CSCI 4560/6560 Computational Geometry

https://www.cs.rpi.edu/~cutler/classes/computationalgeometry/F23/

Lecture 7: Randomized Incremental Construction

Outline for Today

- Homework…
- Last Time: Half-Space Intersections & Randomized Incremental Construction
- A Sample Quiz Problem?
- Motivation/Application: Smallest Bounding Sphere
  - Collision Detection, Ray Tracing, Robot Placement
- Brute Force Minimal Smallest Bounding Circle
- Bounding Circle by Center of Mass
- Incremental Construction of Smallest Bounding Circle
- Randomized Incremental Construction
- Next Time: Point Location & Orthogonal Range Searching

Homework 1 Grading (still in progress)

- Read the book problem (even more) carefully
- Sometimes necessary to get into the nitty gritty math details
    - "Pseudocode" = similar to code, not just high level comments within code
    - How do you compute the angle between two vectors/lines? *Good to know/learn*
    - How do you "sort" points in 2D? *Increasing dimension can make a problem more expensive, unclear, undefined, or even impossible!*

- Sometimes degeneracies can be ignored – *State your assumptions clearly*
- Sometimes degeneracies cannot be ignored:
    - *Convex hull does not include points on a boundary edge between 2 other vertices*

- Proof Writing: "Proof by contradiction", "Proof by induction", etc.
    - What are you actually trying to prove? Have a clear plan.

Homework 1 Grading (still in progress)

- Try not to stress about the homework score
- Semester grades will be generously curved :)

- Remember that sometimes theory is about figuring out the insight (sometimes it even feels like a “trick”) that allows you to contradict an assumption, or simplify/reduce the problem, etc.
  - Try not to stress if you can’t figure it out quickly
  - Try not to stress if you can’t figure it out on your own
  - Ask for a hint or help if you’re stuck

Even expert theorists rely on co-authors/colleagues/reviewers to proofread their proofs and point out typos & counter-examples/bugs

Homework Autograding

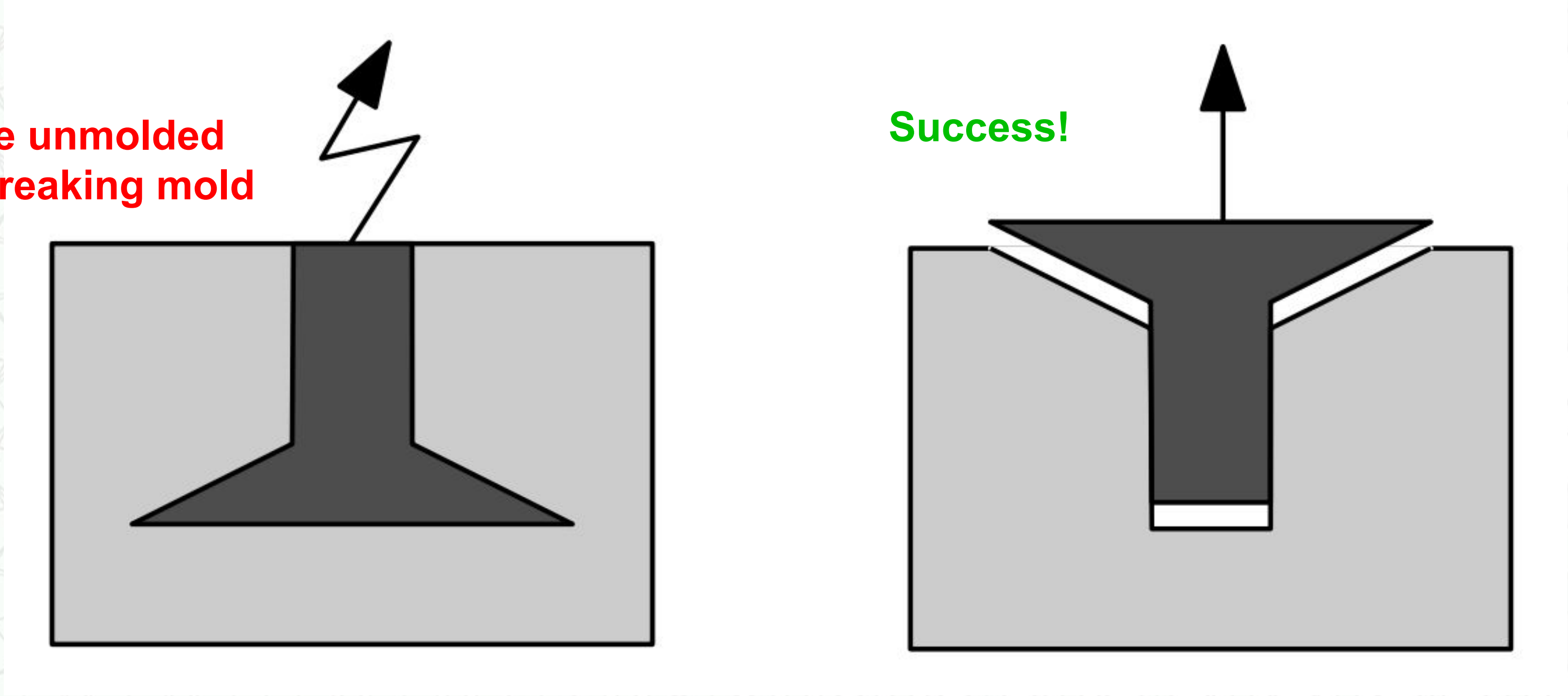
- If it is unclear why you aren't getting full credit, please ask
- Some errors:
  - Specific string keywords/spaces expected
  - Clockwise vs. counter-clockwise winding order
- Qt drawing windows are "blocking"
  - Don't launch before you have written your output files

