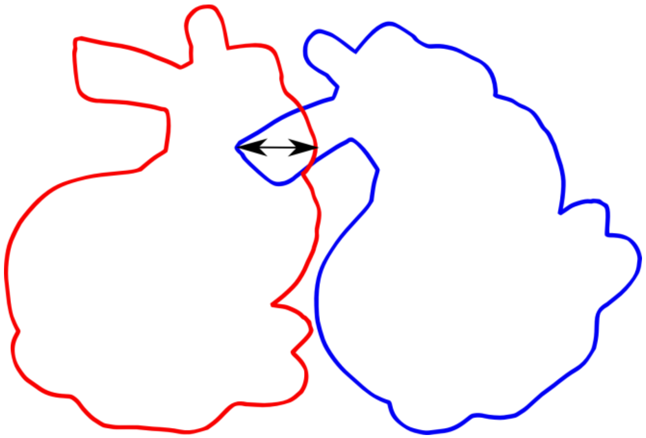


Penetration volume - “the most complicated yet accurate method” [Fisher and Lin, 2001]

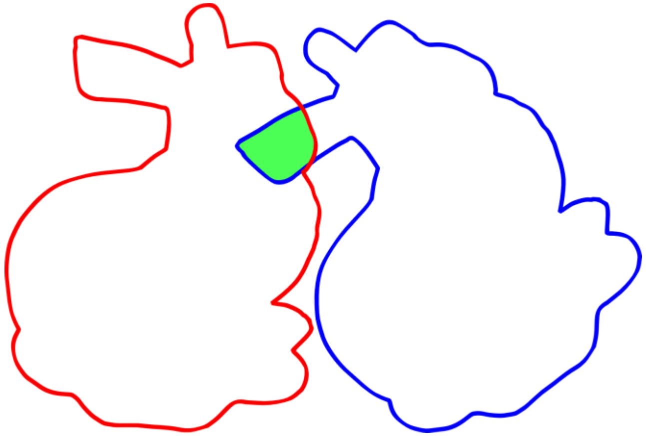
Penetration Measures



Continuous collision detection

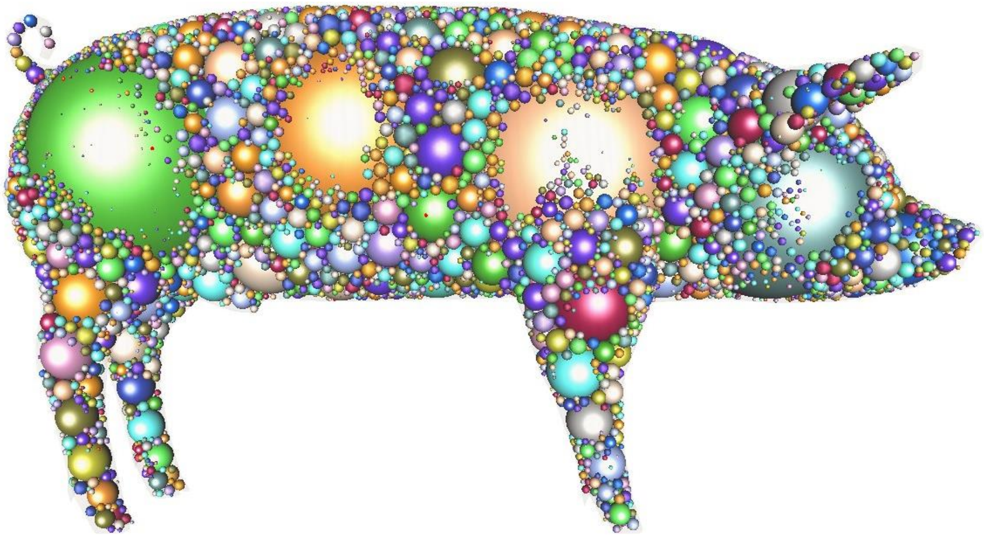


Translational penetration depth

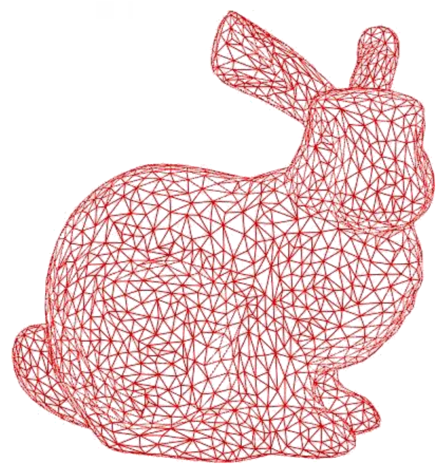


Penetration volume - “the most complicated yet accurate method” [Fisher and Lin, 2001]

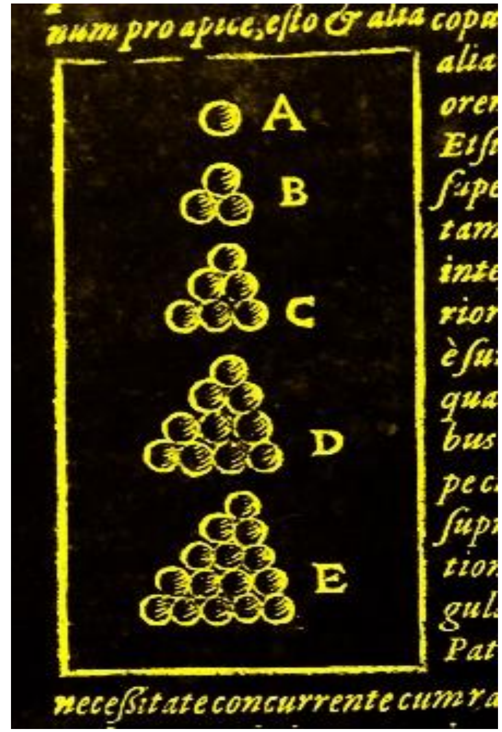
- Fill the object
  - from the *inside*
  - with *non-overlapping* spheres
- Build sphere hierarchy on *inner* spheres



How to get the Spheres into the Object?

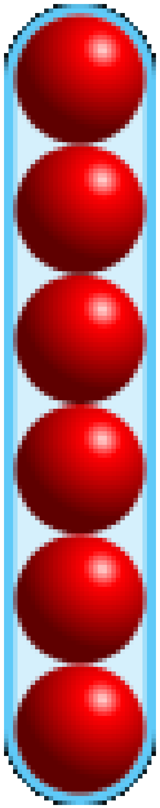


A brief History of Sphere Packings

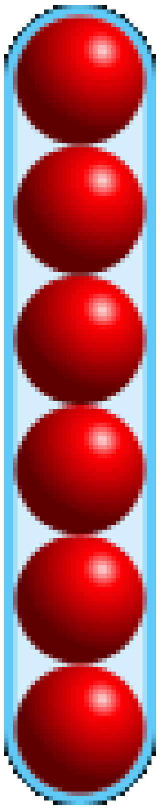


Johannes Kepler (1571 – 1630)

Excurse: The Sausage Conjecture

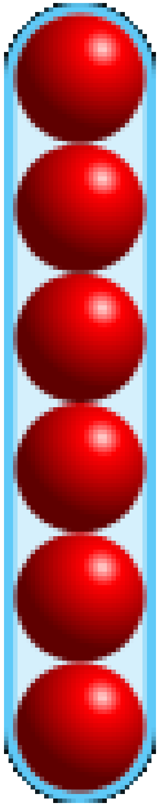


Excurse: The Sausage Conjecture

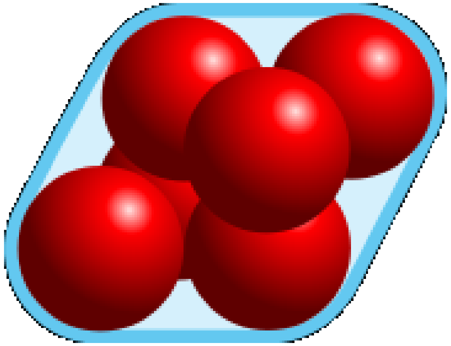


Sausage

Excurse: The Sausage Conjecture



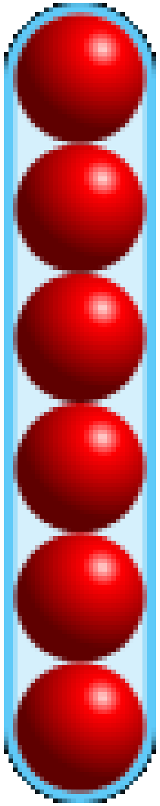
Sausage

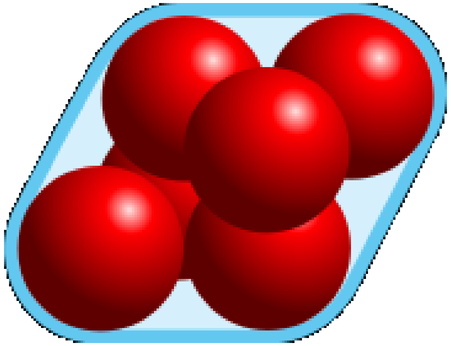


Cluster

Excurse: The Sausage Conjecture


:optimal for n=3 and 4

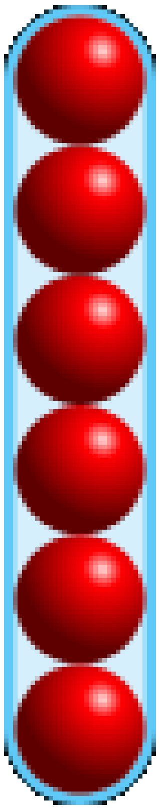




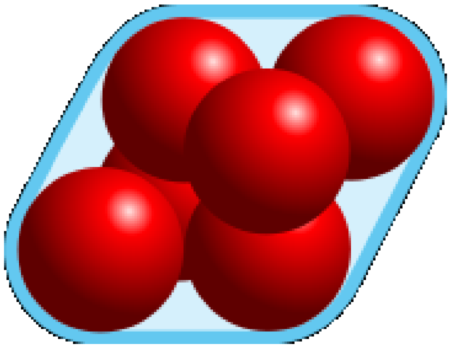
Cluster

Sausage

Excurse: The Sausage Conjecture



Sausage

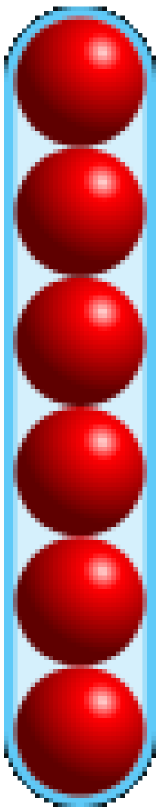


Cluster

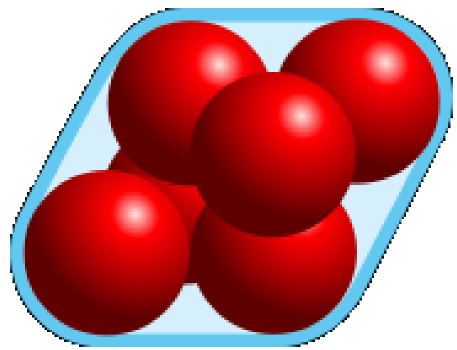

:optimal for n=3 and 4


:optimal for n<56?

Excurse: The Sausage Conjecture



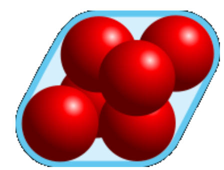
Sausage



Cluster

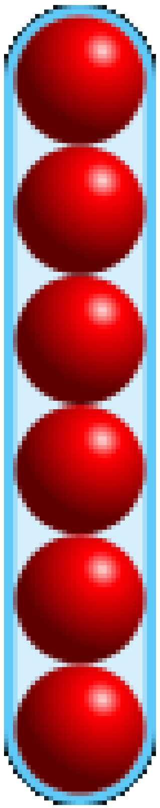
:optimal for n=3 and 4

:optimal for n<56?

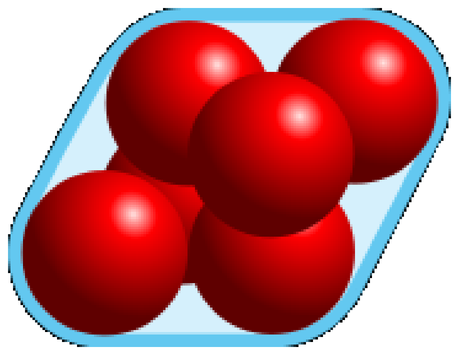


optimal for n=56, 59, 60, 61, 62, 65

Excurse: The Sausage Conjecture



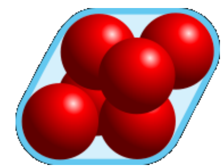
Sausage



Cluster

:optimal for n=3 and 4

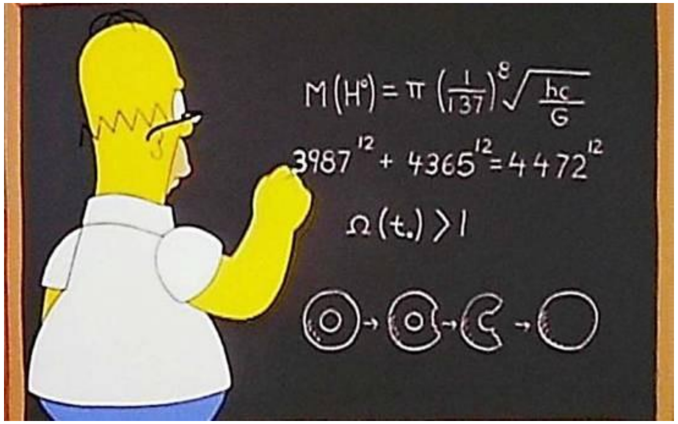
:optimal for n<56?



optimal for n=56, 59, 60, 61, 62, 65

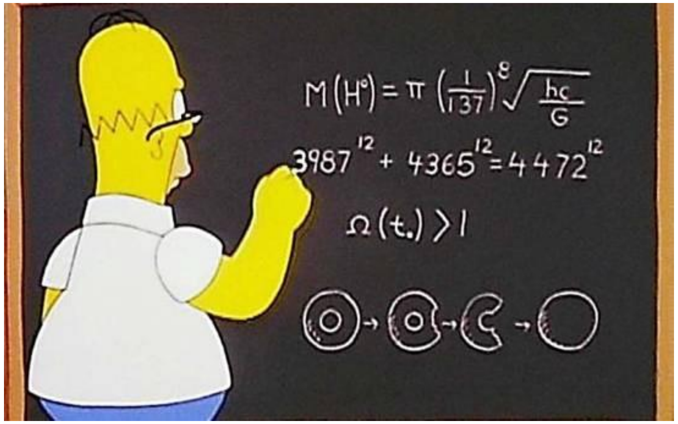
Nobody knows how they look

More Sausage Catastrophes



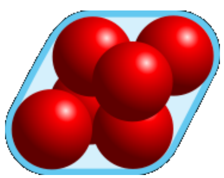
More Sausage Catastrophes

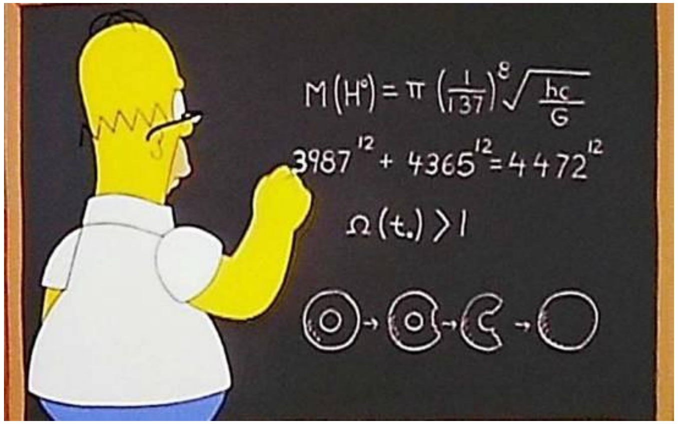
- For dimension d=4: cluster becomes optimal for n>375370?



More Sausage Catastrophes

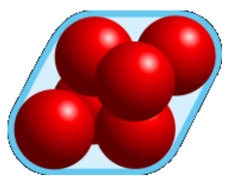
- For dimension d=4: cluster becomes optimal for n>375370?

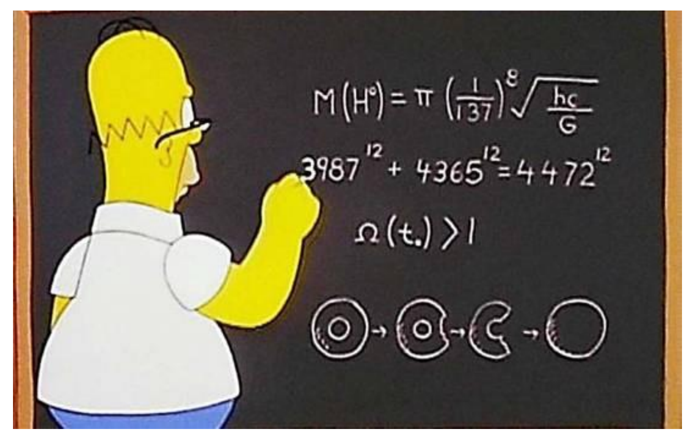




More Sausage Catastrophes

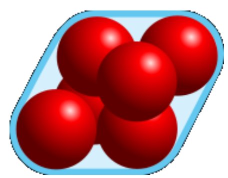
- For dimension d=4: cluster becomes optimal for n>375370?
- For d<11 the optimum packing is always a cluster or a sausage, but never a pizza!

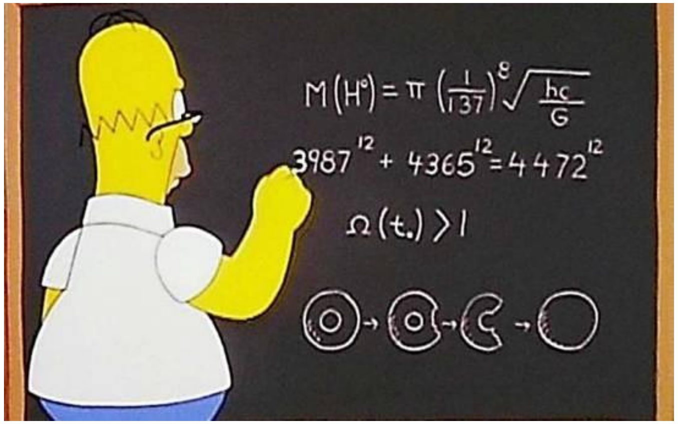




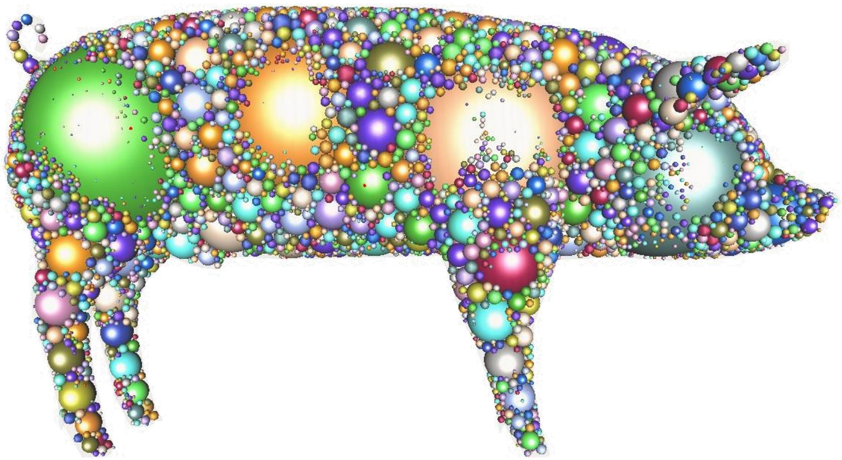
More Sausage Catastrophes

- For dimension d=4: cluster becomes optimal for n>375370?
- For d<11 the optimum packing is always a cluster or a sausage, but never a pizza!
  - Note, this is not true for other convex shapes
  - E.g. it is possible to find for each d>2 a convex shape where the pizza is the closest packaging



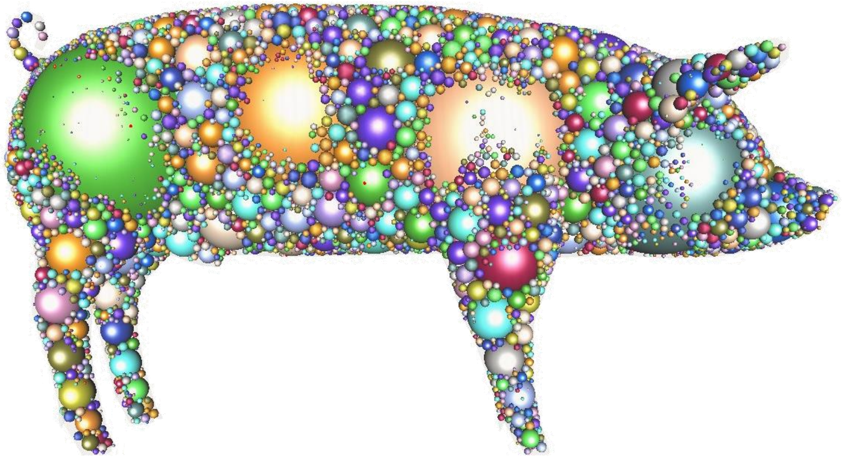


Our Goal



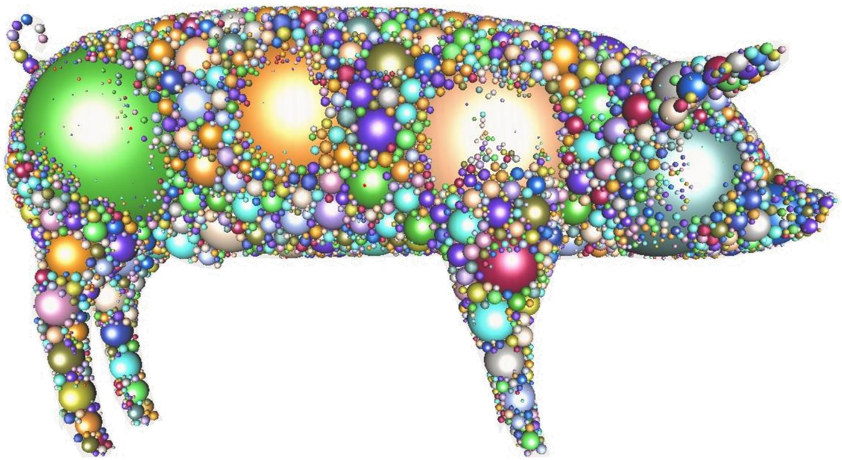
Our Goal

- Feasible Sphere Packing
  - All spheres are inside
  - Spheres do not overlap



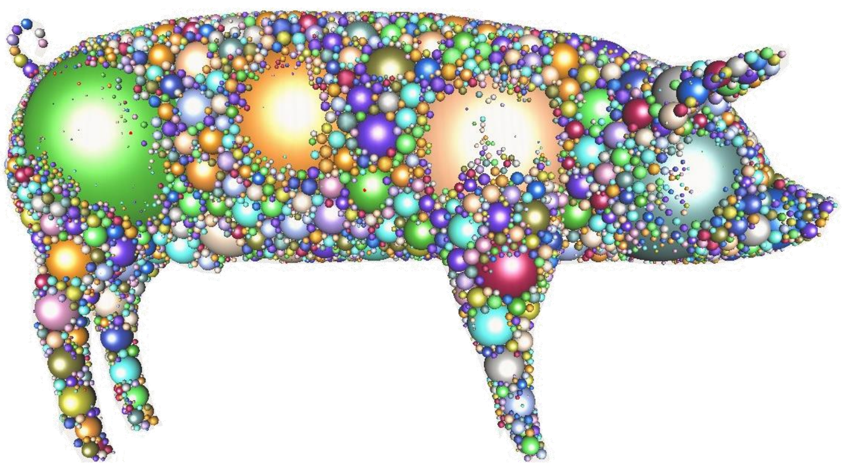
Our Goal

- Feasible Sphere Packing
  - All spheres are inside
  - Spheres do not overlap
- Polydisperse packing
  - Space filling

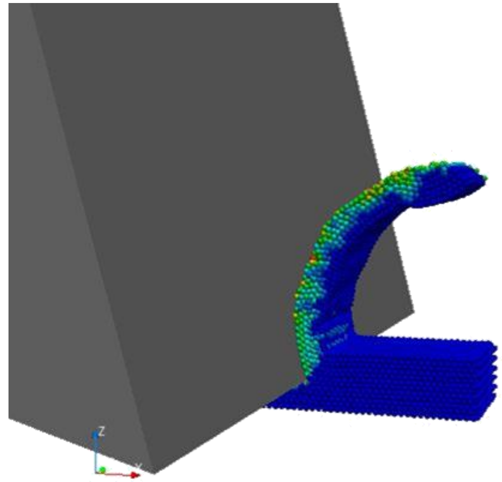


Our Goal

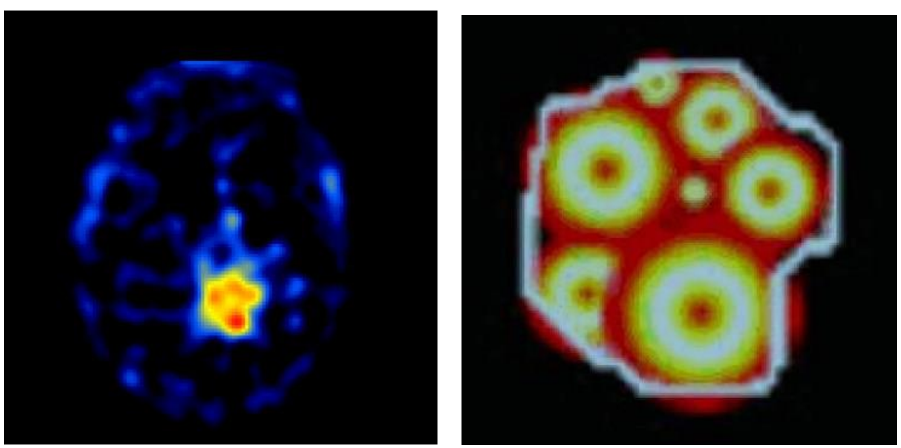
- Feasible Sphere Packing
  - All spheres are inside
  - Spheres do not overlap
- Polydisperse packing
  - Space filling
- Arbitrary objects
  - Arbitrary object representations (Polygonal, NURBS, CSG, point clouds,…)



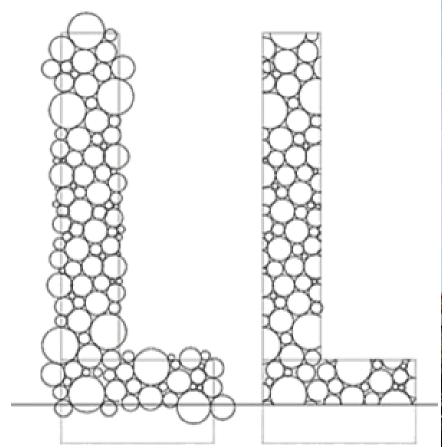
Applications of Polydisperse Sphere Packings