    Submitty isn't attempting to close these windows,
    your program is just force killed after a 10 second timeout

Outline for Today

- Homework…
- Last Time: Half-Space Intersections & Randomized Incremental Construction
- A Sample Quiz Problem?
- Motivation/Application: Smallest Bounding Sphere
    - Collision Detection, Ray Tracing, Robot Placement
- Brute Force Minimal Smallest Bounding Circle
- Bounding Circle by Center of Mass
- Incremental Construction of Smallest Bounding Circle
- Randomized Incremental Construction
- Next Time: Point Location & Orthogonal Range Searching

Motivation: Manufacturing by Mold Casting

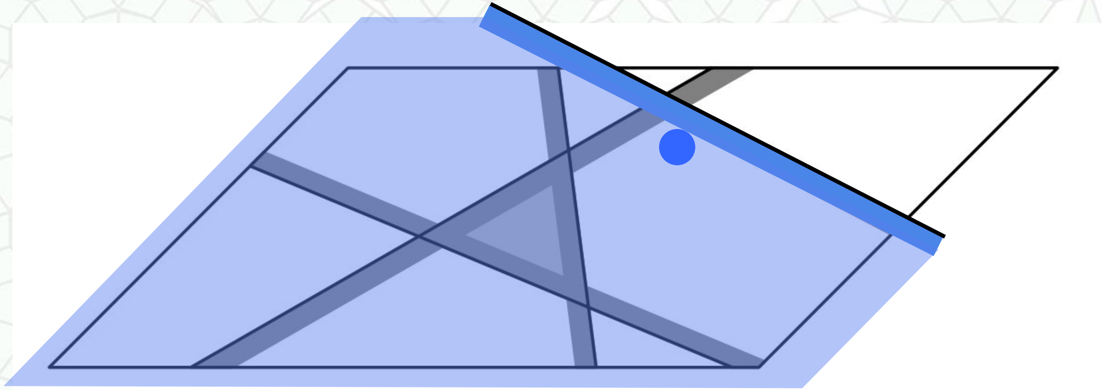


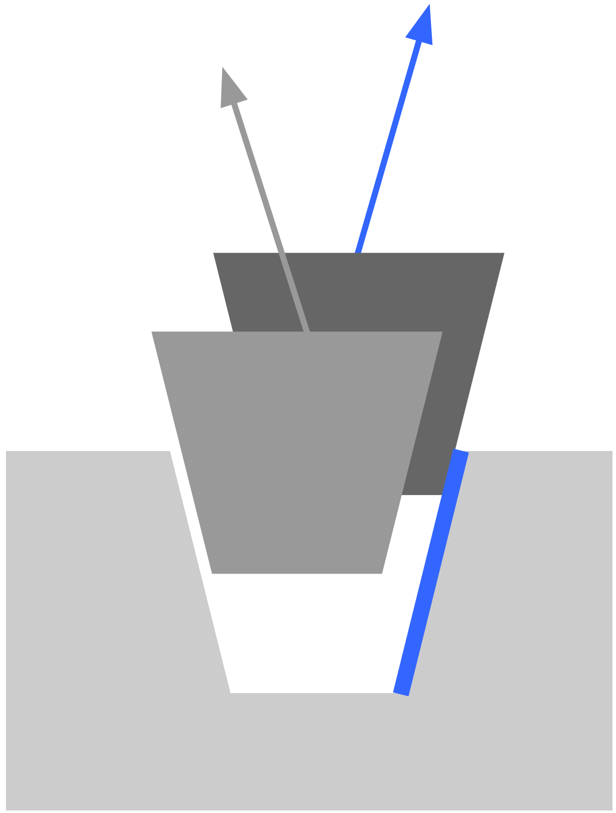
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 4

- Each facet places a *linear constraint* on the valid unmolding directions

$$n_x d_x + n_y d_y + n_z \leq 0$$

- This half-plane / half-space space can be visualized on our dual representation z=1

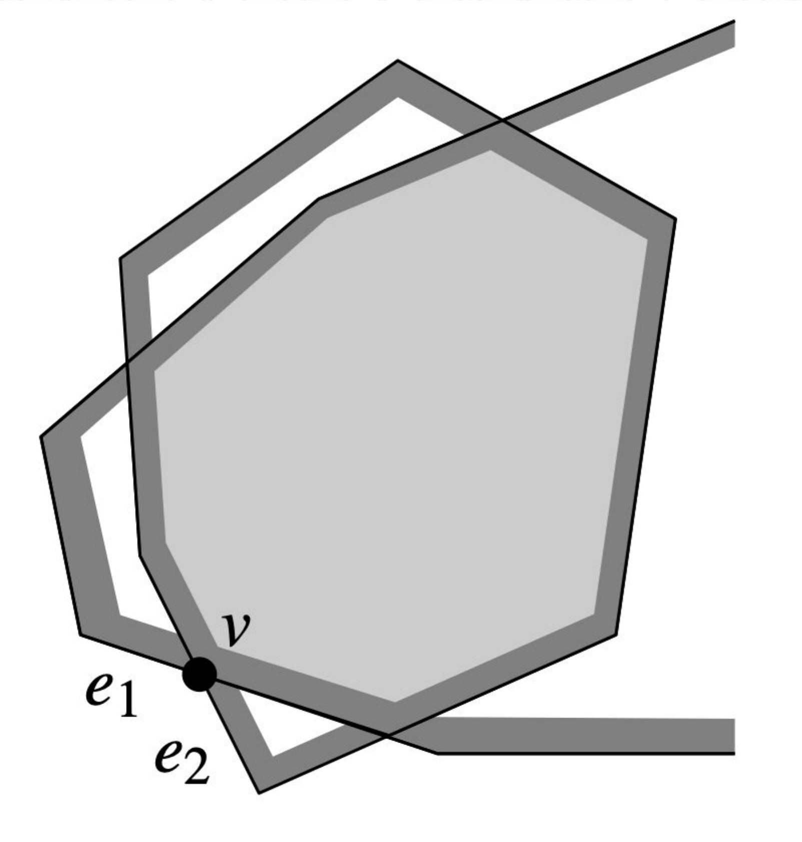




Half Space Intersection

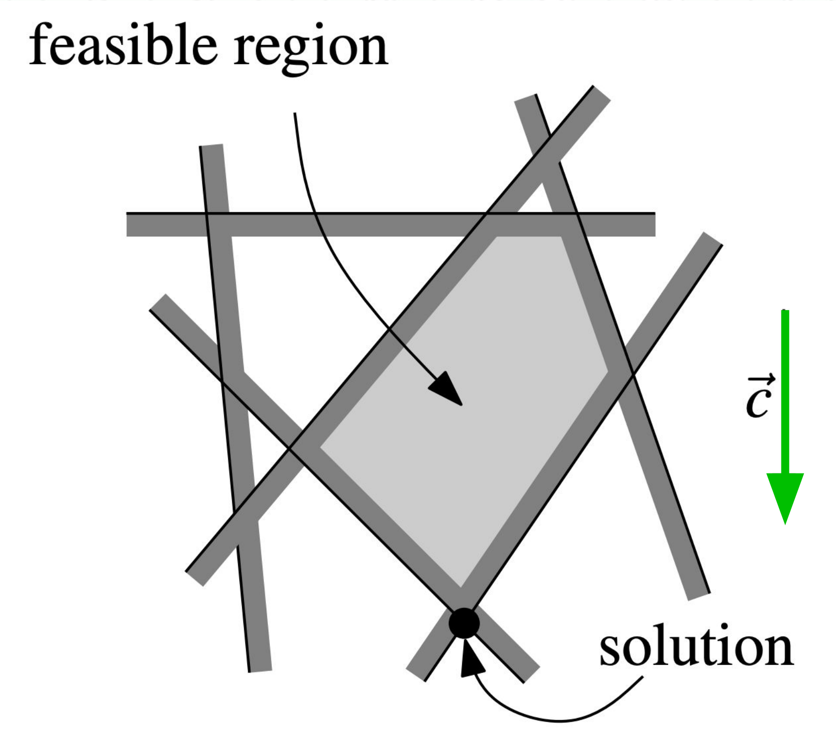
- Compute Feasible Region (a Convex Polygon) by Divide & Conquer:
  - Convex Overlay of 2 Convex Polygons → *O(n)*
  - Full recursive solution: → *O(n log n)*

- *Computing **the region** is expensive & unnecessary if we only need **one valid point** inside the feasible region*



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 4

Linear Optimization, a.k.a. Linear Programming



objective function

Maximize $c_1x_1 + c_2x_2 + \cdots + c_dx_d$

Subject to

$$\begin{aligned} a_{1,1}x_1 + \cdots + a_{1,d}x_d &\leqslant b_1 \\ a_{2,1}x_1 + \cdots + a_{2,d}x_d &\leqslant b_2 \\ &\vdots \\ a_{n,1}x_1 + \cdots + a_{n,d}x_d &\leqslant b_n \end{aligned}$$