P. Eberhard et al, 2009

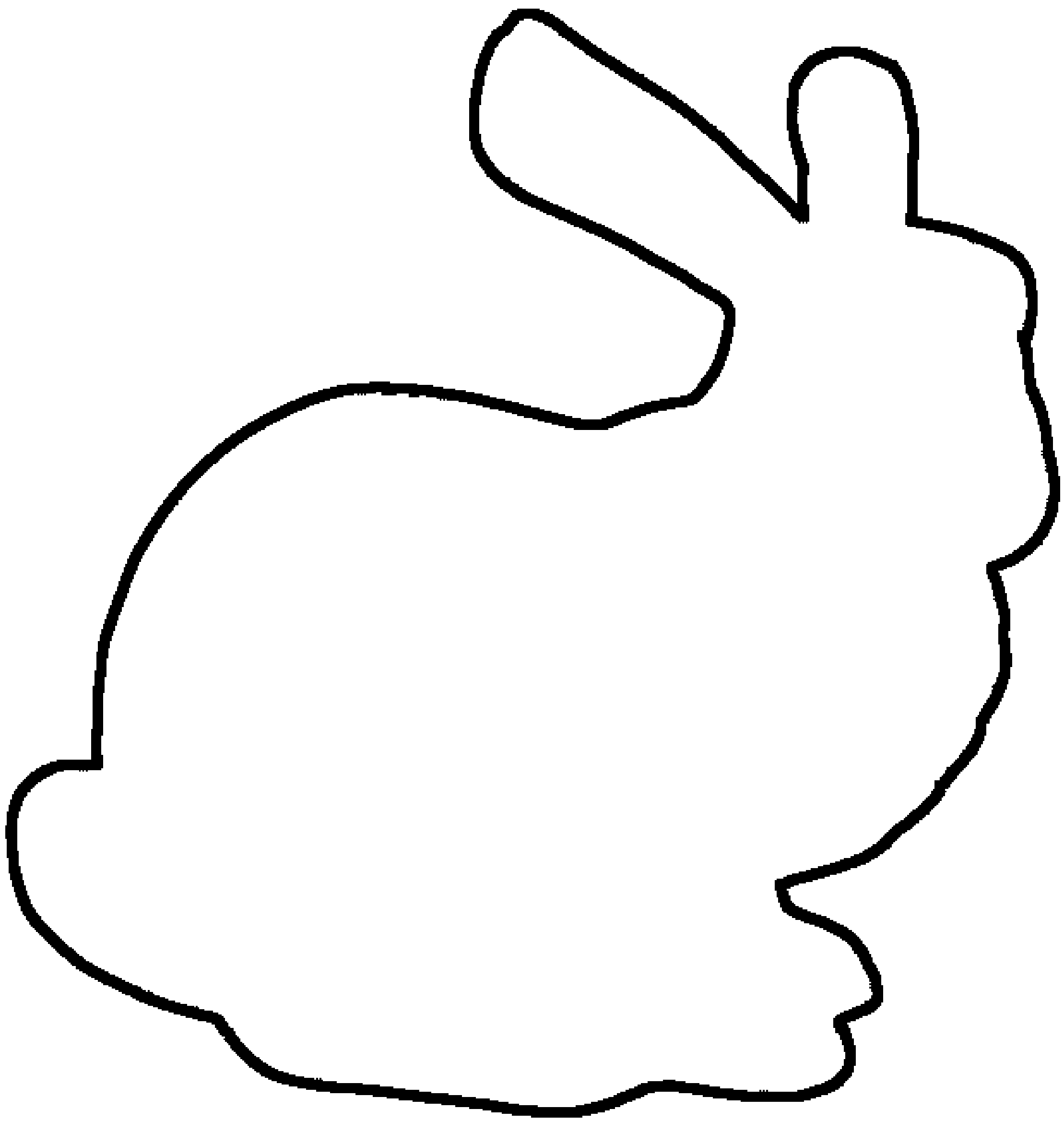


Long Yun et al, 2002

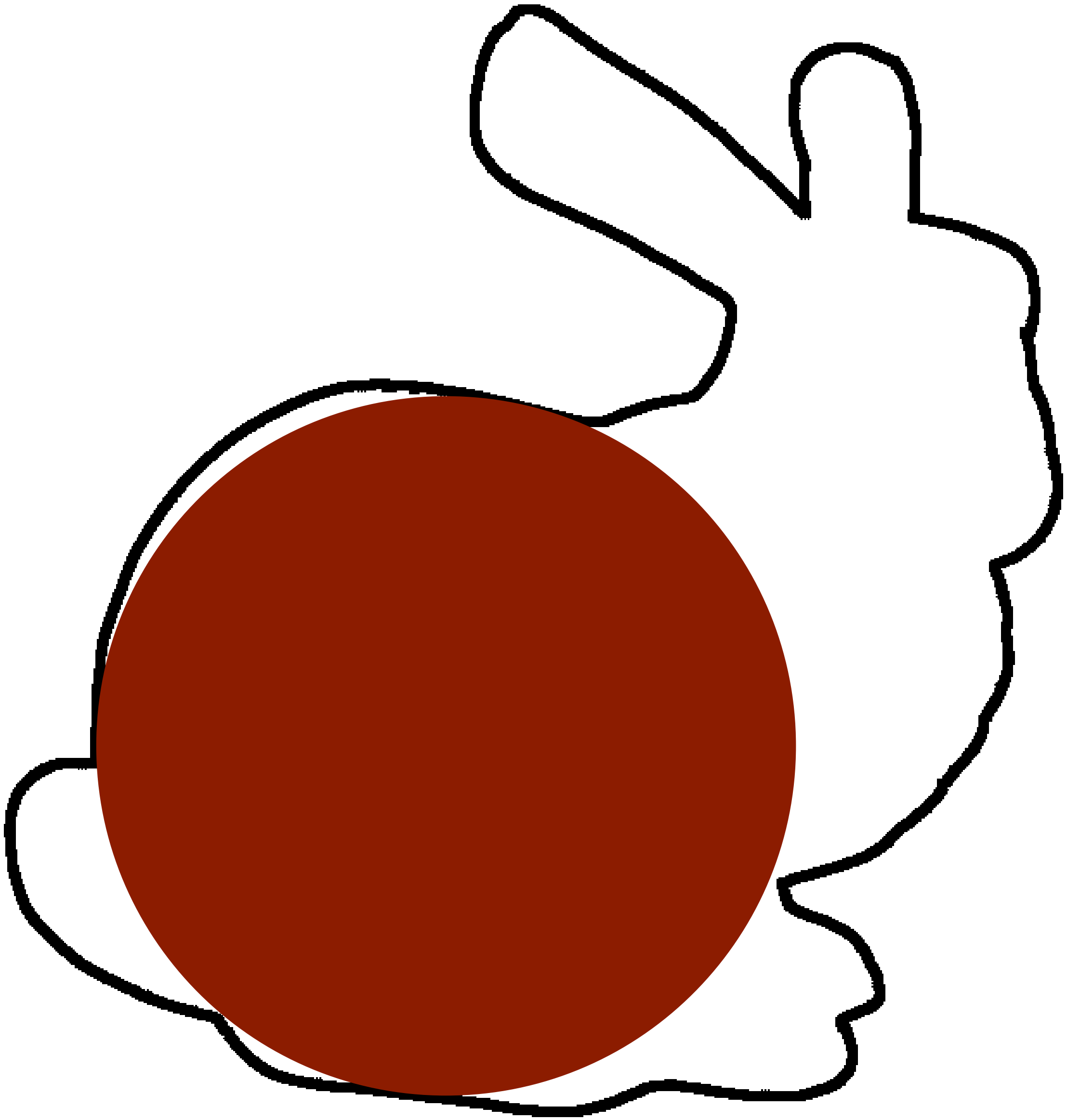


smo architektur, 2006

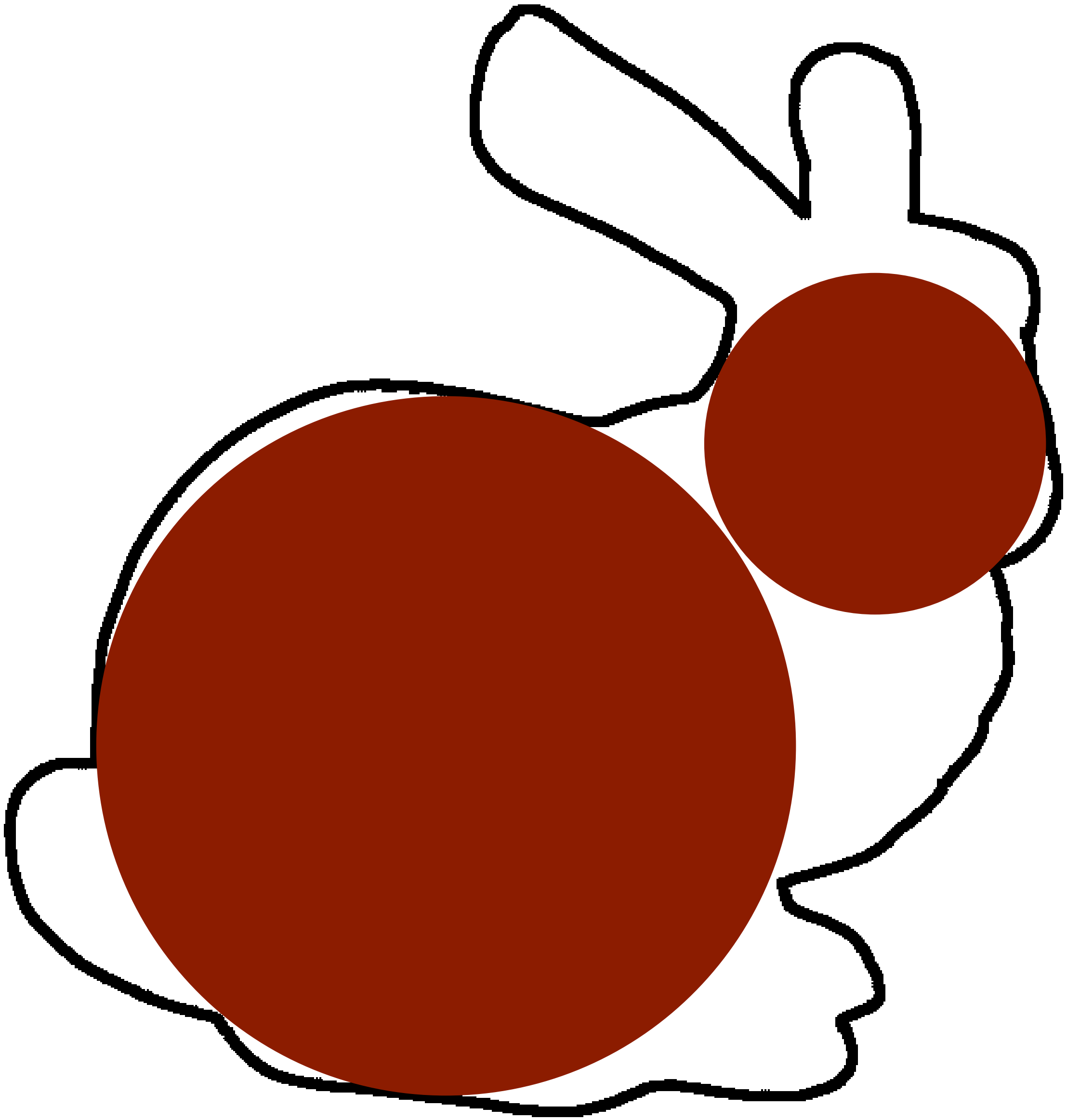
Basic Idea



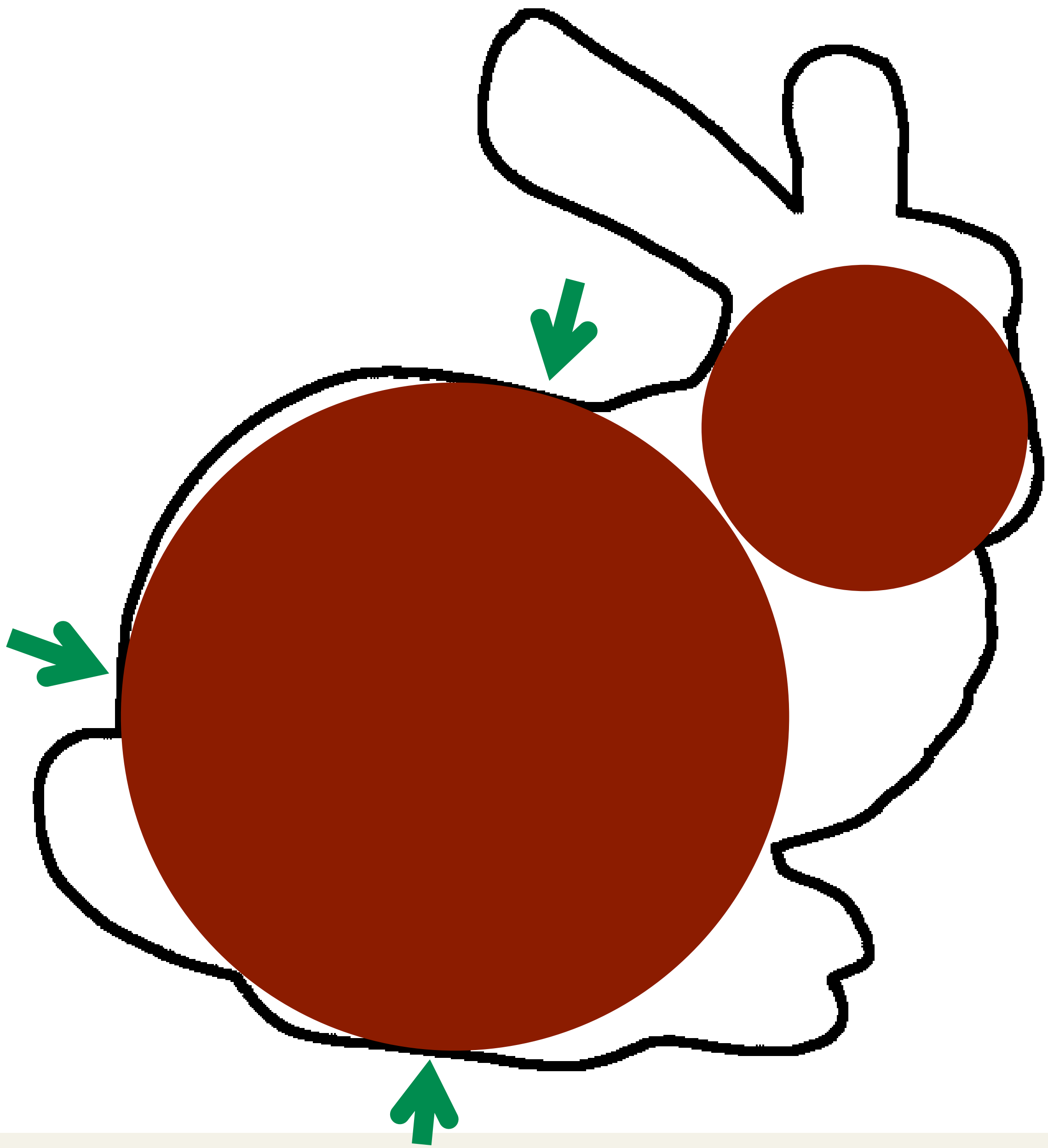
Basic Idea



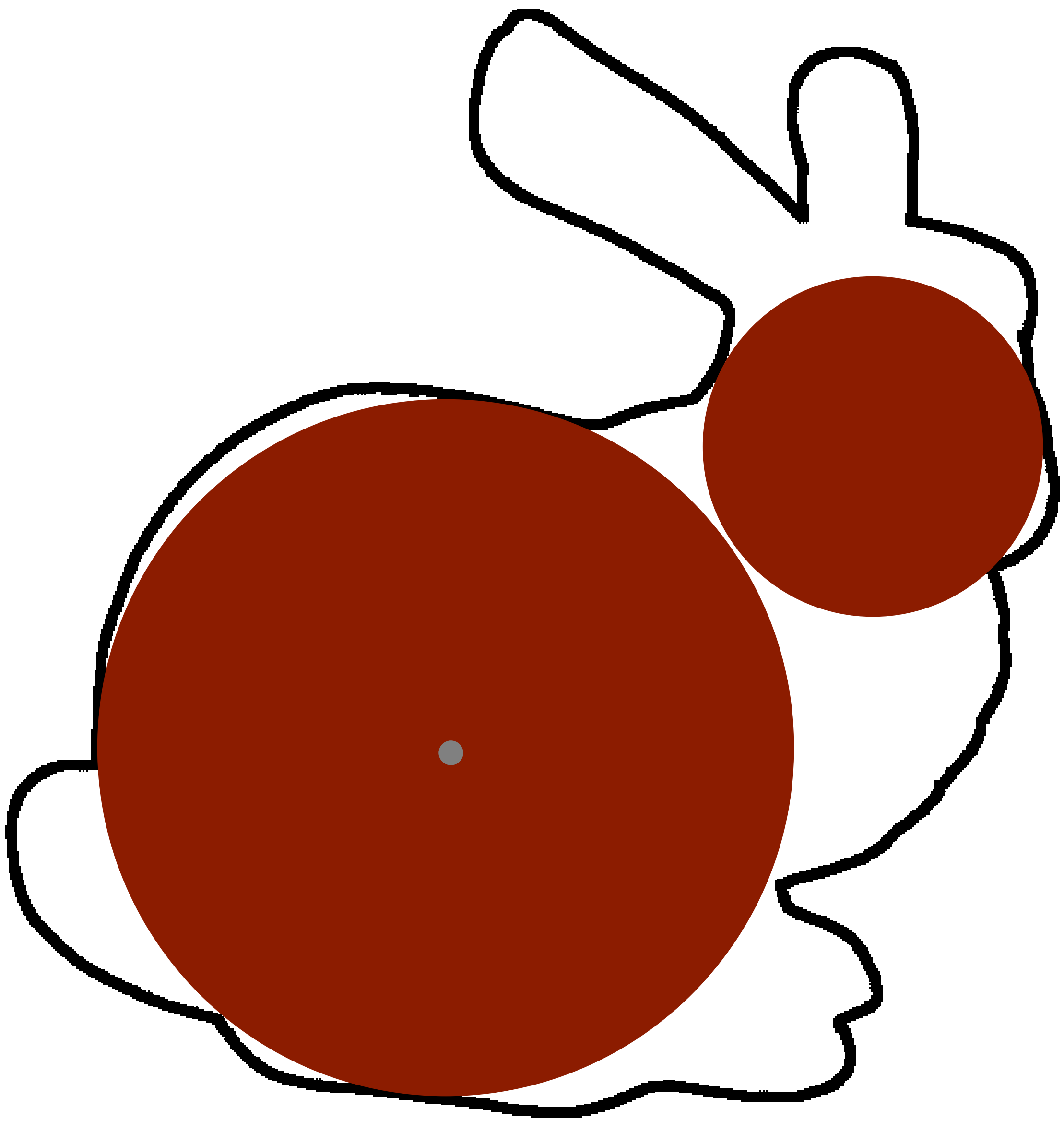
Basic Idea



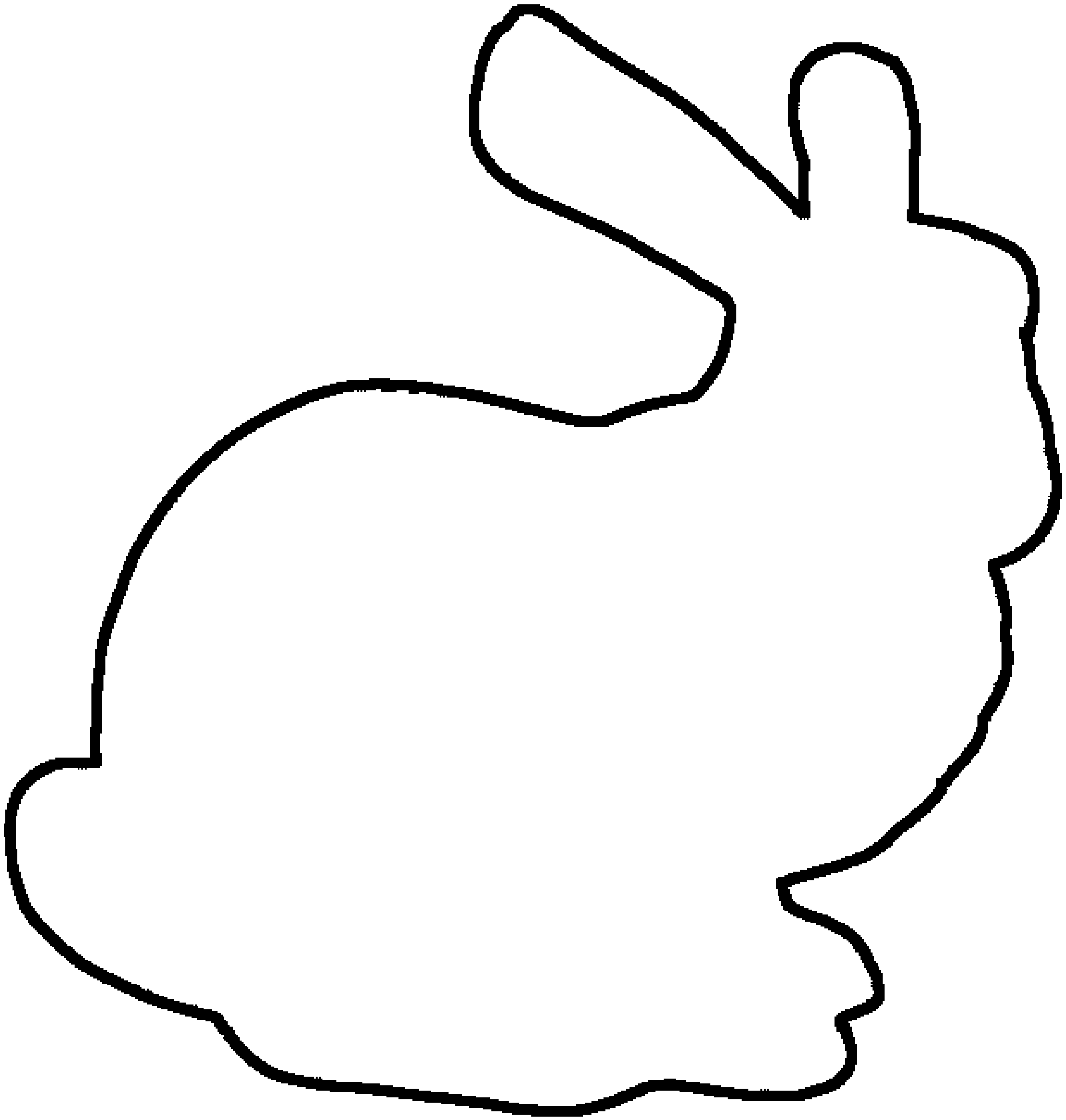
Basic Idea



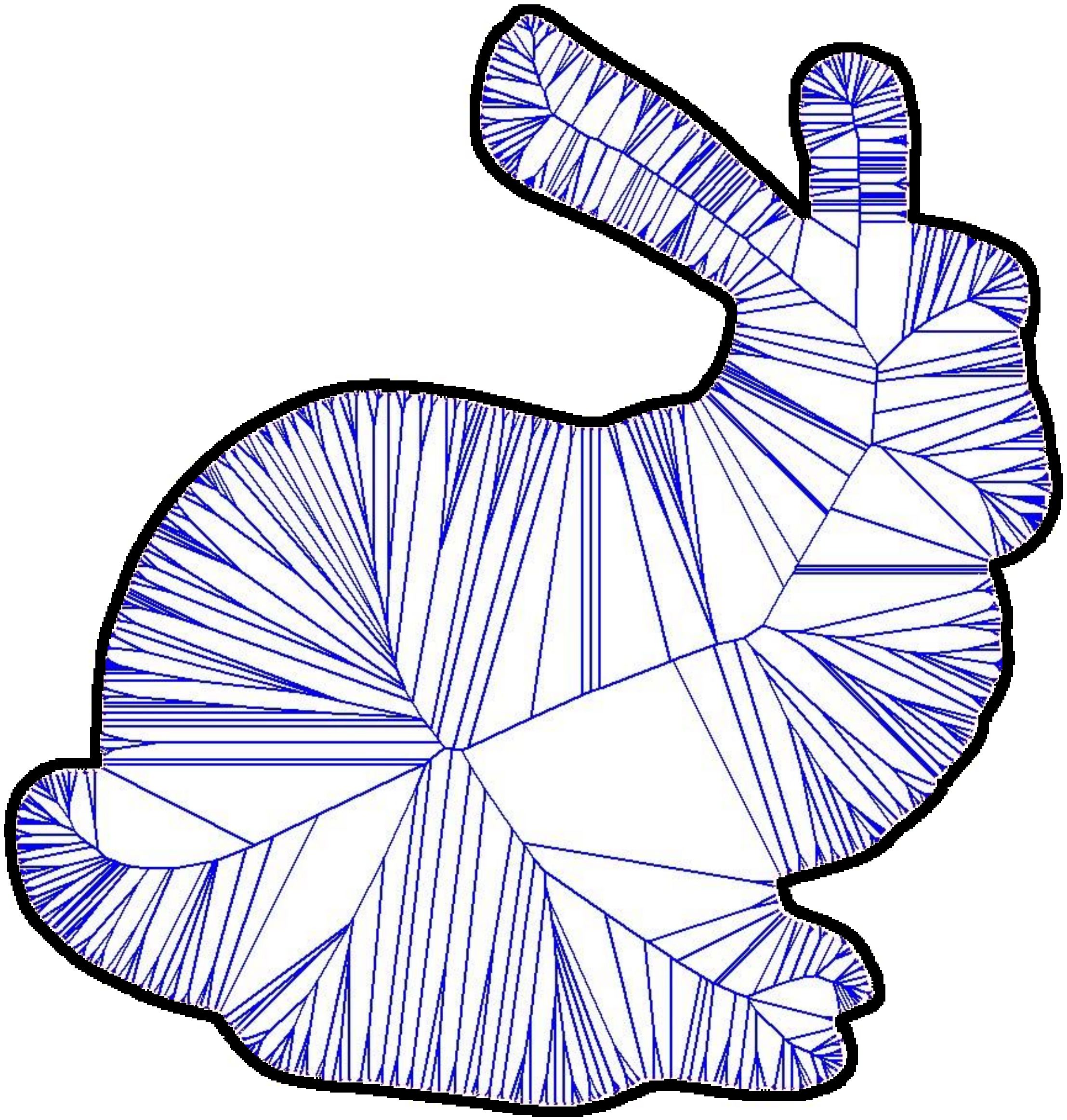
Basic Idea



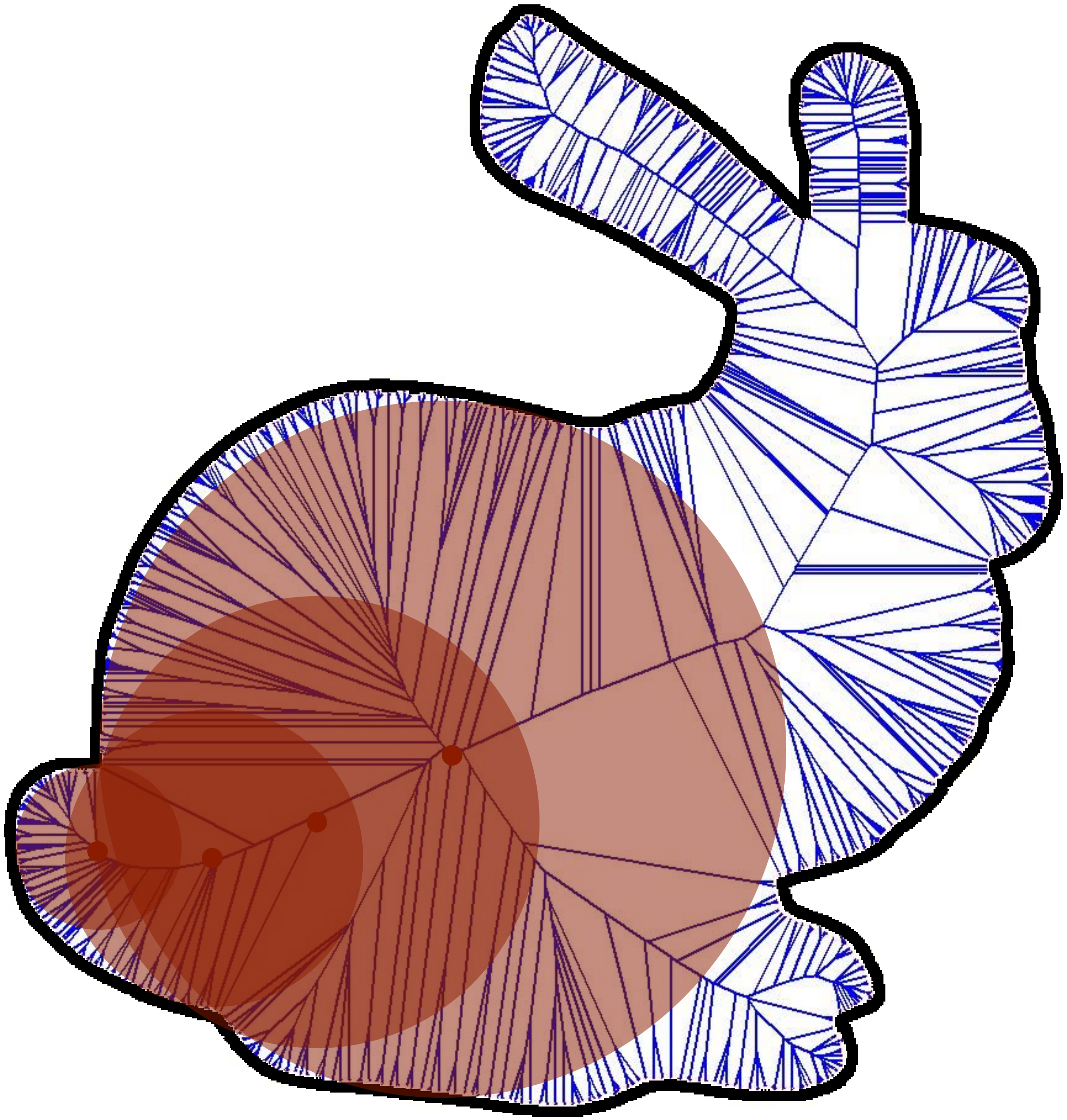
Basic Idea



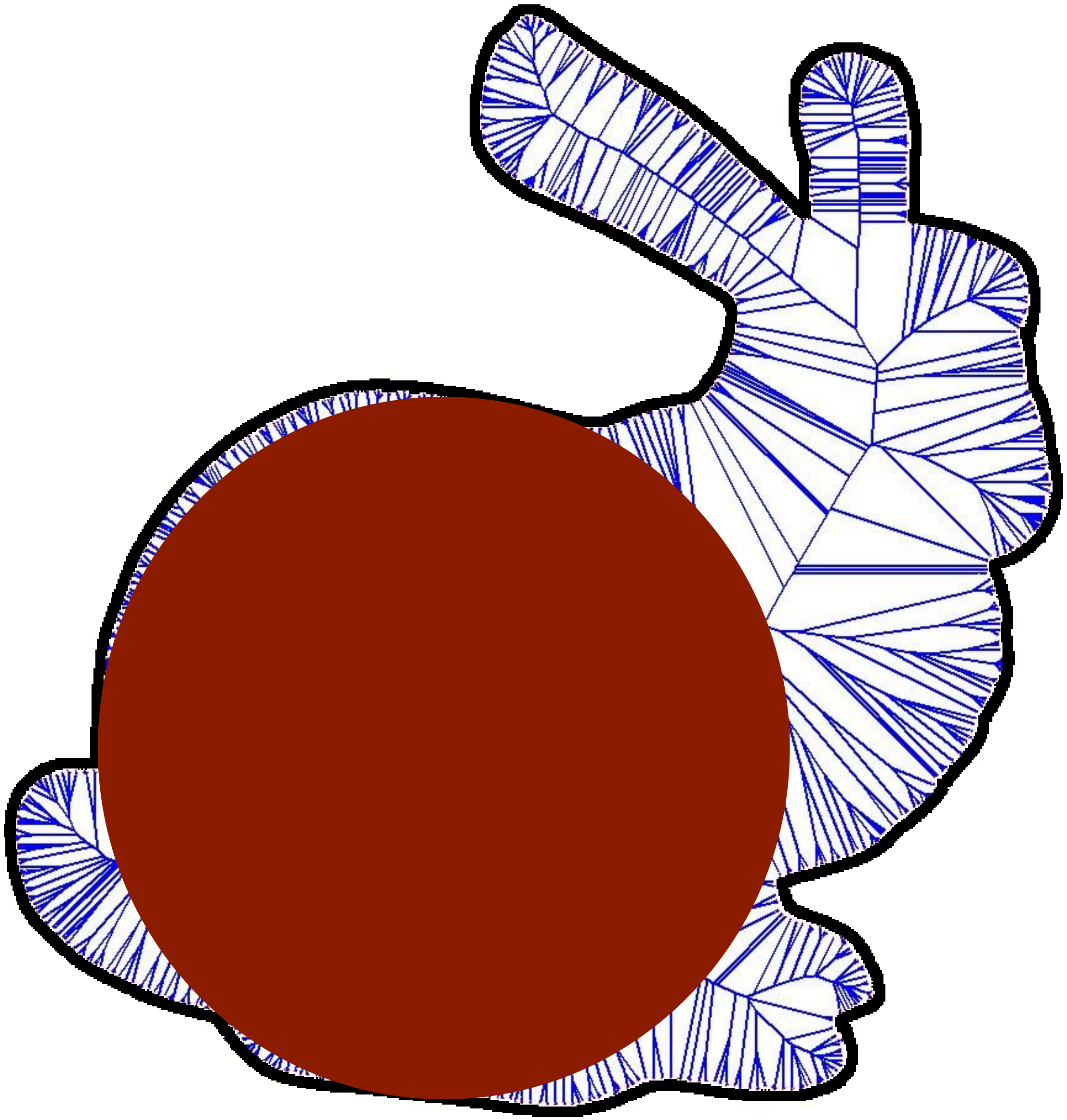
Basic Idea



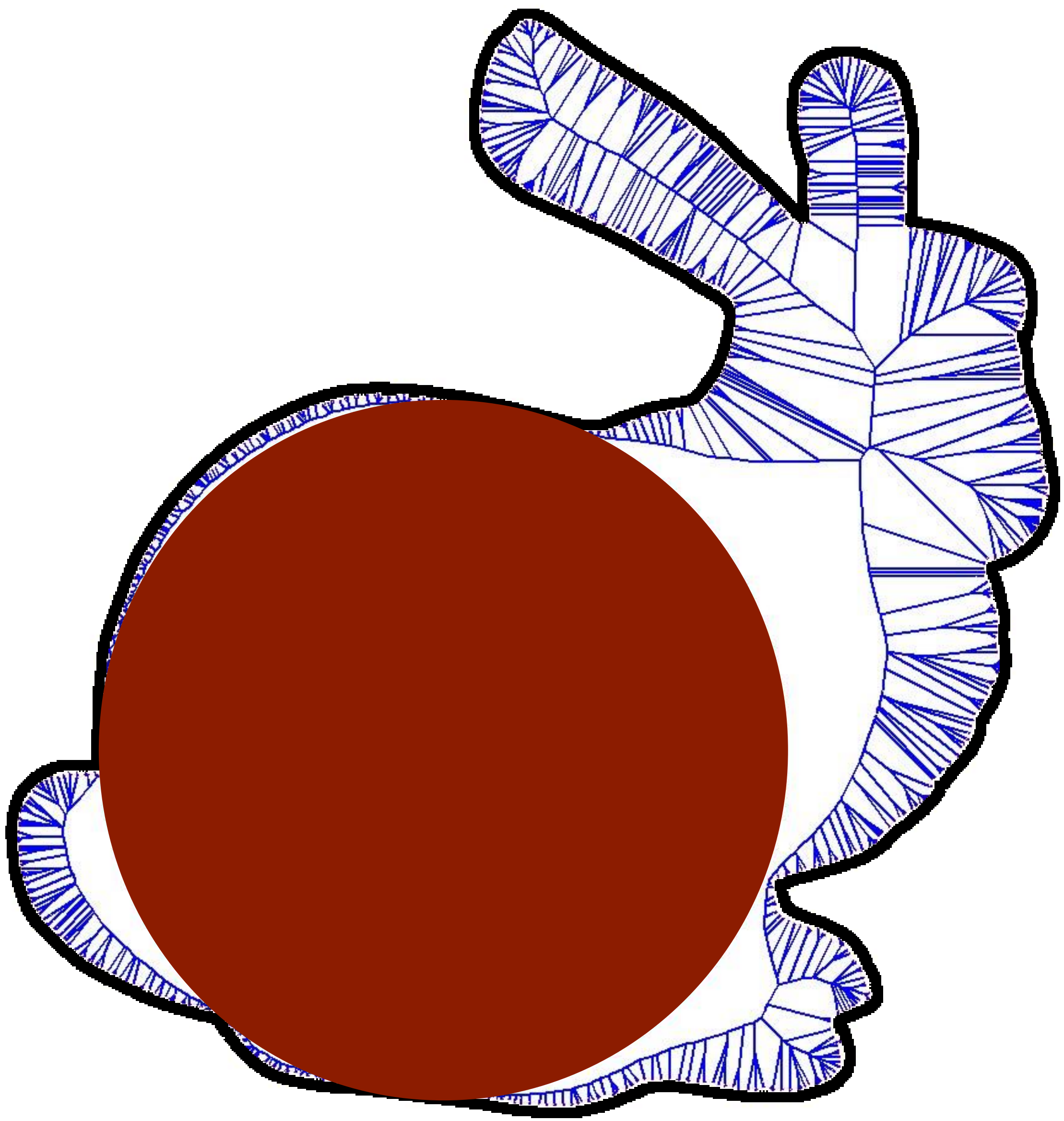
Basic Idea



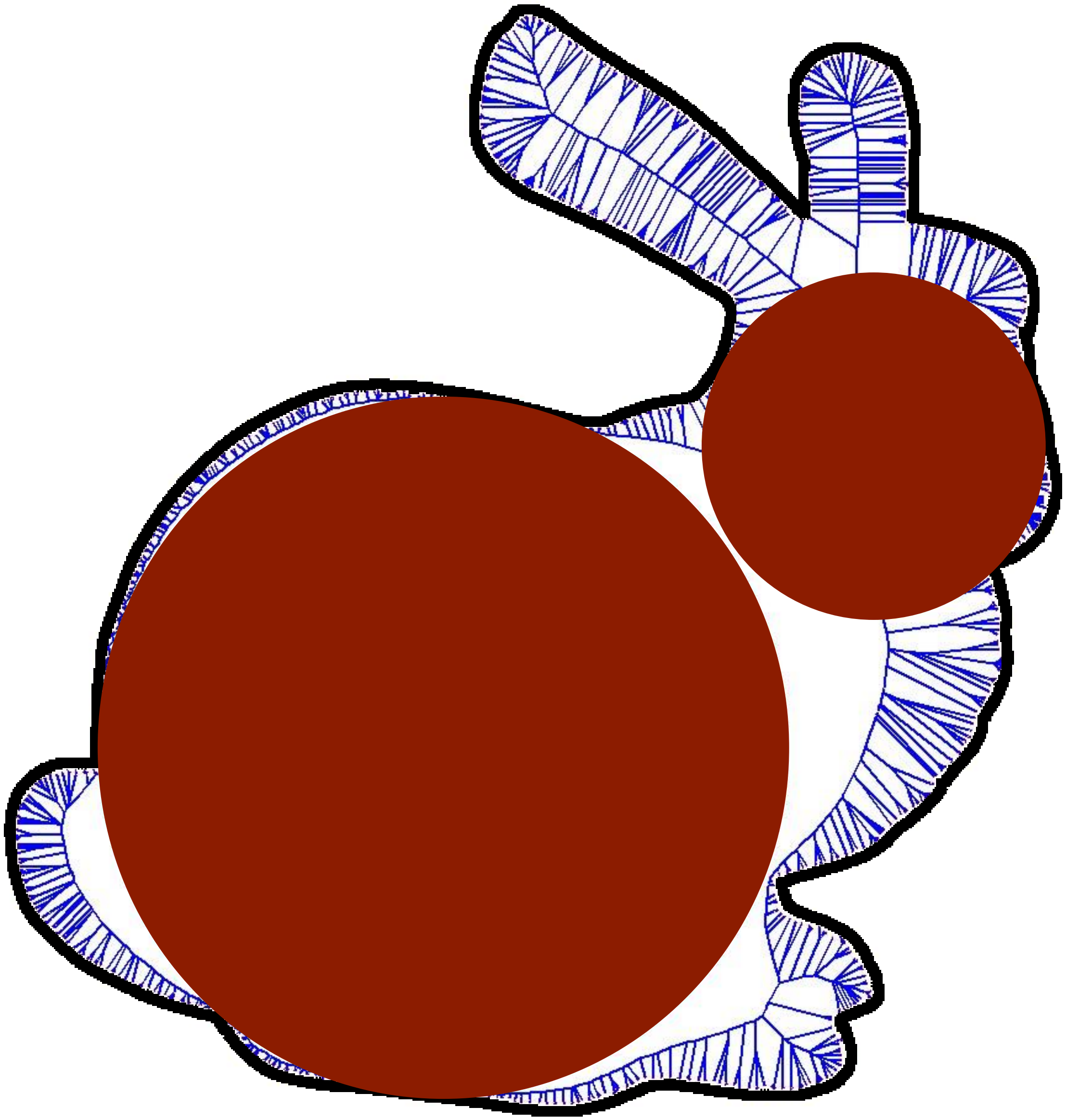
Basic Idea



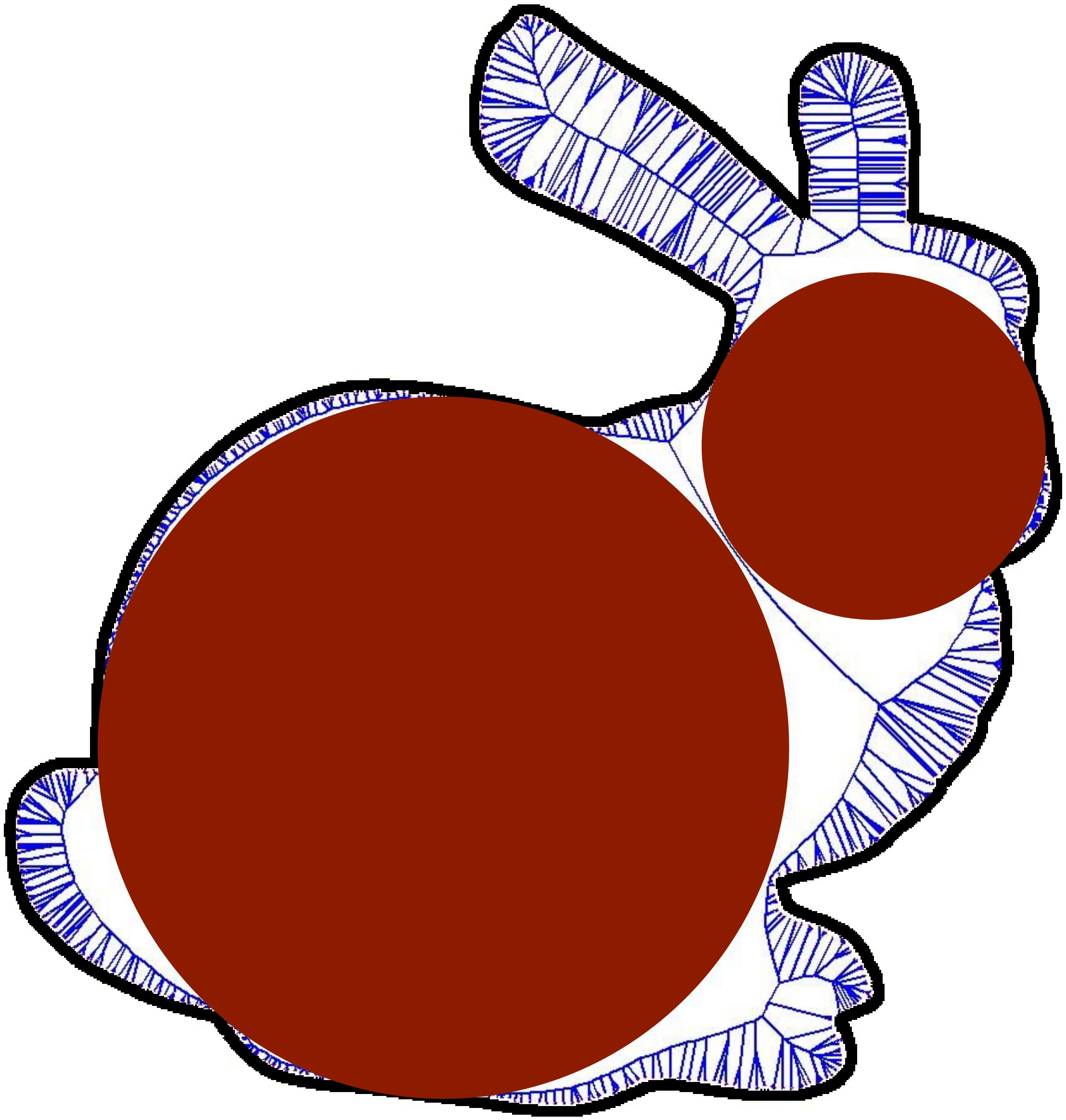
Basic Idea



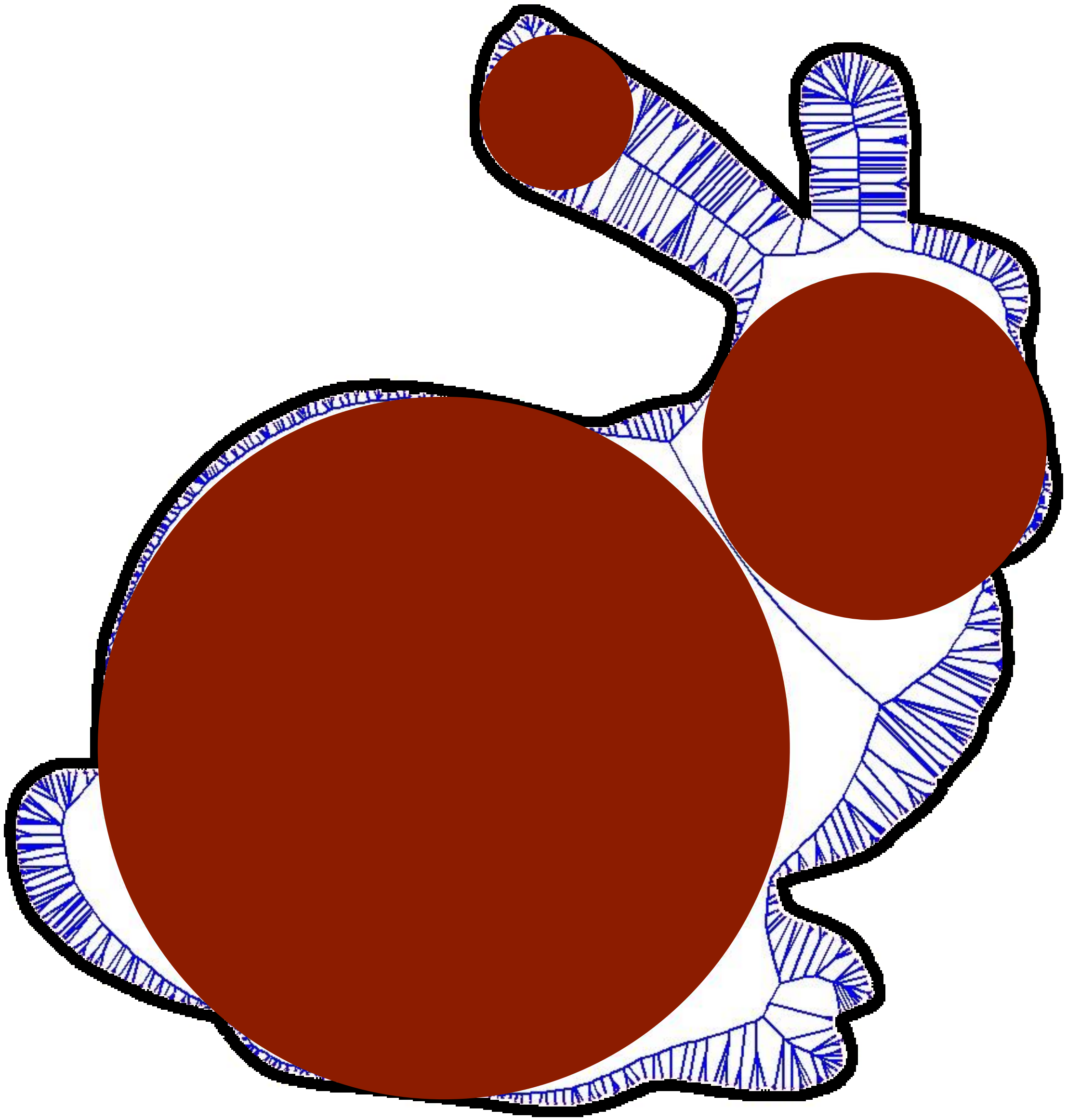
Basic Idea



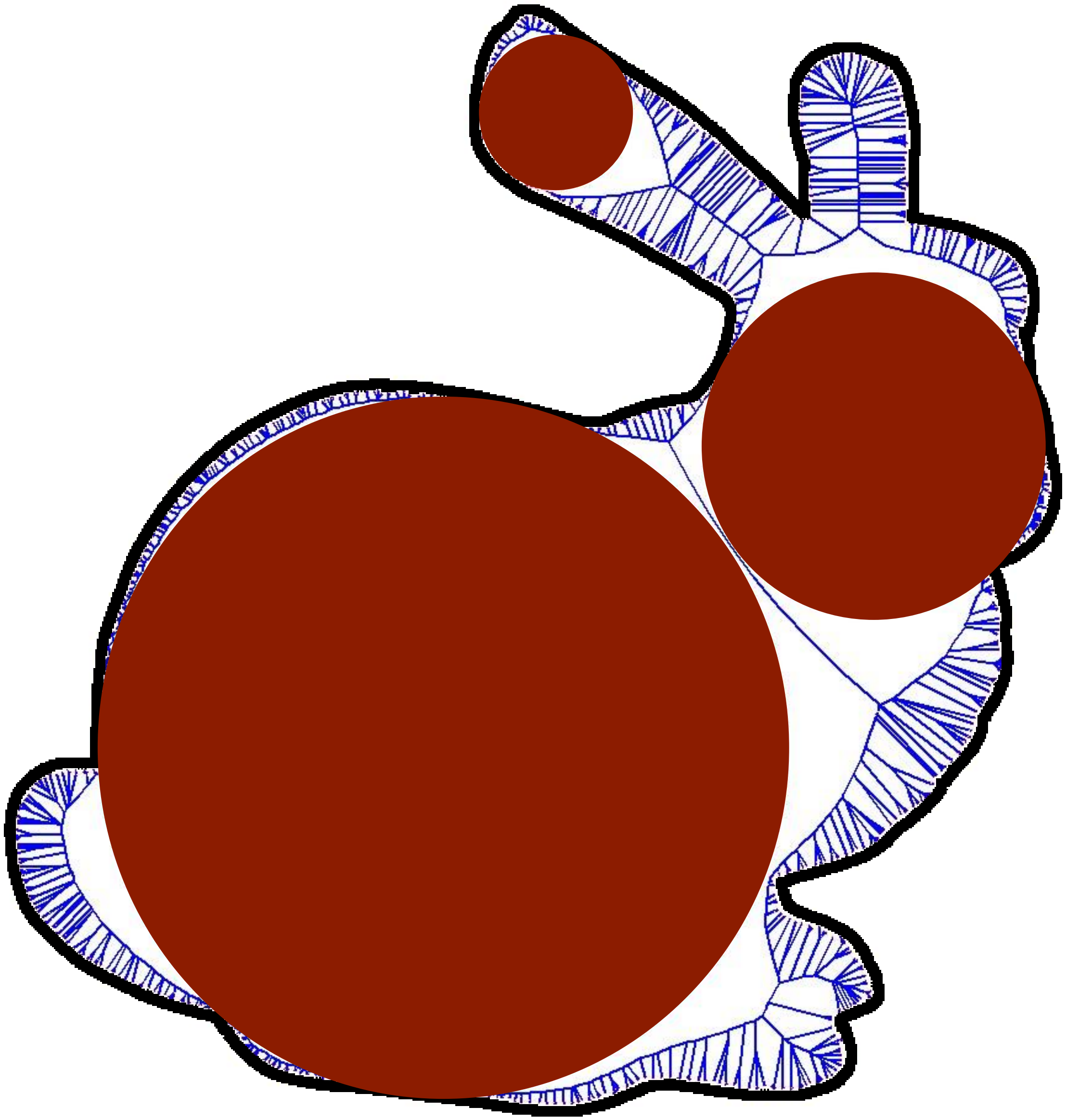
Basic Idea



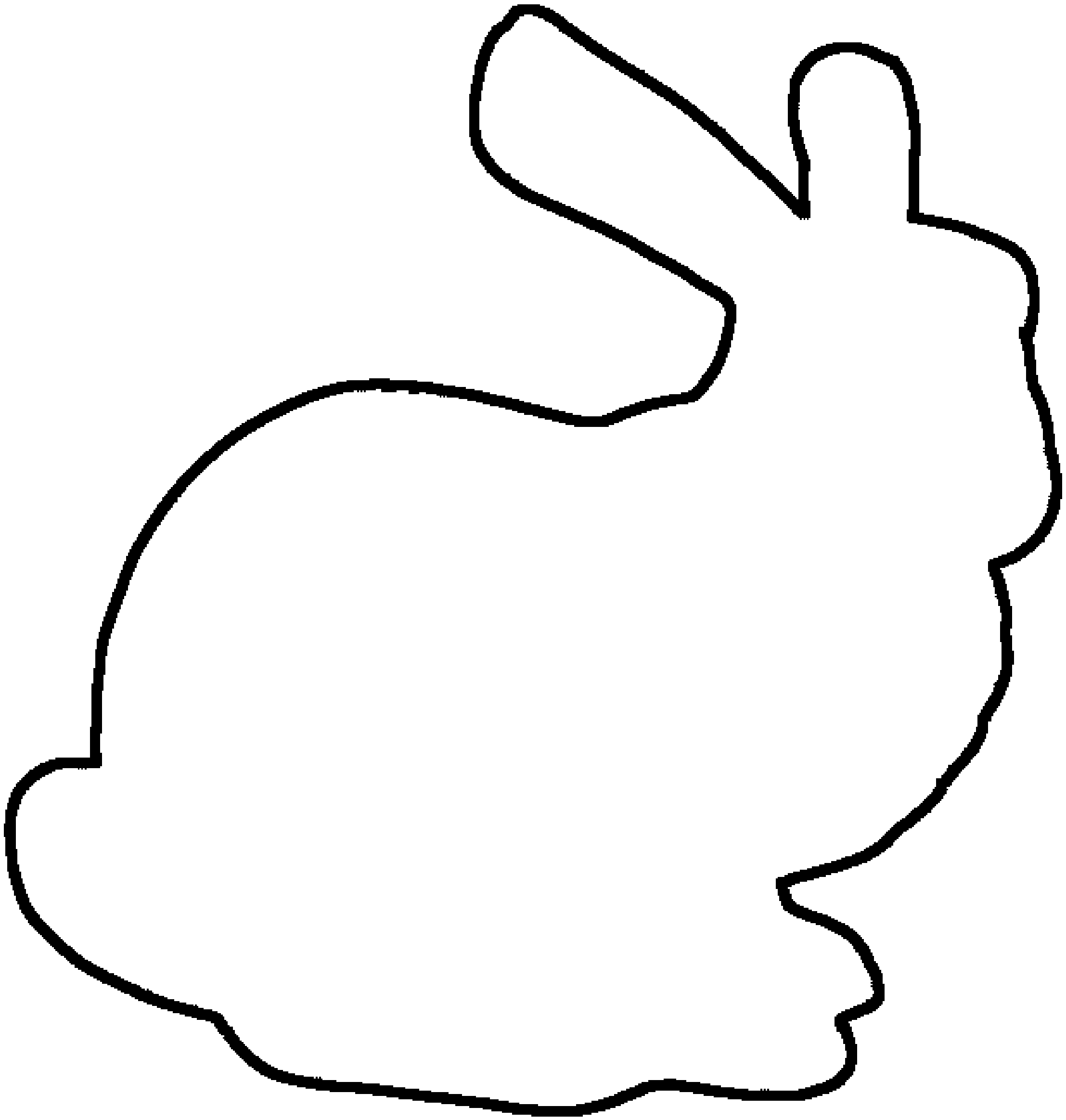
Basic Idea



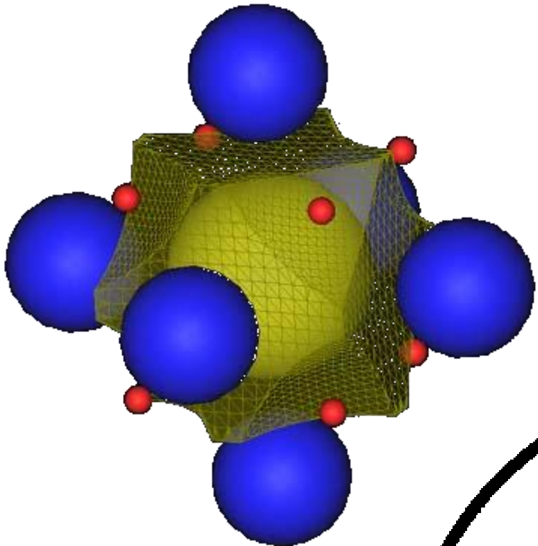
Basic Idea



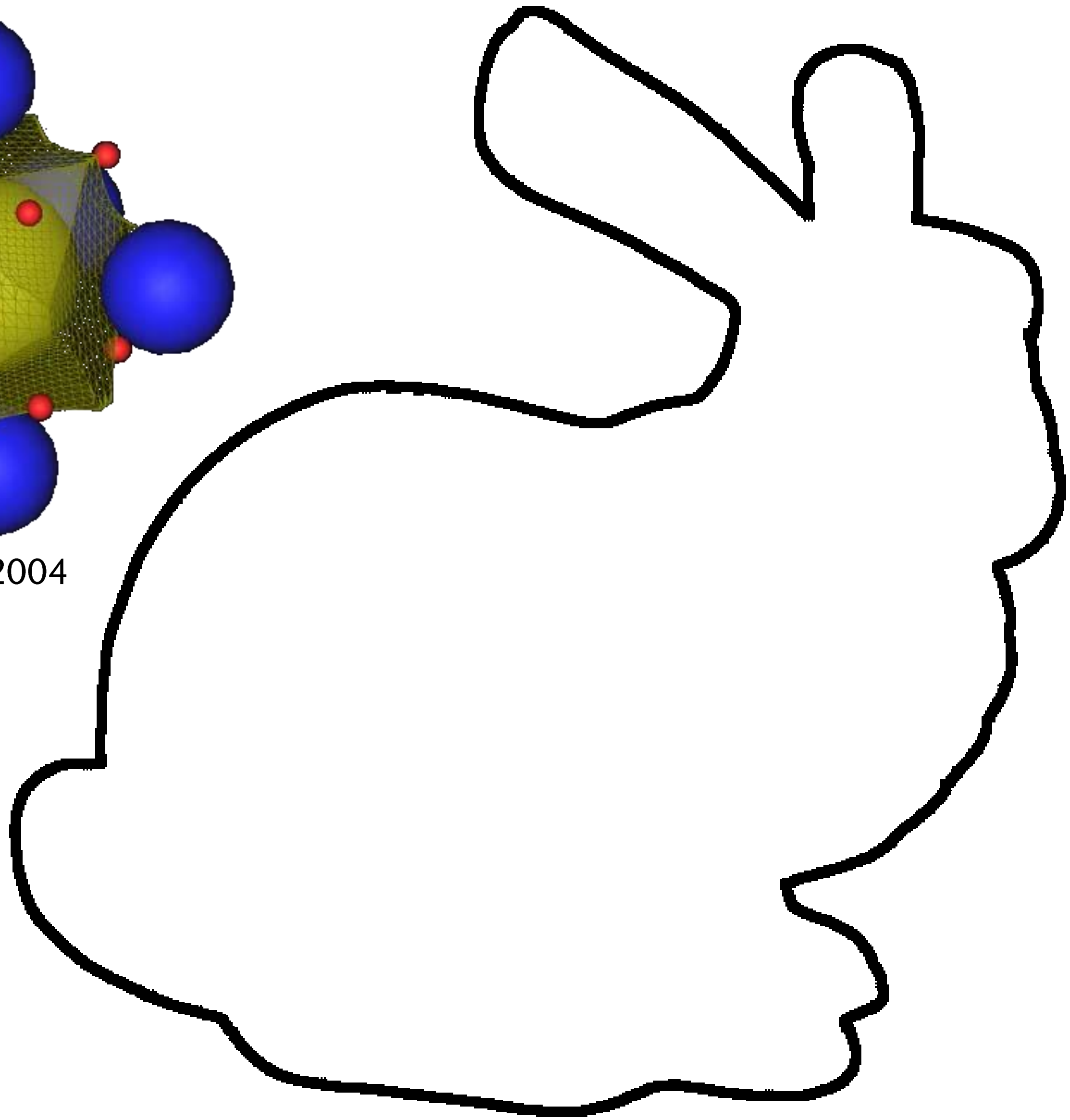
Basic Idea



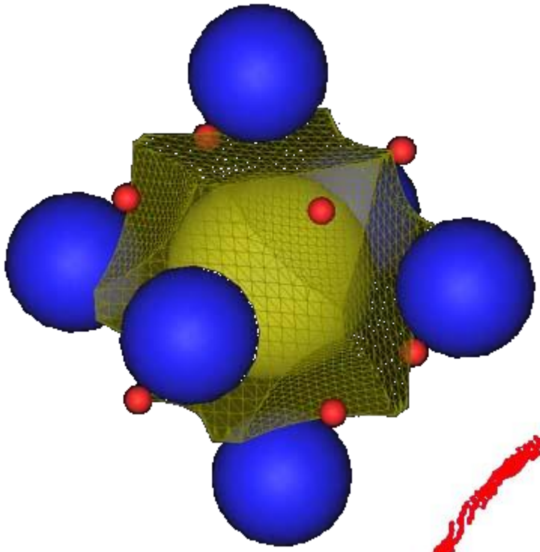
Basic Idea



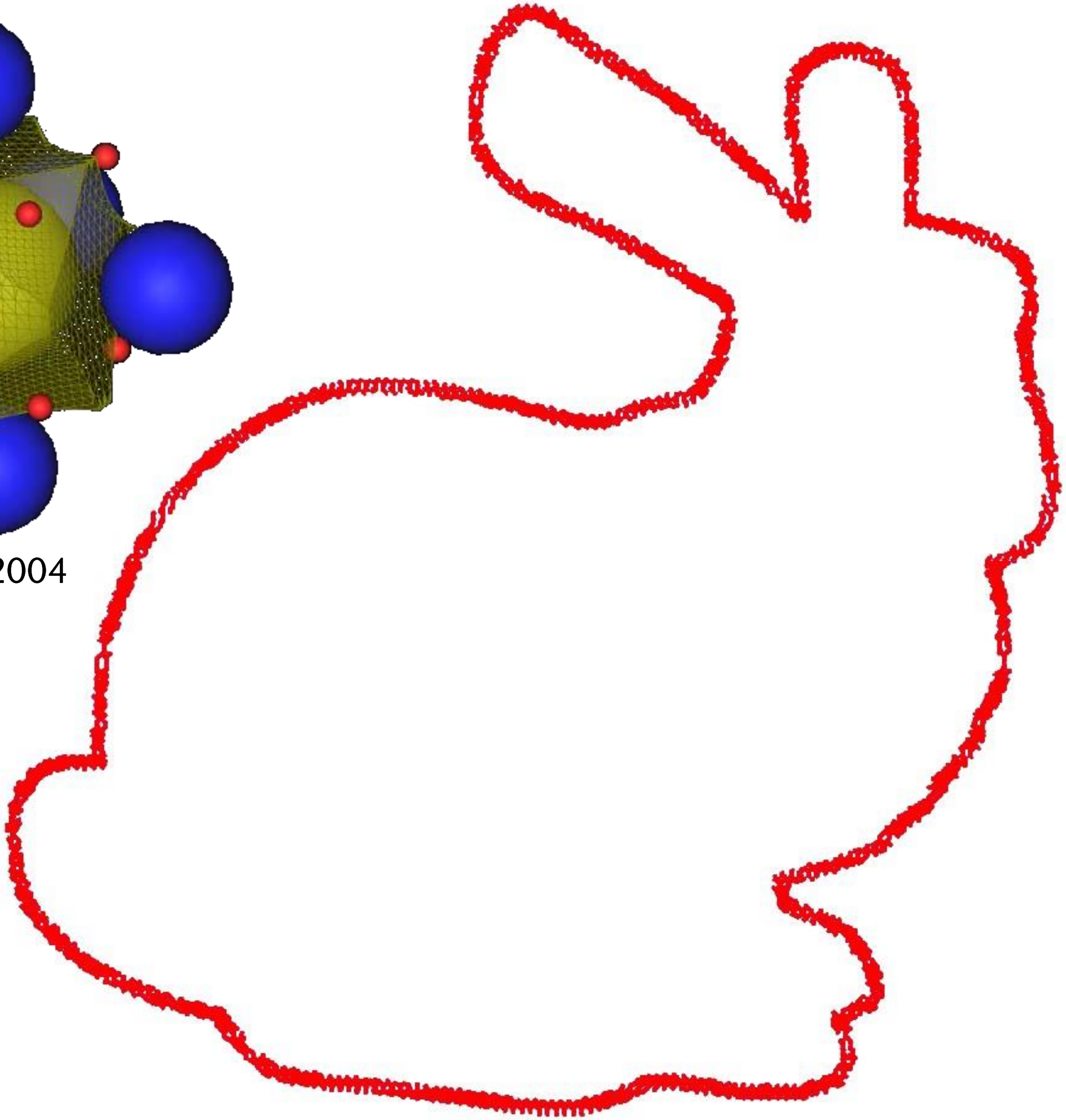
Cho et al., 2004



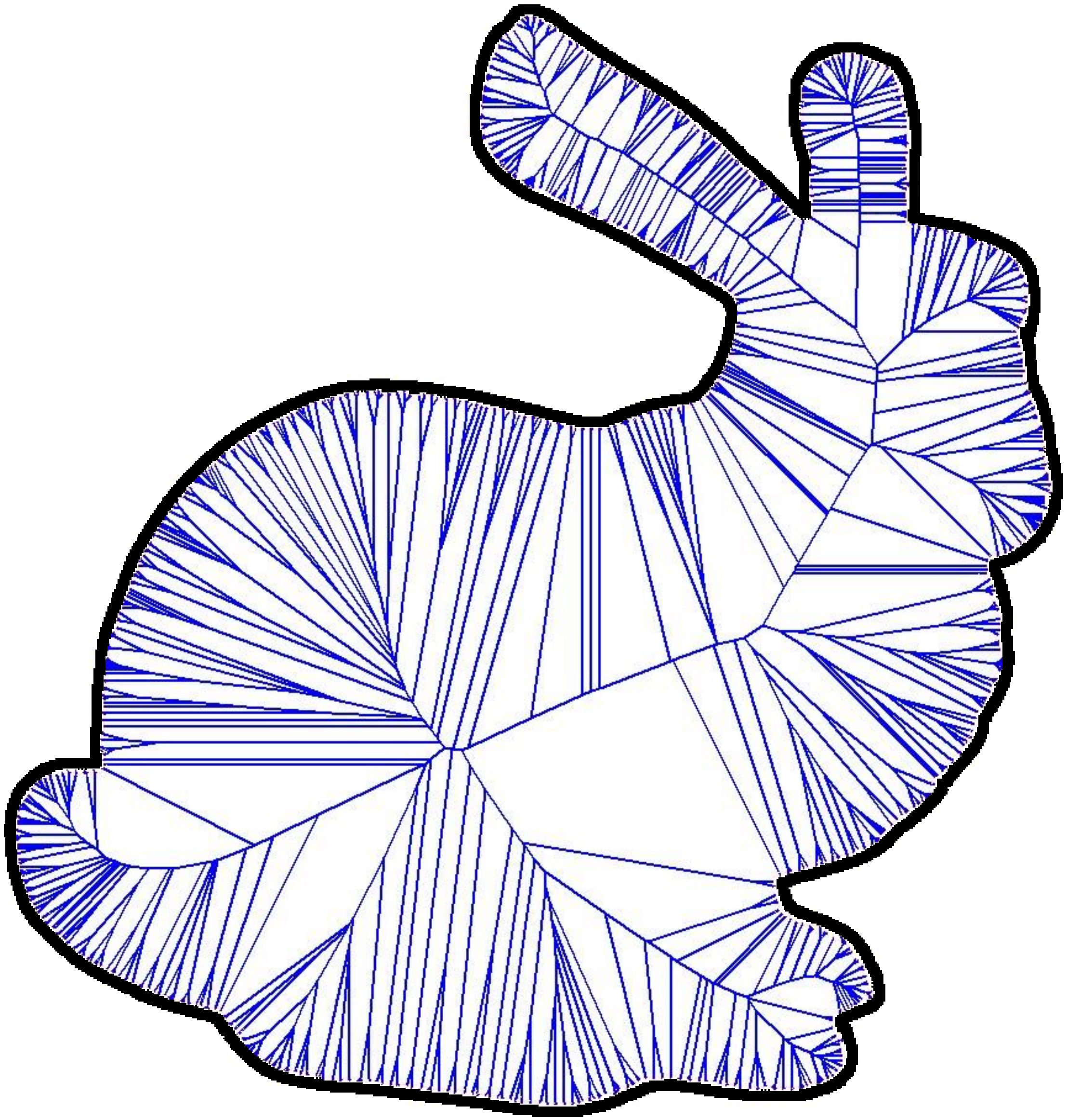
Basic Idea



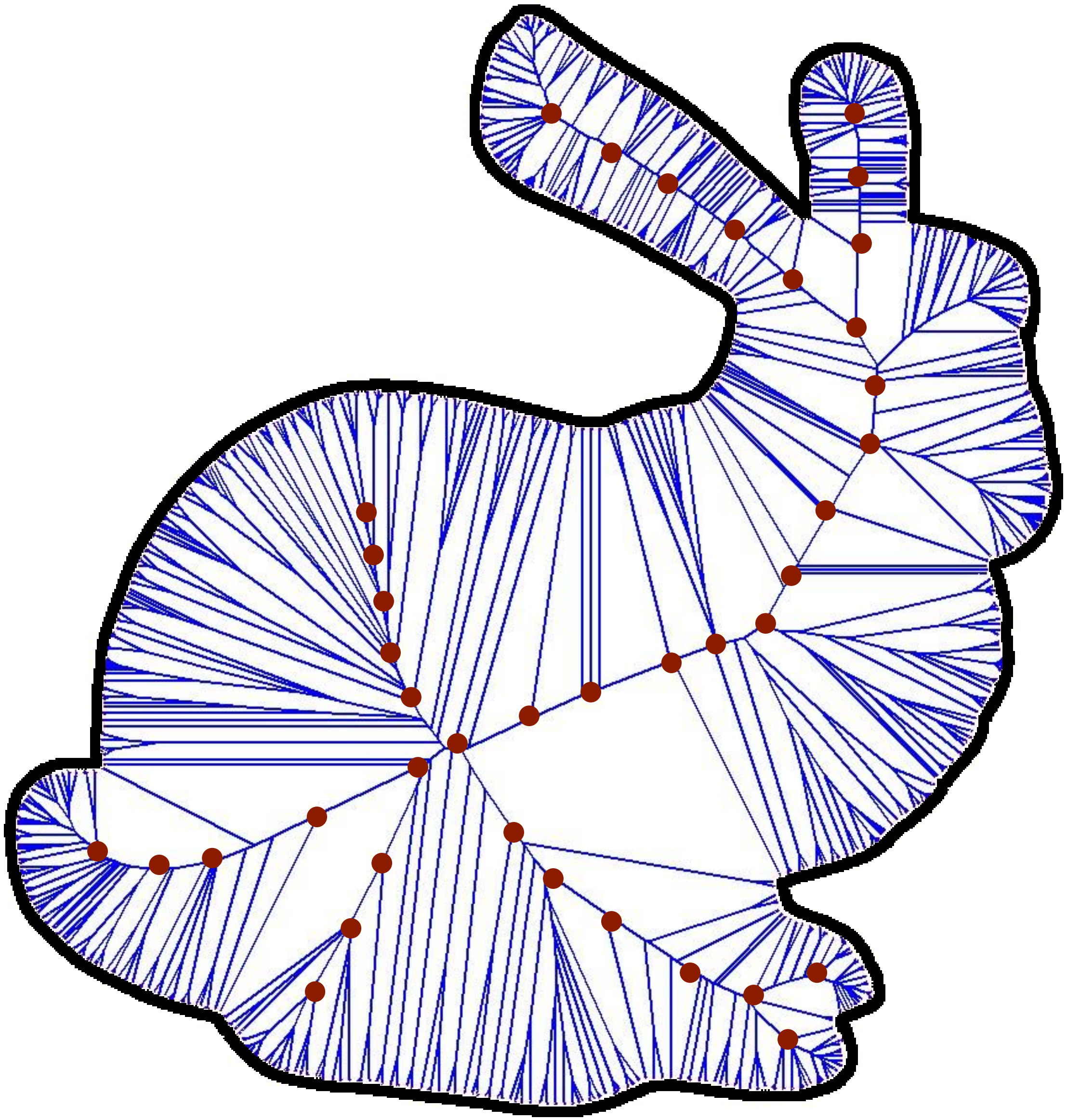
Cho et al., 2004



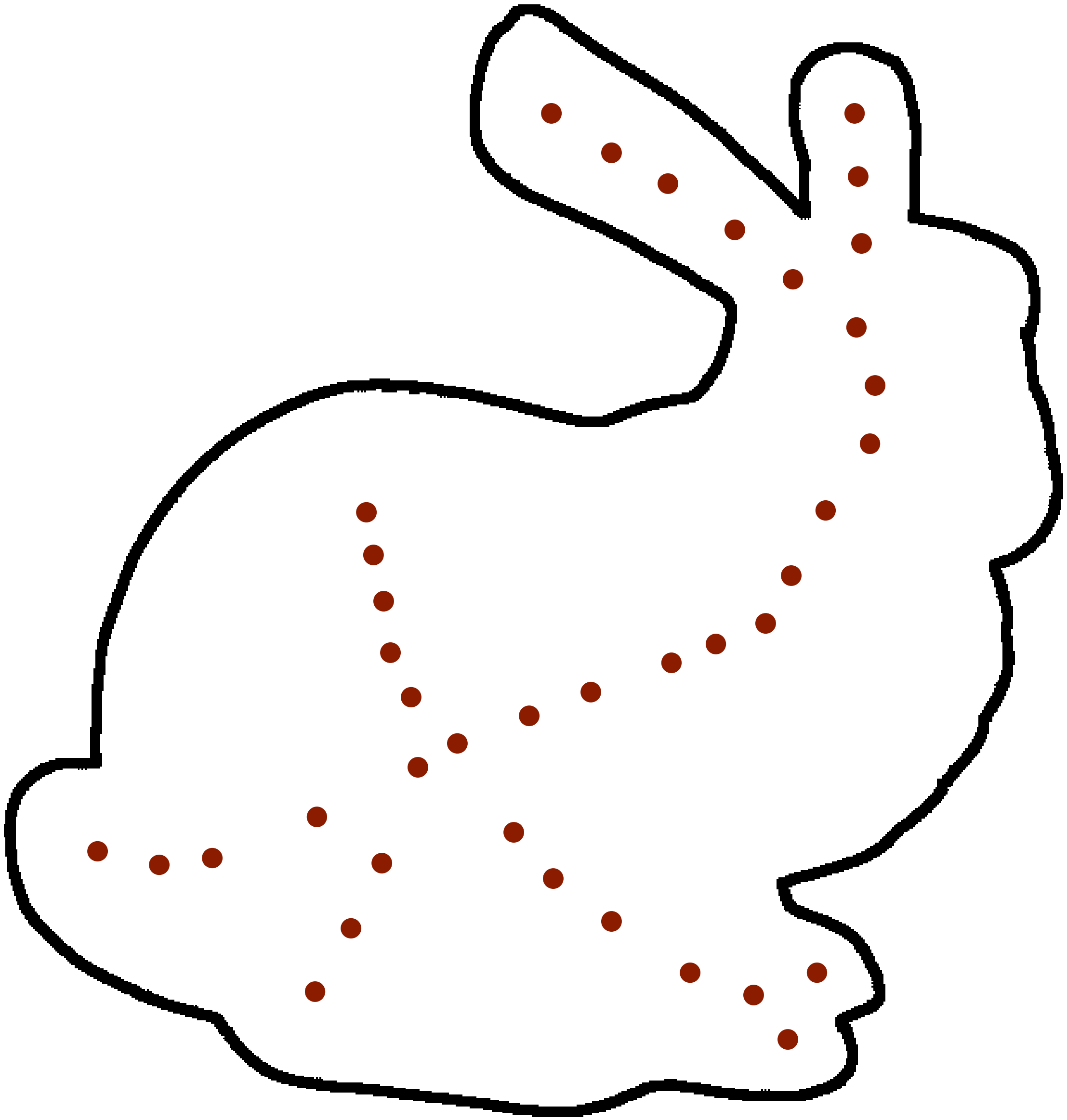
Basic Idea



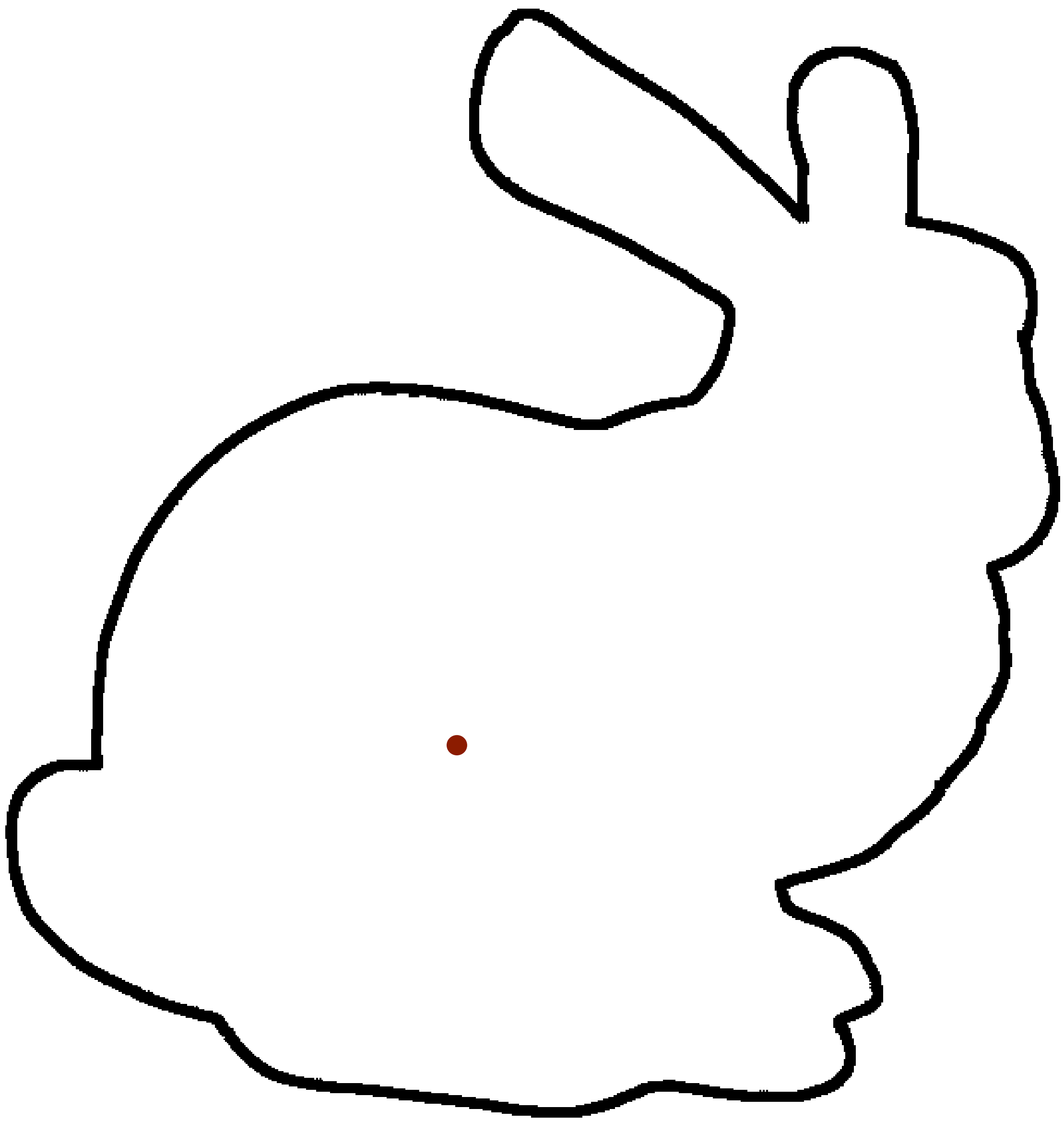
Basic Idea



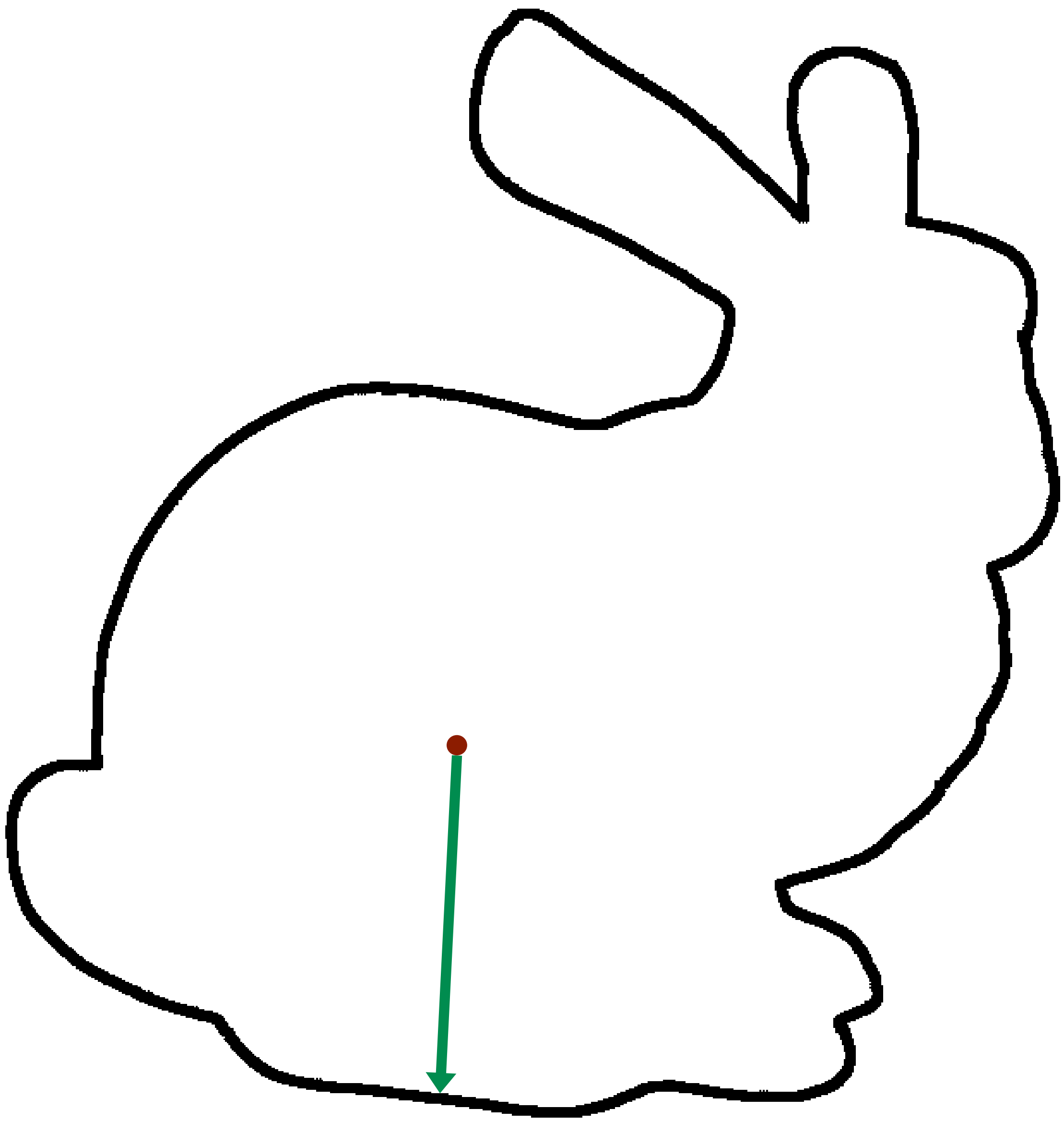
Basic Idea

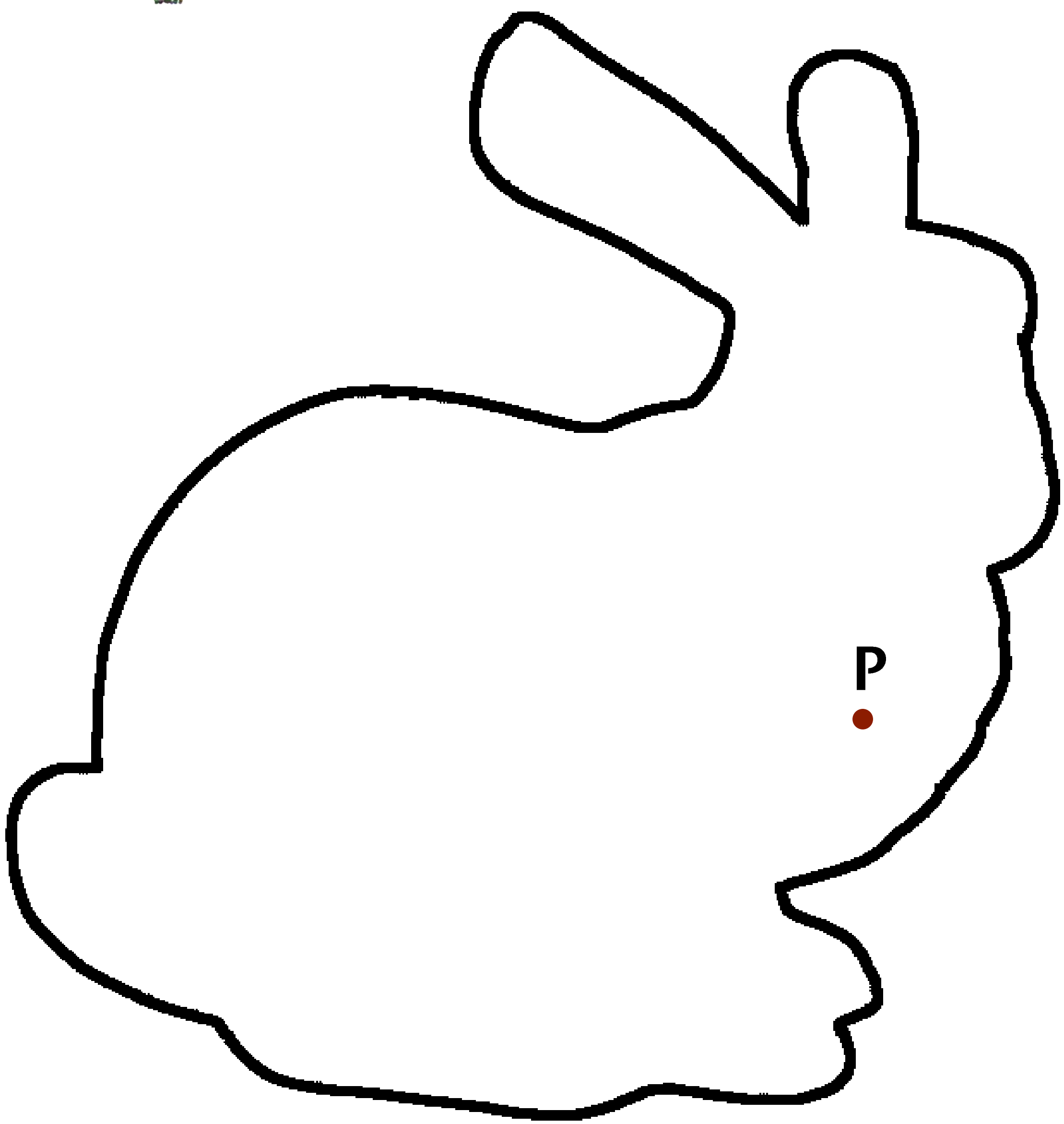


Basic Idea

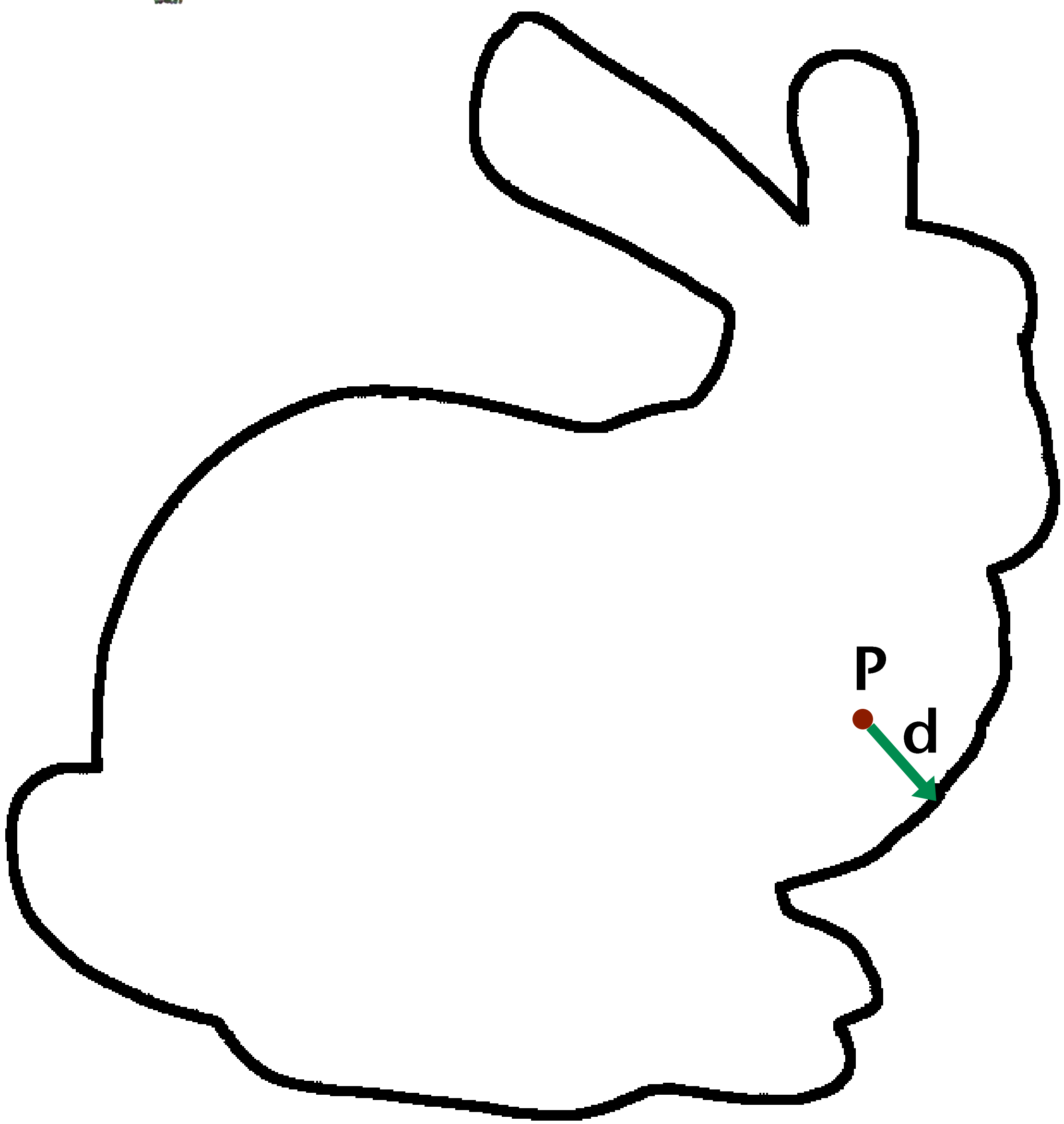


Basic Idea

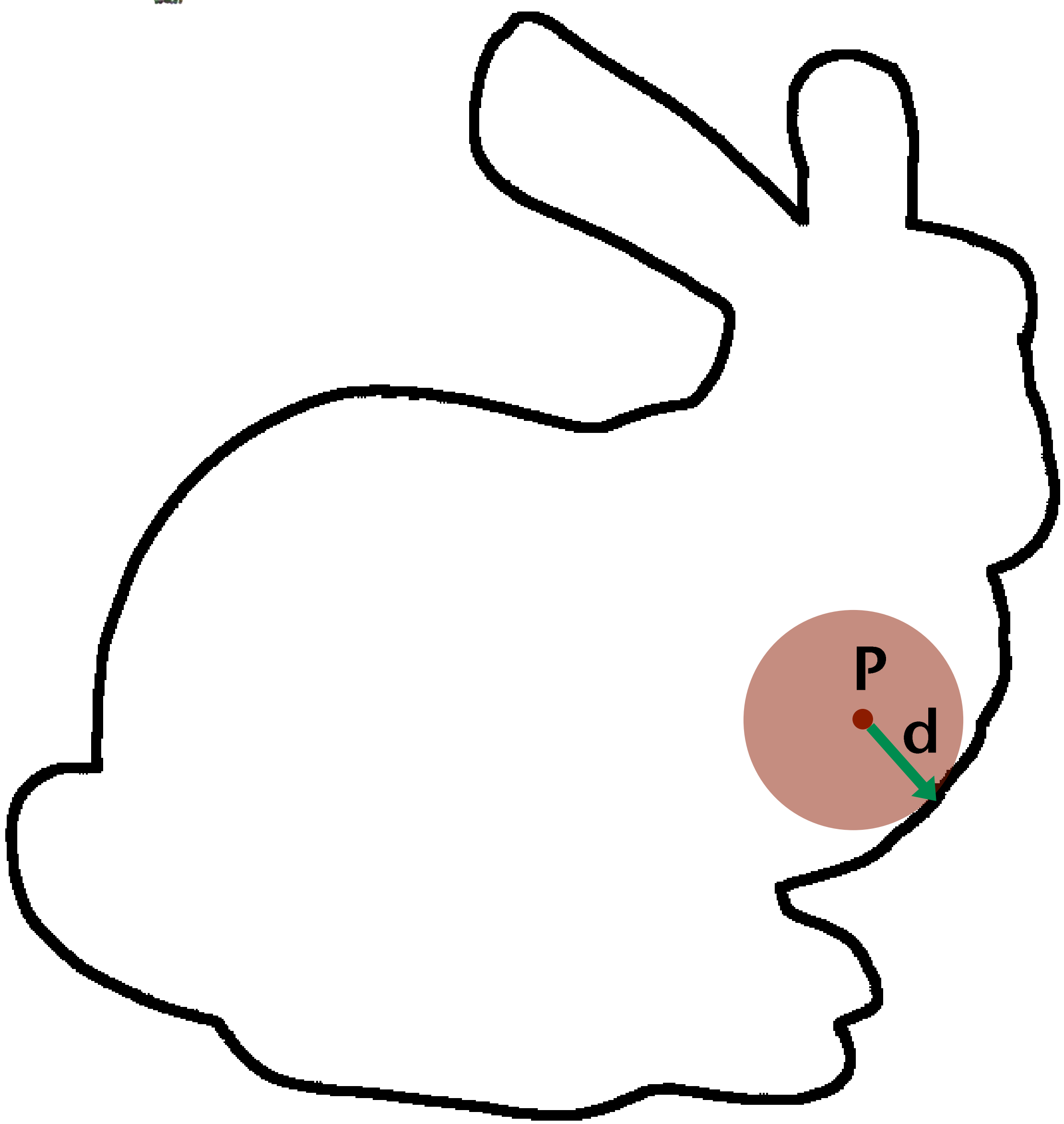


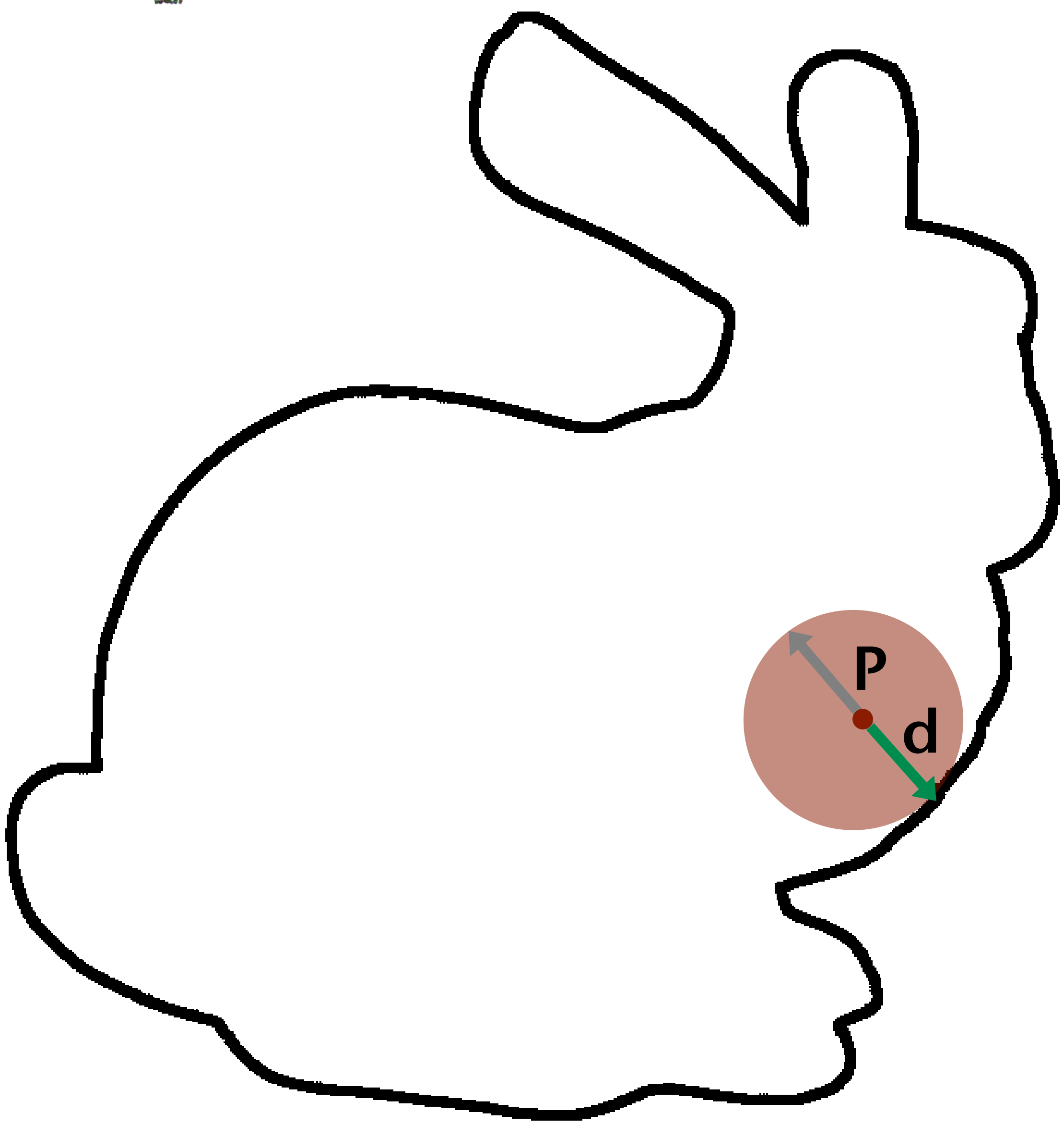

P

Protosphere - Basic Algorithm

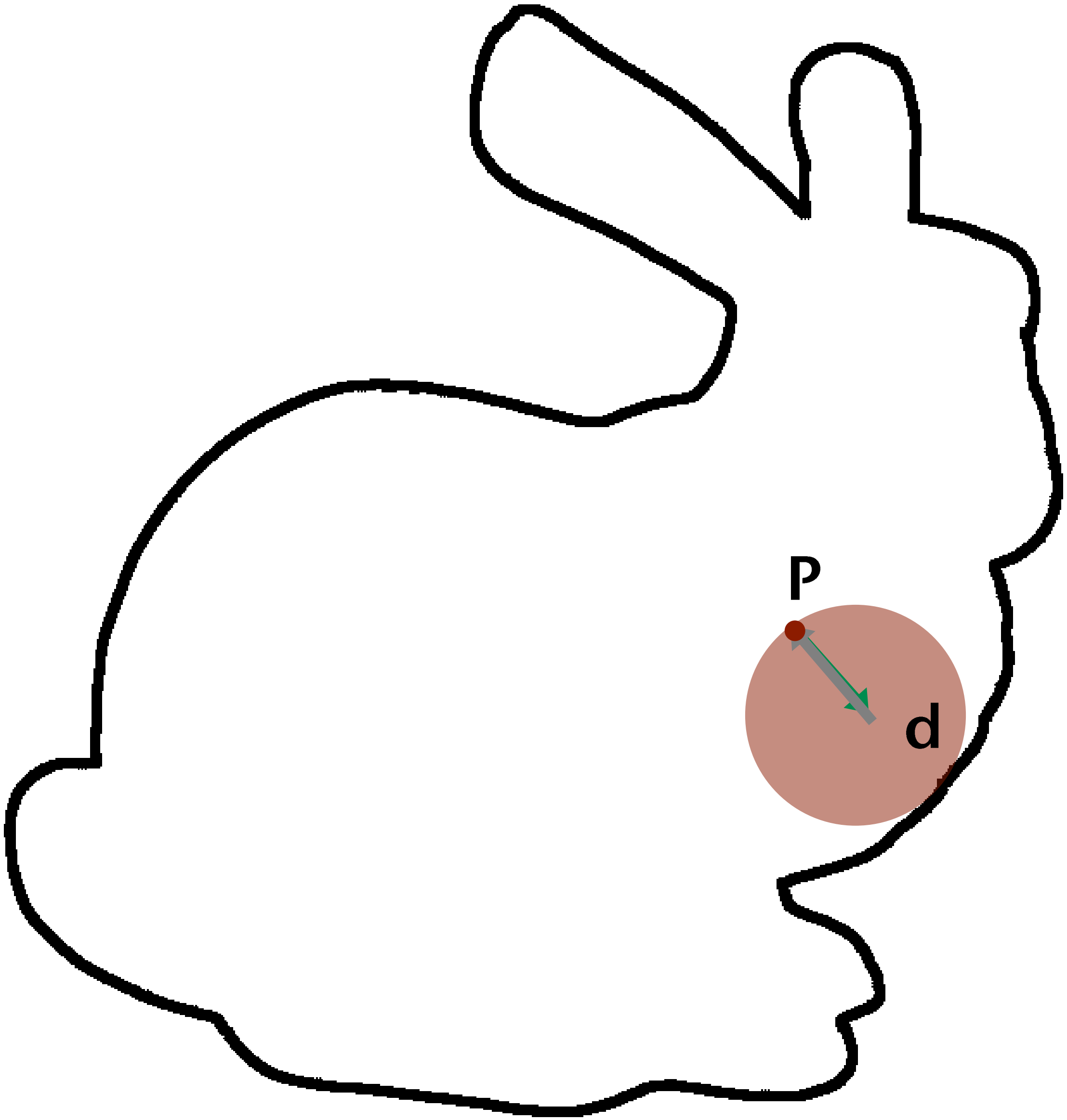


Protosphere - Basic Algorithm

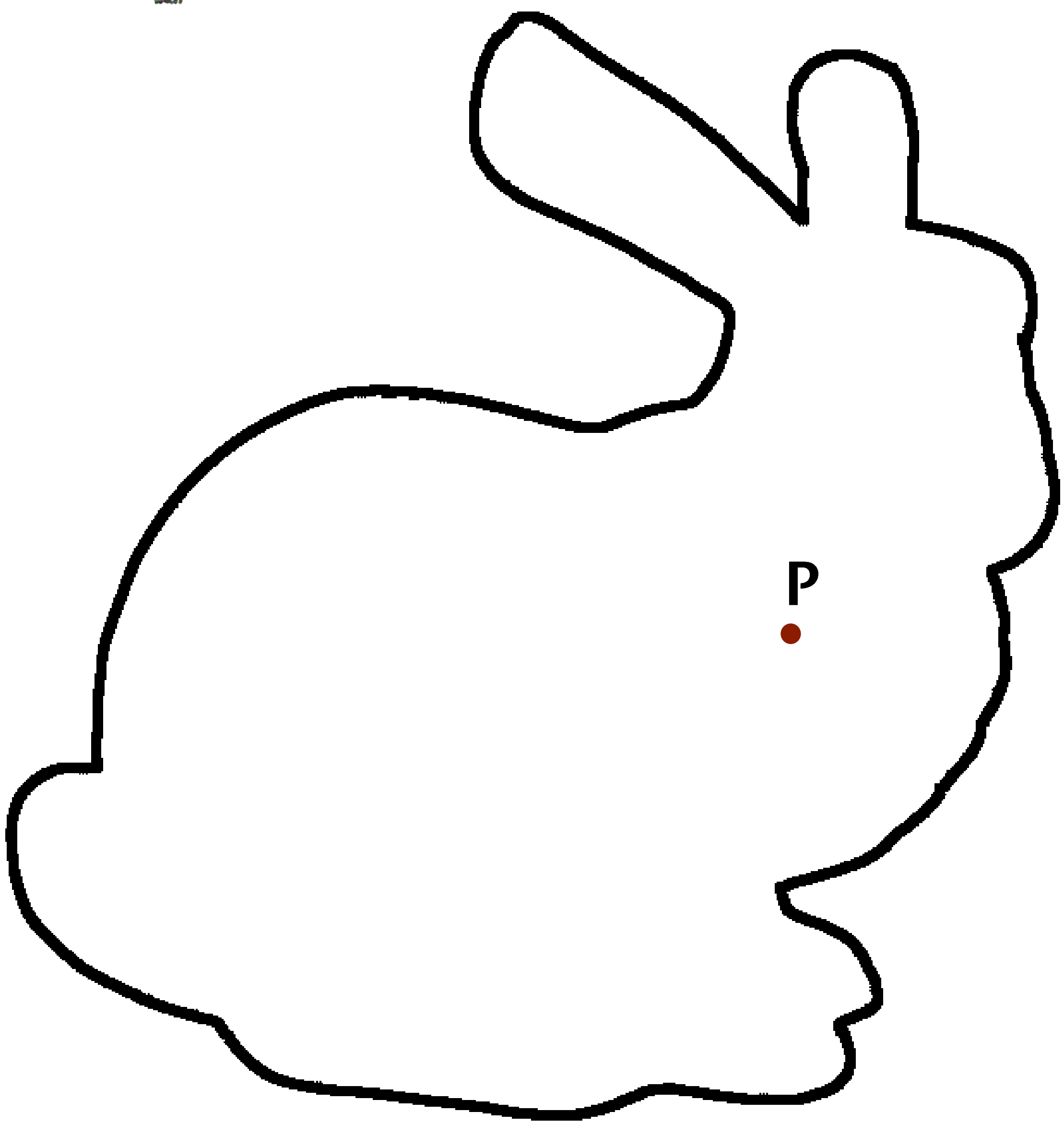



P
d

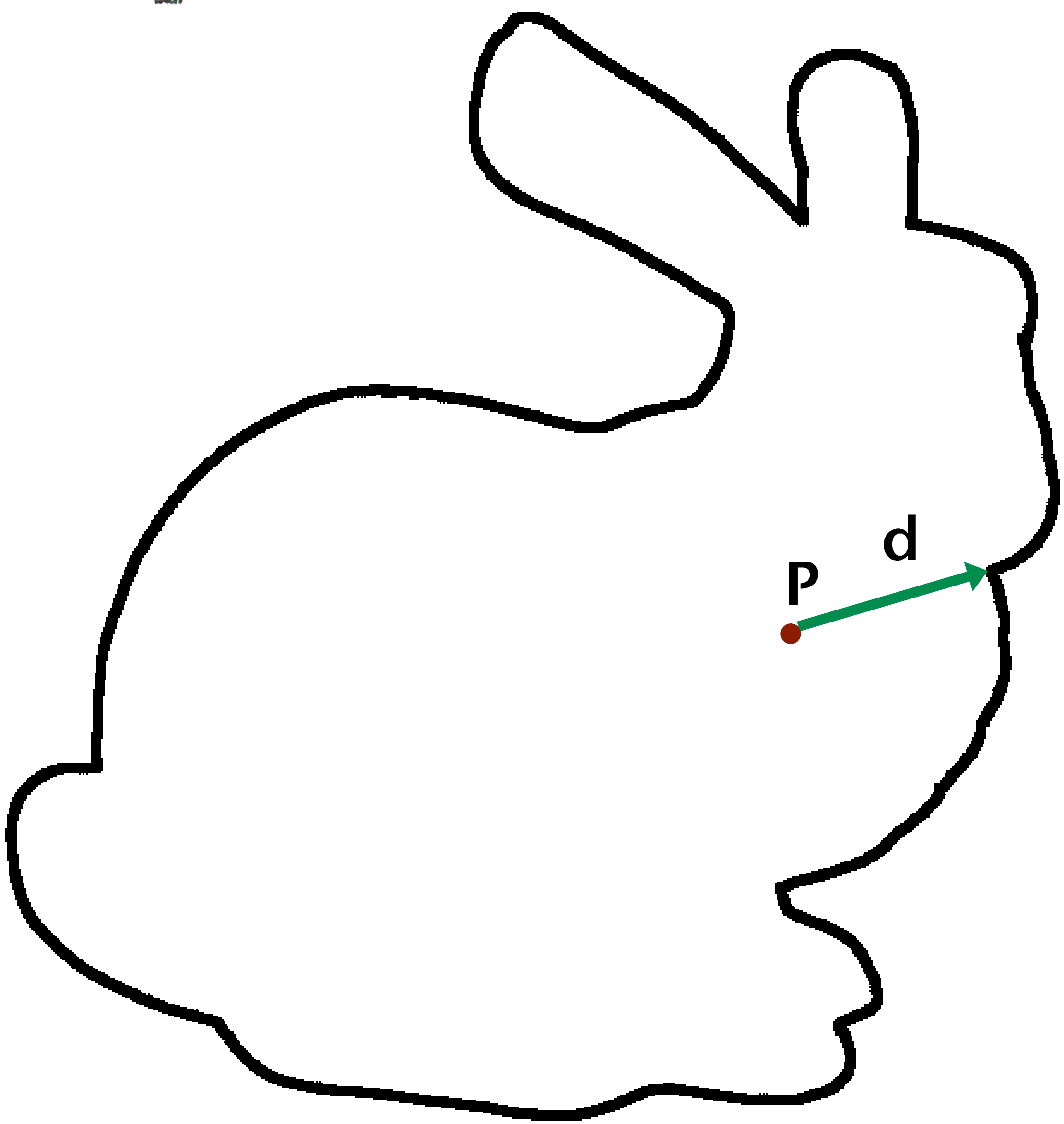
Protosphere - Basic Algorithm



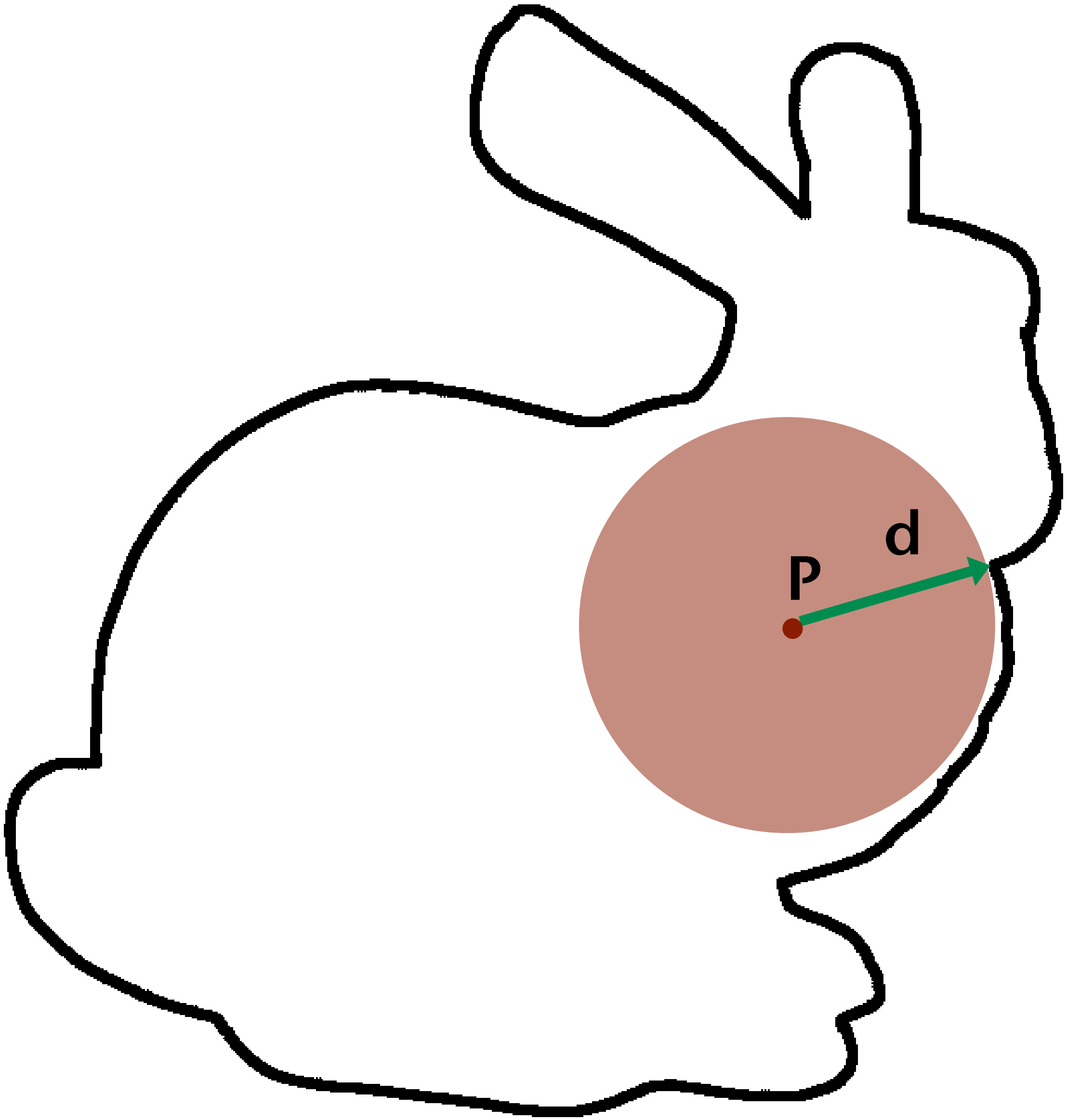
Protosphere - Basic Algorithm



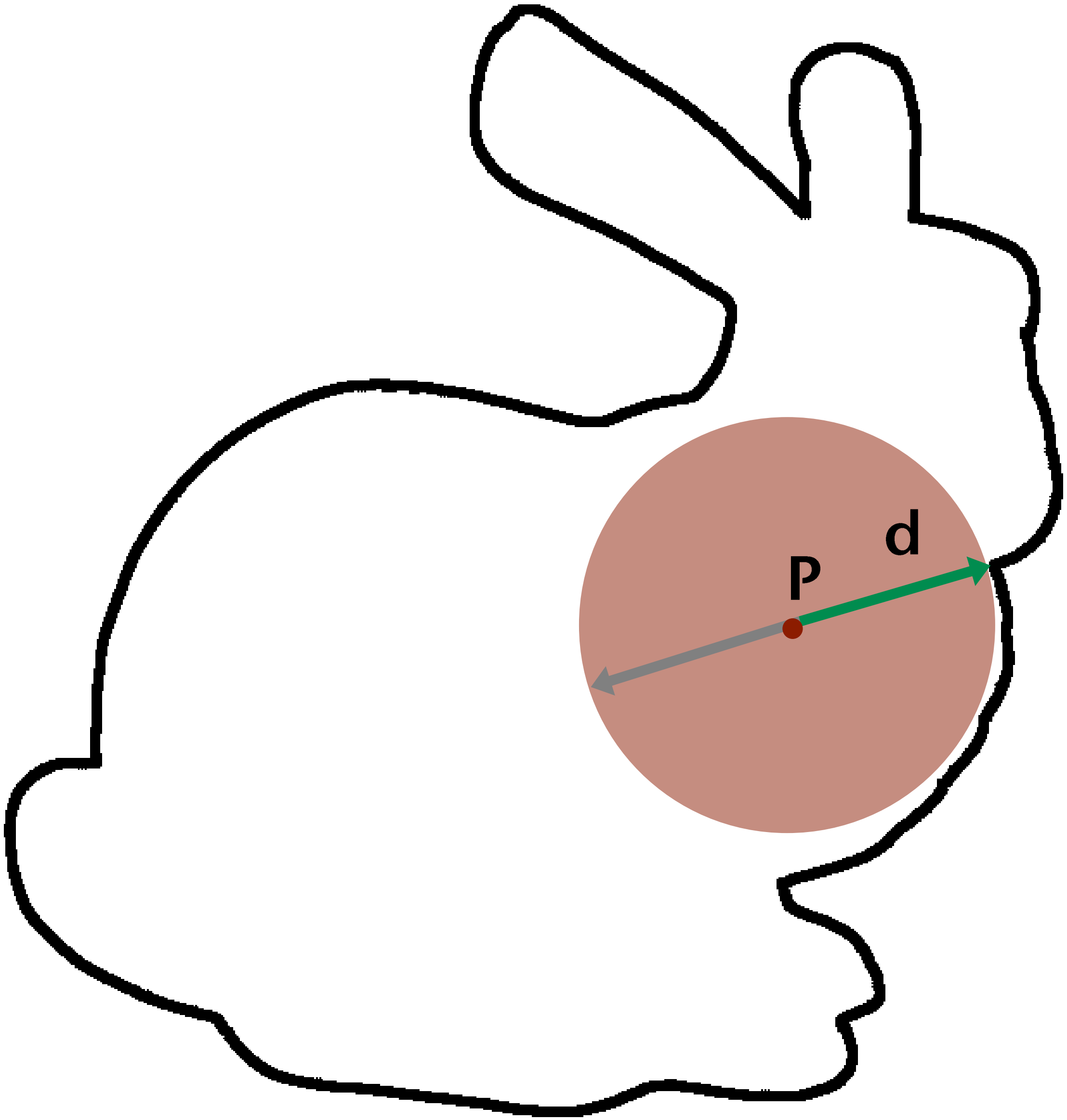
Protosphere - Basic Algorithm



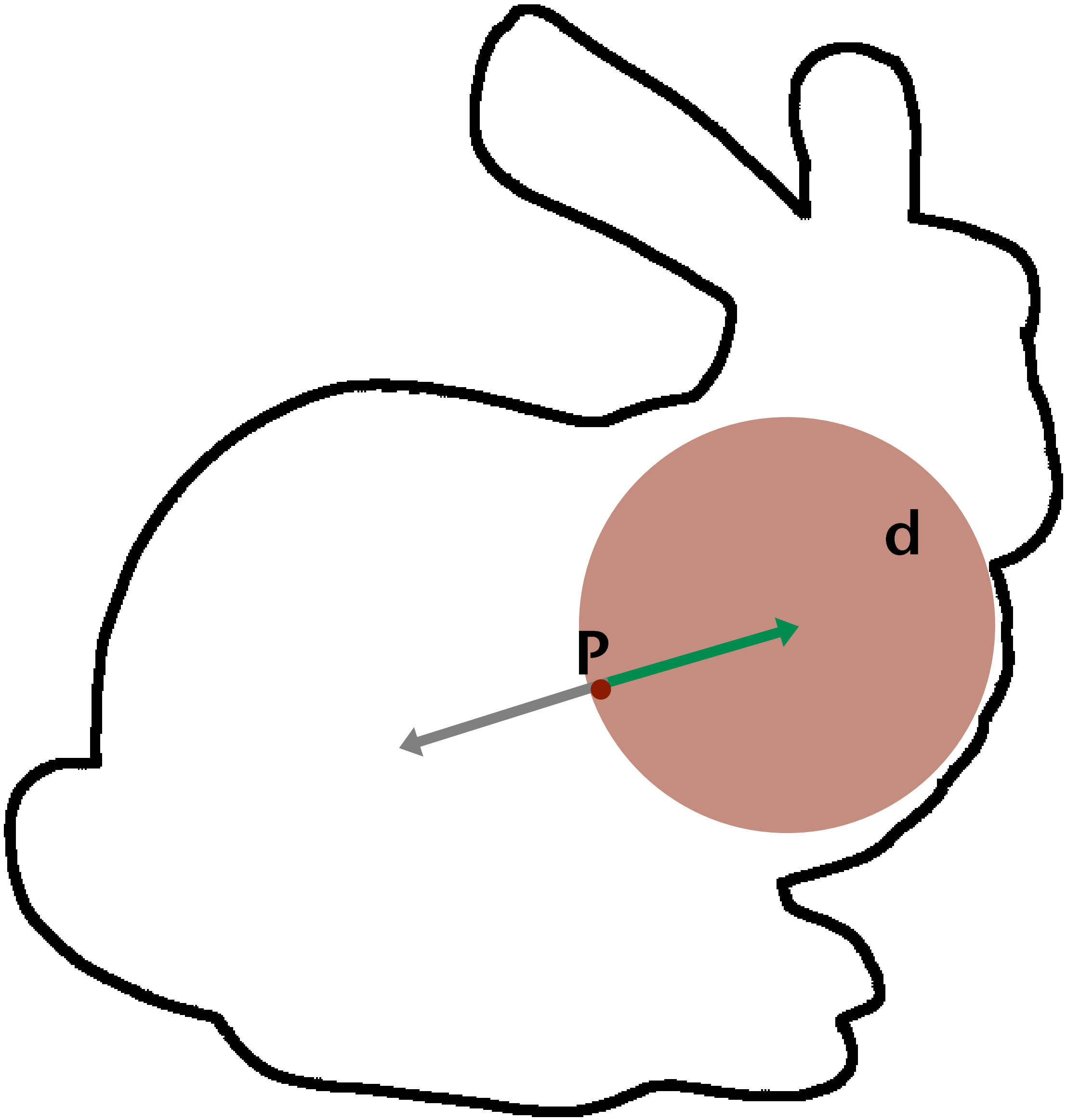
Protosphere - Basic Algorithm

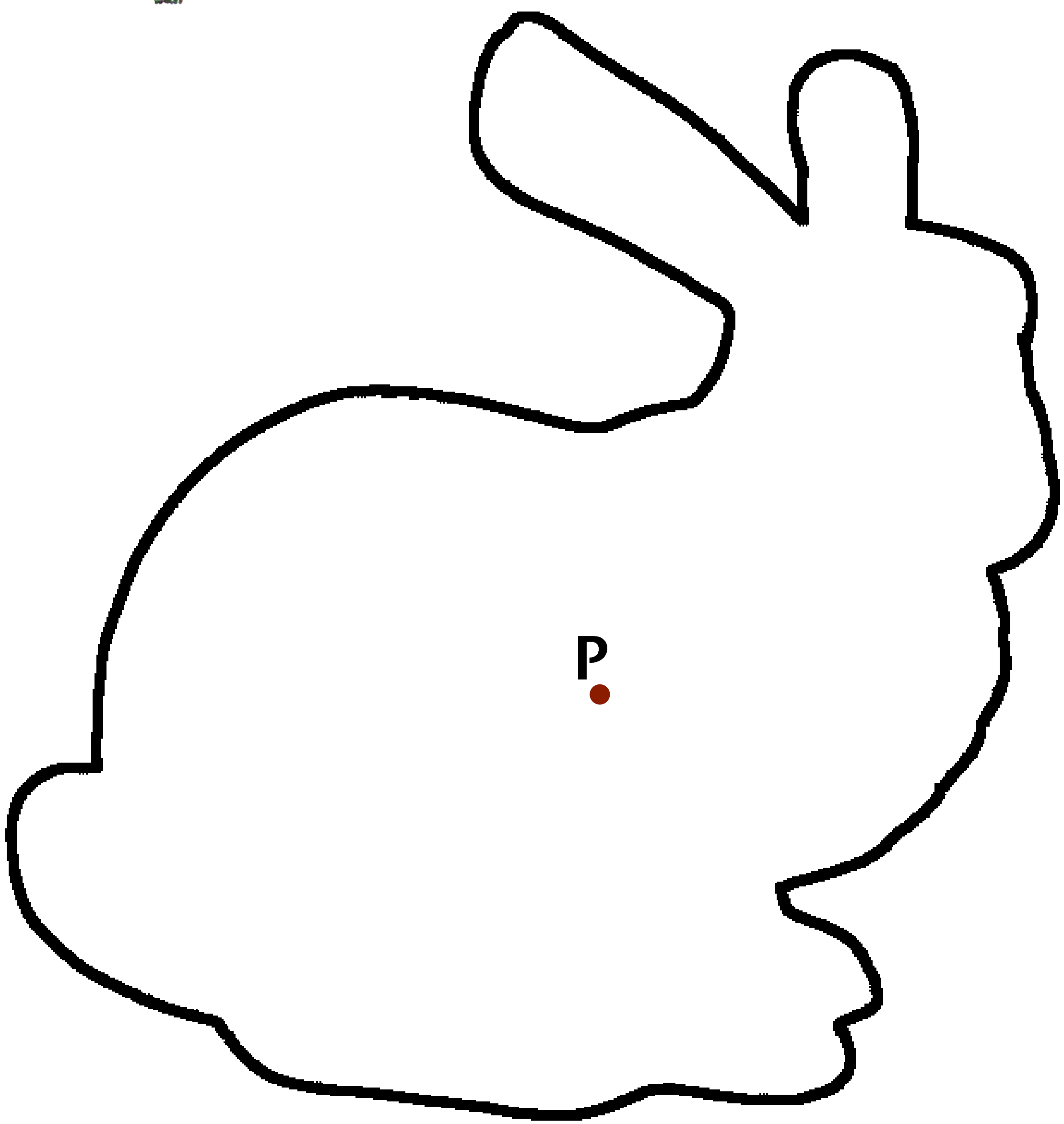


Protosphere - Basic Algorithm

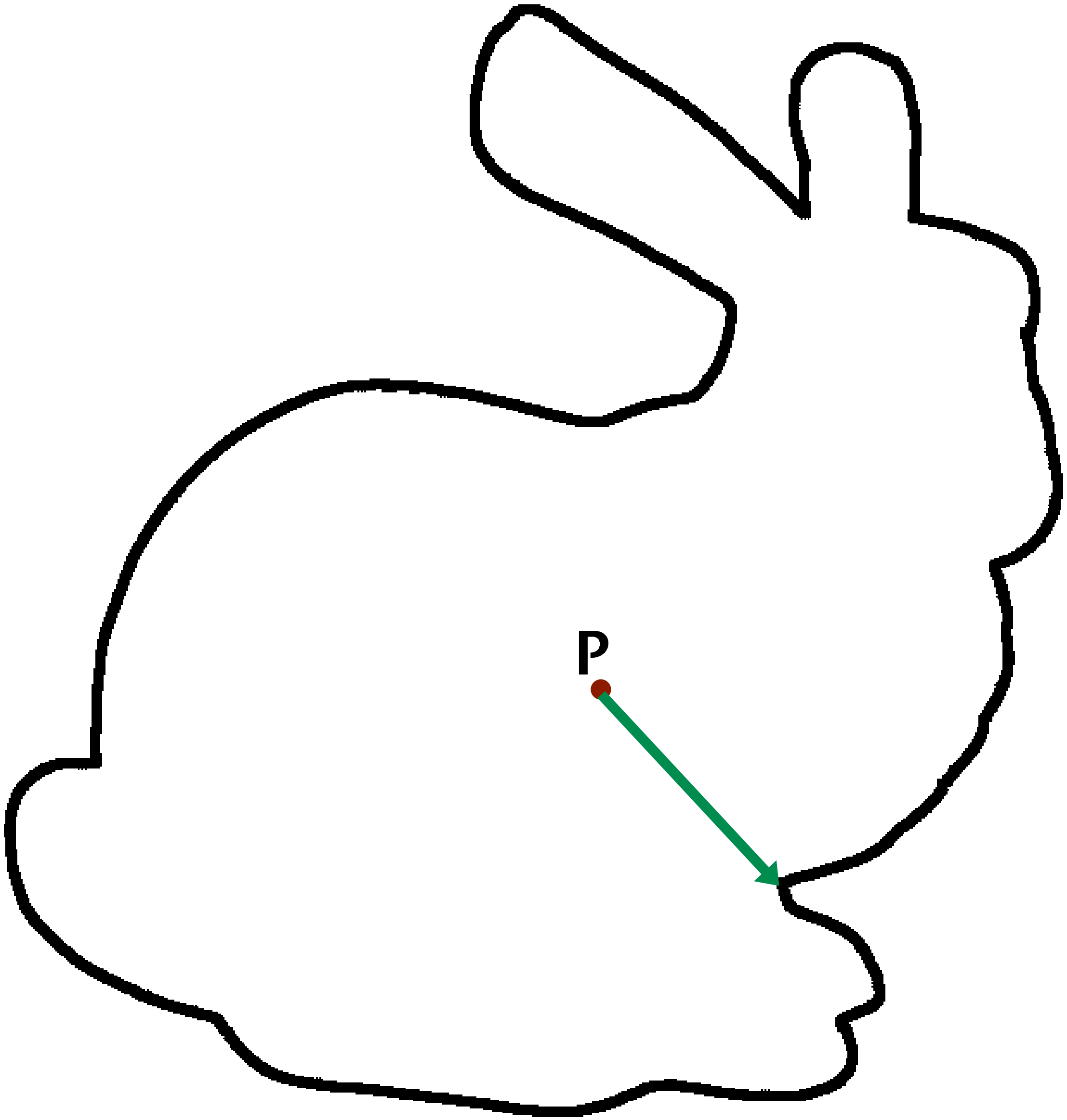


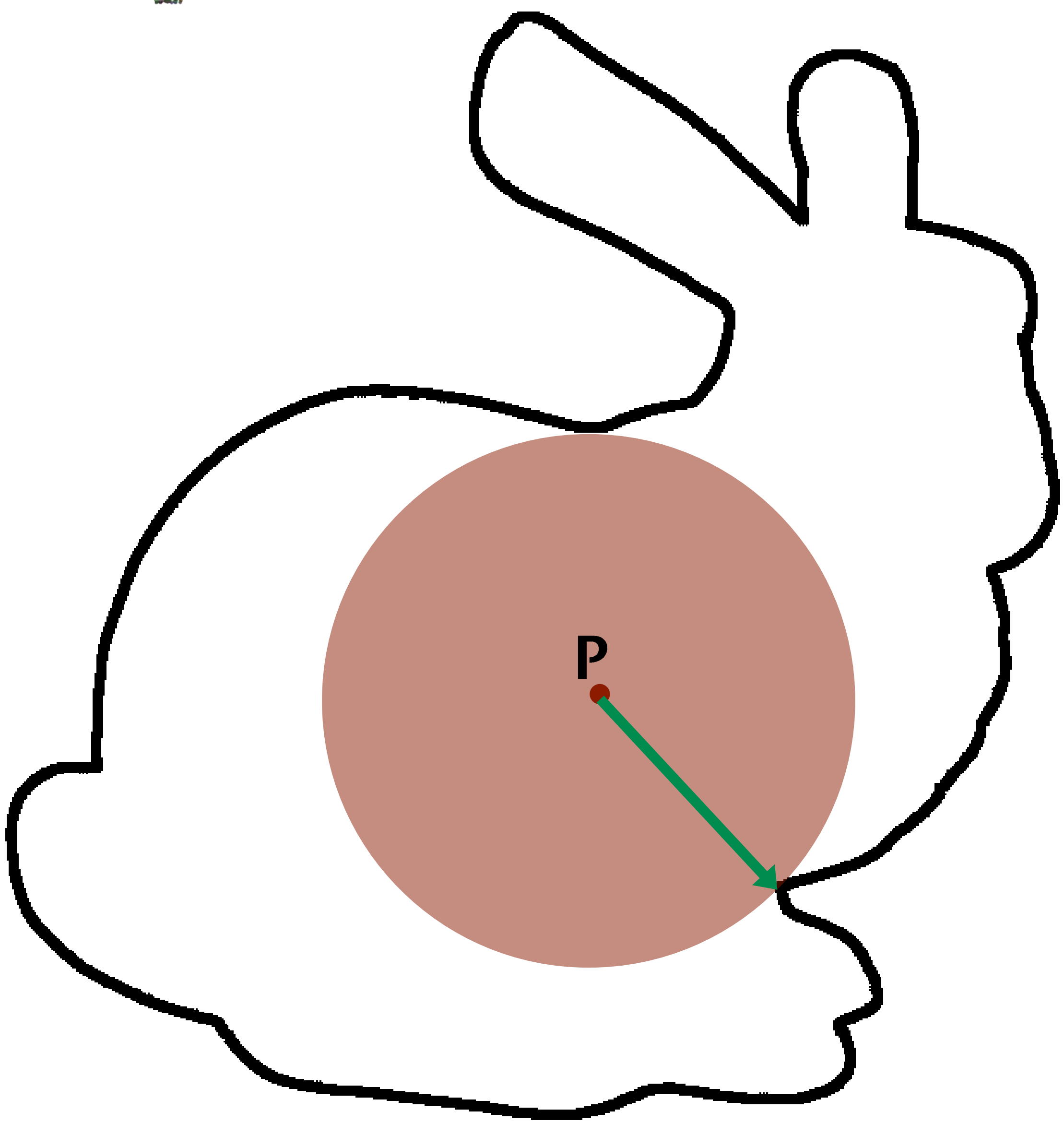
Protosphere - Basic Algorithm



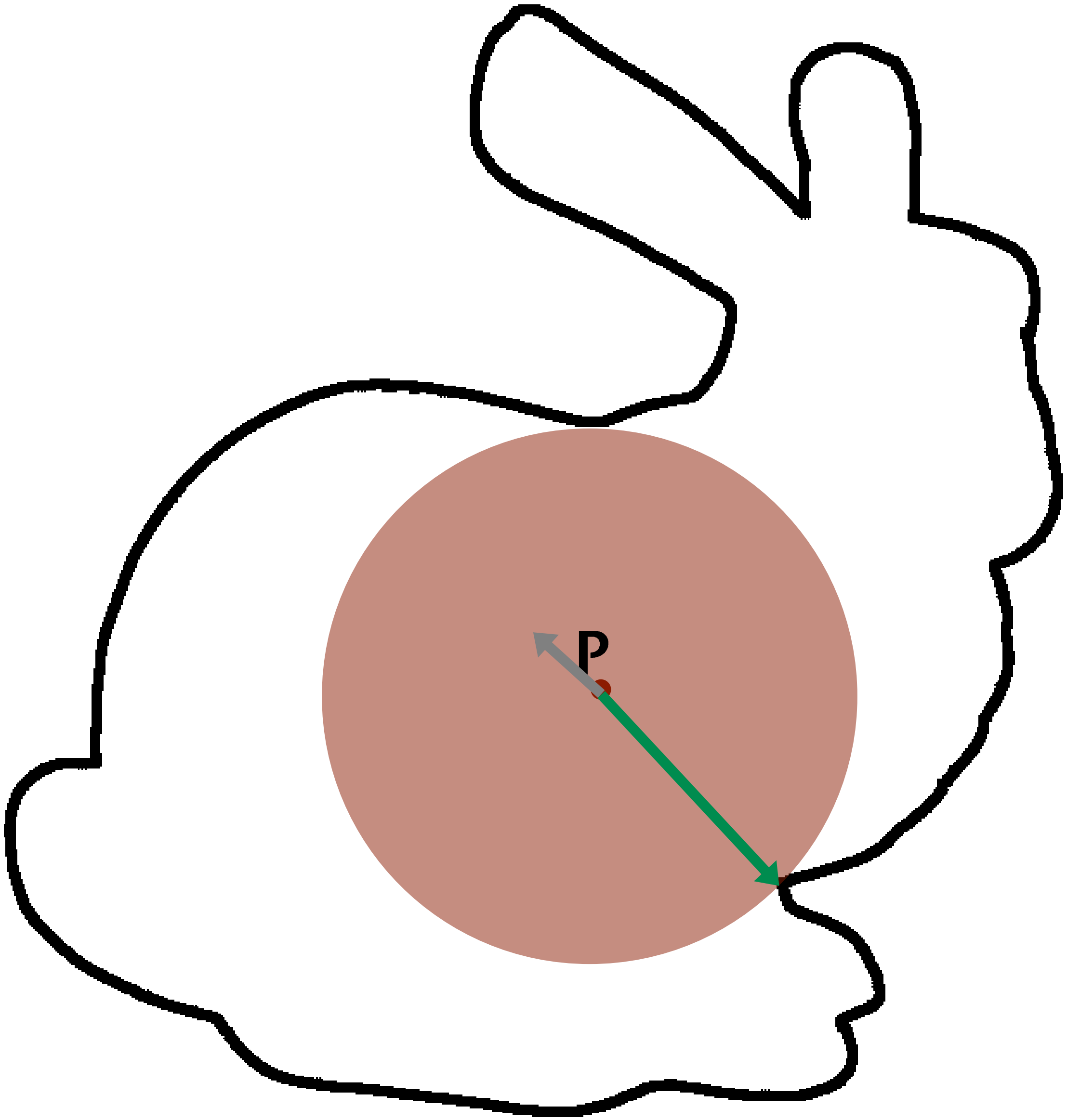

P

Protosphere - Basic Algorithm

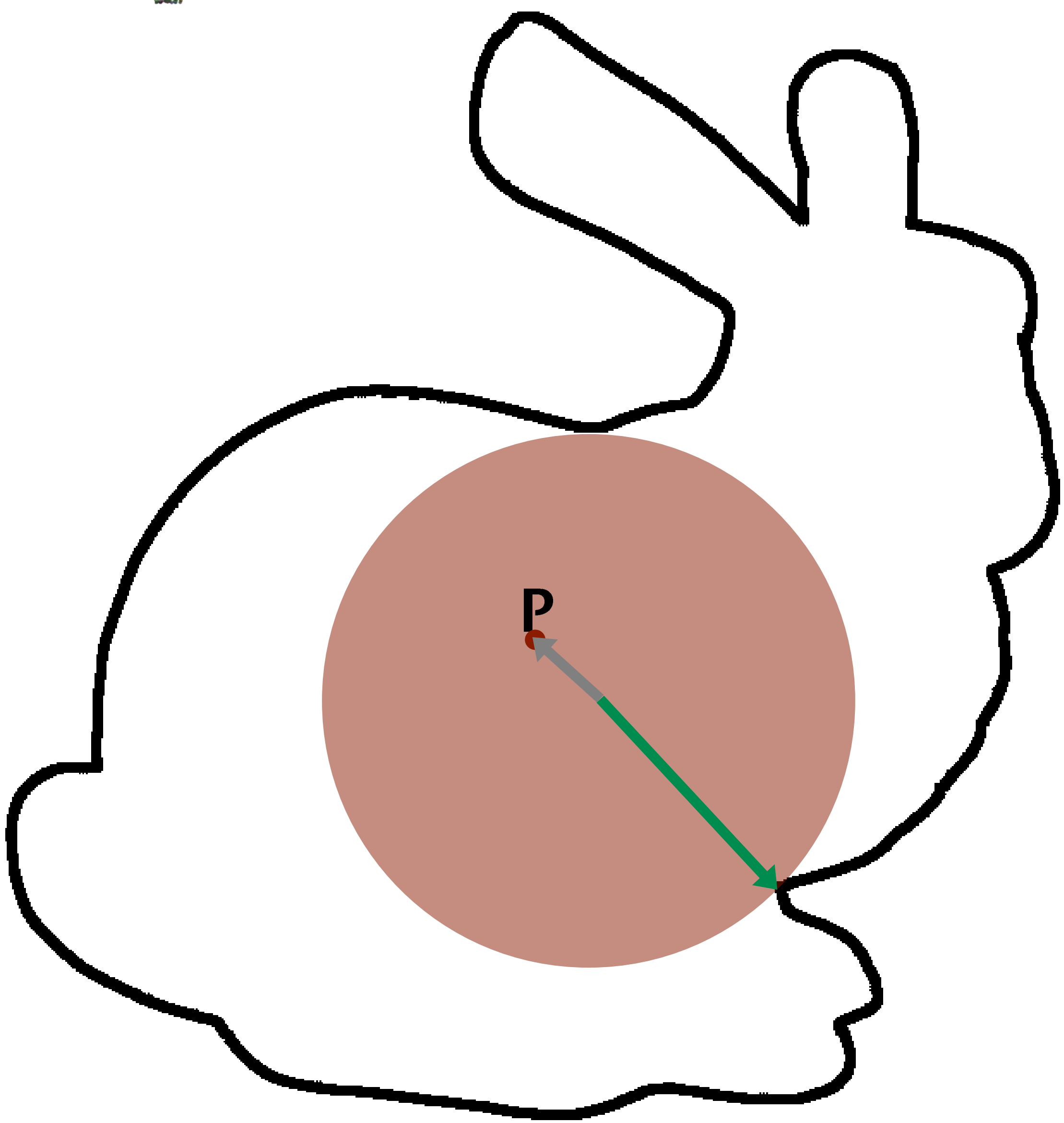


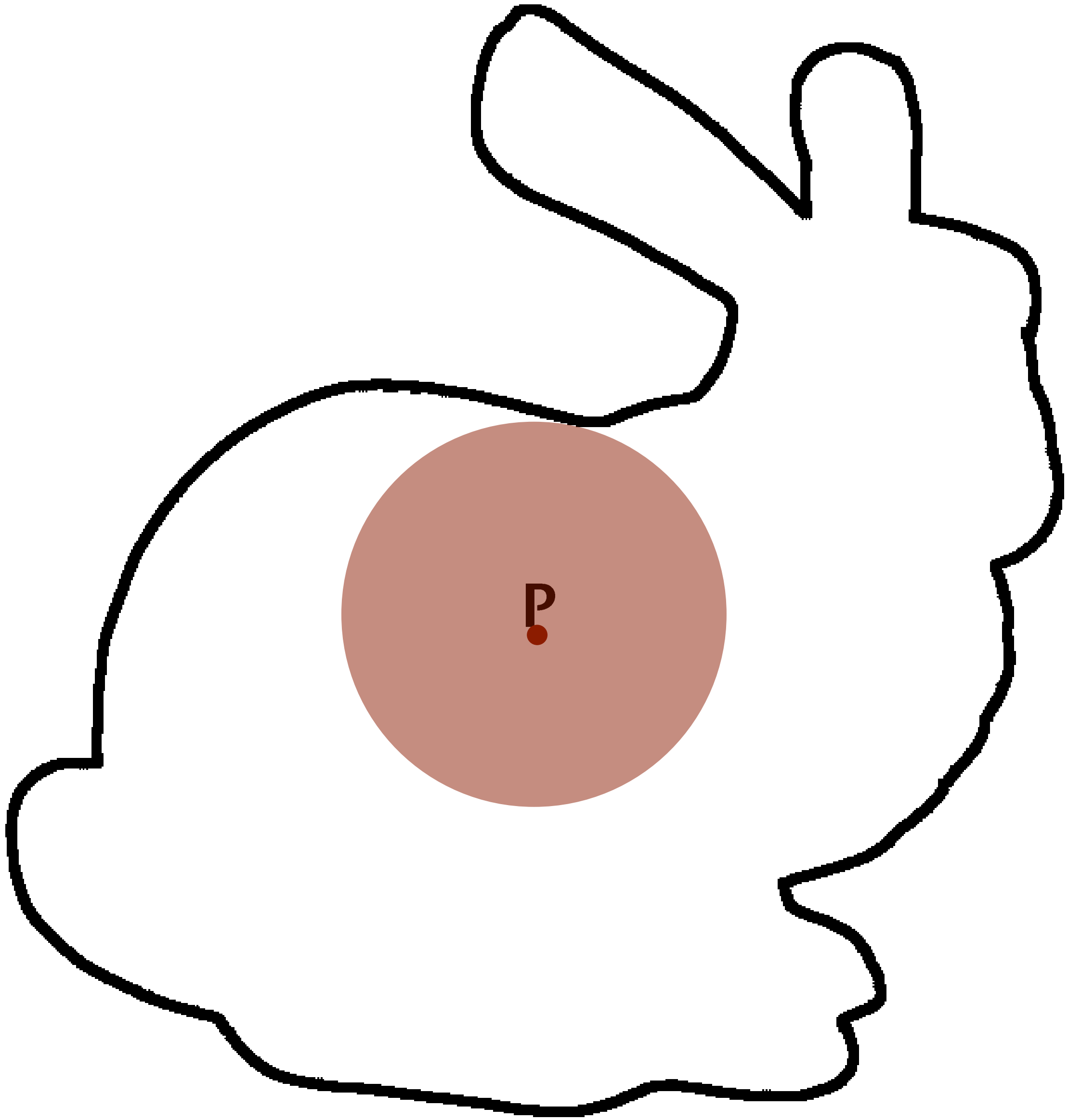

P

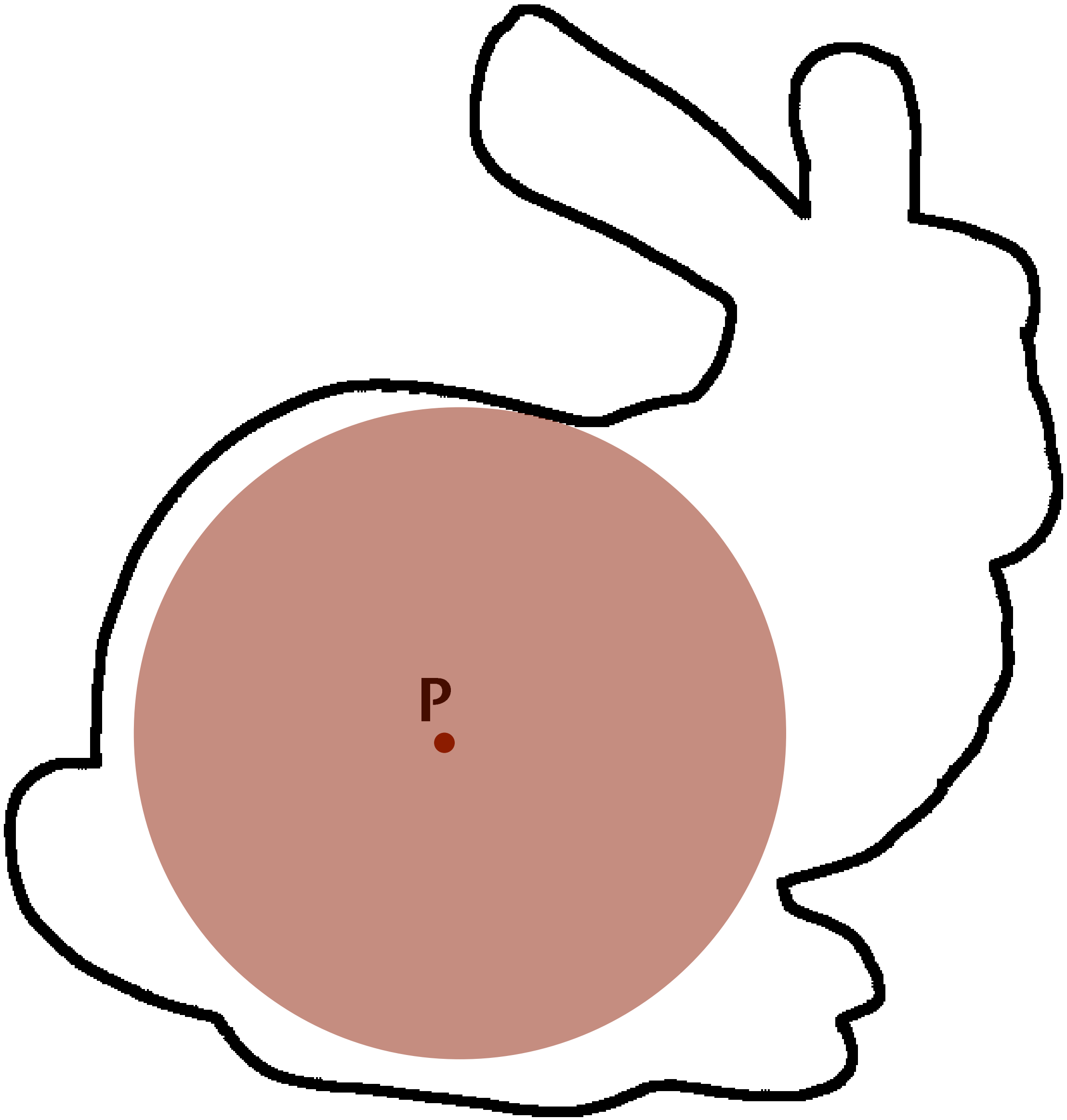
Protosphere - Basic Algorithm

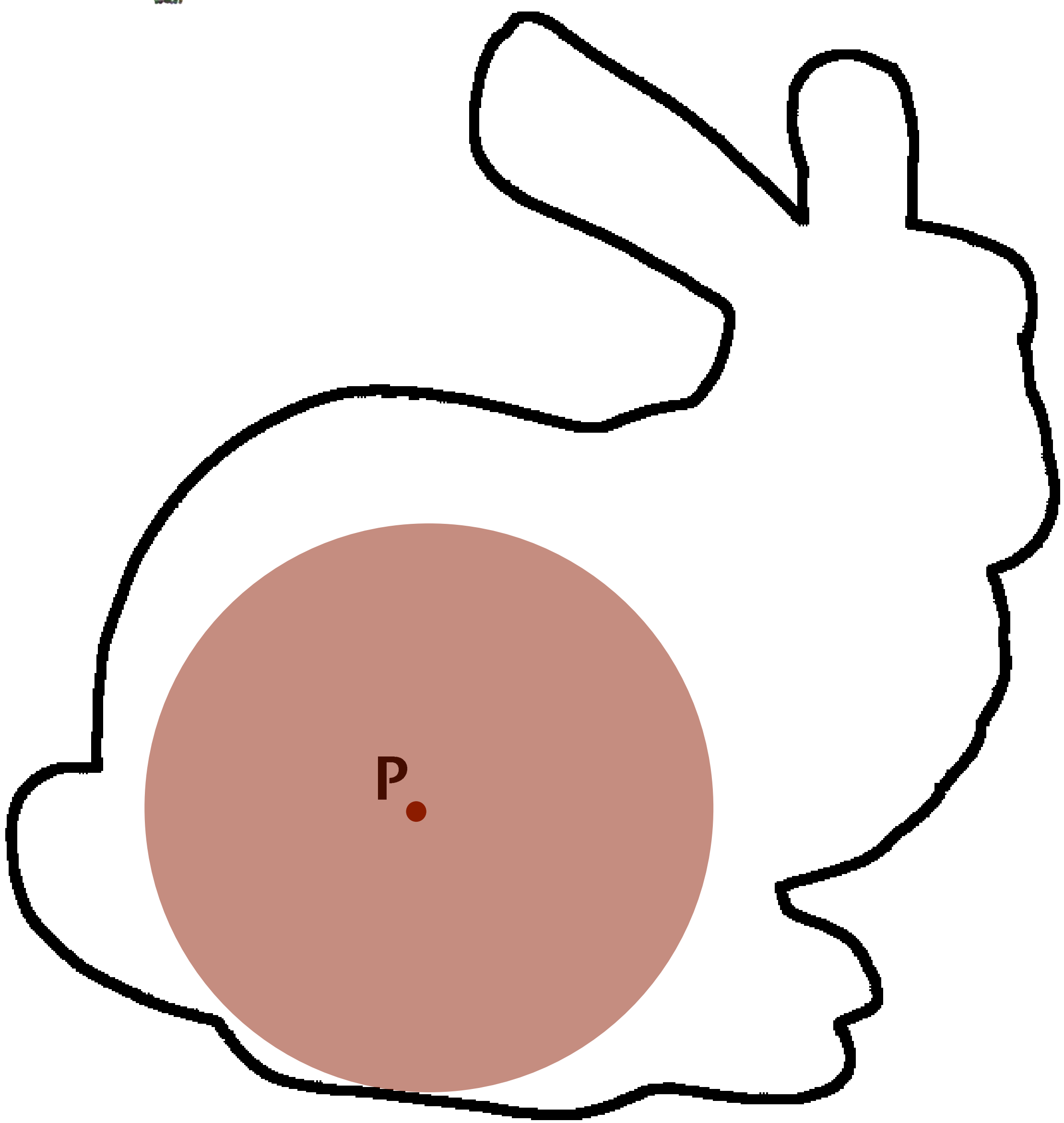


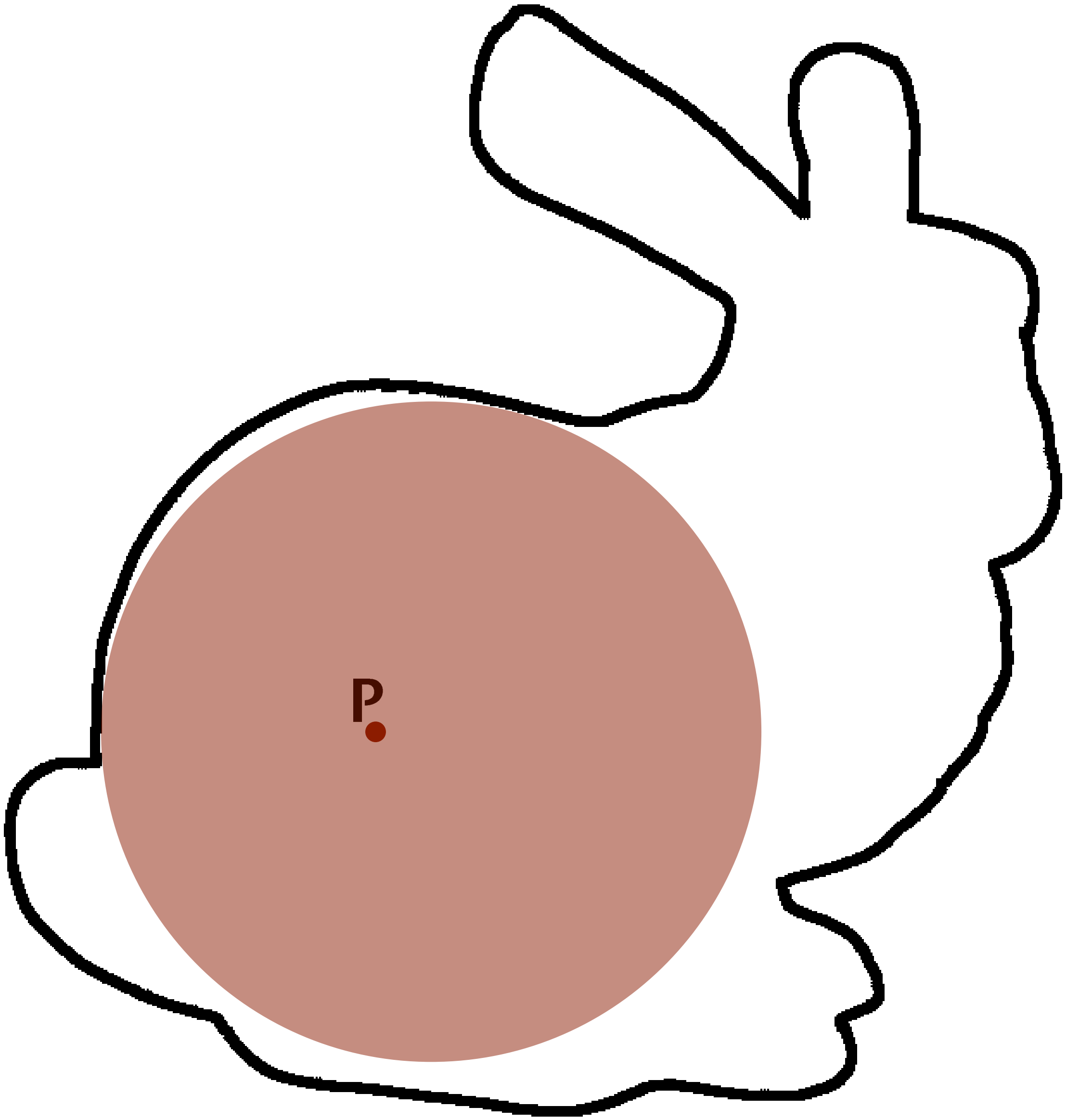
Protosphere - Basic Algorithm



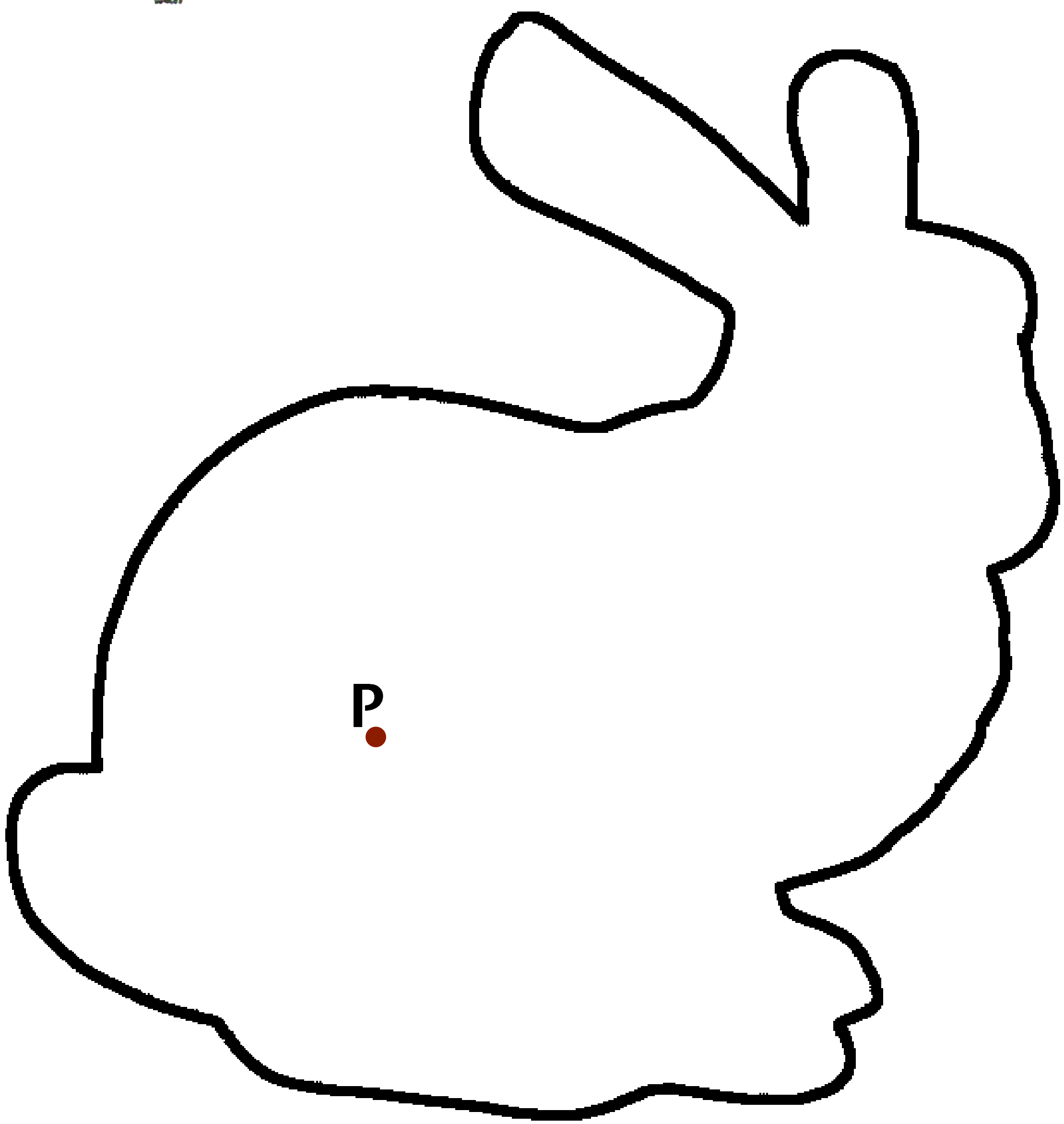

P


P


P

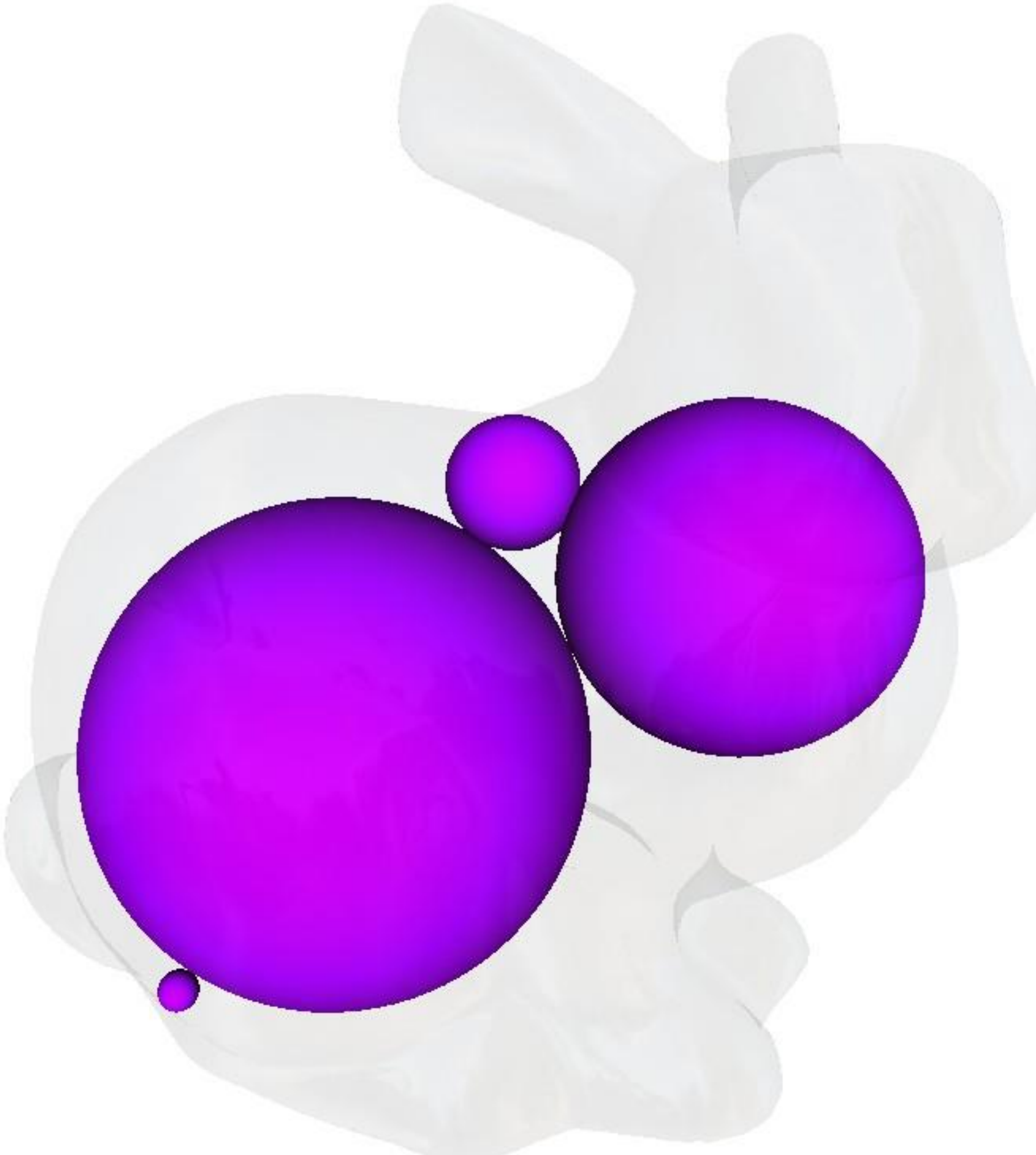

P

Protosphere - Basic Algorithm

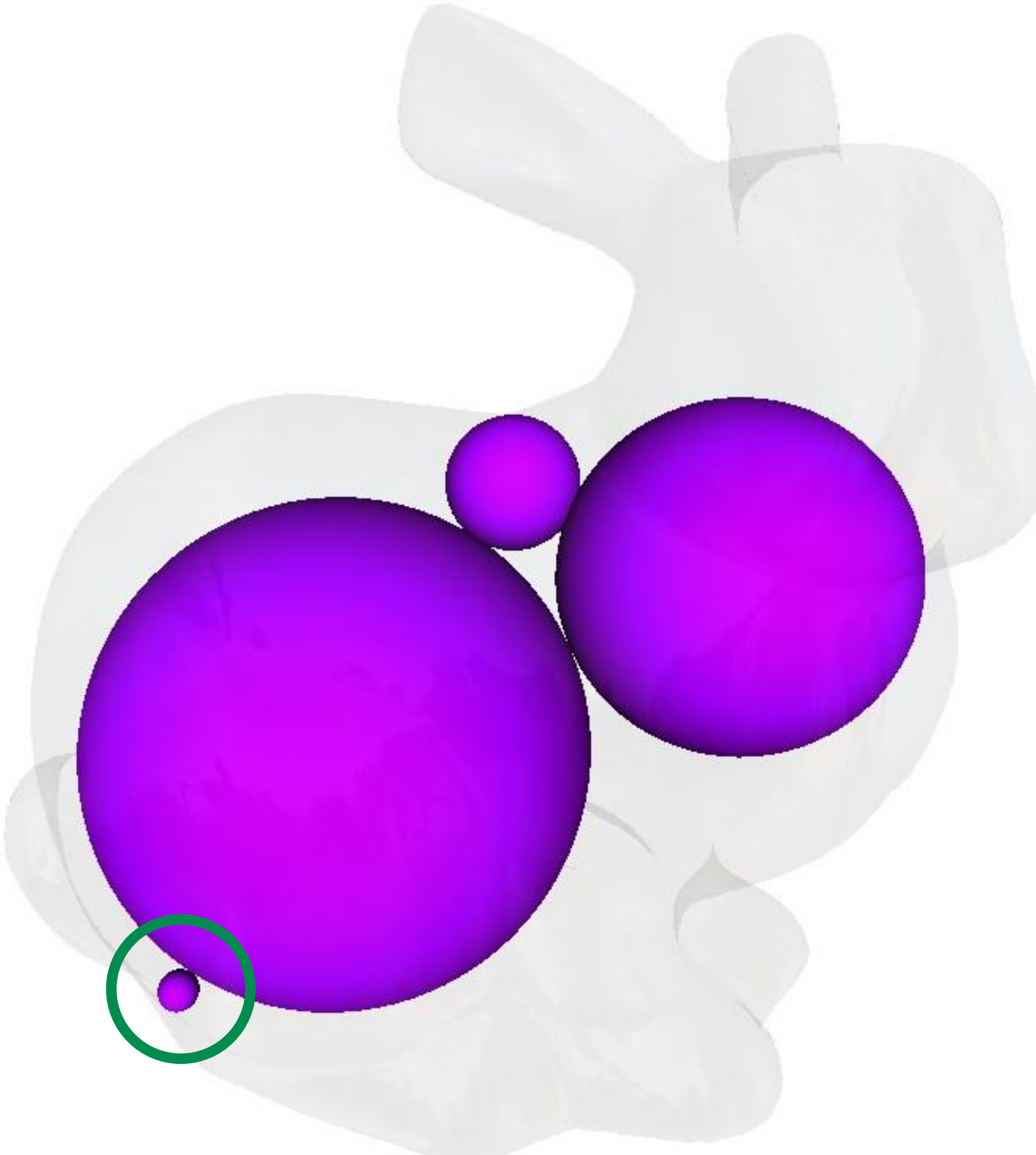


Protosphere - Basic Algorithm

Protosphere - Basic Algorithm



Protosphere - Basic Algorithm



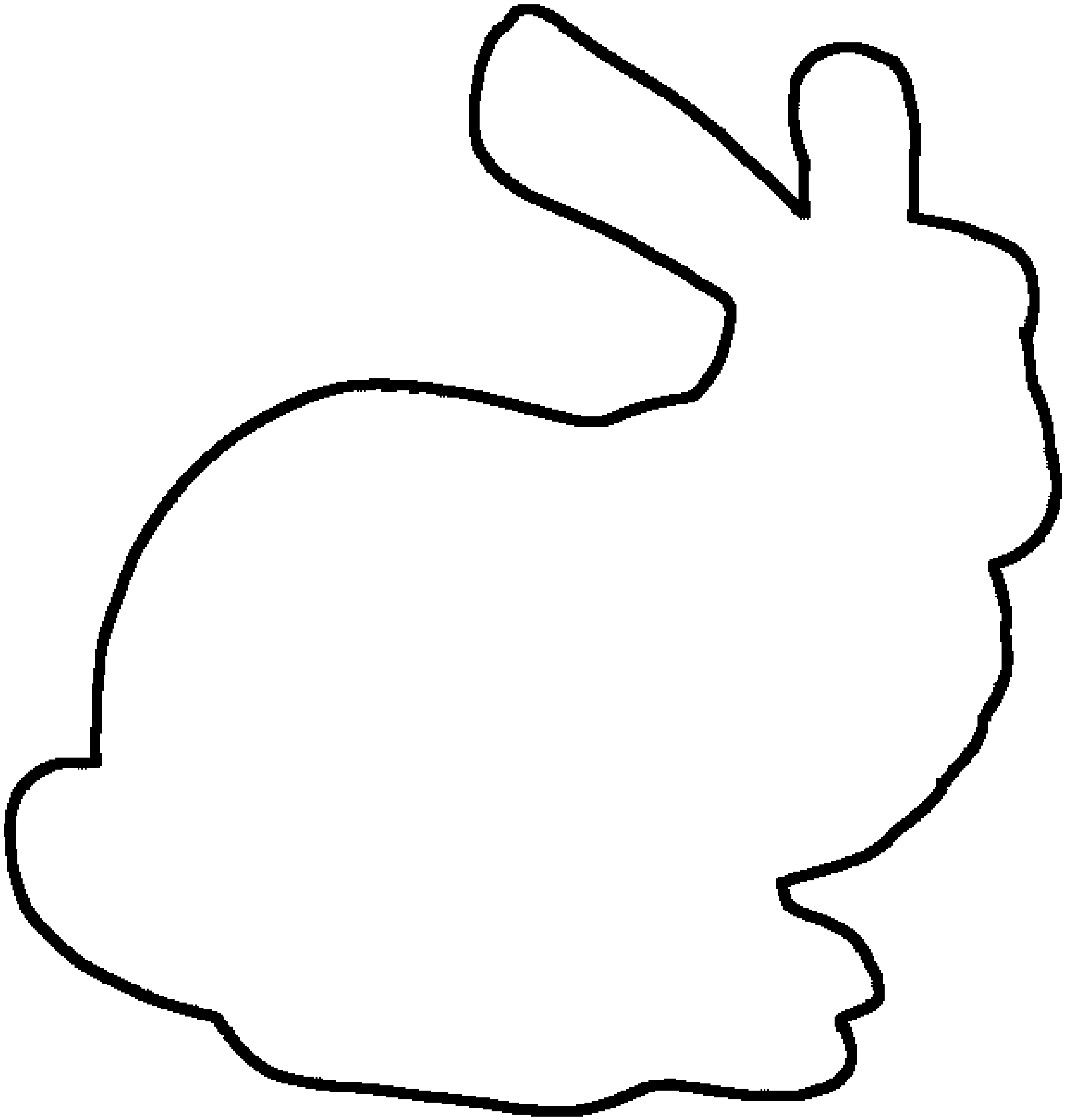
Protosphere - Basic Algorithm

Protosphere - Basic Algorithm

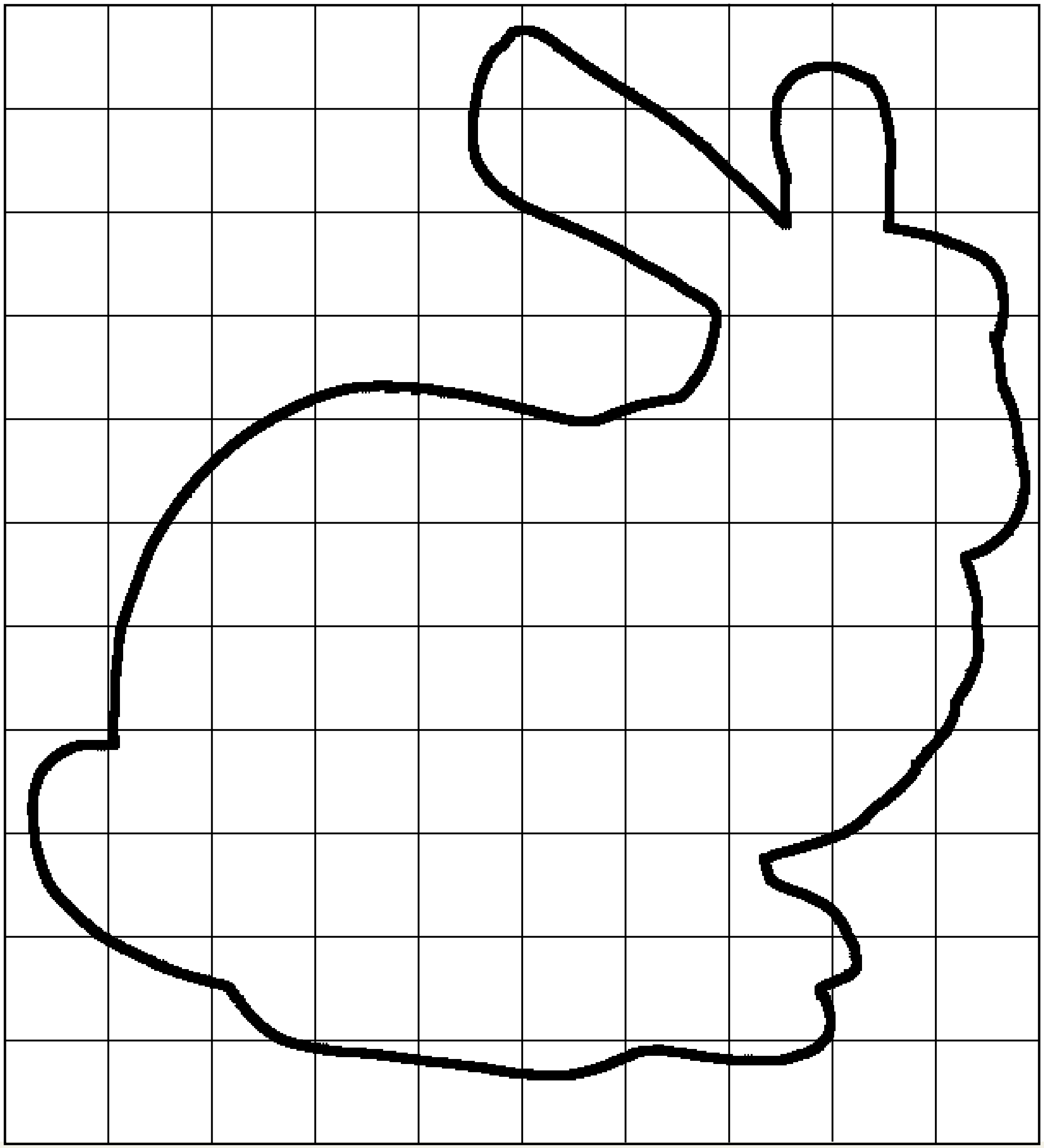