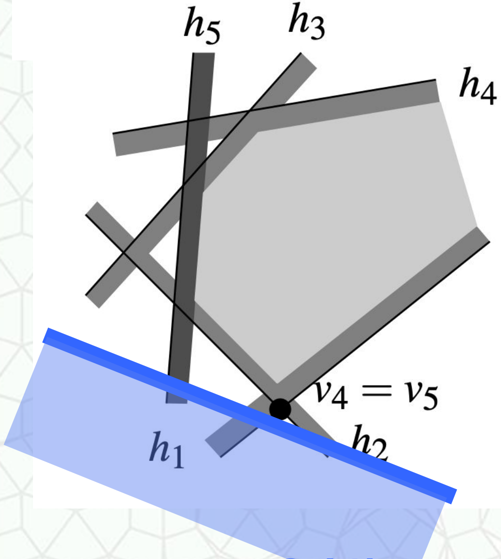
constraints

Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 4

Incremental Solution - Analysis

- At each step, we will add in the next halfspace constraint h_{i+1}

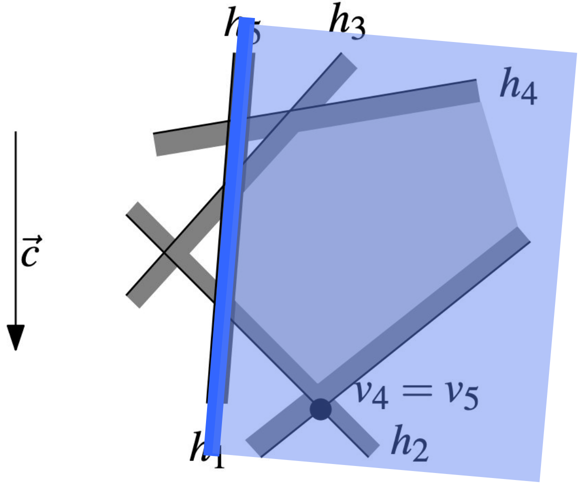
Infeasible - no solution



→ ***O(1)***

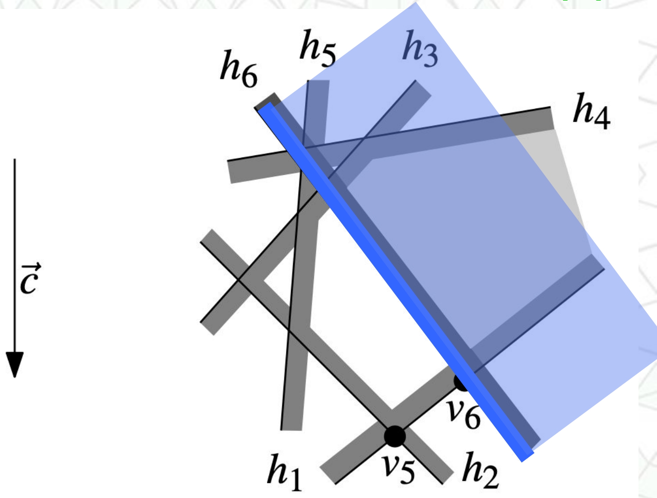
short circuit exit!

Satisfied: $v_1 = v_{i+1}$



→ ***O(1)***

Compute new v_{i+1}



→ ***O(n)***

Incremental Solution - Analysis

- Order the half-space constraints in some order: $h_1, h_2, h_3, \ldots h_n$
- We will solve incremental versions of the problem: $C_1, C_2, C_3, \ldots C_n$

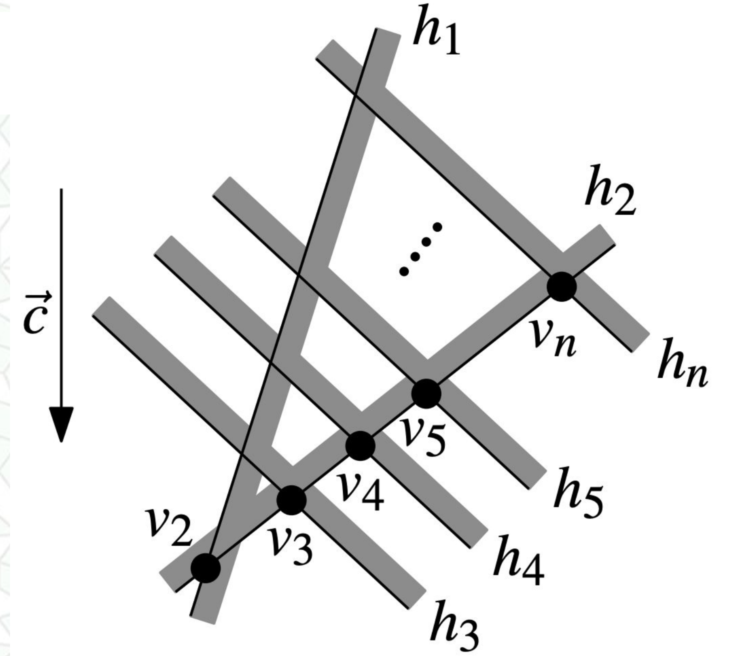
  → ***O(n)***

- Which have optimal solutions:

  $v_1, v_2, v_3, \ldots v_n$

- C_i has with half-space constraints $\{ h_1, h_2, h_3, \ldots h_i \}$ with solution v_i

  Overall:
  → O(n^2) worst case



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 4

Randomized Linear Programming

randomize the order of the halfspaces

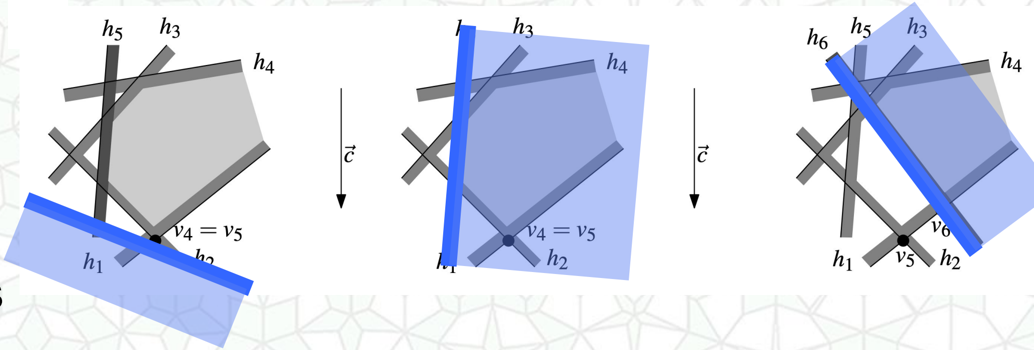
- Order the half-space constraints in some order: $h_1, h_2, h_3, \dots h_n$
- We will solve incremental versions of the problem: $C_1, C_2, C_3, \dots C_n$

  → ***O(n)***

- Which have optimal solutions:

  $v_1, v_2, v_3, \dots v_n$

- C_i has with half-space constraints $\{ h_1, h_2, h_3, \dots h_i \}$ with solution v_i



→ ***O(1)***
short circuit exit!