Protosphere - Basic Algorithm

Protosphere - Basic Algorithm

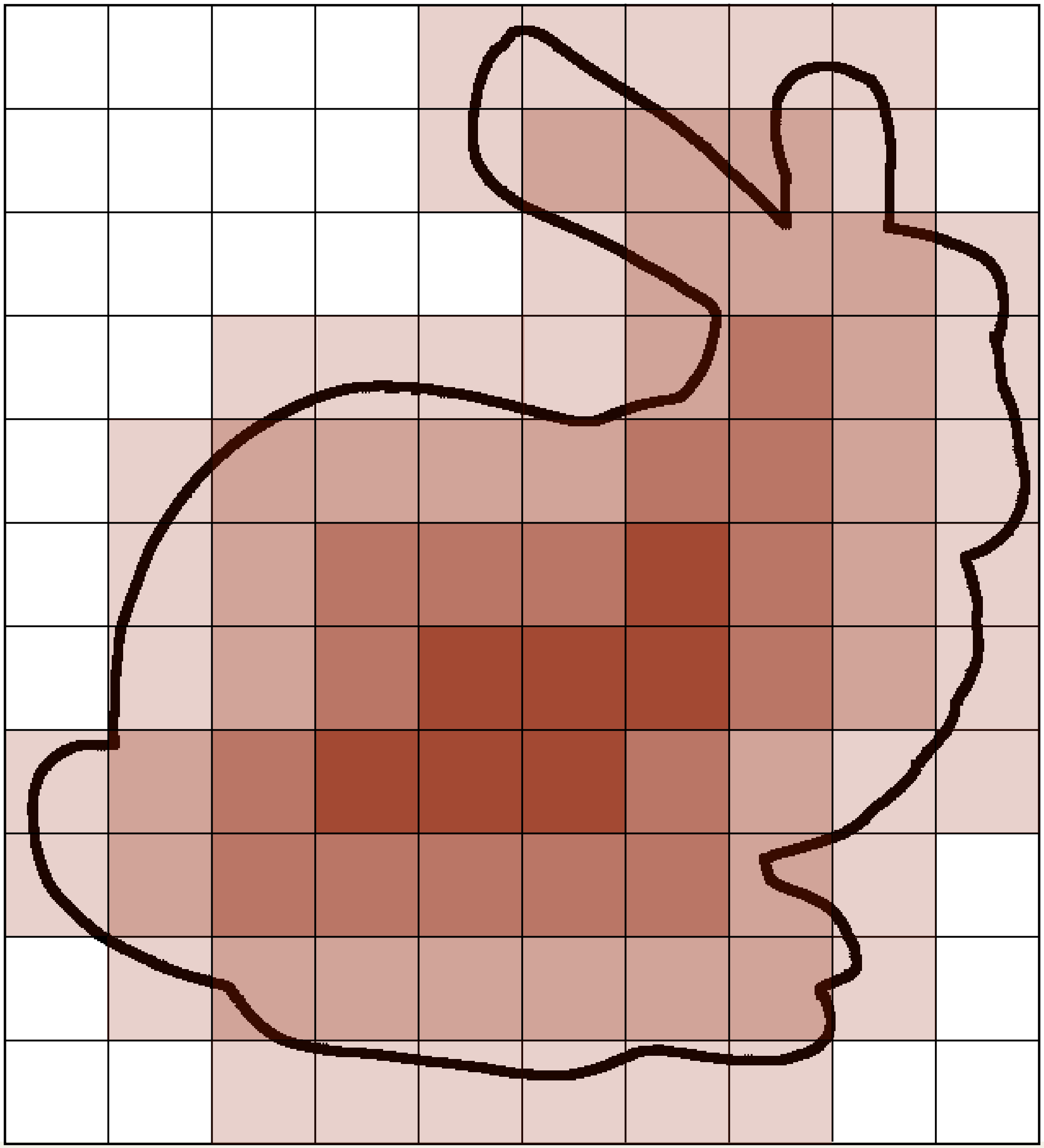
Parallelization



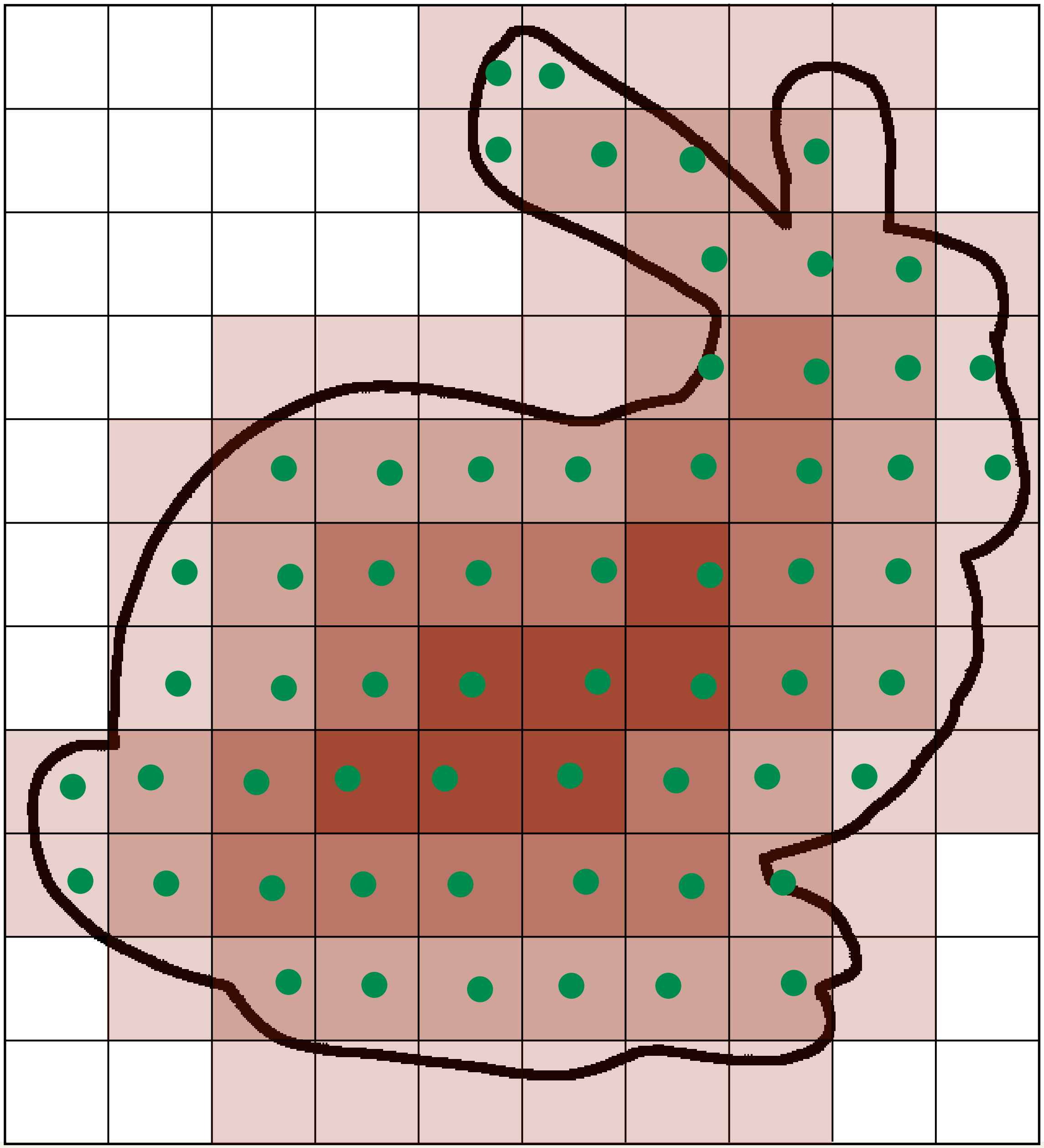
Parallelization



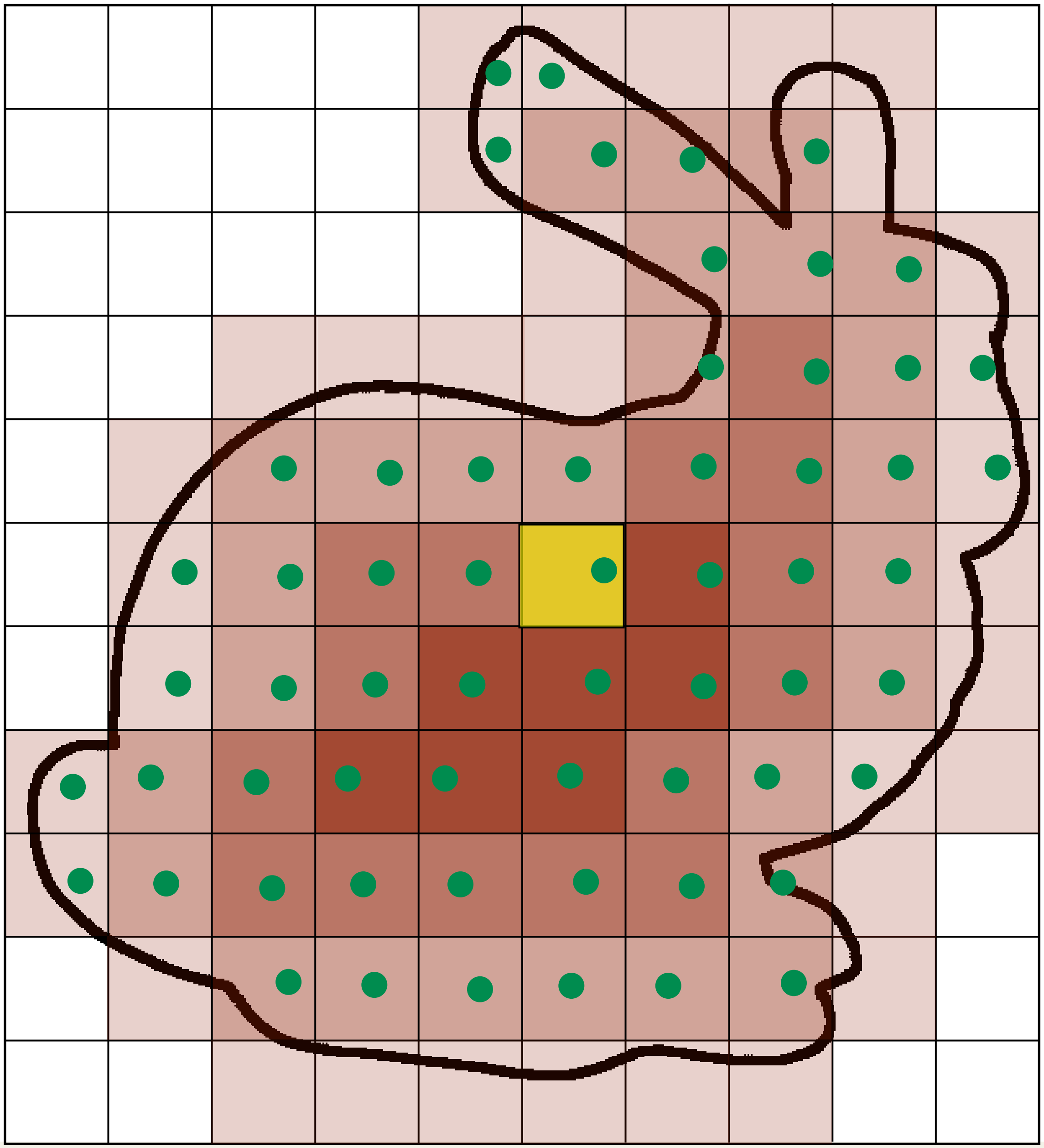
Parallelization



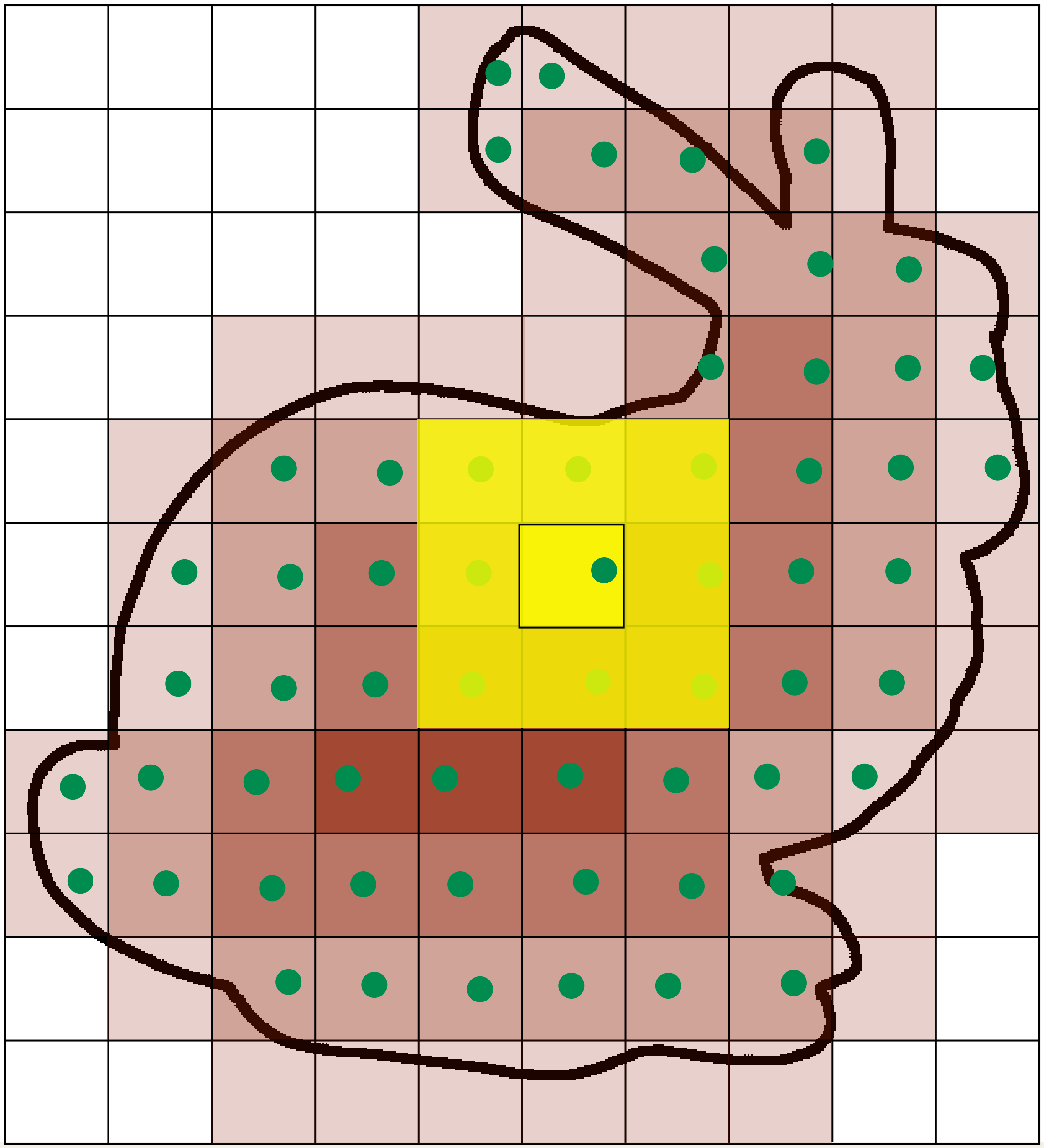
Parallelization



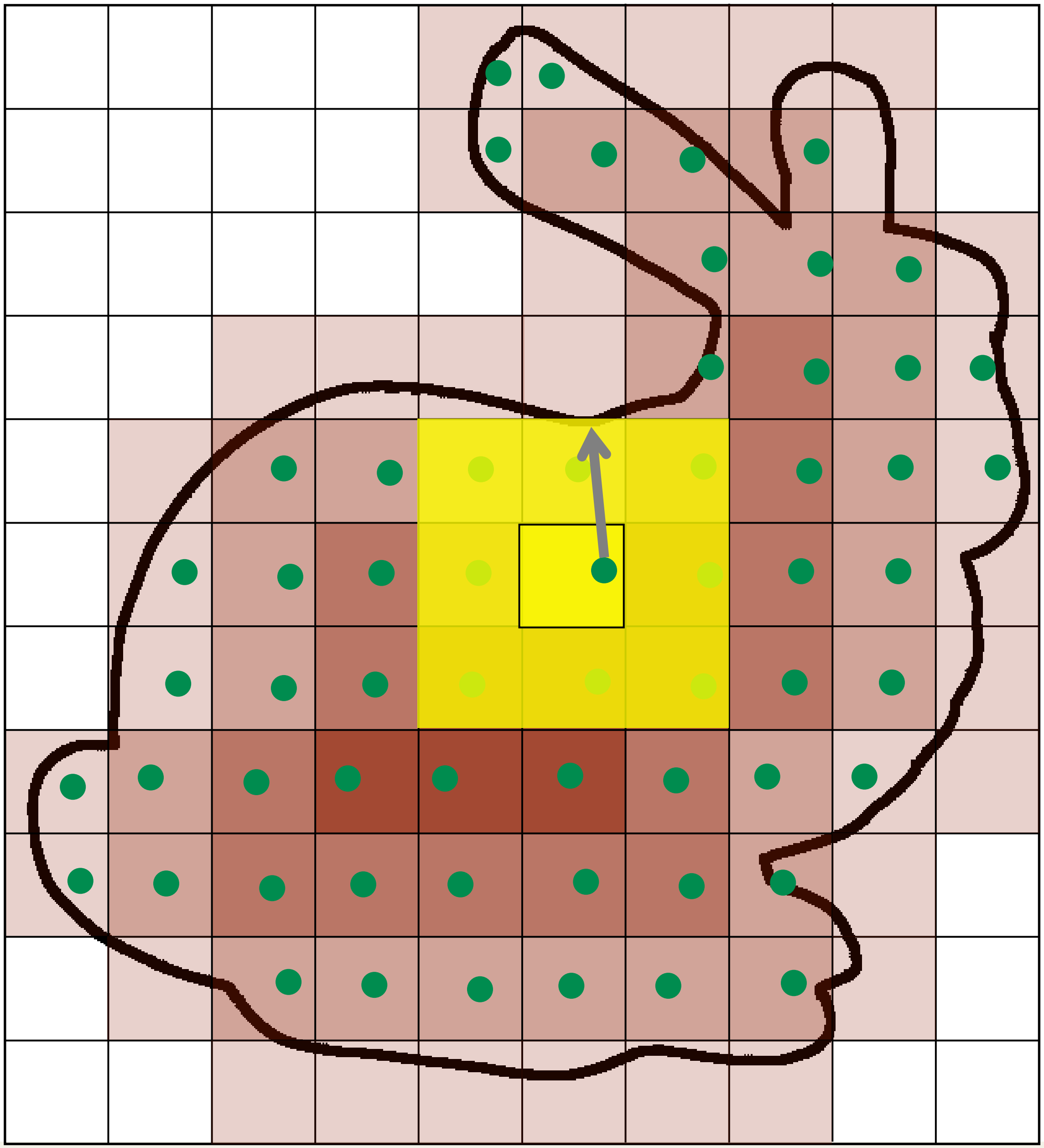
Parallelization



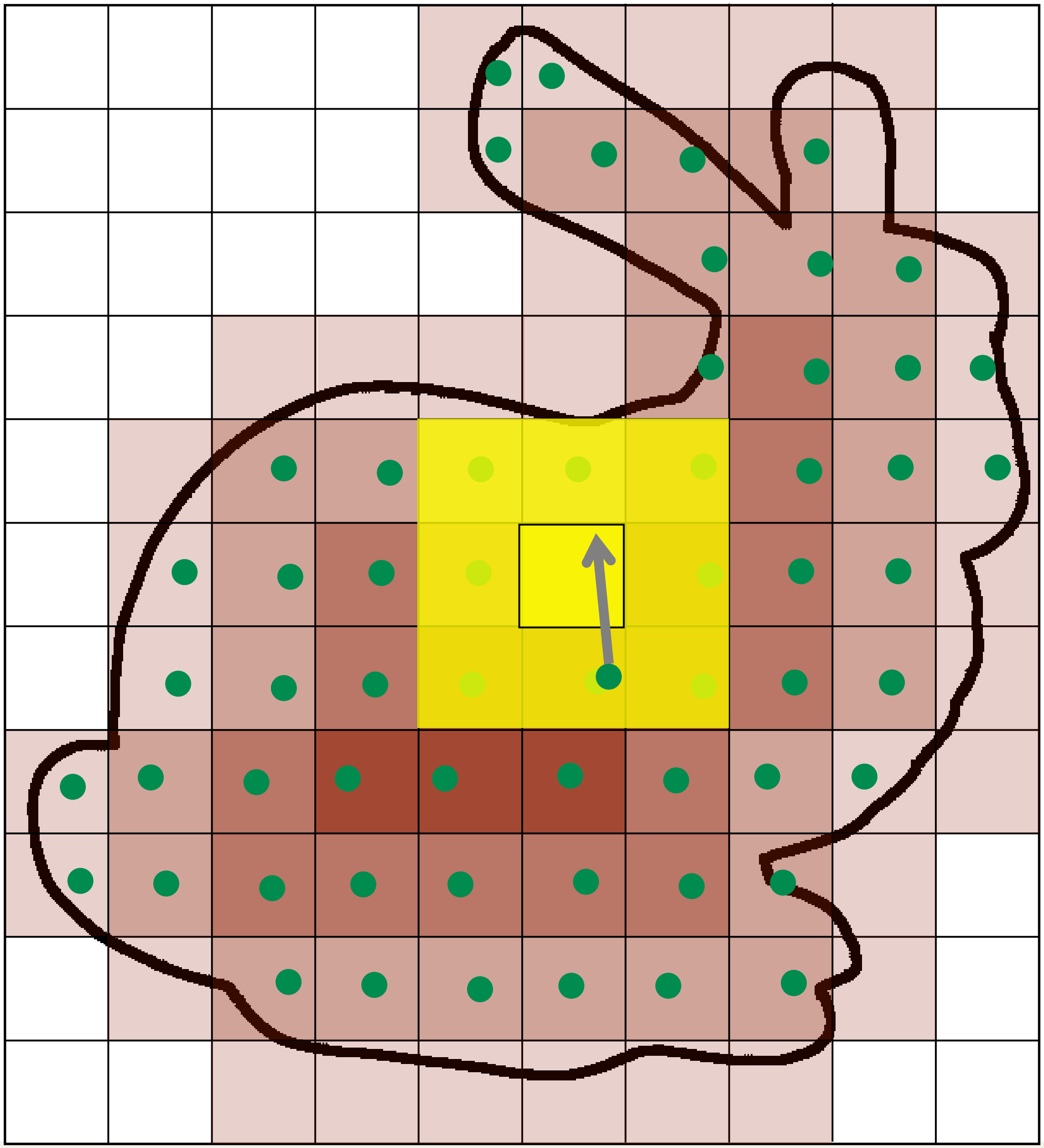
Parallelization



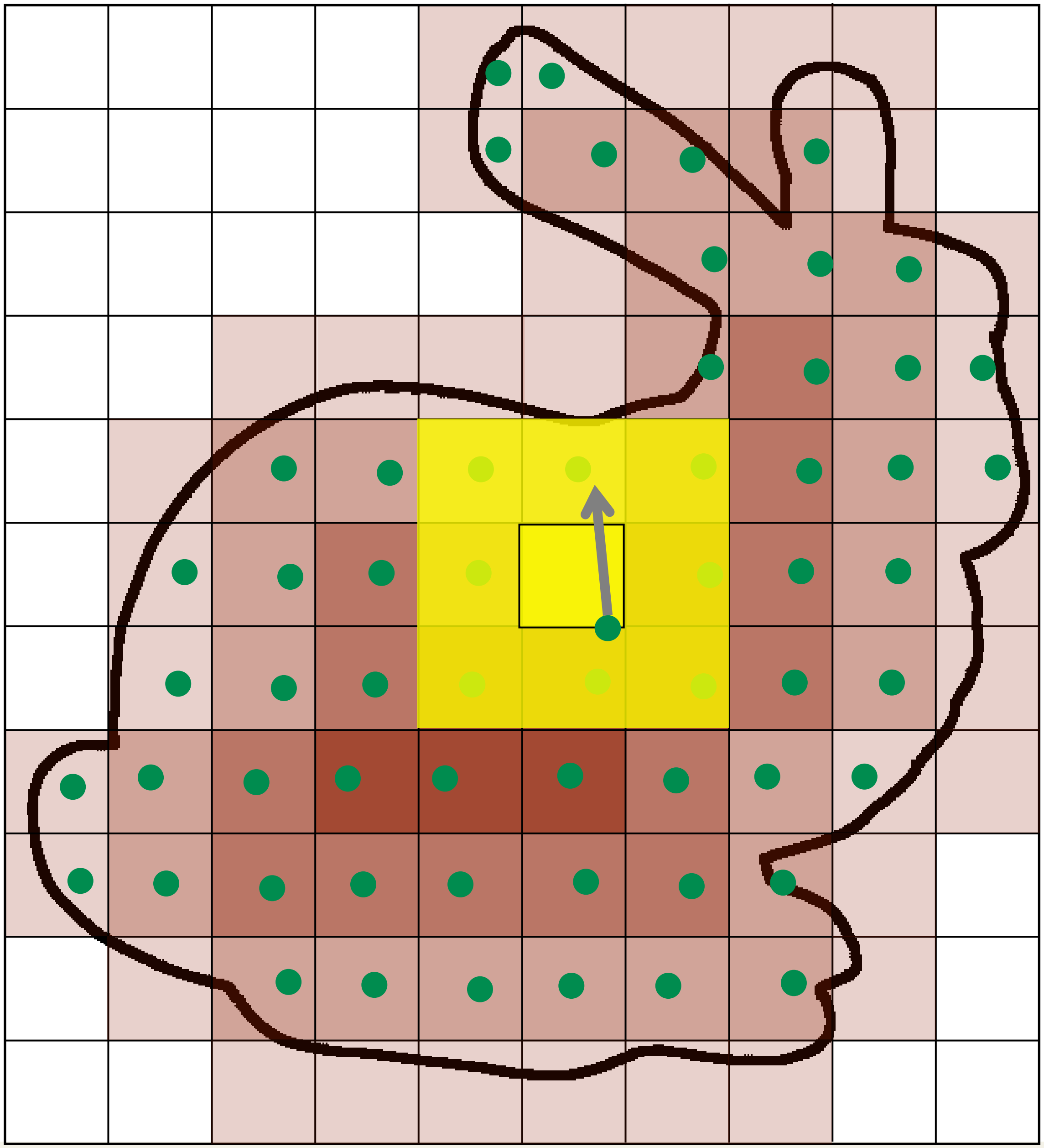
Parallelization



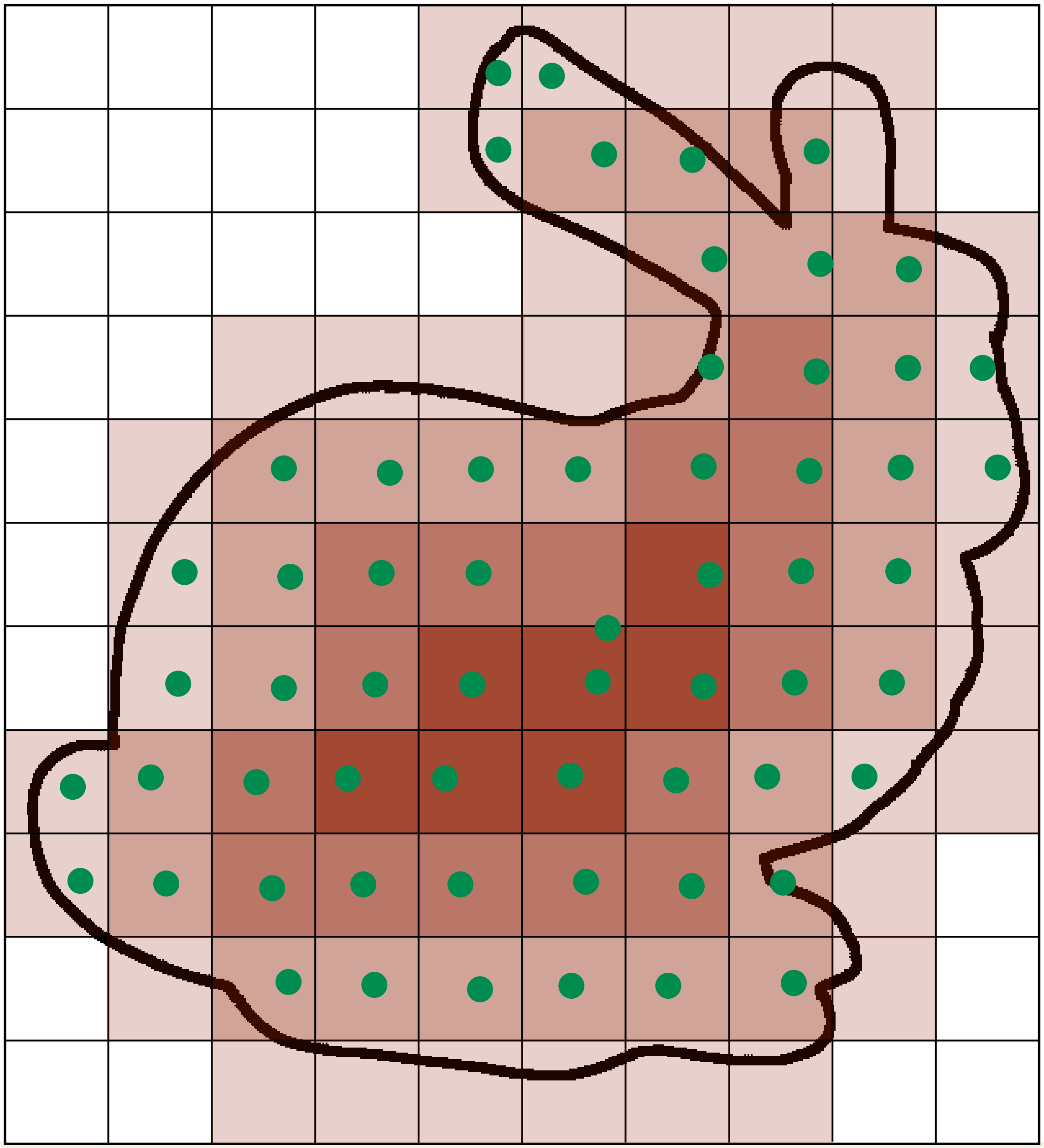
Parallelization



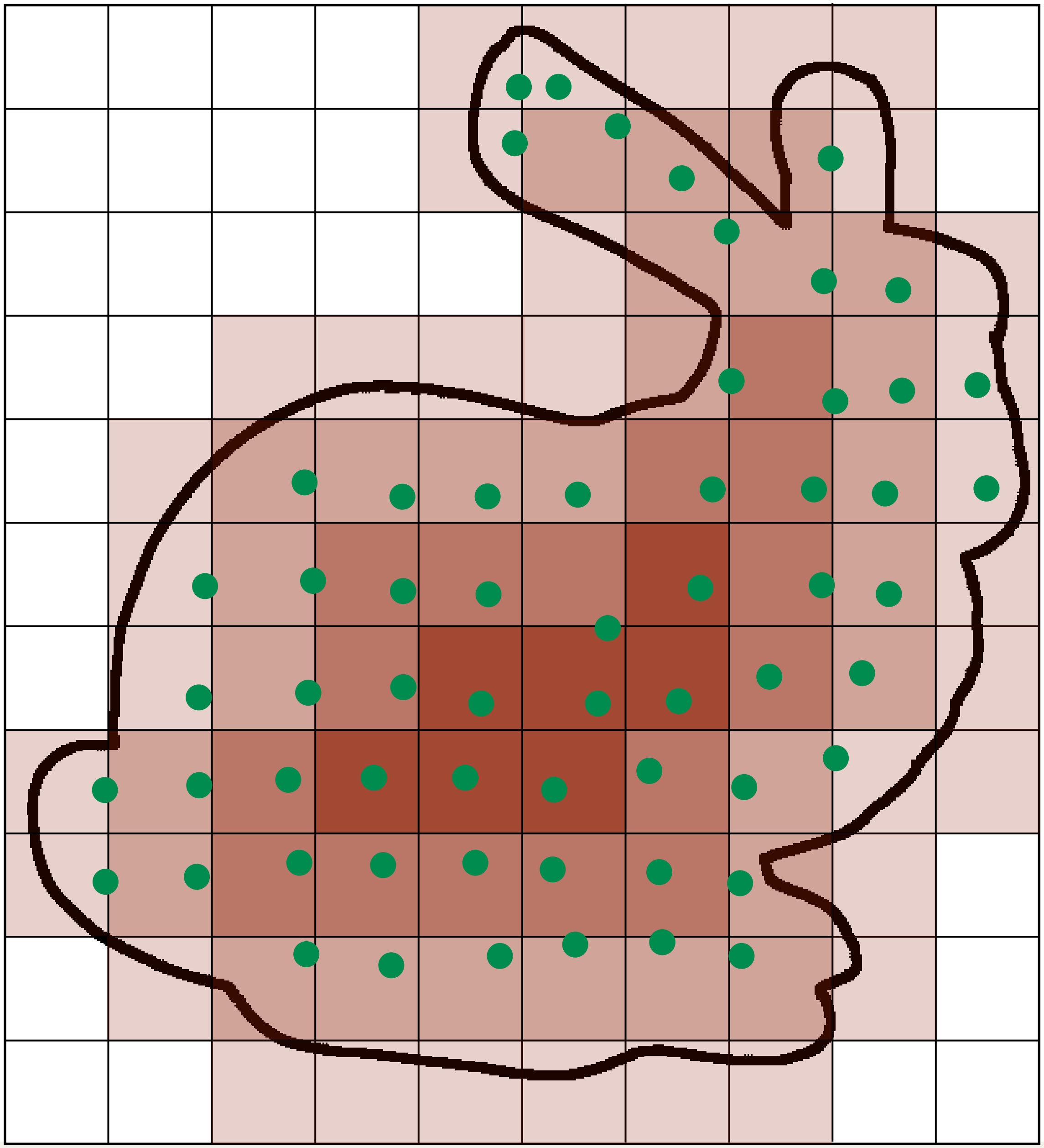
Parallelization



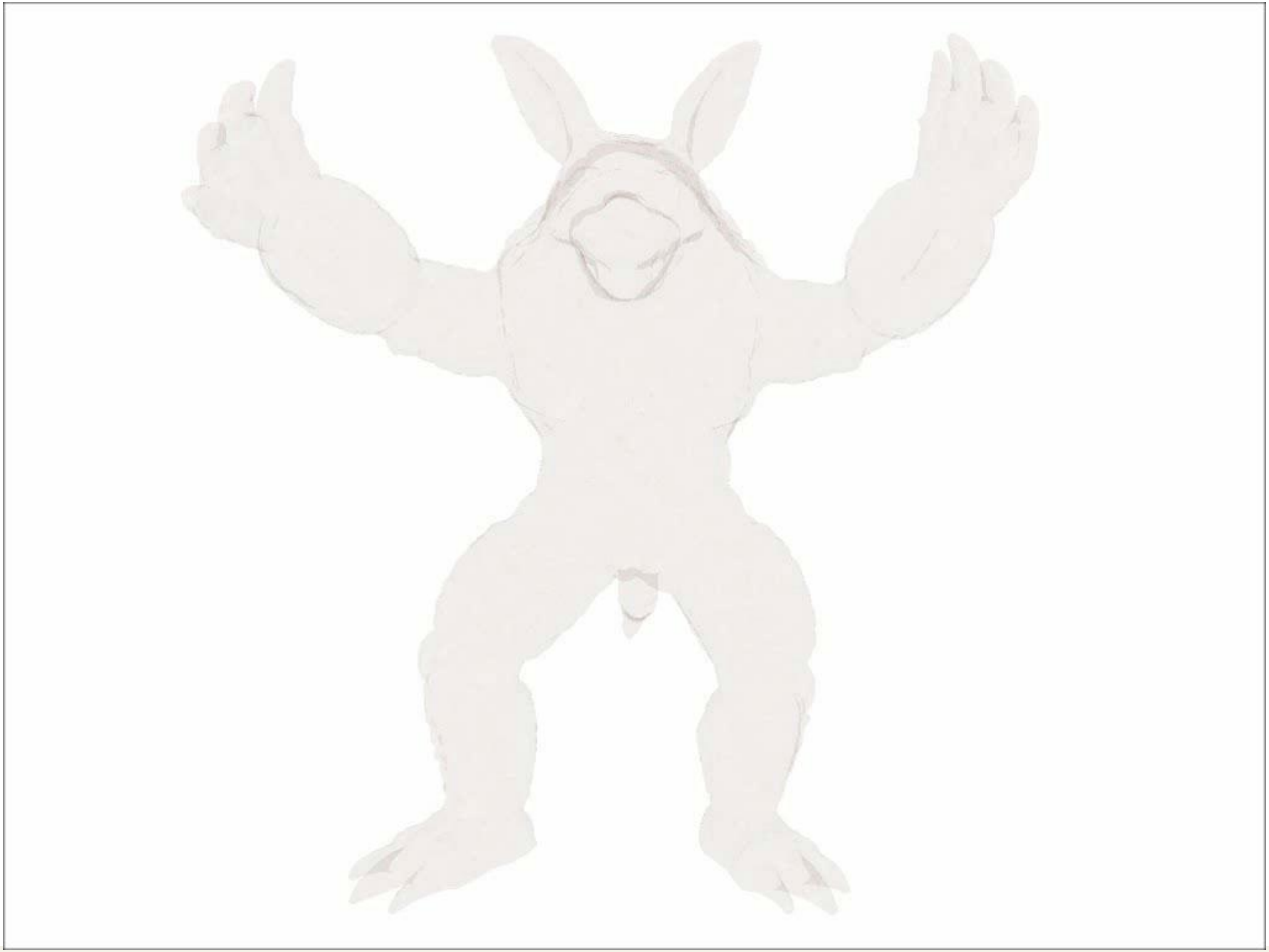
Parallelization



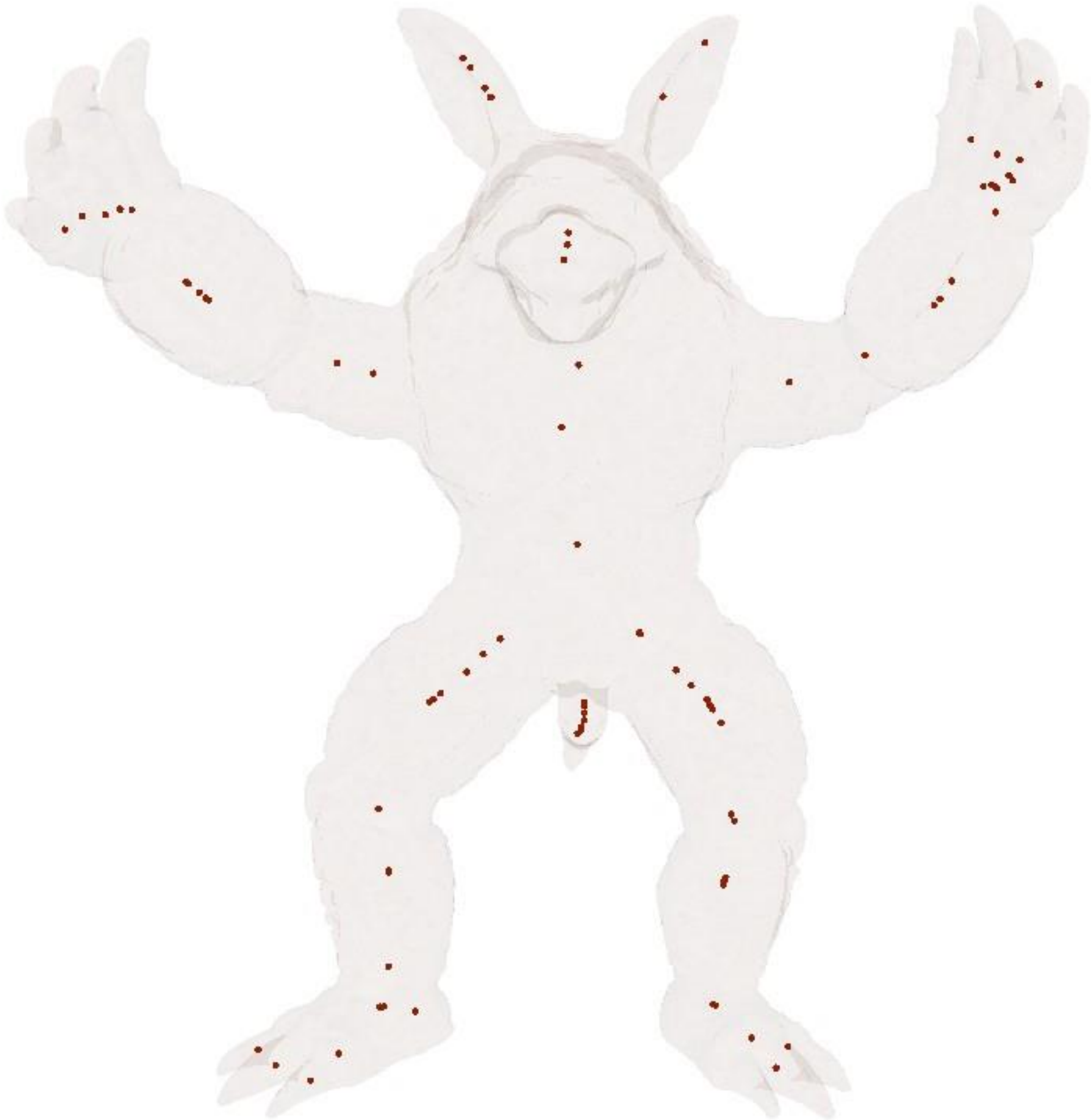
Parallelization



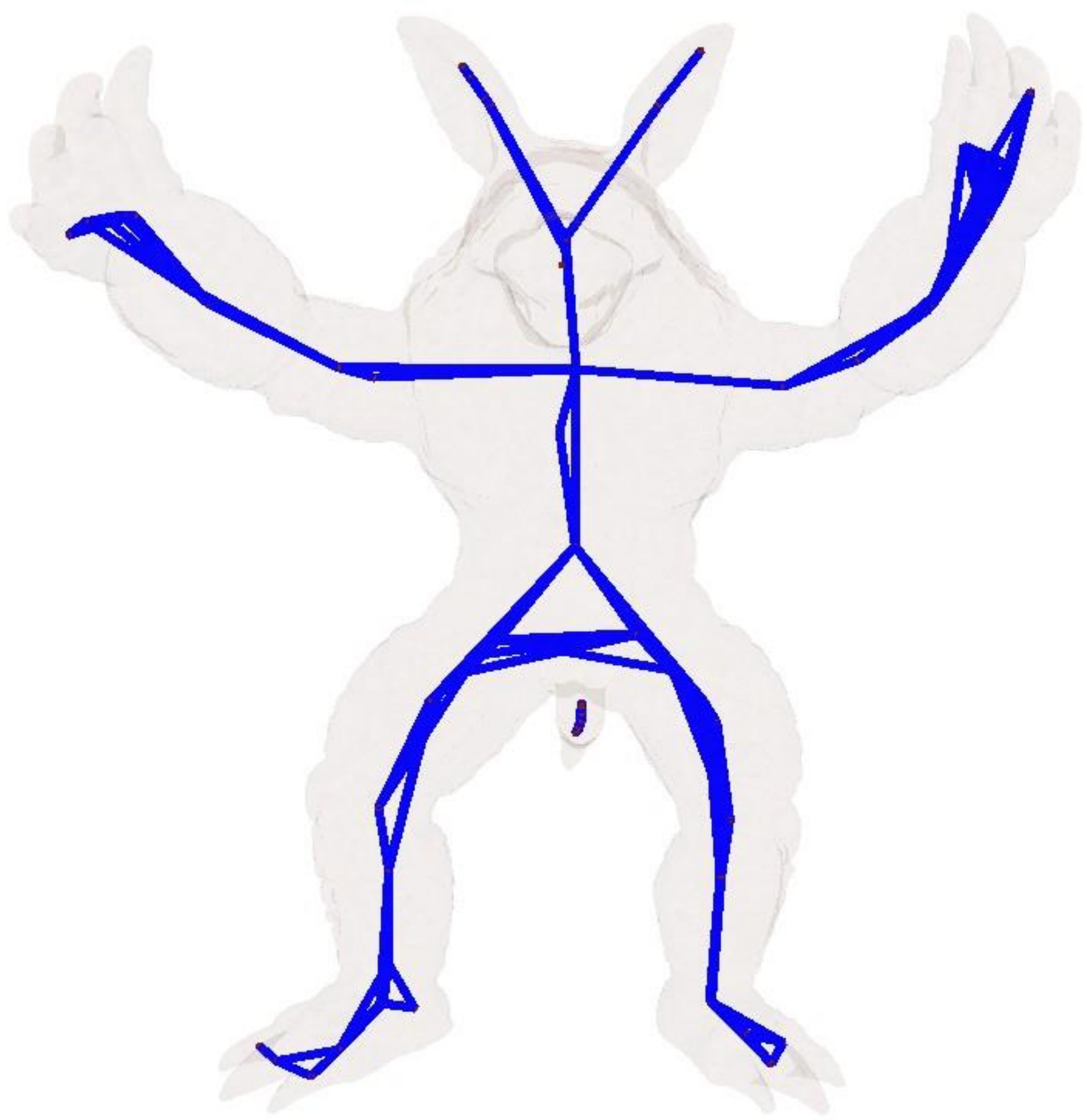
Parallel Protosphere



Parallel Protosphere

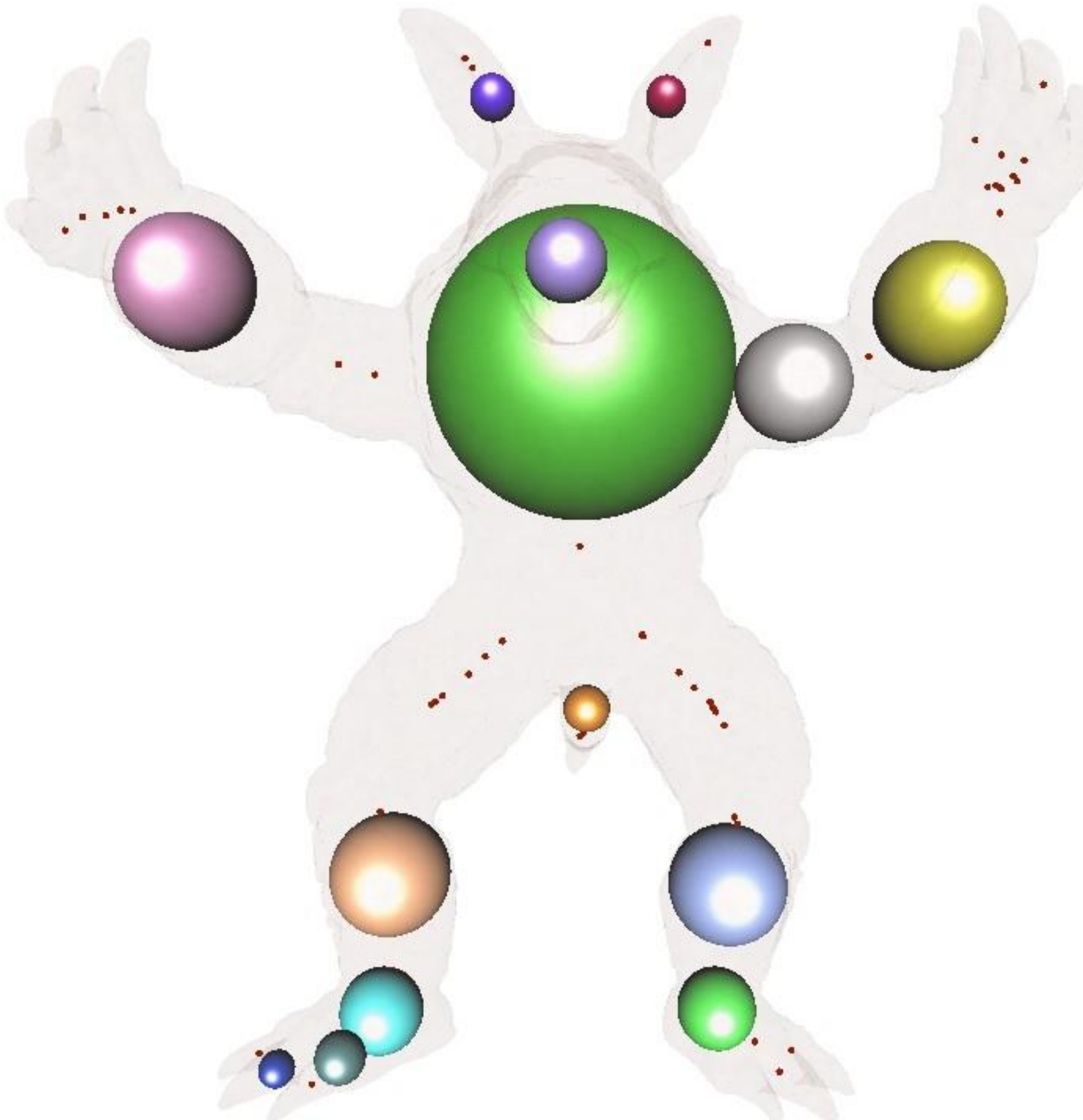


Parallel Protosphere

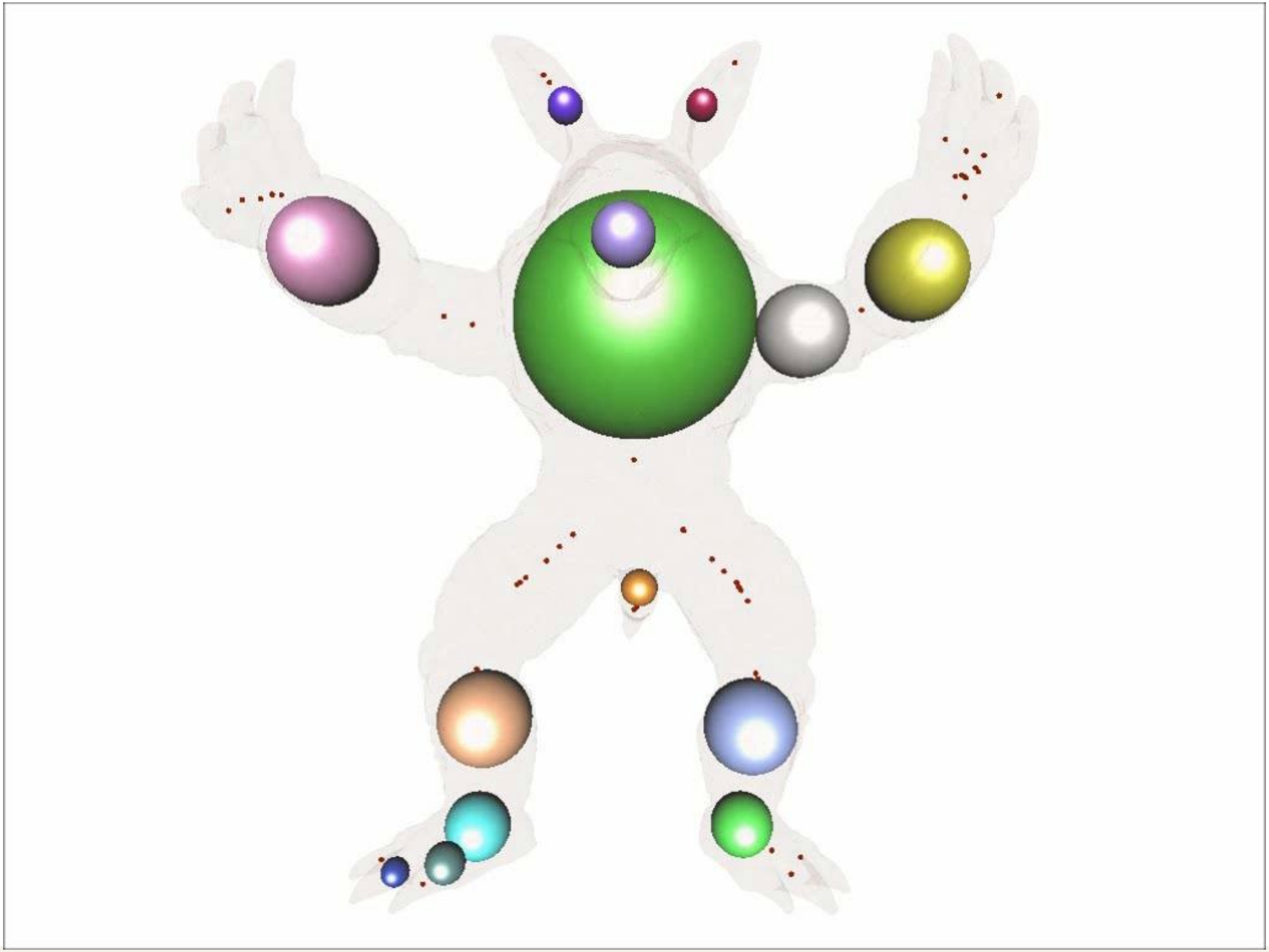


Parallel Protosphere

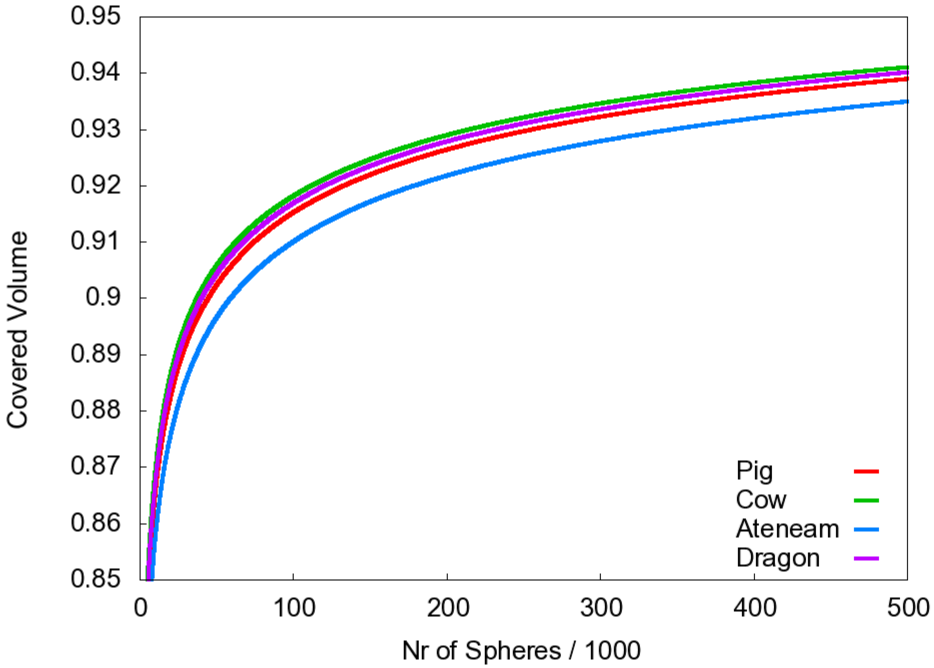
Parallel Protosphere

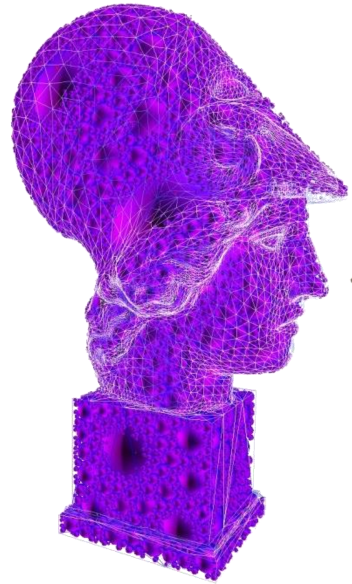


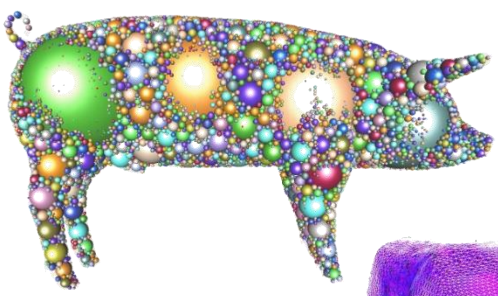
Parallel Protosphere

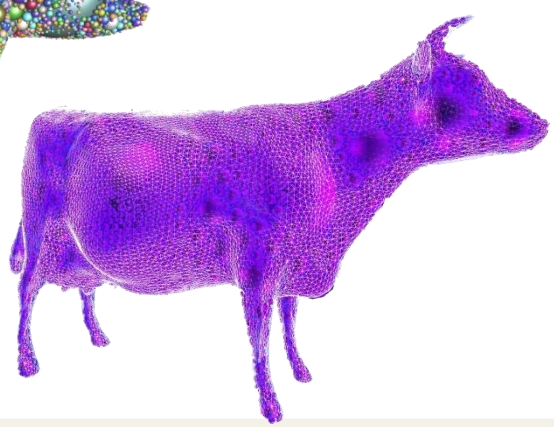


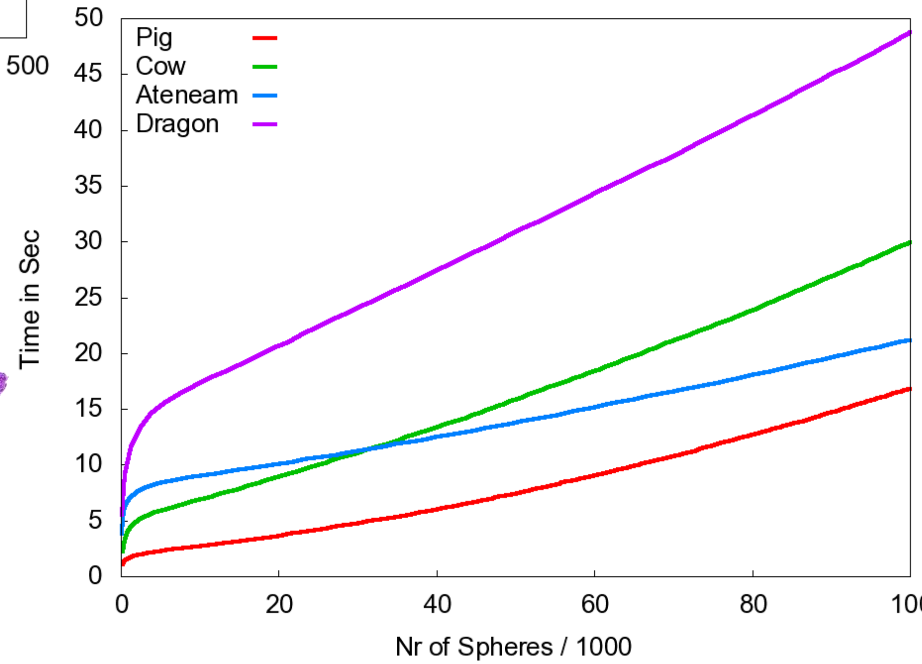
Results





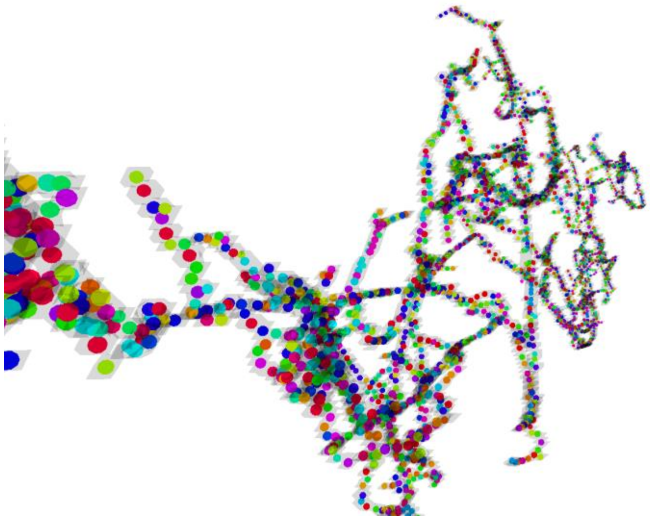






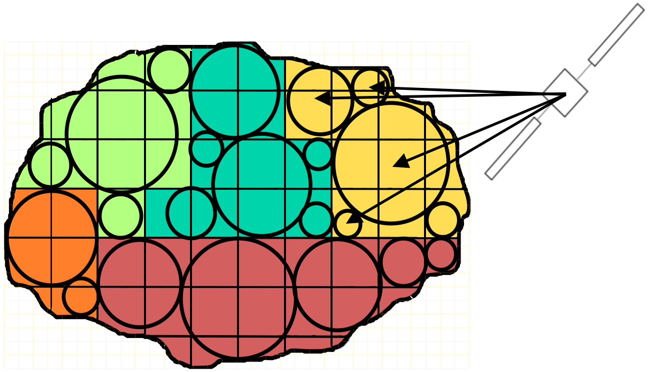
Nvidia Geforce GTX 480

Further Applications of our Sphere Packings

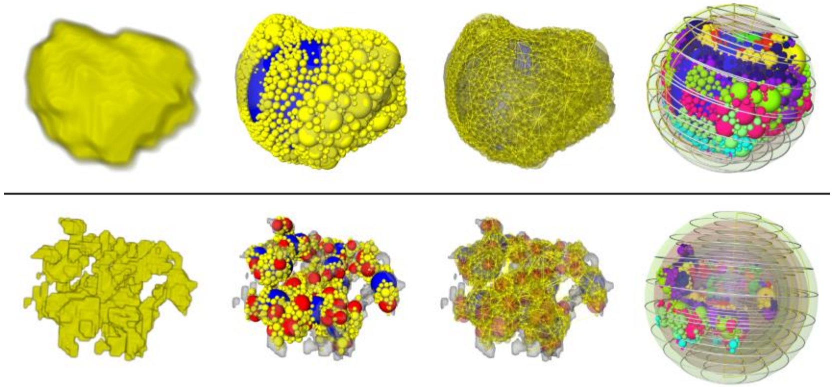


Material Science

3D Printing

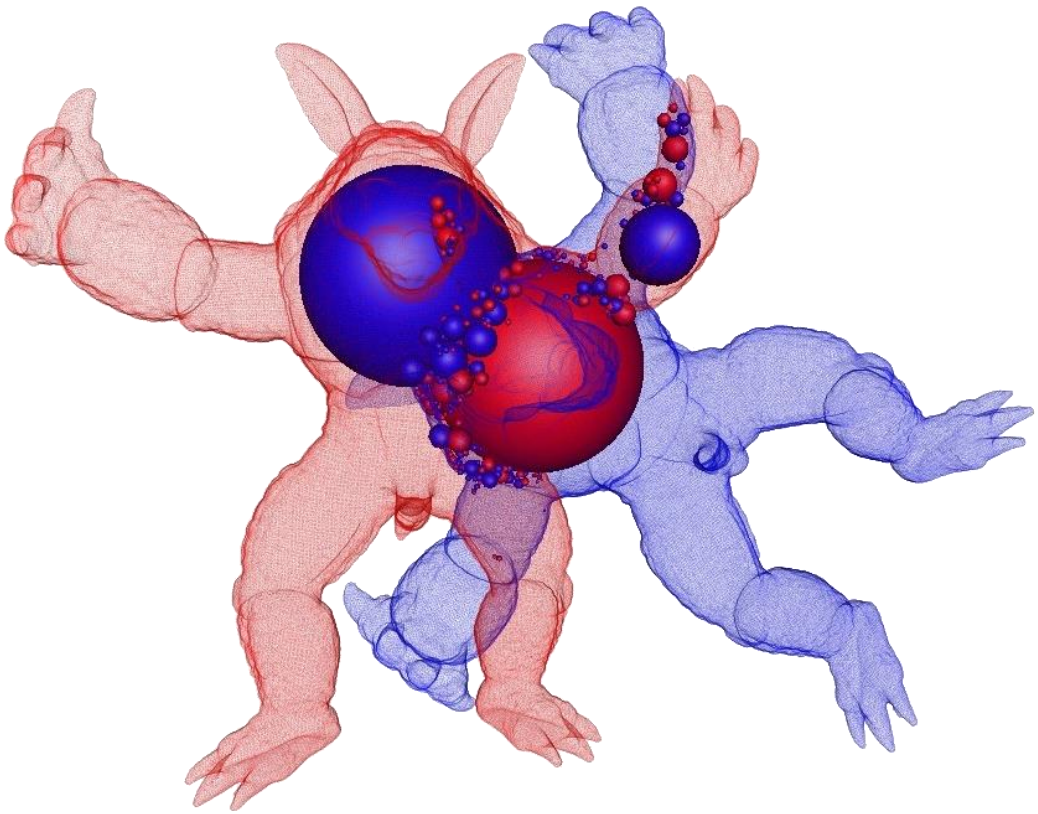


Gravitational Field of Asteroids

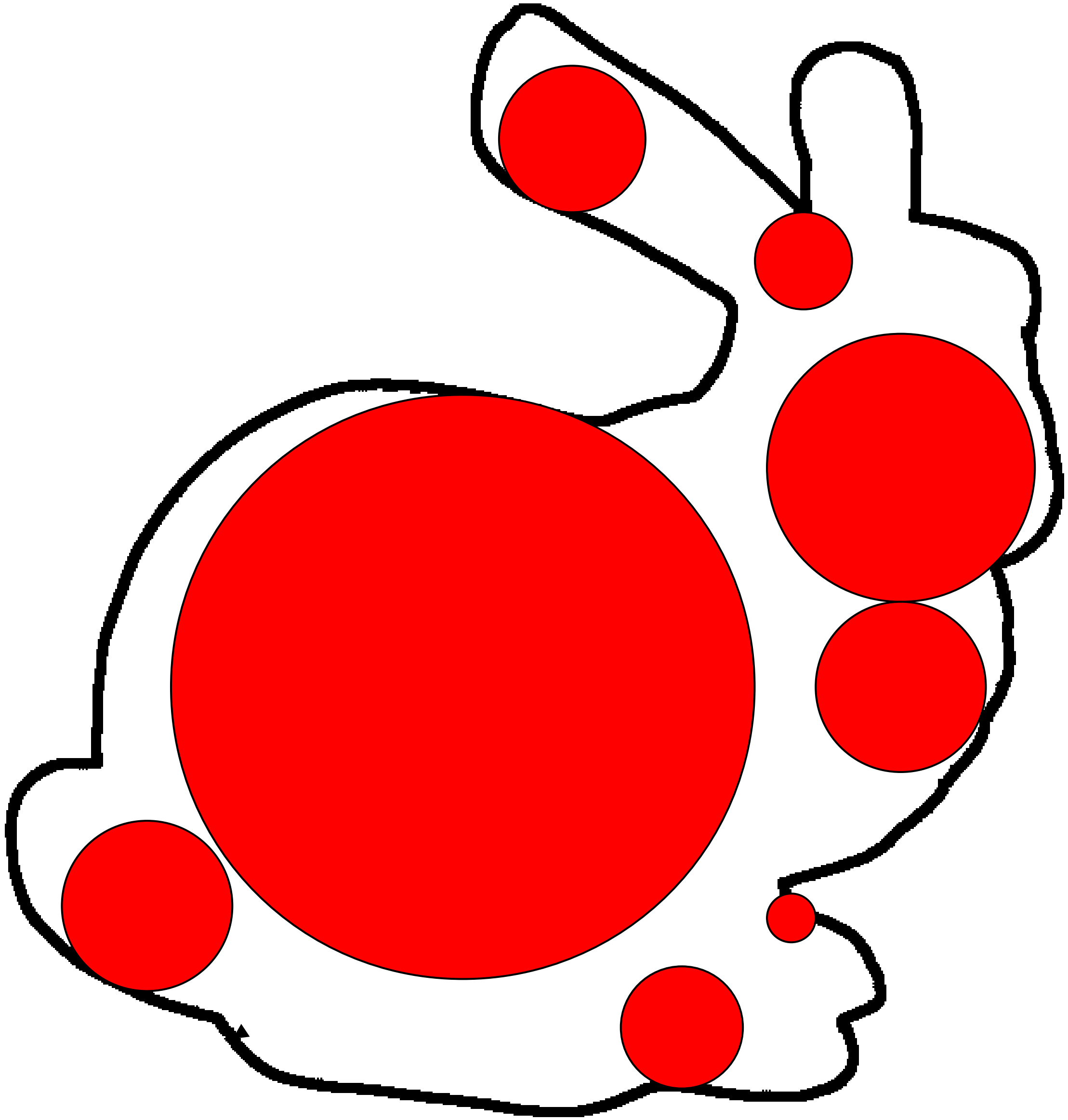


Classification of Carcinoms

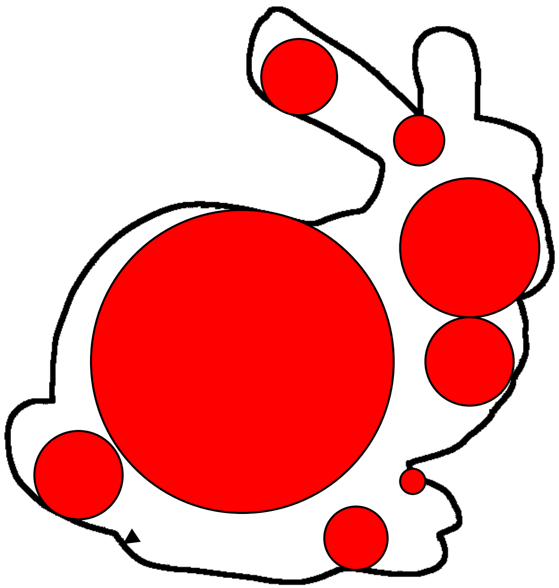
Back to Collision Detection



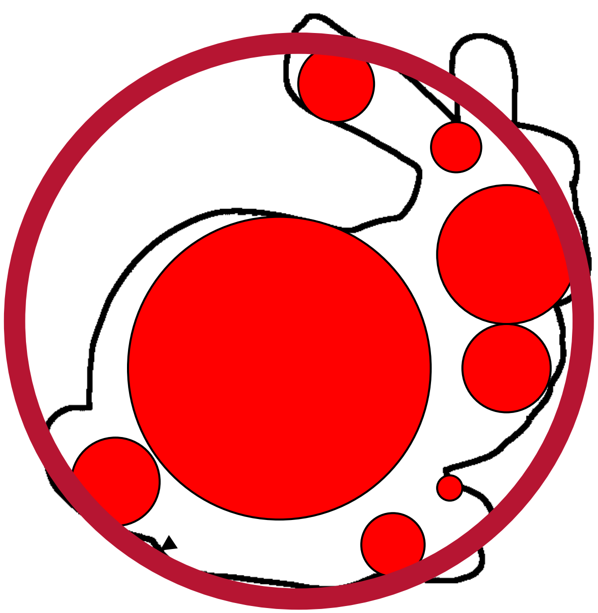
Hierachy Creation



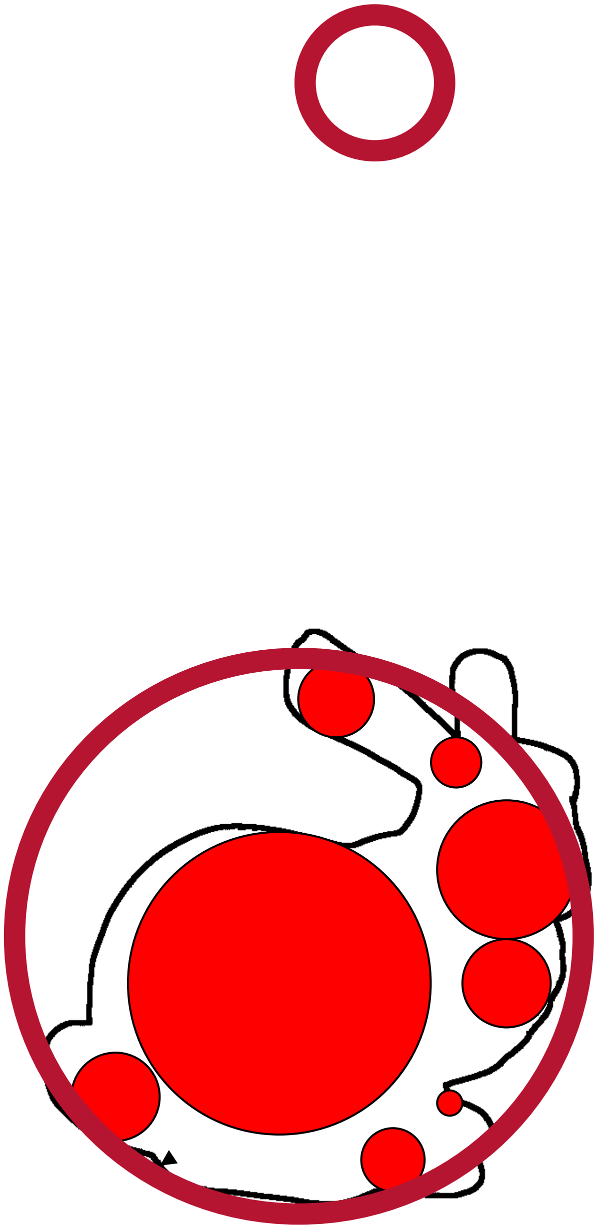
Hierachy Creation



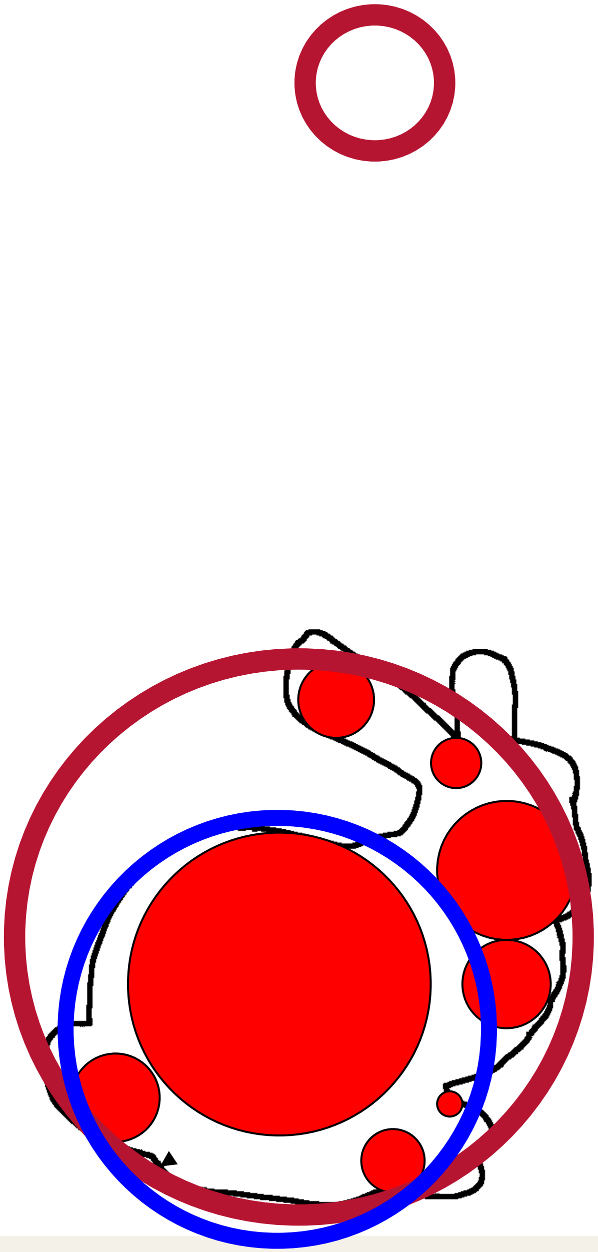
Hierachy Creation



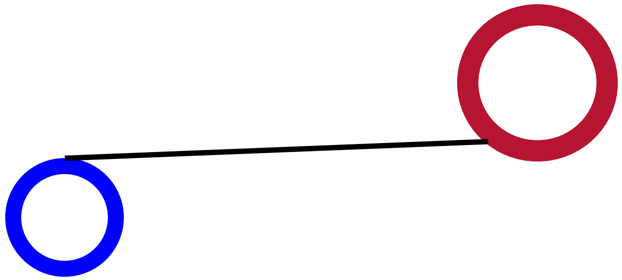
Hierachy Creation

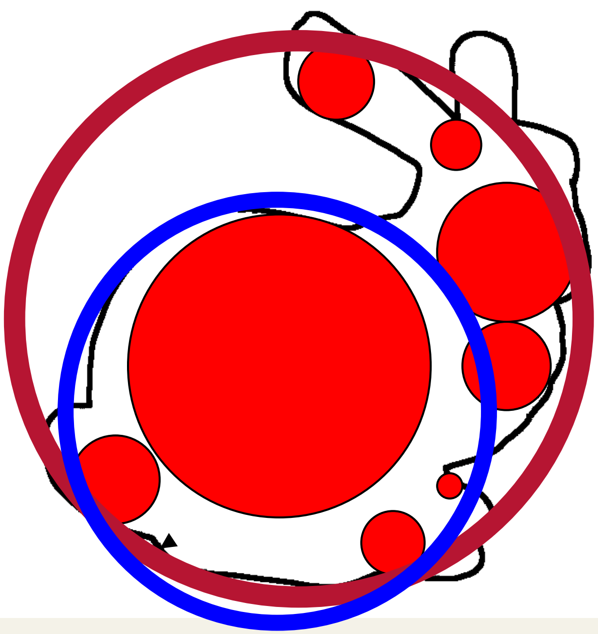


Hierachy Creation

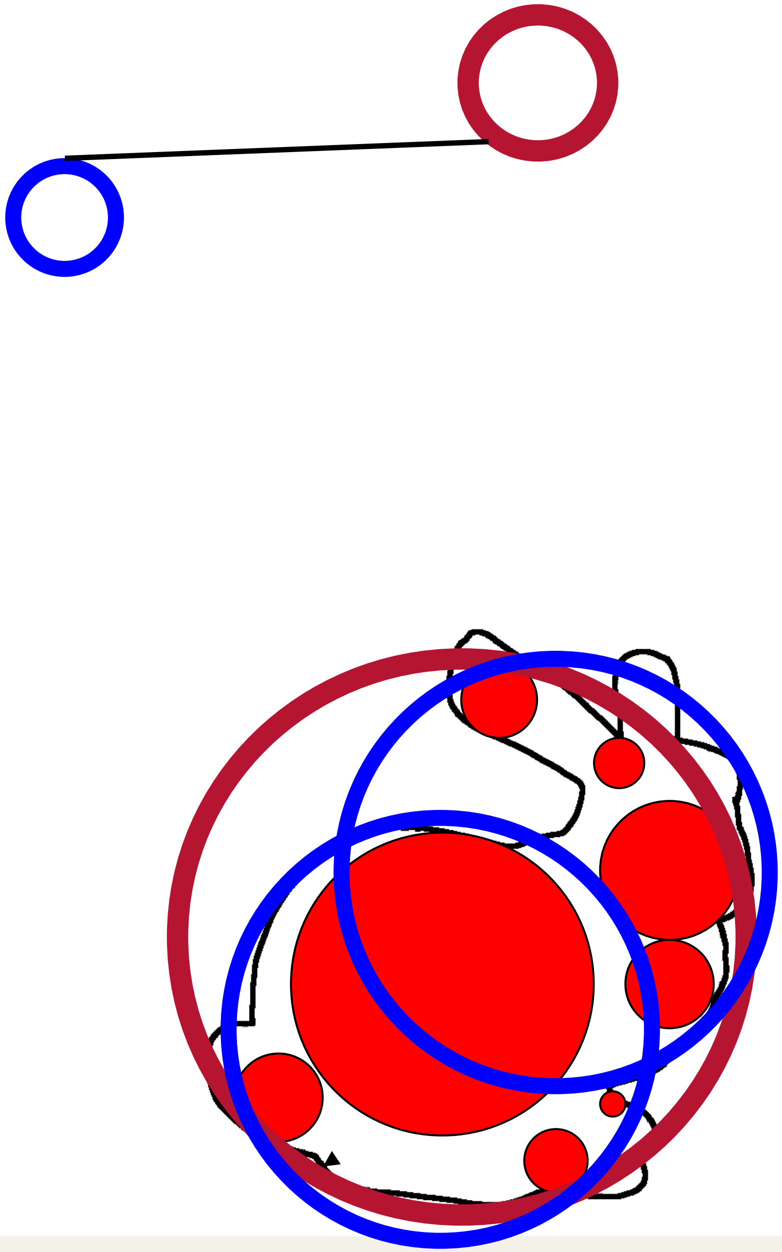


Hierachy Creation

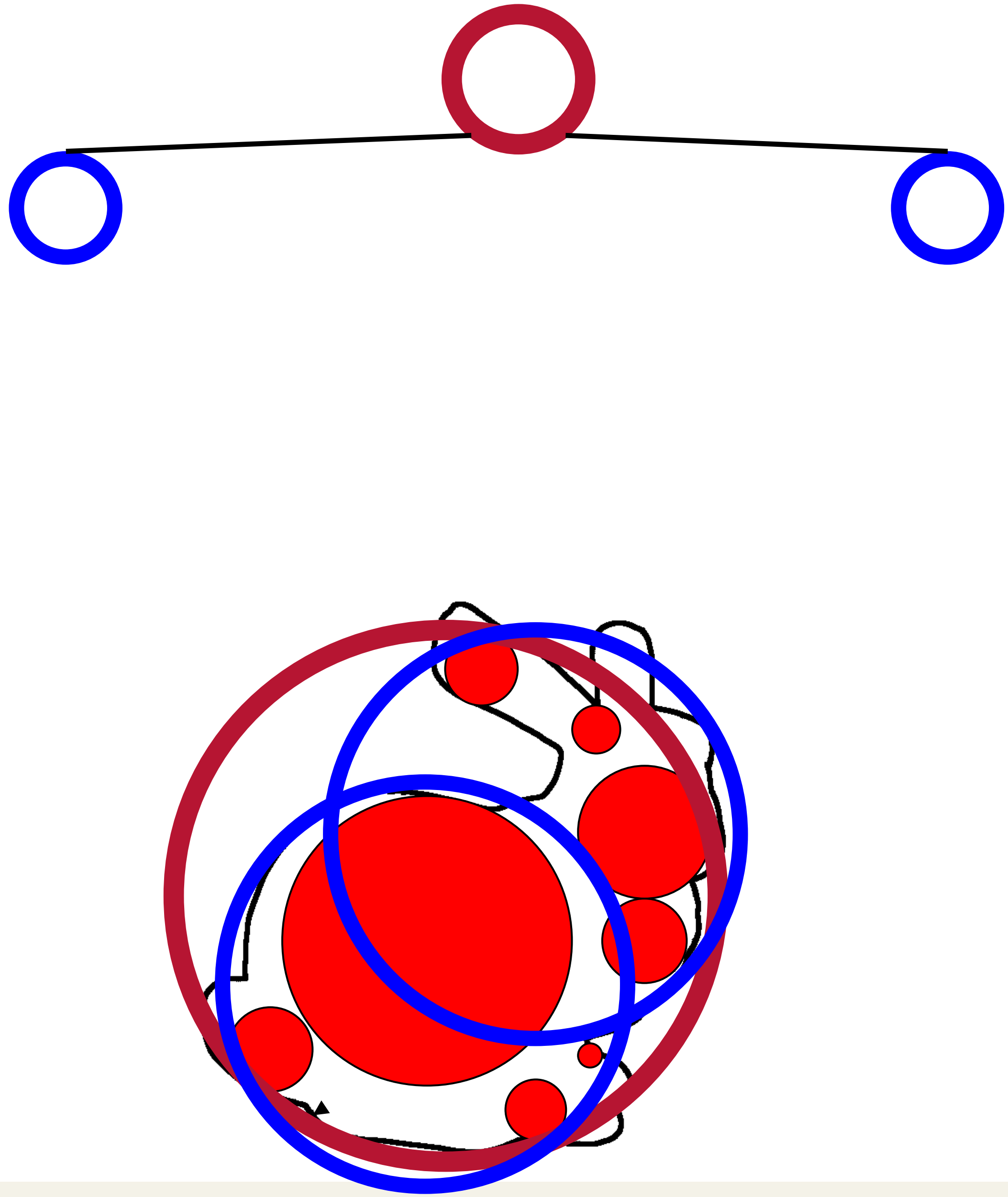




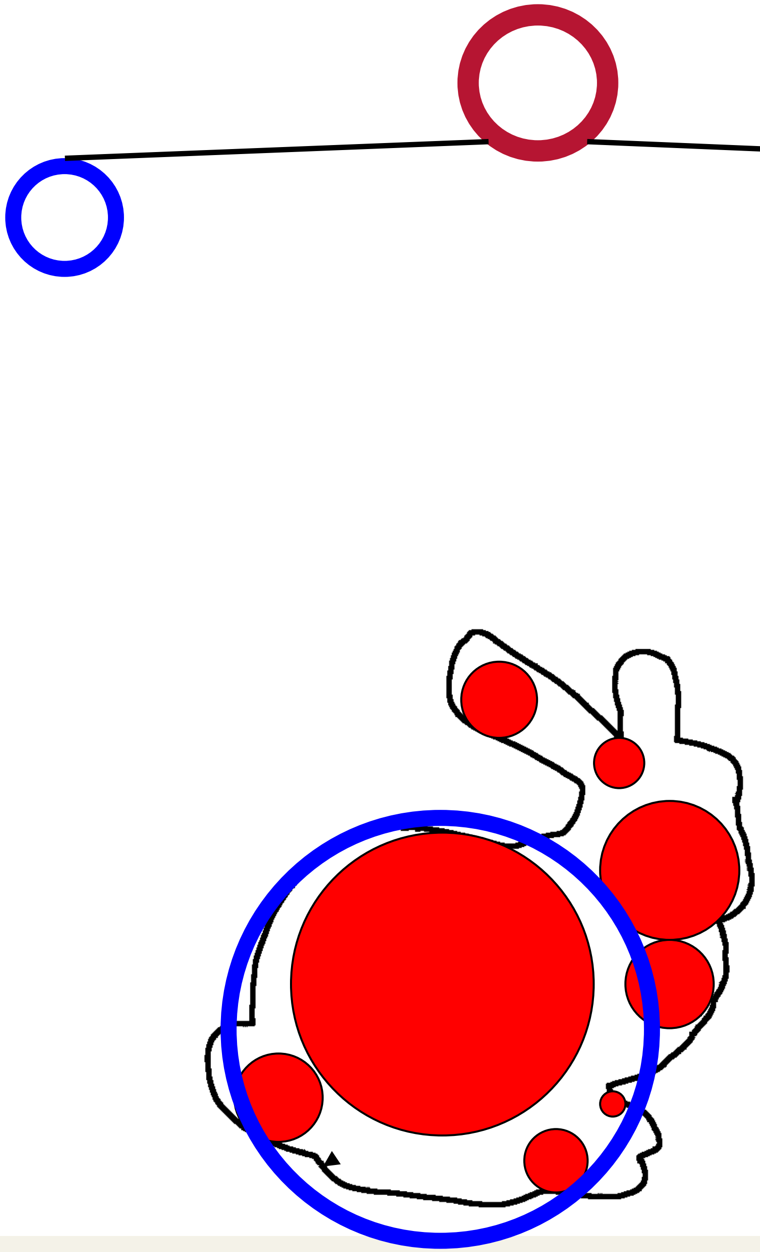
Hierachy Creation



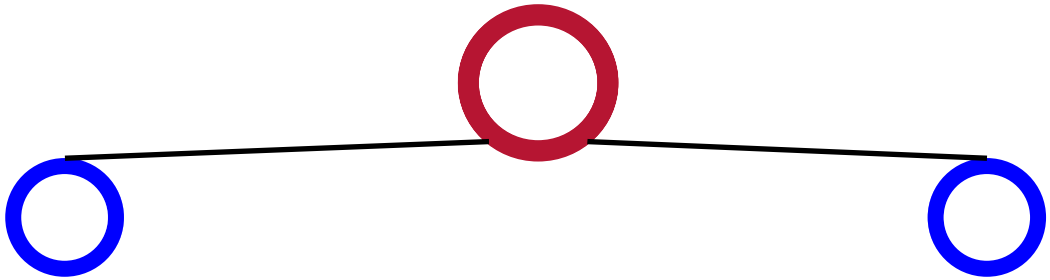
Hierachy Creation

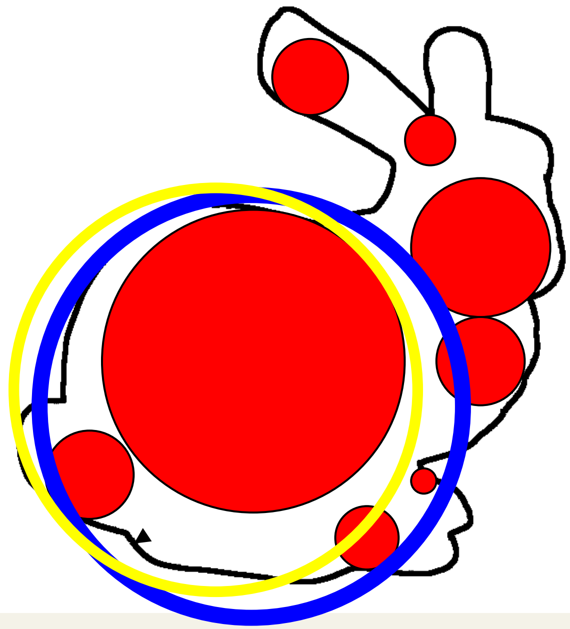


Hierachy Creation

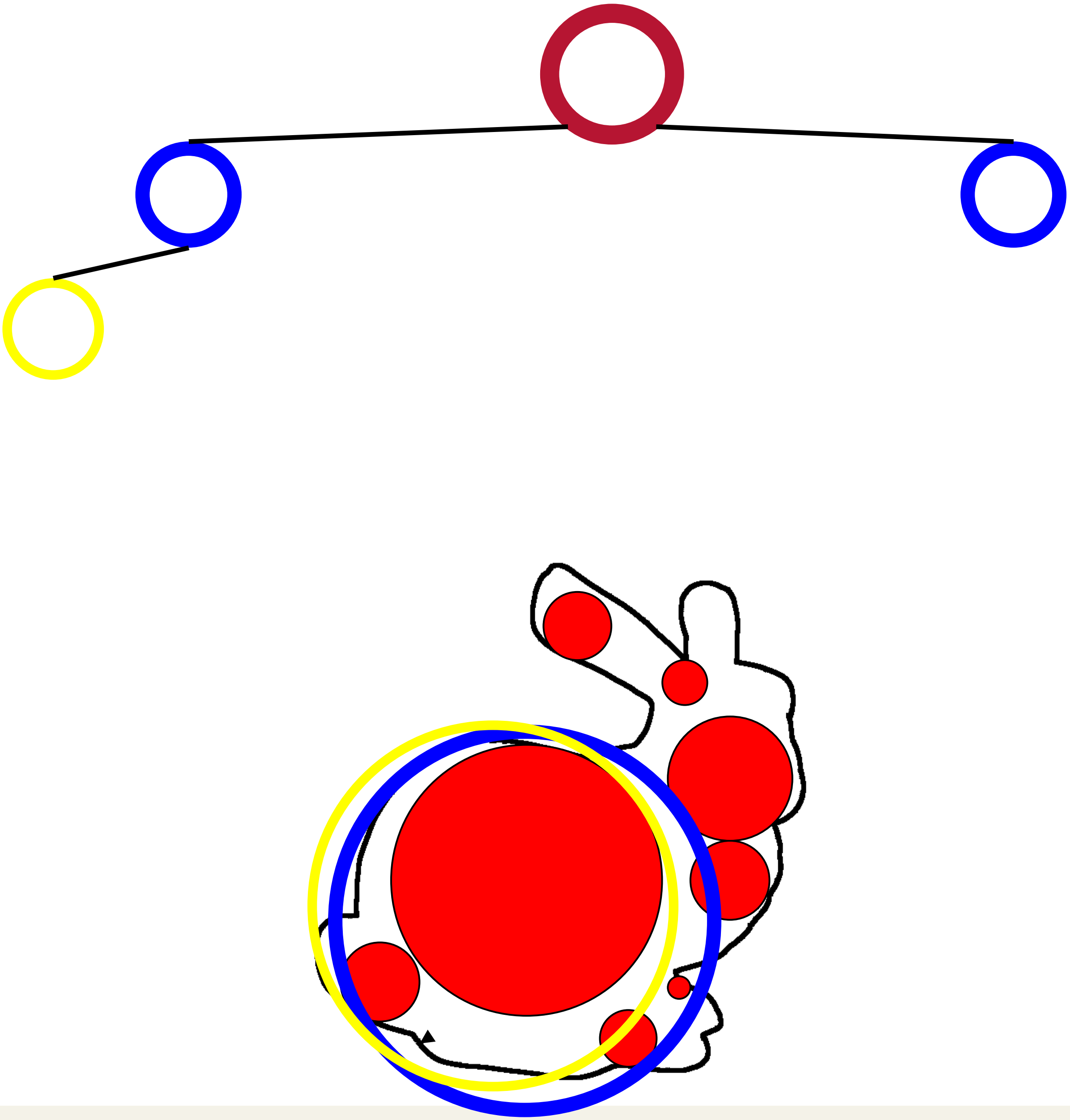


Hierachy Creation

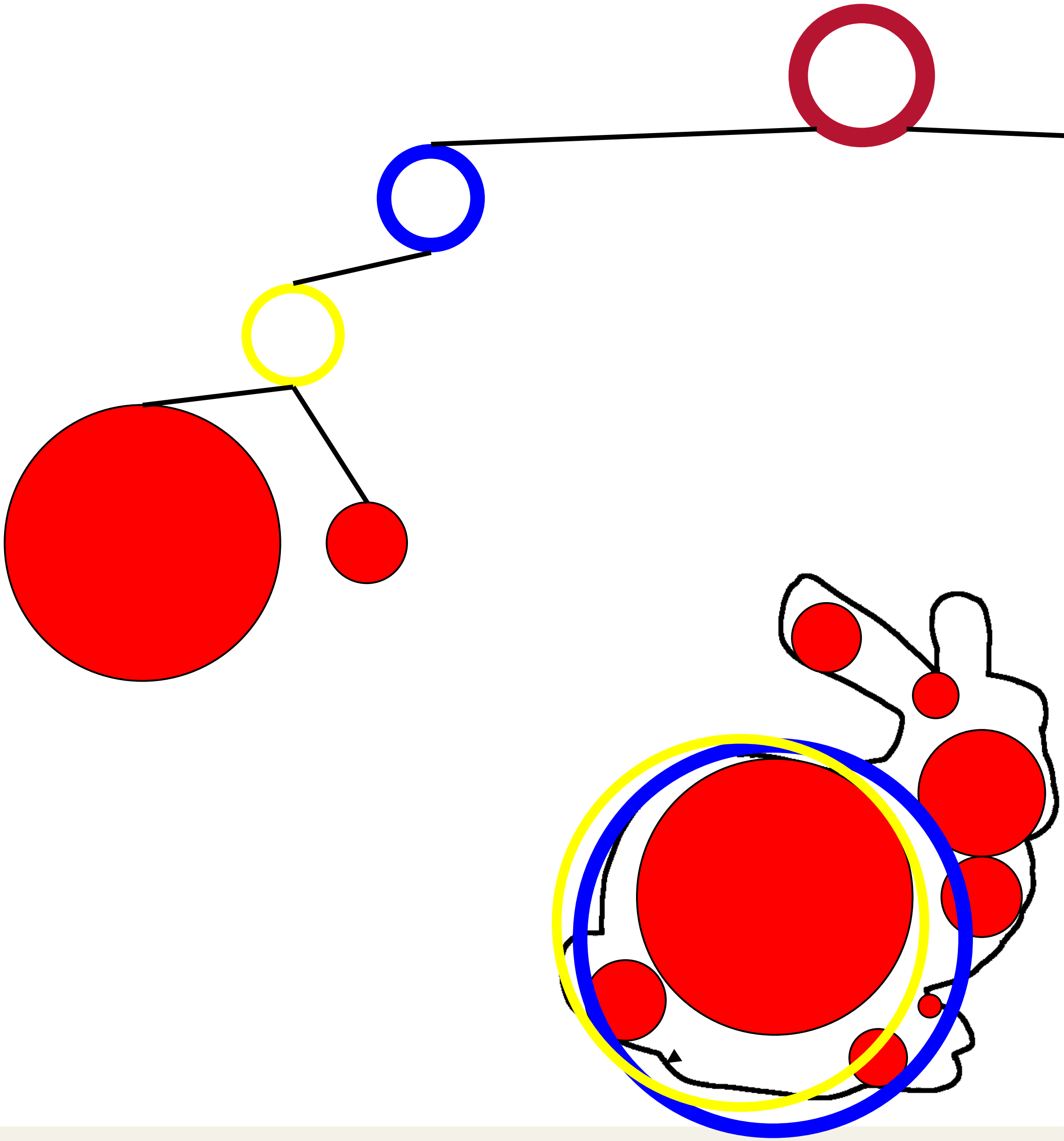




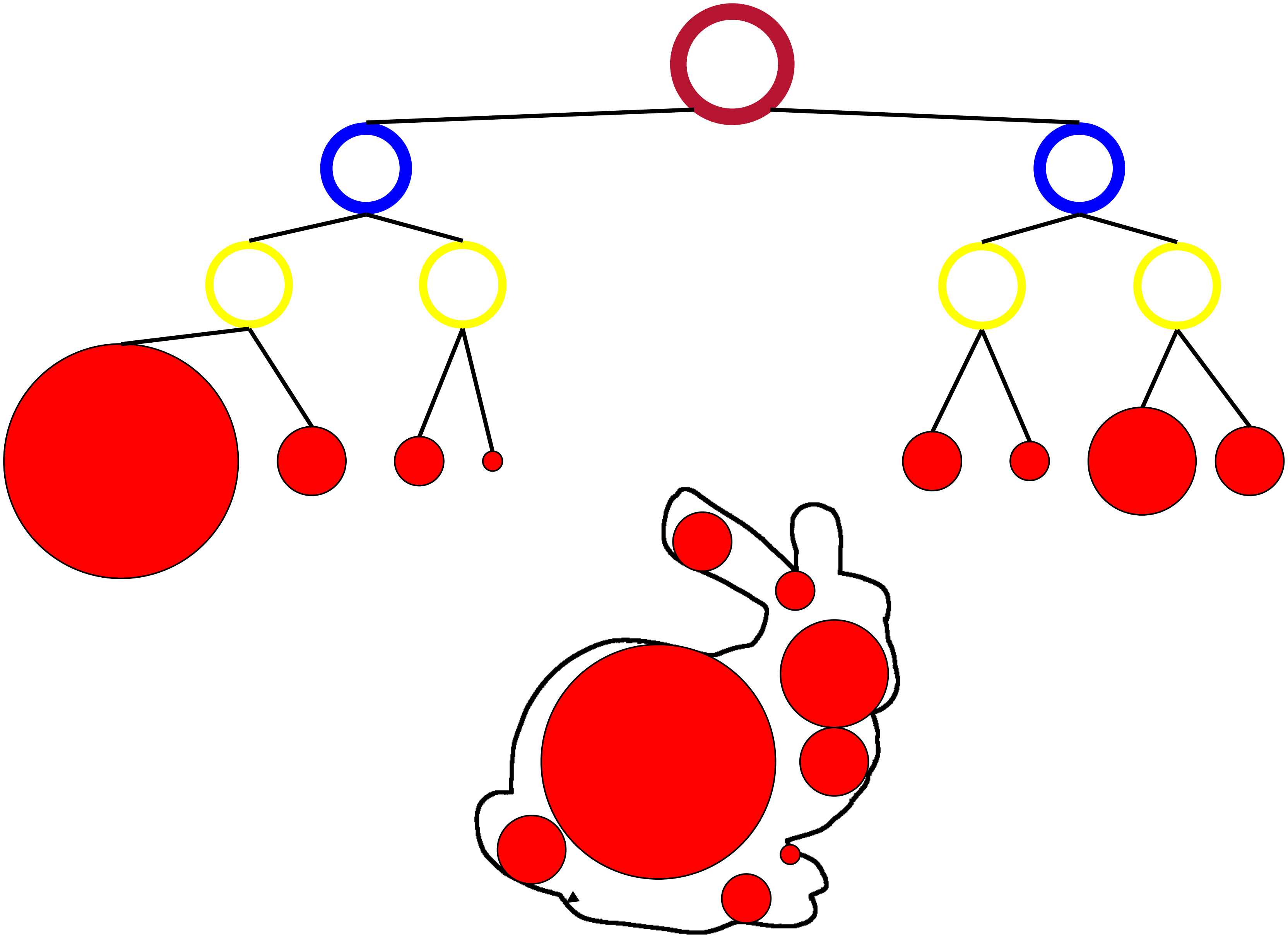
Hierachy Creation



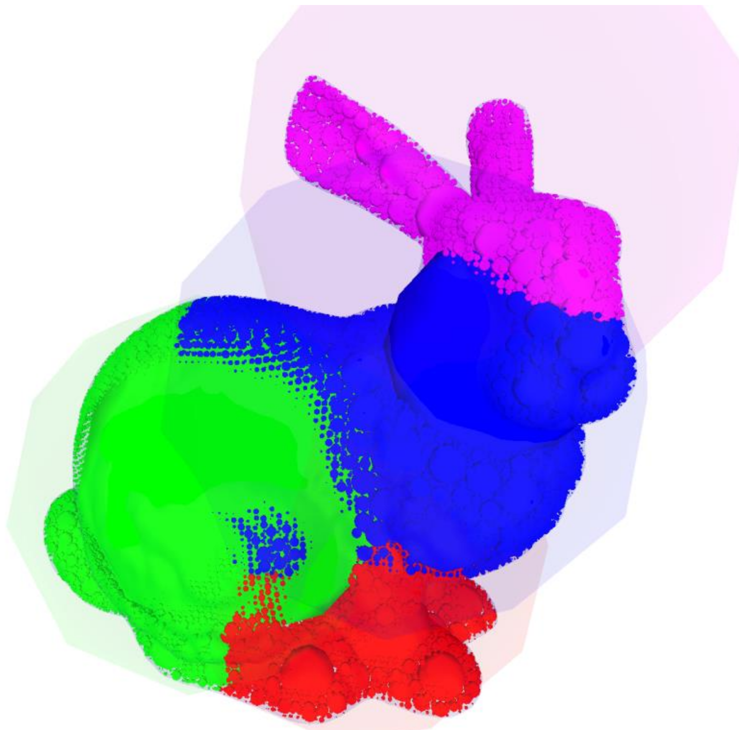
Hierachy Creation

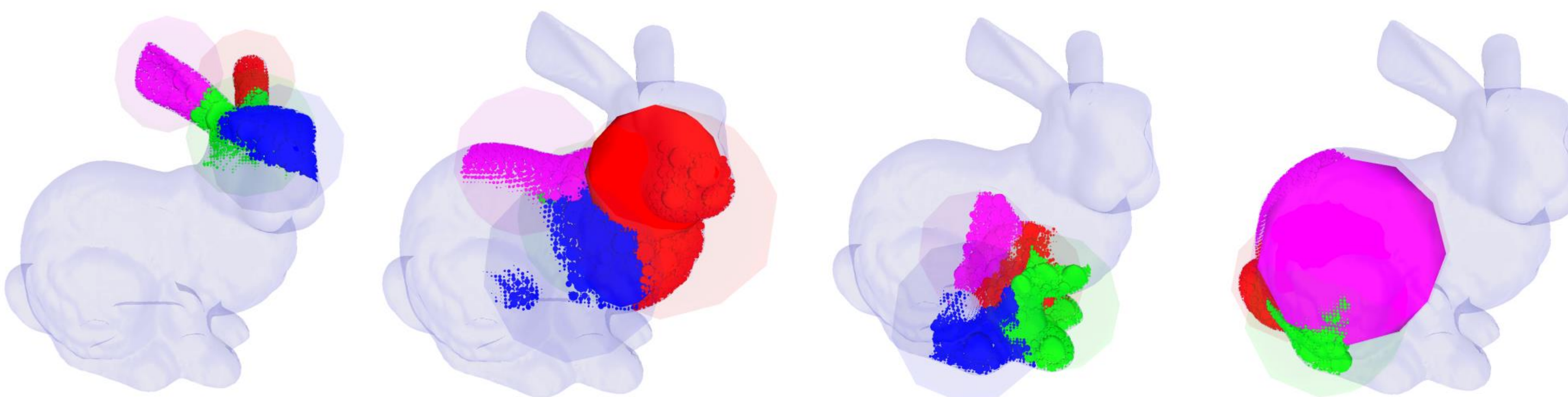


Hierachy Creation

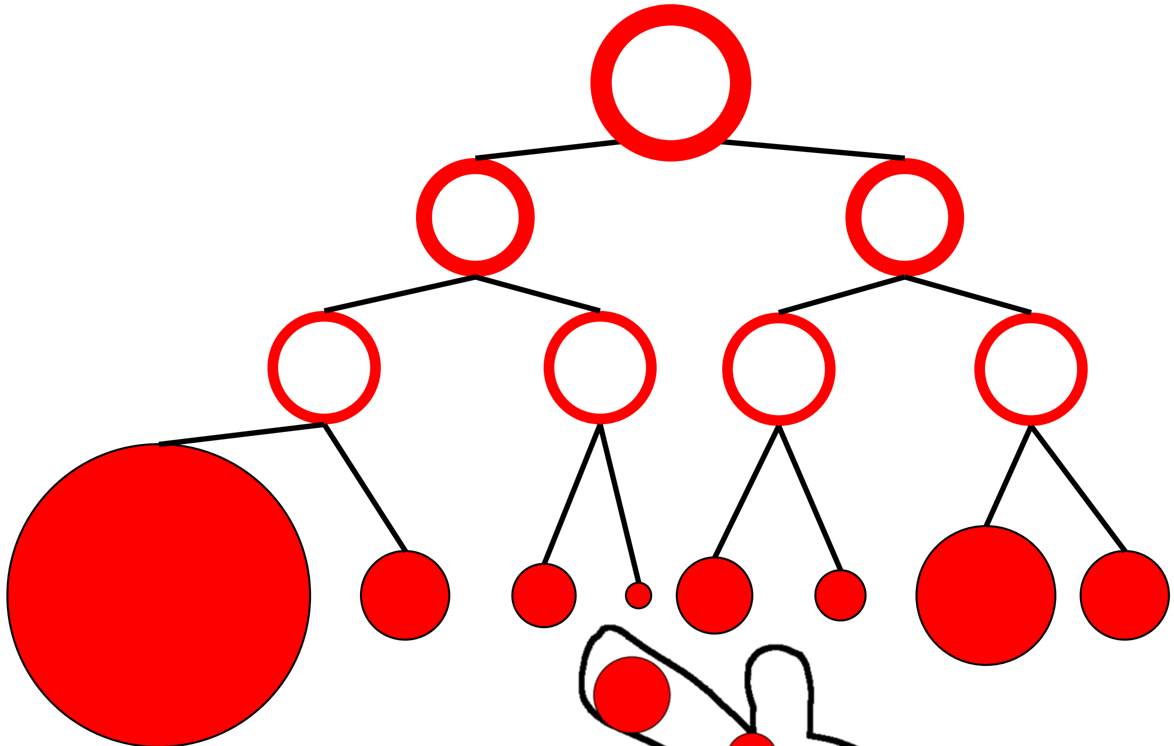


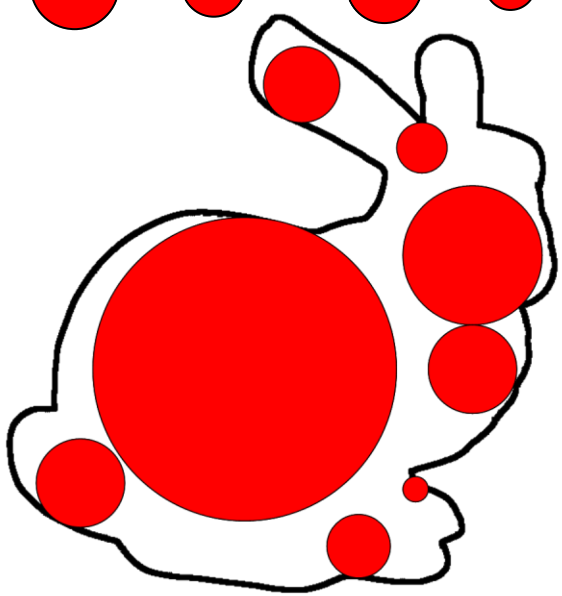
Bounding Volume Hierarchy for Inner Spheres

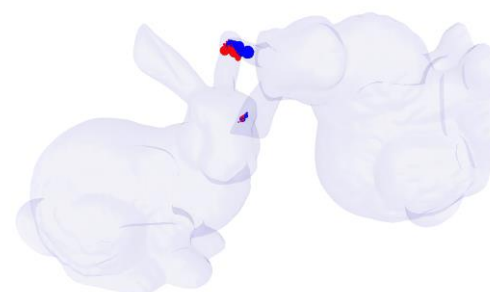




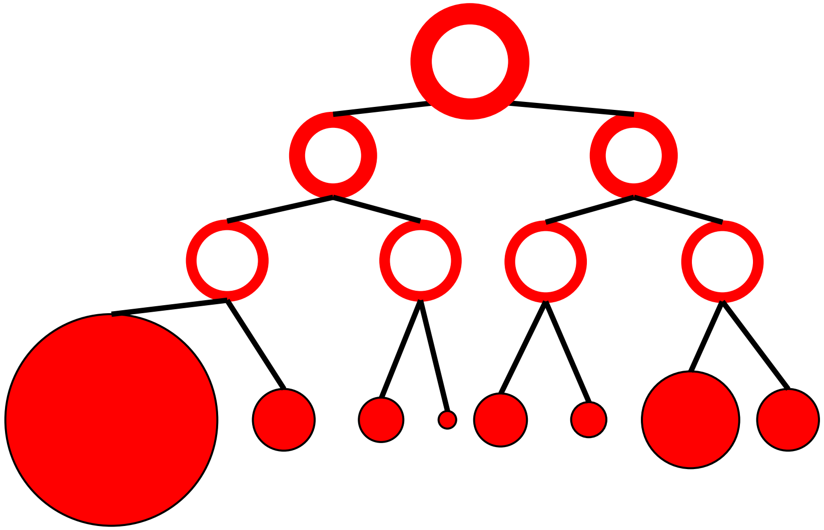
BVH Traversal: Penetration Volume Queries

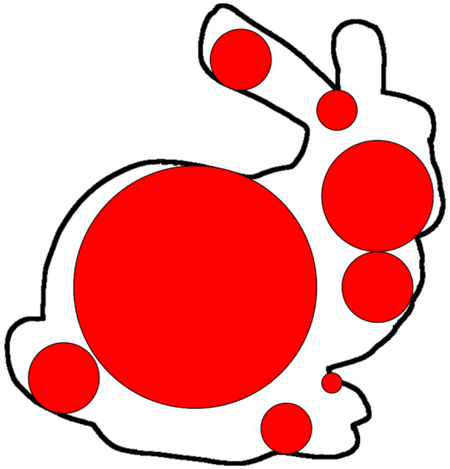


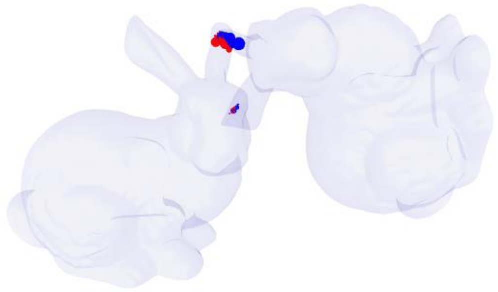




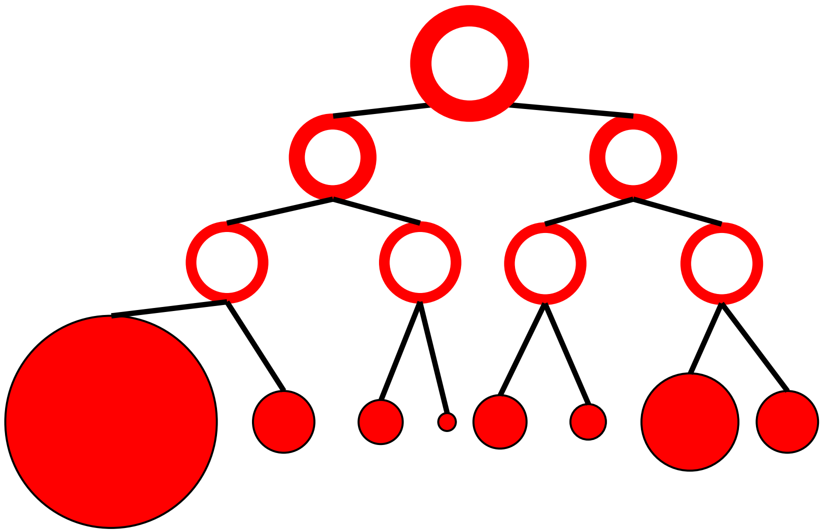
BVH Traversal: Penetration Volume Queries