→ ***O(1)***

→ ***O(n)***

Can be shown that the case to recompute the solution is rare…

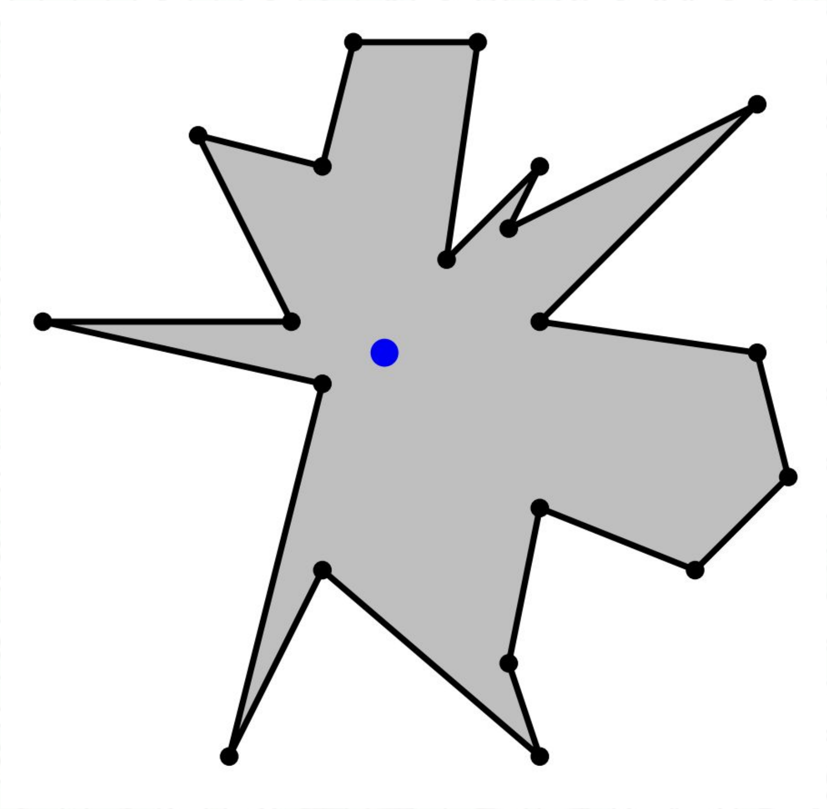
Overall:
→ O(n) expected case

Outline for Today

- Homework…
- Last Time: Half-Space Intersections & Randomized Incremental Construction
- A Sample Quiz Problem?
- Motivation/Application: Smallest Bounding Sphere
    - Collision Detection, Ray Tracing, Robot Placement