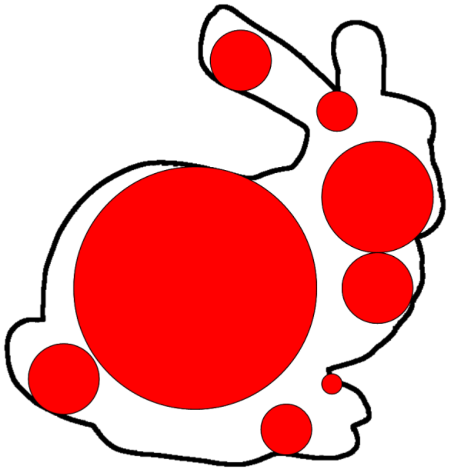


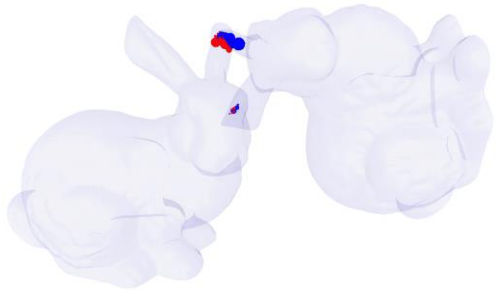




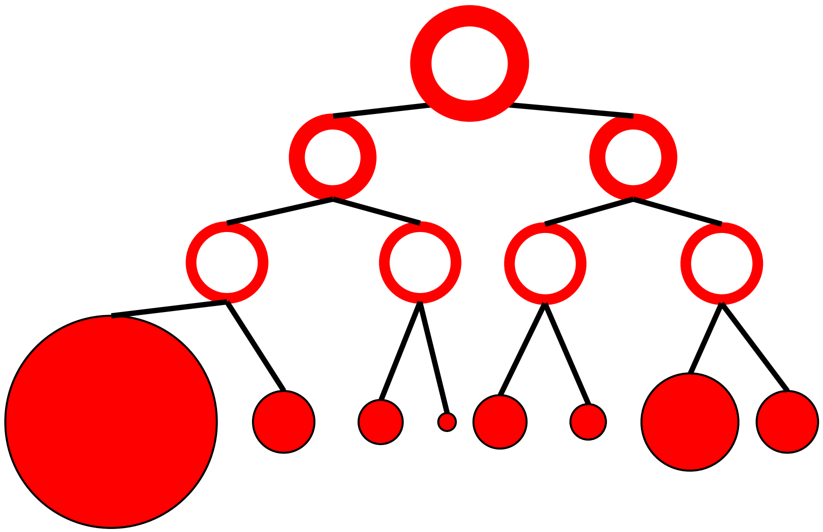
BVH Traversal: Penetration Volume Queries

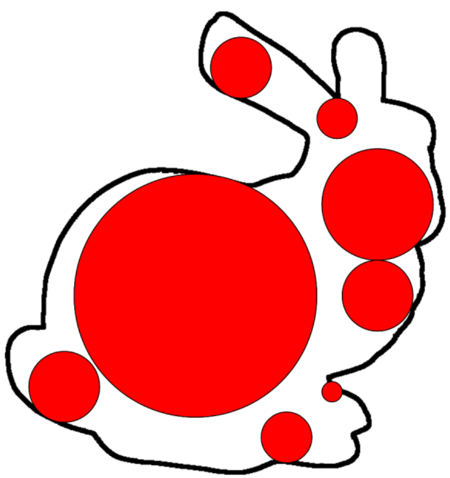


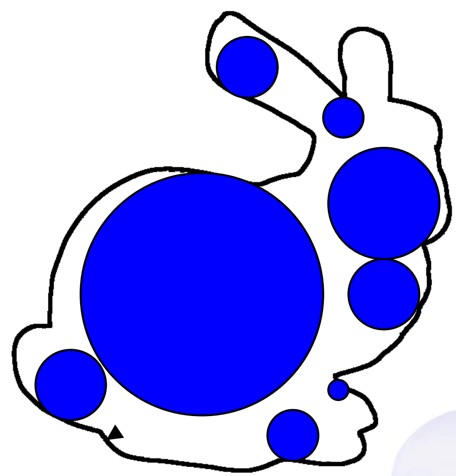


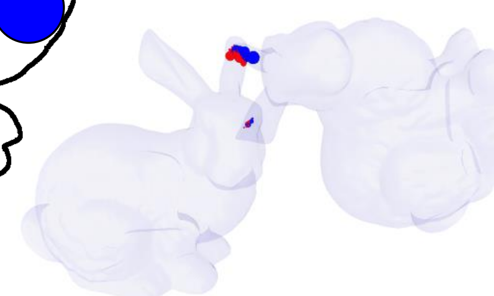


BVH Traversal: Penetration Volume Queries

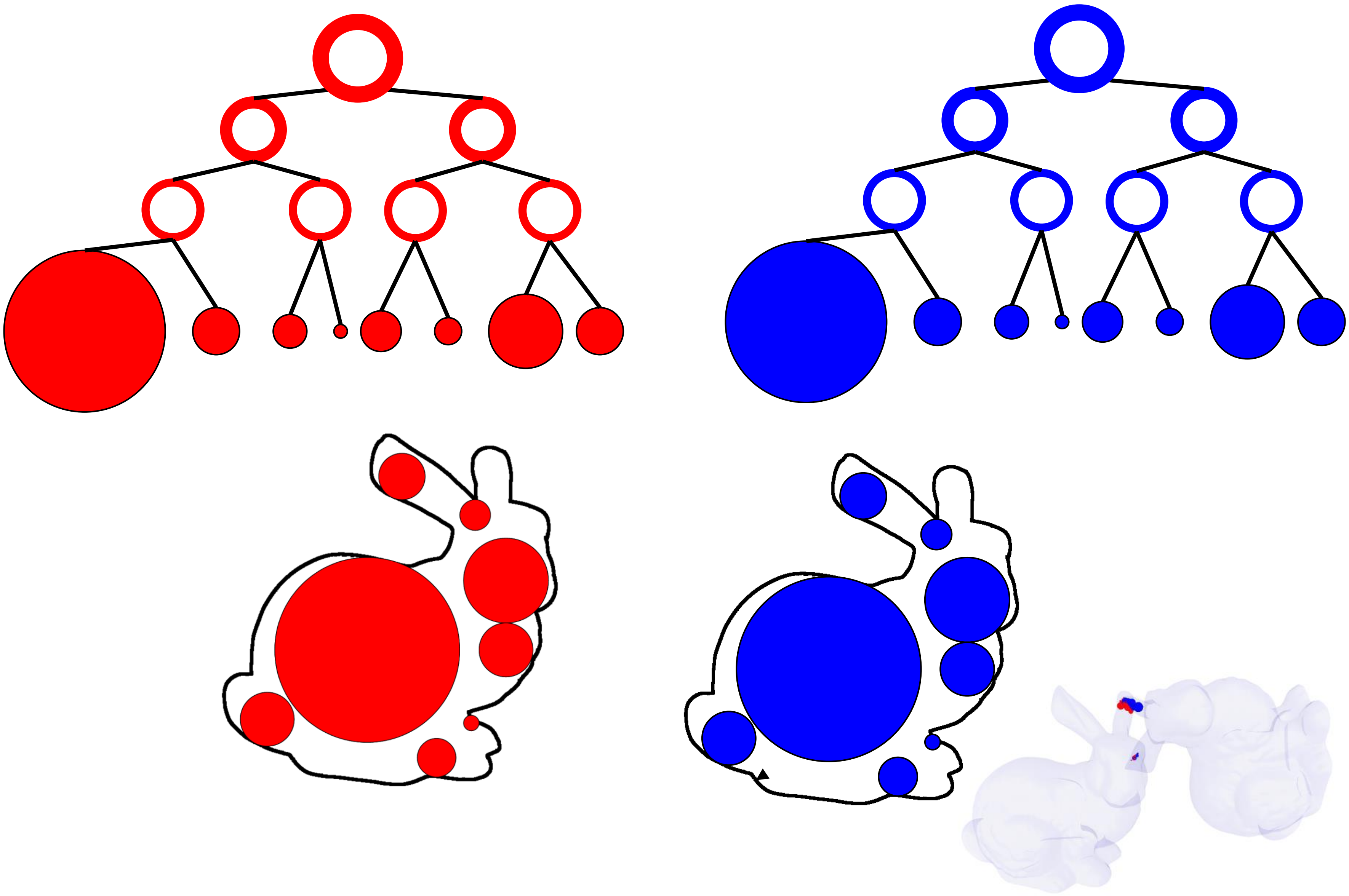




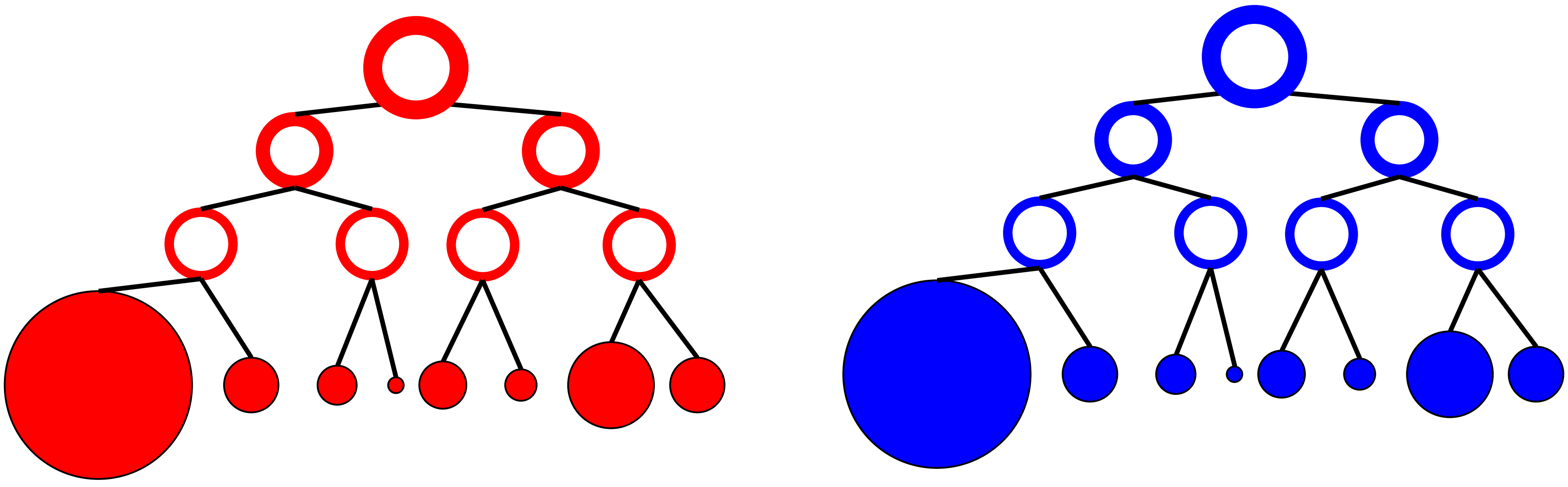


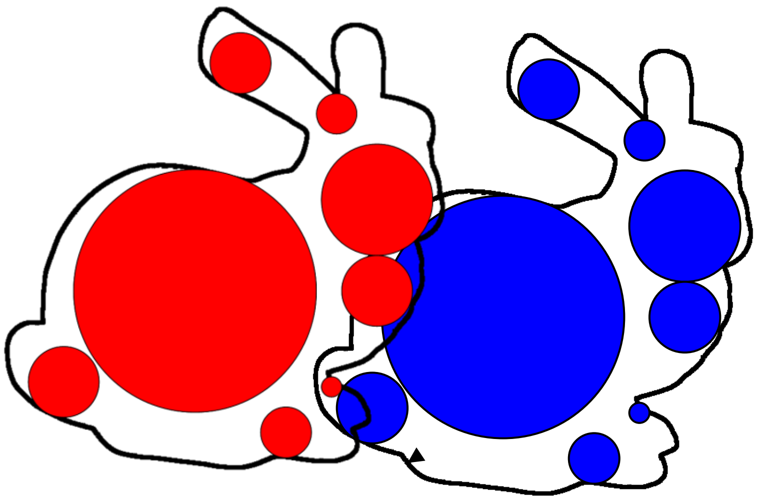


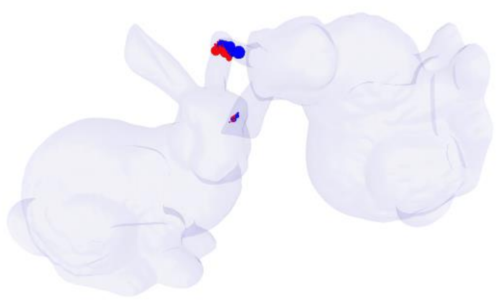
BVH Traversal: Penetration Volume Queries



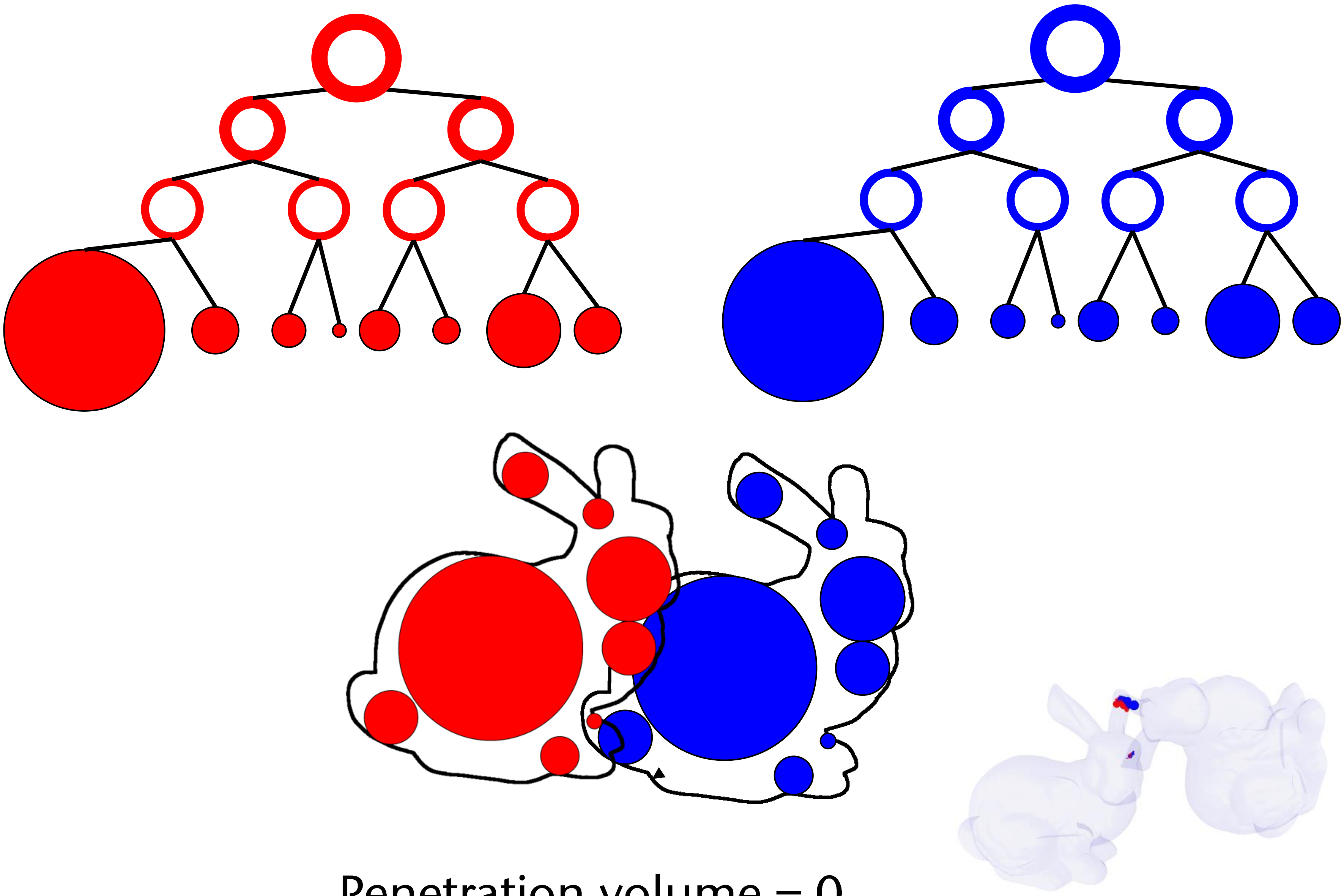
BVH Traversal: Penetration Volume Queries





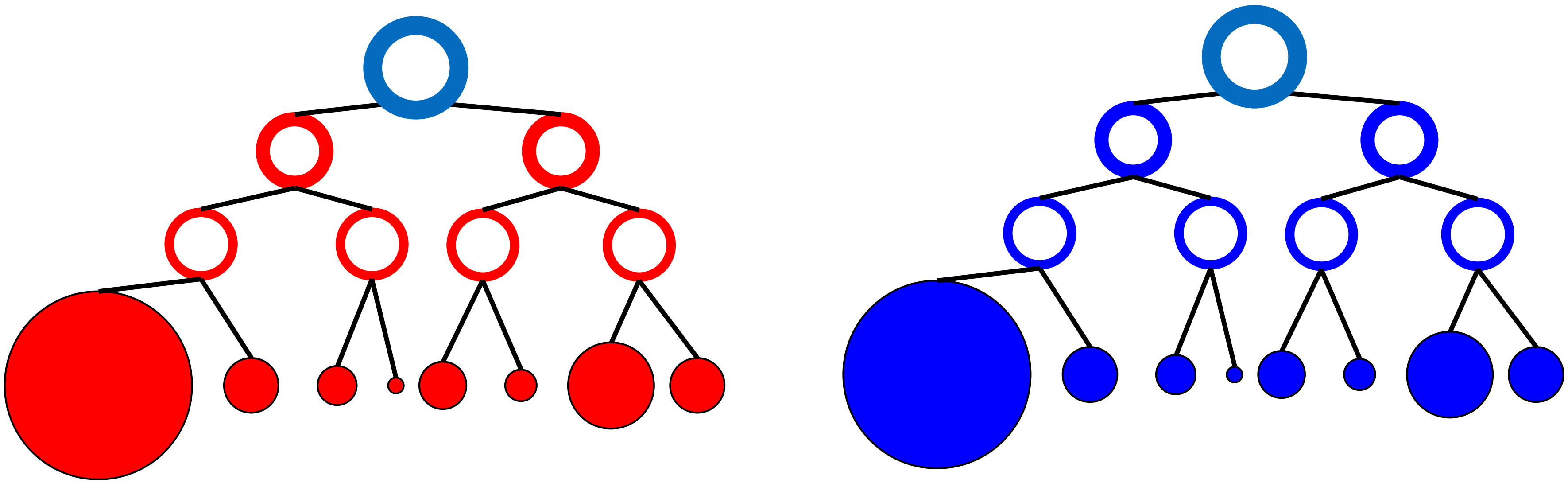


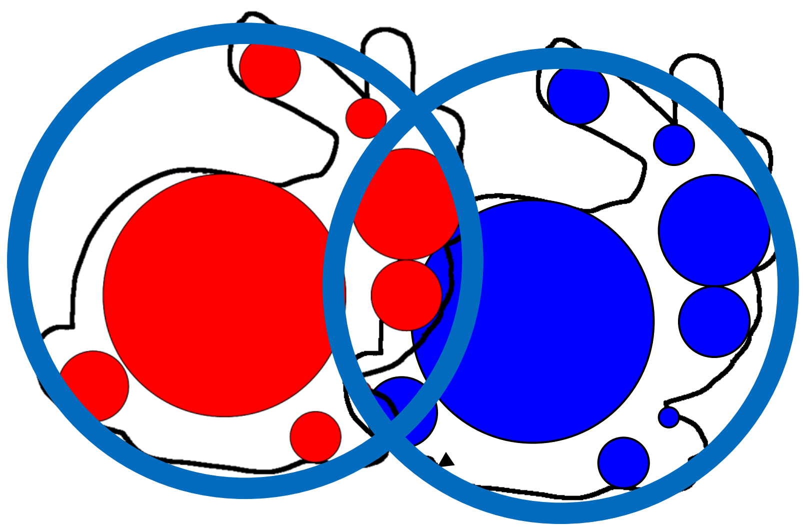
BVH Traversal: Penetration Volume Queries

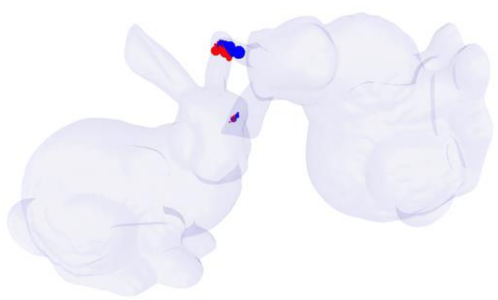


Penetration volume = 0

BVH Traversal: Penetration Volume Queries

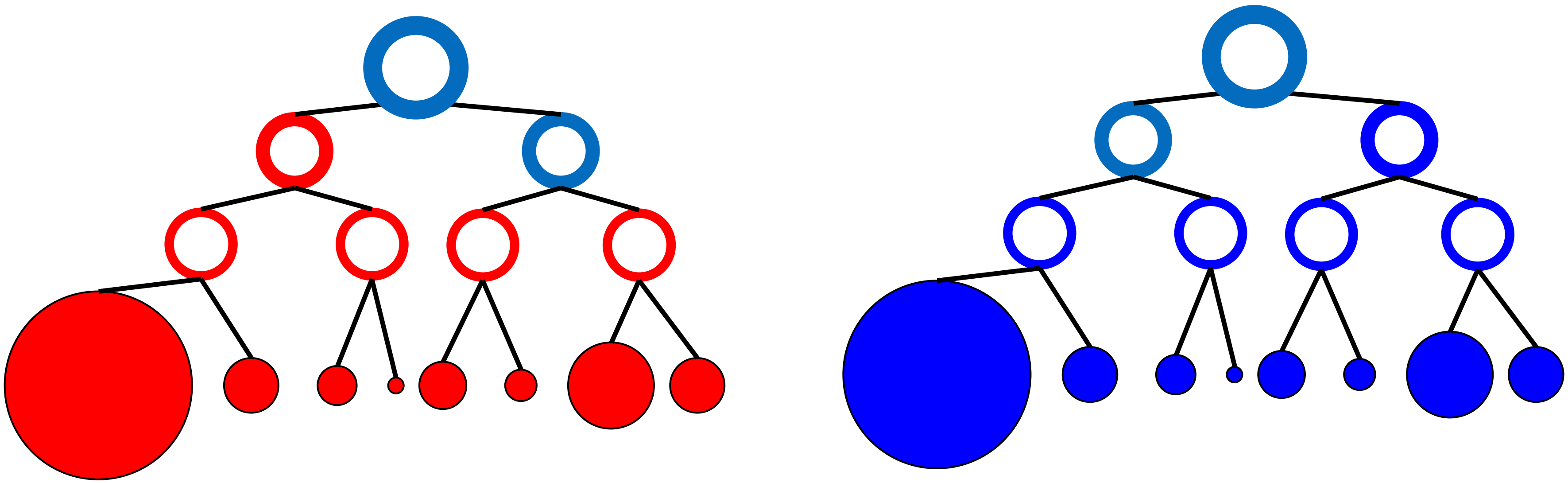


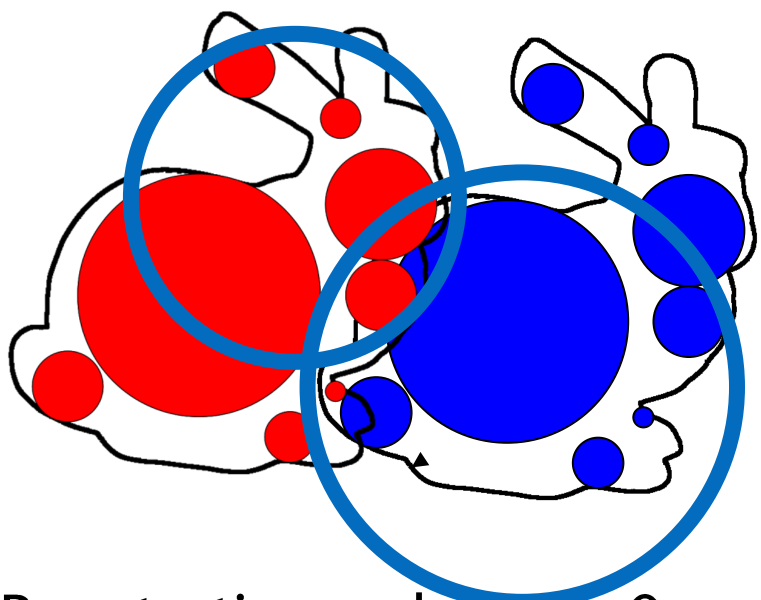


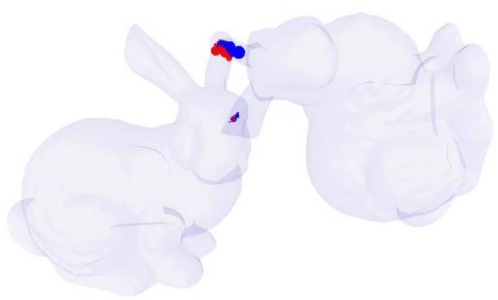


Penetration volume = 0

BVH Traversal: Penetration Volume Queries

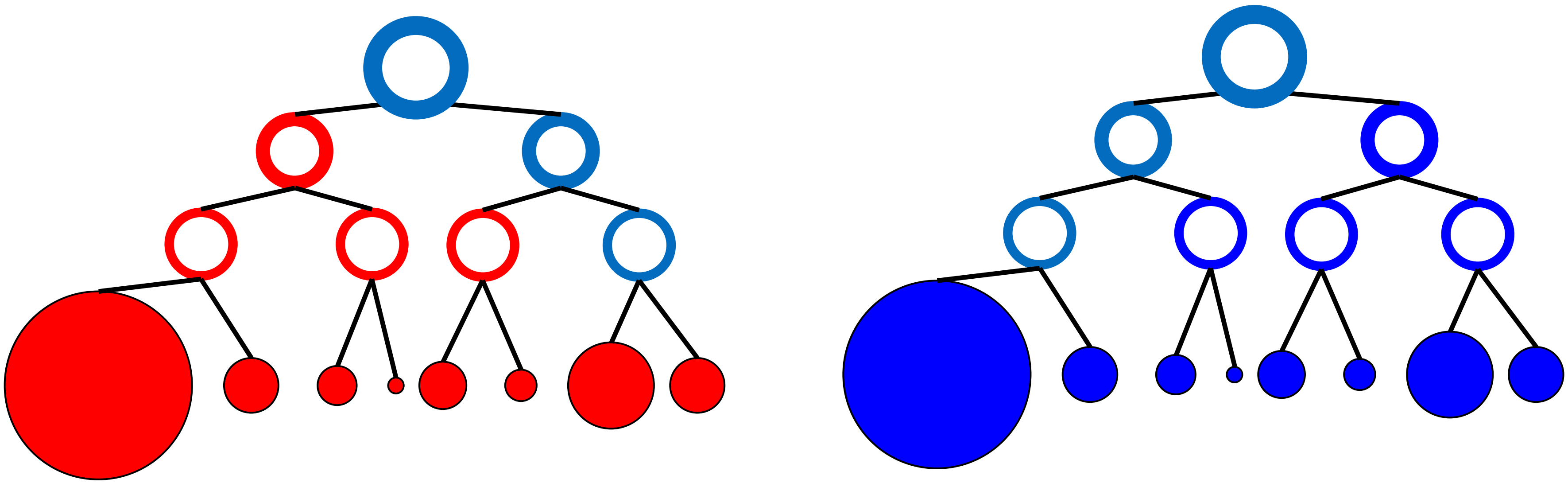


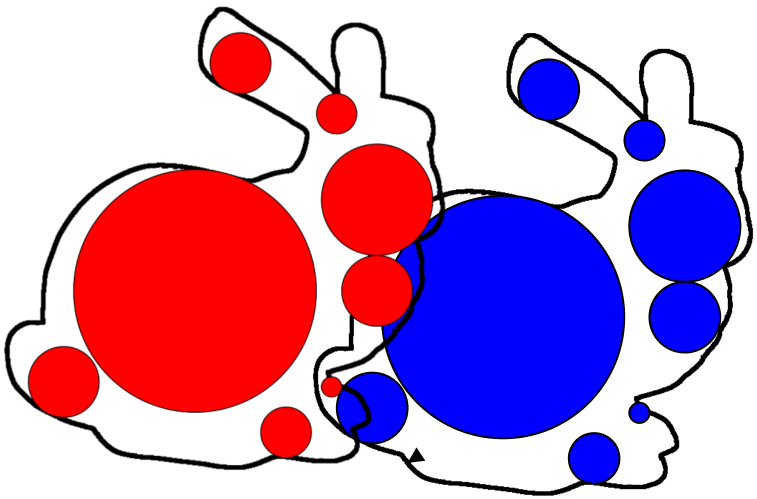


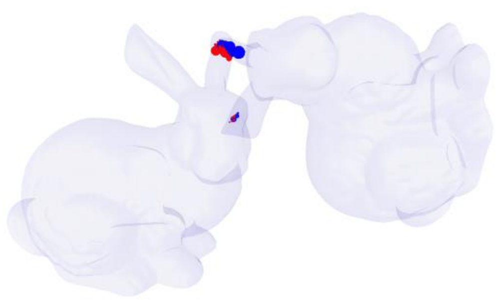


Penetration volume = 0

BVH Traversal: Penetration Volume Queries

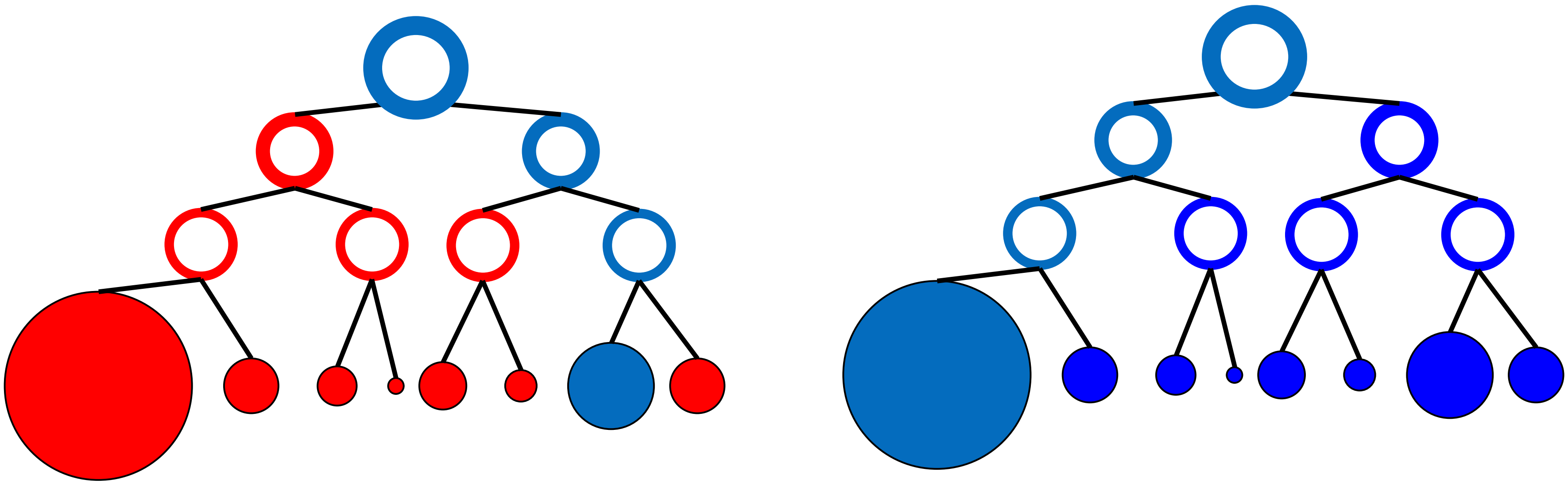


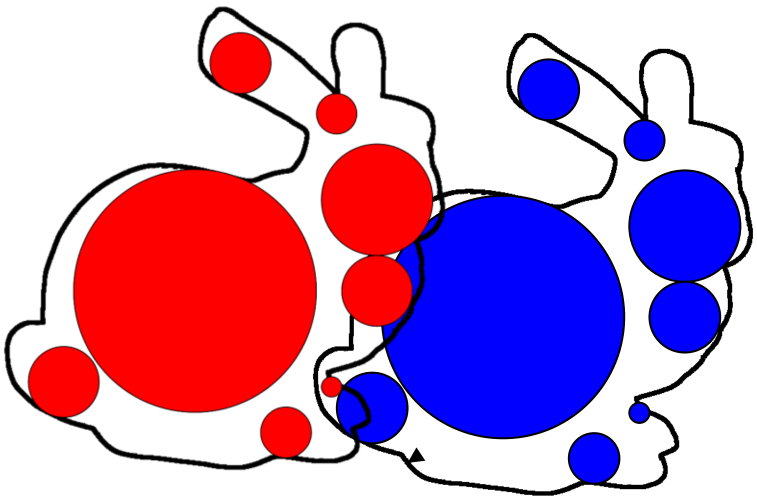




Penetration volume = 0

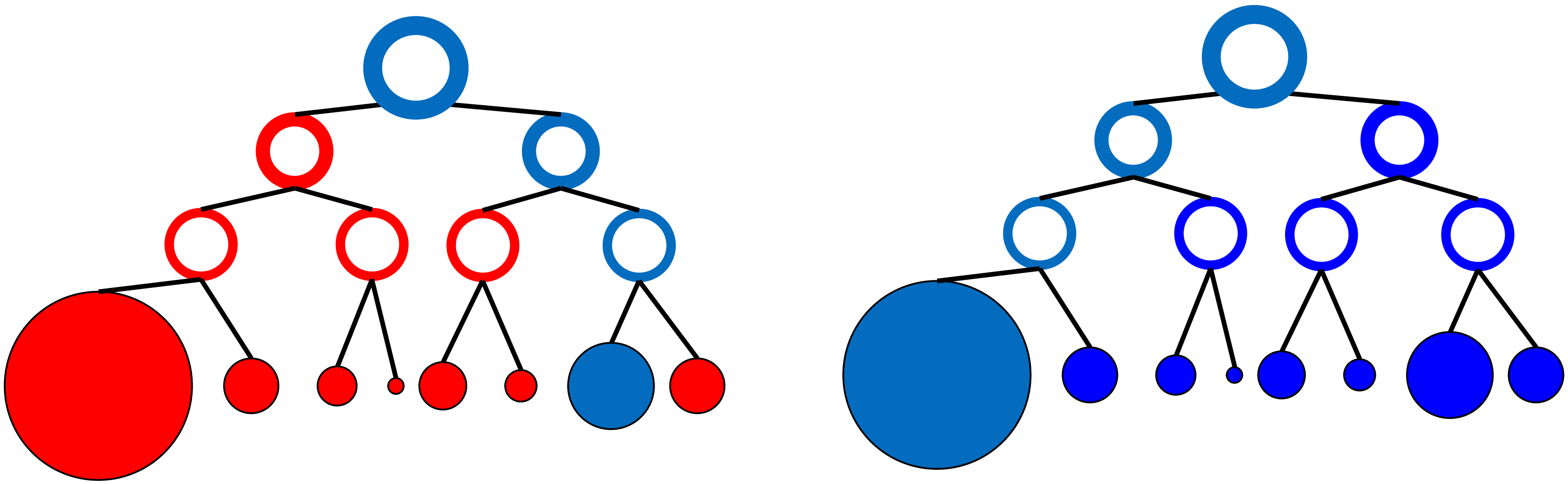
BVH Traversal: Penetration Volume Queries

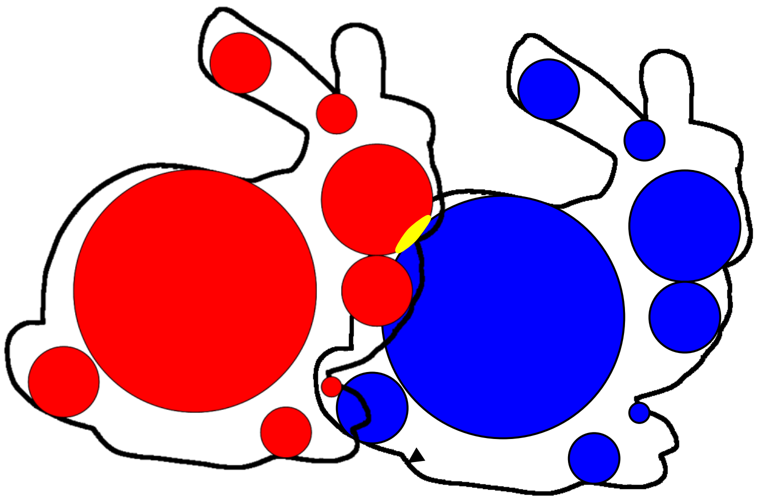


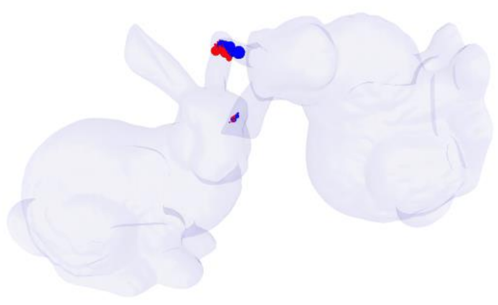


Penetration volume = 0

BVH Traversal: Penetration Volume Queries

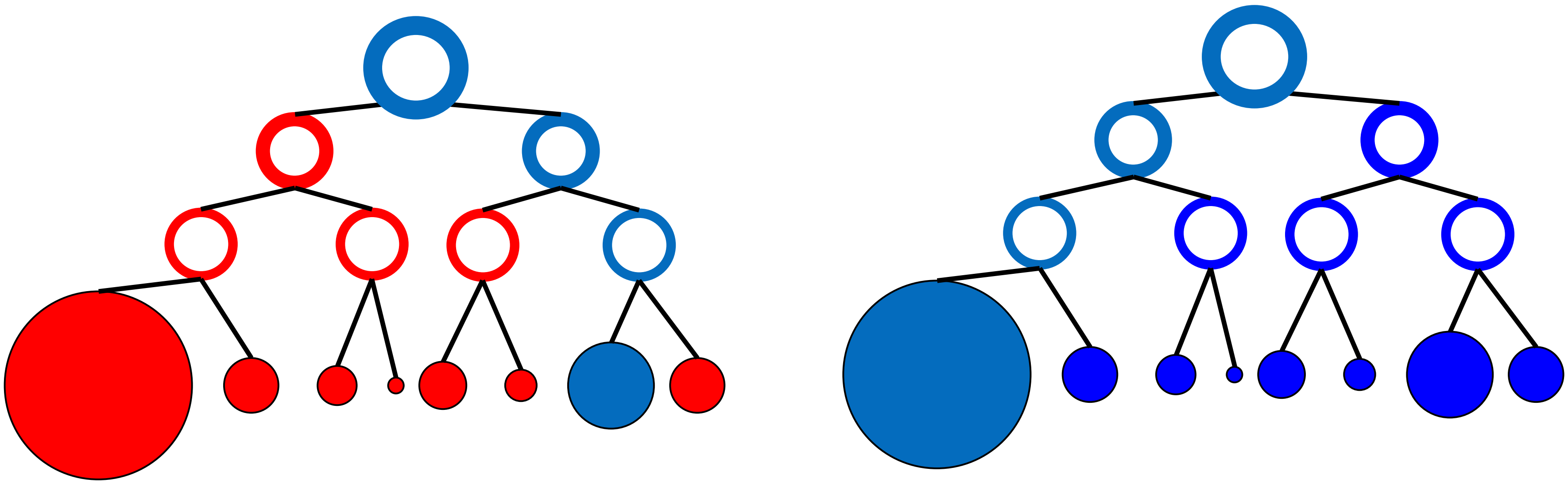


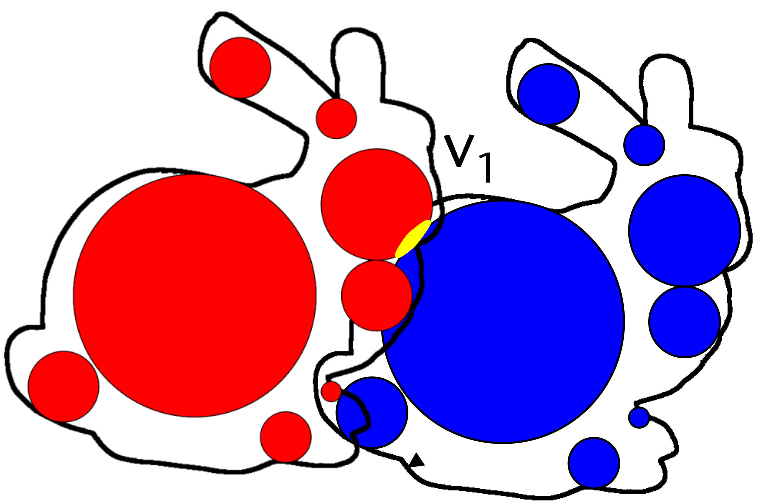


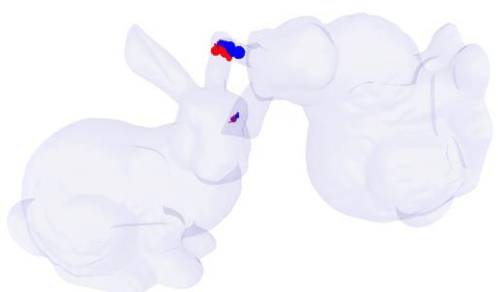


Penetration volume = 0

BVH Traversal: Penetration Volume Queries

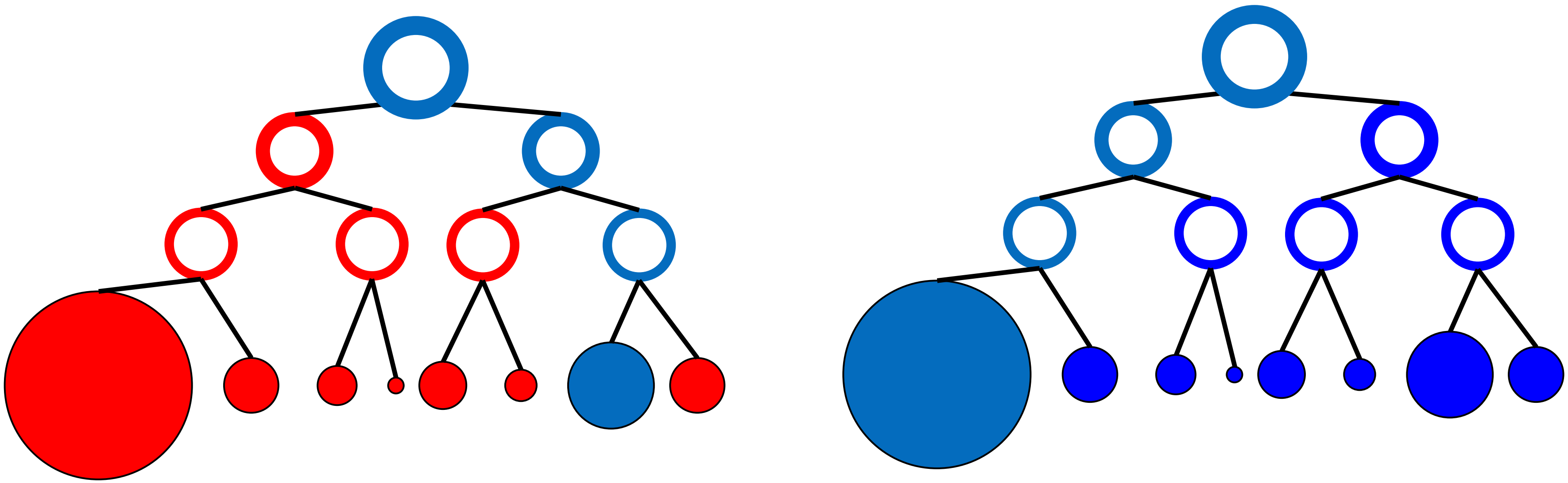


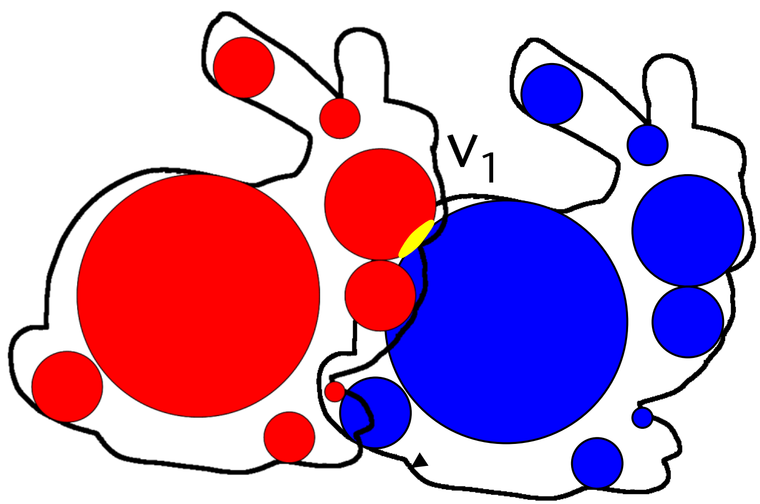


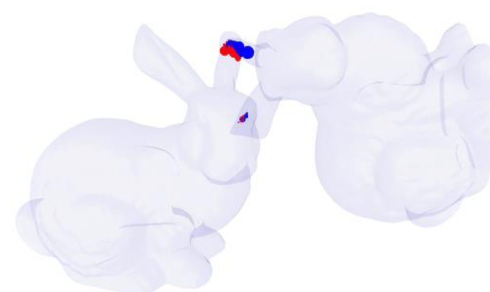


Penetration volume = 0

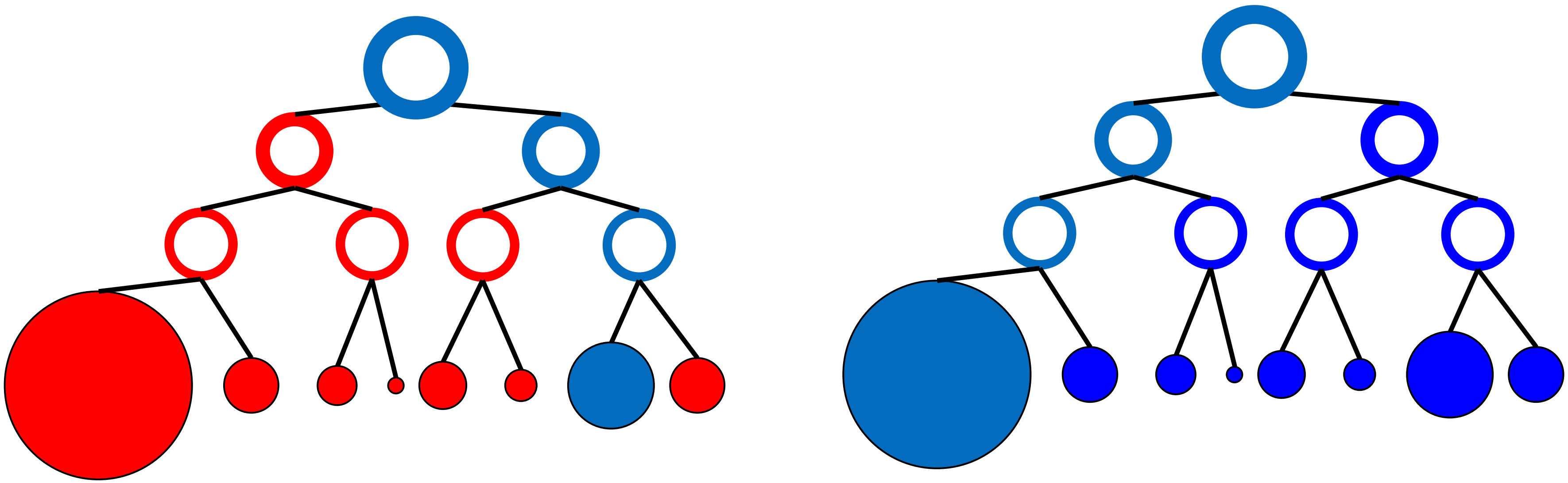
BVH Traversal: Penetration Volume Queries

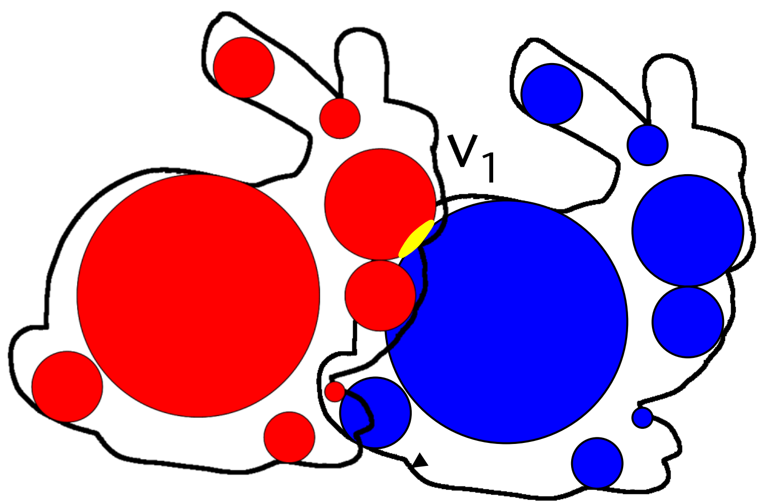






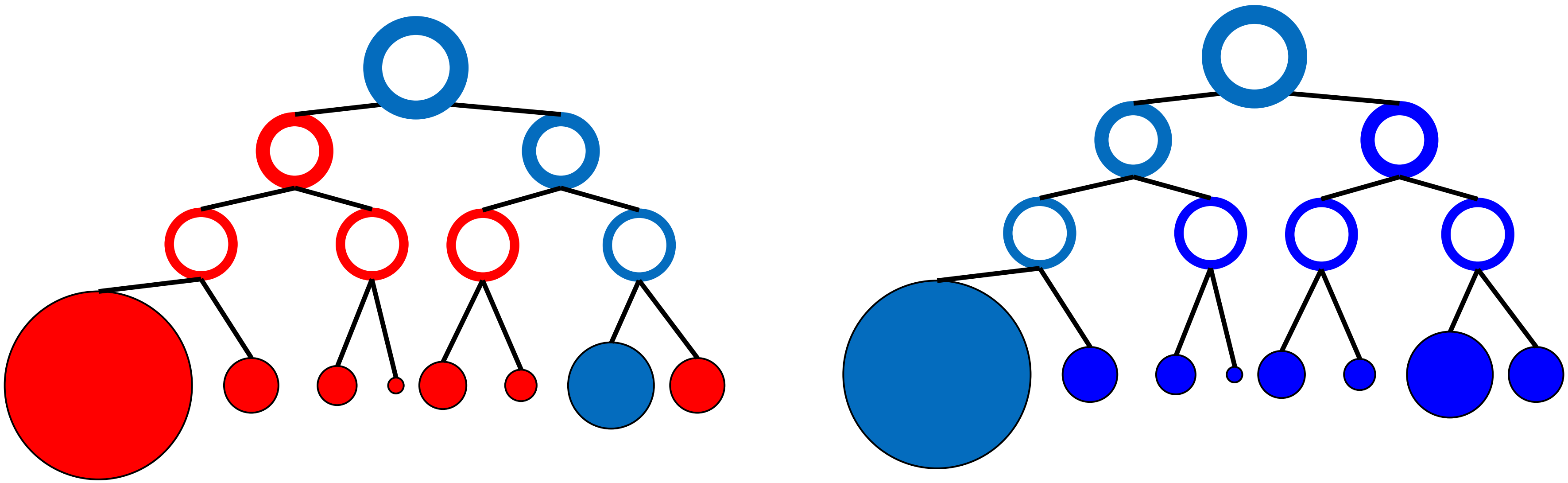
BVH Traversal: Penetration Volume Queries

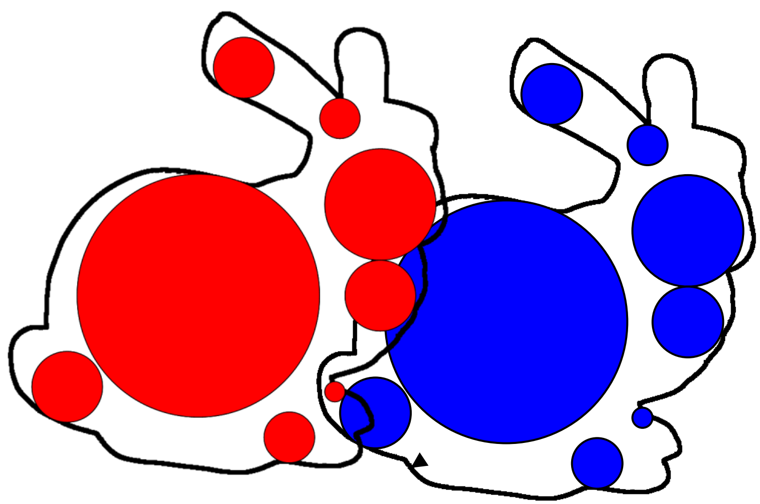


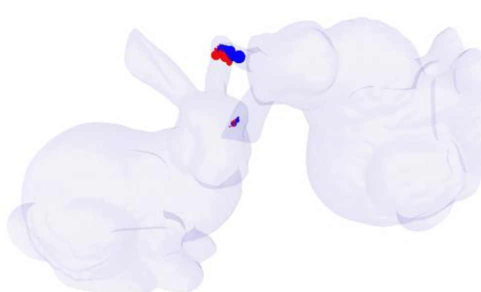


Penetration volume = v_1

BVH Traversal: Penetration Volume Queries

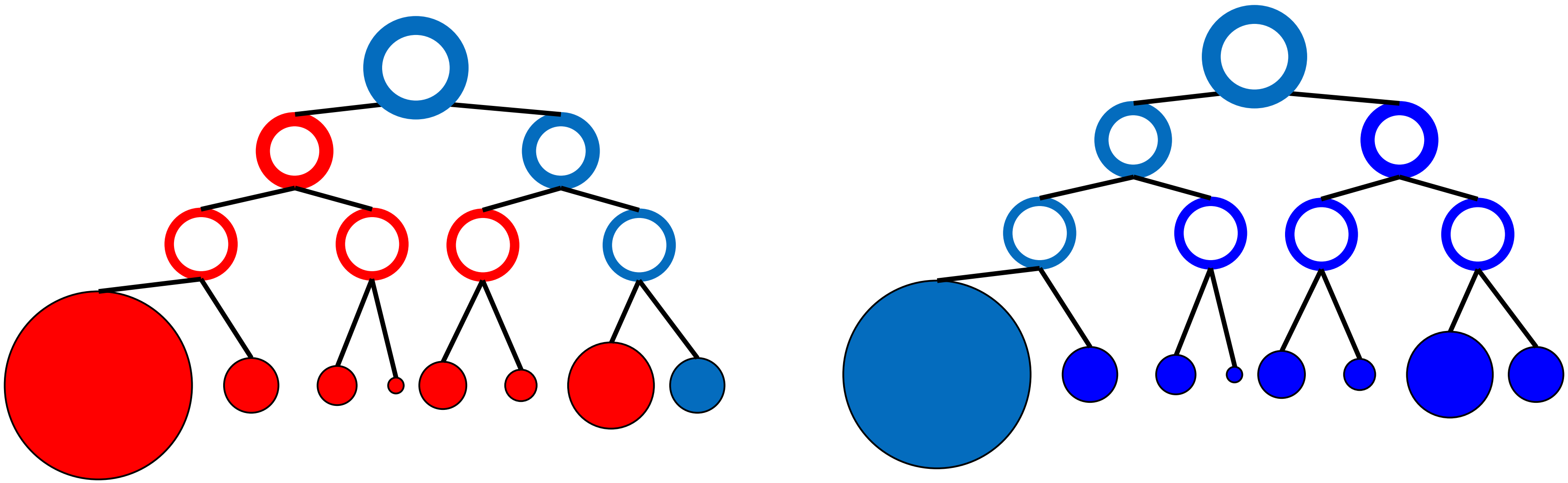


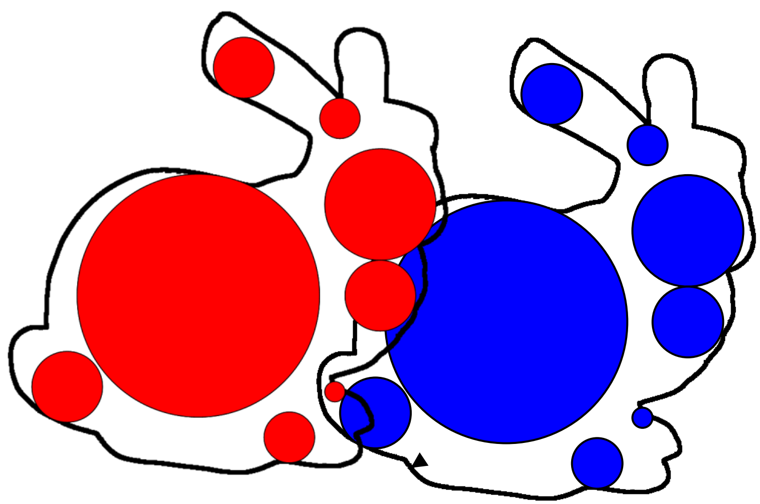


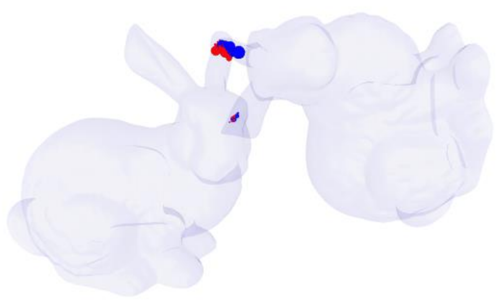


Penetration volume = v_1

BVH Traversal: Penetration Volume Queries

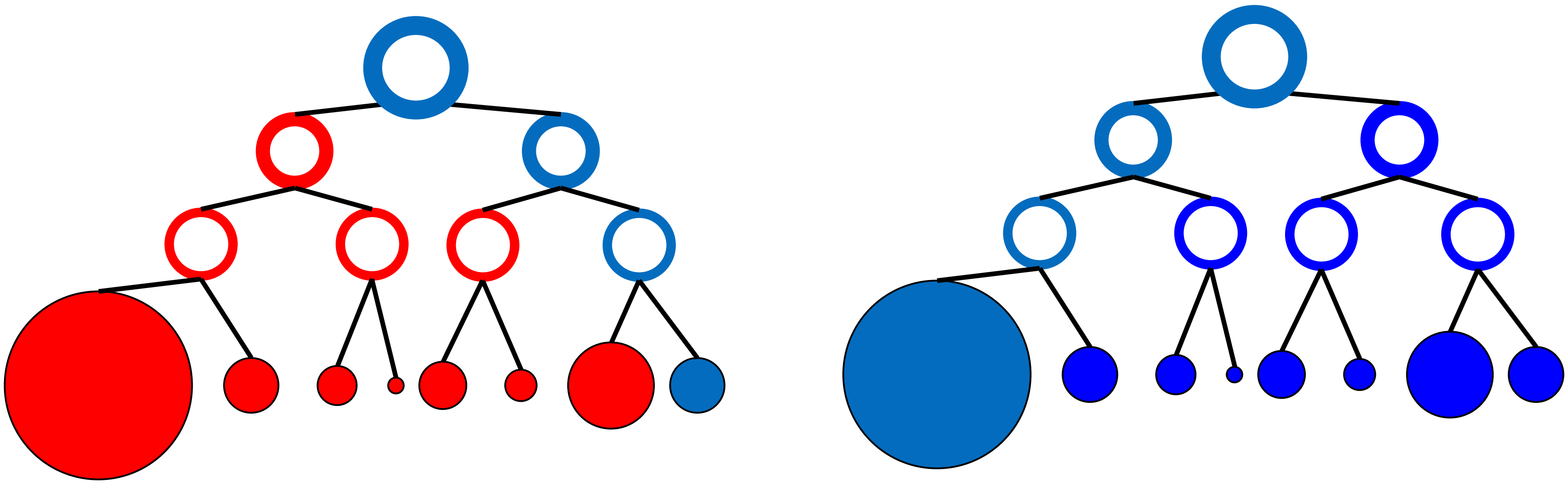


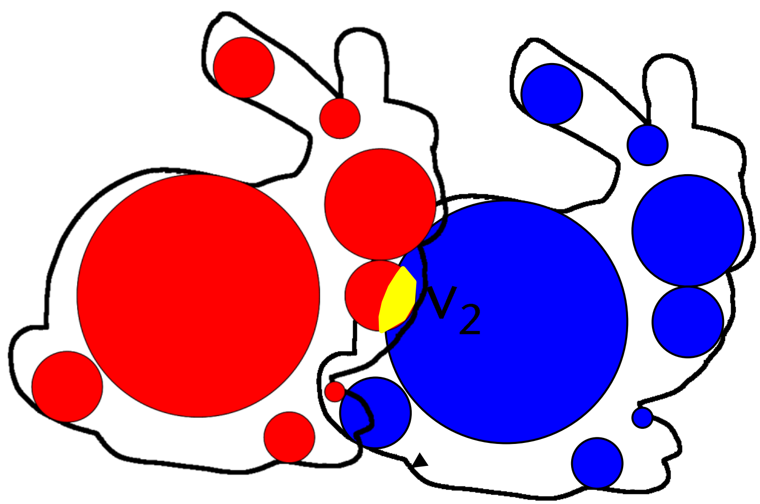


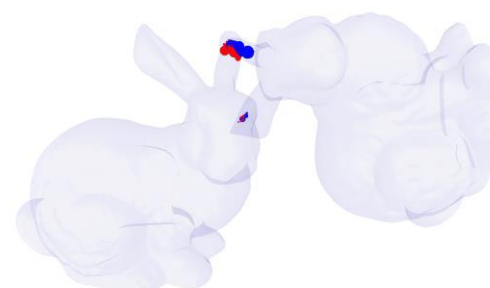


Penetration volume = v_1

BVH Traversal: Penetration Volume Queries

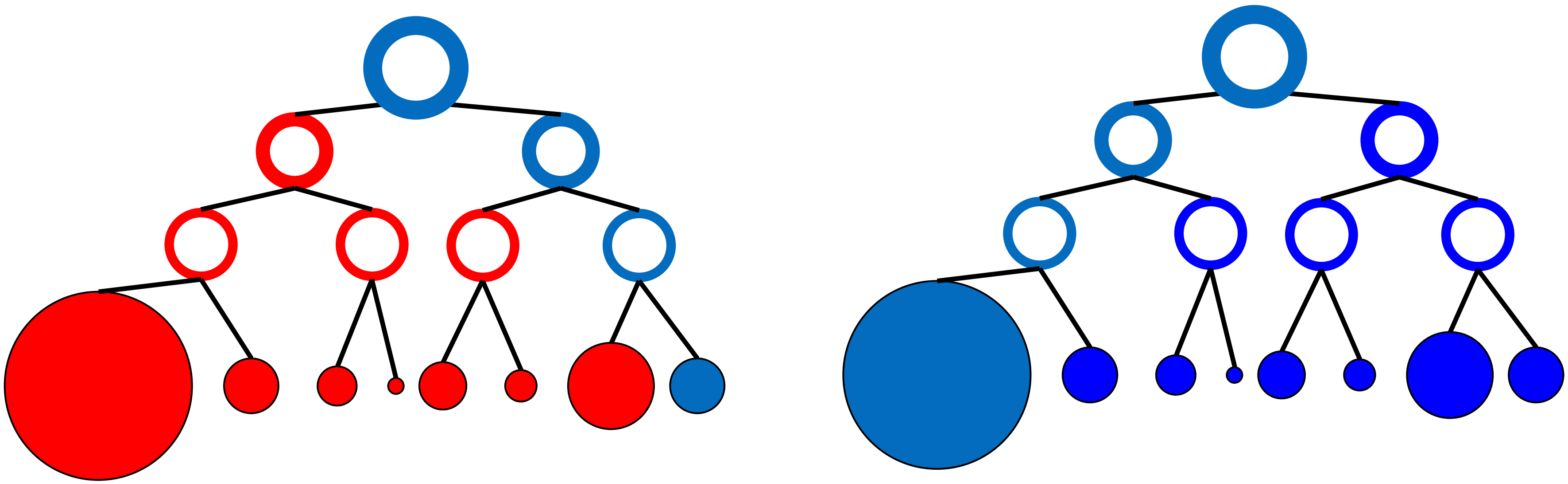


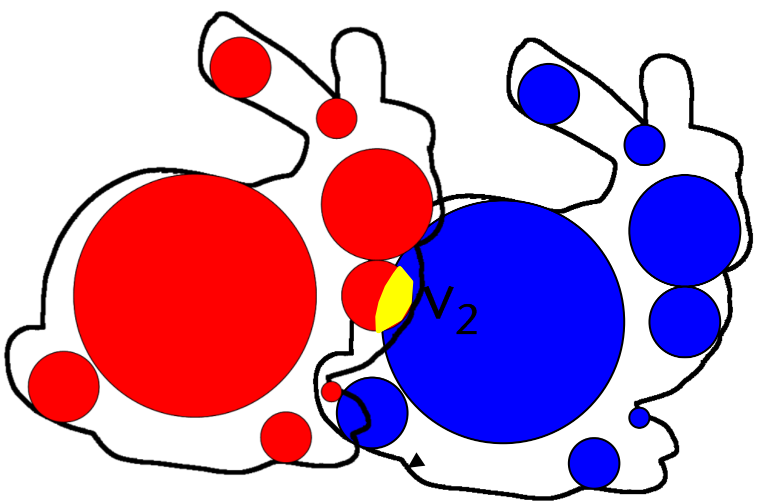


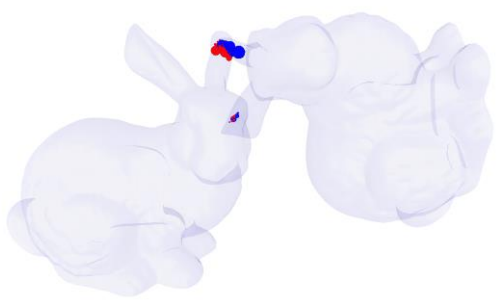


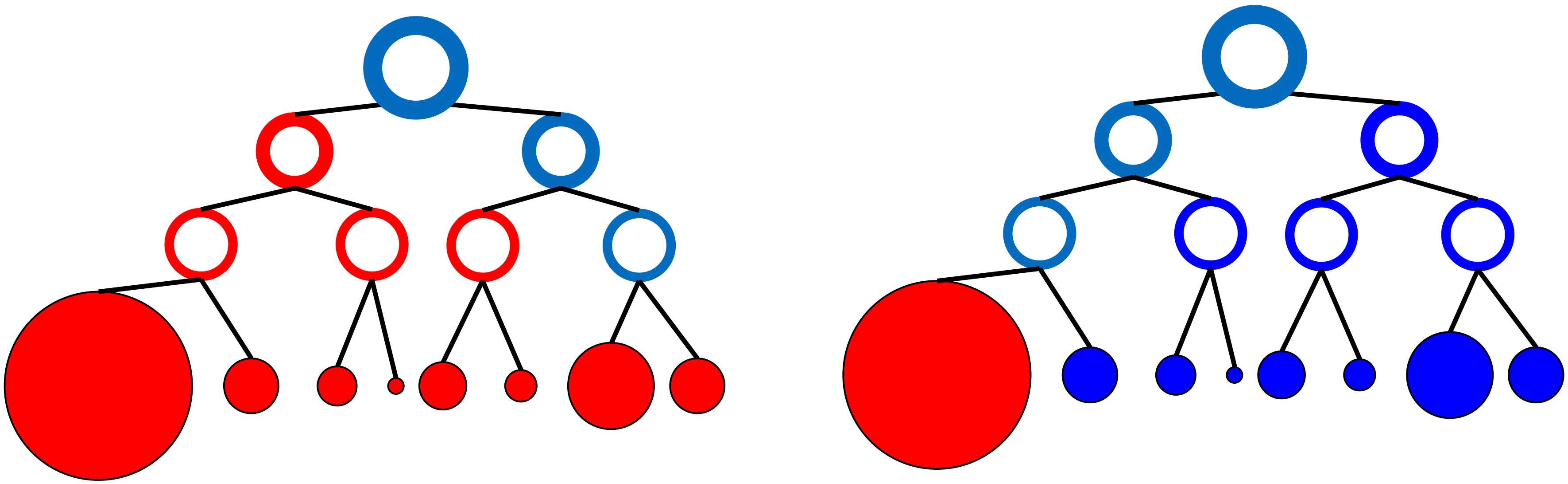
Penetration volume = v_1

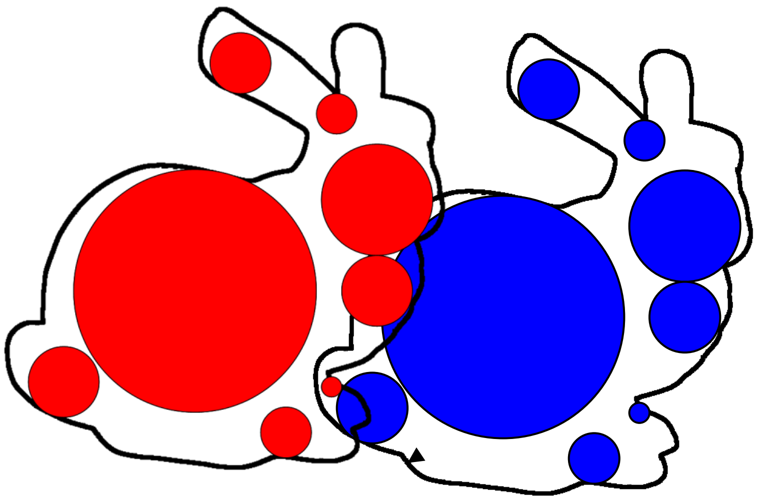
BVH Traversal: Penetration Volume Queries

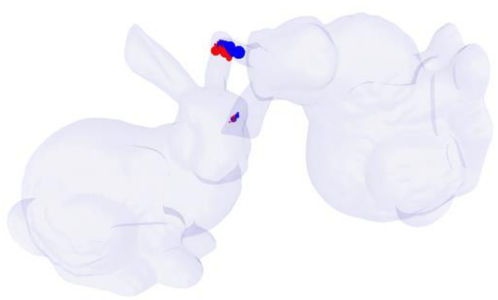






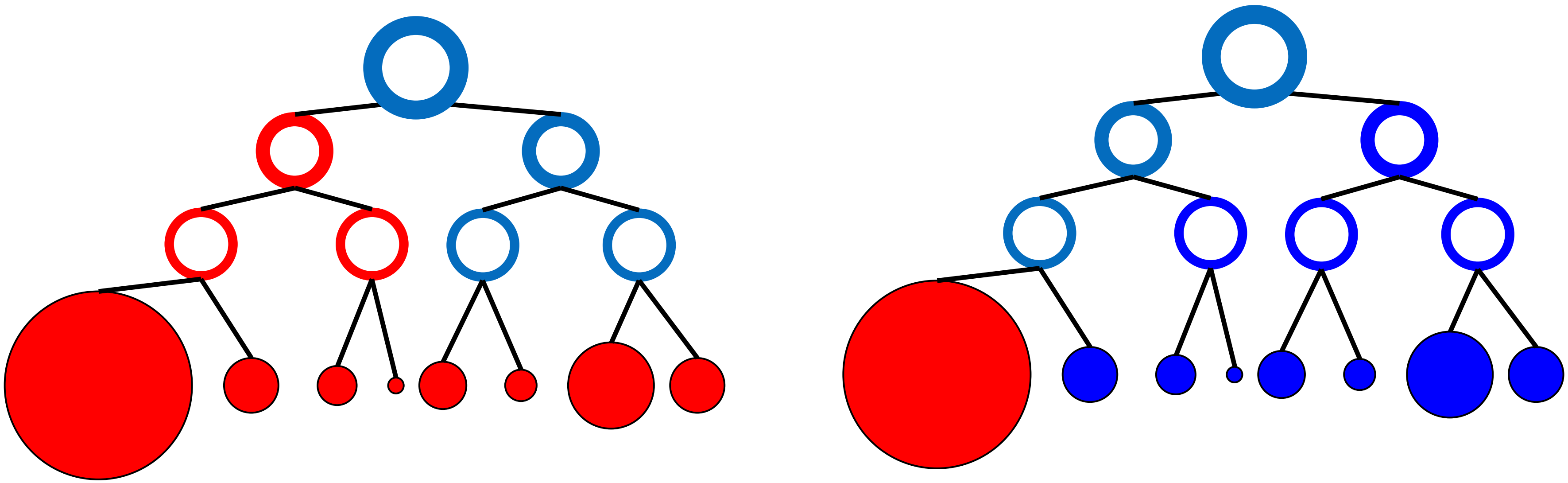


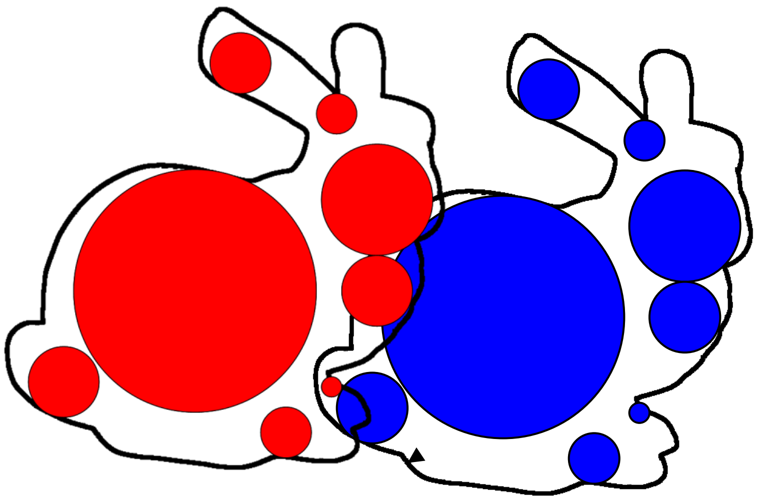


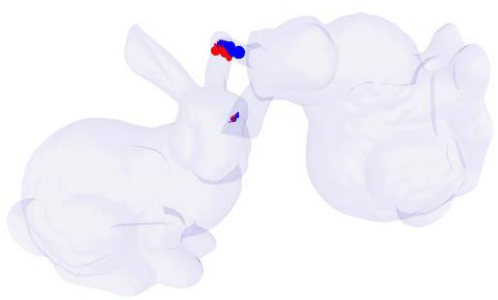


Penetration volume = $v_1 + v_2$

BVH Traversal: Penetration Volume Queries

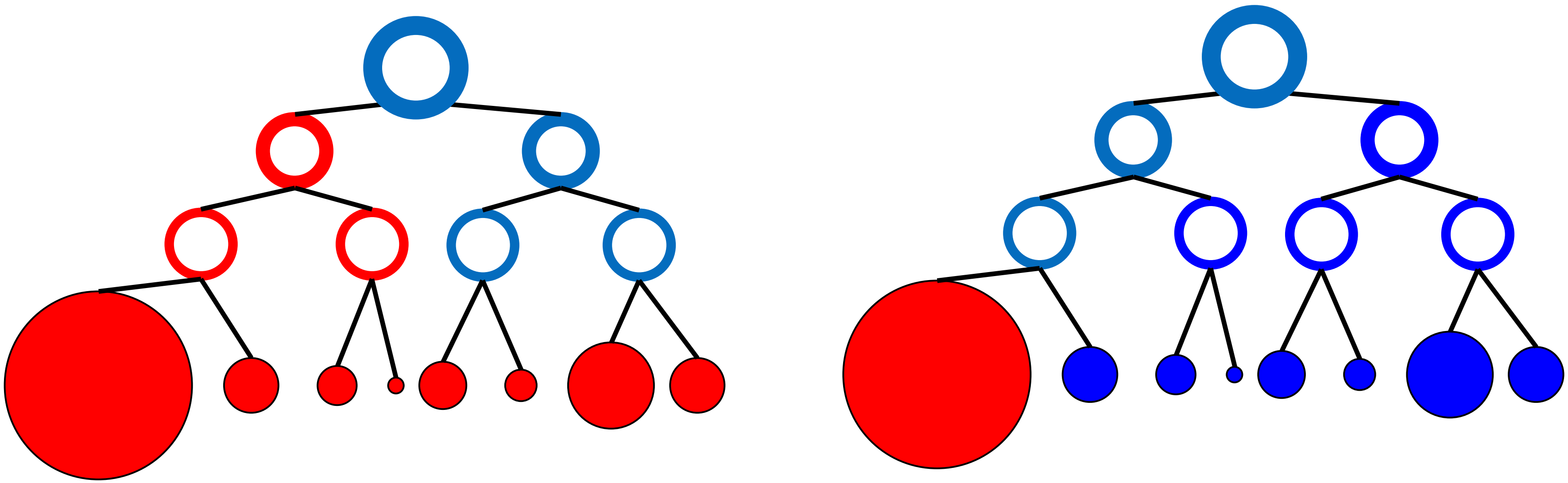


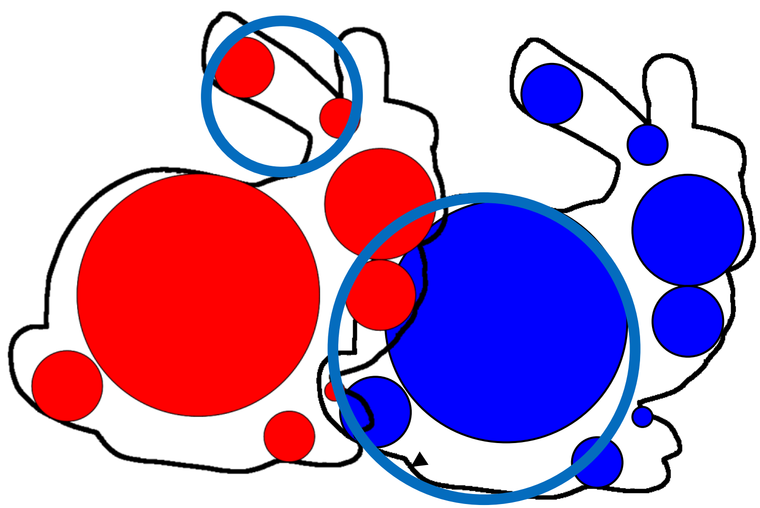


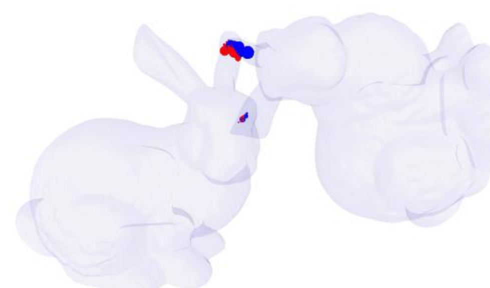


Penetration volume = $v_1 + v_2$

BVH Traversal: Penetration Volume Queries

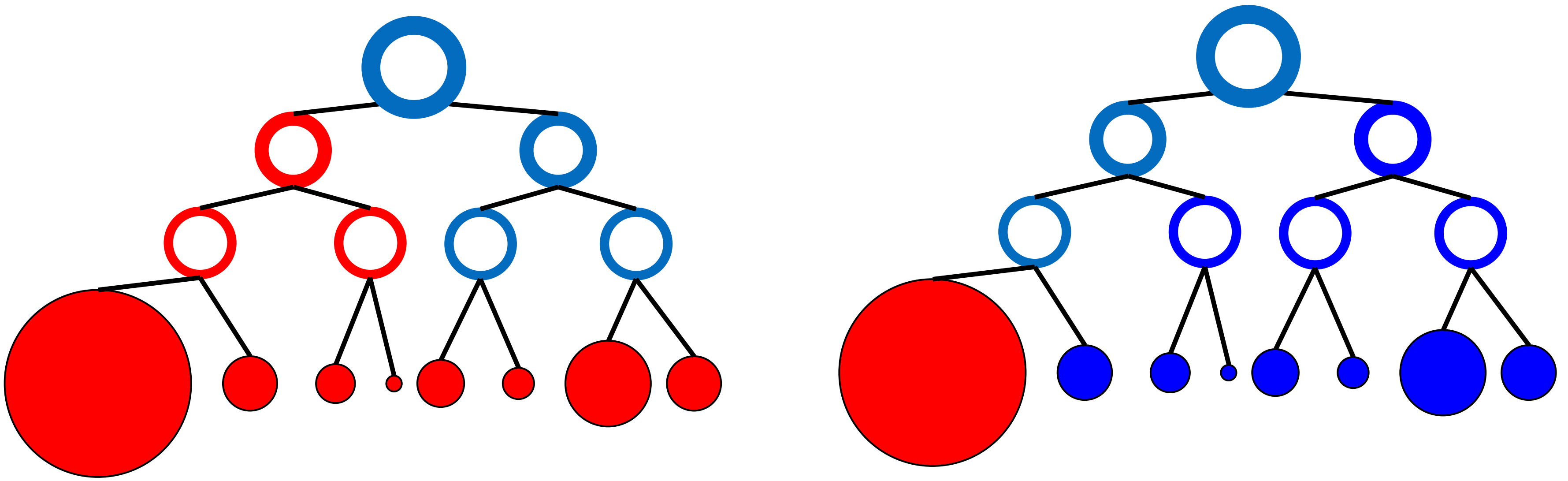


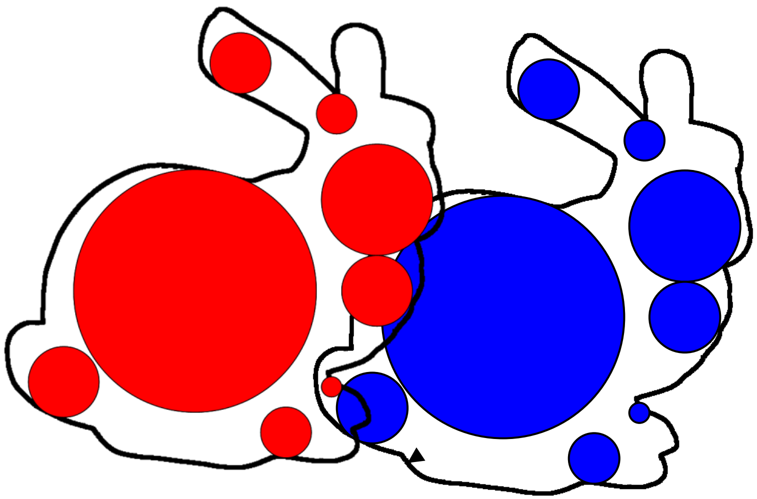


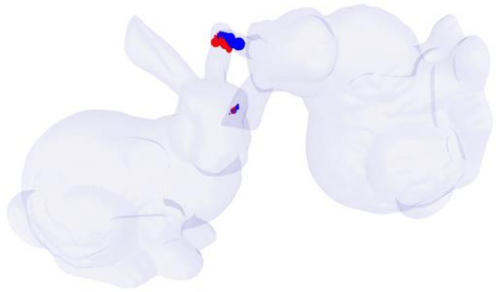


Penetration volume = $v_1 + v_2$

BVH Traversal: Penetration Volume Queries

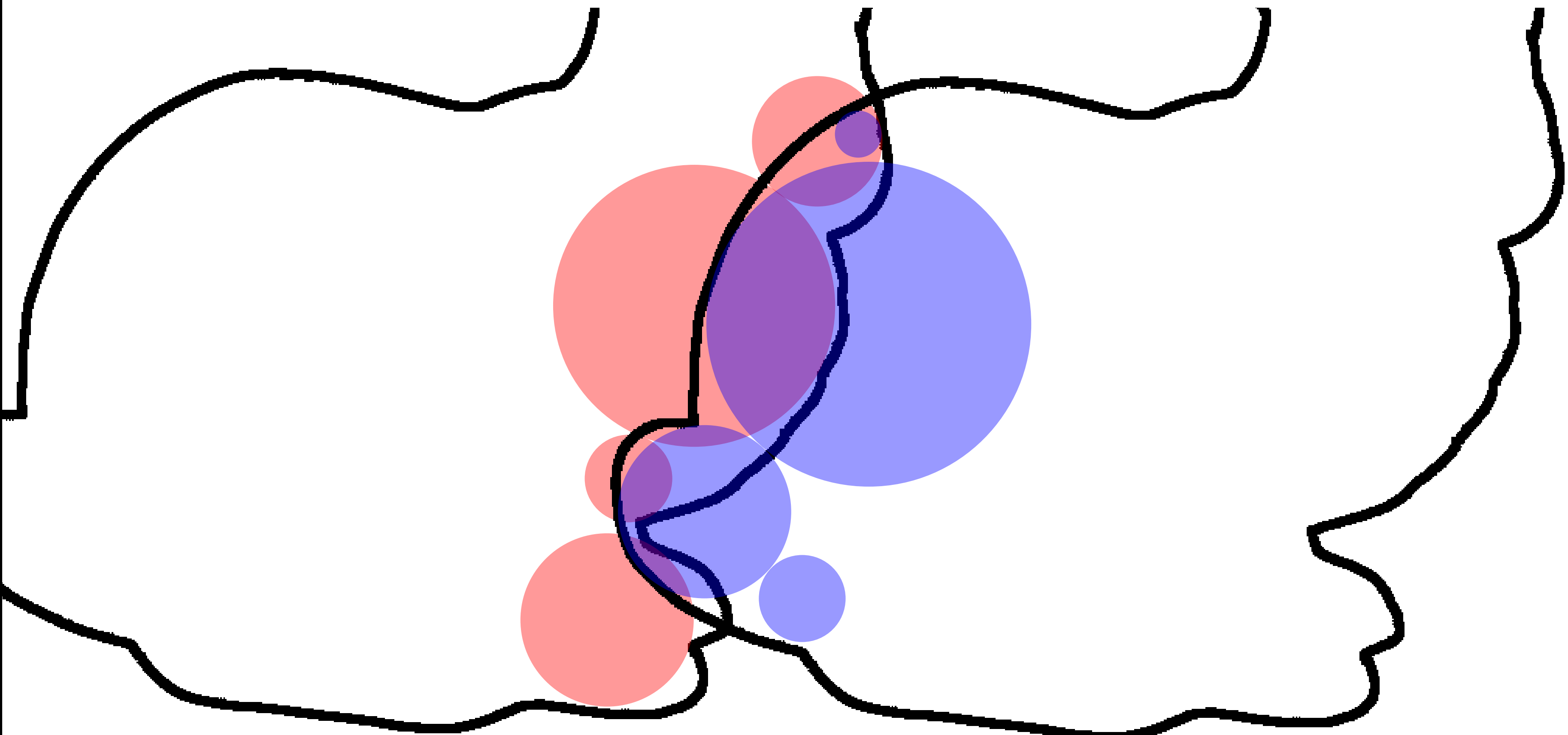




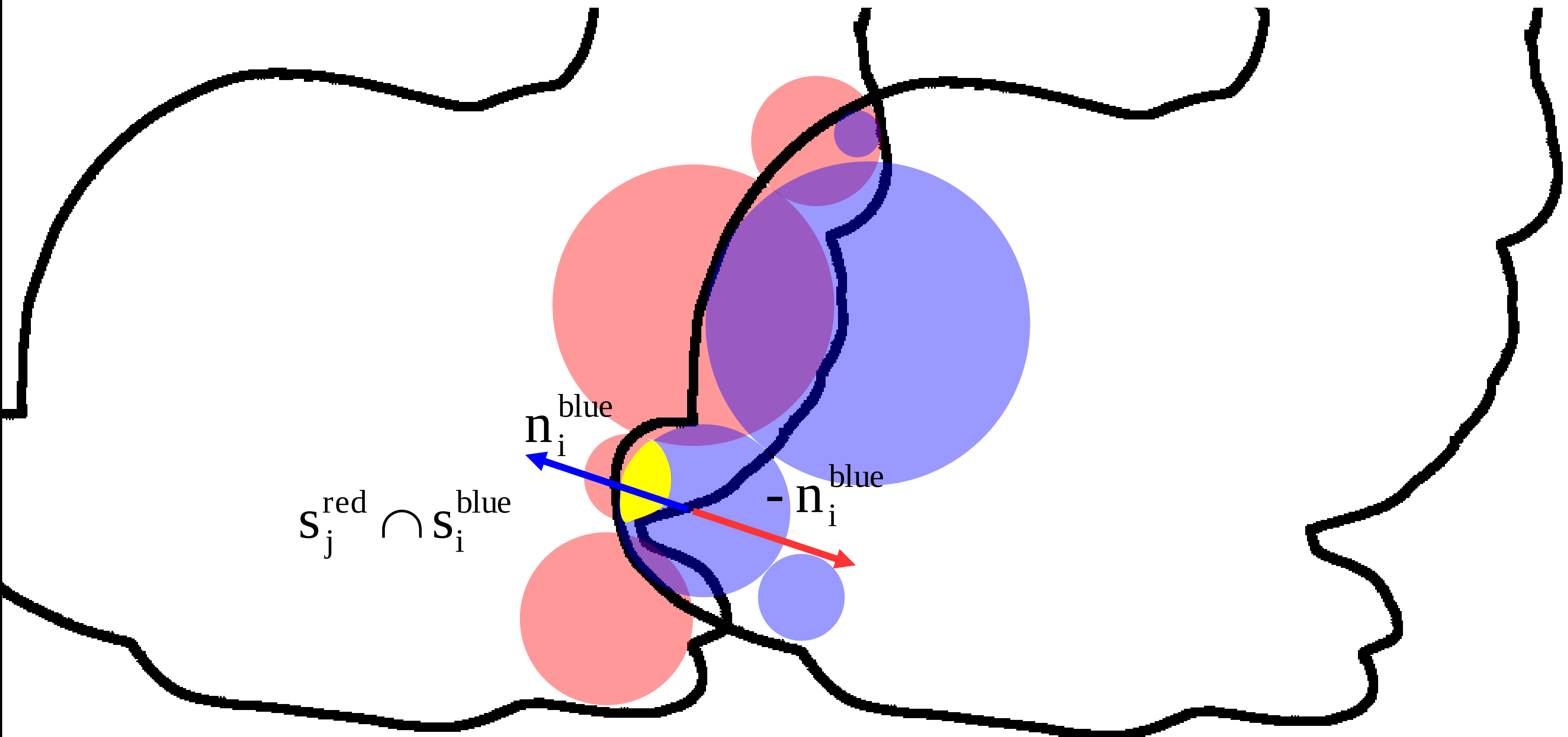


Penetration volume = $v_1 + v_2$

Collision Response: Forces and Torques



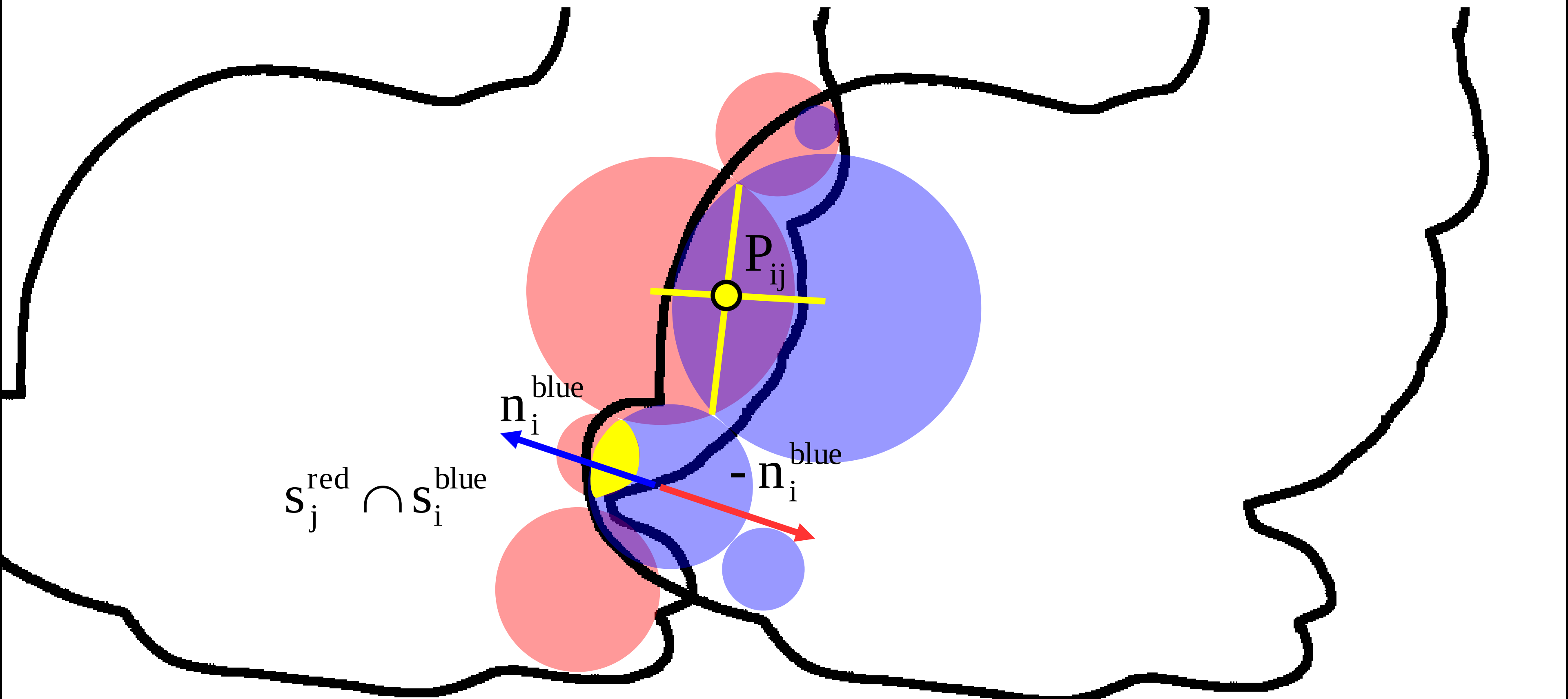
Collision Response: Forces and Torques



$$f_{ij}^{blue} = (s_j^{red} \cap s_i^{blue})(-n_i^{blue})$$

$$f_{total}^{blue} = \sum f_{ij}^{blue}$$

Collision Response: Forces and Torques



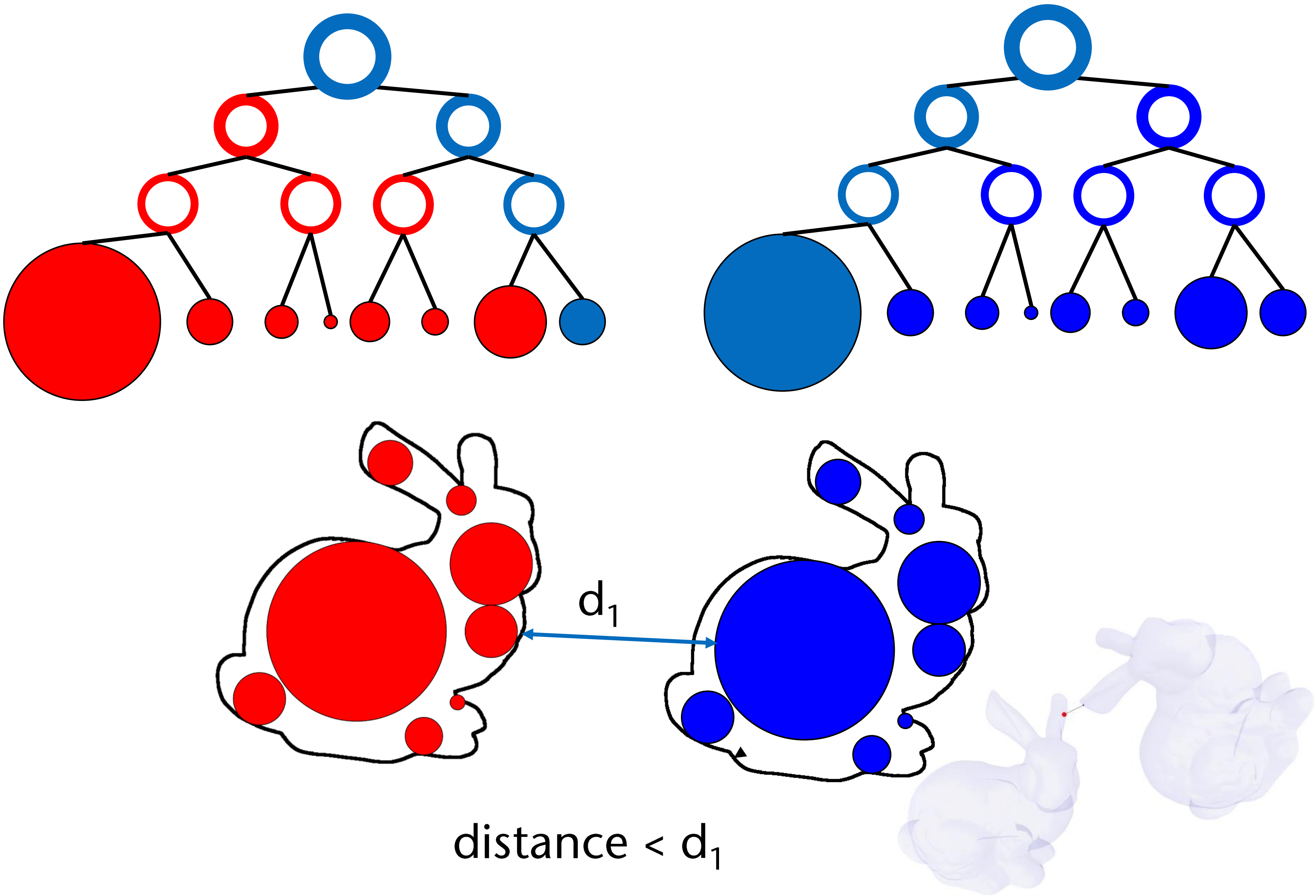
$$f_{ij}^{blue} = (s_j^{red} \cap s_i^{blue})(-n_i^{blue})$$

$$\tau_{ij}^{blue} = (P_{ij} - C_m) \times (f_{ij}^{blue})$$

$$f_{total}^{blue} = \sum f_{ij}^{blue}$$

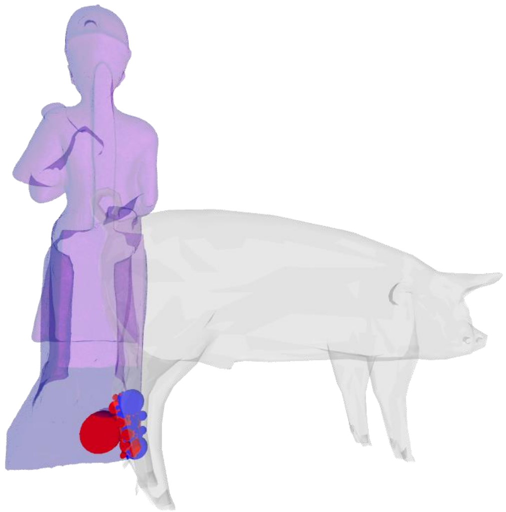
$$\tau_{total}^{blue} = \sum \tau_{ij}^{blue}$$

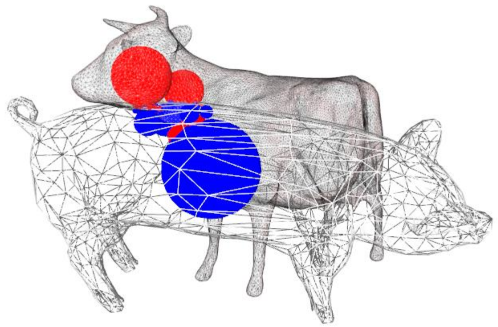
BVH Traversal: Proximity Queries

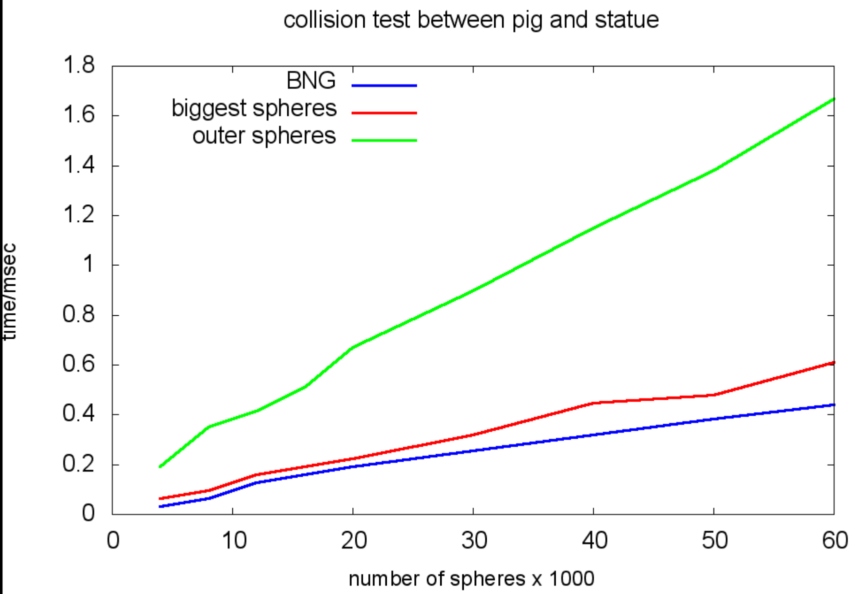


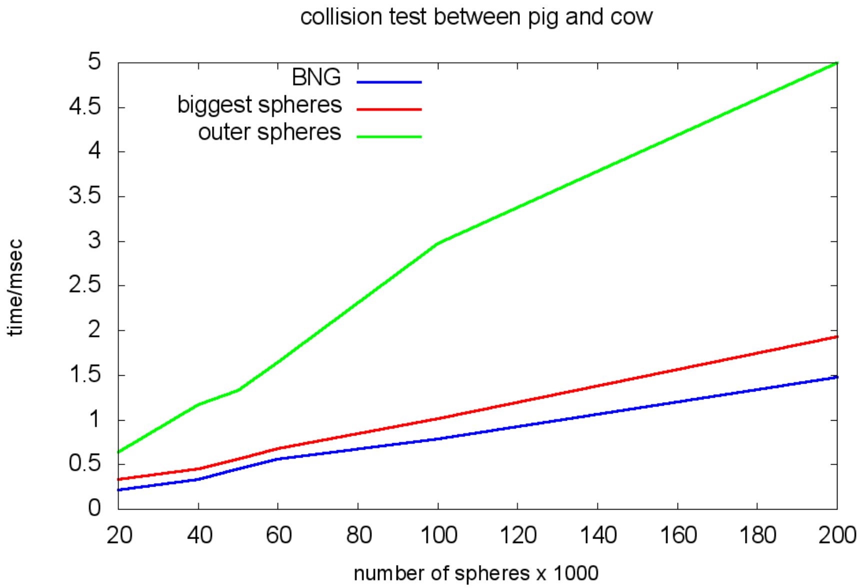
distance < d_1

Results: Runtime Performance



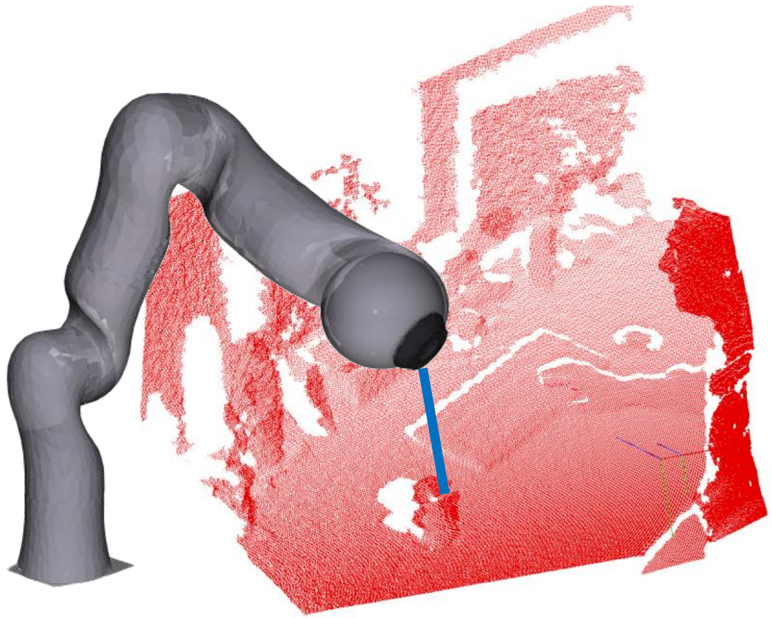




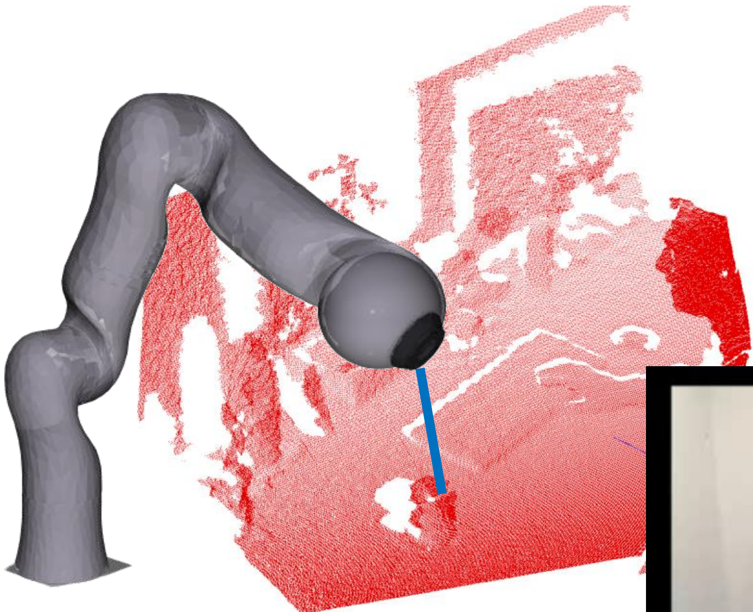


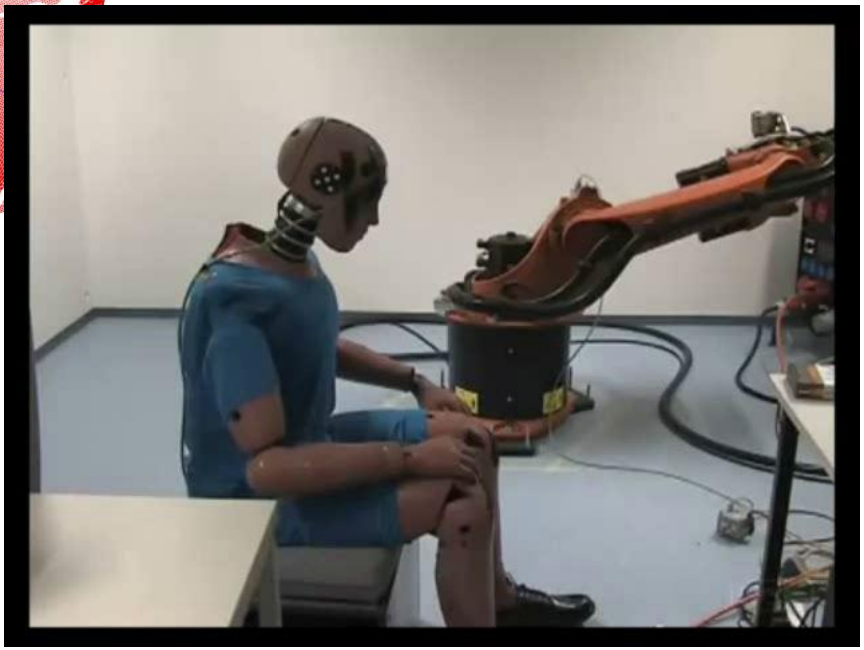
Motivation: Other Object Representations