- Brute Force Minimal Smallest Bounding Circle
- Bounding Circle by Center of Mass
- Incremental Construction of Smallest Bounding Circle
- Randomized Incremental Construction
- Next Time: Point Location & Orthogonal Range Searching

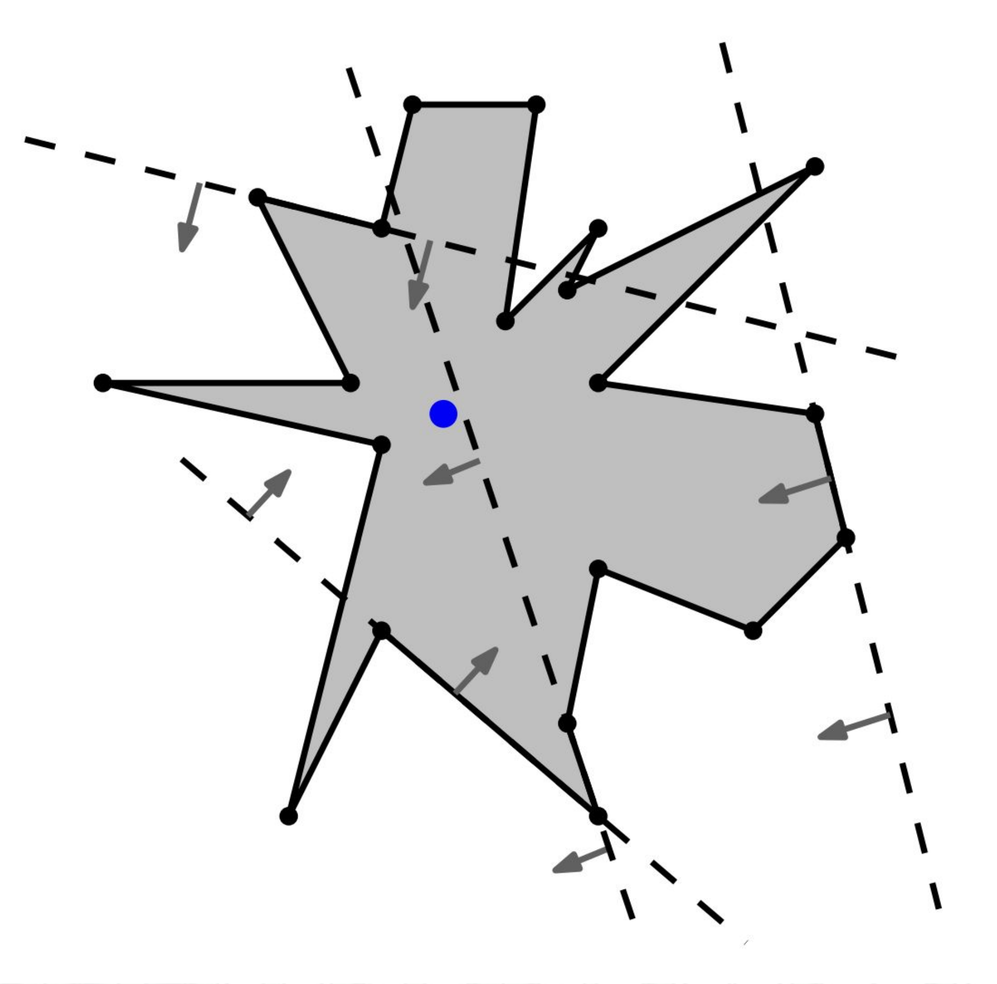
One Guardable Polygons

Problem: Given a simple polygon with n vertices, can we decide efficiently if one guard is enough?