Motivation: Other Object Representations

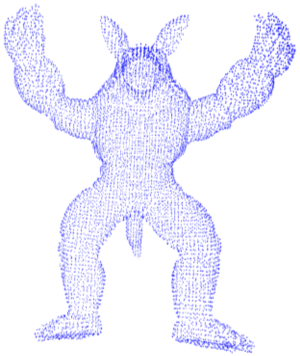


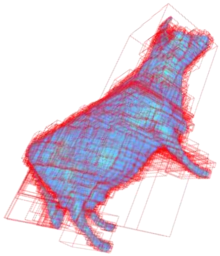
Motivation: Other Object Representations



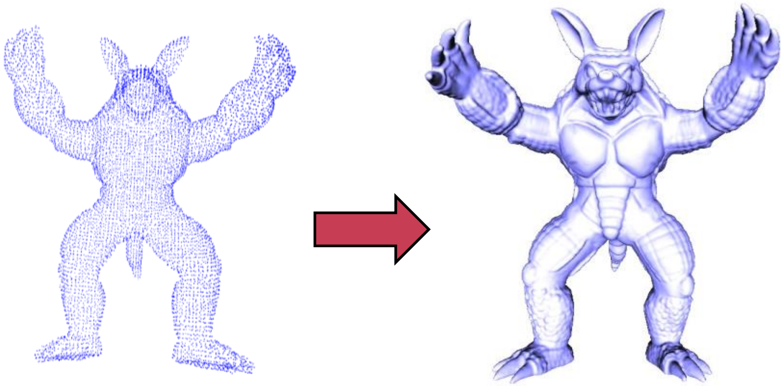


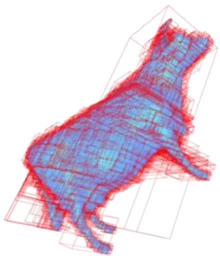
Previous Works



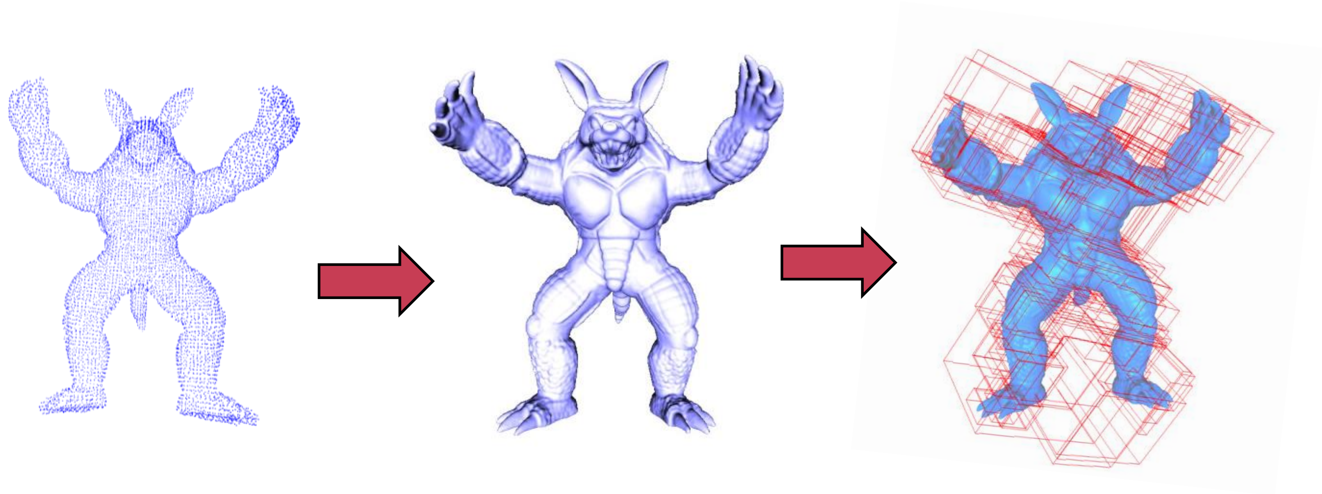


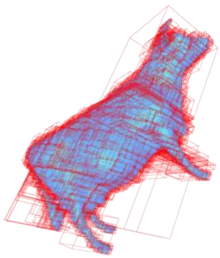
Previous Works



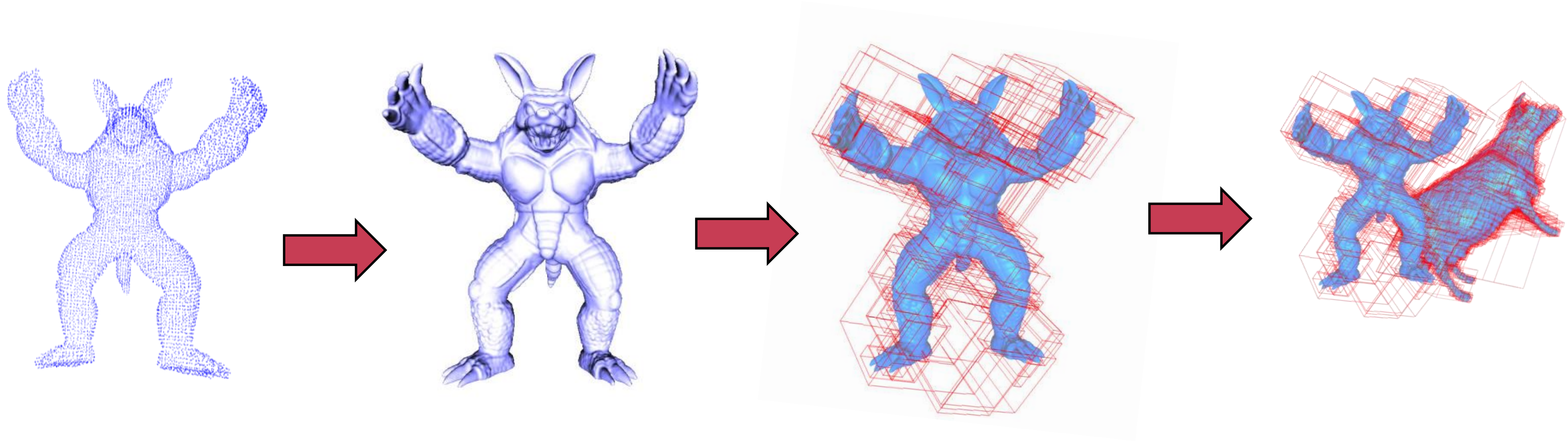


Previous Works

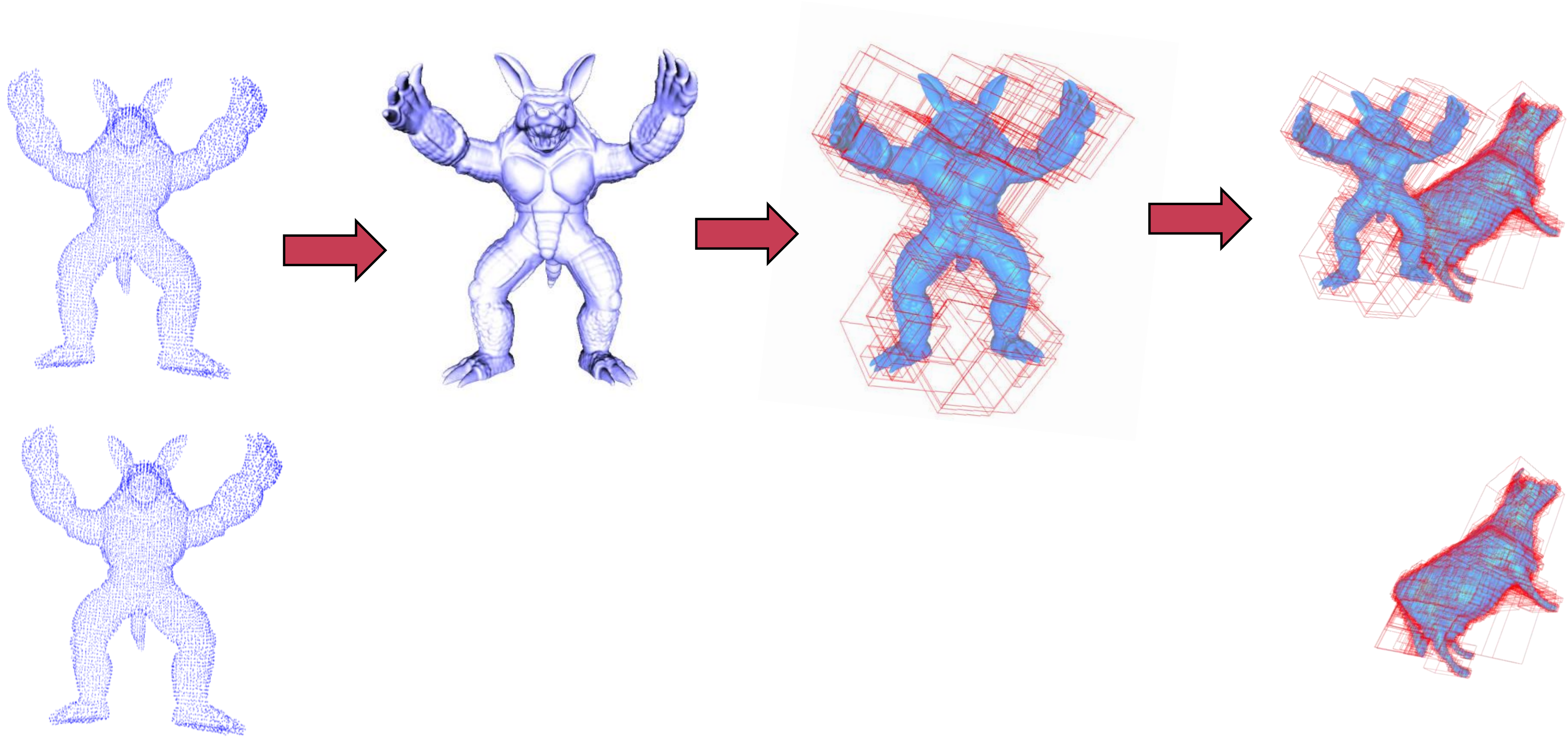




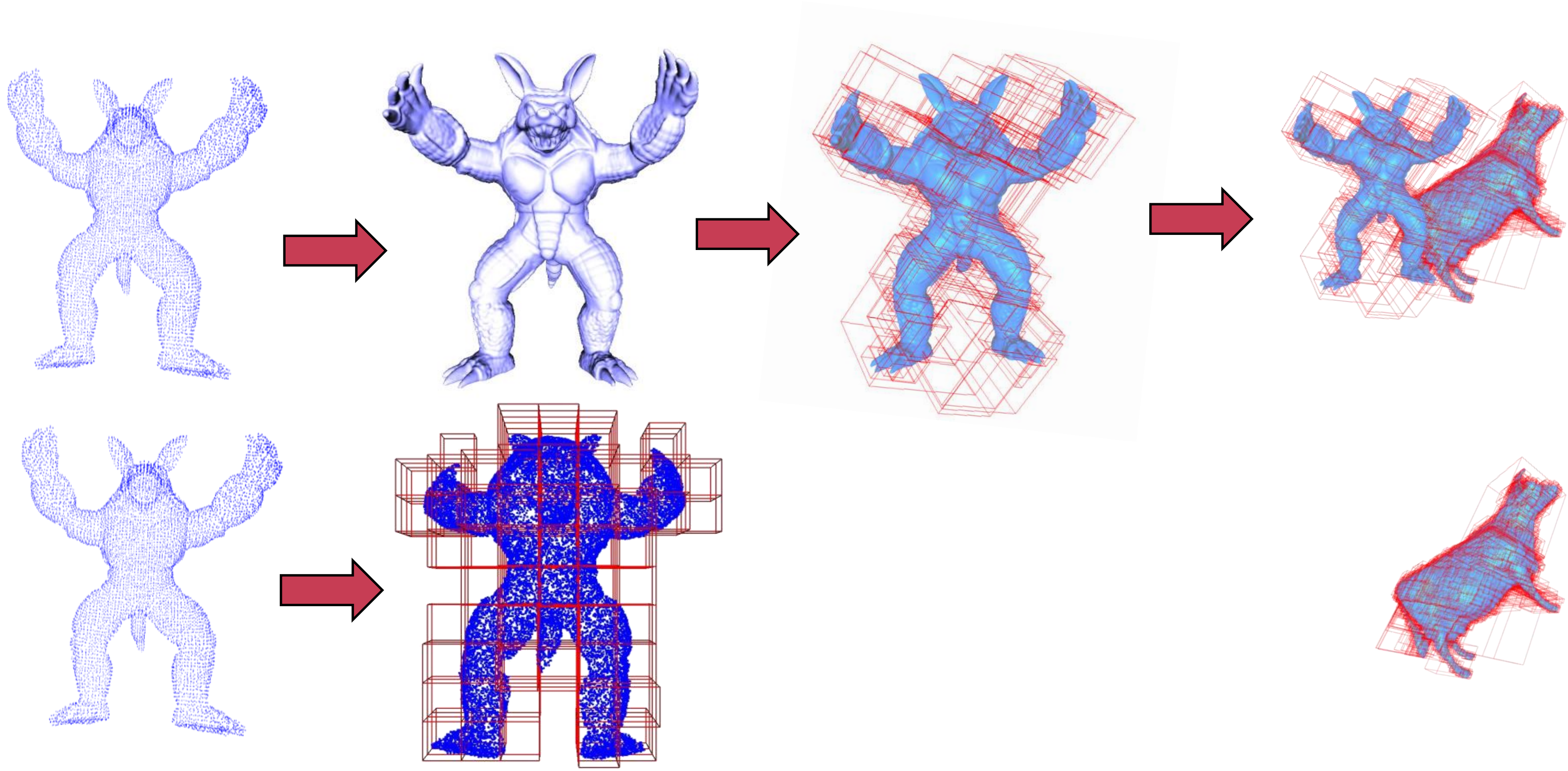
Previous Works



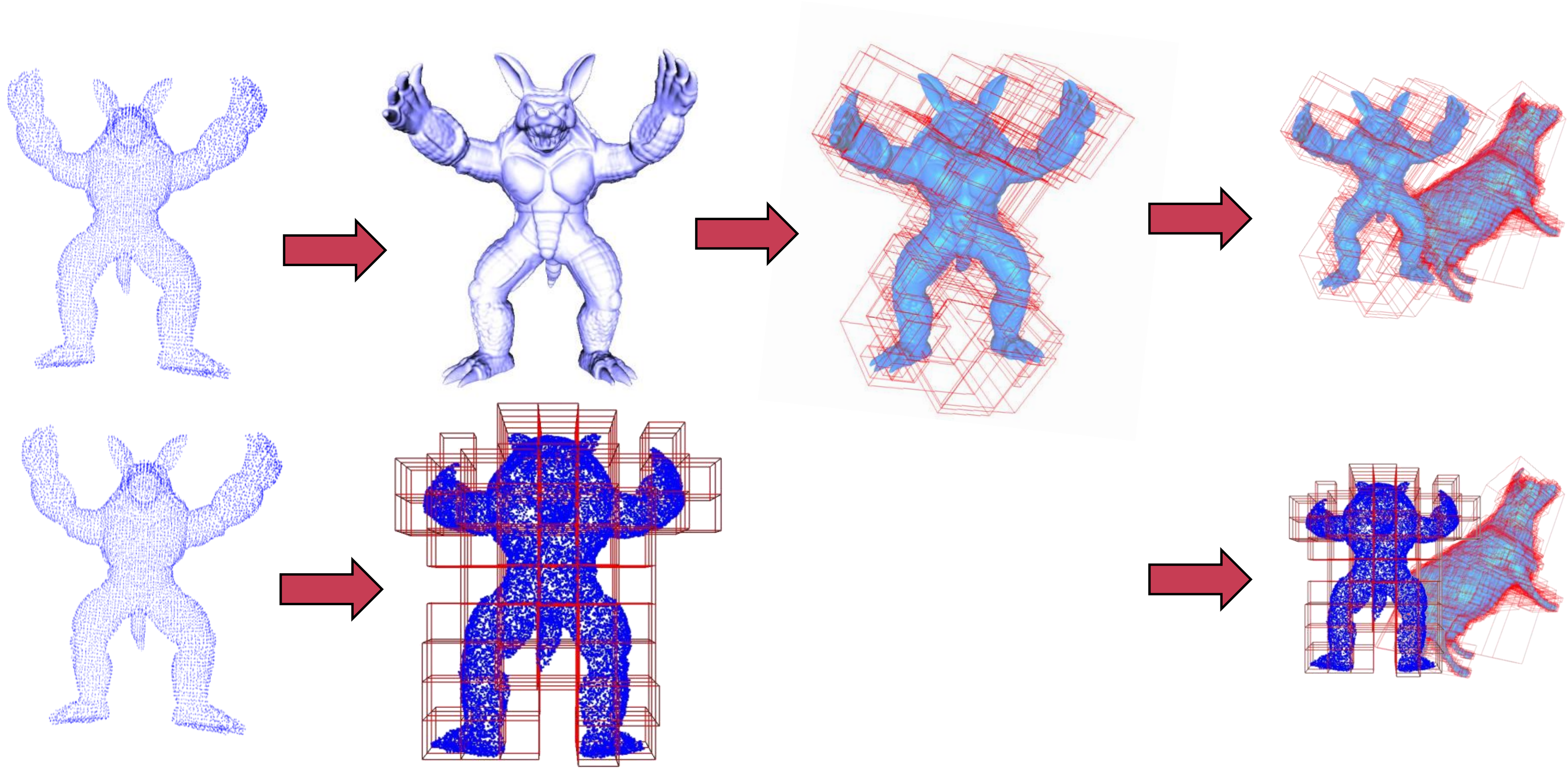
Previous Works



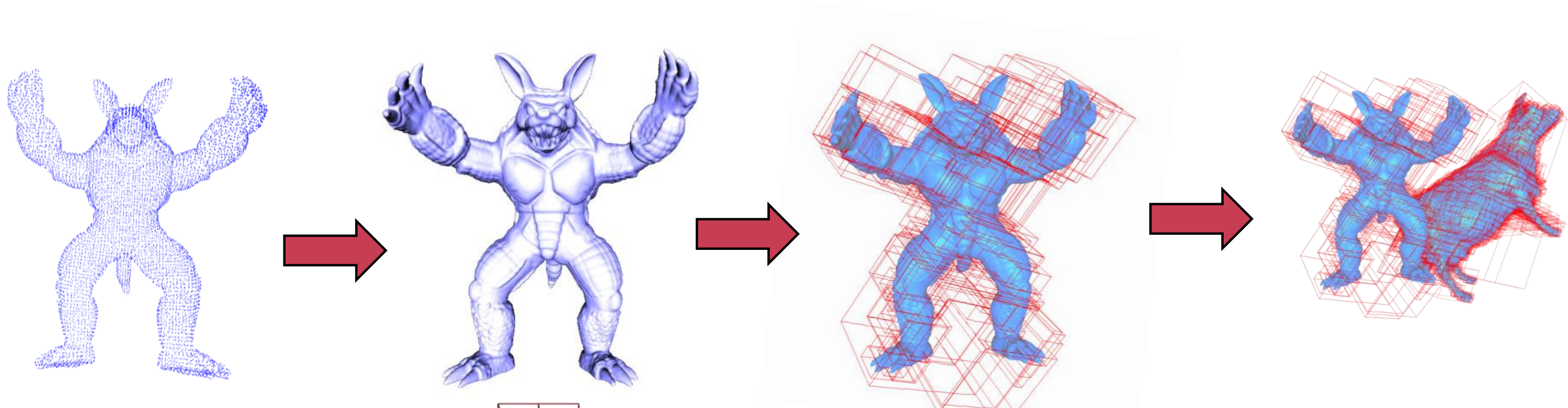
Previous Works

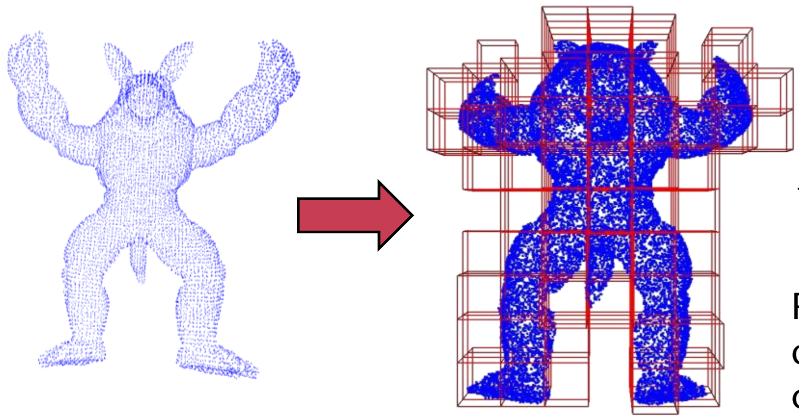


Previous Works



Previous Works

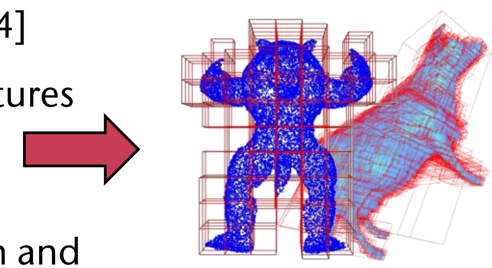




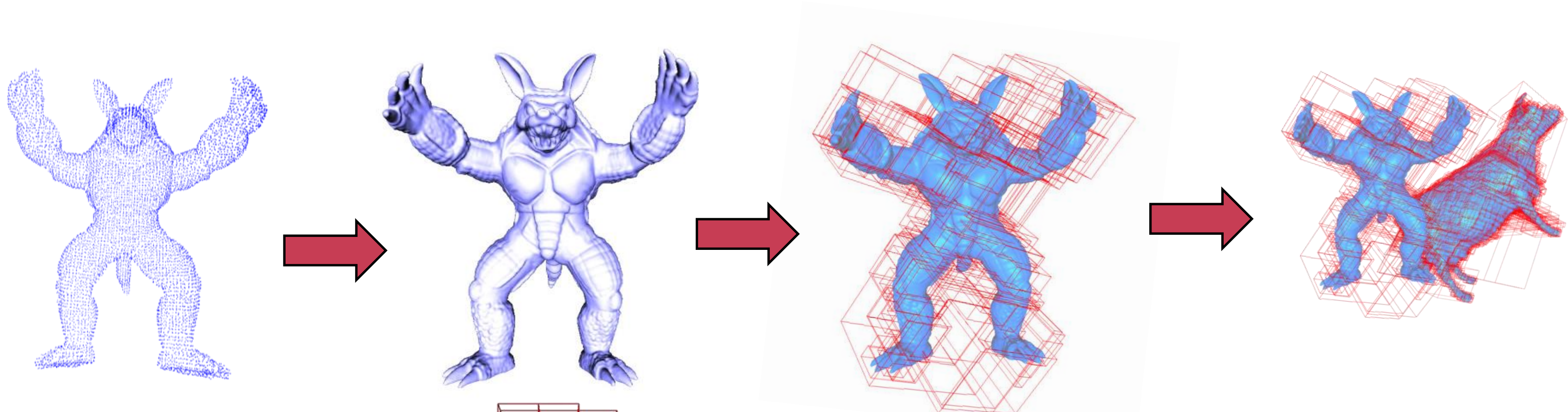
Point cloud collision detection
[Klein and Zachmann, 2004]

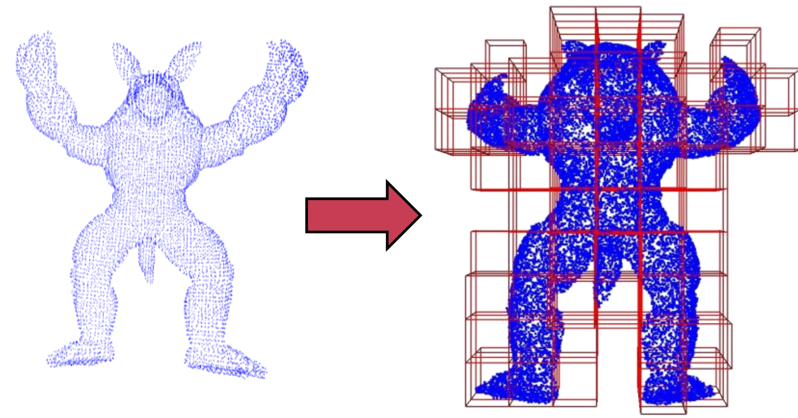
New Geometric Data Structures
for Collision Detection
[Weller, 2012]

Real-time collision detection and
distance computation on point
cloud sensor data
[Pan et. Al., 2013]



Previous Works

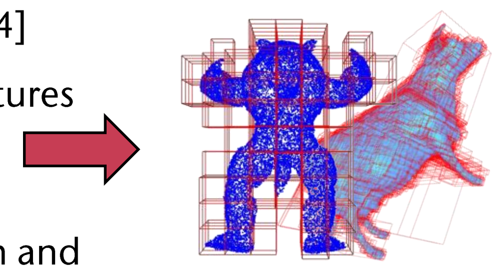


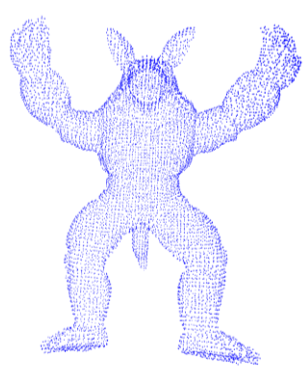


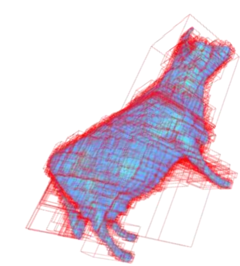
Point cloud collision detection
[Klein and Zachmann, 2004]

New Geometric Data Structures
for Collision Detection
[Weller, 2012]

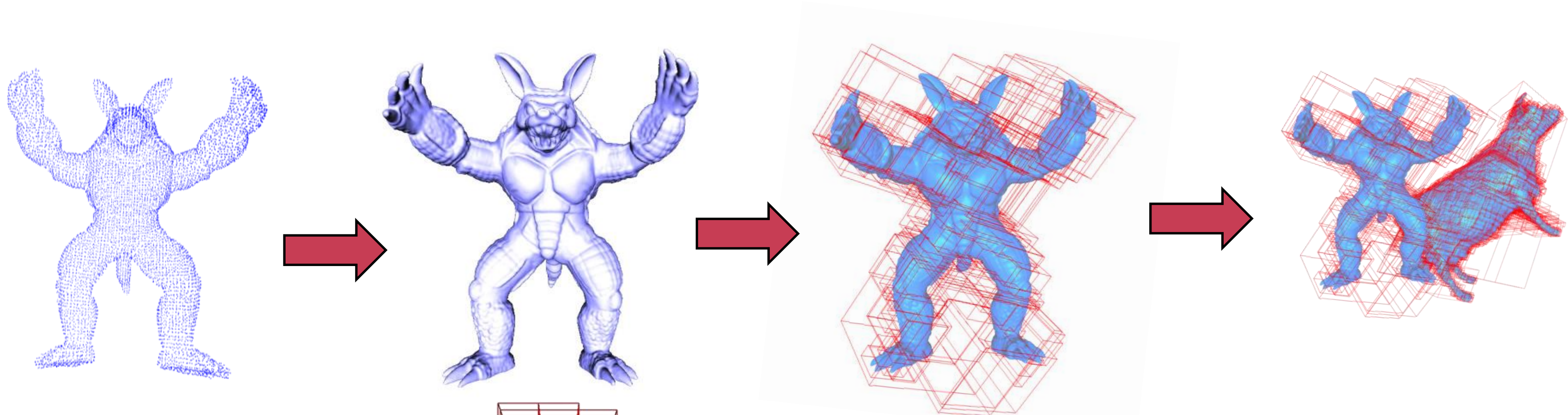
Real-time collision detection and
distance computation on point
cloud sensor data
[Pan et. Al., 2013]

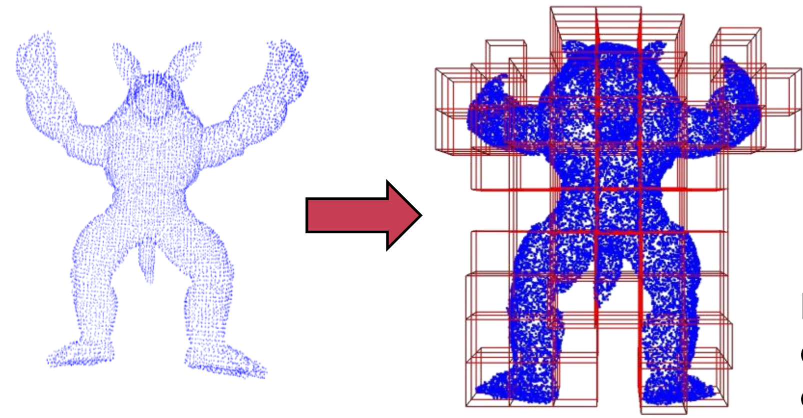






Previous Works

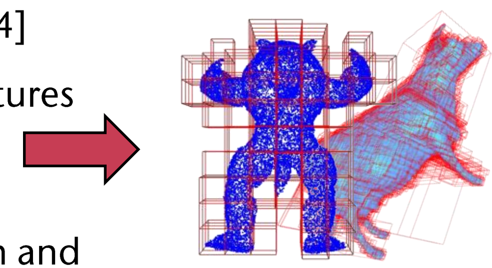


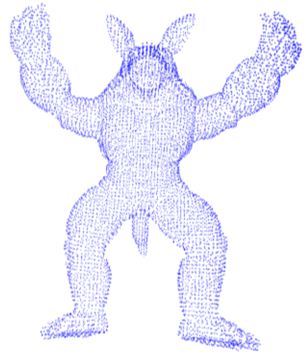


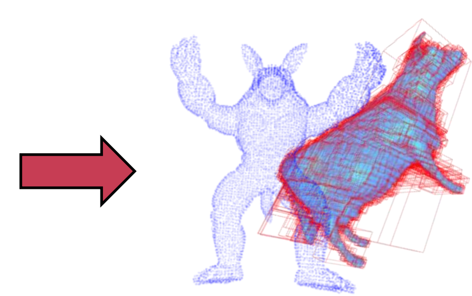
Point cloud collision detection
[Klein and Zachmann, 2004]

New Geometric Data Structures
for Collision Detection
[Weller, 2012]

Real-time collision detection and
distance computation on point
cloud sensor data
[Pan et. Al., 2013]

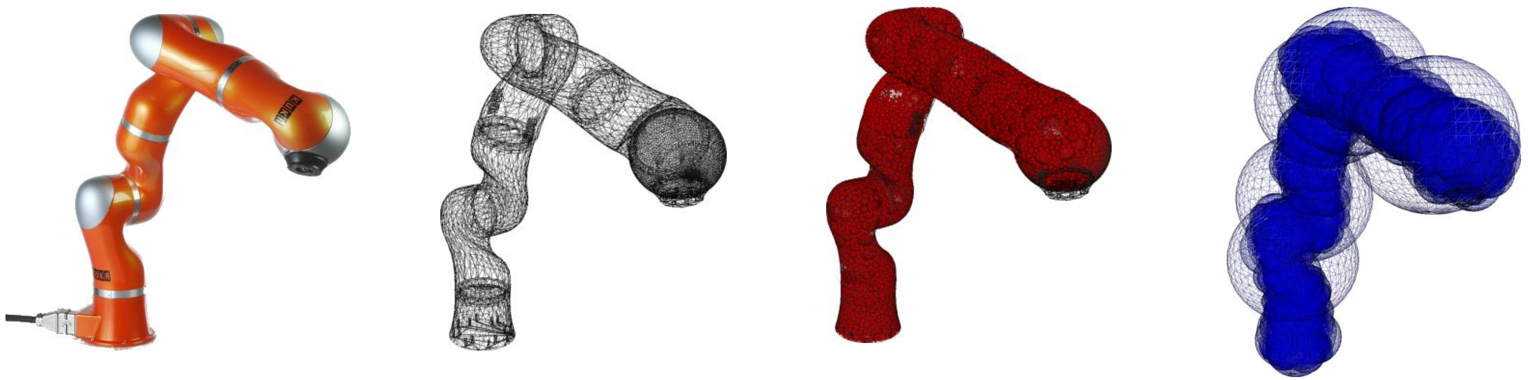






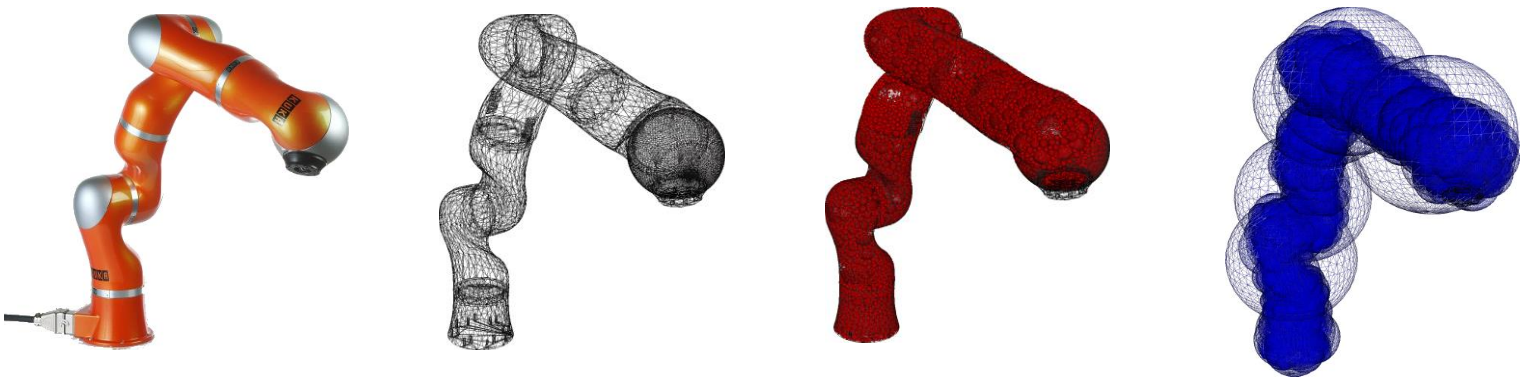
Precondition

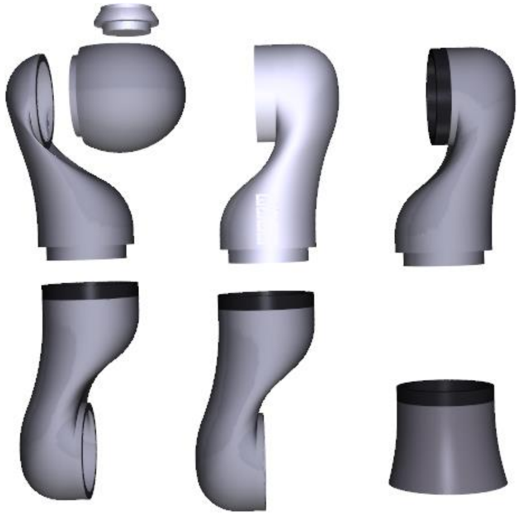
- Polygonal object representation: Inner Sphere Trees (ISTs)



Precondition

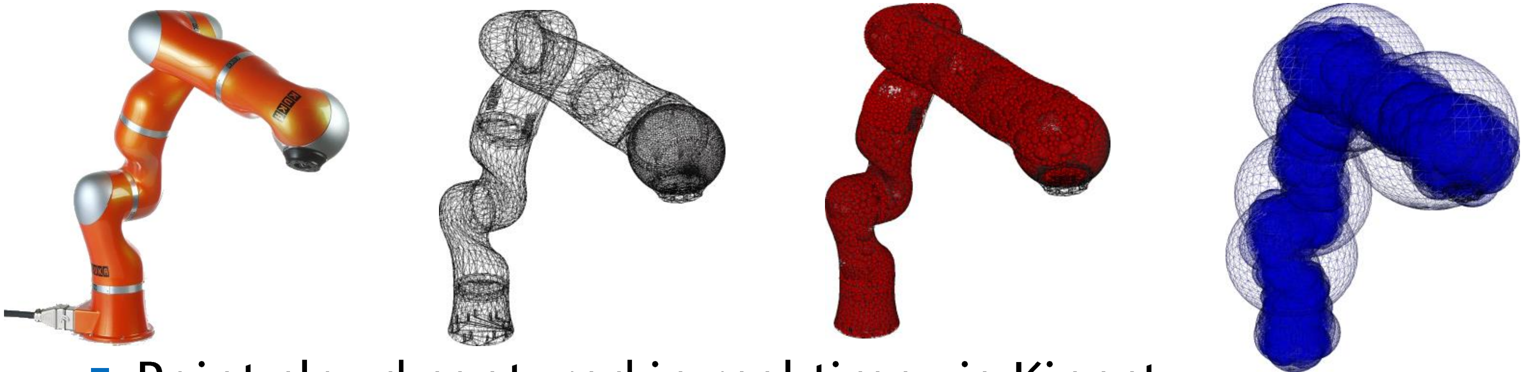
- Polygonal object representation: Inner Sphere Trees (ISTs)



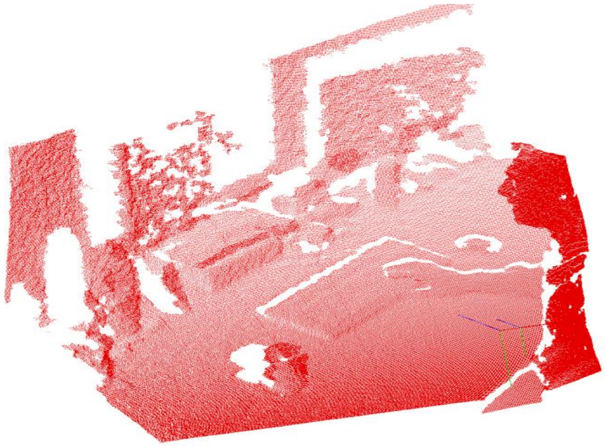


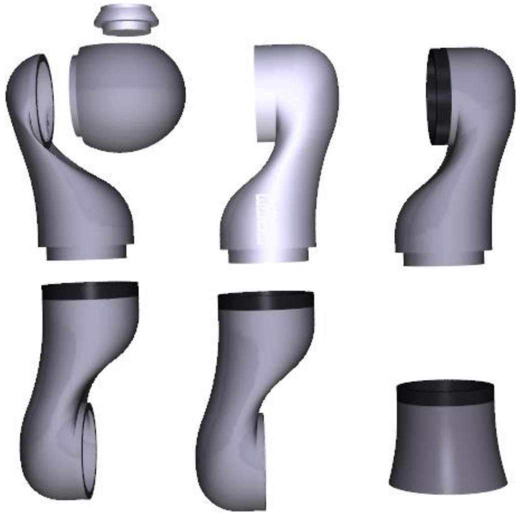
Precondition

- Polygonal object representation: Inner Sphere Trees (ISTs)

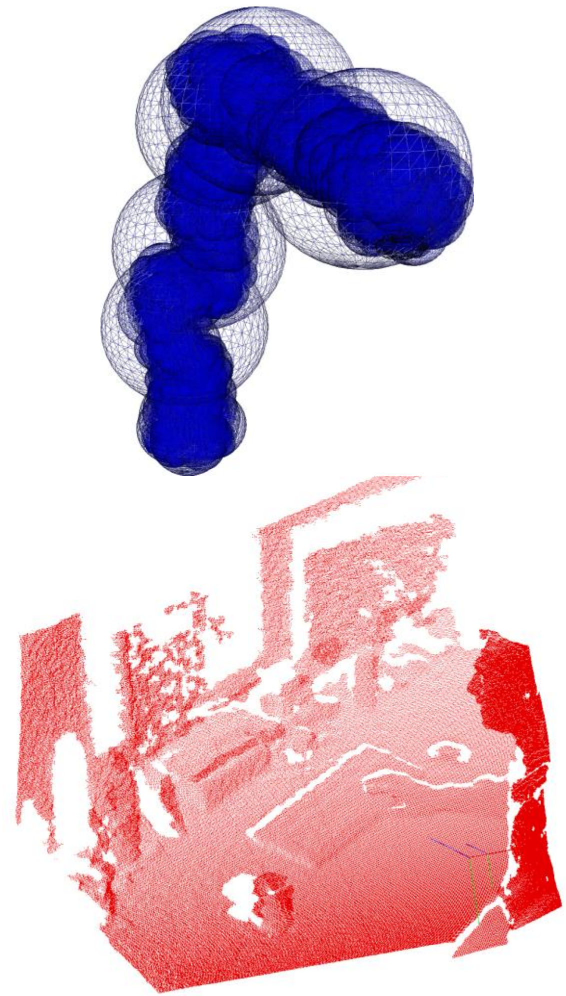


- Point cloud captured in real-time via Kinect



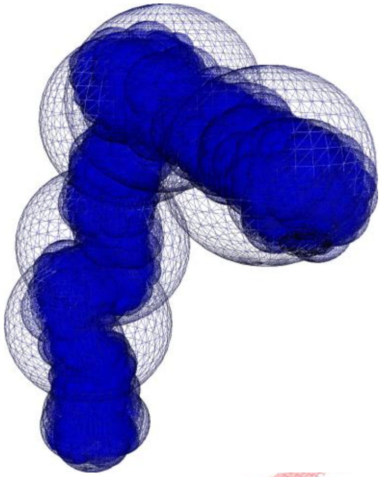


Basic Algorithm



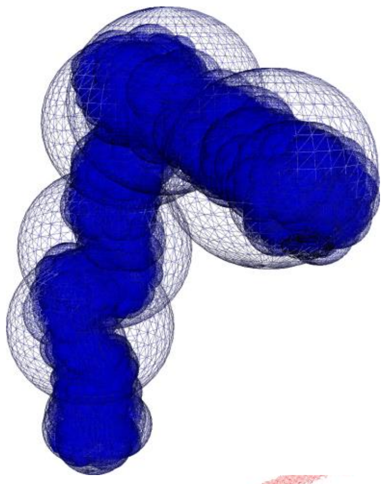
Basic Algorithm

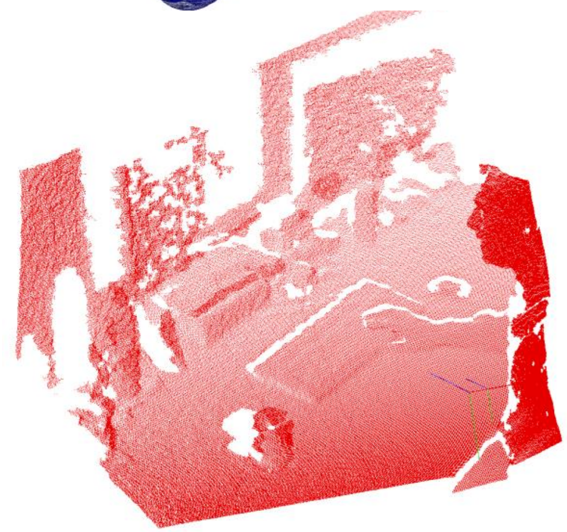
```
minDist = ∞
For each IST ∈ Robot
 For each point p ∈ Point Cloud
   getDistance( Root(IST), p, minDist)
```



Basic Algorithm

```
minDist = ∞
For each IST ∈ Robot
 For each point p ∈ Point Cloud
   getDistance( Root(IST), p, minDist)
```



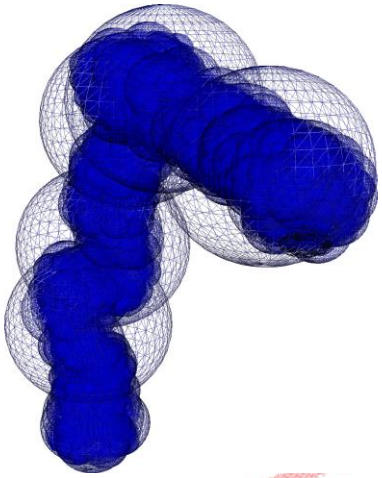


```
getDistance( Sphere s, Point p, d )
  forall the Children sc of s do
    d = distance( sc, p )
    if d < minDist then
      getDistance( sc, p, d )
```

Basic Algorithm

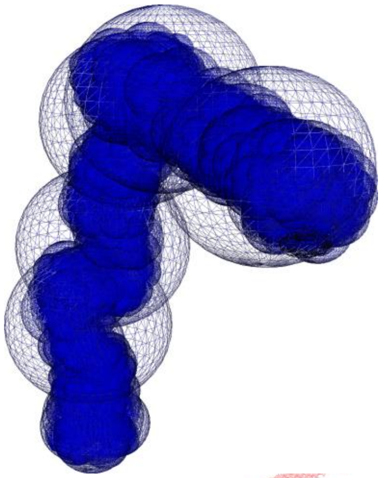
```
minDist = ∞
For each IST ∈ Robot
 For each point p ∈ Point Cloud
   getDistance( Root(IST), p, minDist)
```

```
getDistance( Sphere s, Point p, d )
  forall the Children sc of s do
    d = distance( sc, p )
    if d < minDist then
      getDistance( sc, p, d )
  if s is Leaf then
    minDist = min( d, minDist )
```



Parallel Algorithm

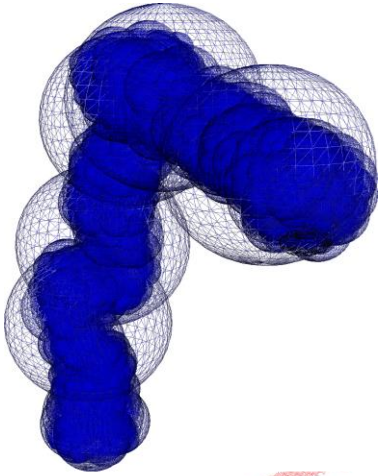
```
minDist = ∞
For each IST ∈ Robot
 For each point p ∈ Point Cloud
   getDistance( Root(IST), p, minDist)
```



```
getDistance( Sphere s, Point p, d )
  forall the Children sc of s do
    d = distance( sc, p )
    if d < minDist then
      getDistance( sc, p, d )
  if s is Leaf then
    minDist = min( d, minDist )
```

Parallel Algorithm

```
minDist = ∞
For each IST ∈ Robot
  For each point p ∈ Point Cloud
    getDistance( Root(IST), p, minDist)
```

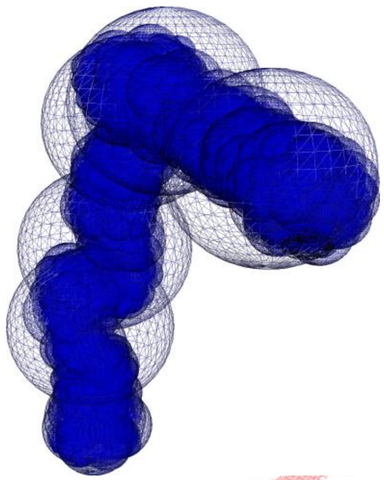


```
getDistance( Sphere s, Point p, d )
  forall the Children sc of s do
    d = distance( sc, p )
    if d < minDist then
      getDistance( sc, p, d )
  if s is Leaf then
    minDist = min( d, minDist )
```

Parallel Algorithm

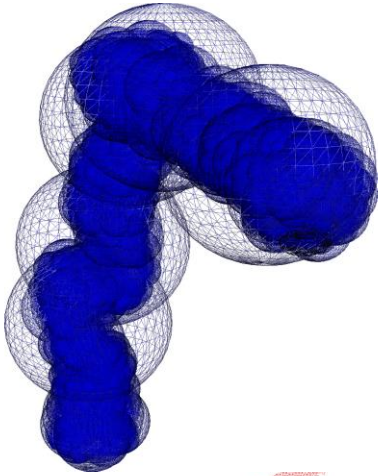
```
minDist = ∞
In Parallel ISTs ∈ Robot:
  For each point p ∈ Point Cloud
    getDistance( Root(IST), p, minDist)
```

```
getDistance( Sphere s, Point p, d )
  forall the Children sc of s do
    d = distance( sc, p )
    if d < minDist then
      getDistance( sc, p, d )
  if s is Leaf then
    minDist = min( d, minDist )
```

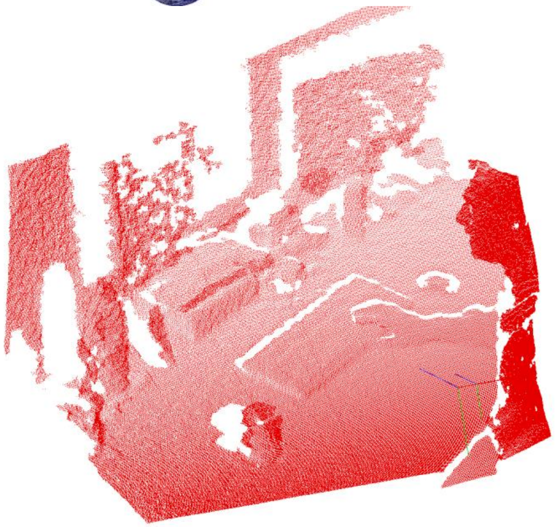


Parallel Algorithm

```
minDist = ∞
In Parallel ISTs ∈ Robot:
 For each point p ∈ Point Cloud
   getDistance( Root(IST), p, minDist)
```

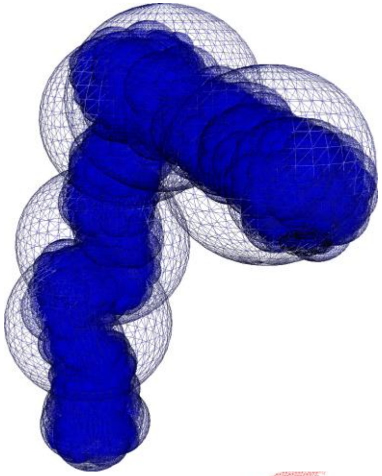


```
getDistance( Sphere s, Point p, d )
  forall the Children sc of s do
    d = distance( sc, p )
    if d < minDist then
      getDistance( sc, p, d )
  if s is Leaf then
    minDist = min( d, minDist )
```

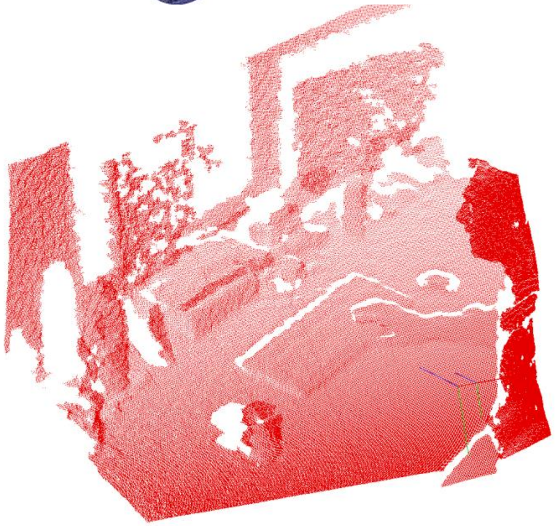


Parallel Algorithm

```
minDist = ∞
In Parallel ISTs ∈ Robot:
 In Parallel point p ∈ Point Cloud:
   getDistance( Root(IST), p, minDist)
```

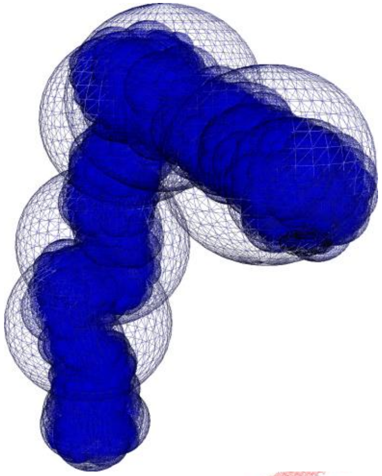


```
getDistance( Sphere s, Point p, d )
  forall the Children sc of s do
    d = distance( sc, p )
    if d < minDist then
      getDistance( sc, p, d )
  if s is Leaf then
    minDist = min( d, minDist )
```



Parallel Algorithm

```
minDist = ∞
In Parallel ISTs ∈ Robot:
  In Parallel point p ∈ Point Cloud:
    getDistance( Root(IST), p, minDist)
```

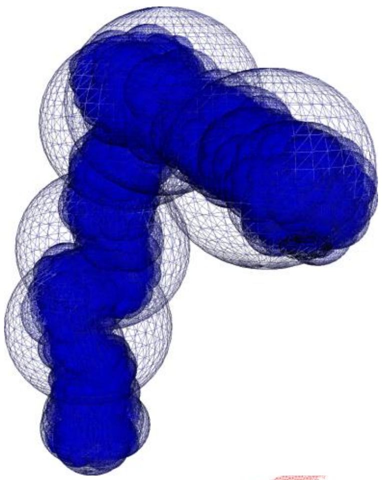


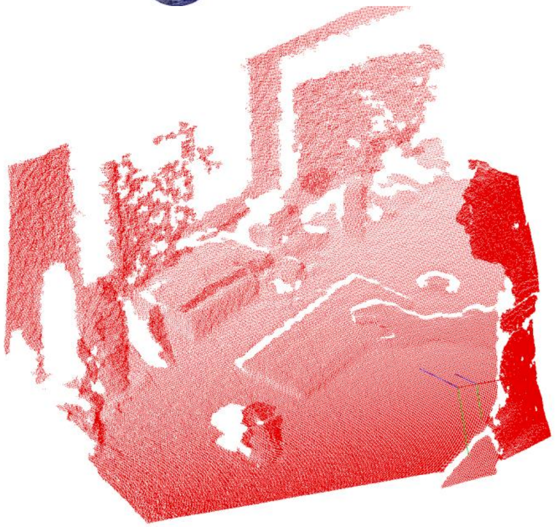
```
getDistance( Sphere s, Point p, d )
  forall the Children sc of s do
    d = distance( sc, p )
    if d < minDist then
      getDistance( sc, p, d )
  if s is Leaf then
    minDist = min( d, minDist )
```

Parallel Algorithm

```
minDist = ∞
In Parallel ISTs ∈ Robot:
 In Parallel point p ∈ Point Cloud:
   getDistance( Root(IST), p, minDist)
```

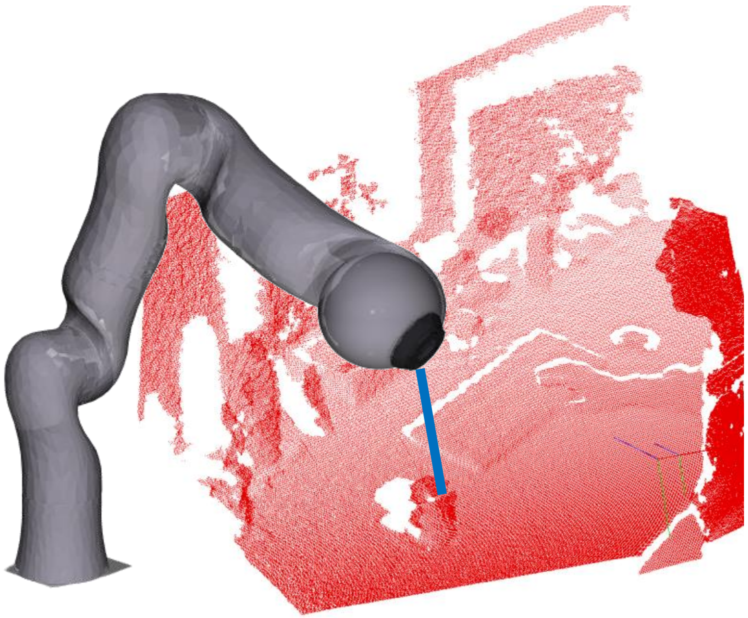
```
getDistance( Sphere s, Point p, d )
  forall the Children sc of s do
    d = distance( sc, p )
    if d < minDist then
      getDistance( sc, p, d )
  if s is Leaf then
    minDist = atomicMin( d, minDist )
```

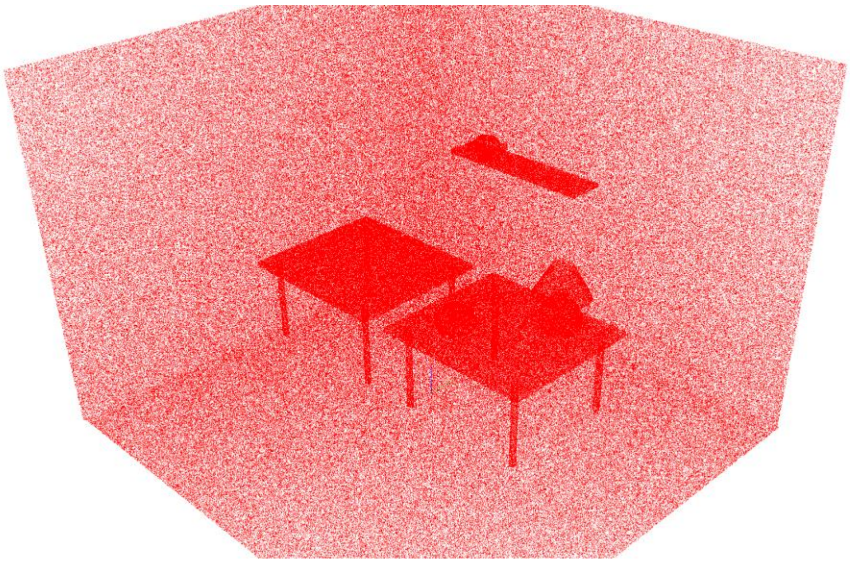




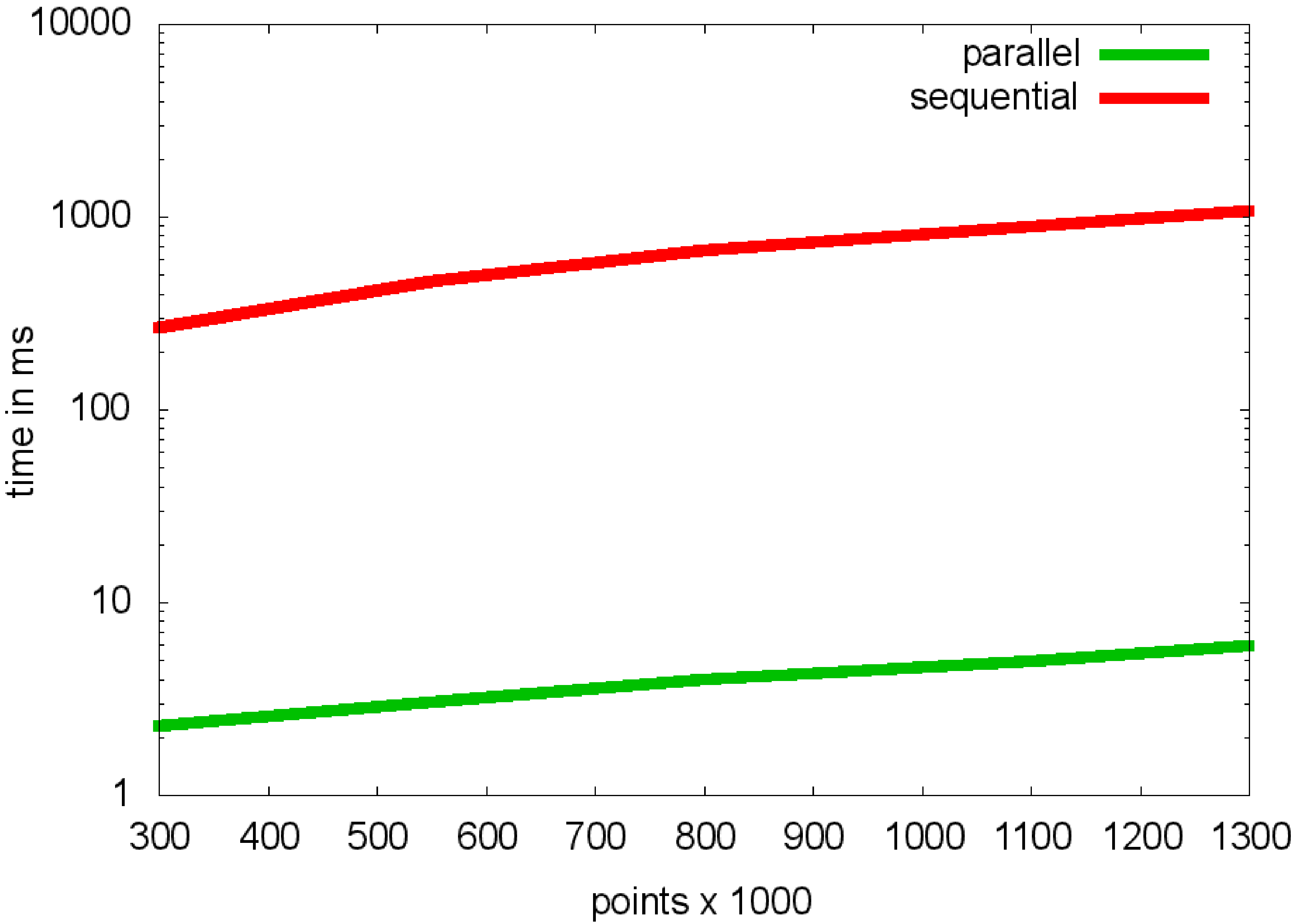
Test Scenarios

- Implemented in CUDA (5.5 & 6.0)
- Geforce GTX 780, 2GByte Memory
- Pre-recorded and artificial point clouds with up tp 5M points

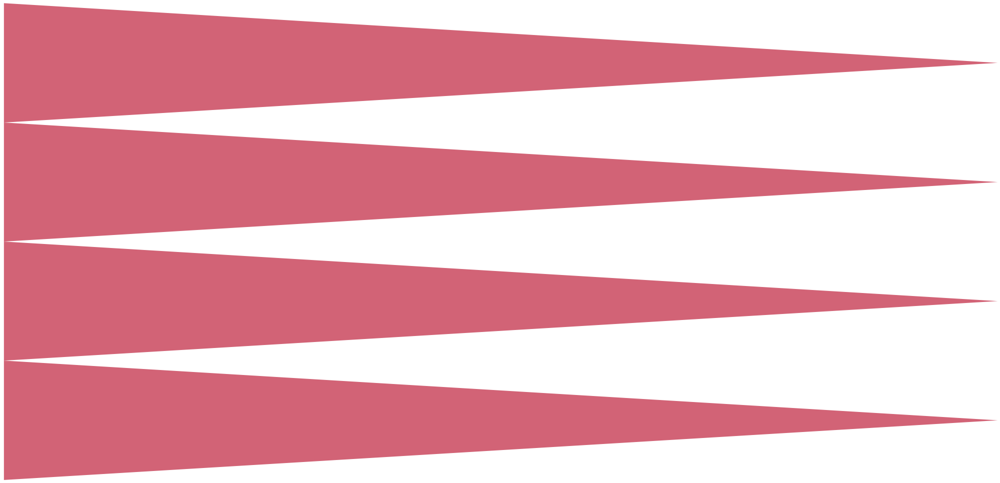




Results: Parallel vs. Sequential

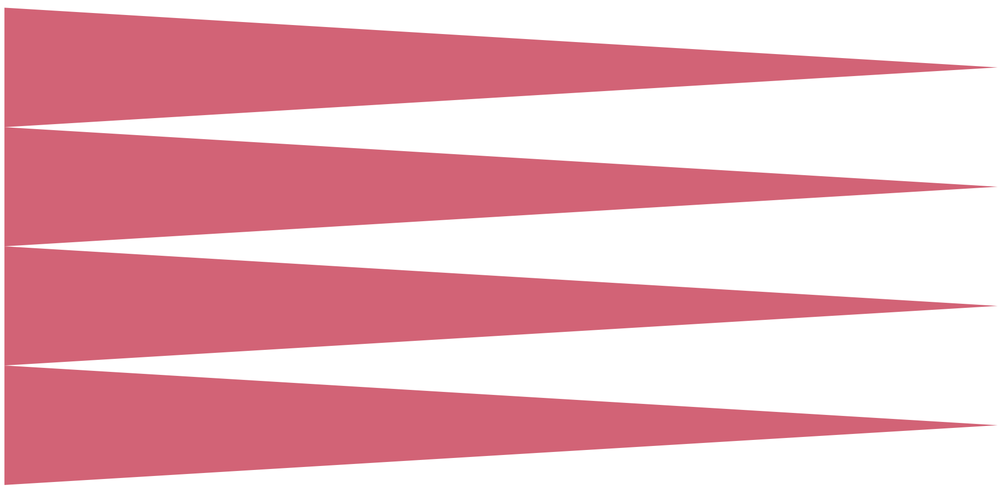


Collision Detection beyond BVHs

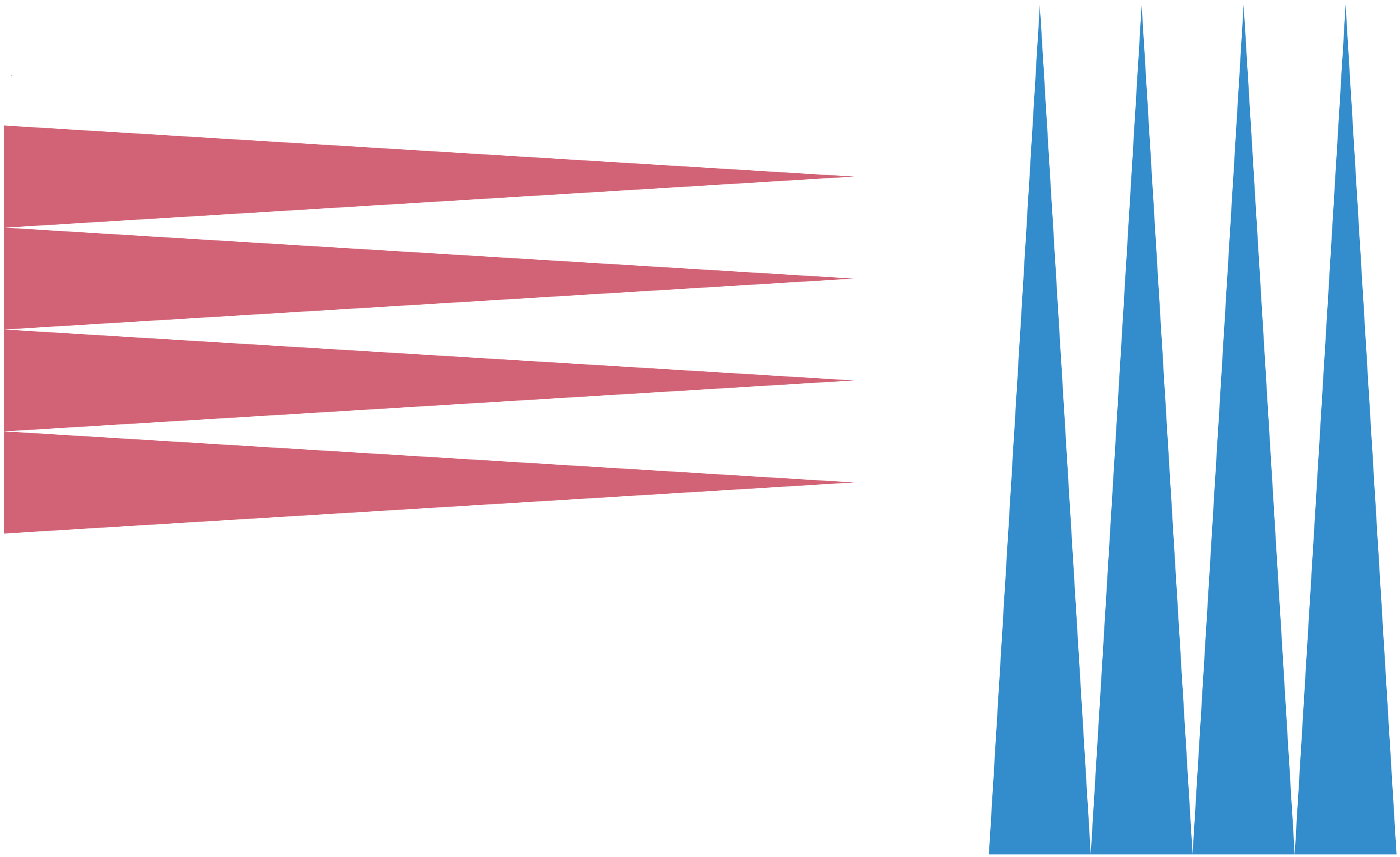


Collision Detection beyond BVHs

- BVHs are just a heuristic

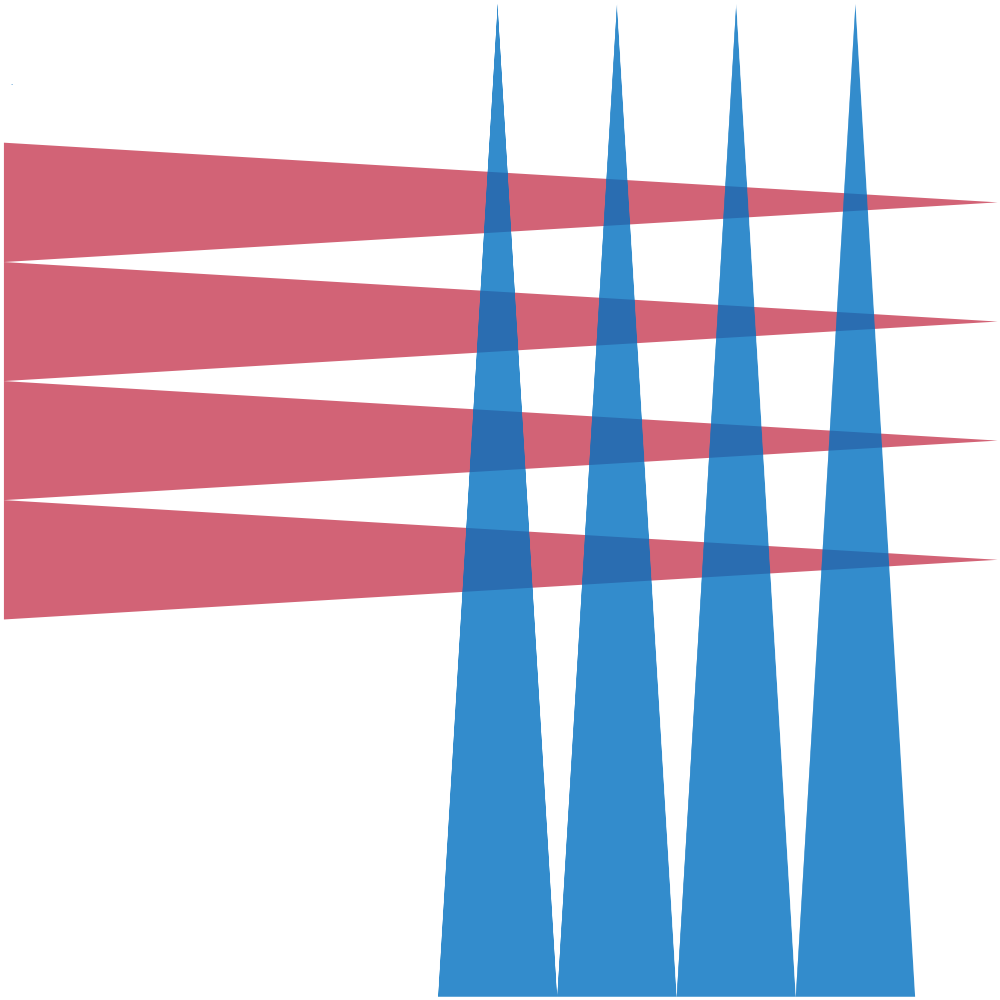


- BVHs are just a heuristic



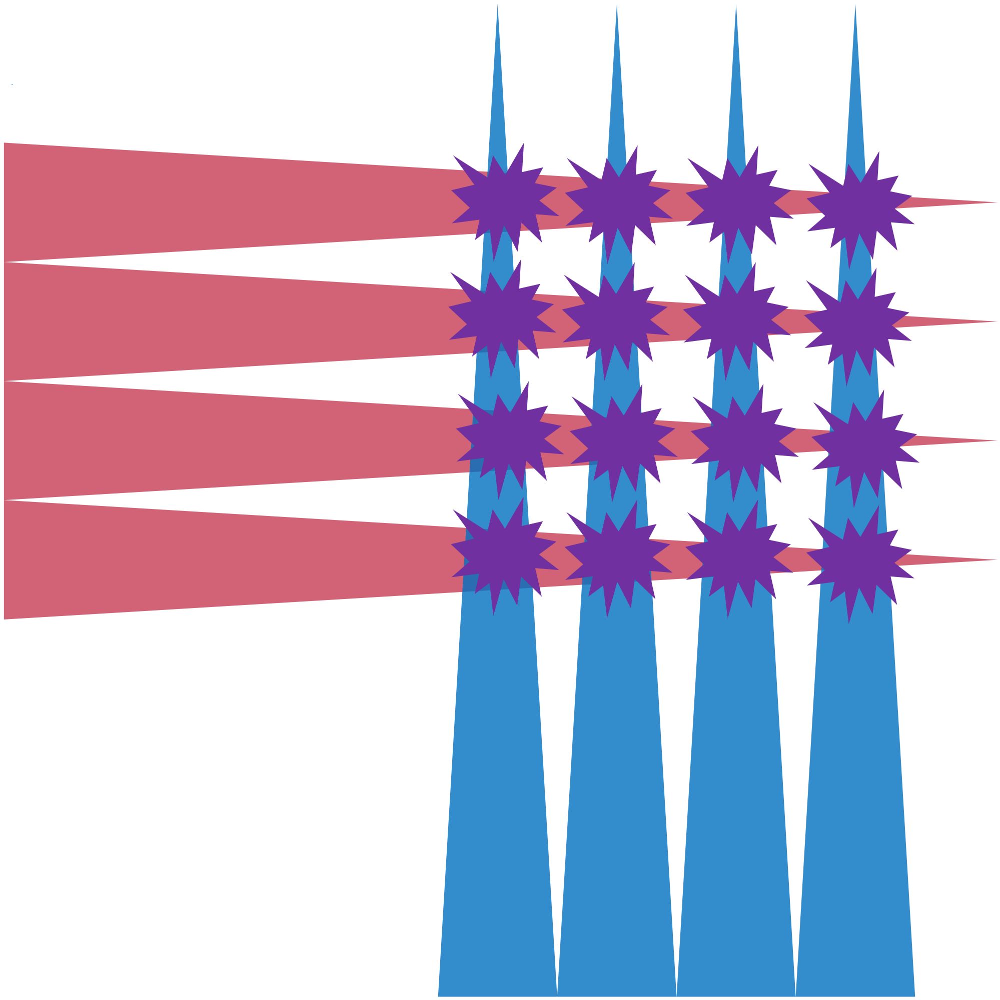
Collision Detection beyond BVHs

- BVHs are just a heuristic



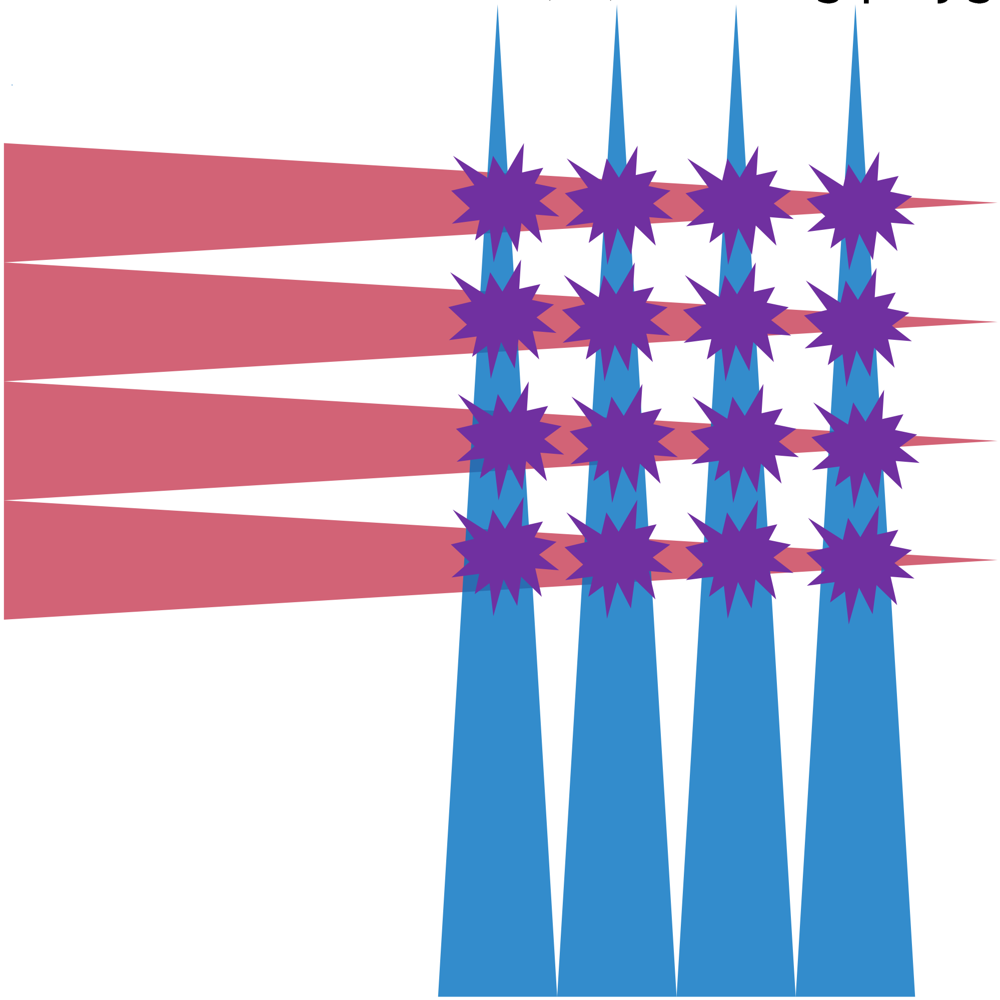
Collision Detection beyond BVHs

- BVHs are just a heuristic



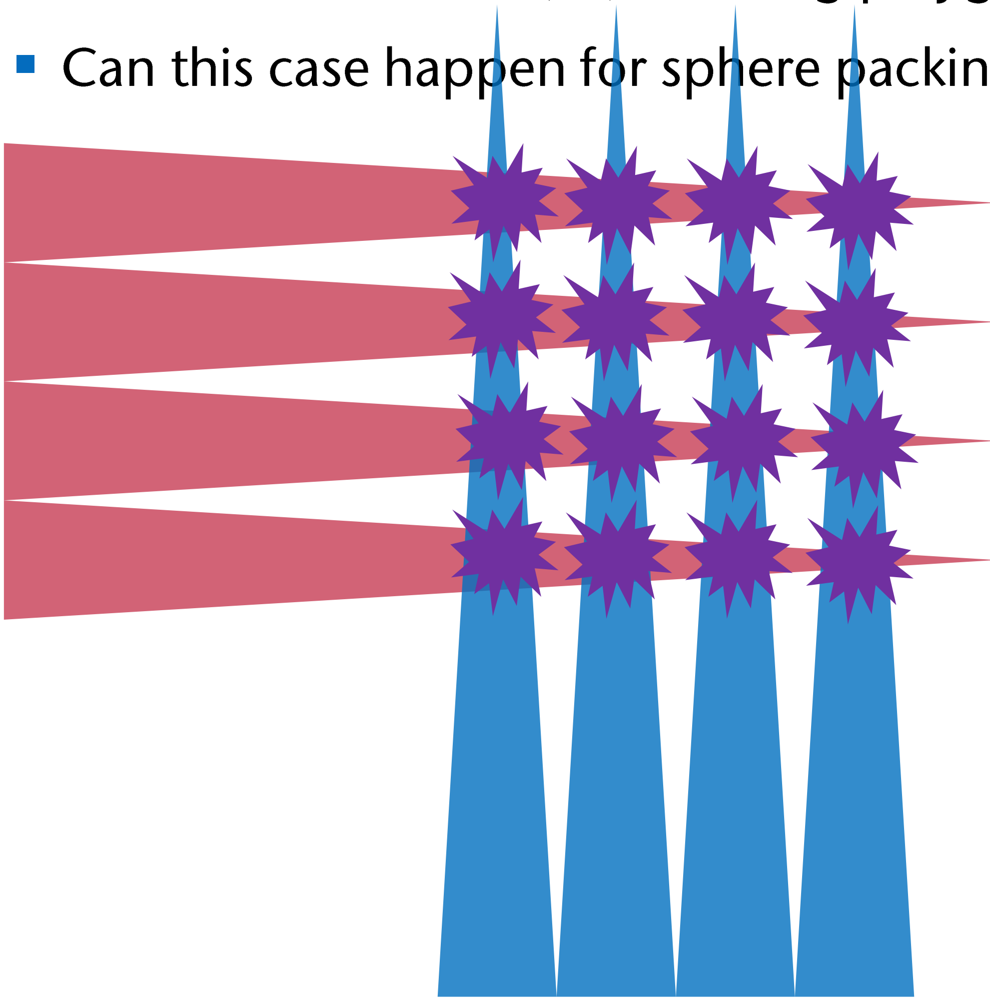
Collision Detection beyond BVHs

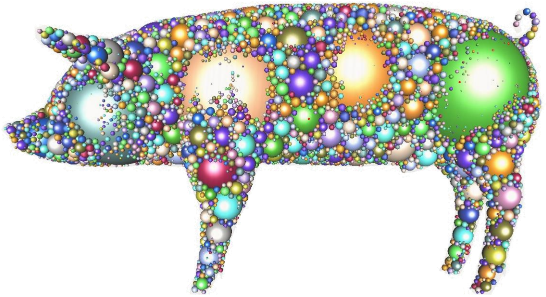
- BVHs are just a heuristic
- In the worst case: $O(n^2)$ colliding polygons



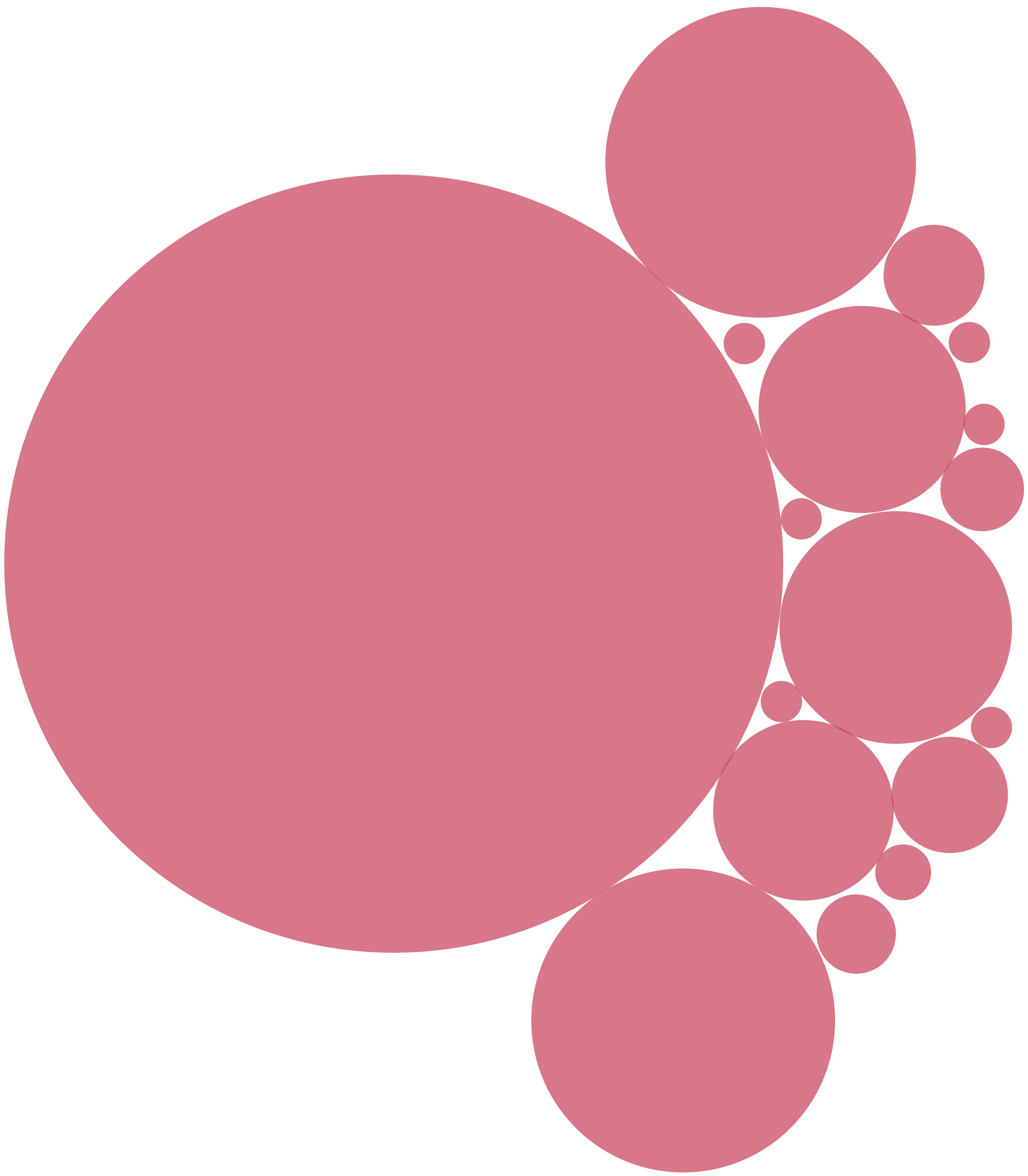
Collision Detection beyond BVHs

- BVHs are just a heuristic
- In the worst case: $O(n^2)$ colliding polygons
- Can this case happen for sphere packings?

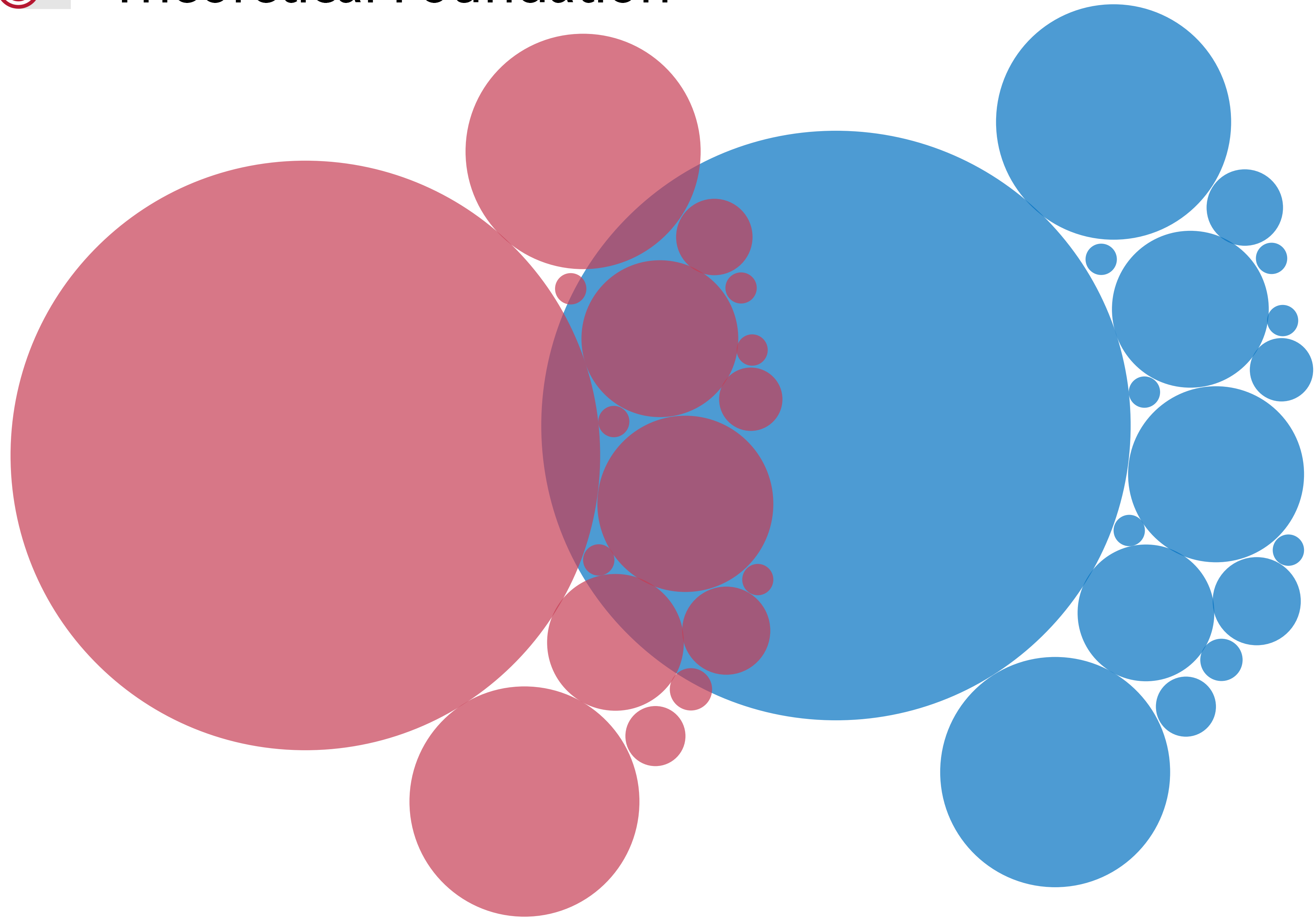




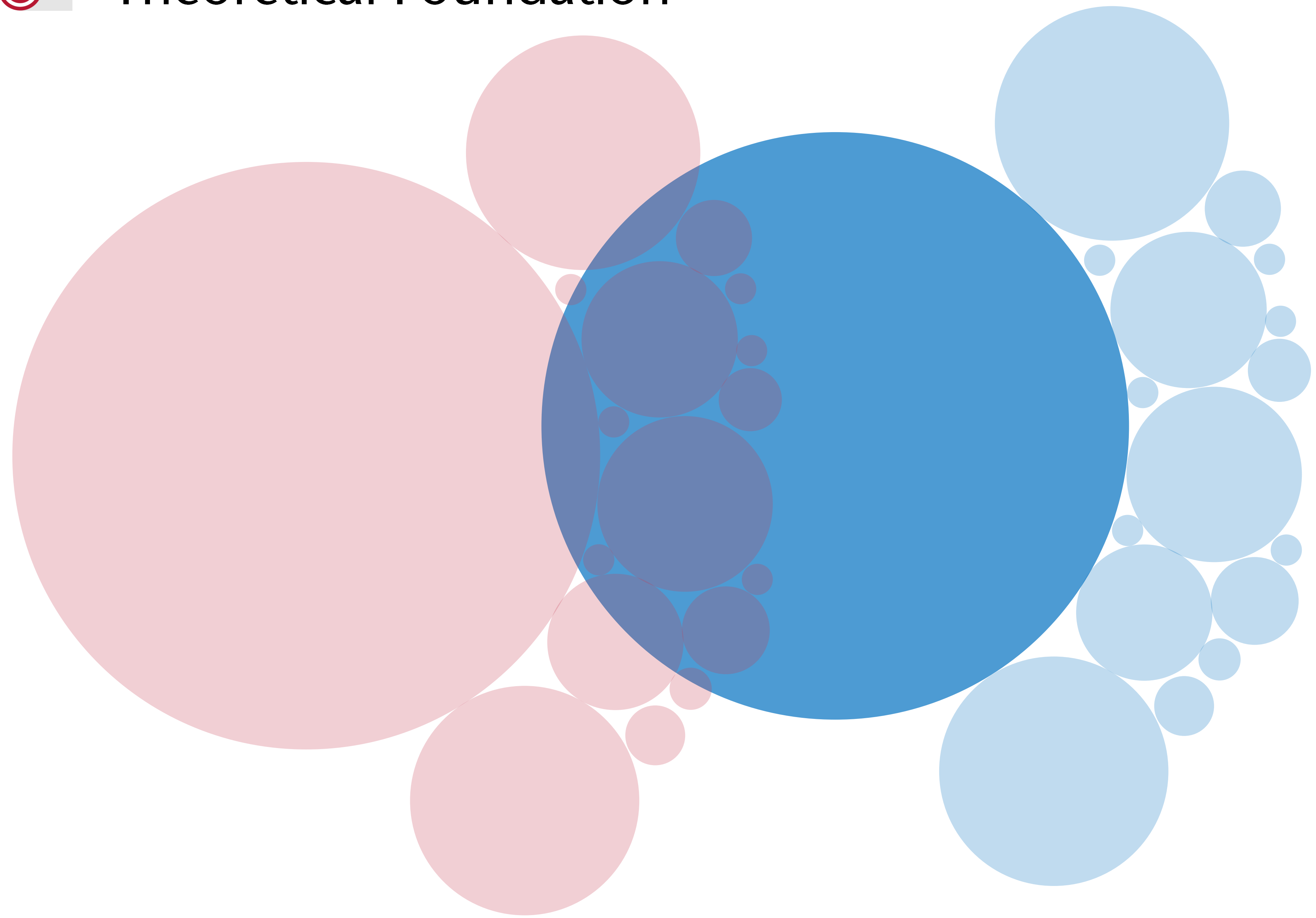
Theoretical Foundation



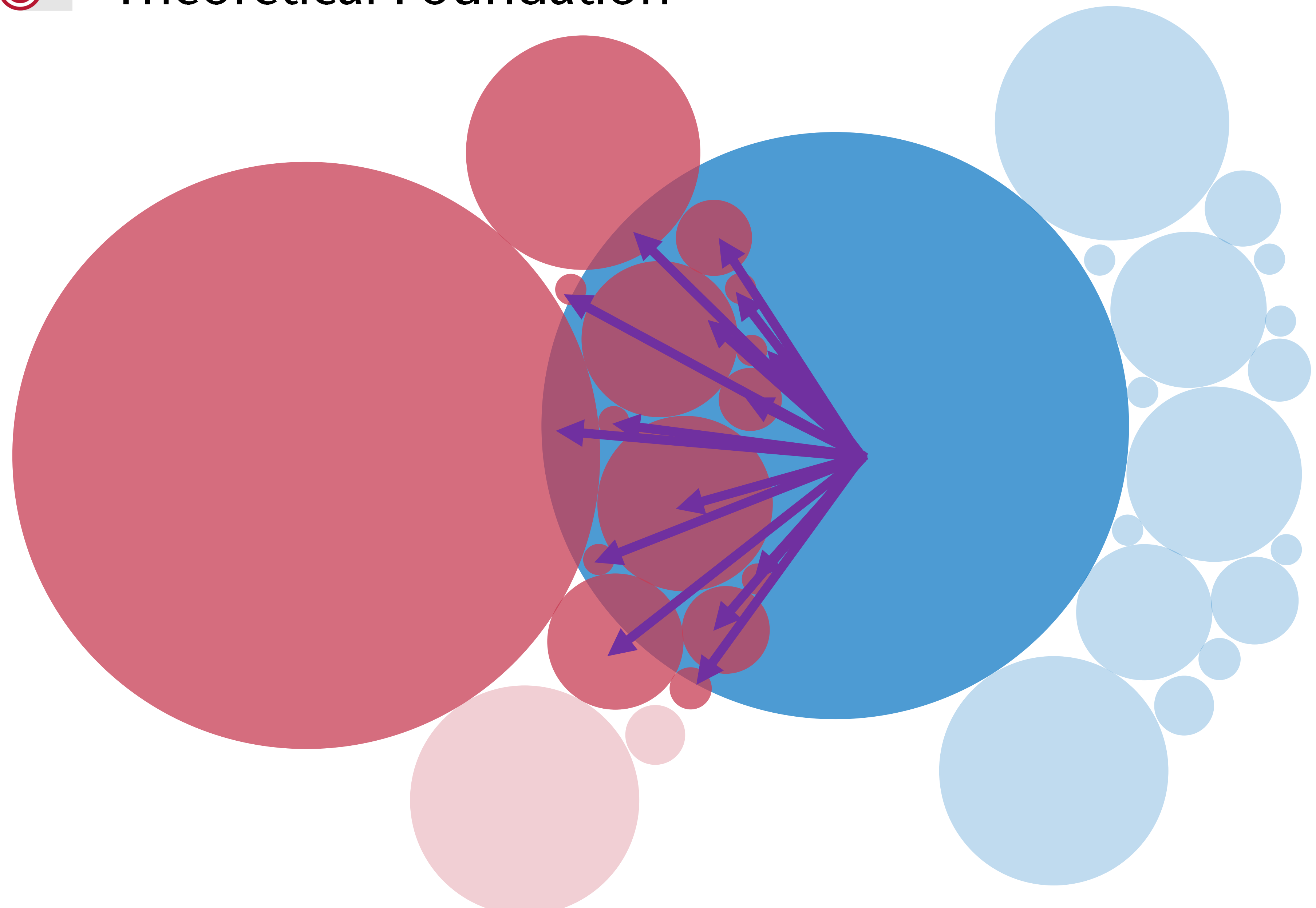
Theoretical Foundation



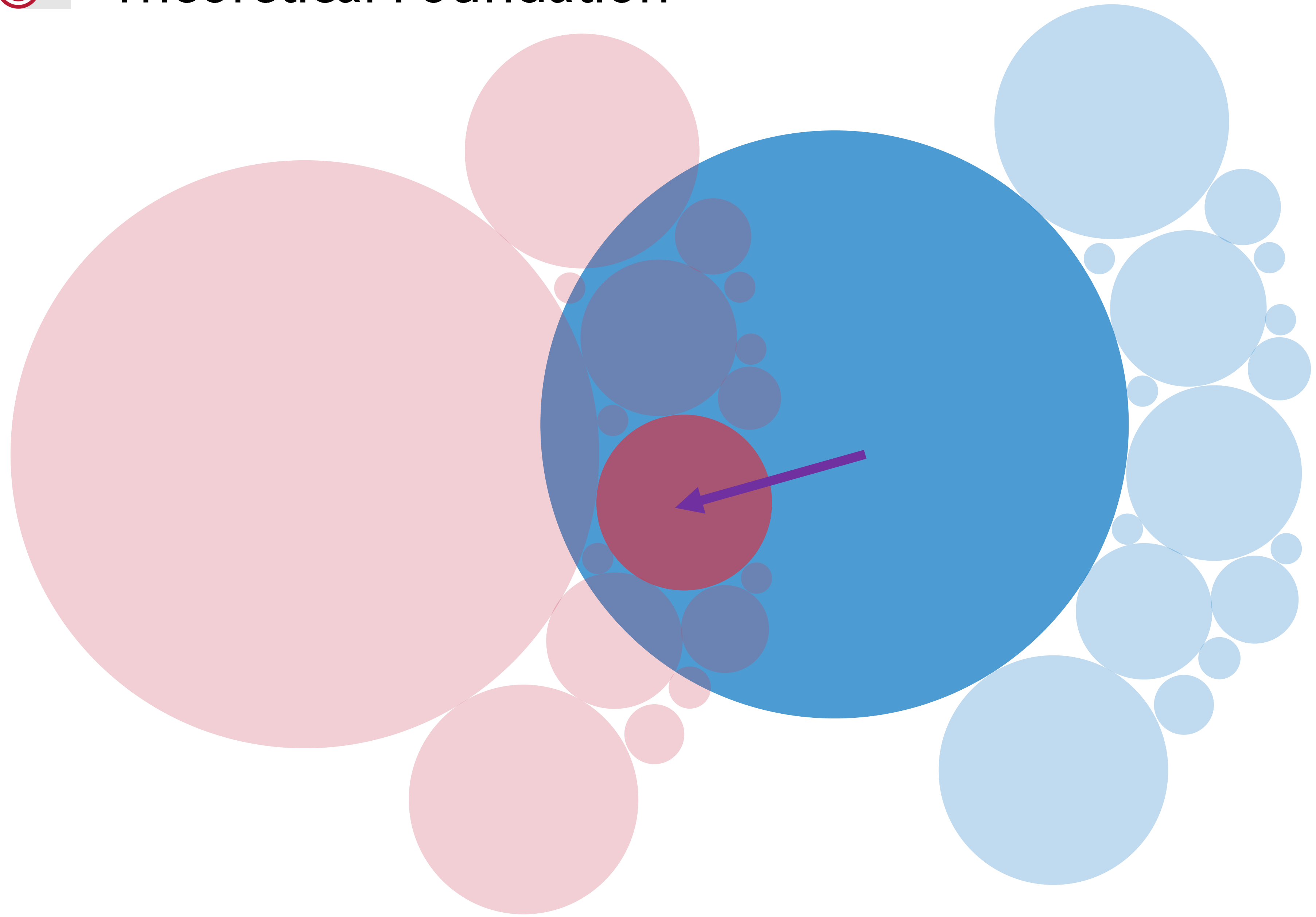
Theoretical Foundation



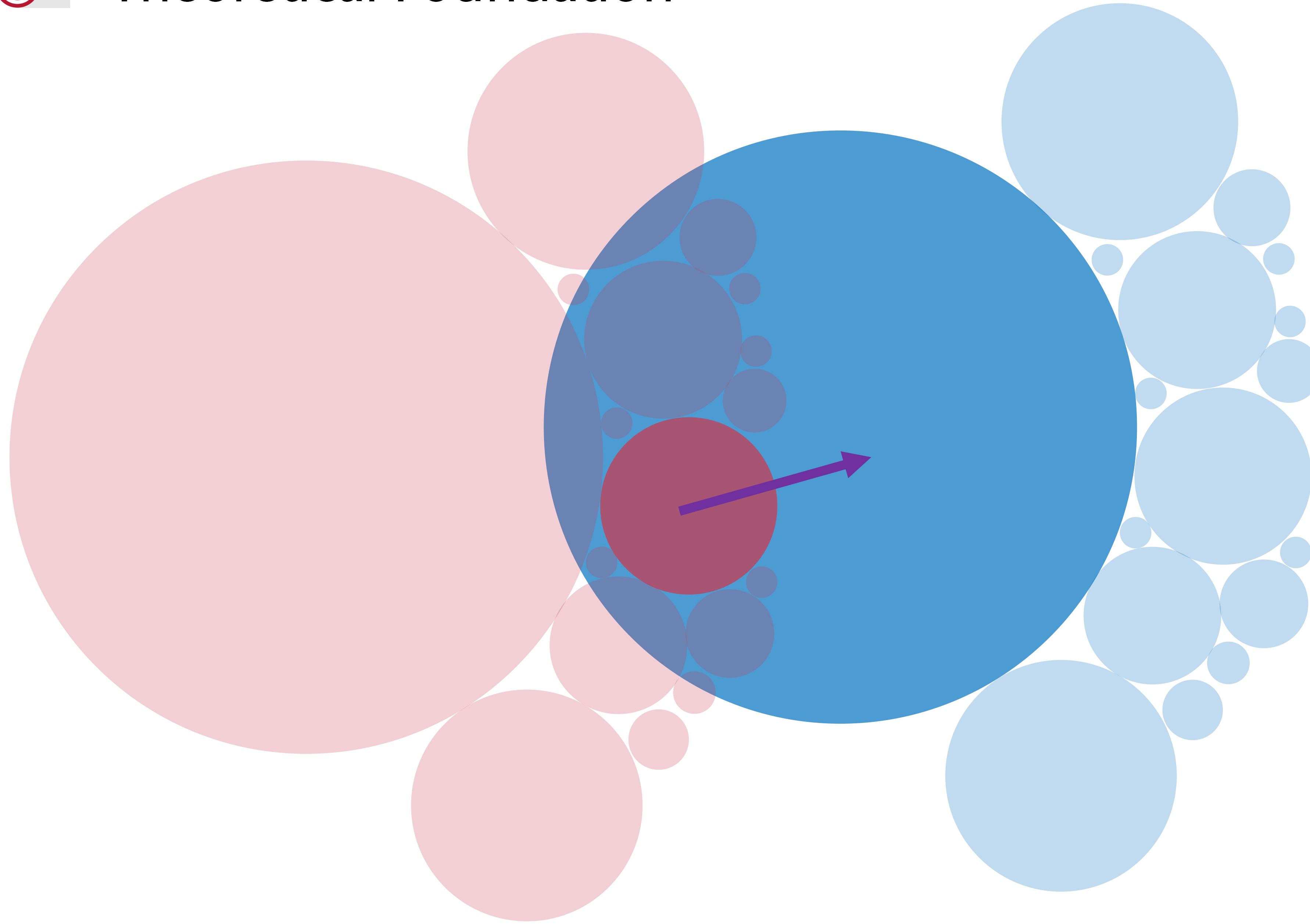
Theoretical Foundation



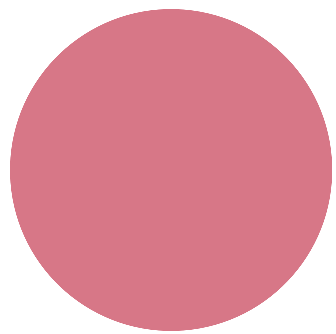
Theoretical Foundation



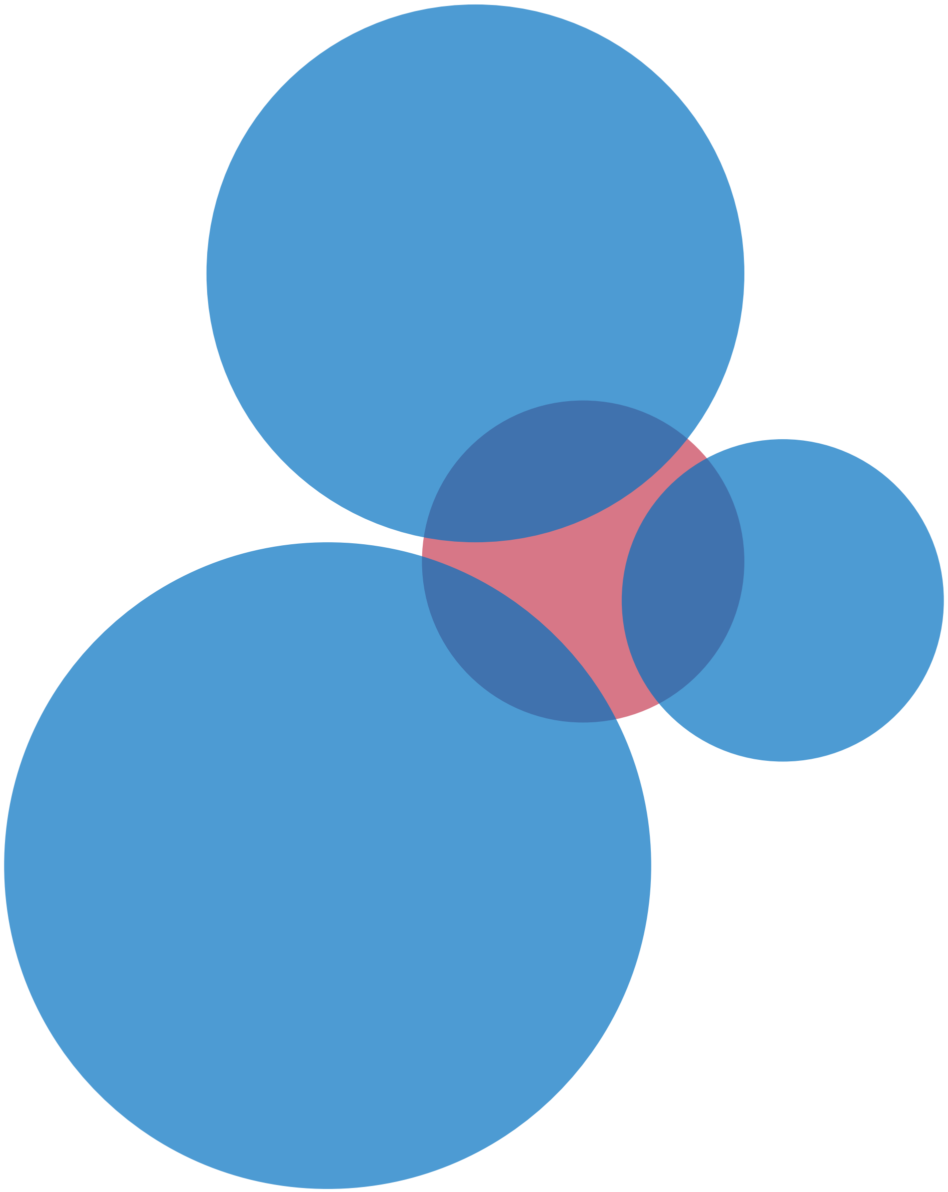
Theoretical Foundation



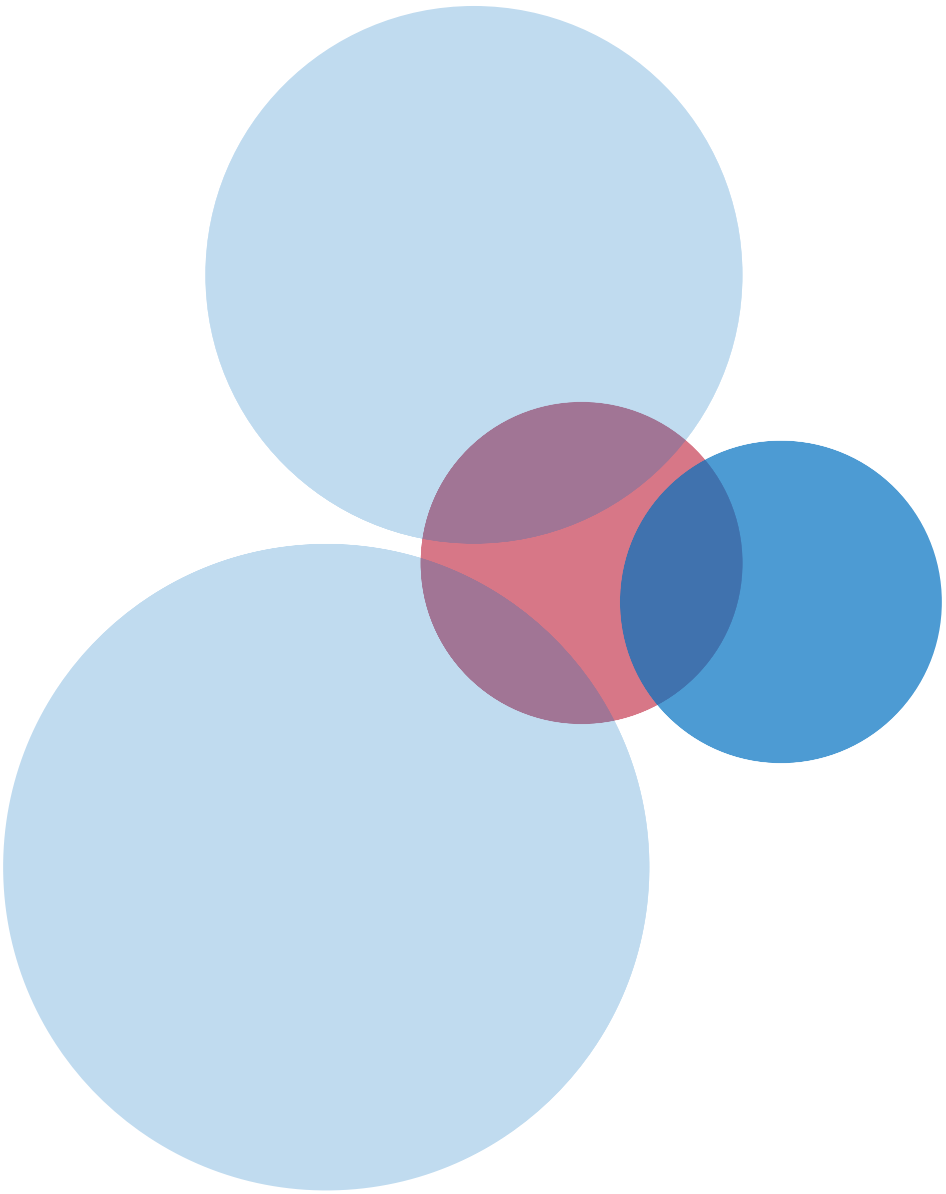
Theoretic Fundament



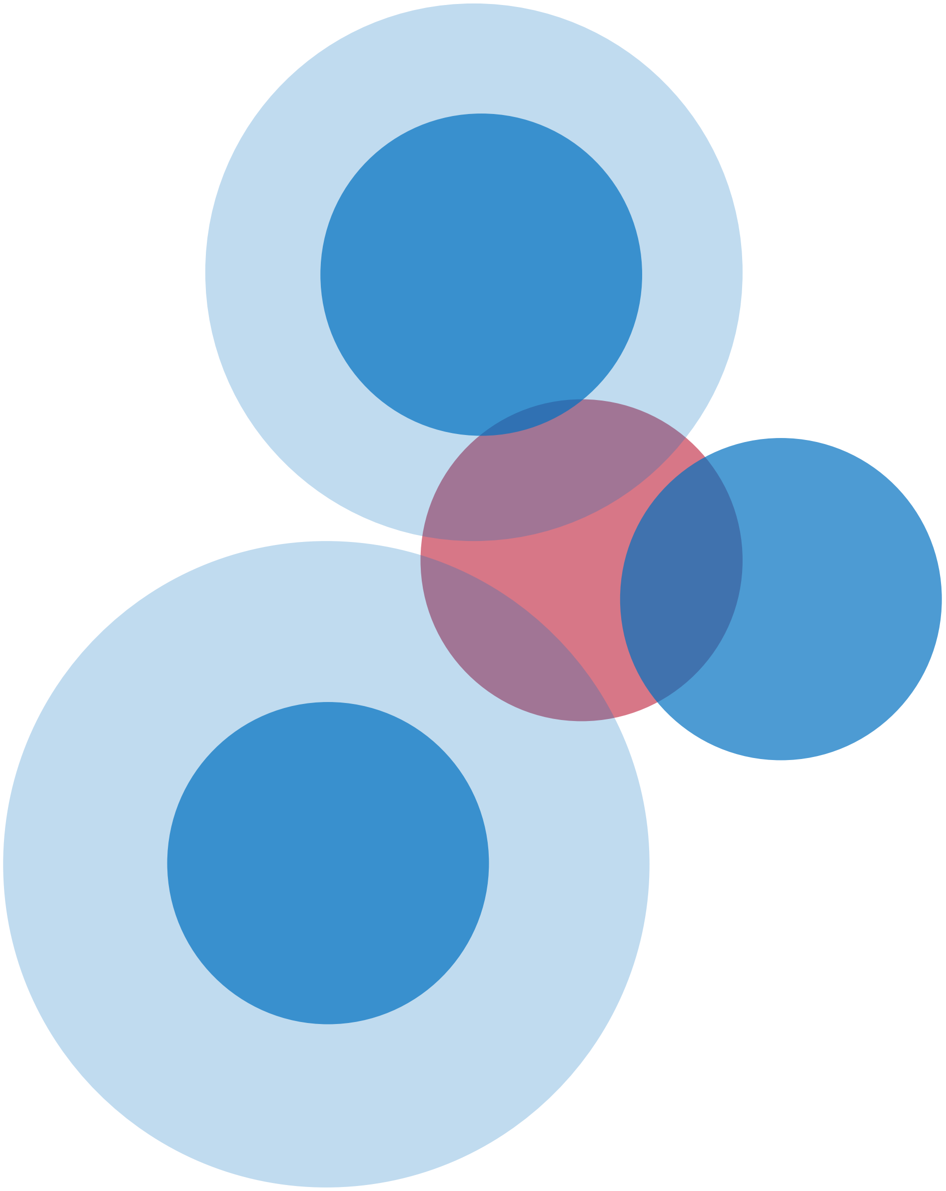
Theoretic Fundament



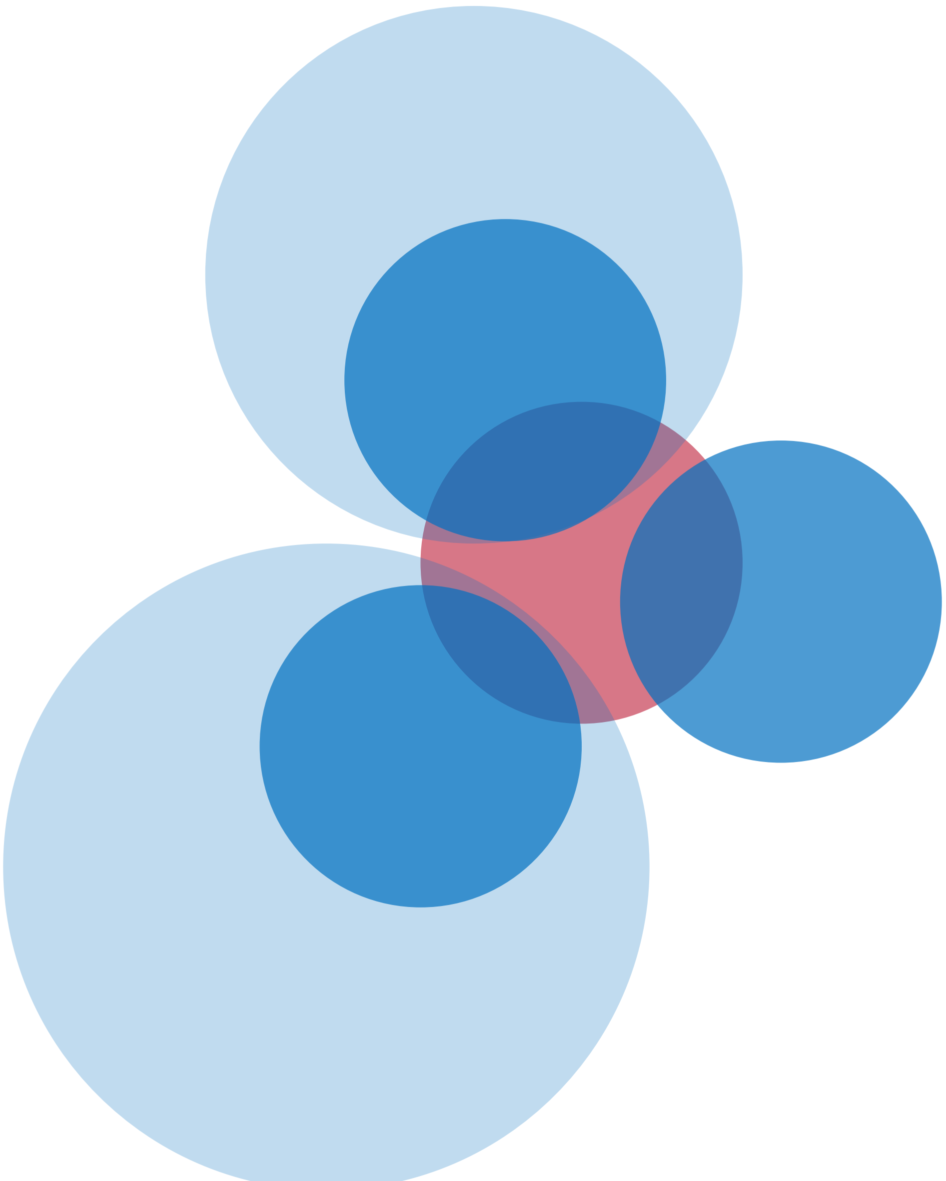
Theoretic Fundament



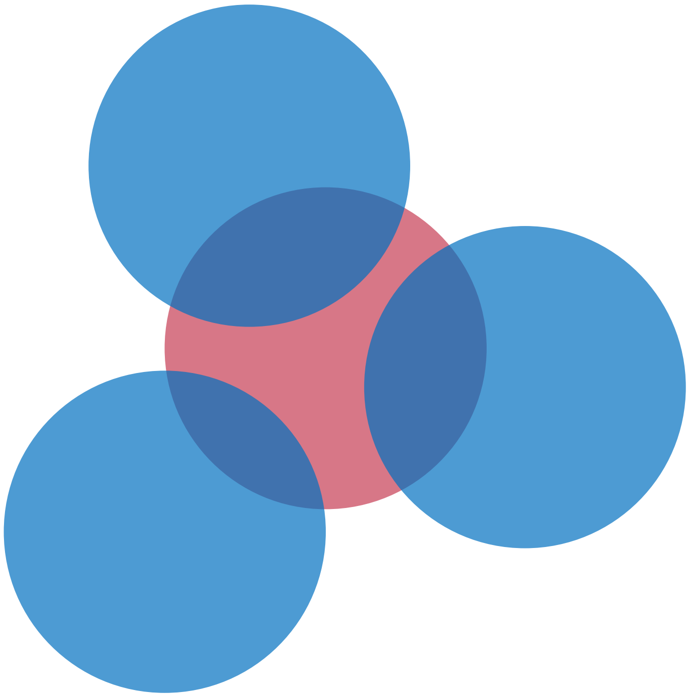
Theoretic Fundament



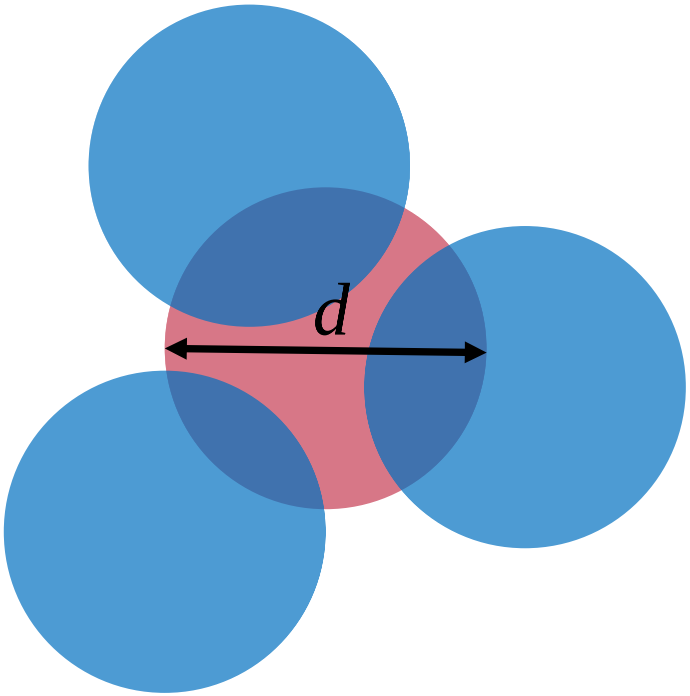
Theoretic Fundament



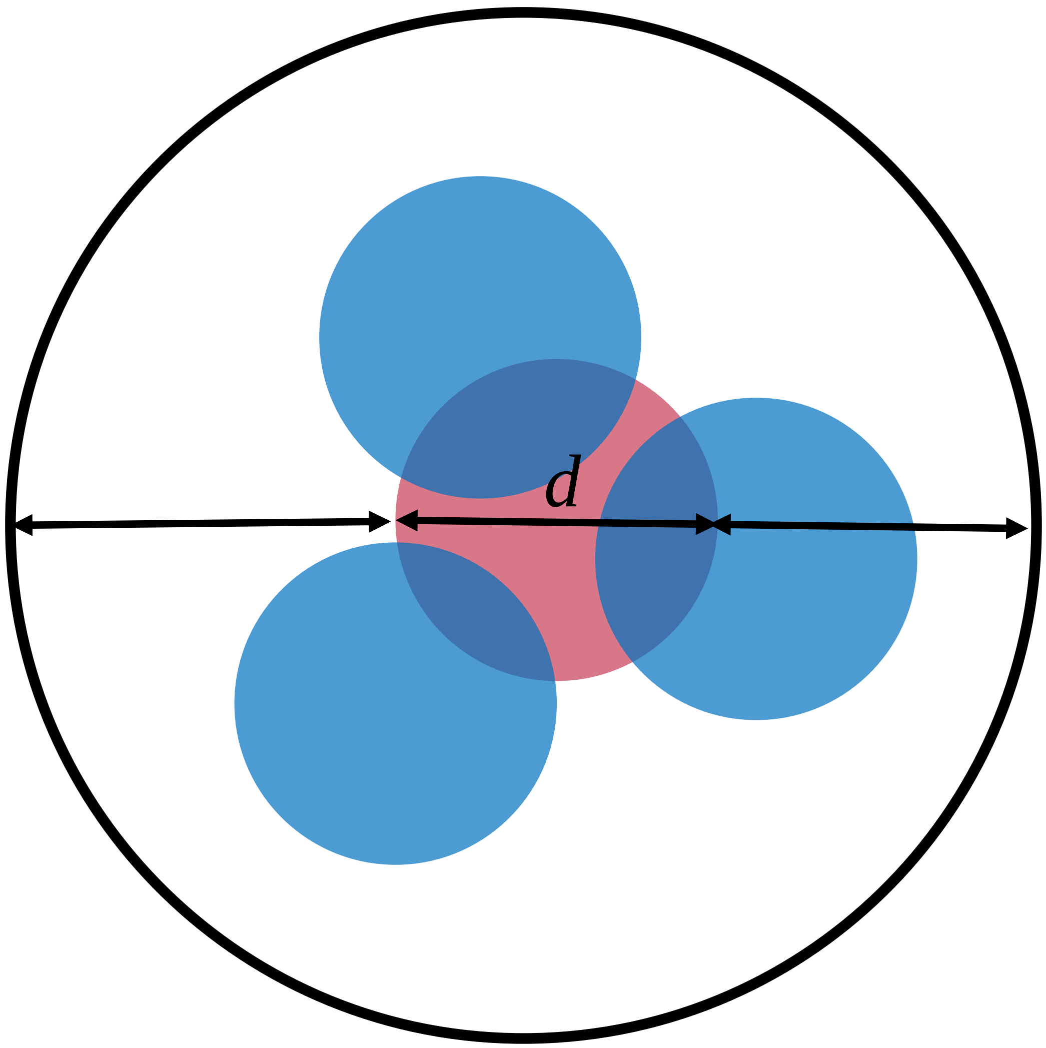
Theoretic Fundament



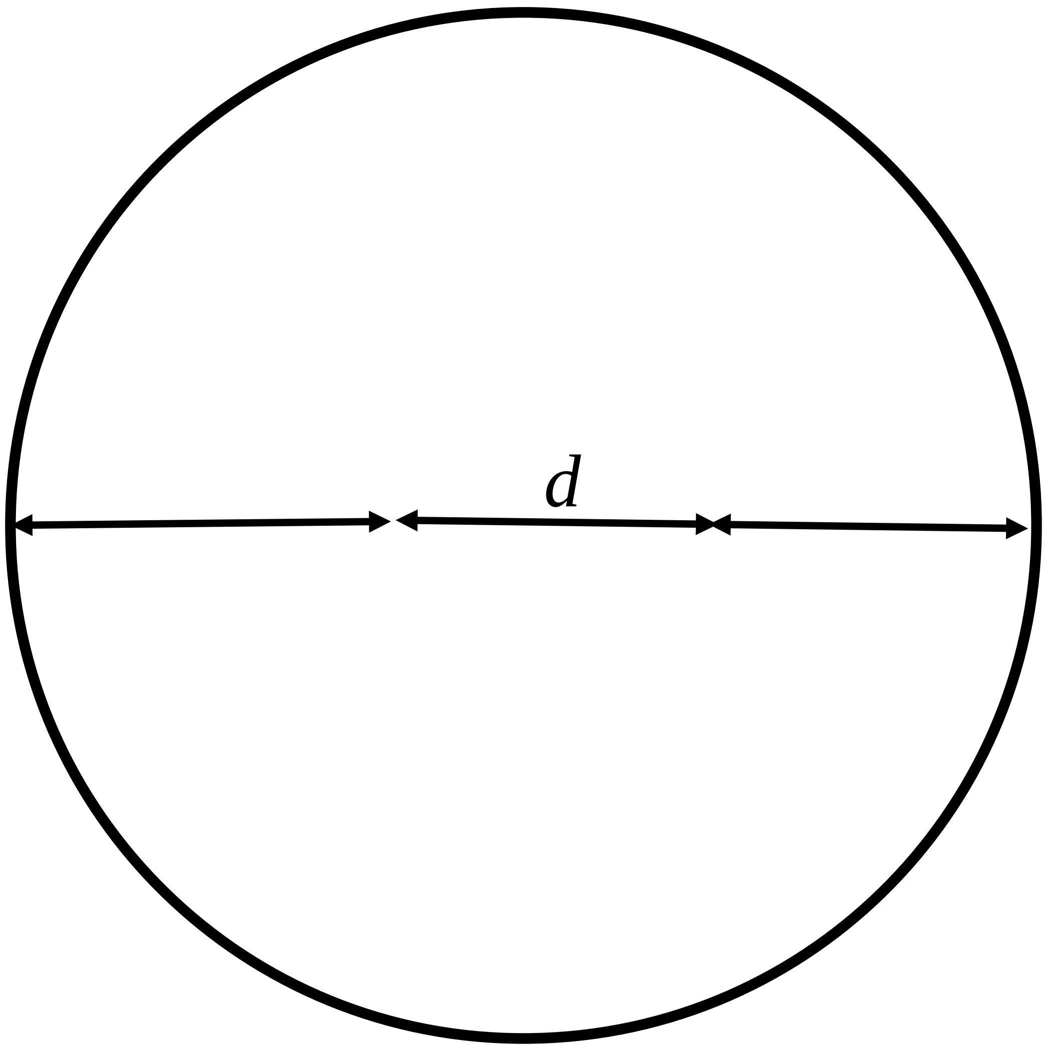
Theoretic Fundament



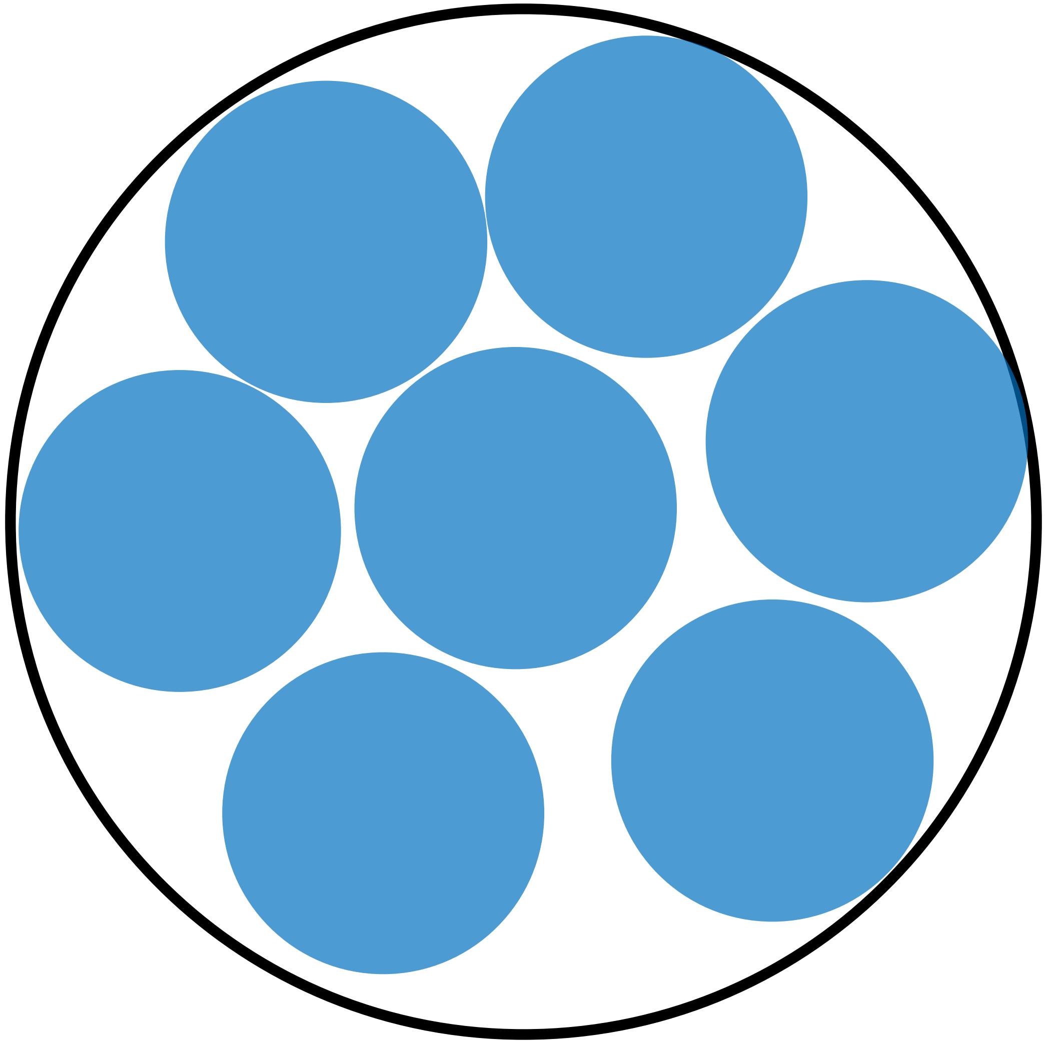
Theoretic Fundament



Theoretic Fundament



Theoretic Fundament



Theoretical Foundation

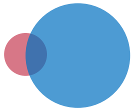
- Lemma: A single sphere s intersects a constant number of disjoint spheres A with at least the same radius.

Theoretical Foundation

- Lemma: A single sphere s intersects a constant number of disjoint spheres A with at least the same radius.
- Theorem: The maximum number of intersecting pairs of spheres of two polydisperse sphere packings A and B with n spheres is in $O(n)$.

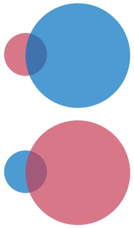
Theoretical Foundation

- Lemma: A single sphere s intersects a constant number of disjoint spheres A with at least the same radius.
- Theorem: The maximum number of intersecting pairs of spheres of two polydisperse sphere packings A and B with n spheres is in $O(n)$.
  - Check each sphere $s_i \in A$ against larger spheres in B



Theoretical Foundation

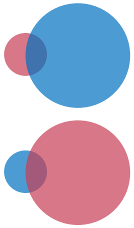
- Lemma: A single sphere s intersects a constant number of disjoint spheres A with at least the same radius.
- Theorem: The maximum number of intersecting pairs of spheres of two polydisperse sphere packings A and B with n spheres is in $O(n)$.
  - Check each sphere $s_i \in A$ against larger spheres in B
  - Check each sphere $s_j \in B$ against larger spheres in A



Theoretical Foundation

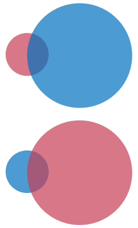
- Lemma: A single sphere s intersects a constant number of disjoint spheres A with at least the same radius.
- Theorem: The maximum number of intersecting pairs of spheres of two polydisperse sphere packings A and B with n spheres is in $O(n)$.
  - Check each sphere $s_i \in A$ against larger spheres in B
  - Check each sphere $s_j \in B$ against larger spheres in A

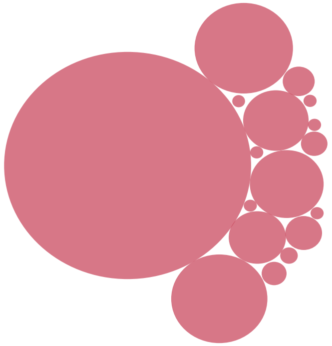
  $\Rightarrow O(1)$ for each $s_i \in A$ and $s_j \in B$



Theoretical Foundation

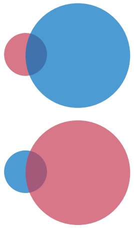
- Lemma: A single sphere s intersects a constant number of disjoint spheres A with at least the same radius.
- Theorem: The maximum number of intersecting pairs of spheres of two polydisperse sphere packings A and B with n spheres is in $O(n)$.
  - Check each sphere $s_i \in A$ against larger spheres in B
  - Check each sphere $s_j \in B$ against larger spheres in A
  - $\Rightarrow O(1)$ for each $s_i \in A$ and $s_j \in B$
  - $\Rightarrow O(n)$ in total





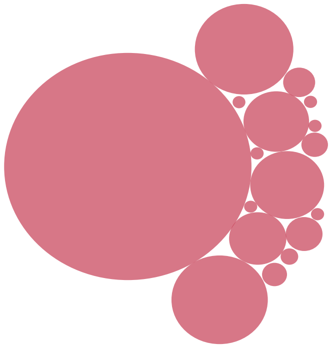
Theoretical Foundation

- Lemma: A single sphere s intersects a constant number of disjoint spheres A with at least the same radius.
- Theorem: The maximum number of intersecting pairs of spheres of two polydisperse sphere packings A and B with n spheres is in $O(n)$.
  - Check each sphere $s_i \in A$ against larger spheres in B
  - Check each sphere $s_j \in B$ against larger spheres in A

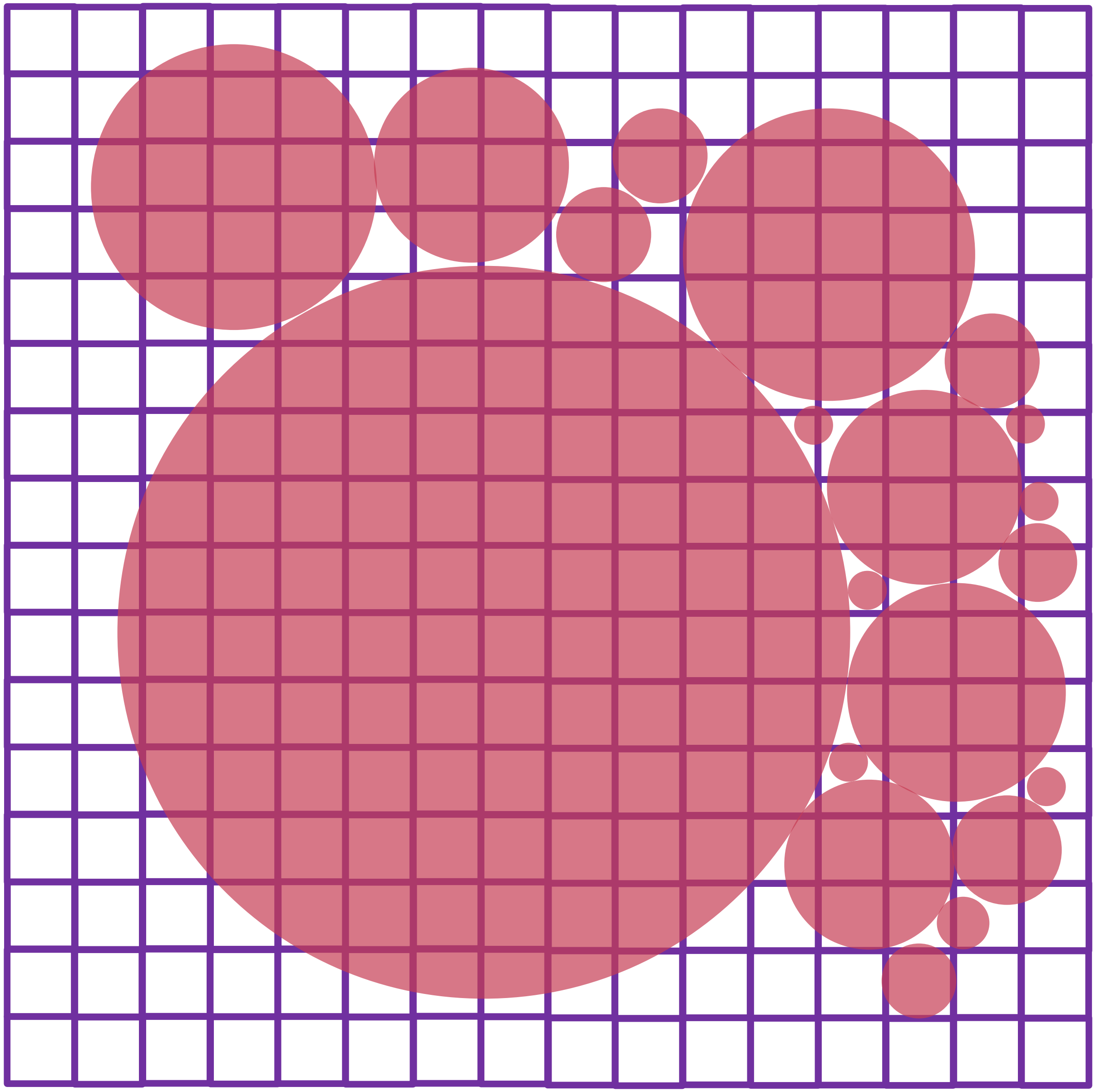


$\Rightarrow O(1)$ for each $s_i \in A$ and $s_j \in B$

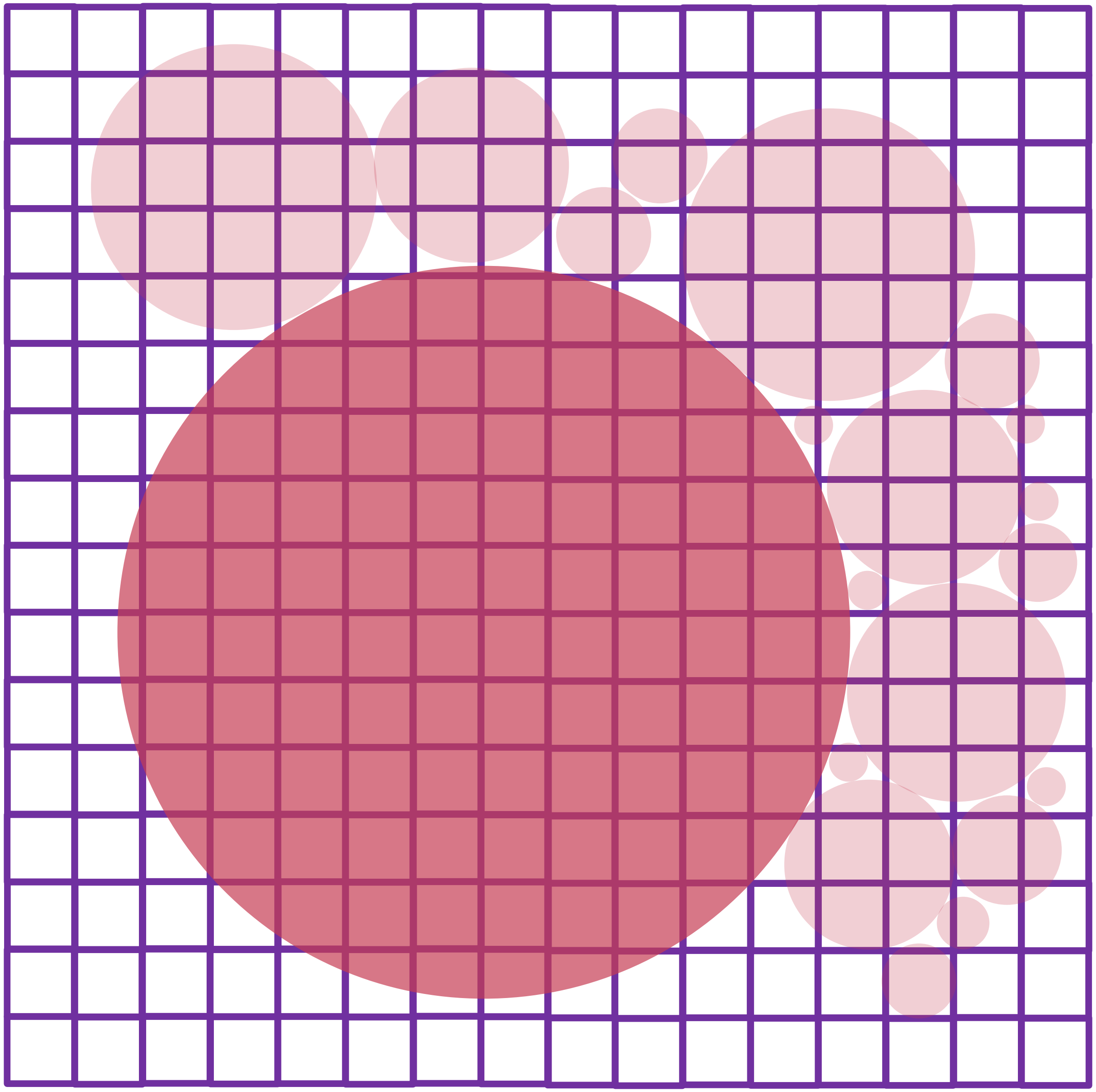
$\Rightarrow O(n)$ in total



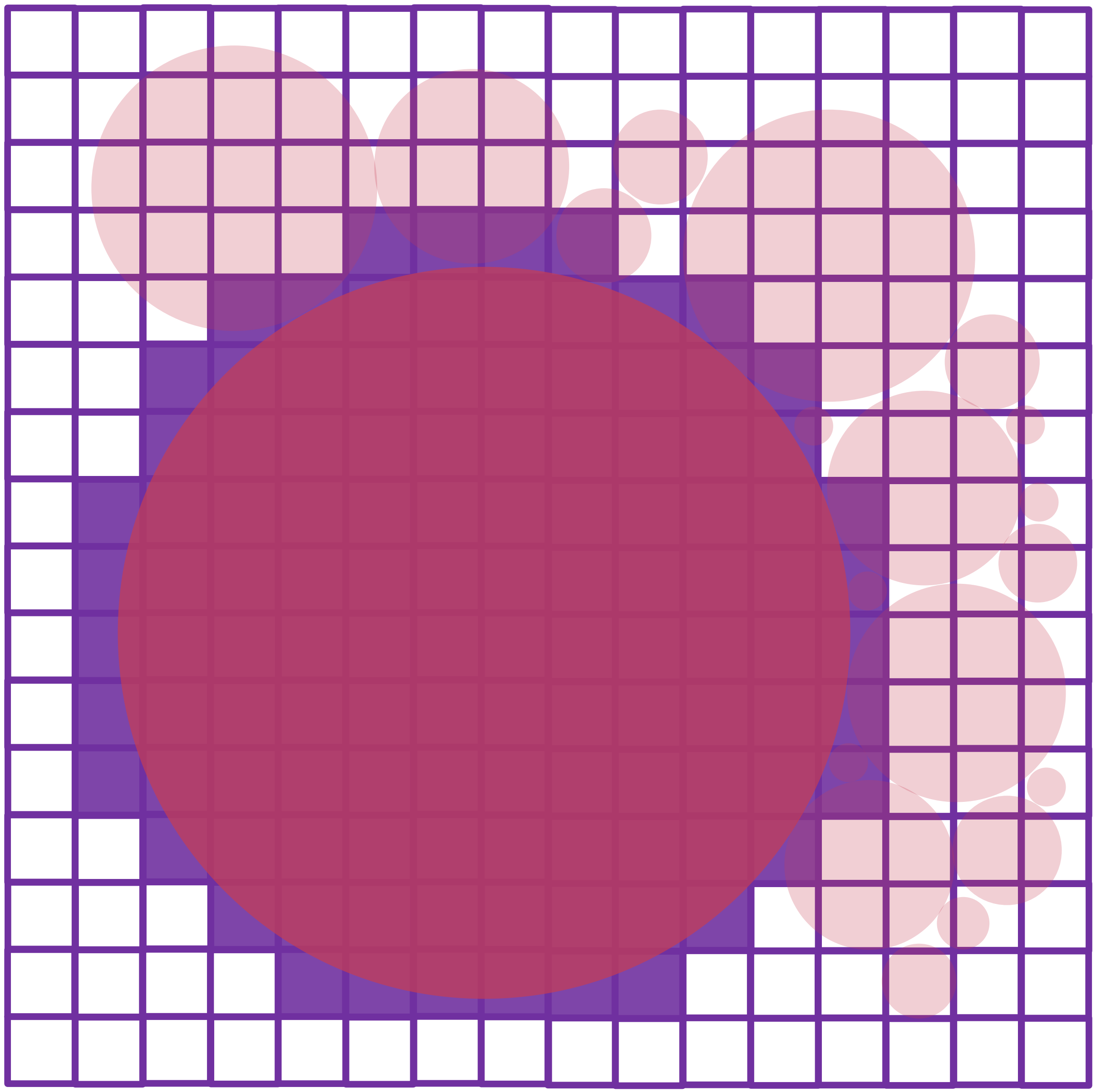
Our Algorithm



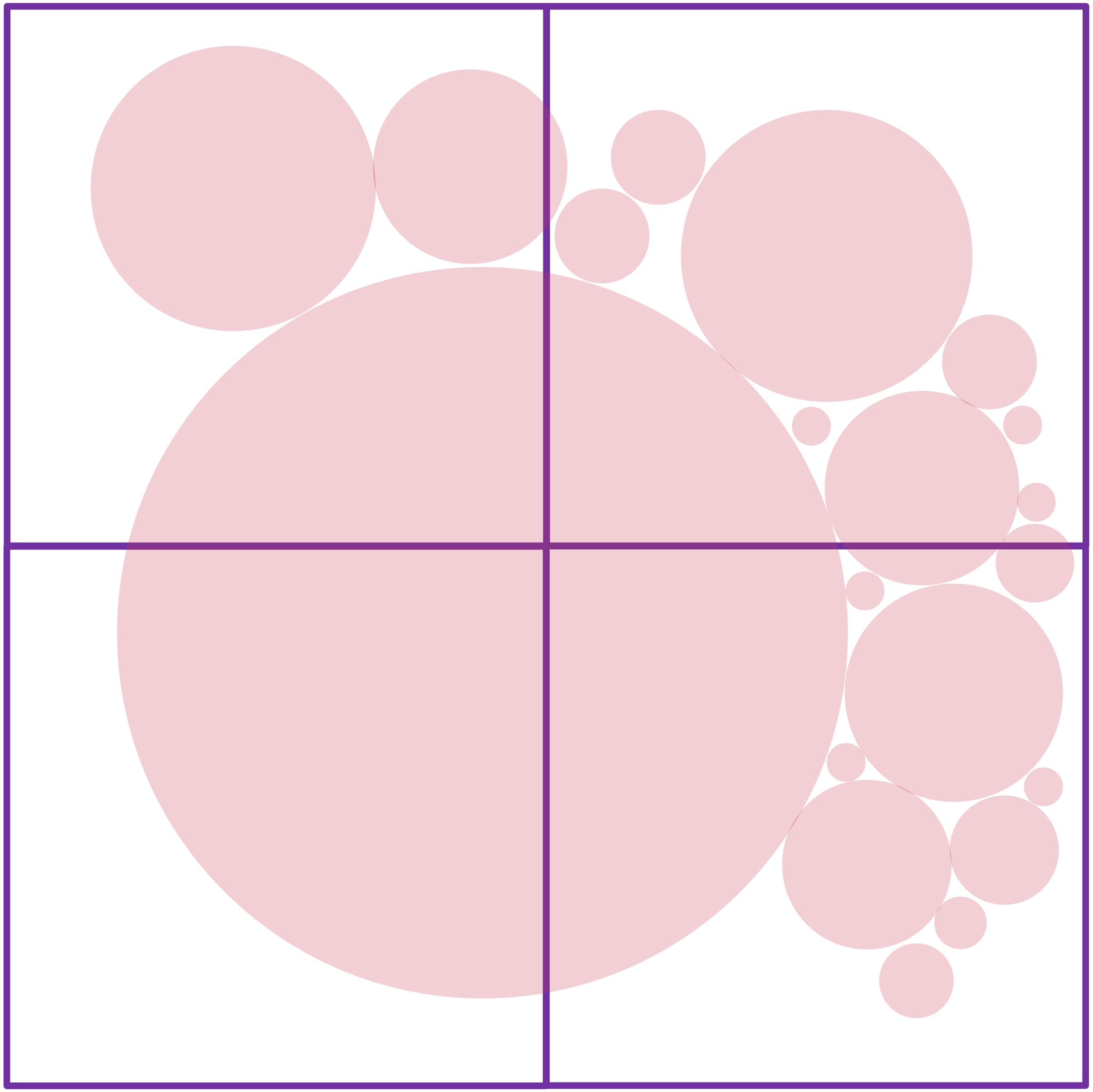
Our Algorithm



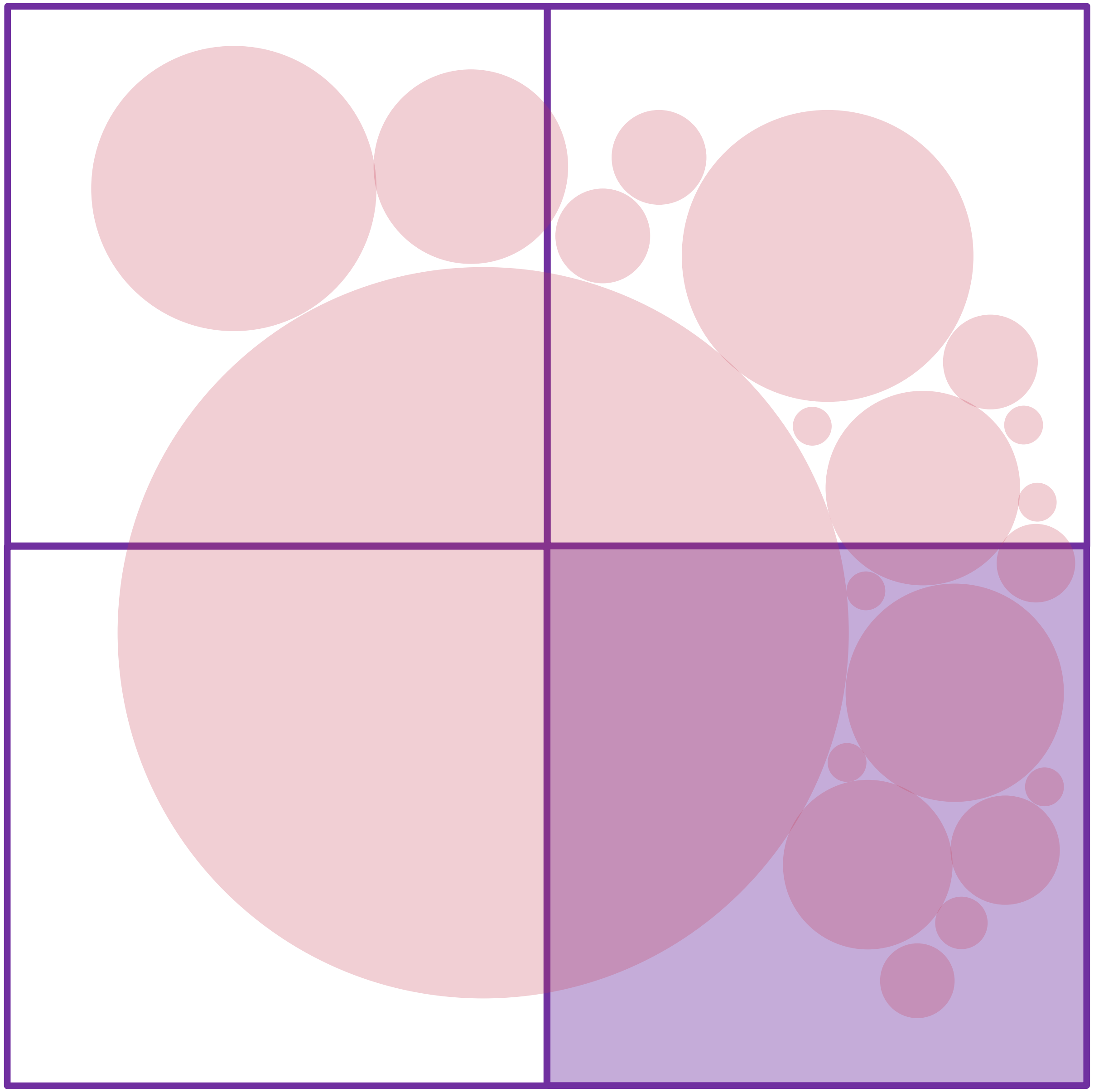
Our Algorithm



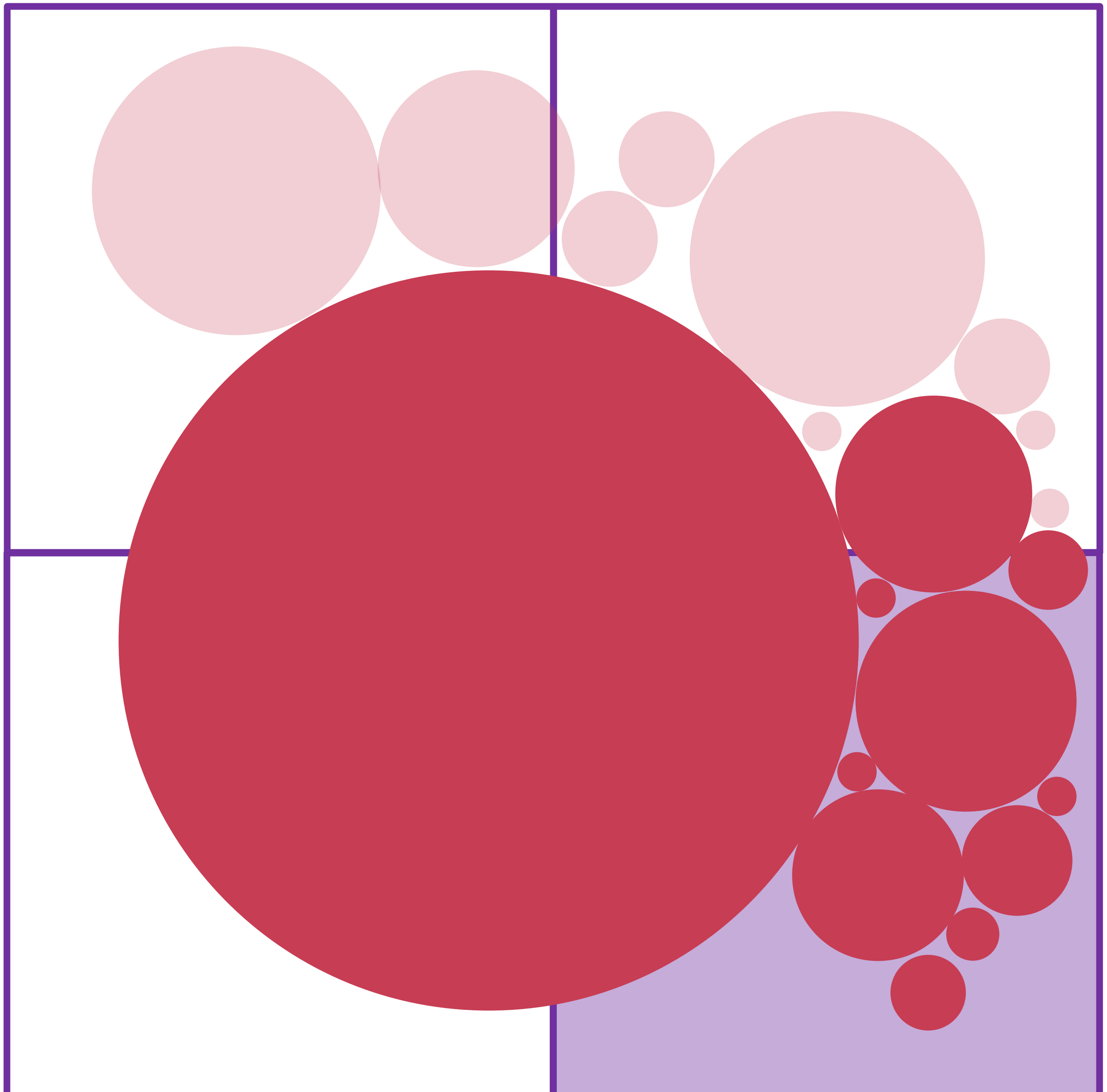
Our Algorithm



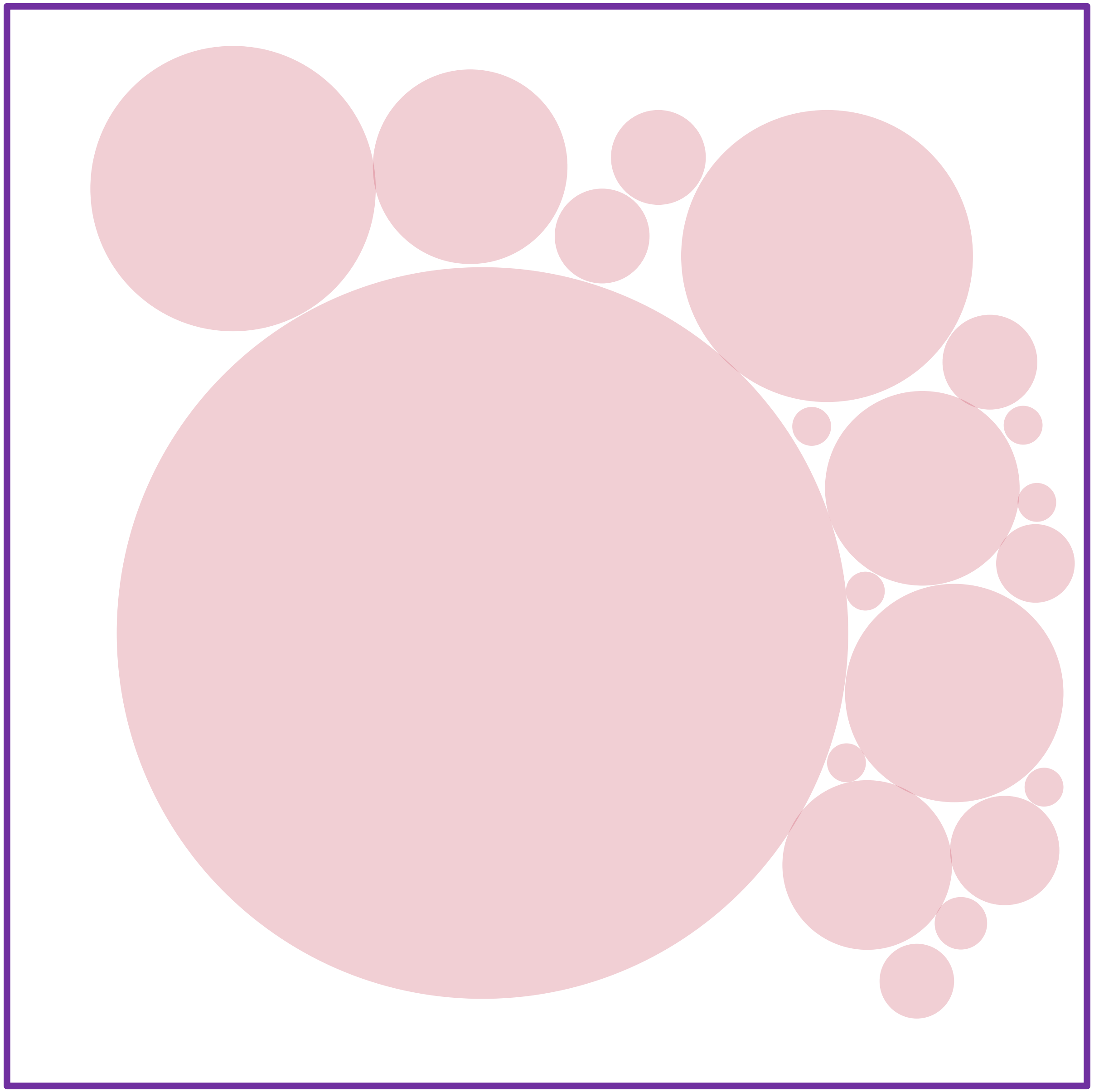
Our Algorithm



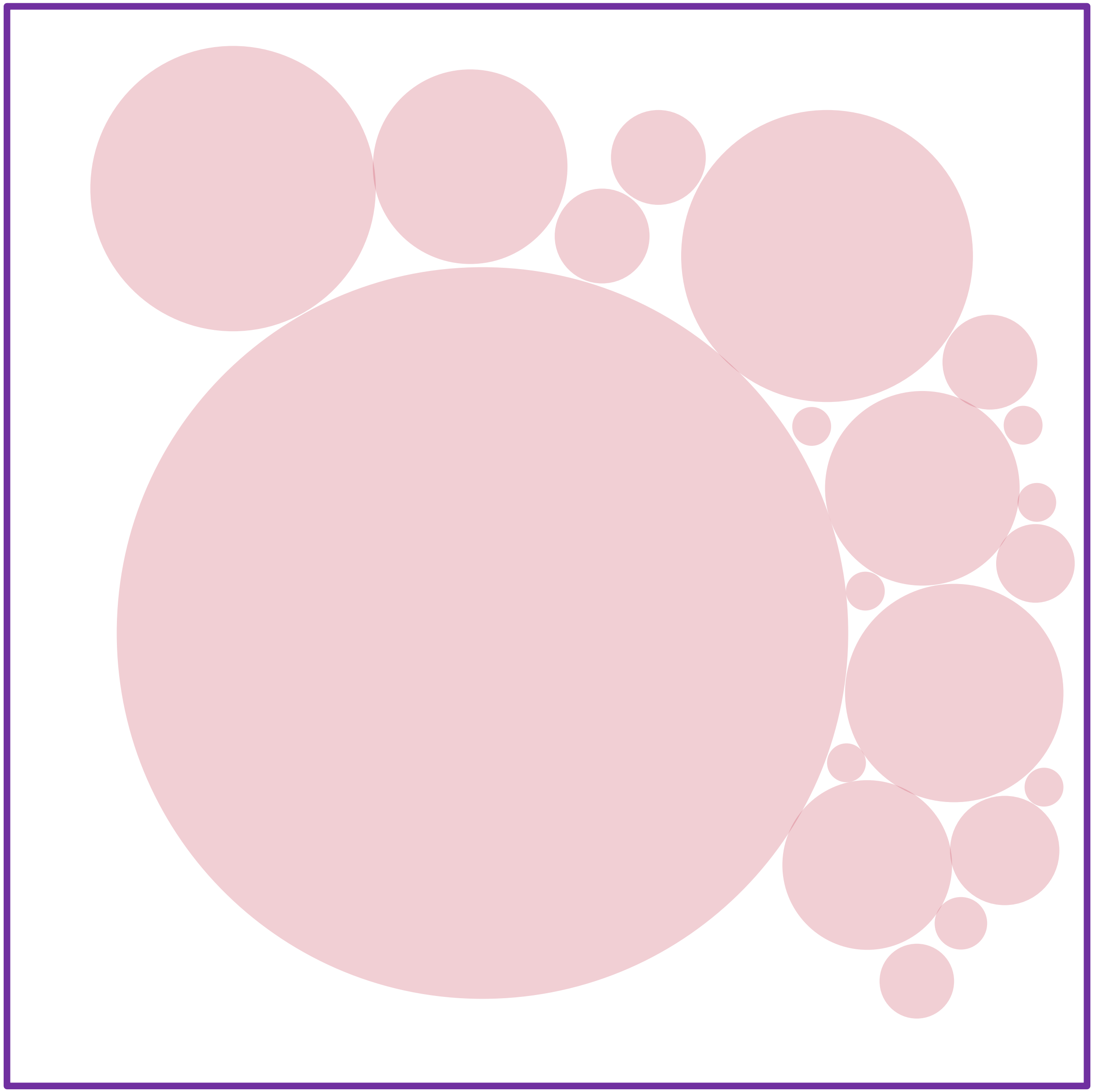
Our Algorithm

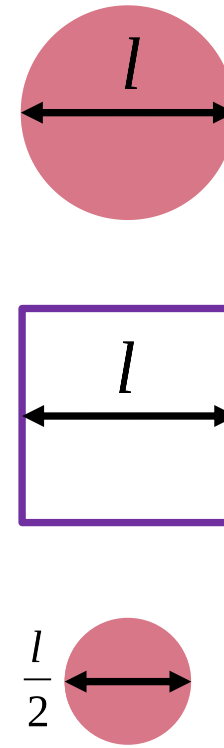


Our Algorithm



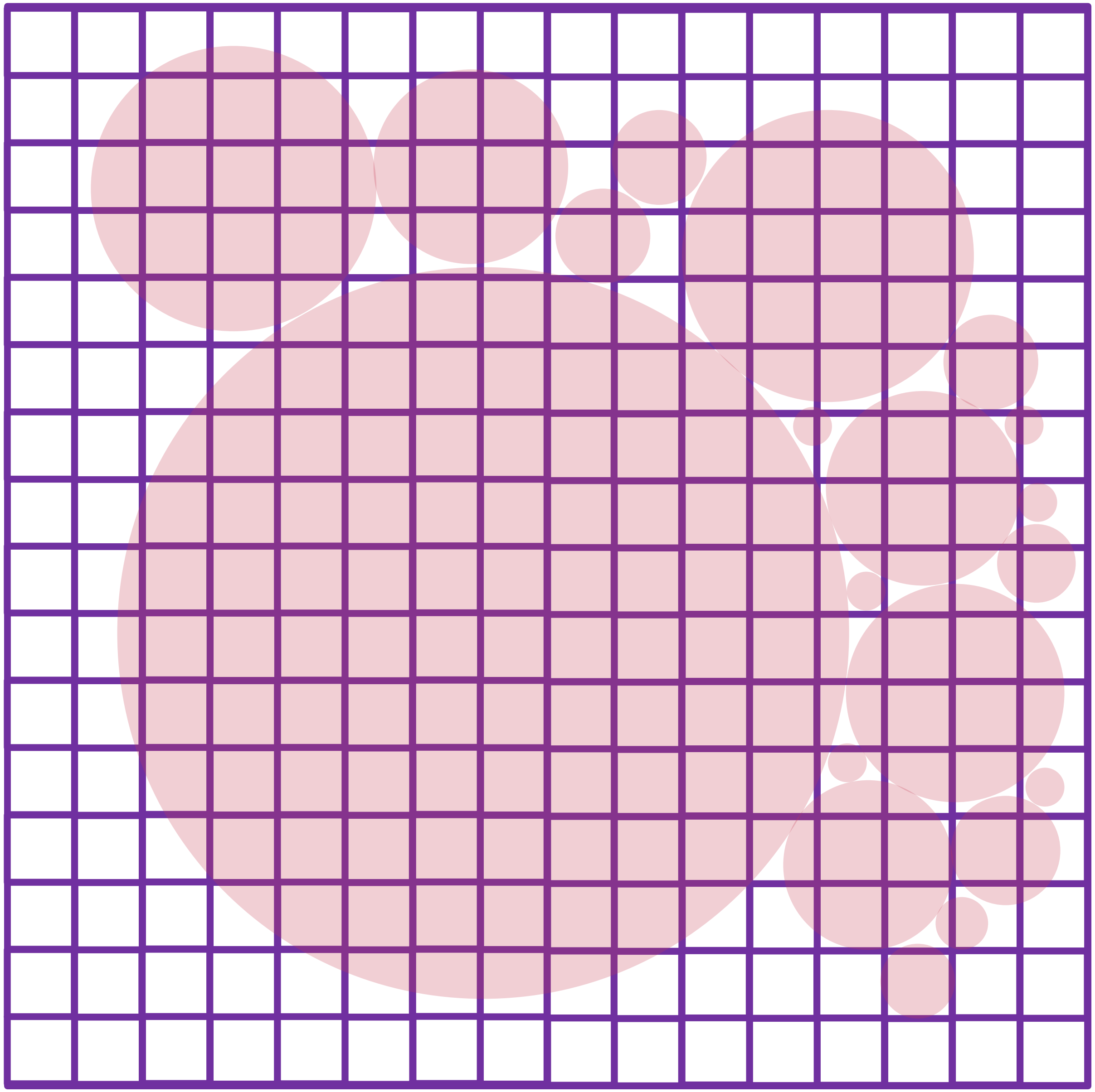
Our Algorithm

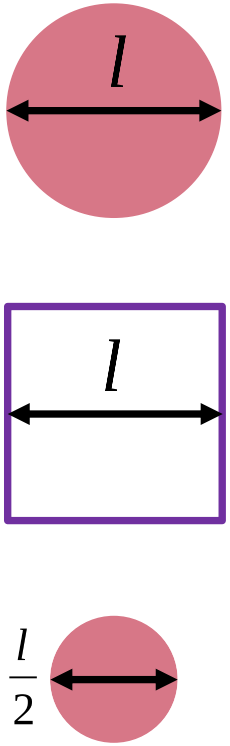




Our Algorithm

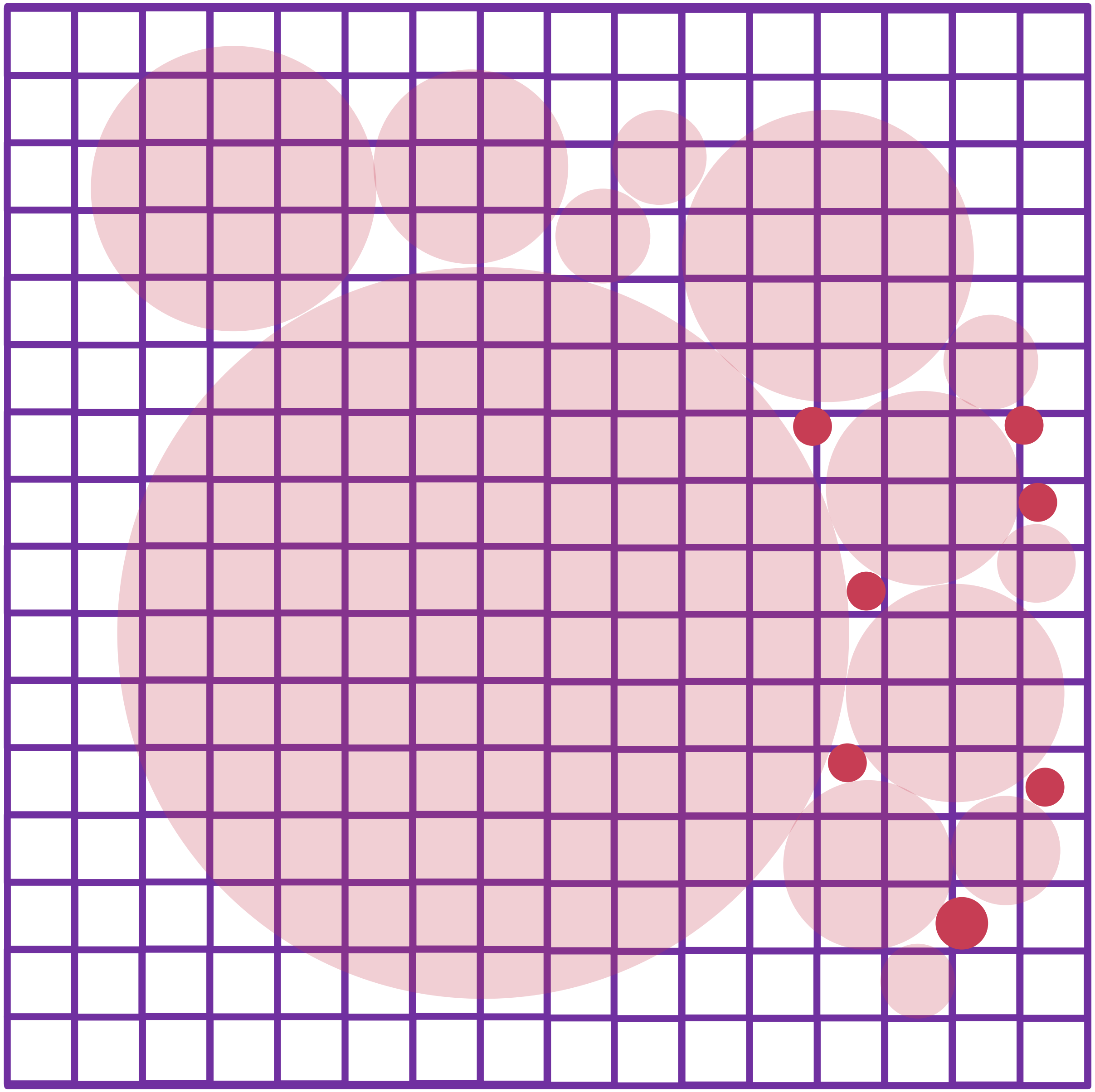
Level 4

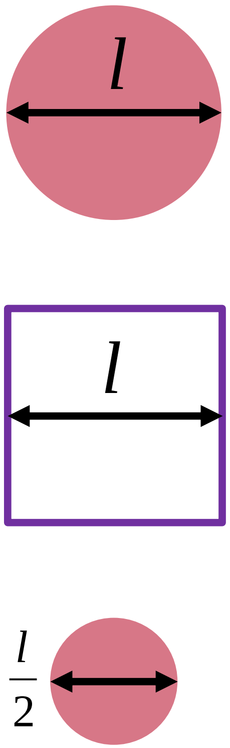




Our Algorithm

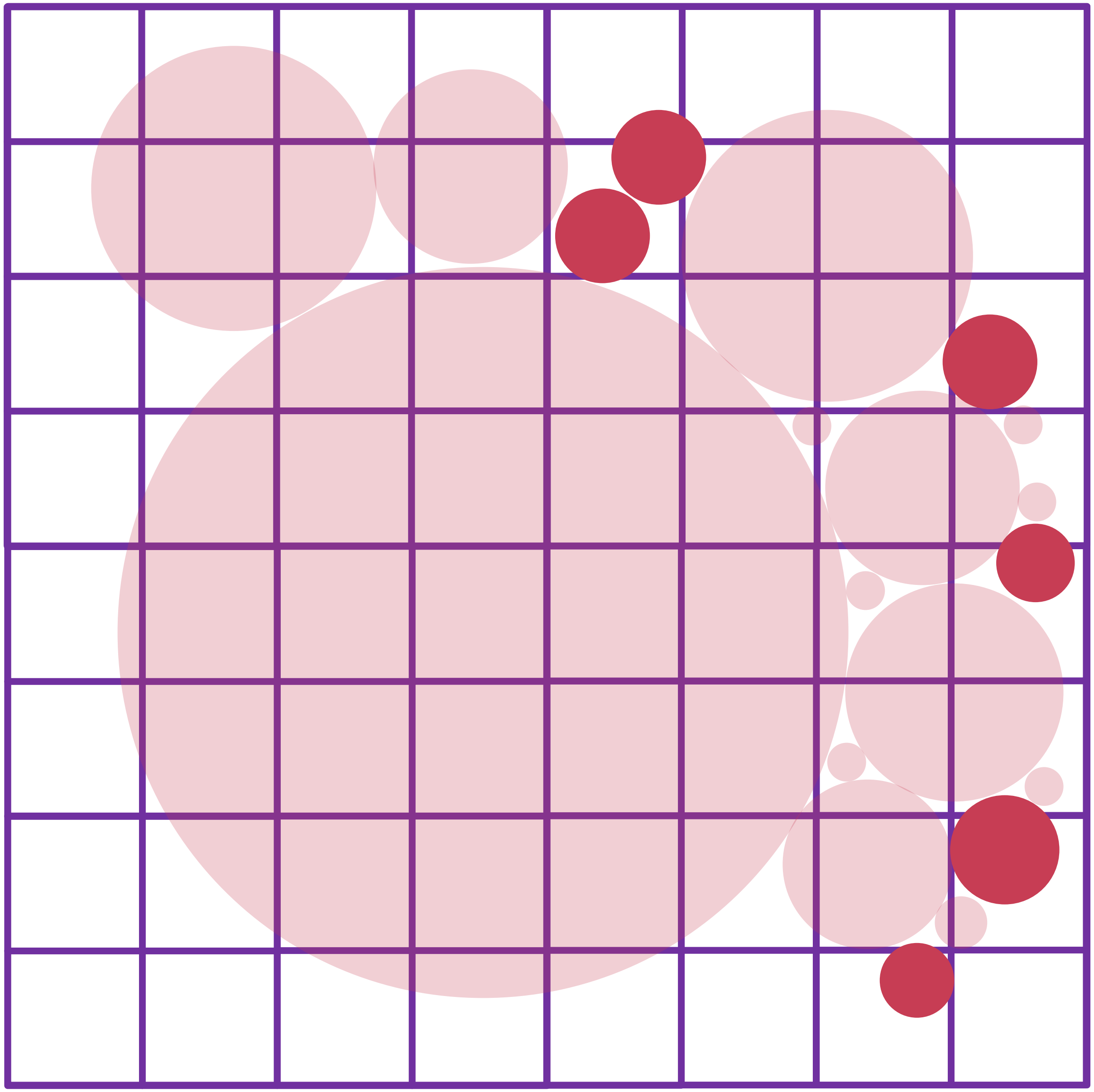
L e v e l 4

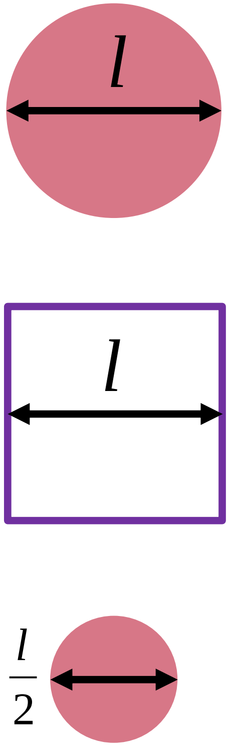




Our Algorithm

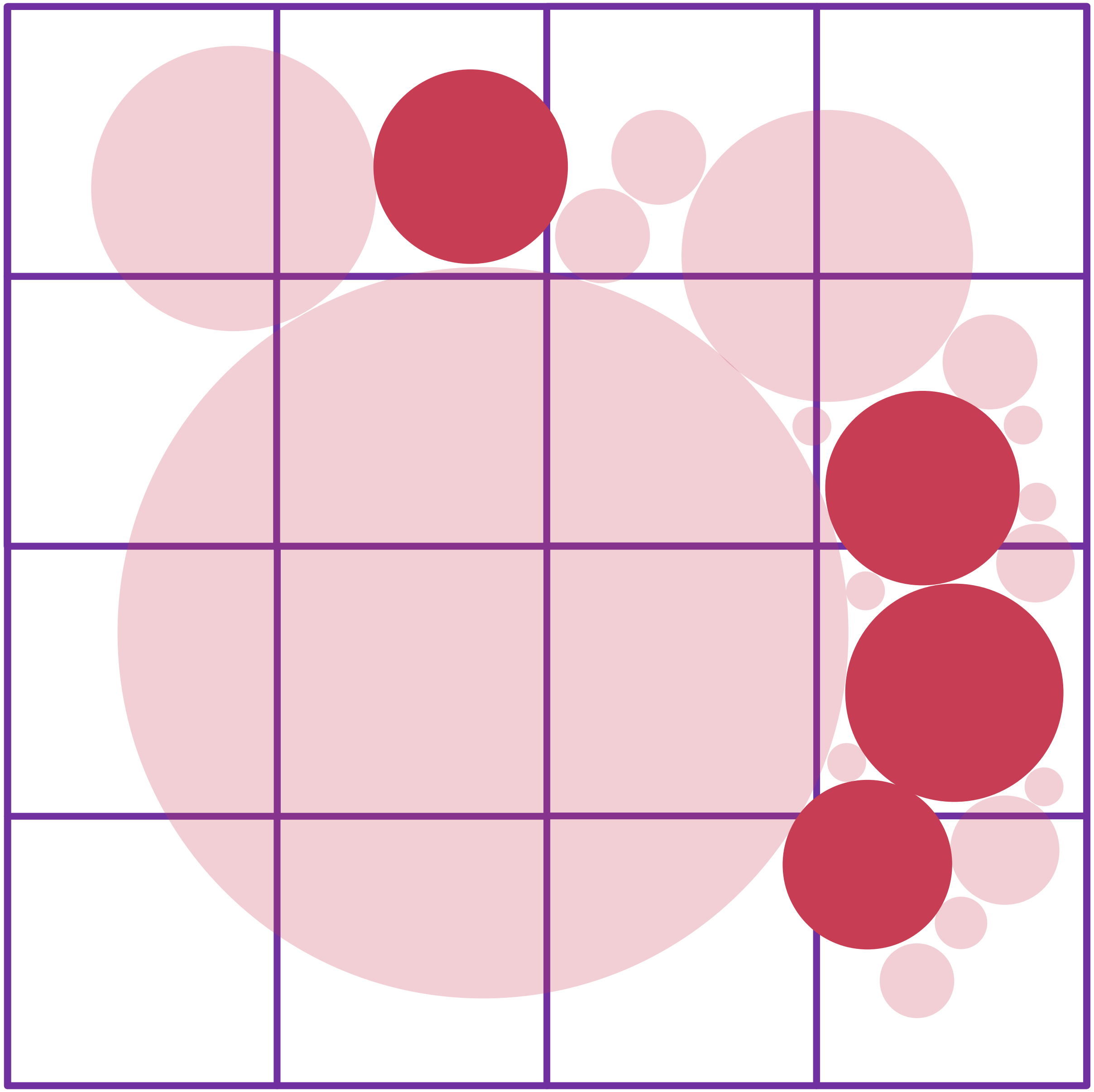
L e v e l 3

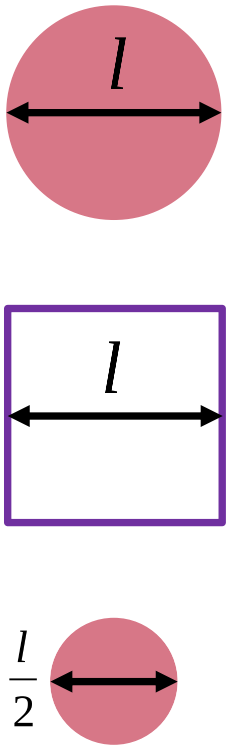




Our Algorithm

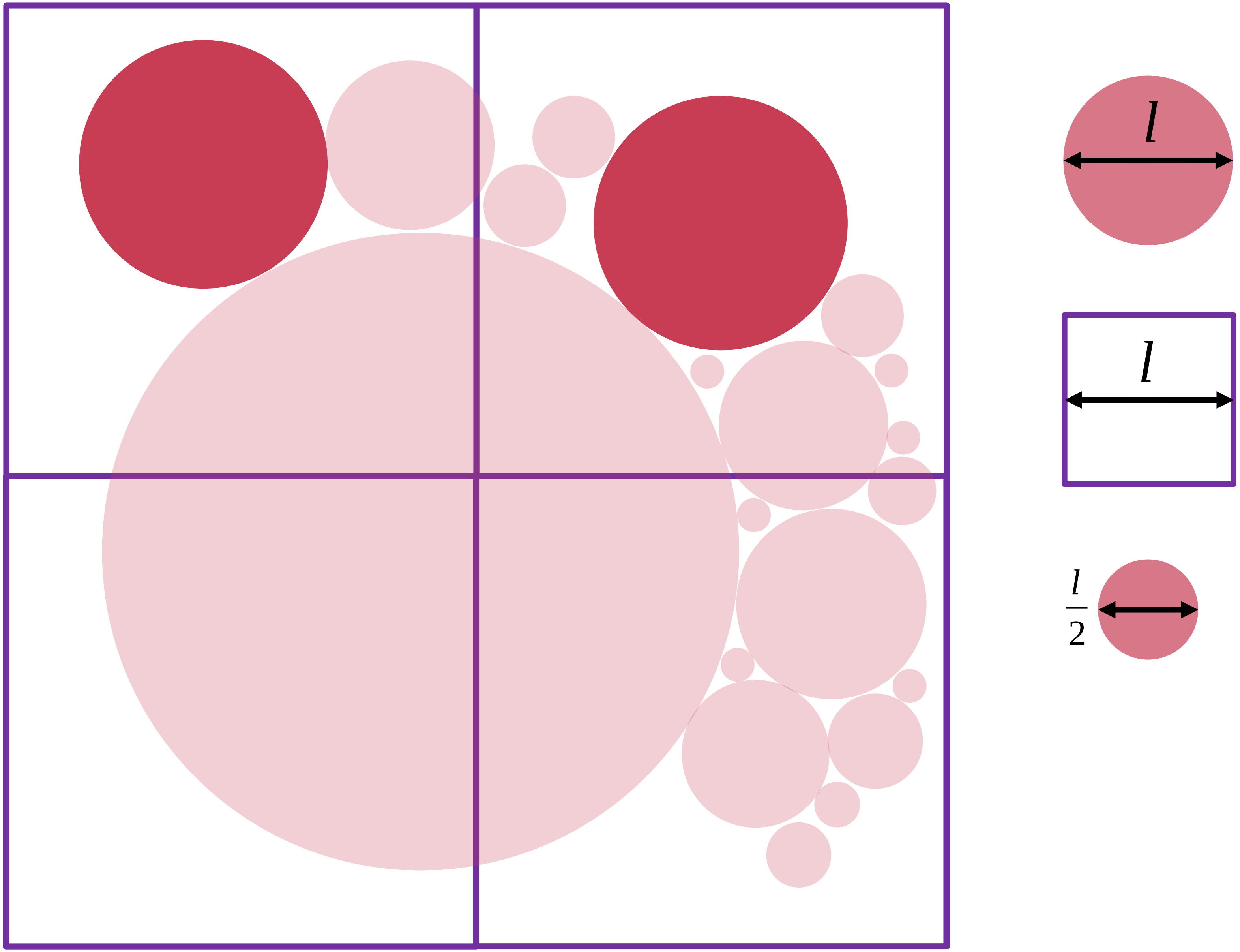
Level 2





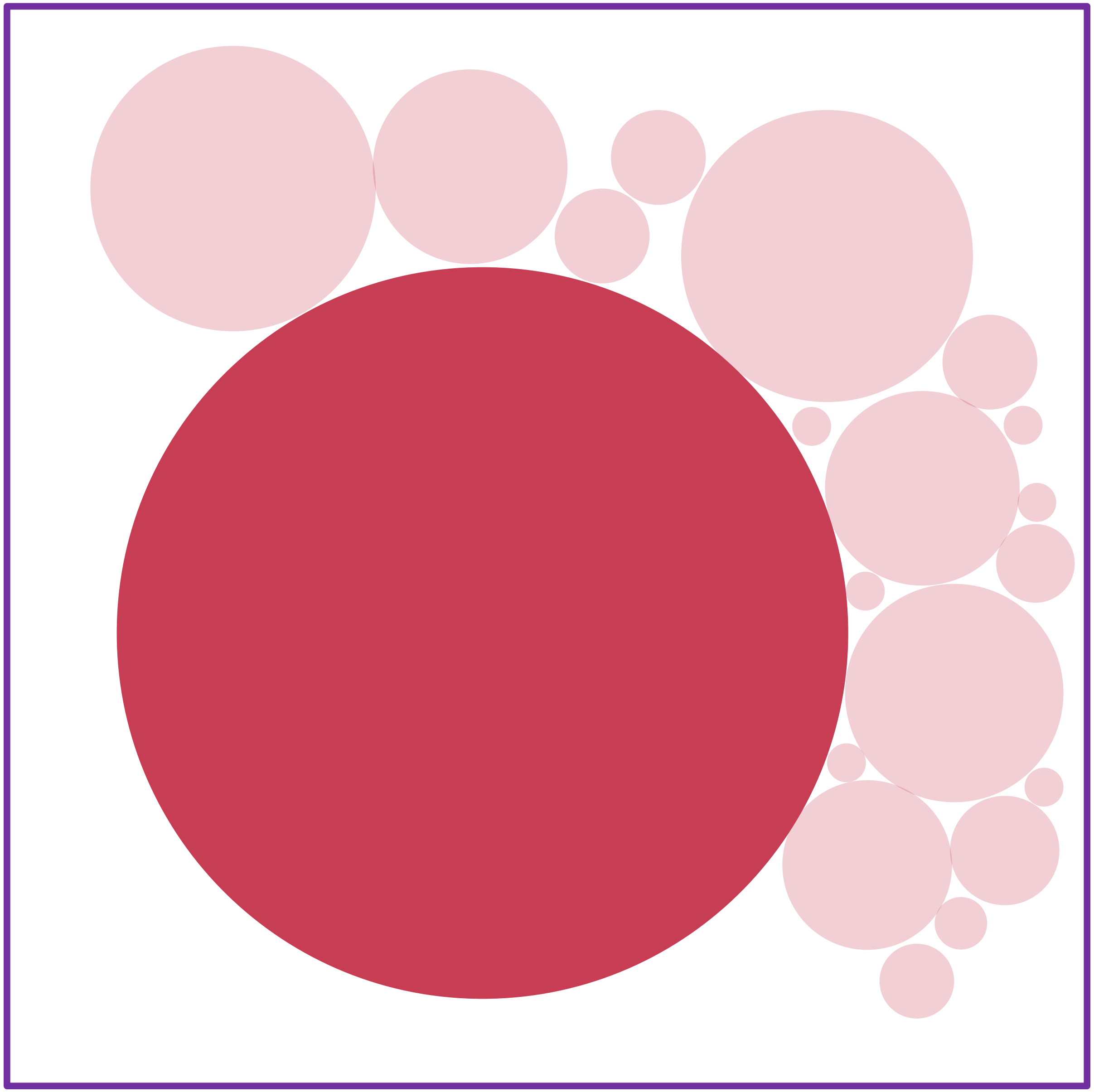
Our Algorithm

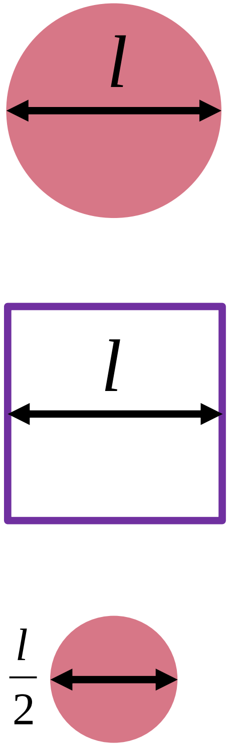
Level 1



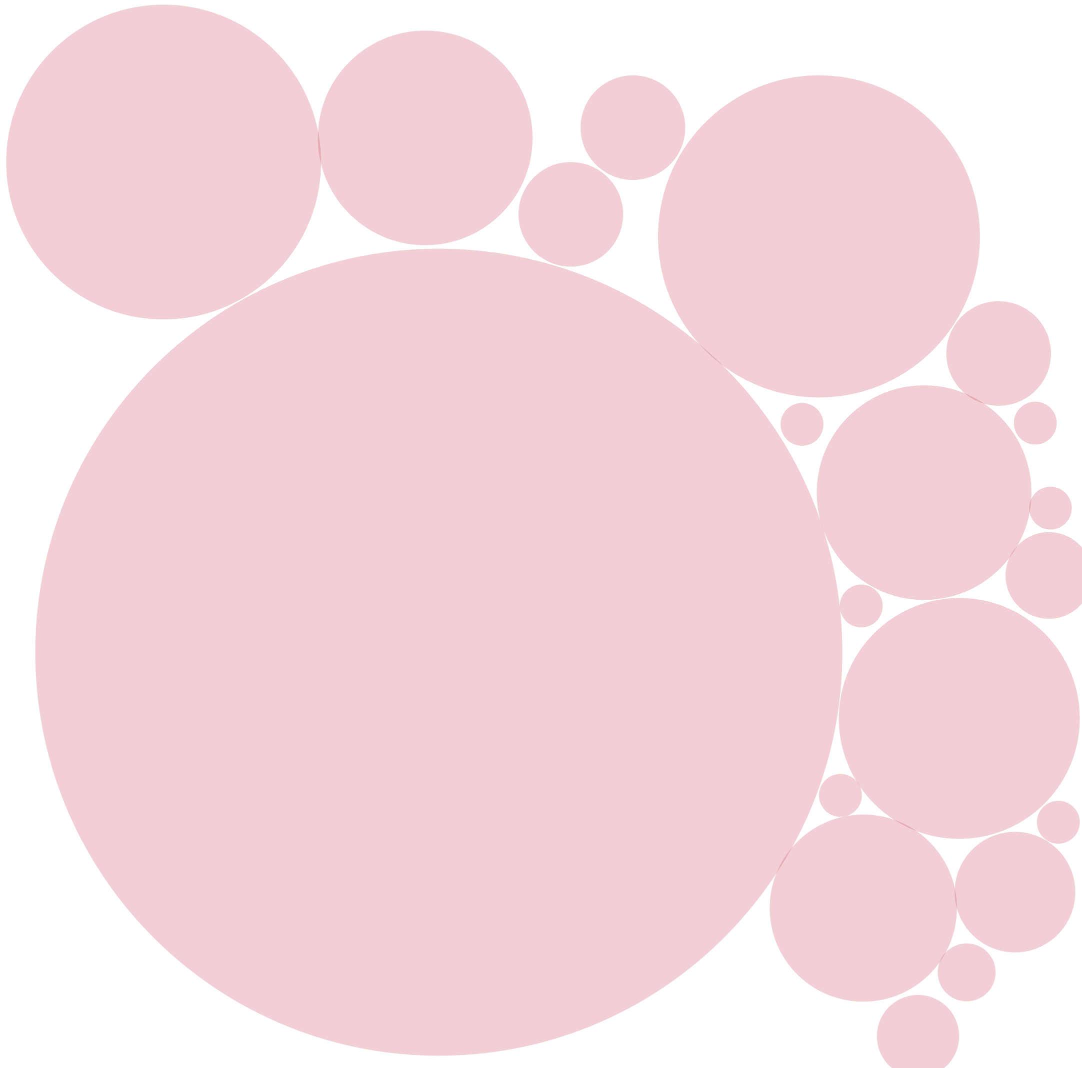
Our Algorithm

Level 0

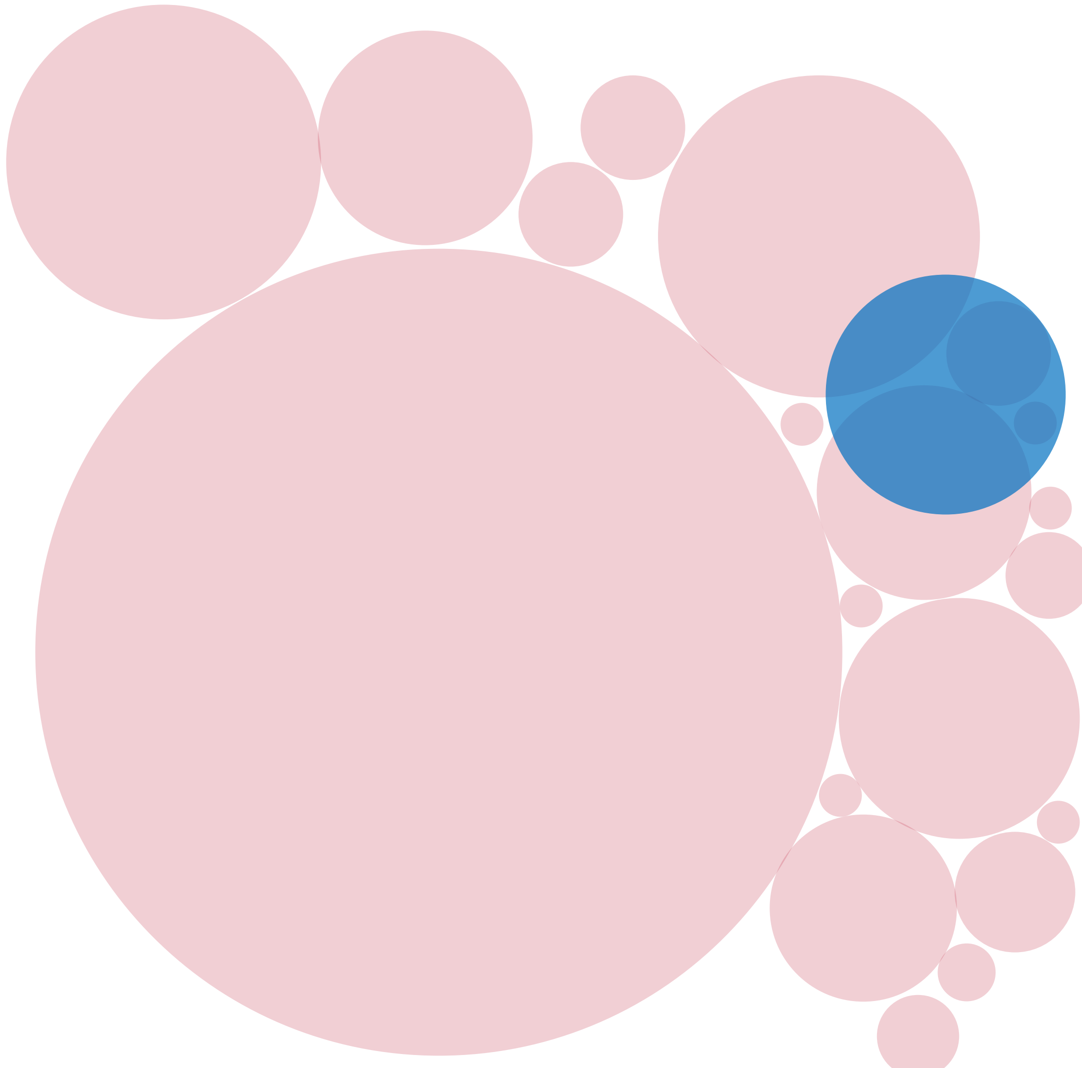




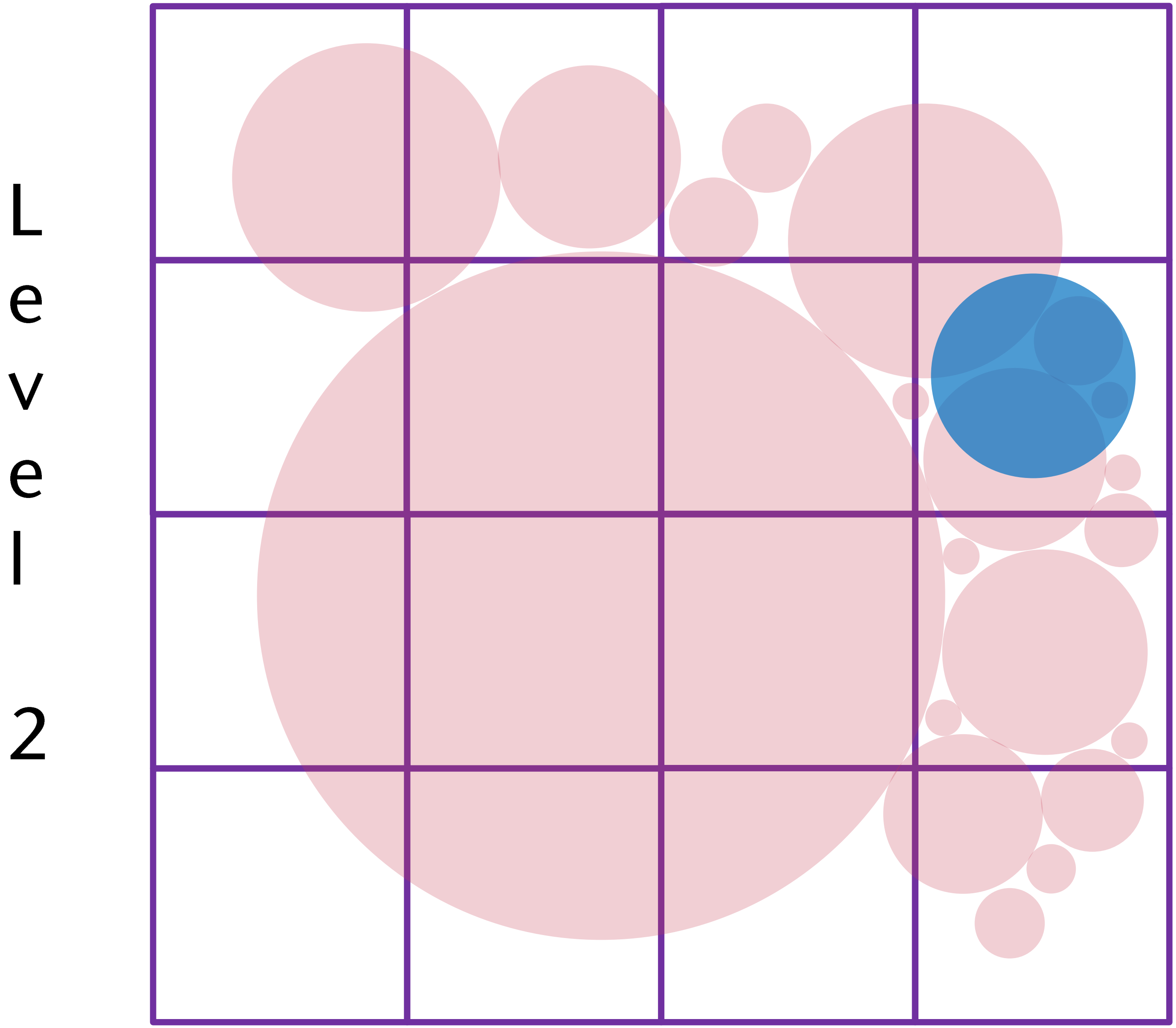
Our Algorithm



Our Algorithm



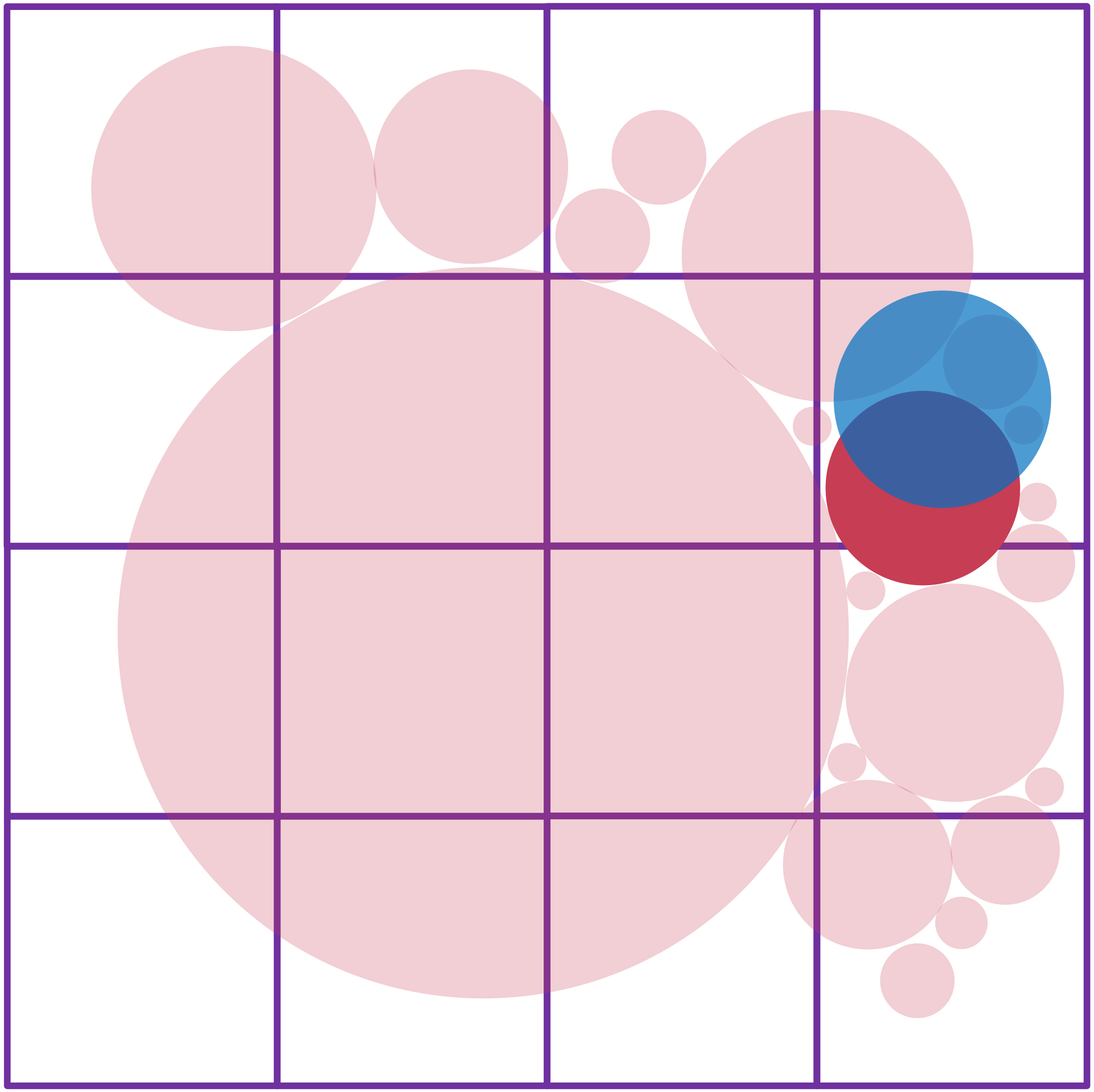
Our Algorithm



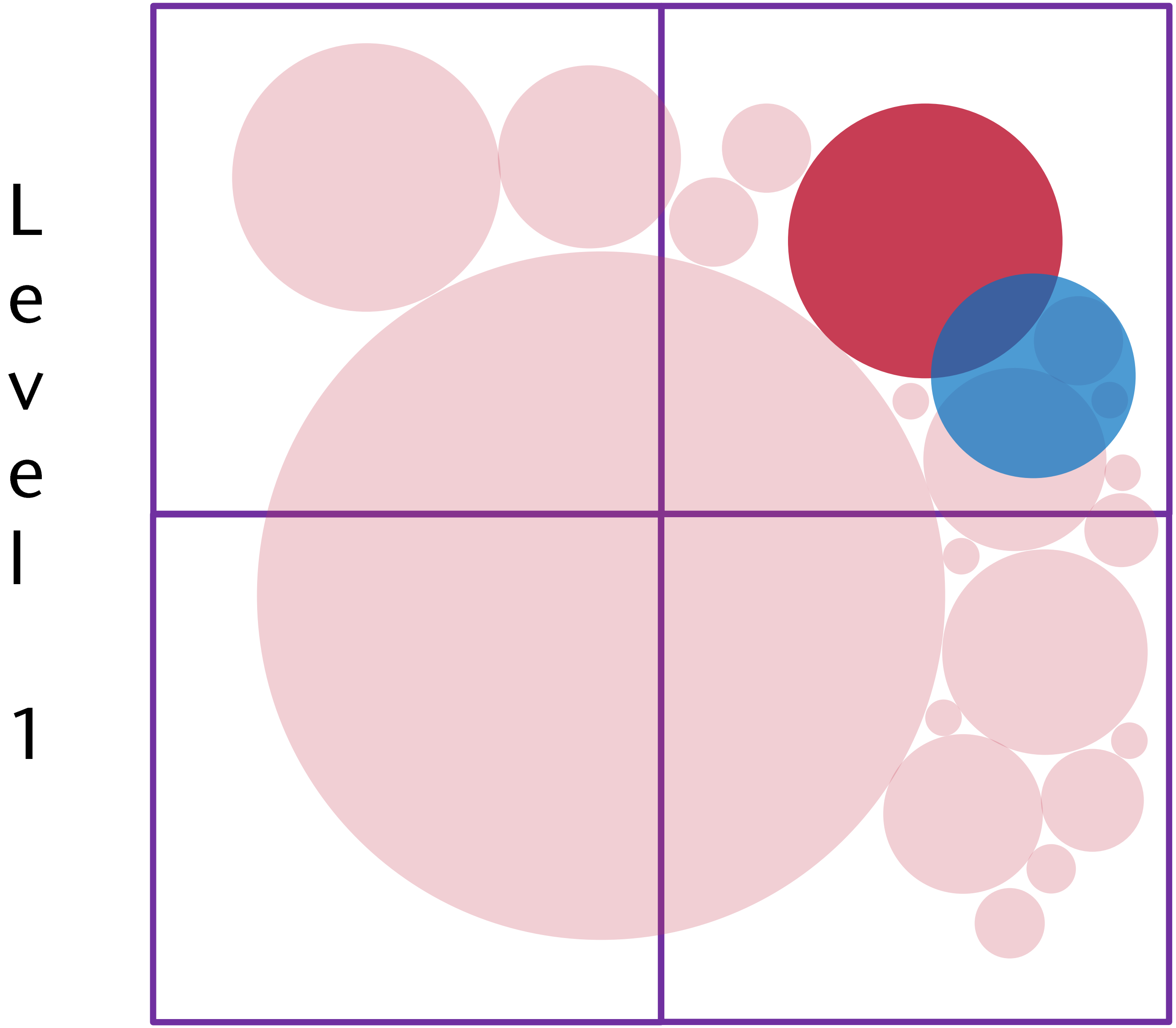
Our Algorithm

L
e
v
e
l

2

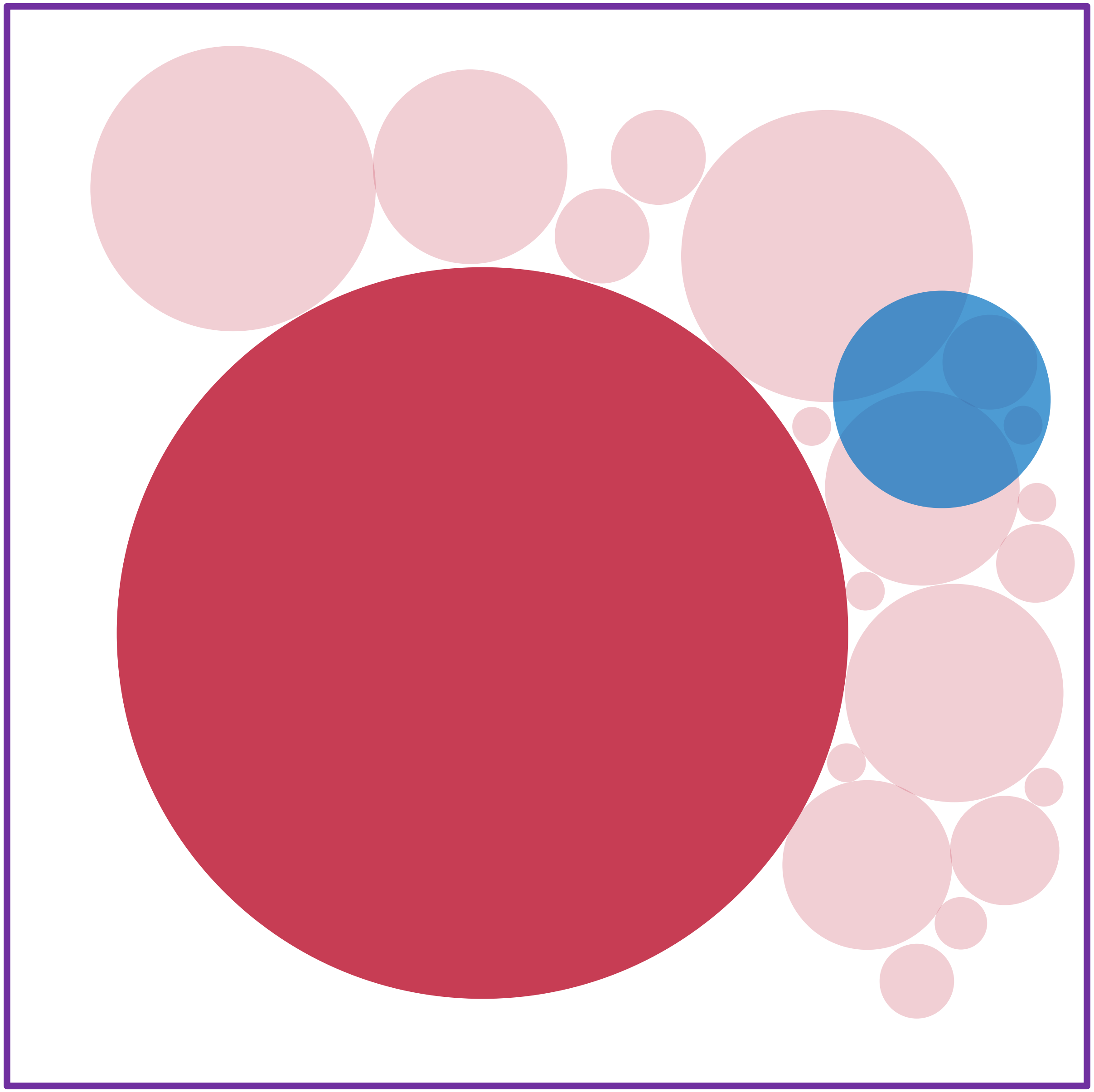


Our Algorithm

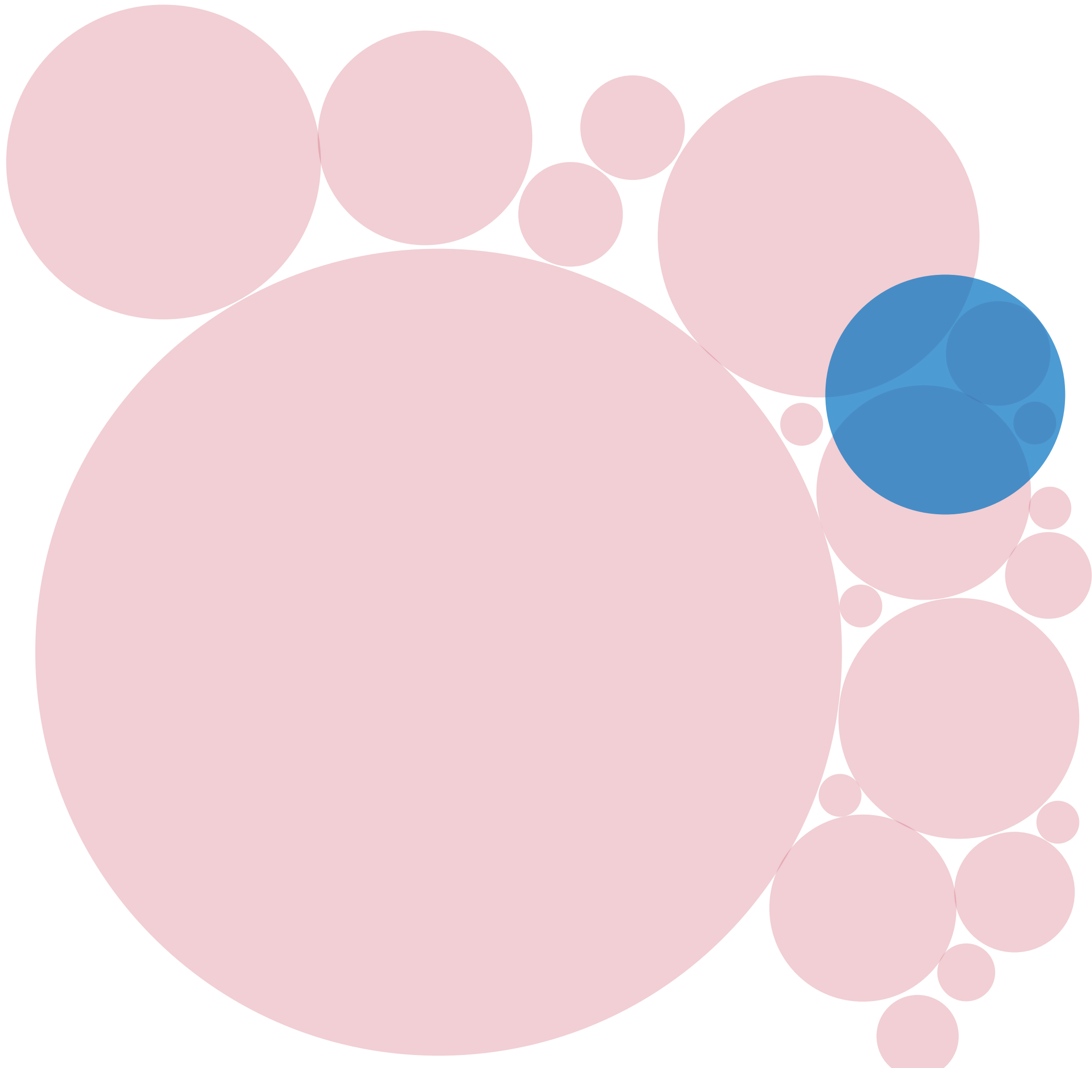


Our Algorithm

L e v e l 0



Our Algorithm



Analysis

For each sphere $s \in \mathbf{B}$:

Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
```

Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
```

Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s
```

Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s
            For all spheres s_k ∈ c_j
```

Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s
            For all spheres s_k ∈ c_j
                Compute overlap volume for s and s_k
```

Analysis

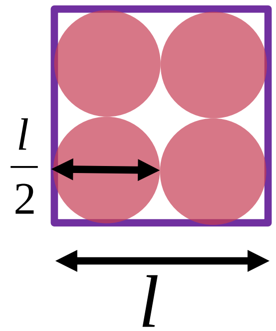
```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s
            For all spheres s_k ∈ c_j
                Compute overlap volume for s and s_k
```

Compute overlap volume for s and s_k $\Rightarrow O(1)$

Analysis

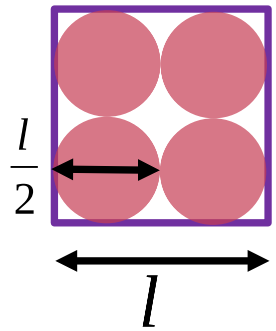
```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s
            For all spheres s_k ∈ c_j
                Compute overlap volume for s and s_k
```

$\Rightarrow O(1)$



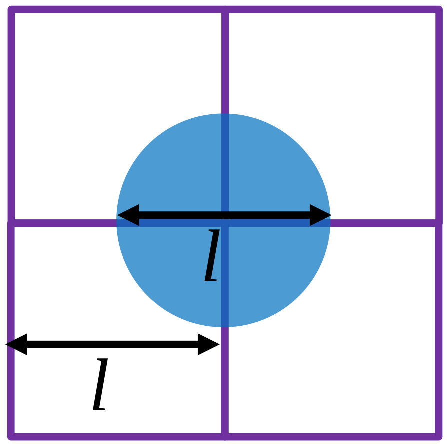
Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s
            For all spheres s_k ∈ c_j                        ⇒ O(1)
                Compute overlap volume for s and s_k         ⇒ O(1)
```



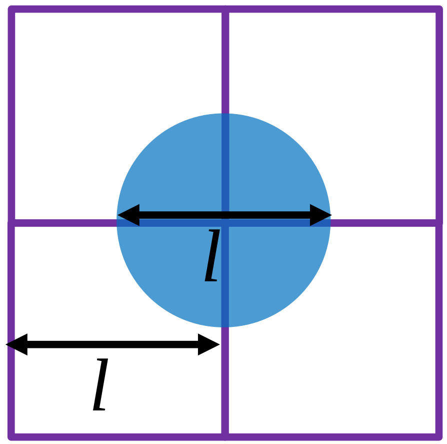
Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s
            For all spheres s_k ∈ c_j                          ⇒ O(1)
                Compute overlap volume for s and s_k          ⇒ O(1)
```



Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s      ⇒ O(1)
            For all spheres s_k ∈ c_j                        ⇒ O(1)
                Compute overlap volume for s and s_k         ⇒ O(1)
```

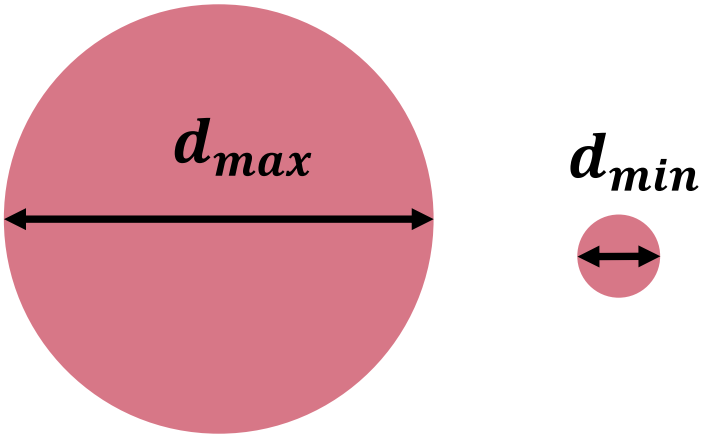


Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s        ⇒ O(1)
            For all spheres s_k ∈ c_j                         ⇒ O(1)
                Compute overlap volume for s and s_k          ⇒ O(1)
```

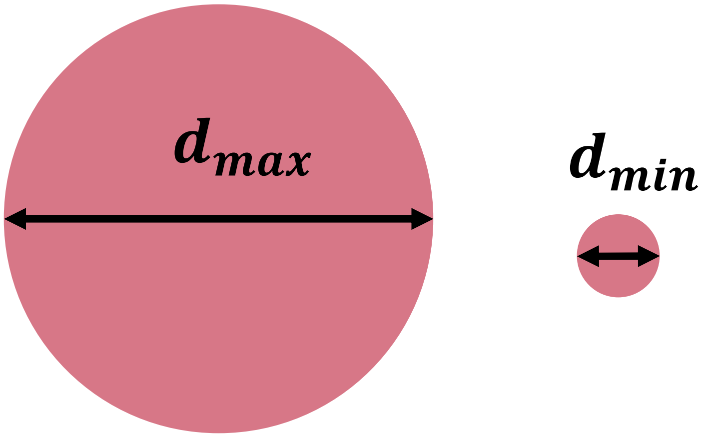
Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s          ⇒ O(1)
            For all spheres s_k ∈ c_j                           ⇒ O(1)
                Compute overlap volume for s and s_k            ⇒ O(1)
```



Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s        ⇒ O(1)
            For all spheres s_k ∈ c_j                        ⇒ O(1)
                Compute overlap volume for s and s_k         ⇒ O(1)
```



$$\Rightarrow O\left(\log \frac{d_{\max}}{d_{\min}}\right)$$

Analysis

For each sphere $s \in B$:

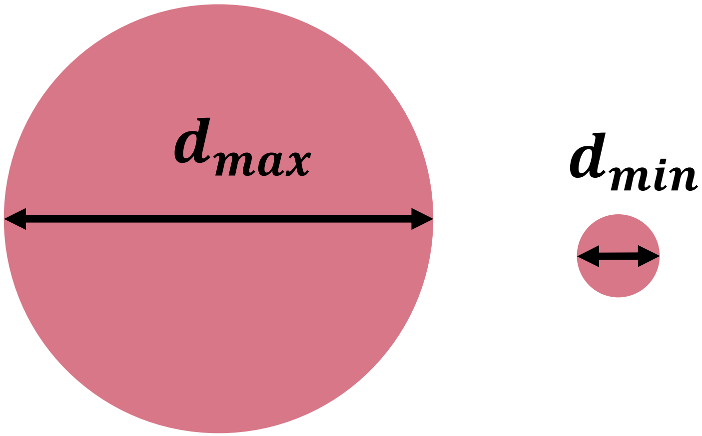
  Compute hierarchy level l

  For all levels $l \le l_i \le l_{max}$: $\Rightarrow O(1)$

    For all cells c_j in level l_i overlapped by s $\Rightarrow O(1)$

      For all spheres $s_k \in c_j$ $\Rightarrow O(1)$

        Compute overlap volume for s and s_k $\Rightarrow O(1)$



$$\Rightarrow O\left(\log \frac{d_{\max}}{d_{\min}}\right)$$

Analysis

```
For each sphere s ∈ B:
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by s
            For all spheres s_k ∈ c_j
                Compute overlap volume for s and s_k
```

For each sphere $s \in B$:

- Compute hierarchy level l
- For all levels $l \leq l_i \leq l_{max}$: $\Rightarrow O(1)$
  - For all cells c_j in level l_i overlapped by s $\Rightarrow O(1)$
    - For all spheres $s_k \in c_j$ $\Rightarrow O(1)$
      - Compute overlap volume for s and s_k $\Rightarrow O(1)$

Analysis

```
For each sphere s ∈ B:                                        ⇒ O(n)
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:                           ⇒ O(1)
        For all cells c_j in level l_i overlapped by s        ⇒ O(1)
            For all spheres s_k ∈ c_j                         ⇒ O(1)
                Compute overlap volume for s and s_k          ⇒ O(1)
```

Analysis

```
For each sphere s ∈ B:                                        ⇒ O(n)
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:                           ⇒ O(1)
        For all cells c_j in level l_i overlapped by s        ⇒ O(1)
            For all spheres s_k ∈ c_j                         ⇒ O(1)
                Compute overlap volume for s and s_k          ⇒ O(1)
```

$$\textit{Total Time}: \quad O(n)$$

Analysis

```
For each sphere s ∈ B:                                    ⇒ O(n)
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:                       ⇒ O(1)
        For all cells c_j in level l_i overlapped by s    ⇒ O(1)
            For all spheres s_k ∈ c_j                     ⇒ O(1)
                Compute overlap volume for s and s_k      ⇒ O(1)
```

For each sphere $s \in B$: $\Rightarrow O(n)$

Compute hierarchy level l

For all levels $l \leq l_i \leq l_{max}$: $\Rightarrow O(1)$

For all cells c_j in level l_i overlapped by s $\Rightarrow O(1)$

For all spheres $s_k \in c_j$ $\Rightarrow O(1)$

Compute overlap volume for s and s_k $\Rightarrow O(1)$

$$\textit{Total Time}: \quad O(n)$$

Analysis

```
In Parallel for all spheres s ∈ B:                          ⇒ O(n)
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:                          ⇒ O(1)
        For all cells c_j in level l_i overlapped by s       ⇒ O(1)
            For all spheres s_k ∈ c_j                        ⇒ O(1)
                Compute overlap volume for s and s_k         ⇒ O(1)
```

Total Time: $O(n)$

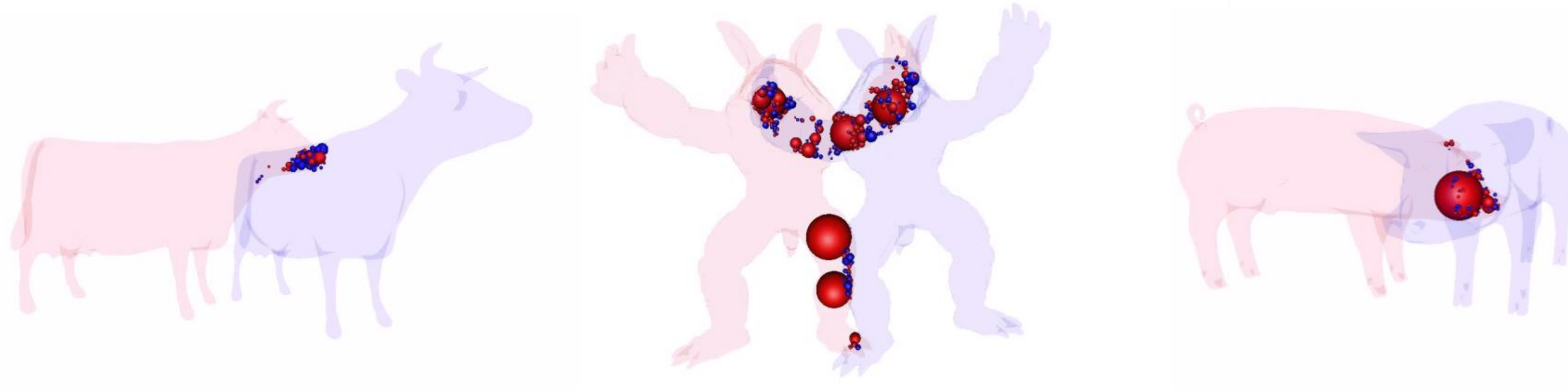
Analysis

```
In Parallel for all spheres s ∈ B:                                  ⇒ O(n)
    Compute hierarchy level l
    For all levels l ≤ l_i ≤ l_max:                                 ⇒ O(1)
        For all cells c_j in level l_i overlapped by s              ⇒ O(1)
            For all spheres s_k ∈ c_j                               ⇒ O(1)
                Compute overlap volume for s and s_k                ⇒ O(1)
```

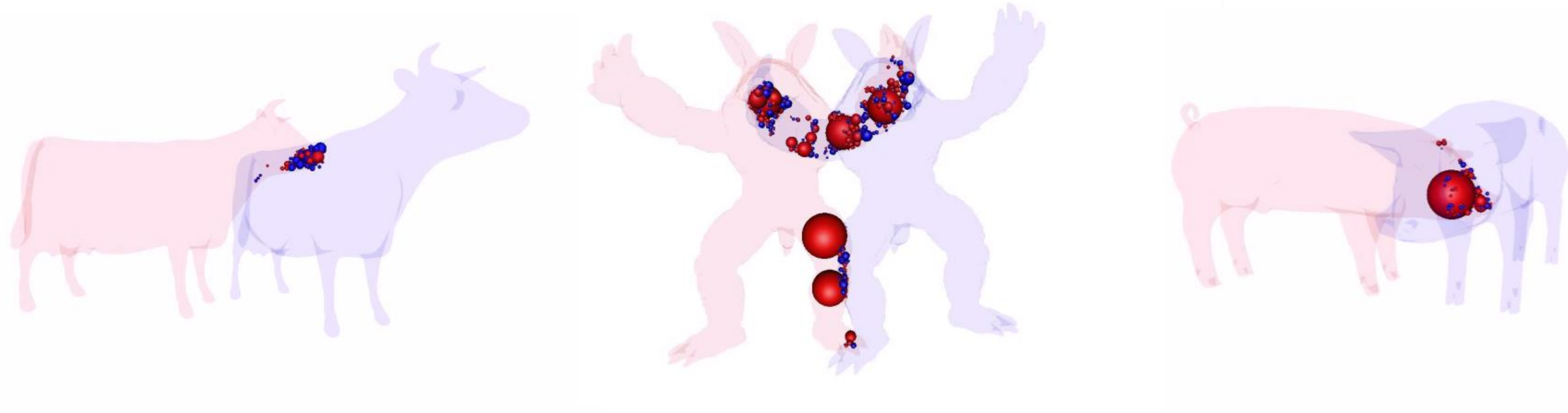
Total Time: $O(n)$

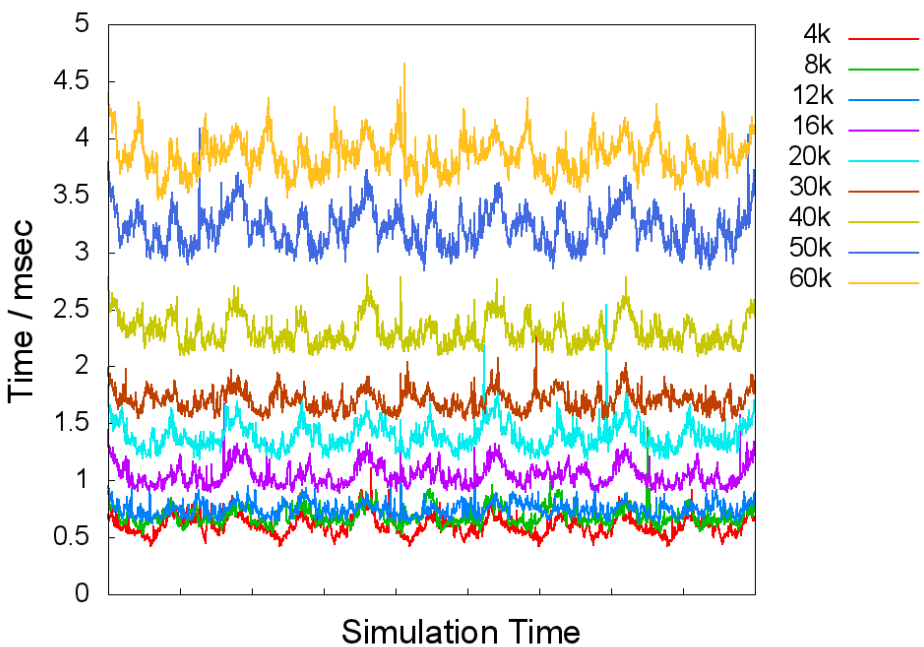
Total Parallel Time: $O(1)$

Results

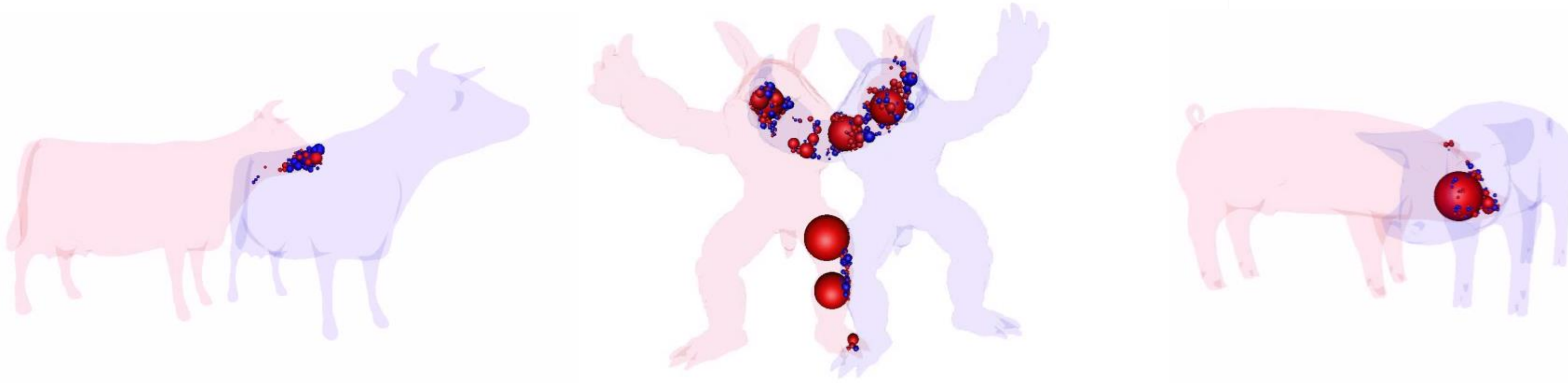


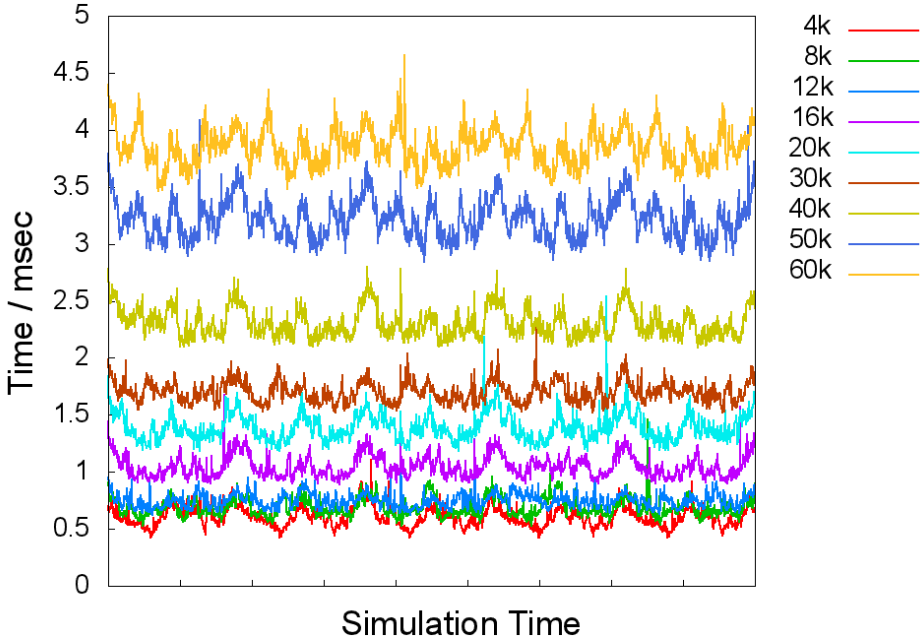
Results

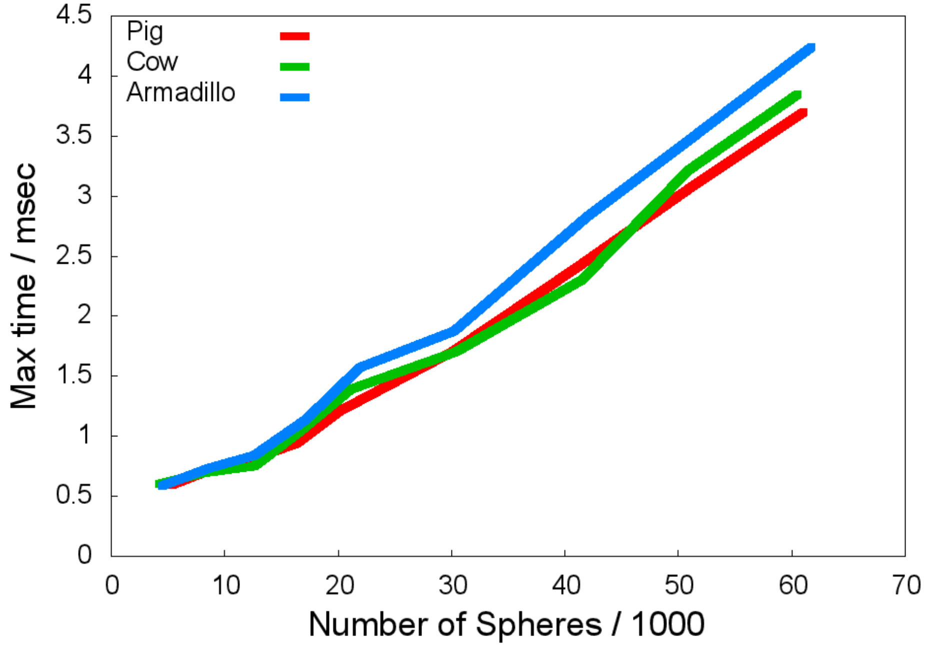



Time / msec
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Simulation Time
4k
8k
12k
16k
20k
30k
40k
50k
60k

Results

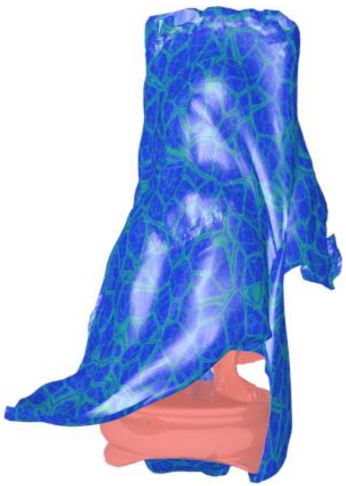


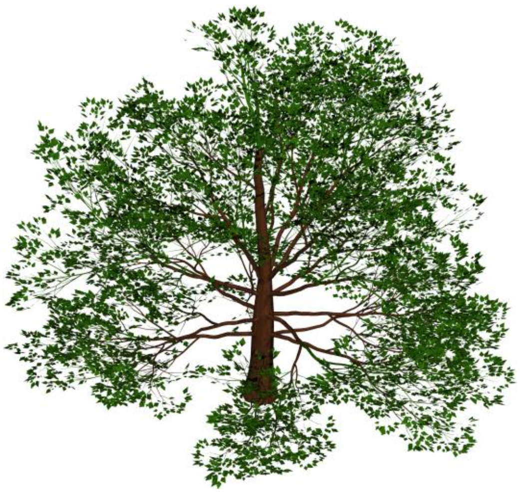

5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Time / msec
Simulation Time
4k
8k
12k
16k
20k
30k
40k
50k
60k


4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Max time / msec
0
10
20
30
40
50
60
70
Number of Spheres / 1000
Pig
Cow
Armadillo

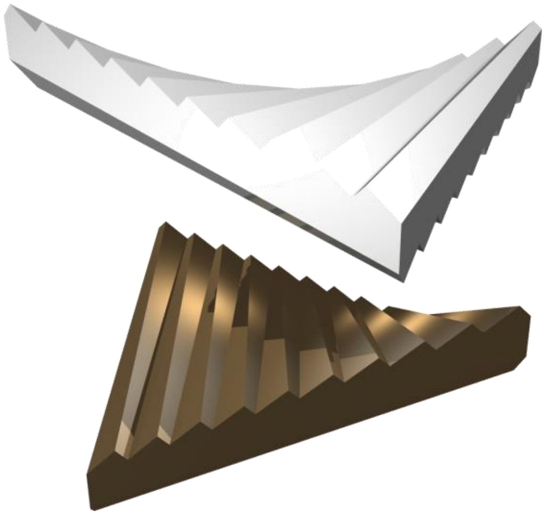
Challenges

- Precondition for ISTs:
  - Watertight
  - Rigid (?)



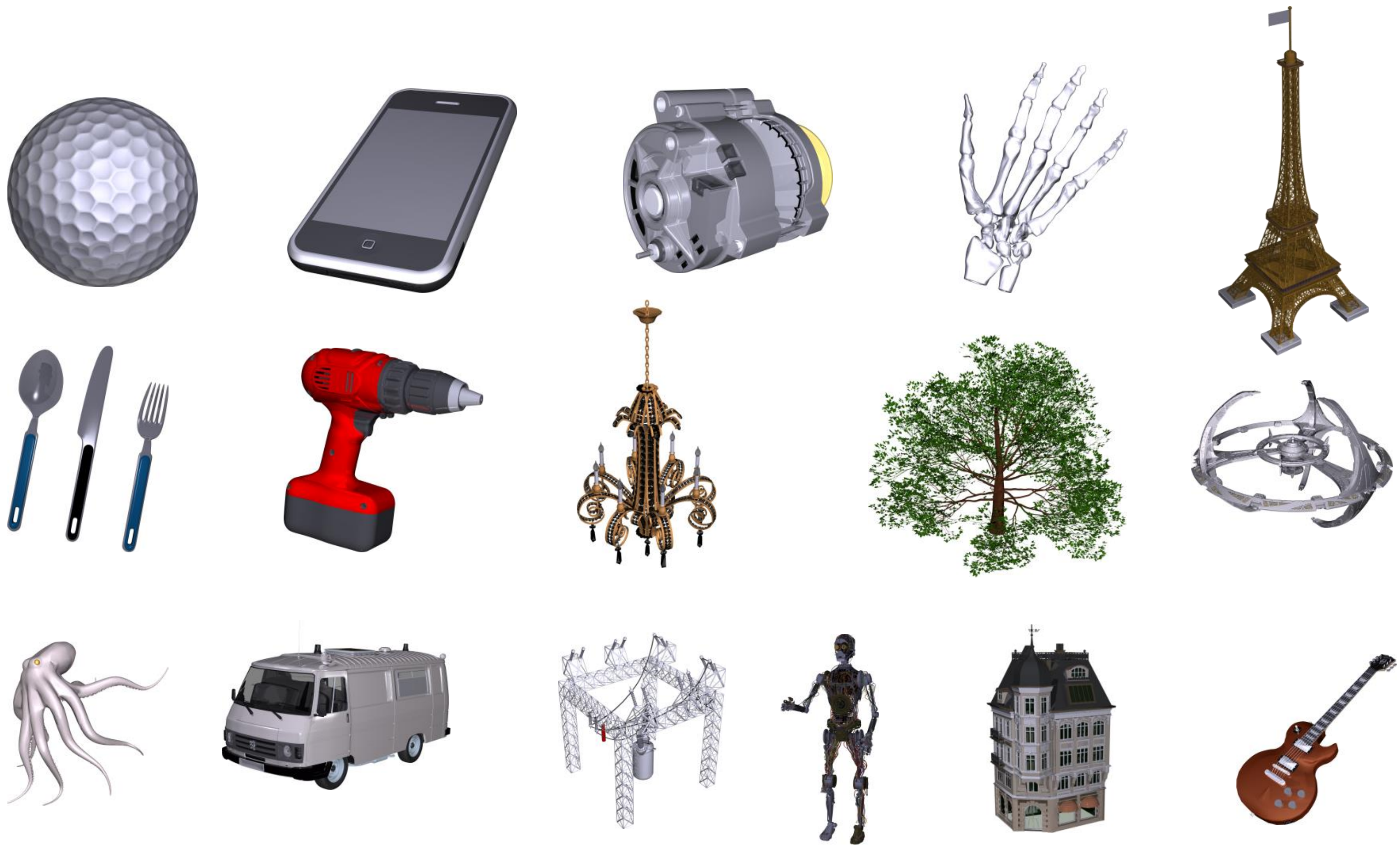


Collision Detection Worst Case



$$O(n^2)$$

What about these objects?

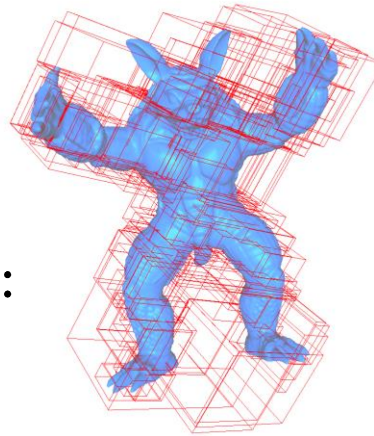


Previous Works

- Distance of convex polytopes [Lin & Canny, 1991]:

  Worst case: $O(\sqrt{n})$, n = # faces

Previous Works



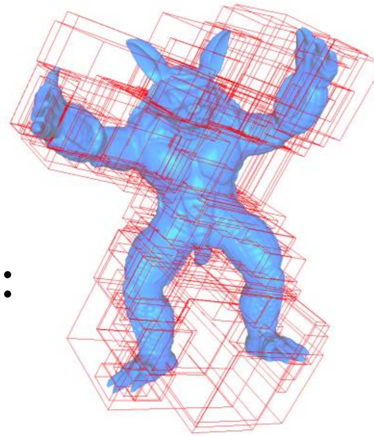
- Distance of convex polytopes [Lin & Canny, 1991]:

  Worst case: $O(\sqrt{n})$, n = # faces

- All intersections of n convex polytopes [Suri et al., 1998]:

  $O((n+k)\log^2 n)$, k = # intersecting pairs

Previous Works



- Distance of convex polytopes [Lin & Canny, 1991]:

  Worst case: $O(\sqrt{n})$, n = # faces

- All intersections of n convex polytopes [Suri et al., 1998]:

  $O((n+k)\log^2 n)$, k = # intersecting pairs

- Expected $O(\log n)$ depending on bounding volume and object configuration [Weller et al., 2006]

Our Contribution

- A novel geometric predicate that defines a class of „well-shaped“ objects

Our Contribution

- A novel geometric predicate that defines a class of „well-shaped“ objects

- Proof to show that objects in that class have $O(n)$ intersections

Our Contribution

- A novel geometric predicate that defines a class of „well-shaped“ objects
  - Predicate needs to consider only a single object
  - Easy to check
  - Contains (almost) all practically relevant objects
- Proof to show that objects in that class have $O(n)$ intersections

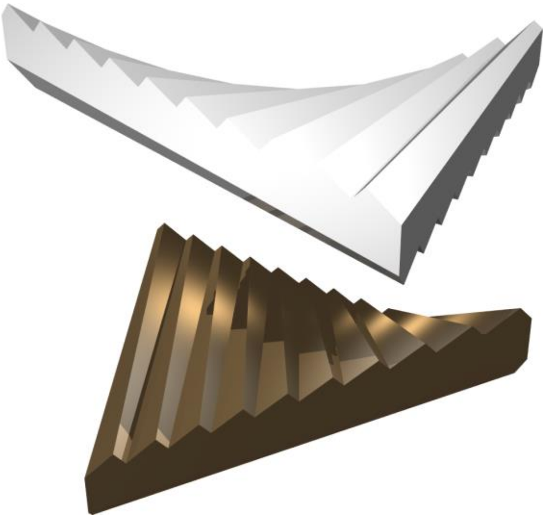
Our Contribution

- A novel geometric predicate that defines a class of „well-shaped“ objects
  - Predicate needs to consider only a single object
  - Easy to check
  - Contains (almost) all practically relevant objects

- Proof to show that objects in that class have $O(n)$ intersections
- New algorithm with (almost) linear running time for objects fullfilling our predicate

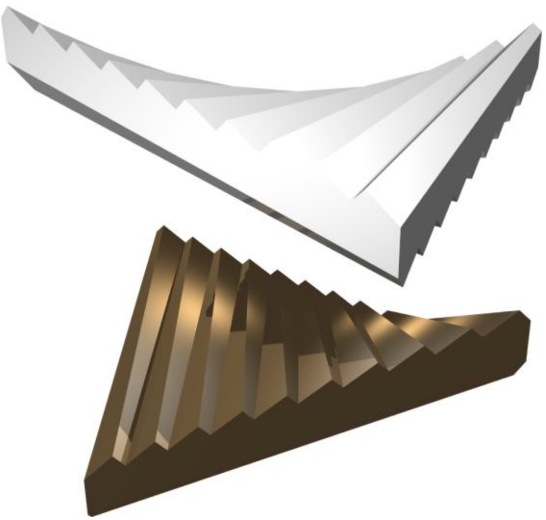
Basic Idea

- What makes objects like the Chazelle polyhedron so complex to check for collision?



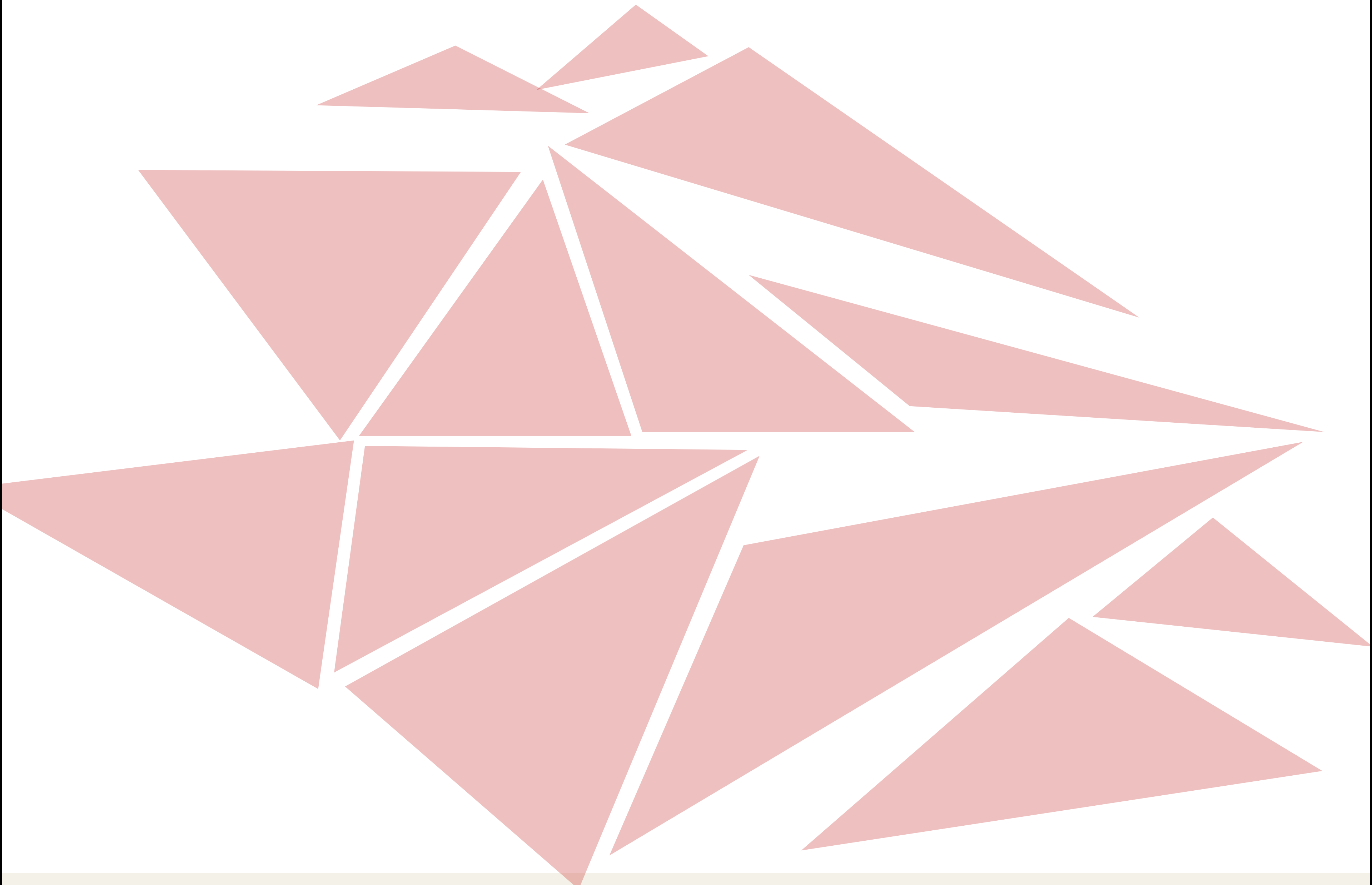
Basic Idea

- What makes objects like the Chazelle polyhedron so complex to check for collision?

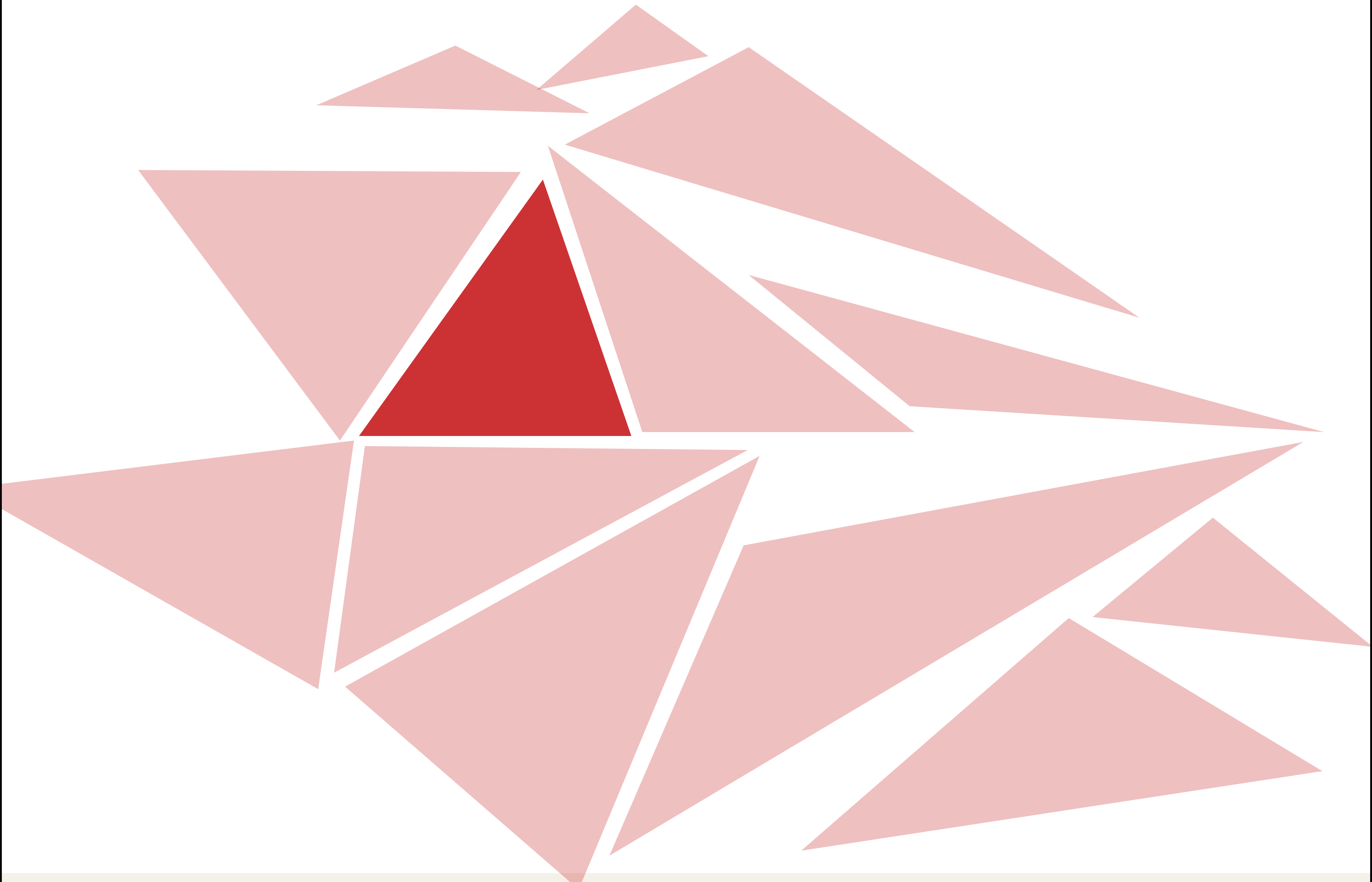


- There can be $O(n)$ polygons in the neighbourhood of each polygon

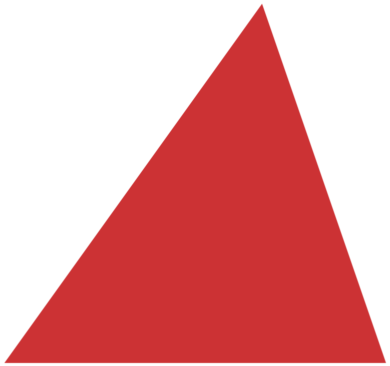
Our Geometric Predicate



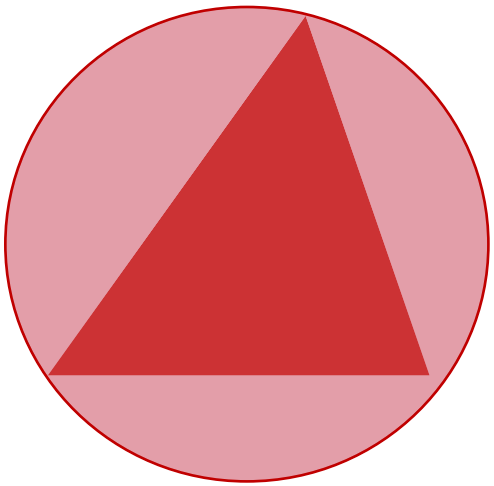
Our Geometric Predicate



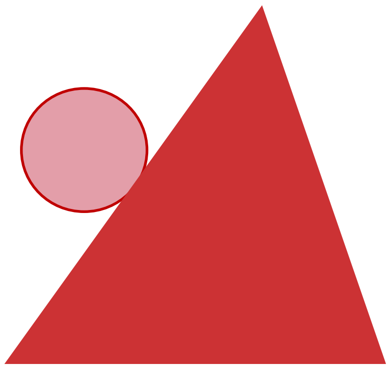
Our Geometric Predicate



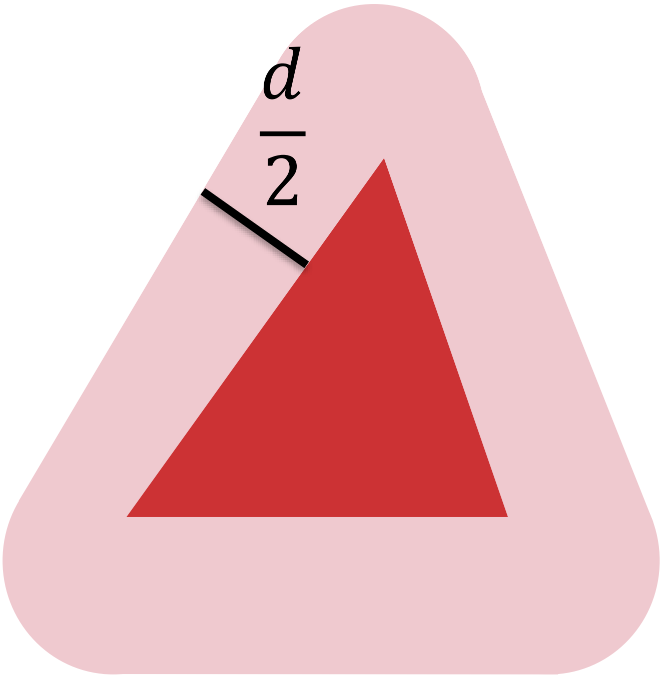
Our Geometric Predicate



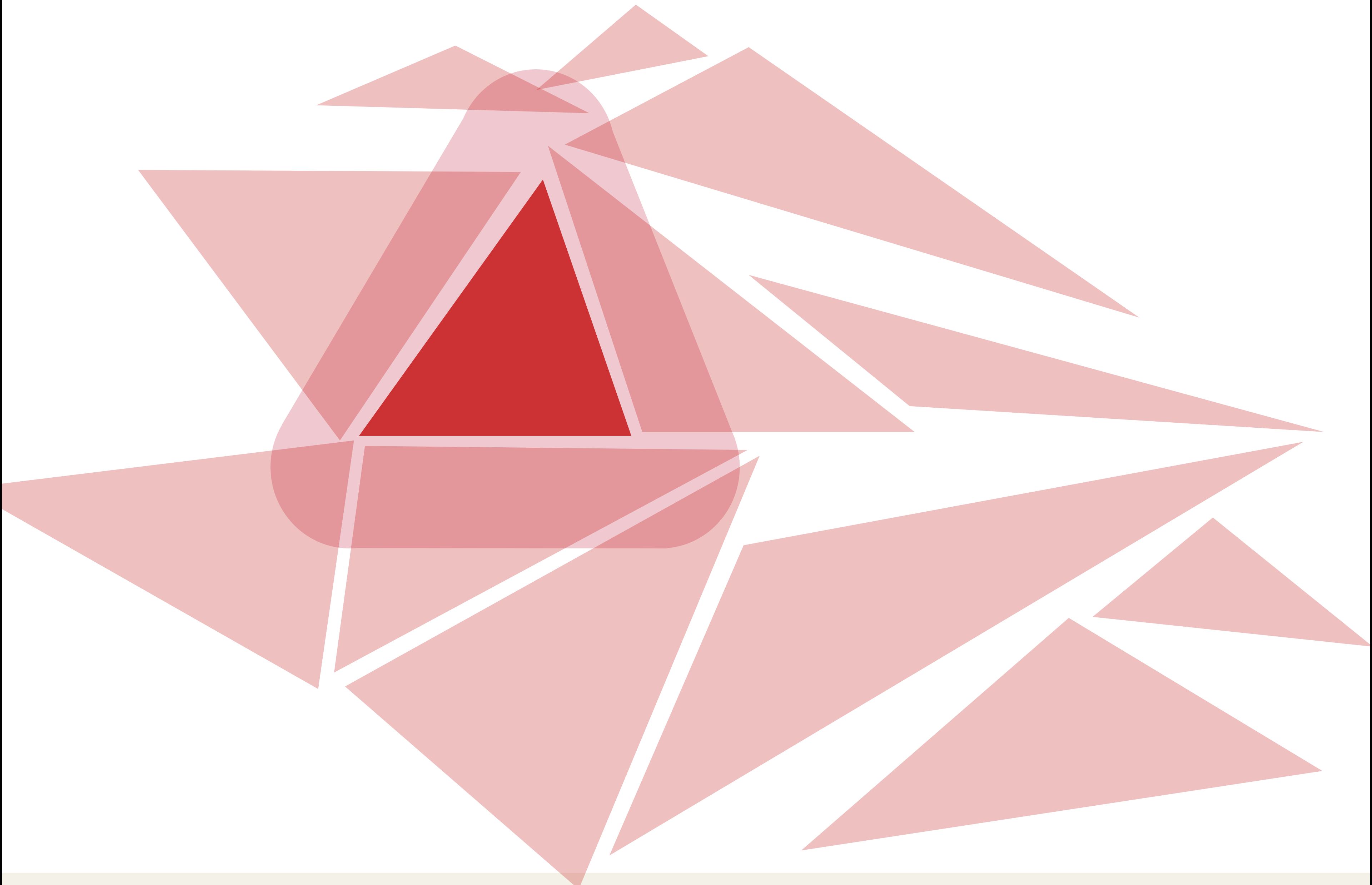
Our Geometric Predicate



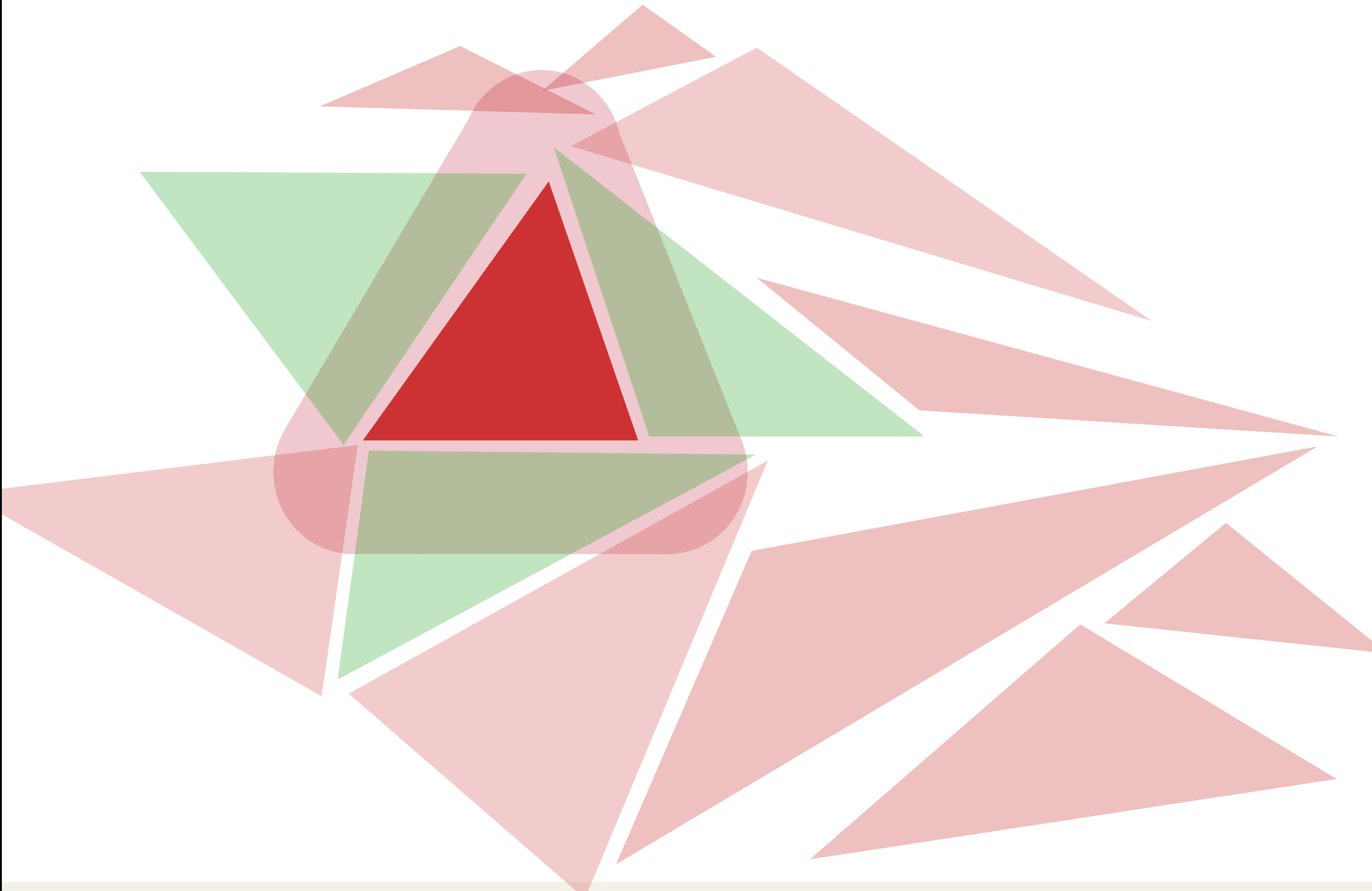
Our Geometric Predicate



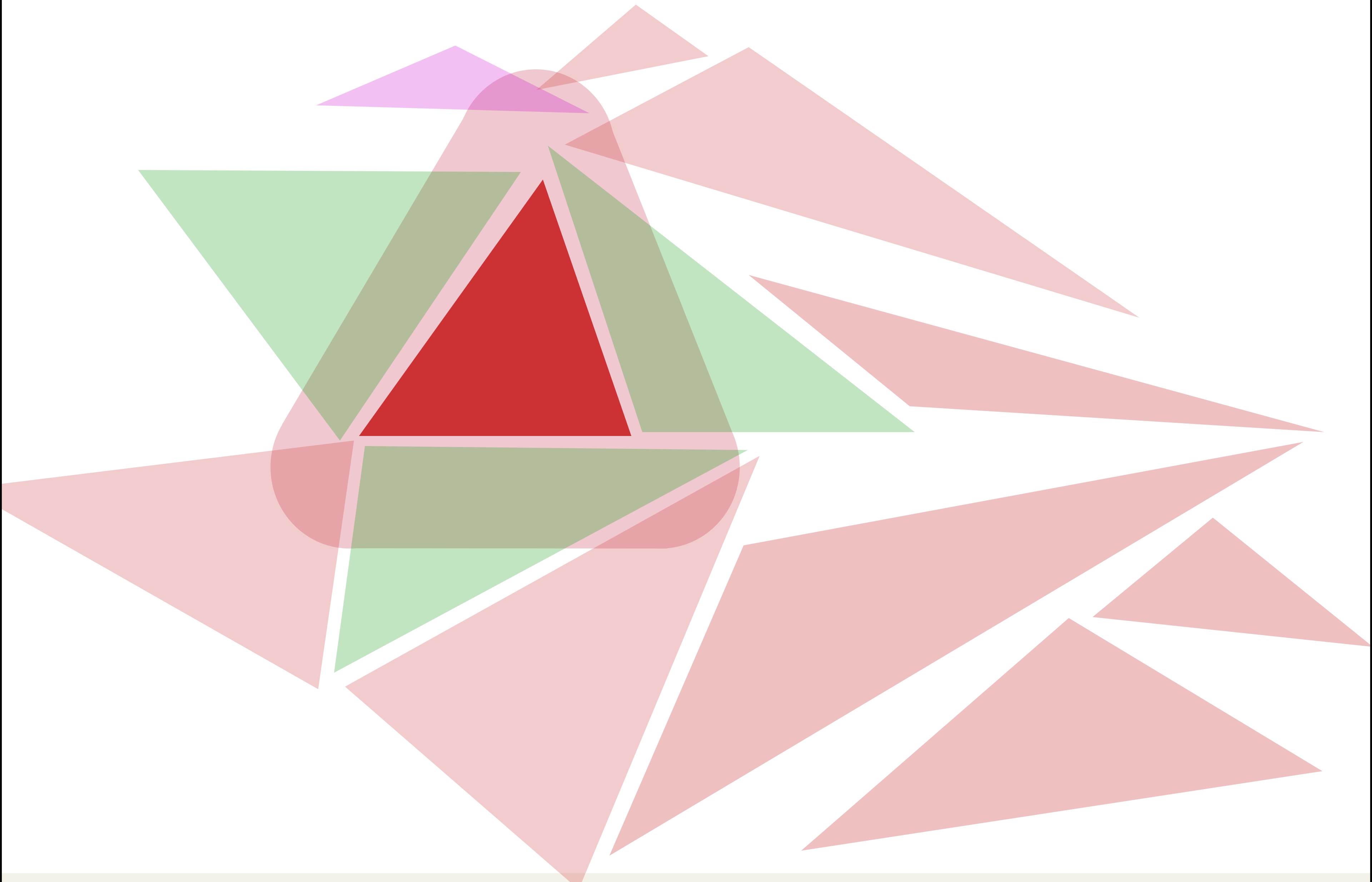
Our Geometric Predicate



Our Geometric Predicate

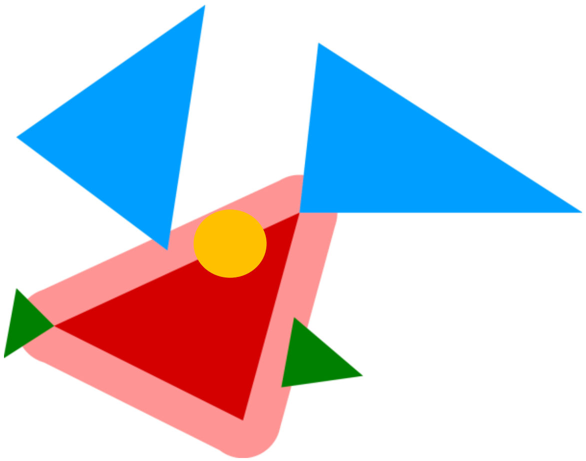


Our Geometric Predicate

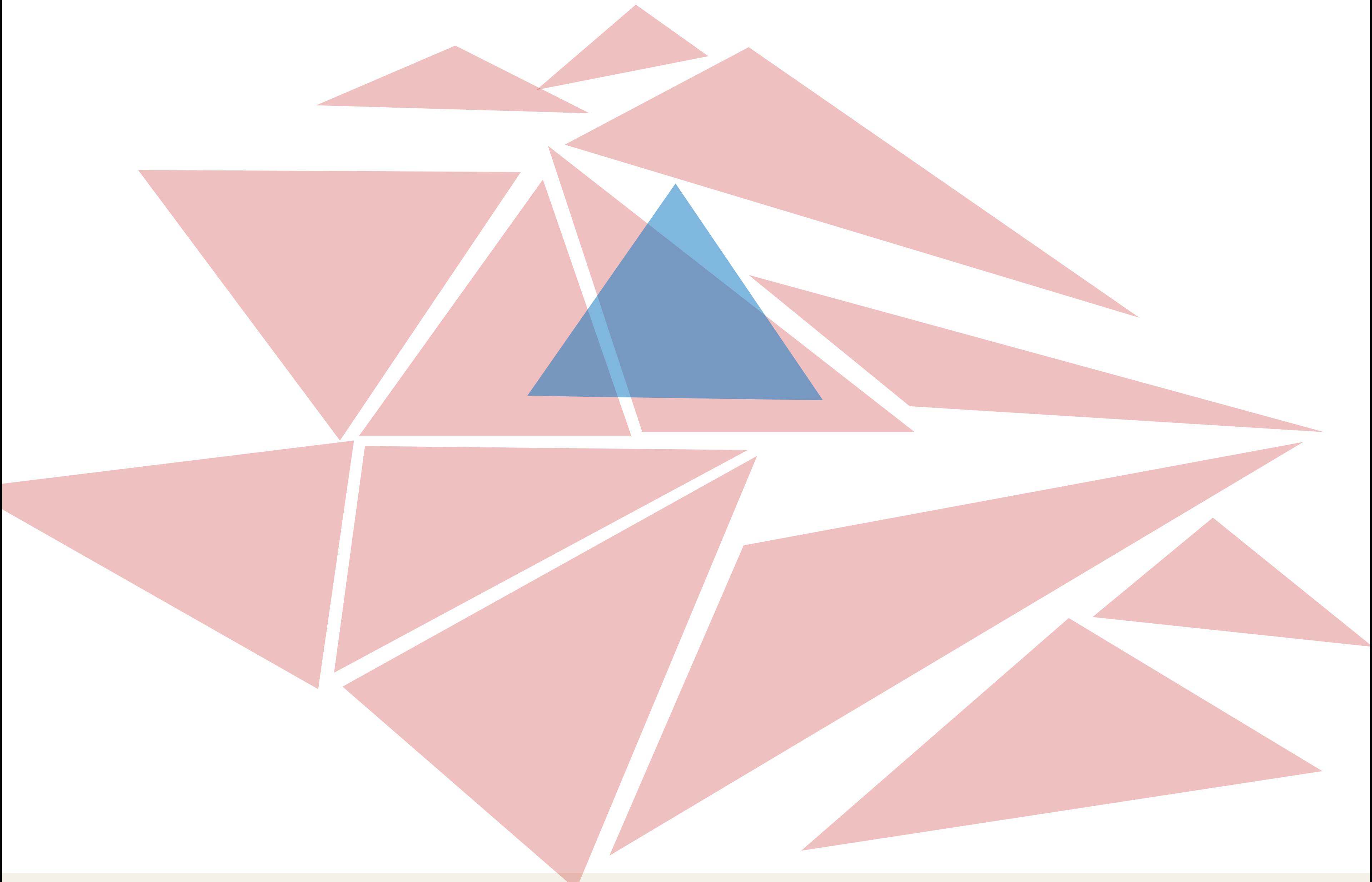


Definition: k-free

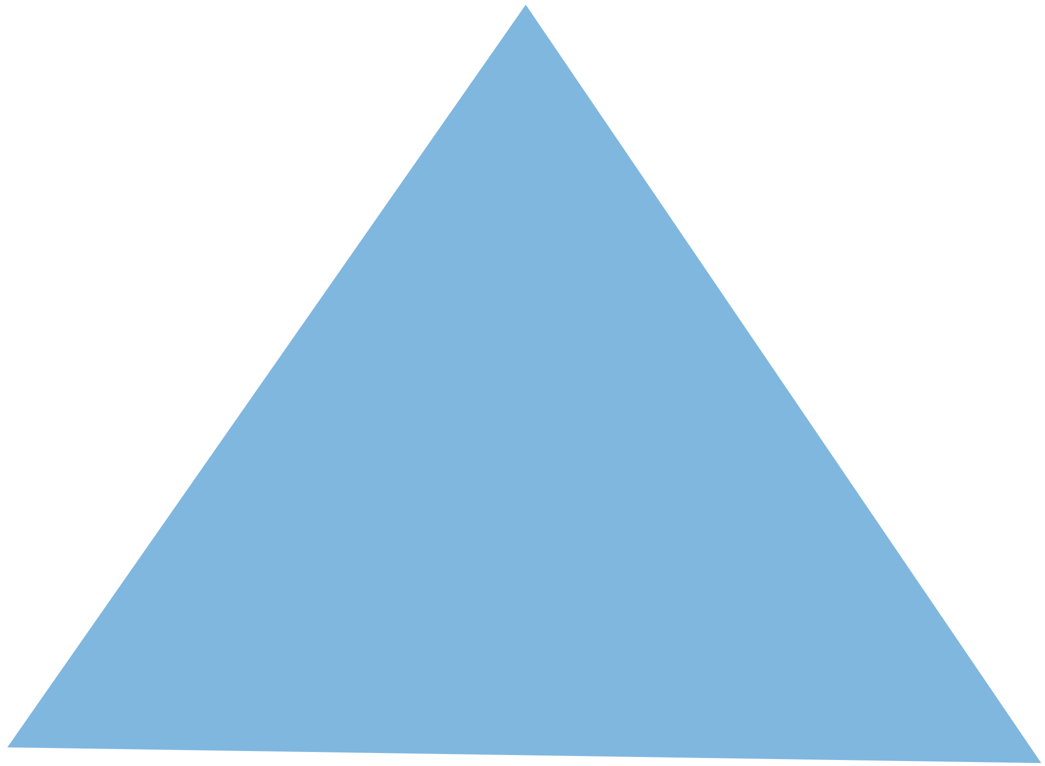
- Let t $\in A$ be a triangle in a triangle set A and $k > 0$ some constant. Let s be a sphere with diameter $\frac{d}{2}$, where d is the diameter of the smallest enclosing sphere of t. We call t k-free if $\left|\{t_j \in A; r \leq r_j \; and \; t_j \cap (t \oplus s) \neq \emptyset\}\right| < k$ where d_j is diameter of the smallest enclosing sphere of triangle t_j and $t \oplus s$ is the Minkowski sum of s and t.
- Accordingly, we call the whole set of triangles A k-free, if all triangles $t_j \in A$ are k-free.



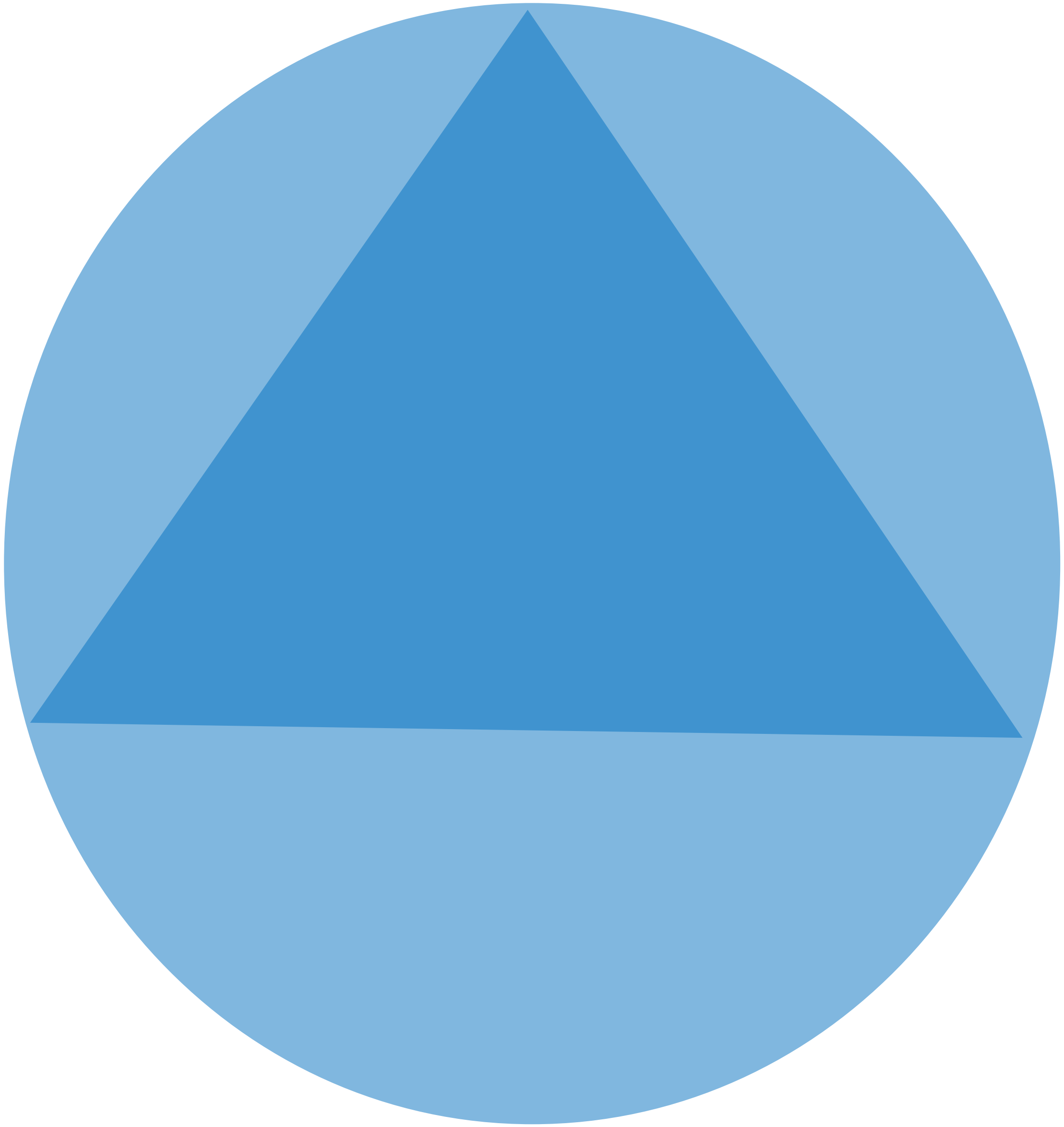
Our Geometric Predicate



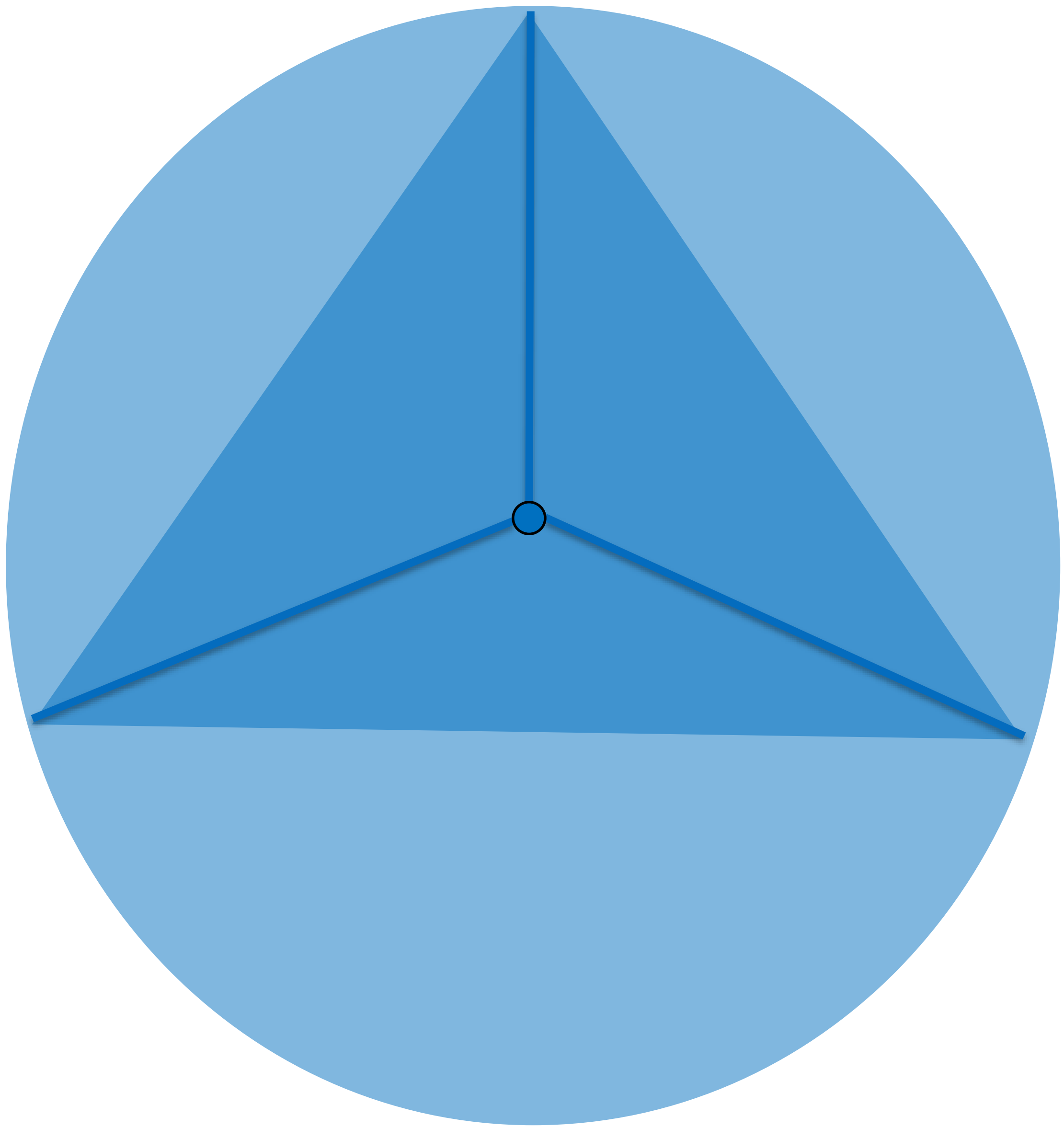
Our Geometric Predicate



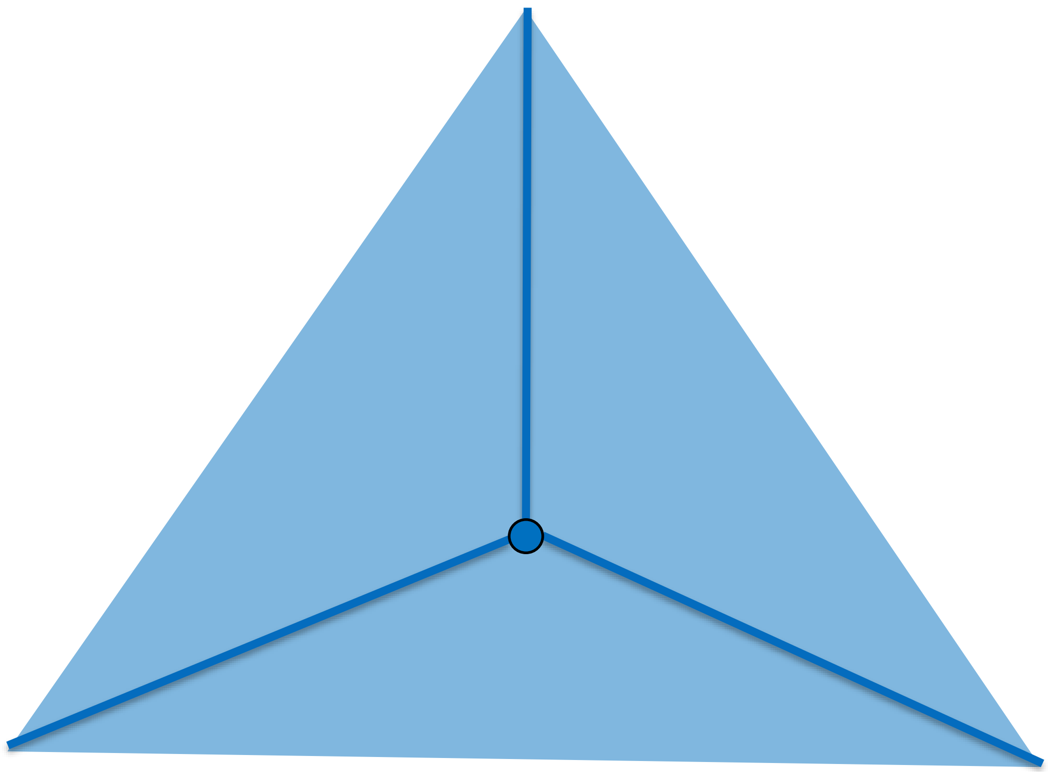
Our Geometric Predicate



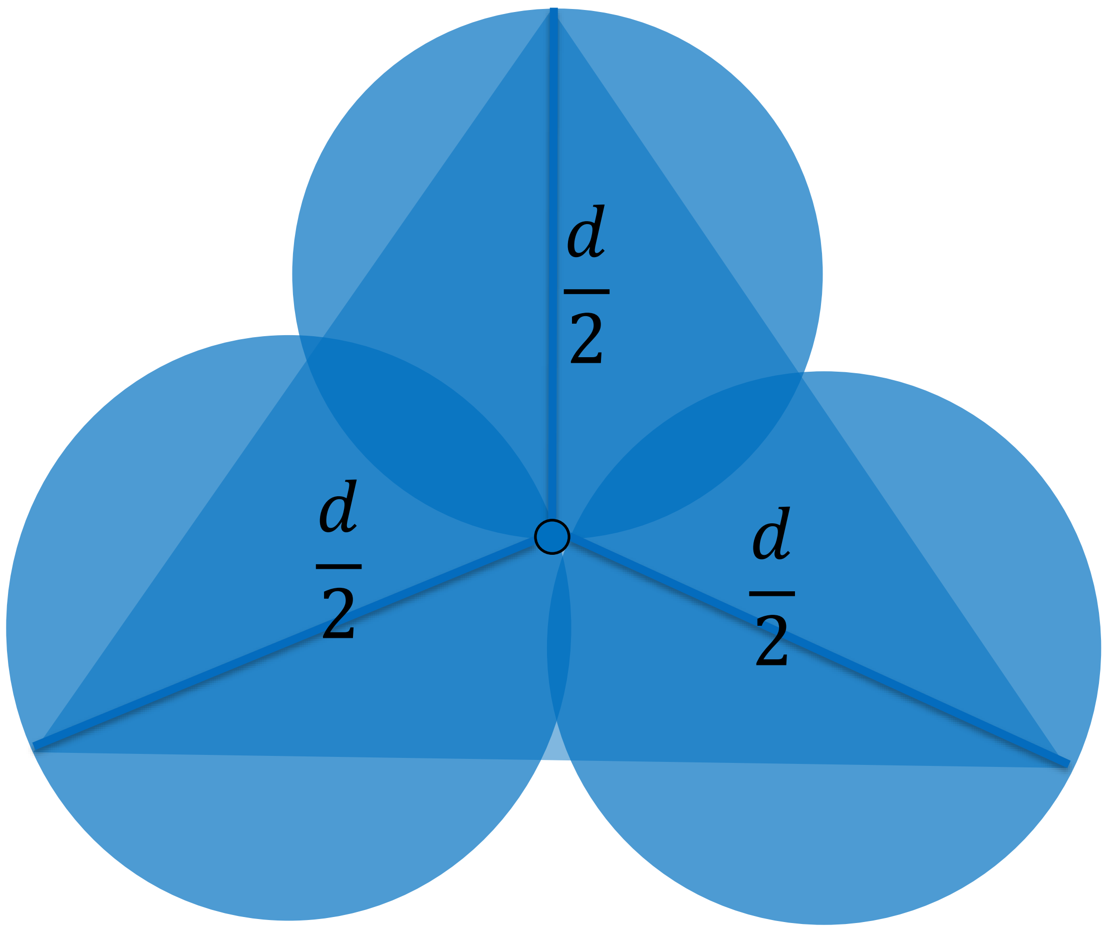
Our Geometric Predicate



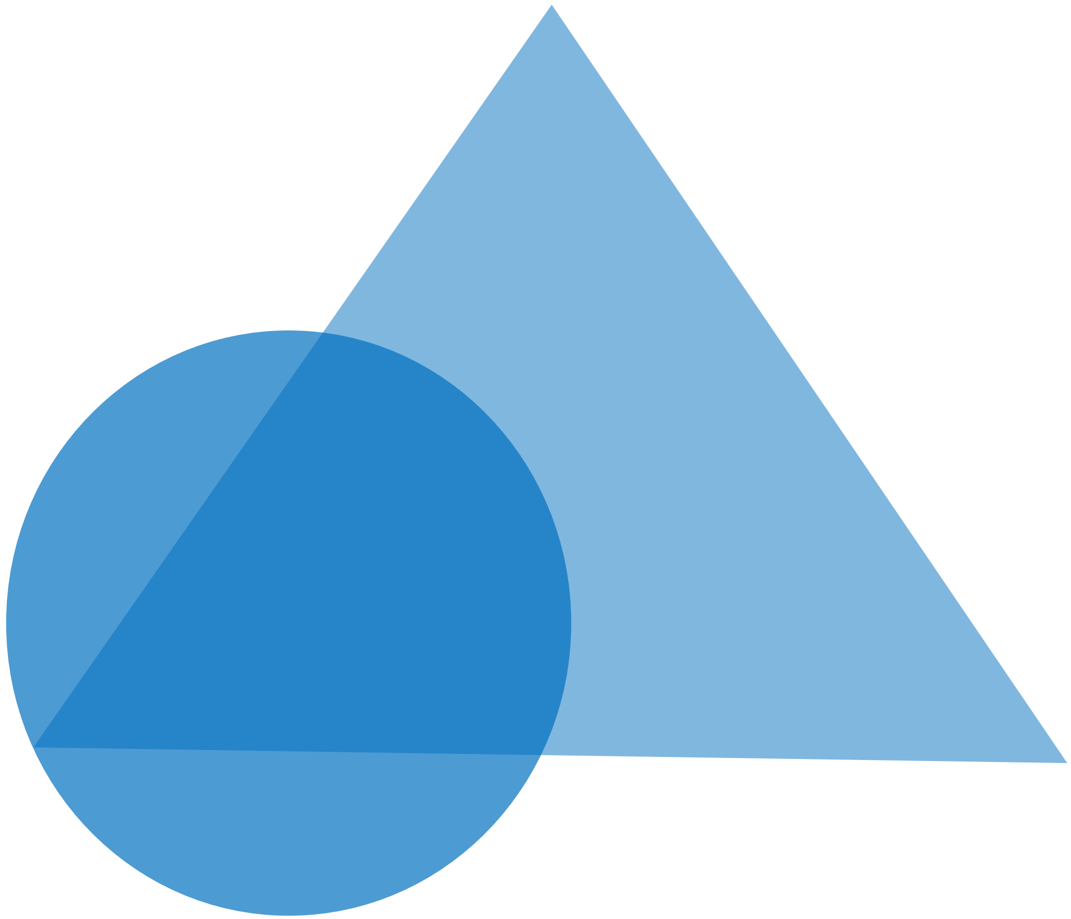
Our Geometric Predicate



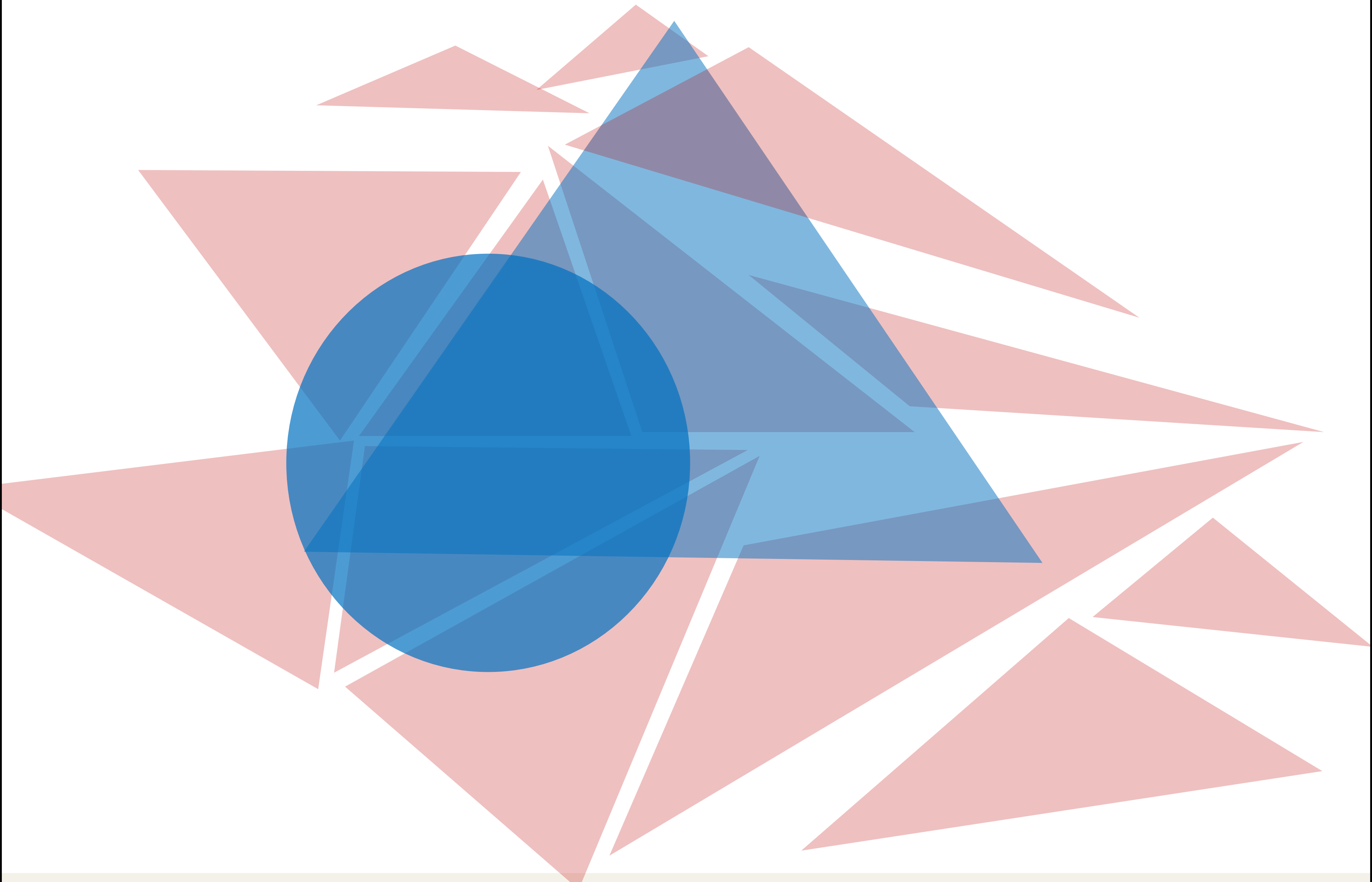
Our Geometric Predicate



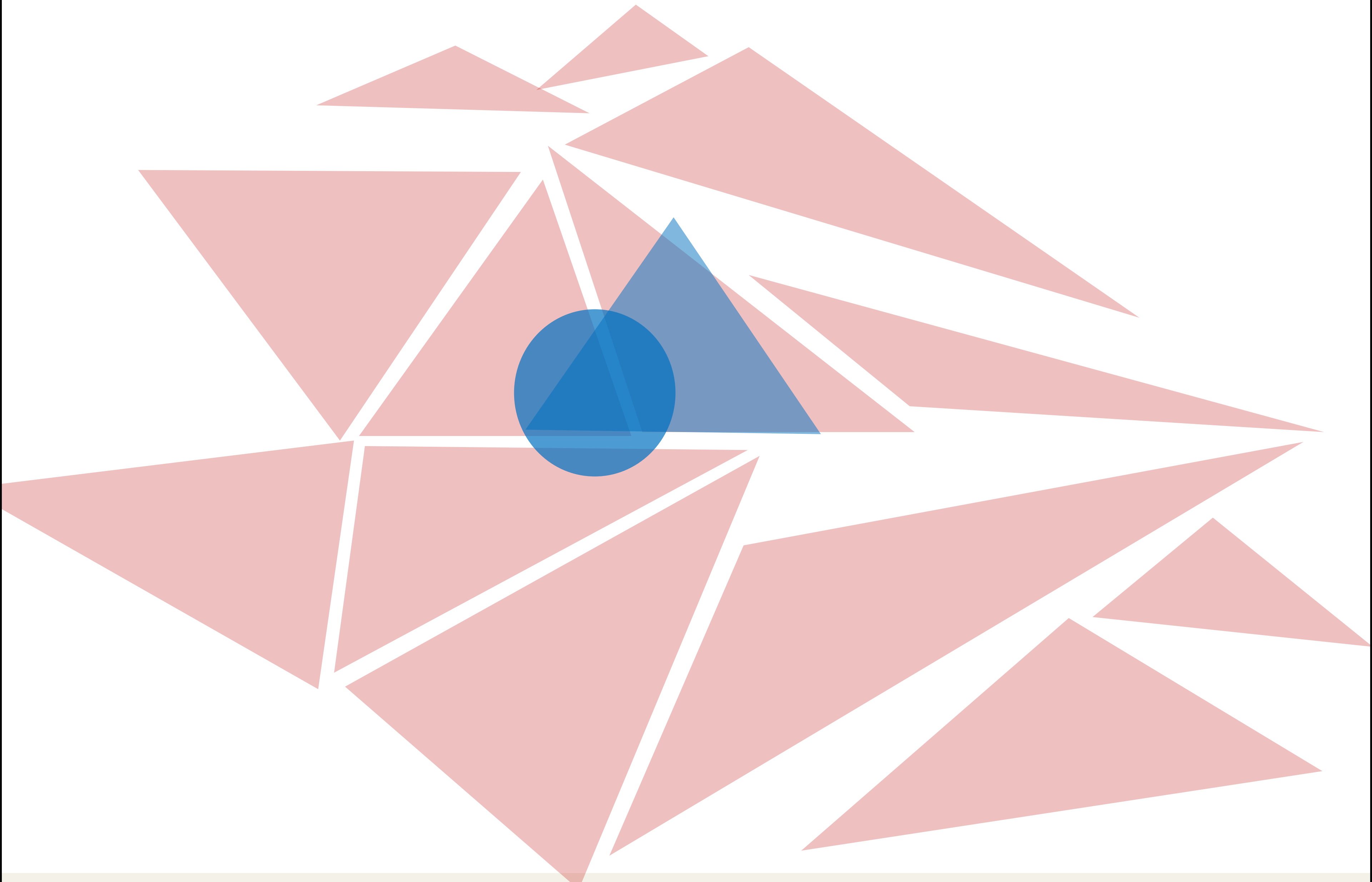
Our Geometric Predicate



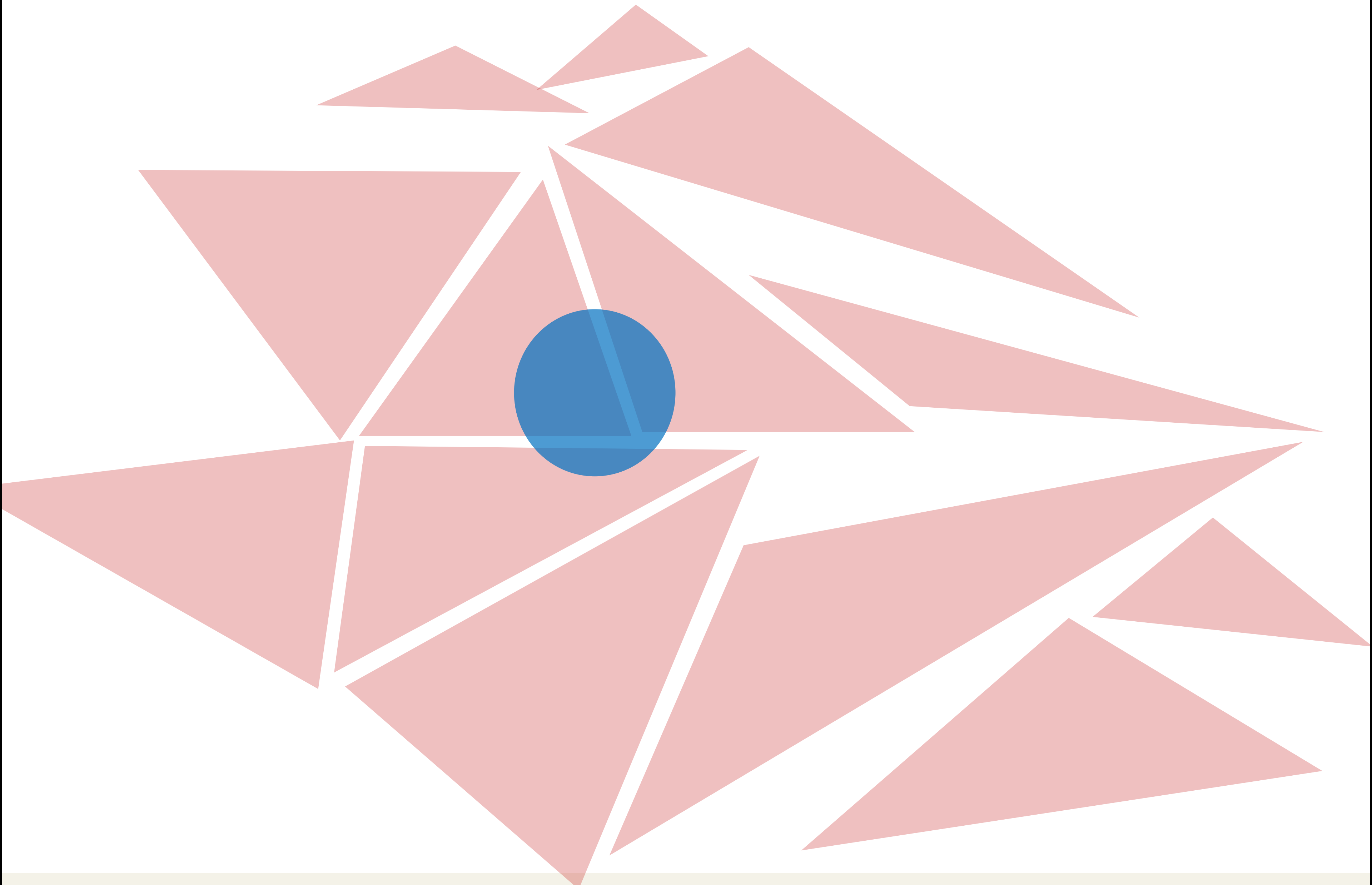
Our Geometric Predicate



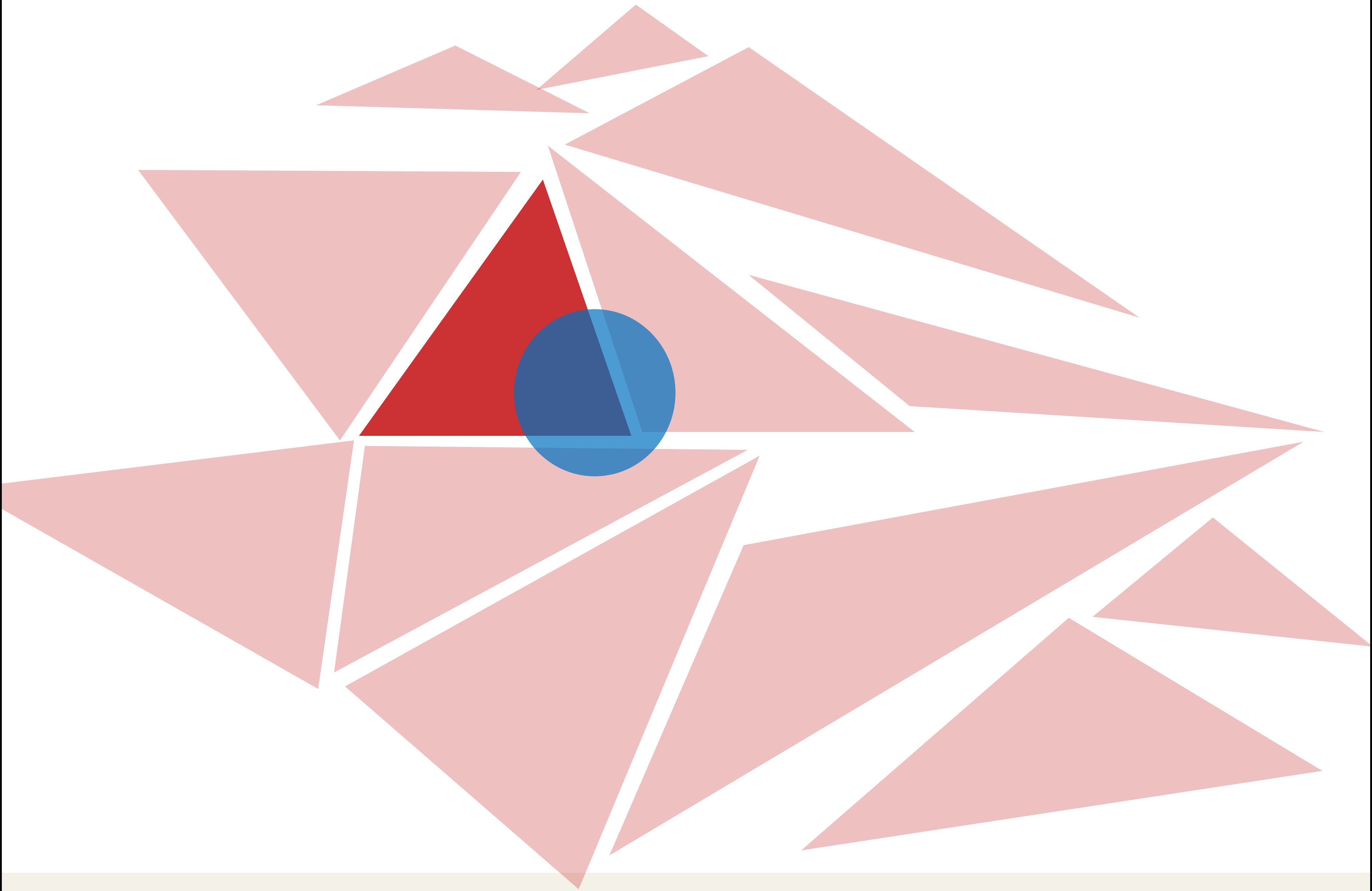
Our Geometric Predicate



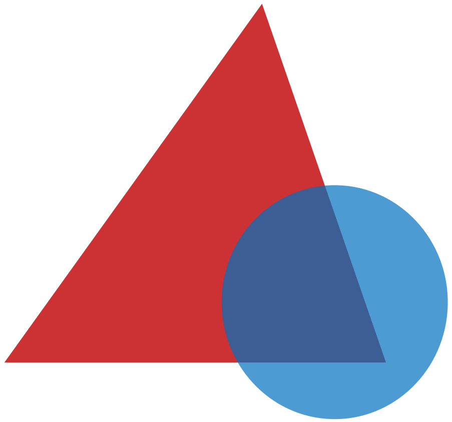
Our Geometric Predicate



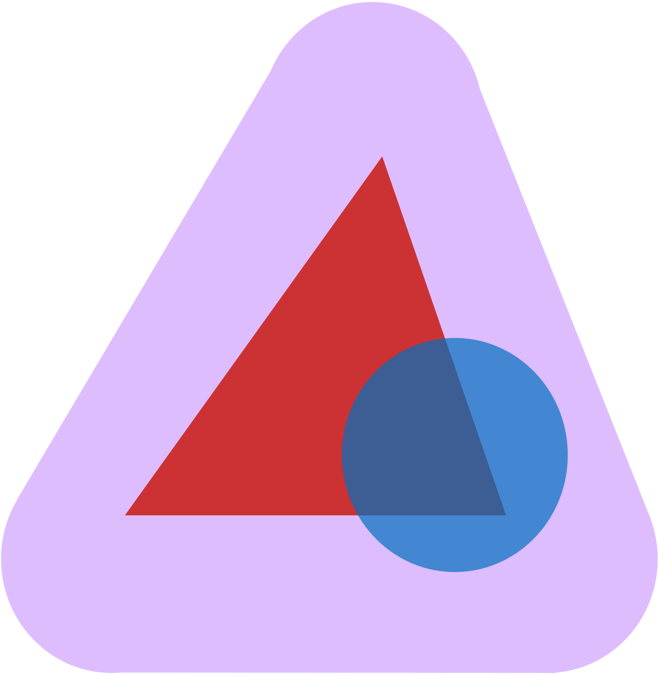
Our Geometric Predicate



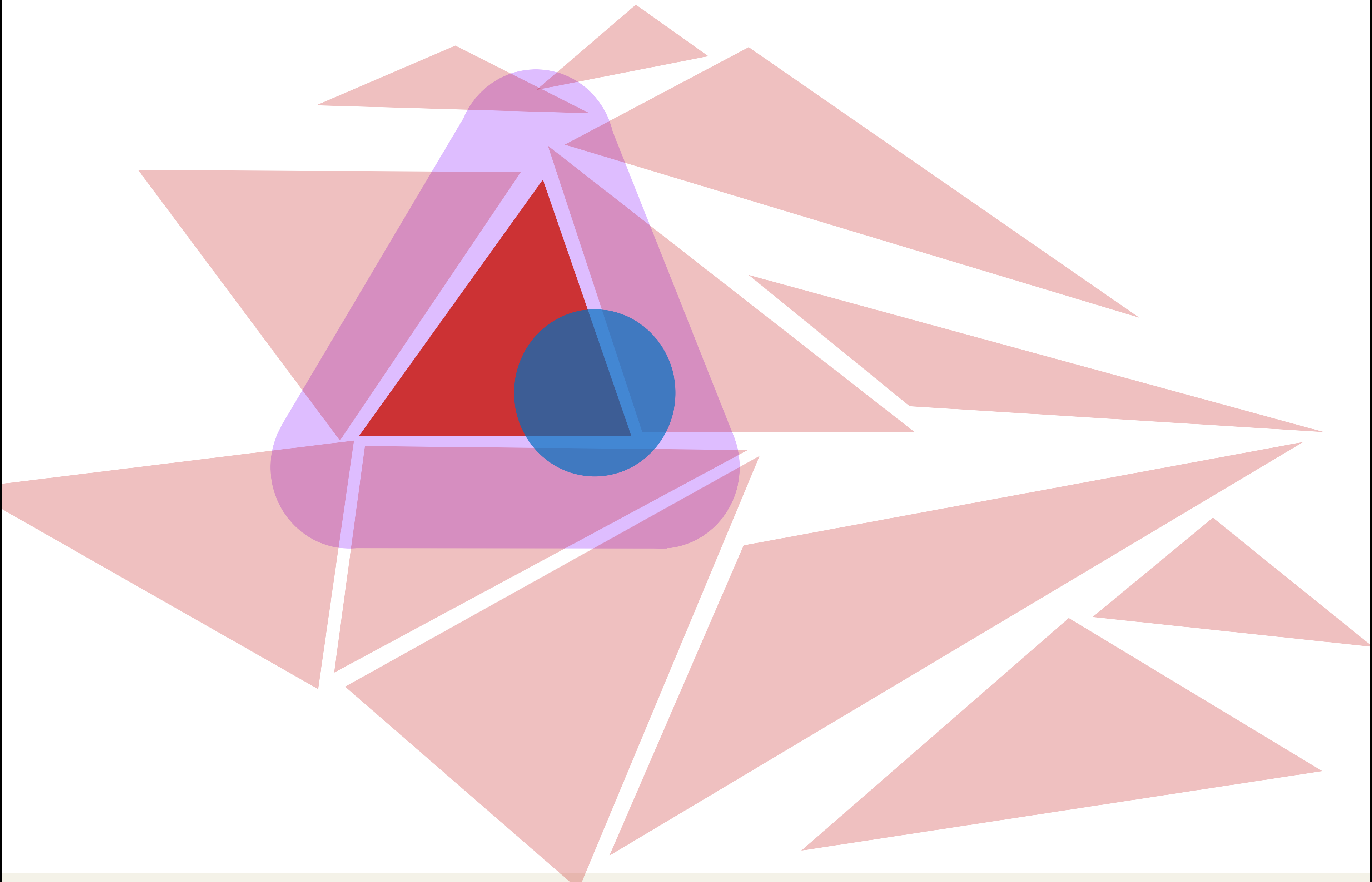
Our Geometric Predicate



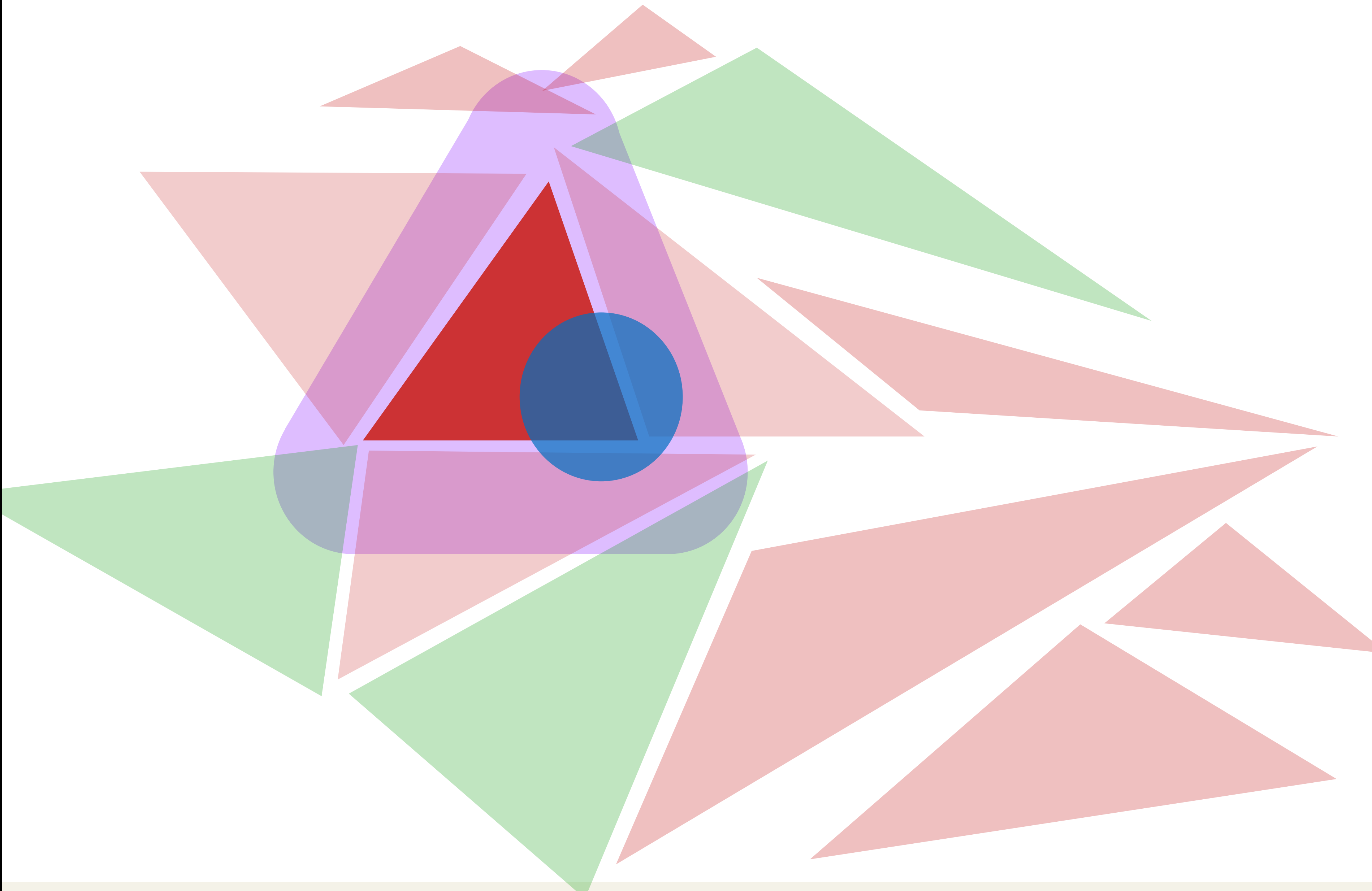
Our Geometric Predicate



Our Geometric Predicate



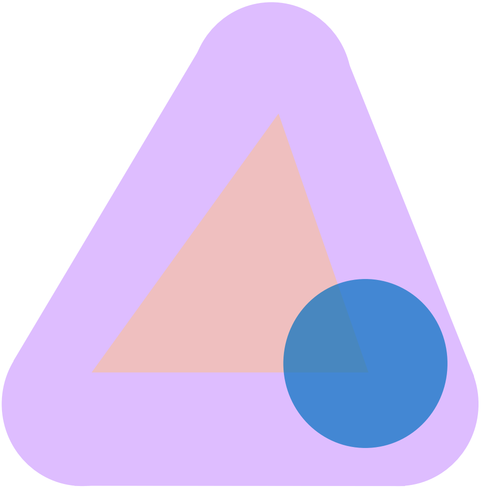
Our Geometric Predicate



Lemma

- Let A be a k-free set of triangles and let t $\notin A$ be an arbitrary triangle that is not included in A. Then t intersects at most a constant number of larger triangles $t_j \in A$.

  More precisely, the number of intersections between t and larger triangles $t_j \in A$ is at most $3k$.



Theorem

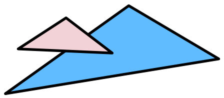
- Let A and B be two k-free sets of triangles. Then the total number of colliding triangles of A and B is in $O(n)$, where n is the number of triangles in A and B. More precisely, the number of intersections is at most $3nk$.

Theorem

- Let A and B be two k-free sets of triangles. Then the total number of colliding triangles of A and B is in $O(n)$, where n is the number of triangles in A and B. More precisely, the number of intersections is at most $3nk$.
- Proof:

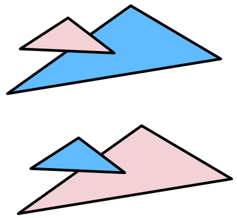
Theorem

- Let A and B be two k-free sets of triangles. Then the total number of colliding triangles of A and B is in $O(n)$, where n is the number of triangles in A and B. More precisely, the number of intersections is at most $3nk$.
- Proof:
  - Test small triangles $t_i \in A$ against larger triangles in B



Theorem

- Let A and B be two k-free sets of triangles. Then the total number of colliding triangles of A and B is in $O(n)$, where n is the number of triangles in A and B. More precisely, the number of intersections is at most $3nk$.
- Proof:
  - Test small triangles $t_i \in A$ against larger triangles in B
  - Test small triangles $t_j \in A$ against larger triangles in A



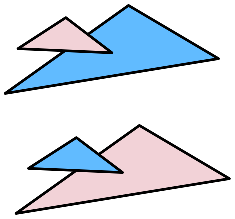
Theorem

- Let A and B be two k-free sets of triangles. Then the total number of colliding triangles of A and B is in $O(n)$, where n is the number of triangles in A and B. More precisely, the number of intersections is at most $3nk$.
- Proof:
  - Test small triangles $t_i \in A$ against larger triangles in B
  - Test small triangles $t_j \in A$ against larger triangles in A

  $\Rightarrow \leq 3k$ intersections for each $t_i \in A$ and $t_j \in B$

  $\Rightarrow$ Total number of intersection $\leq 3kn$

  $\Rightarrow O(n)$ intersections



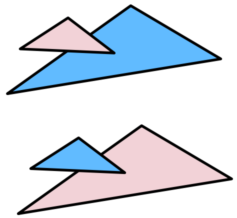
Theorem

- Let A and B be two k-free sets of triangles. Then the total number of colliding triangles of A and B is in $O(n)$, where n is the number of triangles in A and B. More precisely, the number of intersections is at most $3nk$.
- Proof:
  - Test small triangles $t_i \in A$ against larger triangles in B
  - Test small triangles $t_j \in A$ against larger triangles in A

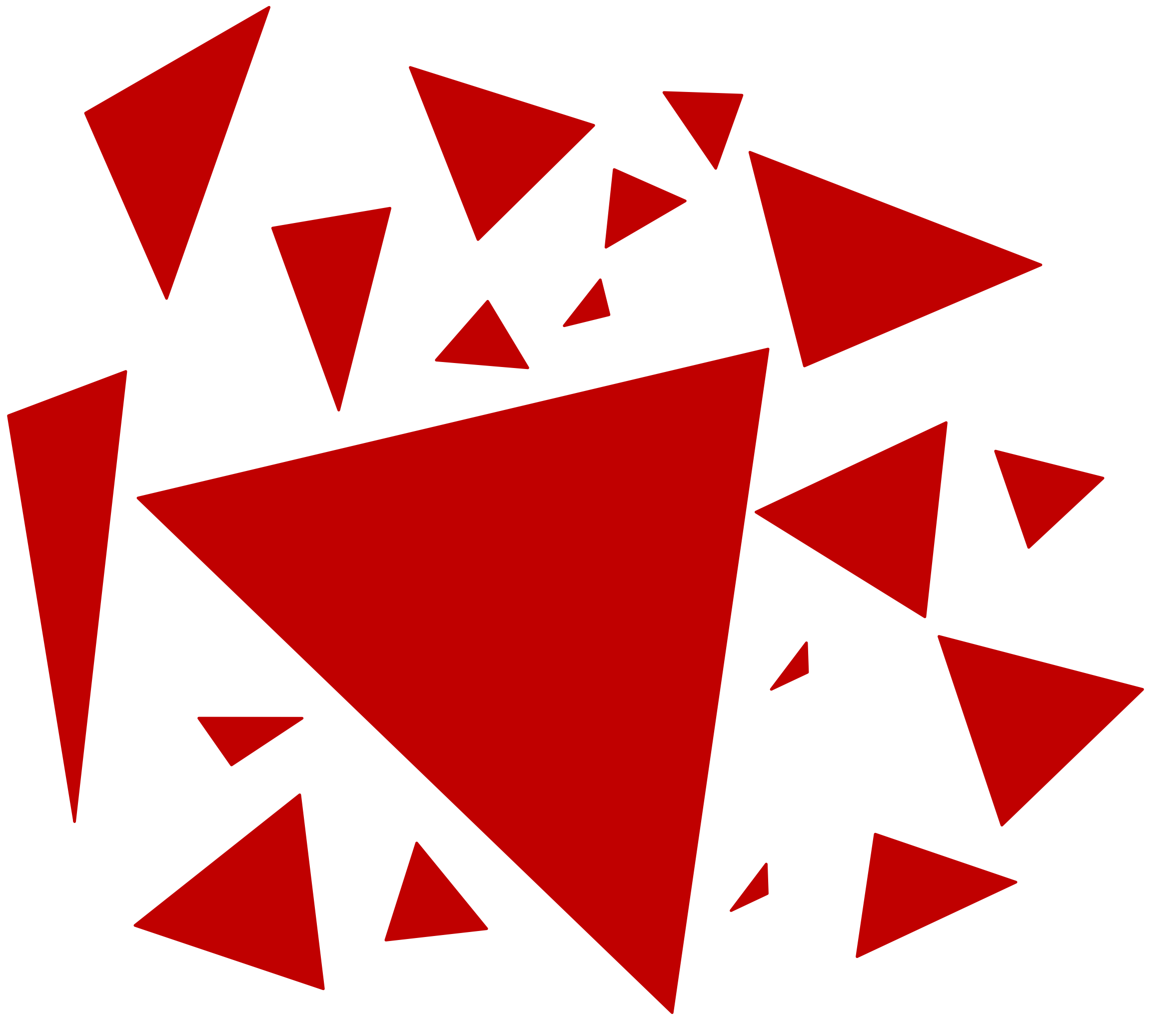
  $\Rightarrow \leq 3k$ intersections for each $t_i \in A$ and $t_j \in B$

  $\Rightarrow$ Total number of intersection $\leq 3kn$

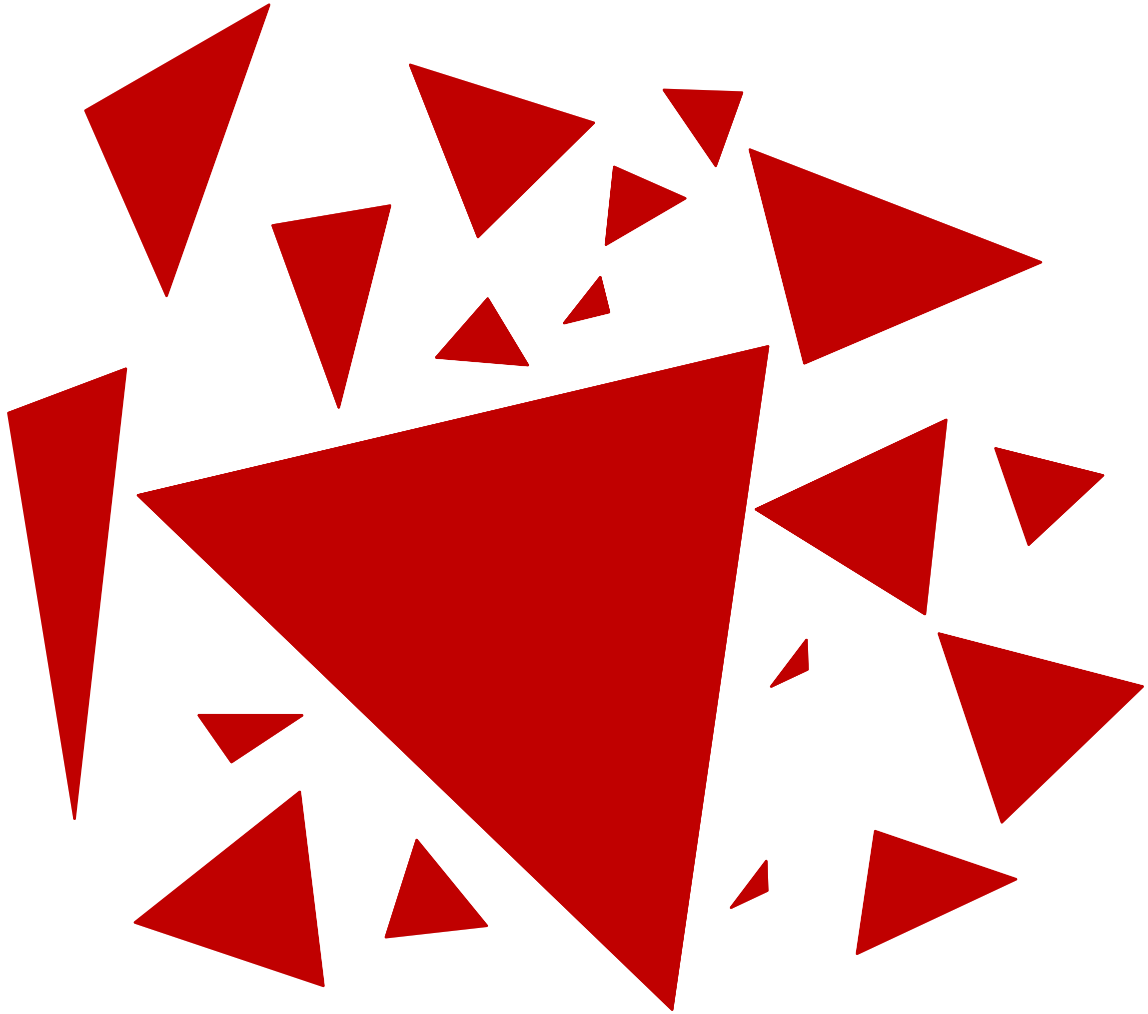
  $\Rightarrow O(n)$ intersections

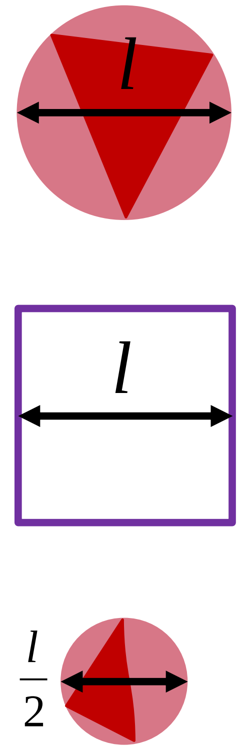


Our Algorithm

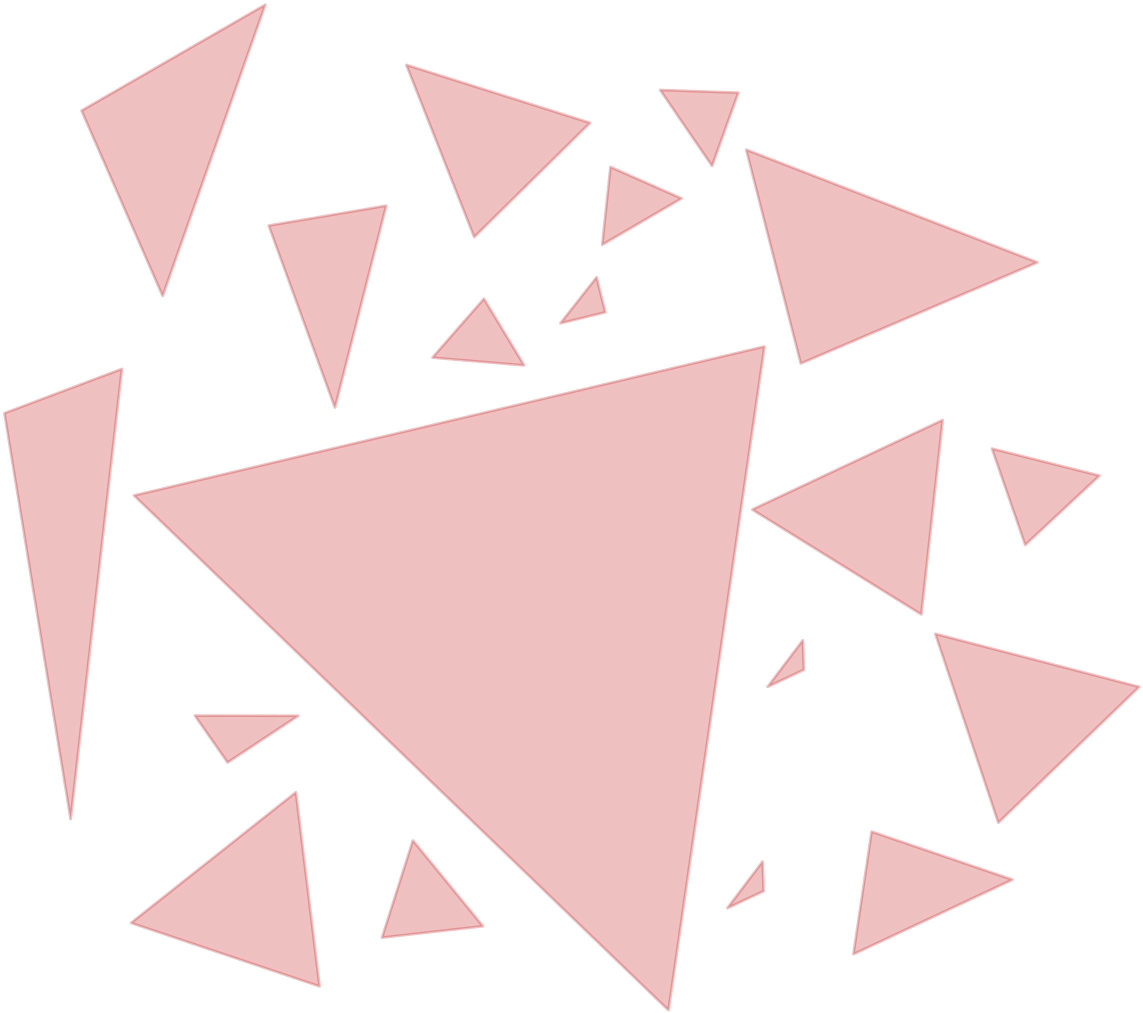


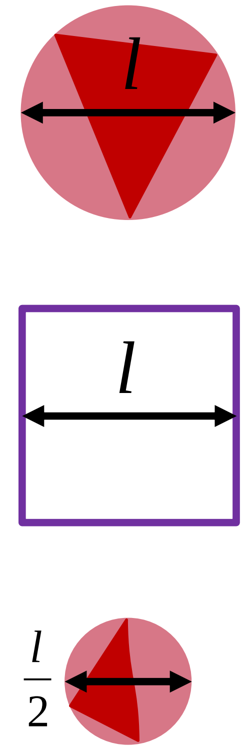
Our Algorithm



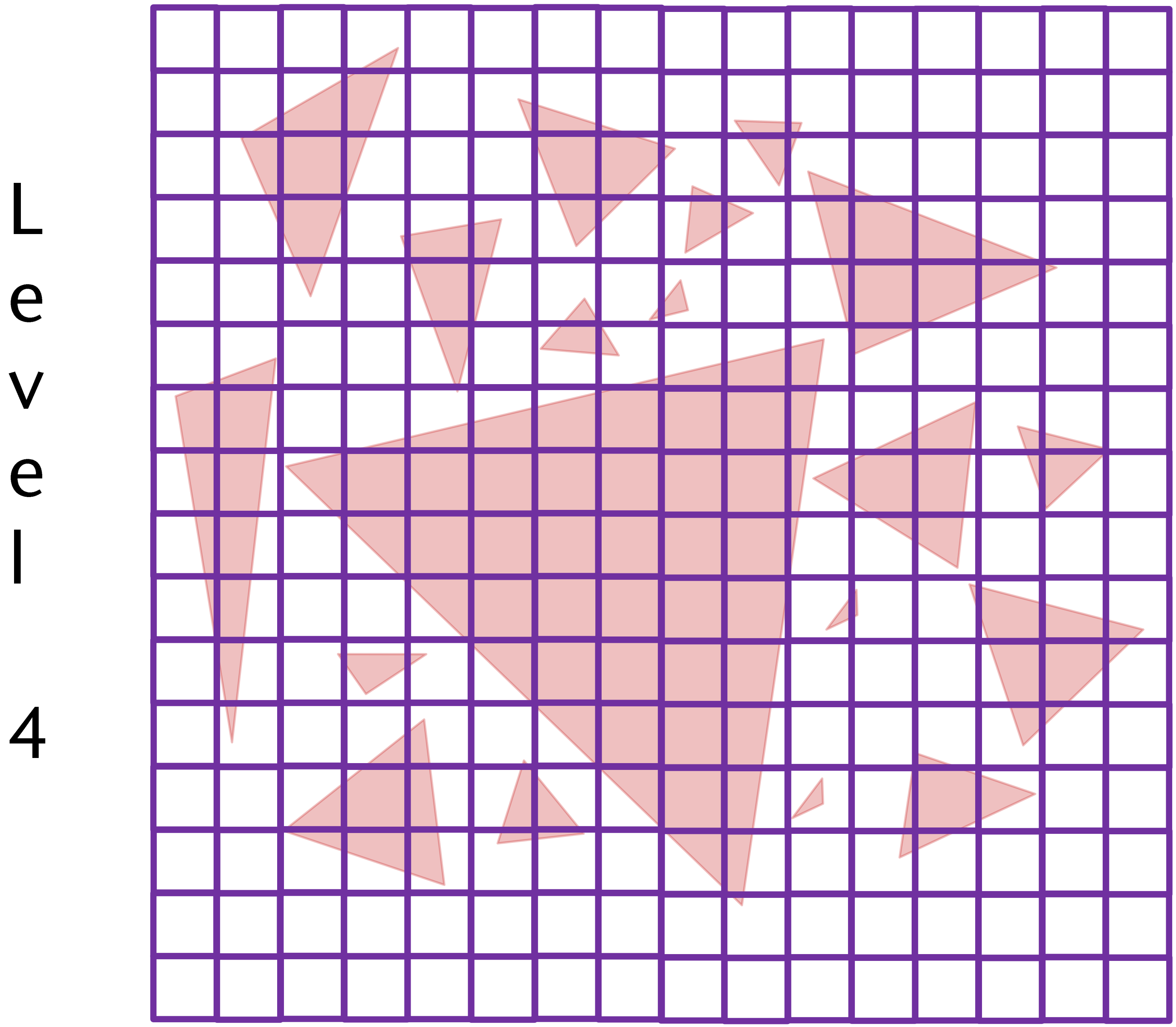


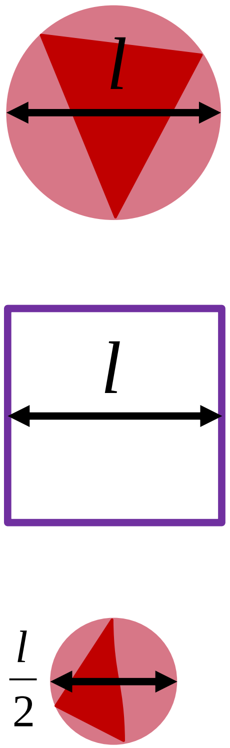
Our Algorithm



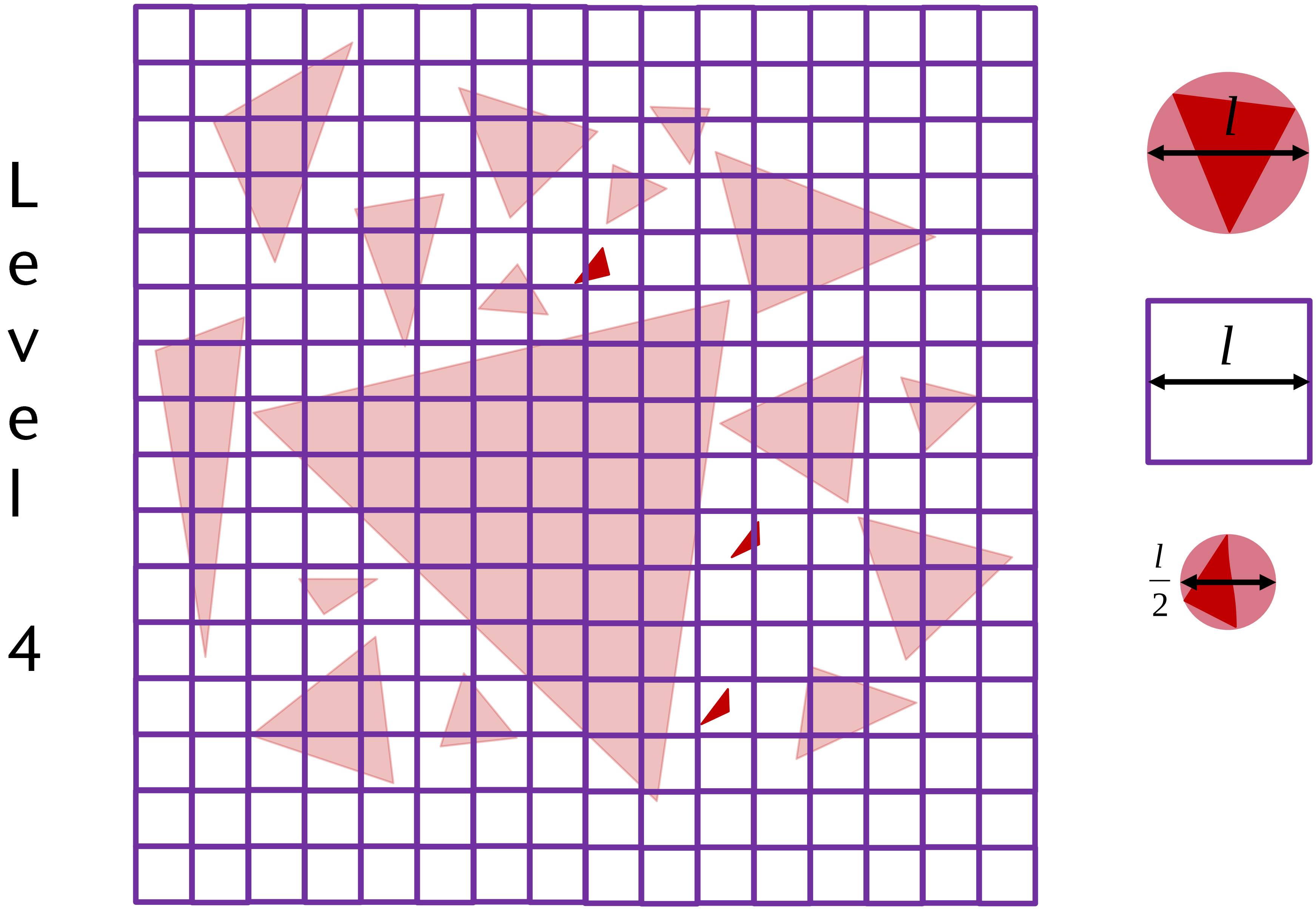


Our Algorithm

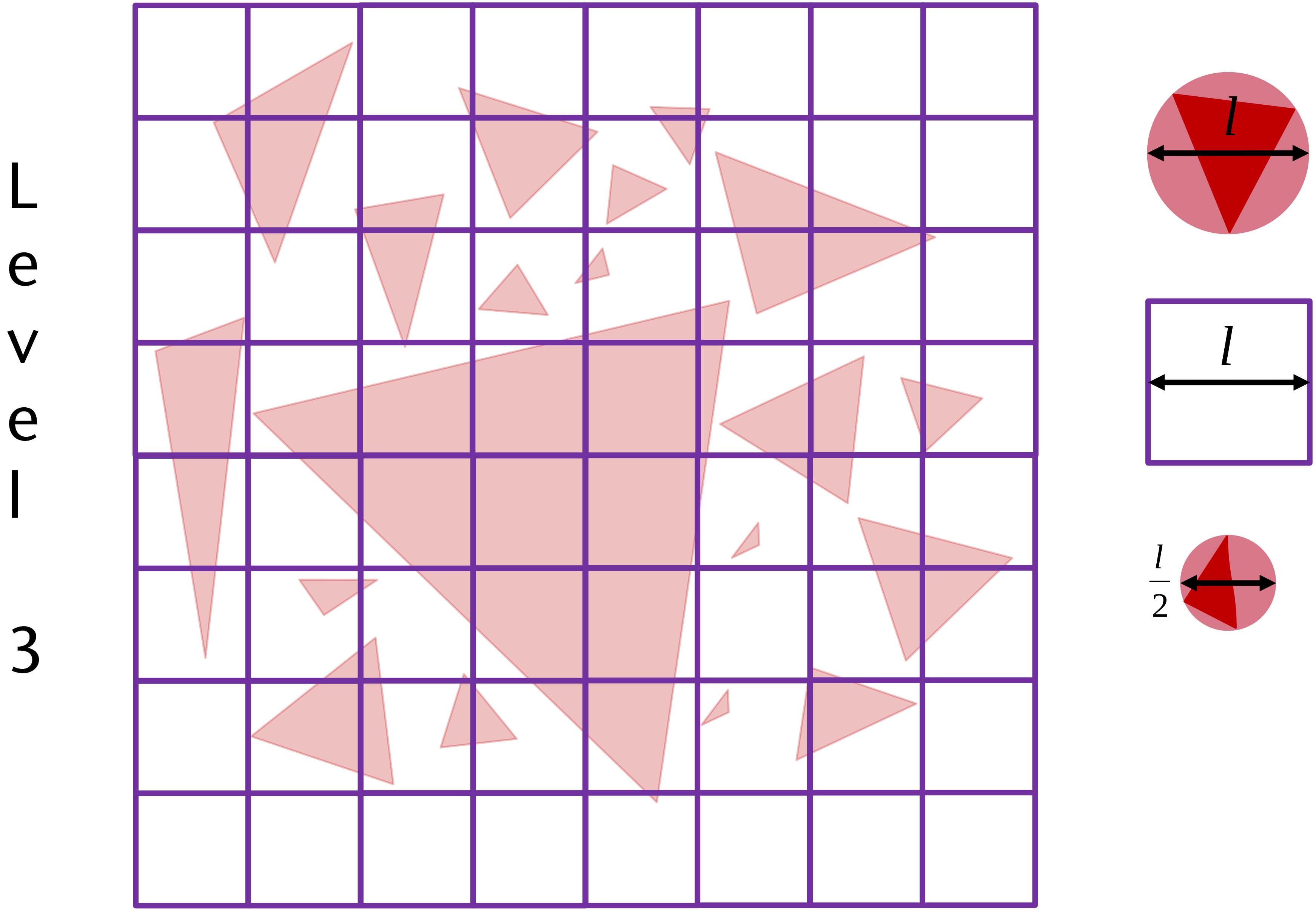




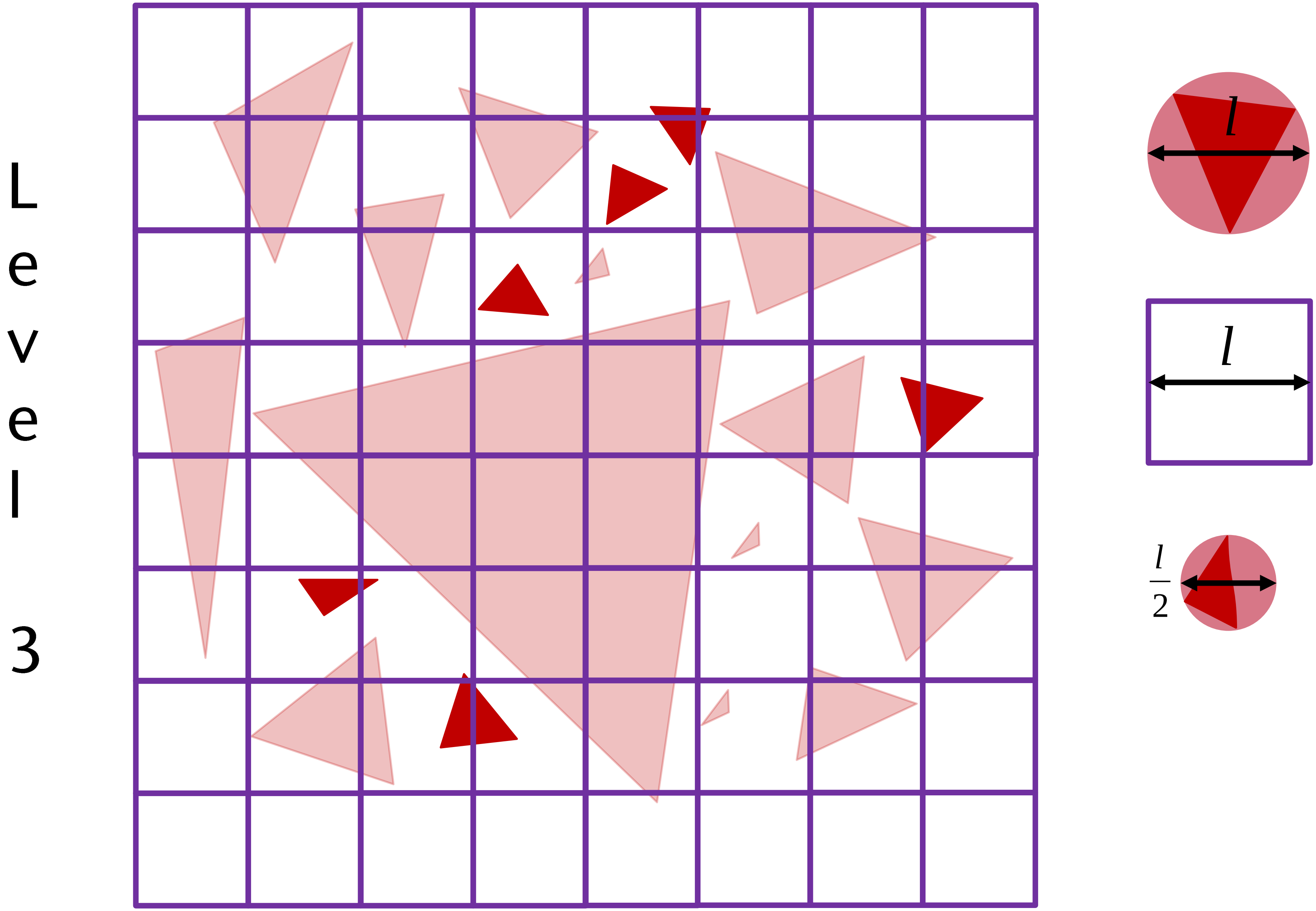
Our Algorithm



Our Algorithm

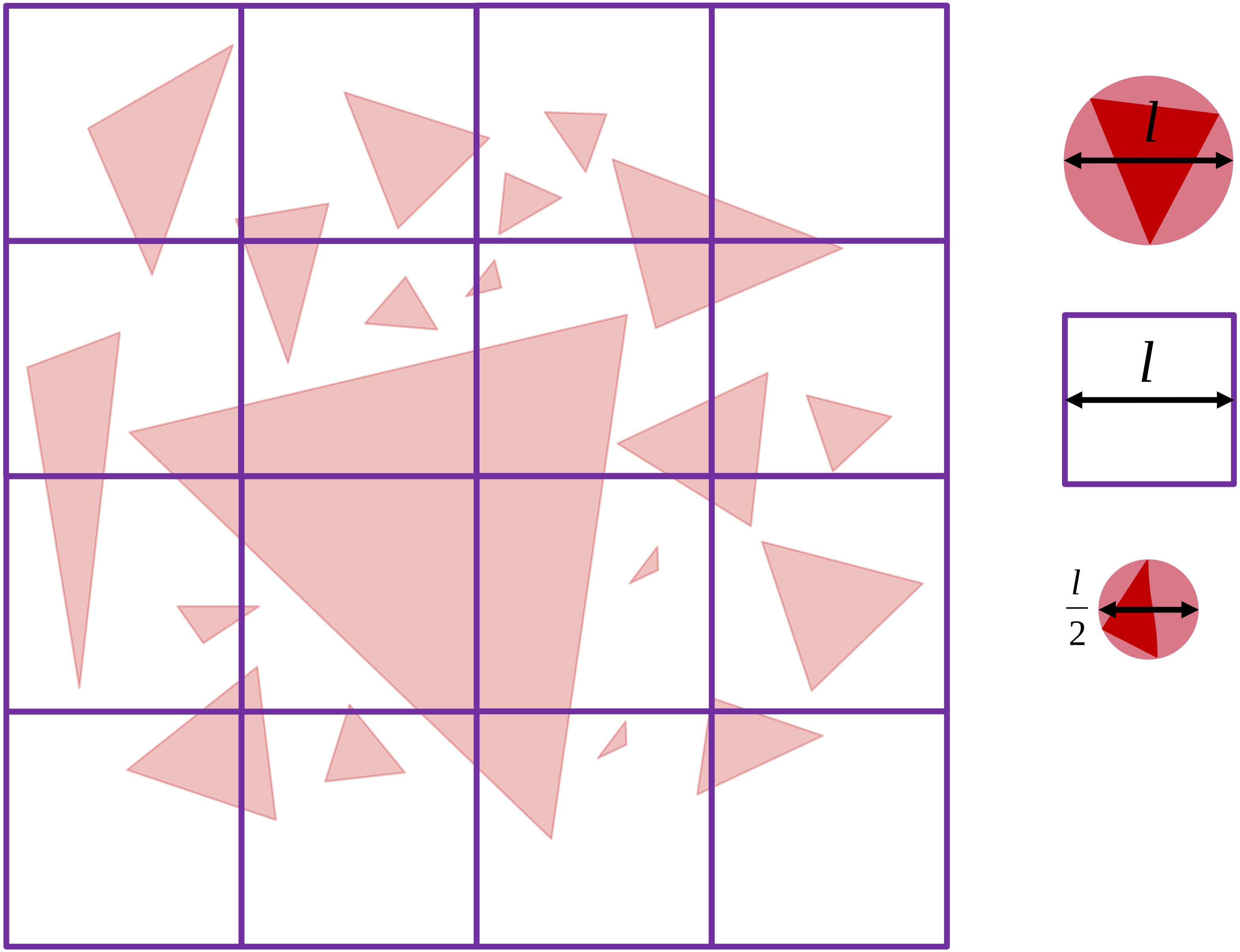


Our Algorithm



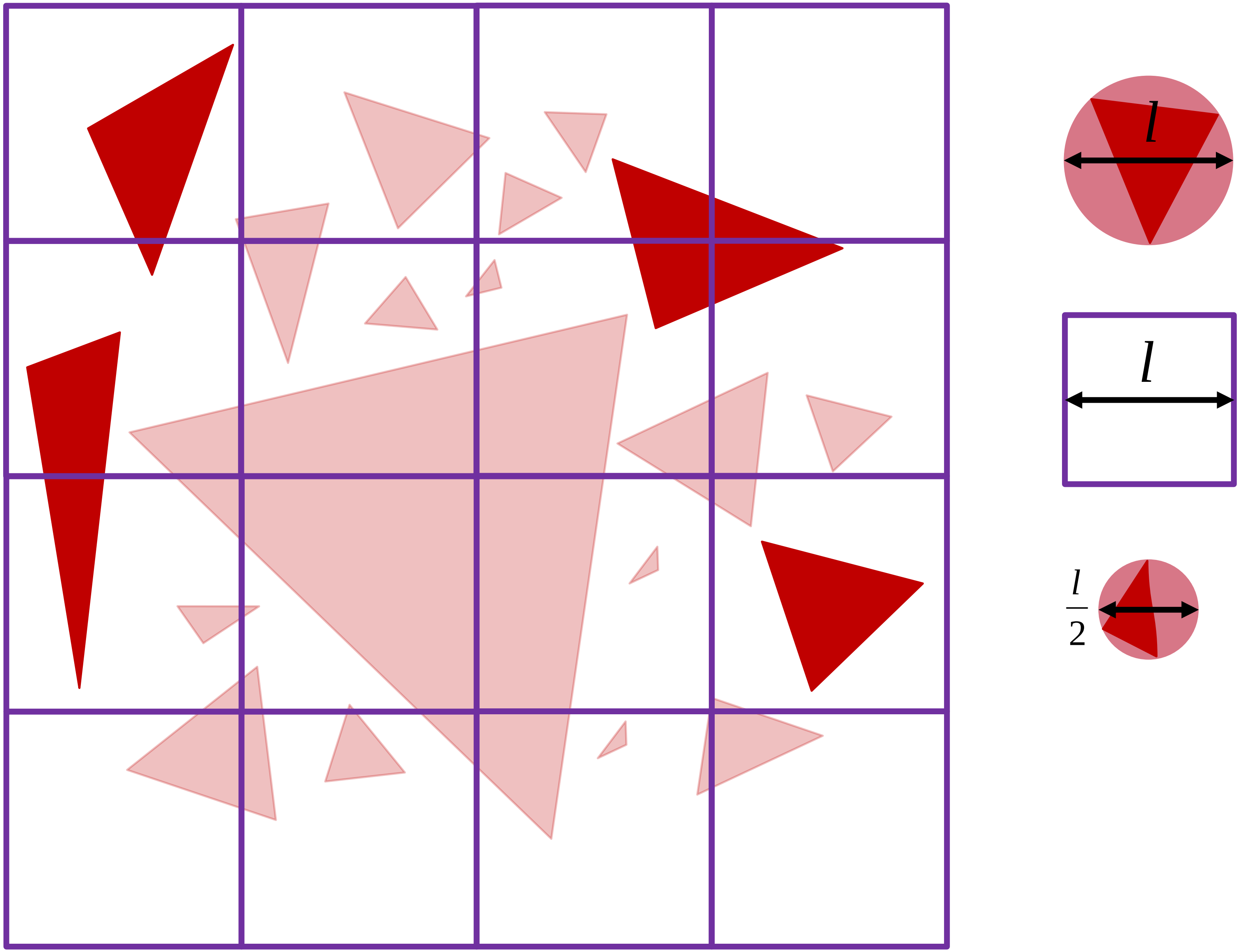
Our Algorithm

Level 2



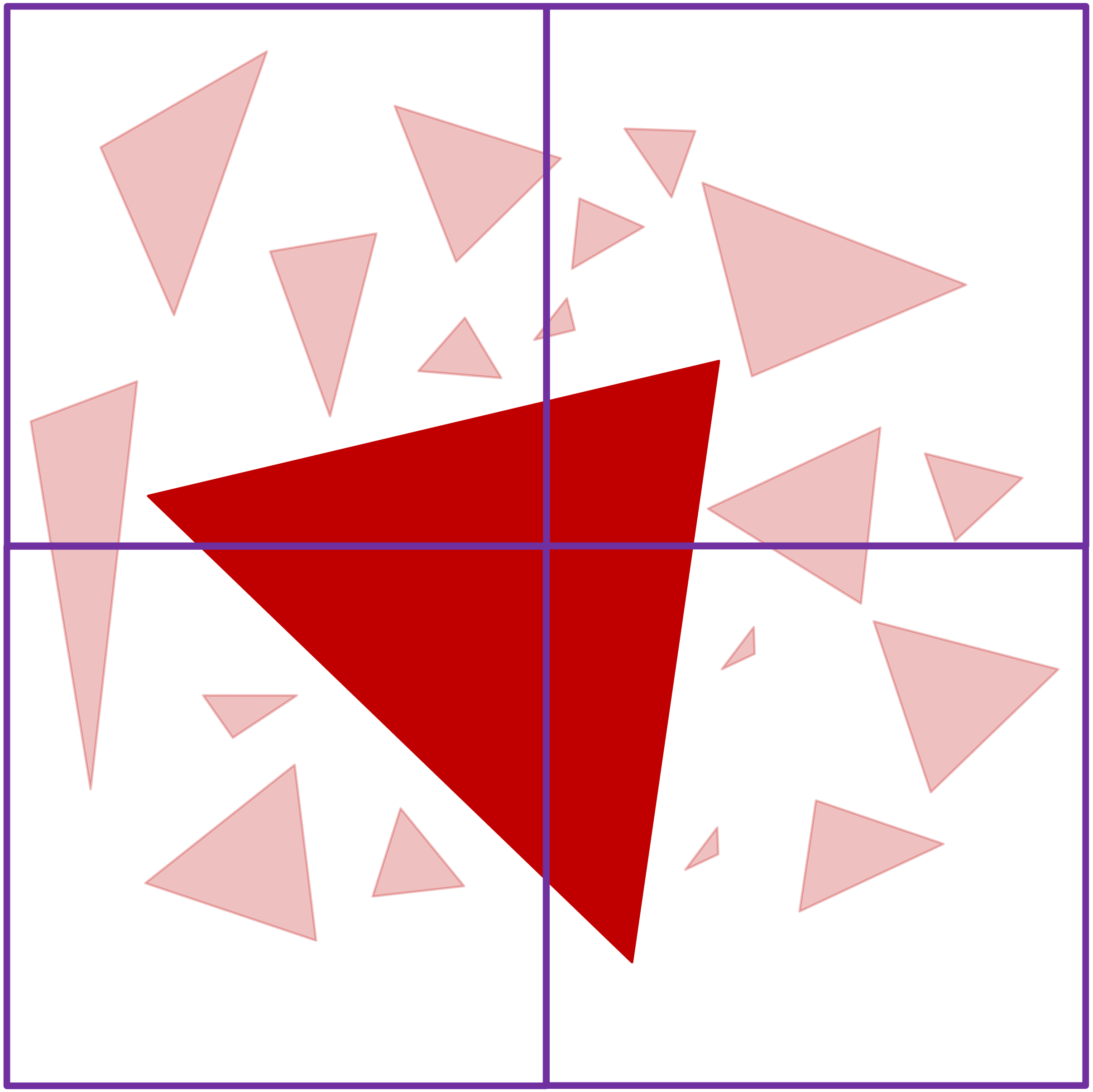
Our Algorithm

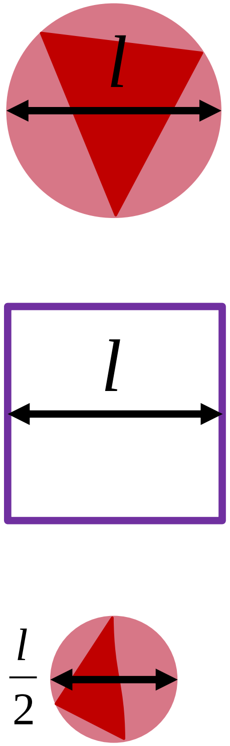
Level 2



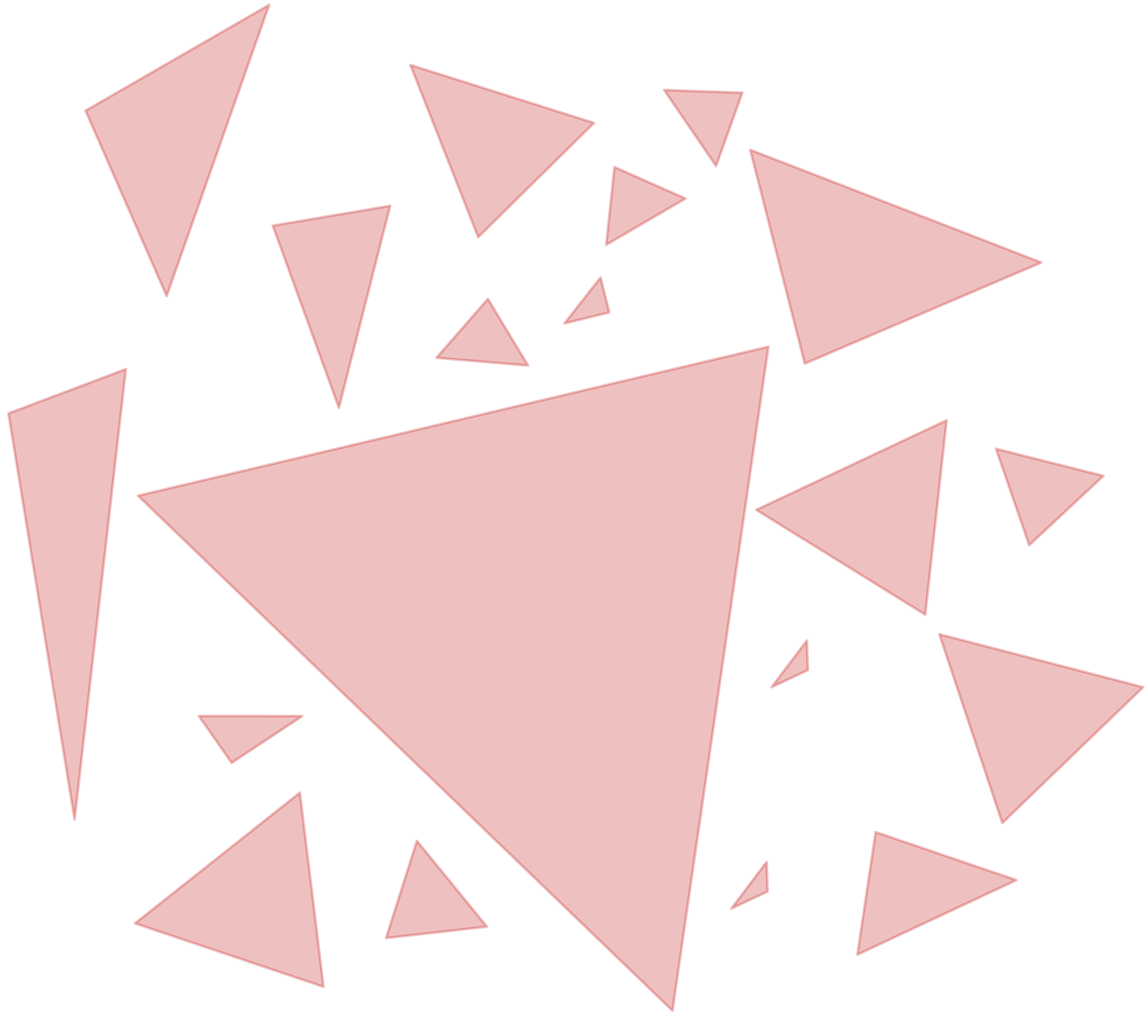
Our Algorithm

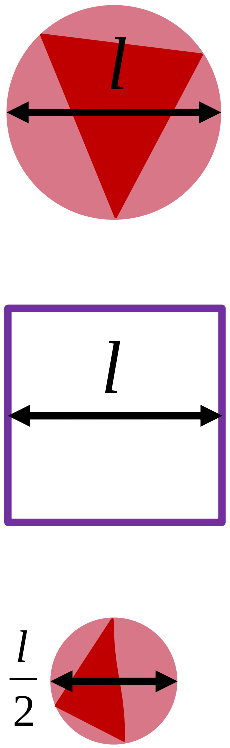
Level 1



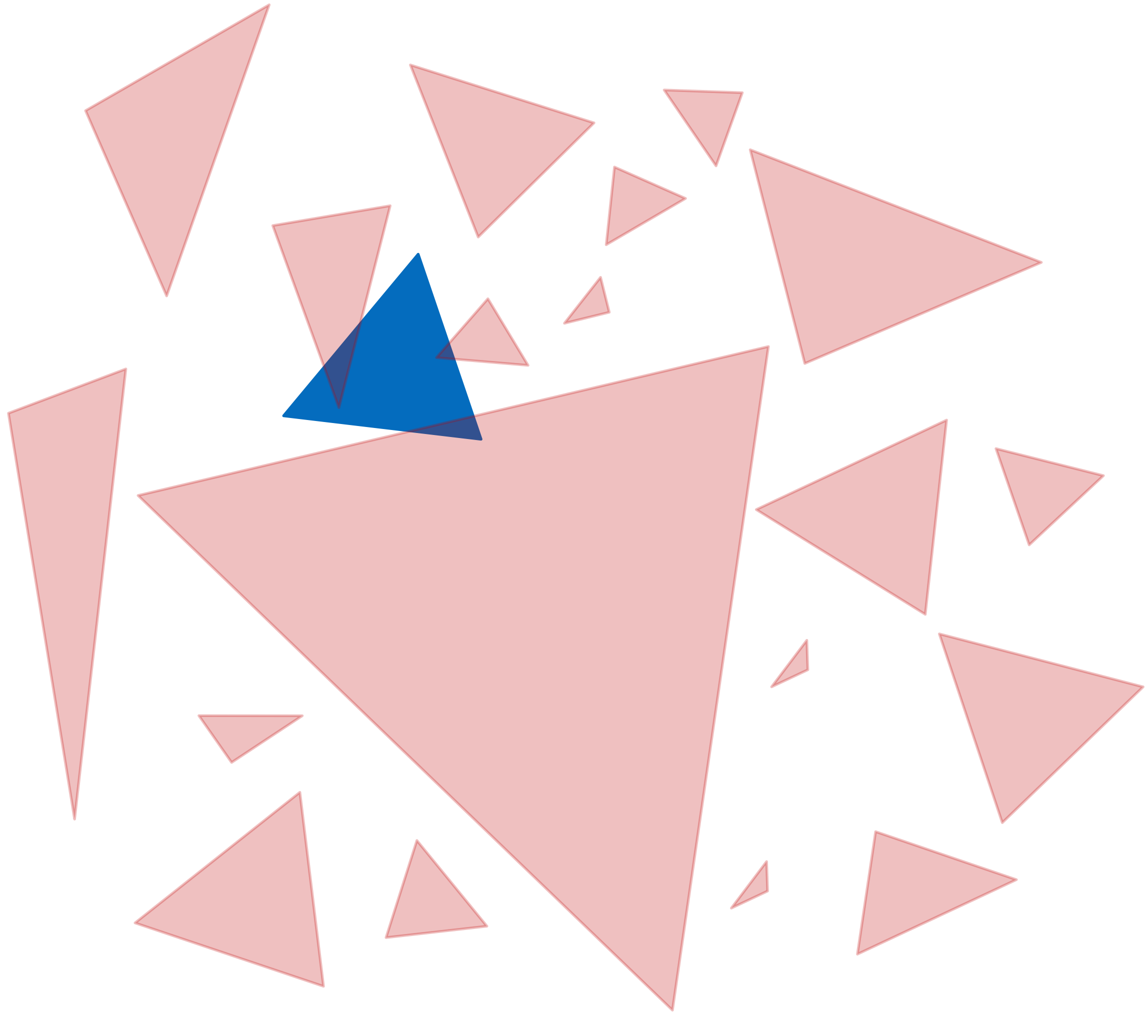


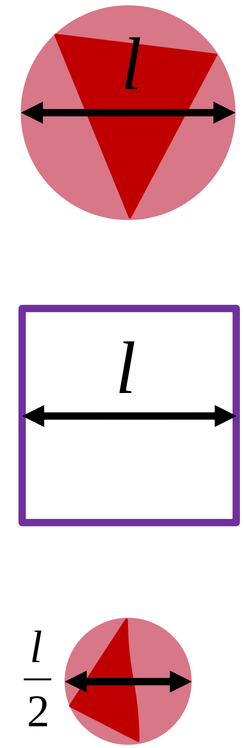
Our Algorithm



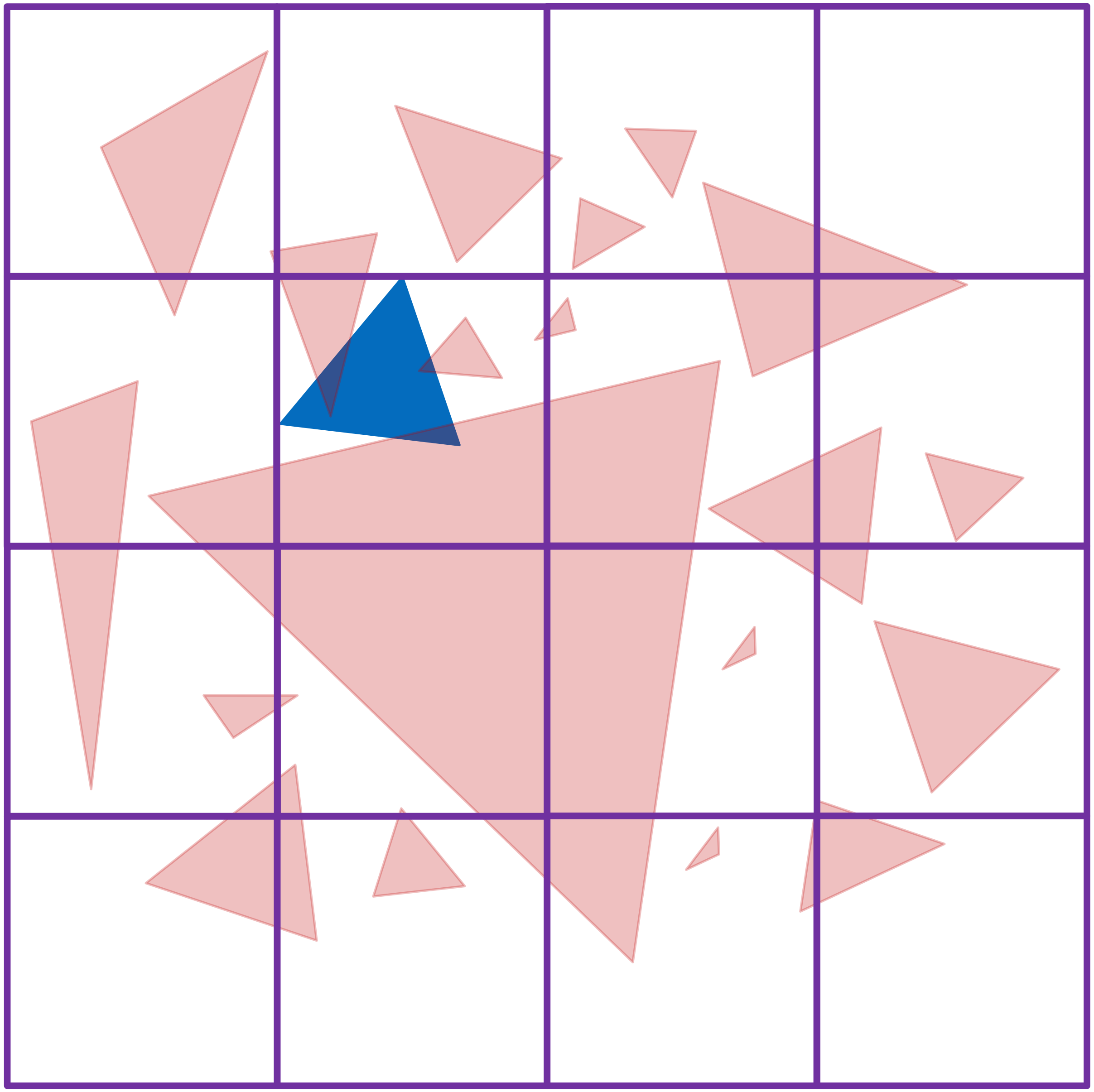


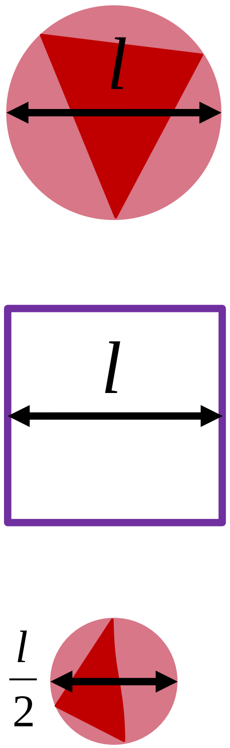
Our Algorithm



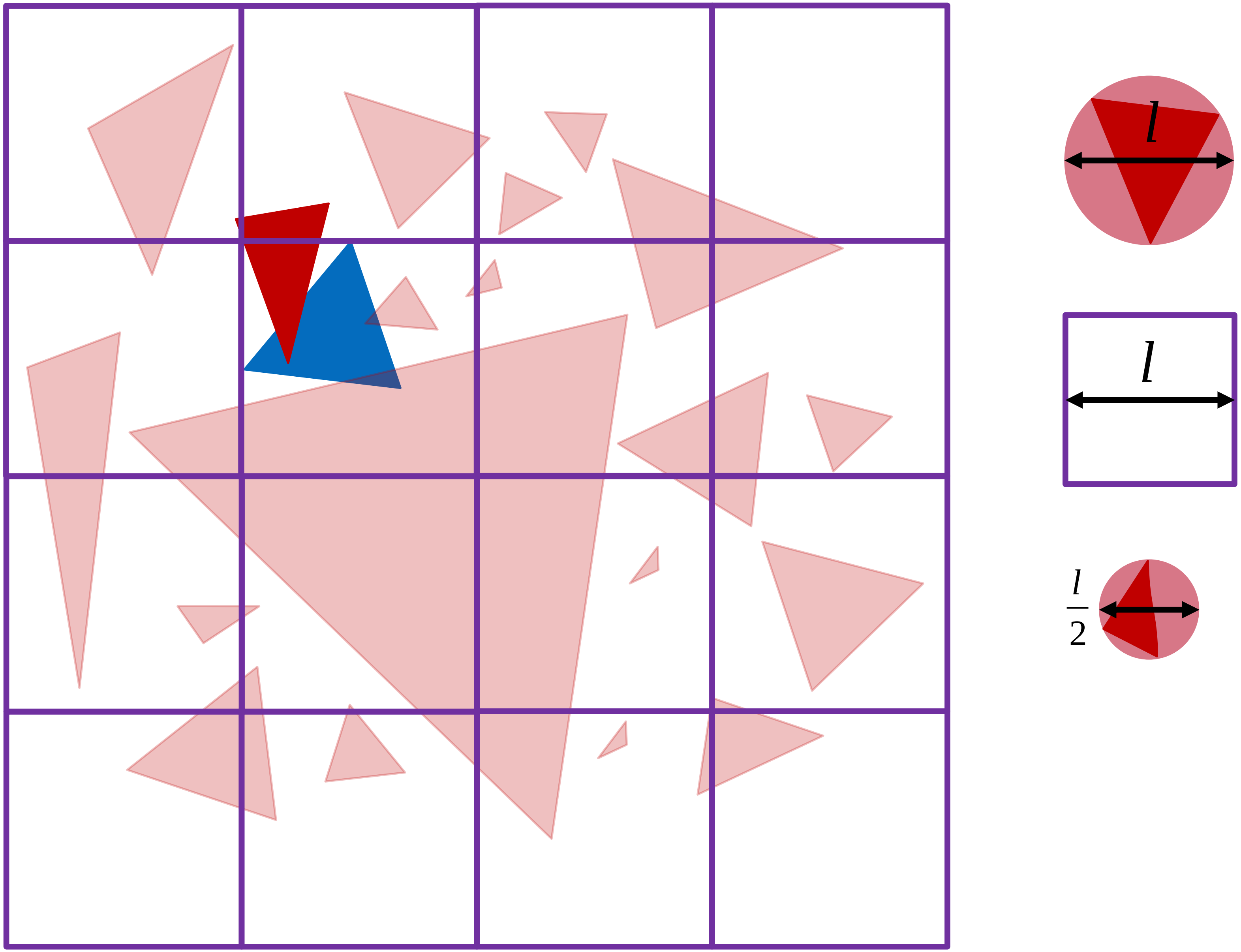


Our Algorithm

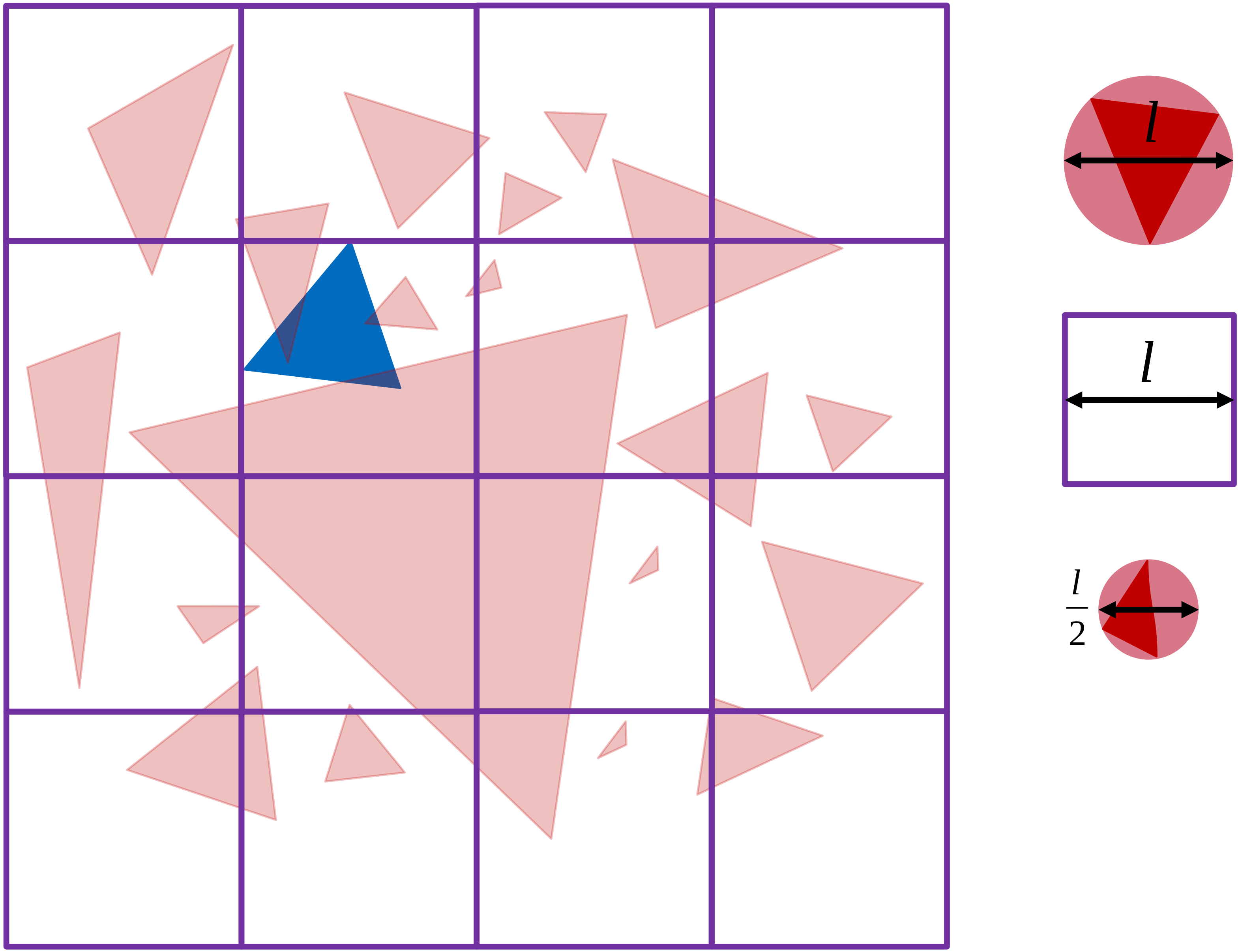




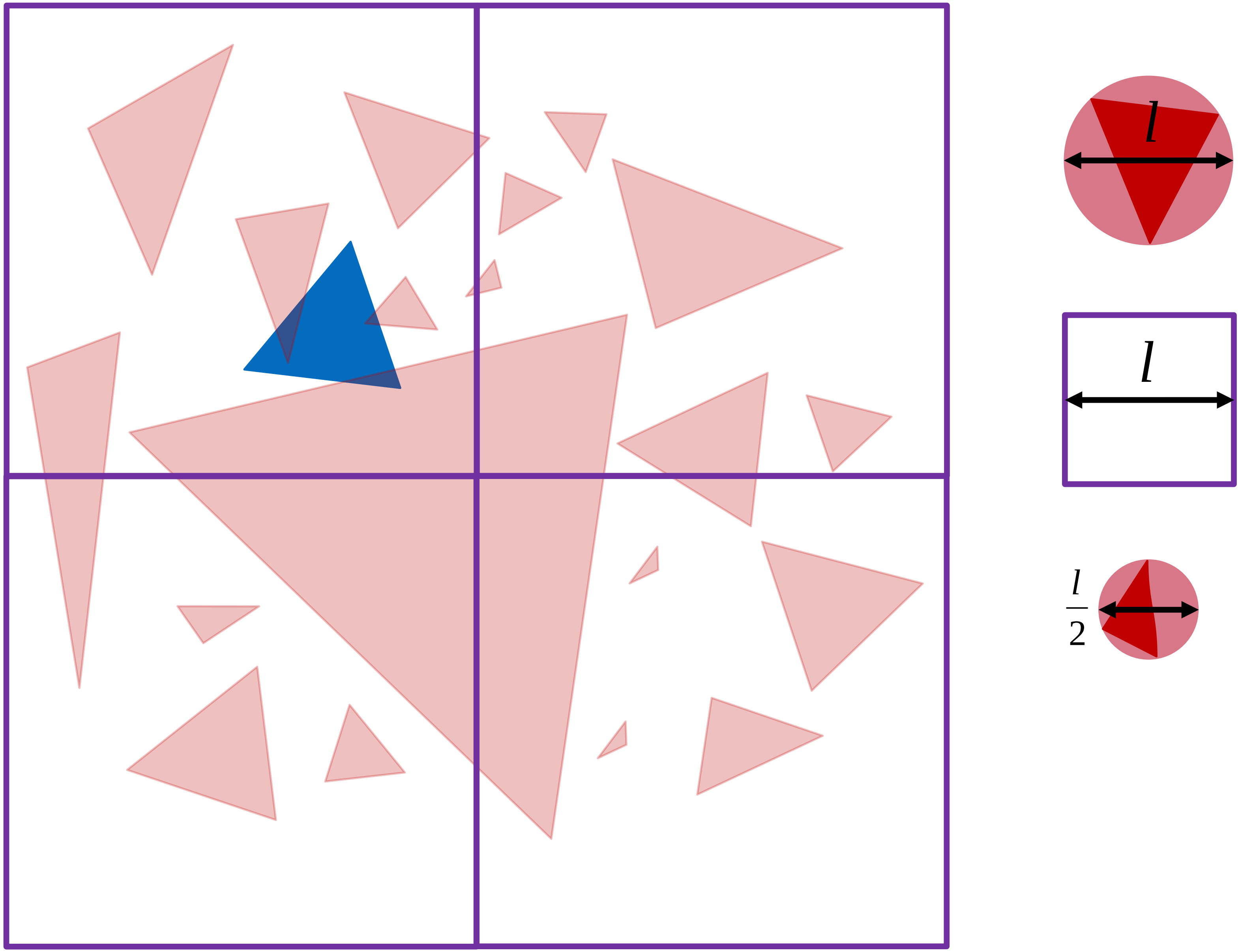
Our Algorithm



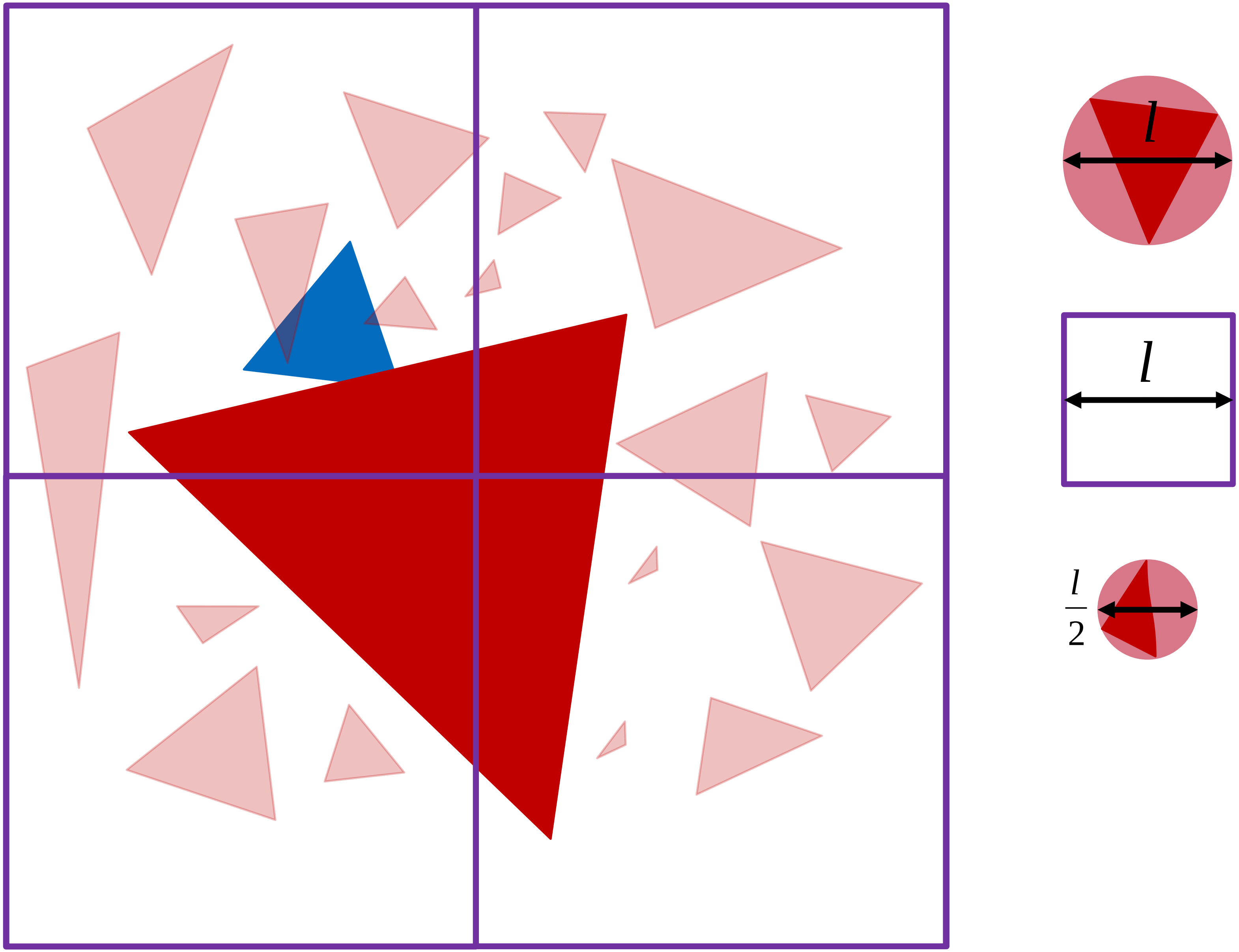
Our Algorithm



Our Algorithm



Our Algorithm



Analysis

Analysis

```
For each triangle t ∈ B:
```

Analysis

```
For each triangle t ∈ B:
    Compute hierarchy level l
```

Analysis

```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:
```

Analysis

```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by t
```

Analysis

```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by t
            For all triangles t_k ∈ c_j
```

Analysis

```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by t
            For all triangles t_k ∈ c_j
                Compute intersection for t and t_k
```

Analysis

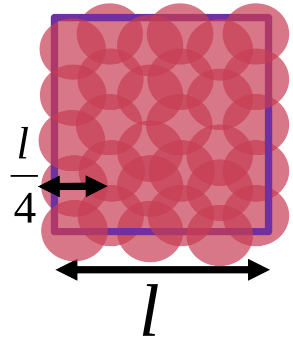
```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by t
            For all triangles t_k ∈ c_j
                Compute intersection for t and t_k
```

$\Rightarrow O(1)$

Analysis

For each triangle $t \in B$:
    Compute hierarchy level l
    For all levels l_i, $l \le l_i \le l_{max}$:
        For all cells c_j in level l_i overlapped by t
            For all triangles $t_k \in c_j$
                Compute intersection for t and t_k

$\Rightarrow O(1)$

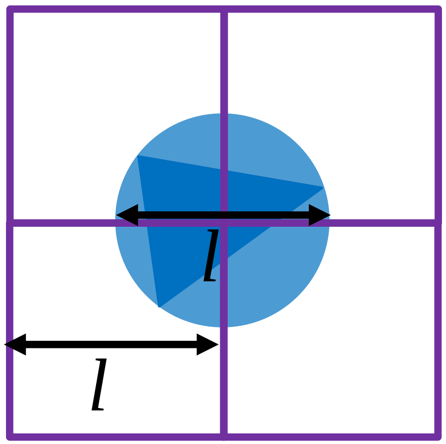


Analysis

```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by t
            For all triangles t_k ∈ c_j                      ⇒ O(1)
                Compute intersection for t and t_k           ⇒ O(1)
```

Analysis

For each triangle $t \in B$:
    Compute hierarchy level l
    For all levels l_i, $l \le l_i \le l_{max}$:
        For all cells c_j in level l_i overlapped by t
            For all triangles $t_k \in c_j$ $\Rightarrow O(1)$
                Compute intersection for t and t_k $\Rightarrow O(1)$

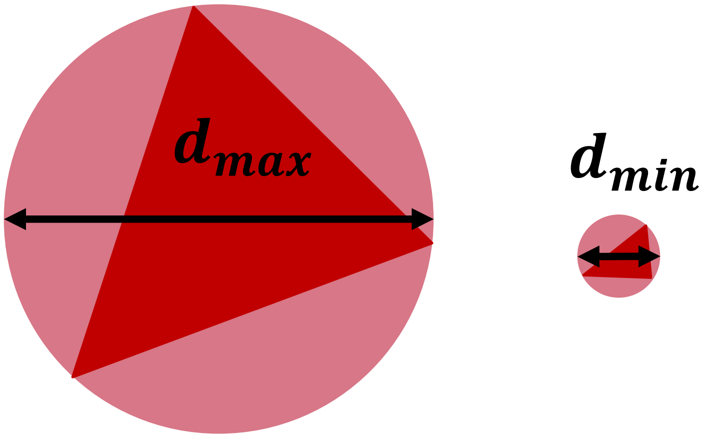


Analysis

```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by t          ⇒ O(1)
            For all triangles t_k ∈ c_j                         ⇒ O(1)
                Compute intersection for t and t_k              ⇒ O(1)
```

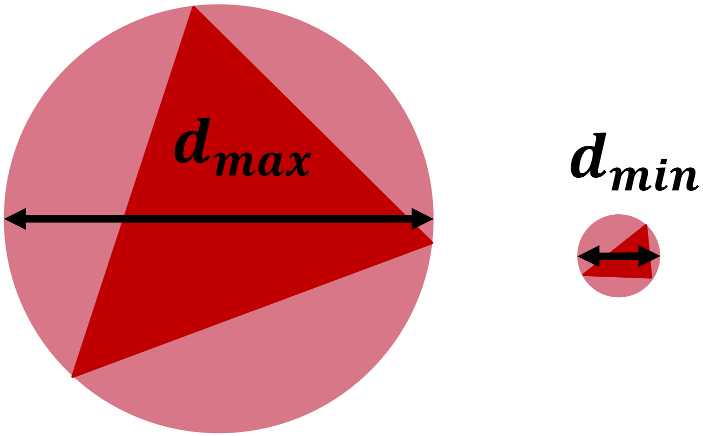
Analysis

```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by t          ⇒ O(1)
            For all triangles t_k ∈ c_j                          ⇒ O(1)
                Compute intersection for t and t_k               ⇒ O(1)
```



Analysis

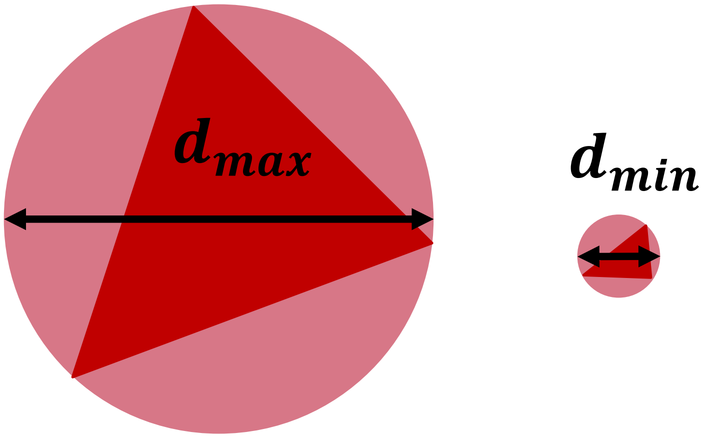
```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by t        ⇒ O(1)
            For all triangles t_k ∈ c_j                      ⇒ O(1)
                Compute intersection for t and t_k           ⇒ O(1)
```



$$\Rightarrow O\left(\log \frac{d_{\max}}{d_{\min}}\right)$$

Analysis

```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:                         ⇒ O(1)
        For all cells c_j in level l_i overlapped by t          ⇒ O(1)
            For all triangles t_k ∈ c_j                          ⇒ O(1)
                Compute intersection for t and t_k               ⇒ O(1)
```



$$\Rightarrow O\left(\log \frac{d_{\max}}{d_{\min}}\right)$$

Analysis

```
For each triangle t ∈ B:
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:
        For all cells c_j in level l_i overlapped by t
            For all triangles t_k ∈ c_j
                Compute intersection for t and t_k
```

- For all levels: $\Rightarrow O(1)$
- For all cells: $\Rightarrow O(1)$
- For all triangles: $\Rightarrow O(1)$
- Compute intersection: $\Rightarrow O(1)$

Analysis

```
For each triangle t ∈ B:                                              ⇒ O(n)
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:                               ⇒ O(1)
        For all cells c_j in level l_i overlapped by t                ⇒ O(1)
            For all triangles t_k ∈ c_j                               ⇒ O(1)
                Compute intersection for t and t_k                    ⇒ O(1)
```

Analysis

```
For each triangle t ∈ B:                                       ⇒ O(n)
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:                       ⇒ O(1)
        For all cells c_j in level l_i overlapped by t         ⇒ O(1)
            For all triangles t_k ∈ c_j                        ⇒ O(1)
                Compute intersection for t and t_k             ⇒ O(1)
```

$$\text{Total Time:} \quad O(n)$$

Analysis

```
For each triangle t ∈ B:                                   ⇒ O(n)
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:                   ⇒ O(1)
        For all cells c_j in level l_i overlapped by t     ⇒ O(1)
            For all triangles t_k ∈ c_j                    ⇒ O(1)
                Compute intersection for t and t_k         ⇒ O(1)
```

Total Time: $O(n)$

Analysis

```
In Parallel for all triangles t ∈ B:                     ⇒ O(n)
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:                   ⇒ O(1)
        For all cells c_j in level l_i overlapped by t     ⇒ O(1)
            For all triangles t_k ∈ c_j                    ⇒ O(1)
                Compute intersection for t and t_k         ⇒ O(1)
```

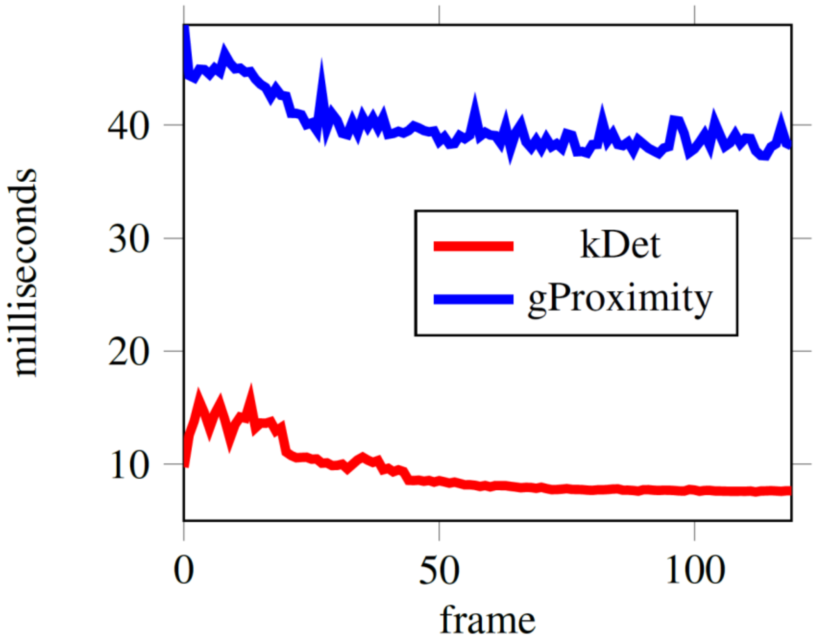
Total Time: $O(n)$

```
In Parallel for all triangles t ∈ B:                         ⇒ O(n)
    Compute hierarchy level l
    For all levels l_i, l ≤ l_i ≤ l_max:                     ⇒ O(1)
        For all cells c_j in level l_i overlapped by t       ⇒ O(1)
            For all triangles t_k ∈ c_j                      ⇒ O(1)
                Compute intersection for t and t_k           ⇒ O(1)
```

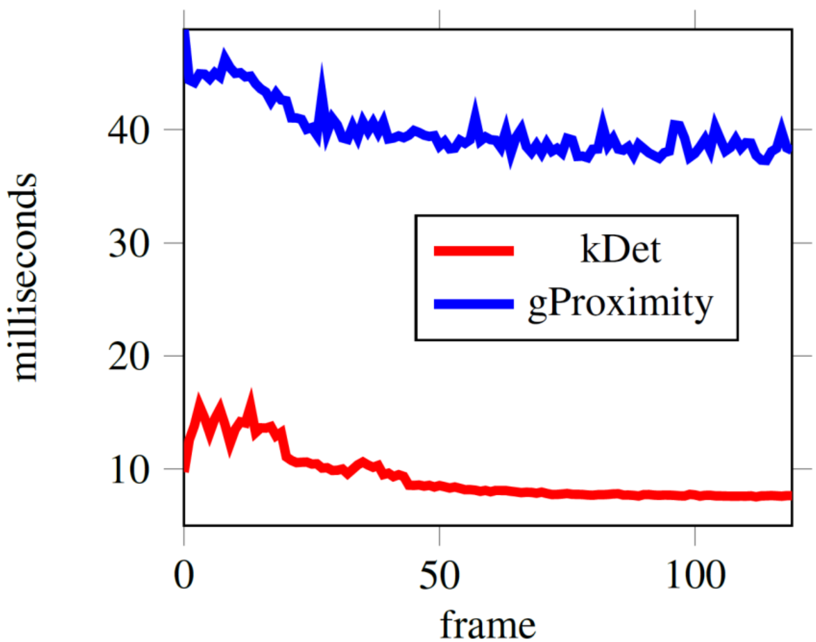
Total Time: $O(n)$

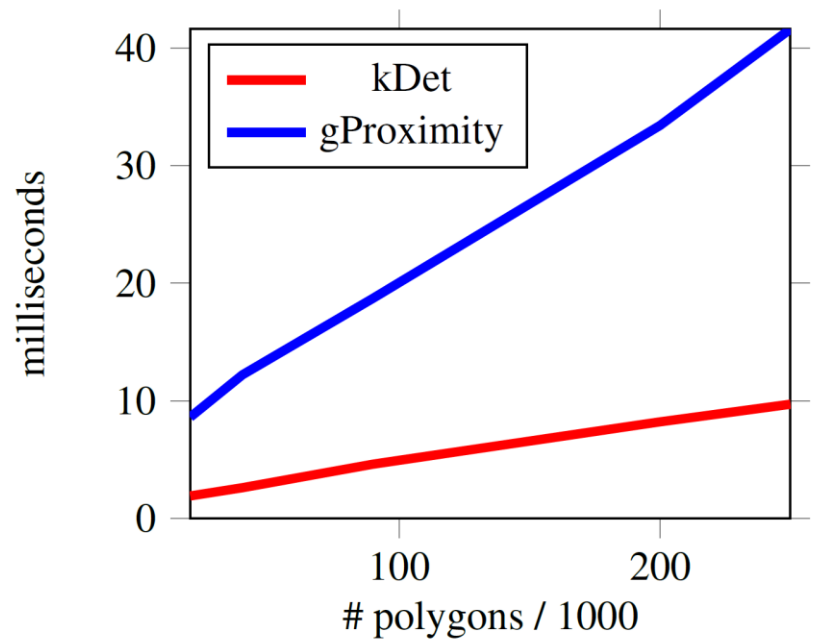
Total Parallel Time: $O(1)$

Results

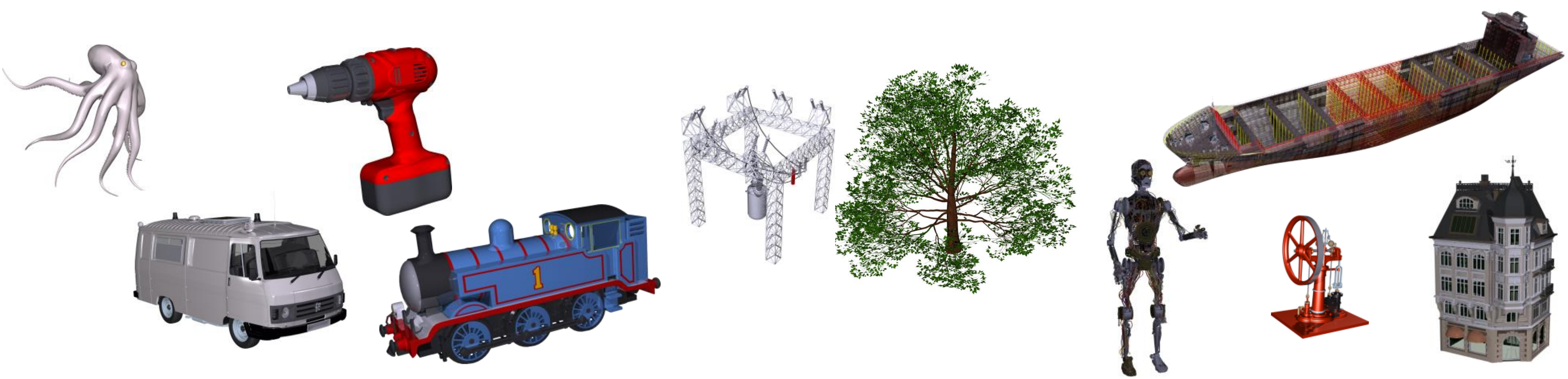


Results

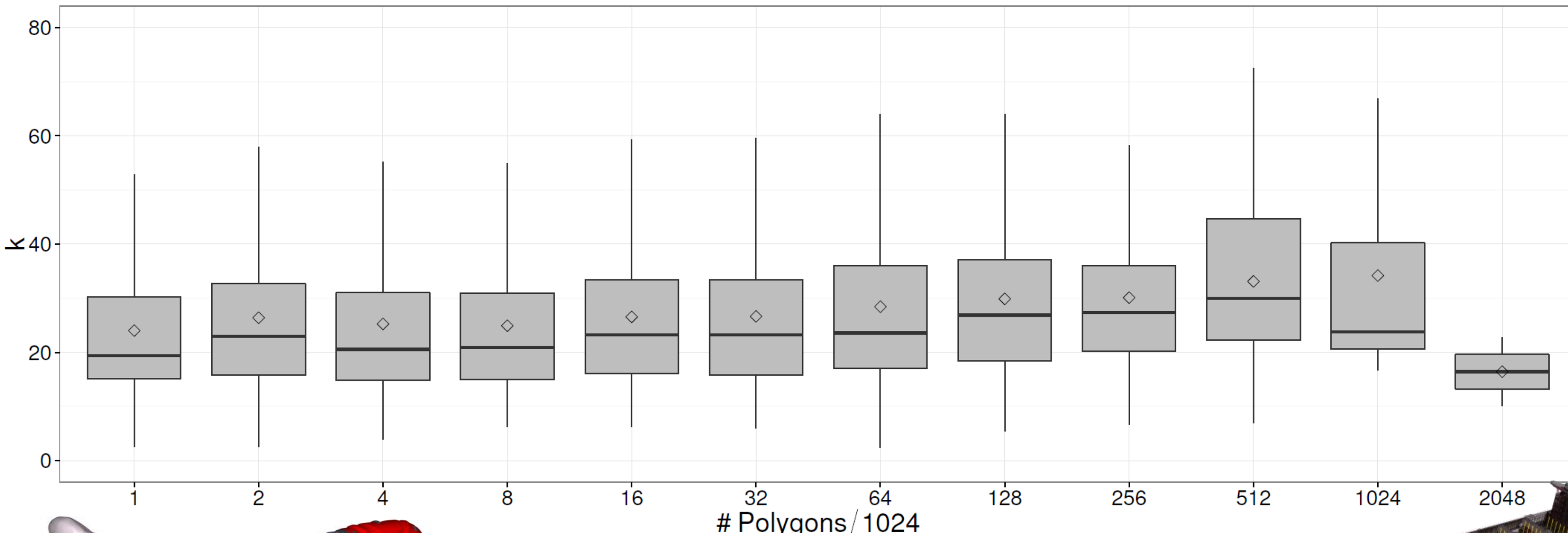

milliseconds
40
30
20
10
0
50
100
frame
kDet
gProximity

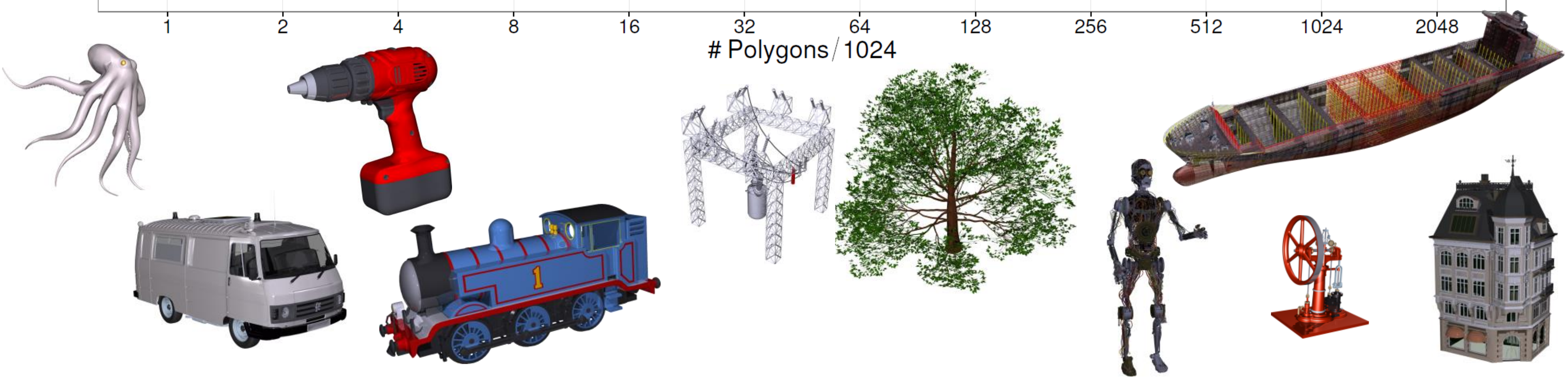

milliseconds
40
30
20
10
0
100
200
polygons / 1000
kDet
gProximity

Results



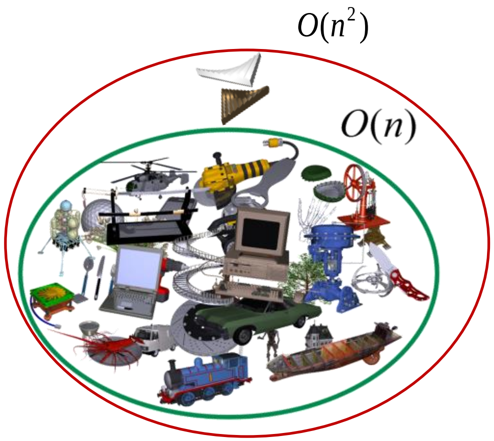
Results



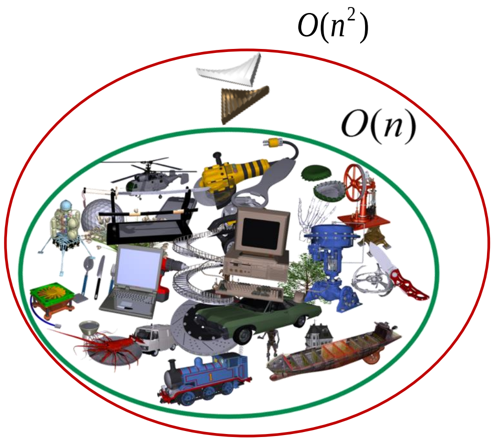


Conclusions

- Novel geometric predicate for polygonal objects defining a class with provable $O(n)$ worst-case intersecting polygon pairs
- Algorithm that realizes linear running-time
  - Parallel running-time: $O(1)$
- <10 msec to check two objects with 250k triangles for collision



Conclusions

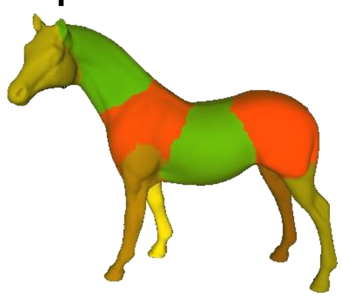


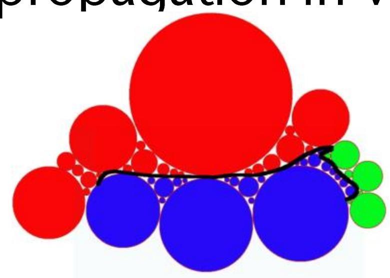
- Novel geometric predicate for polygonal objects defining a class with provable $O(n)$ worst-case intersecting polygon pairs
- Algorithm that realizes linear running-time
  - Parallel running-time: $O(1)$
- <10 msec to check two objects with 250k triangles for collision
- Future challenges:
  - Triangulations that optimize k and $\frac{d_{max}}{d_{min}}$
  - Deformation methods that maintain k and $\frac{d_{max}}{d_{min}}$
  - Application to other problems

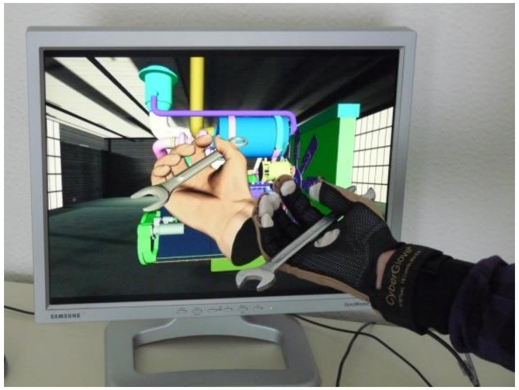
More related MSc Thesis Topics

- Massively-parallel Voronoi diagram computation with

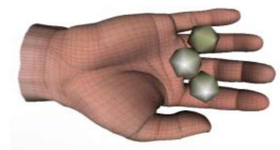
- Massively-parallel machine learning algorithms for CG challenges (e.g. self-organizing-maps, k-means)
- Application of sphere packings to other CG challenges
  - Object classification
  - Object segmentation
- Sphere-based sound propagation in VR
-





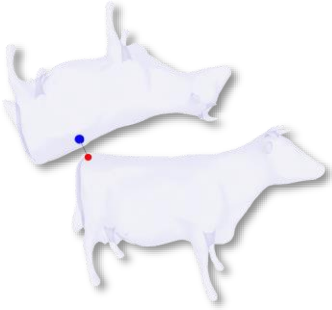


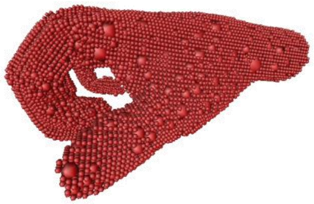
Pick Up Your MSc-Thesis!



Protosphere

Kinetic CollDet





VR-Coralreef

KINAPTIK