Frank Staals,
http://www.cs.uu.nl/docs/vakken/ga/2021/

One Guardable Polygons



Frank Staals,
http://www.cs.uu.nl/docs/vakken/ga/2021/

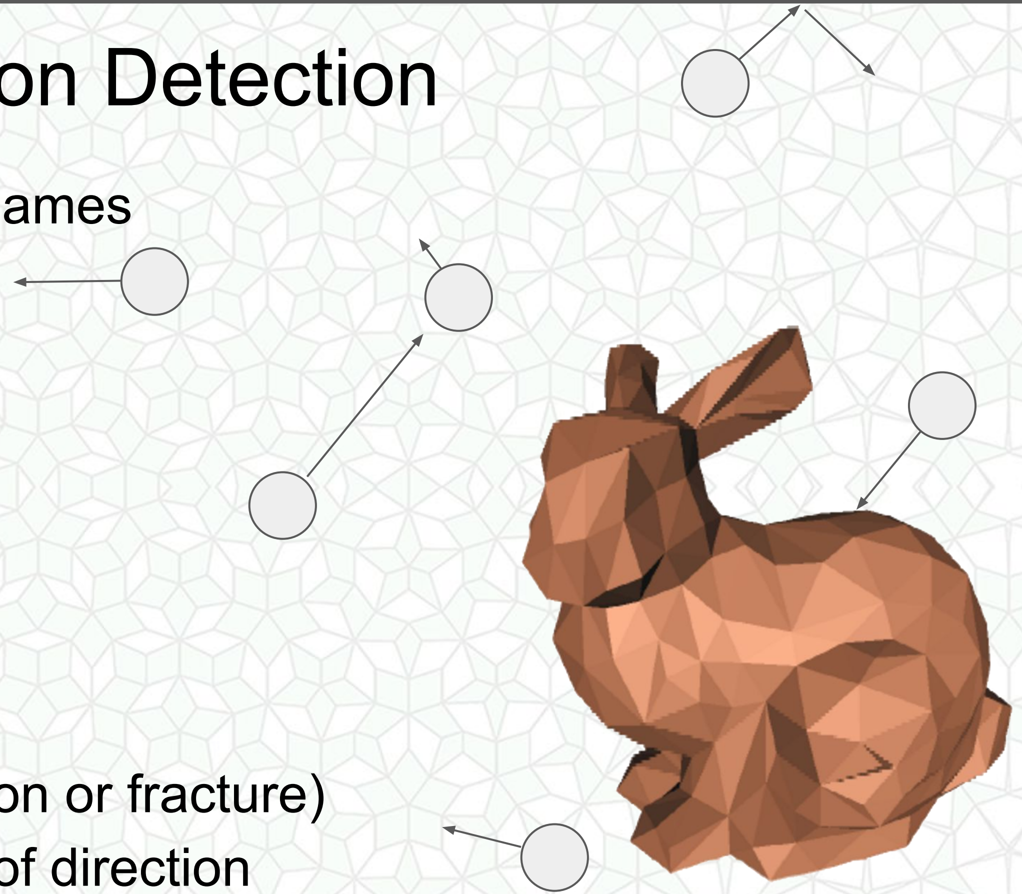
Outline for Today

- Homework…
- Last Time: Half-Space Intersections & Randomized Incremental Construction
- A Sample Quiz Problem?
- Motivation/Application: Smallest Bounding Sphere
    - Collision Detection, Ray Tracing, Robot Placement
- Brute Force Minimal Smallest Bounding Circle
- Bounding Circle by Center of Mass
- Incremental Construction of Smallest Bounding Circle
- Randomized Incremental Construction
- Next Time: Point Location & Orthogonal Range Searching

Application: Collision Detection

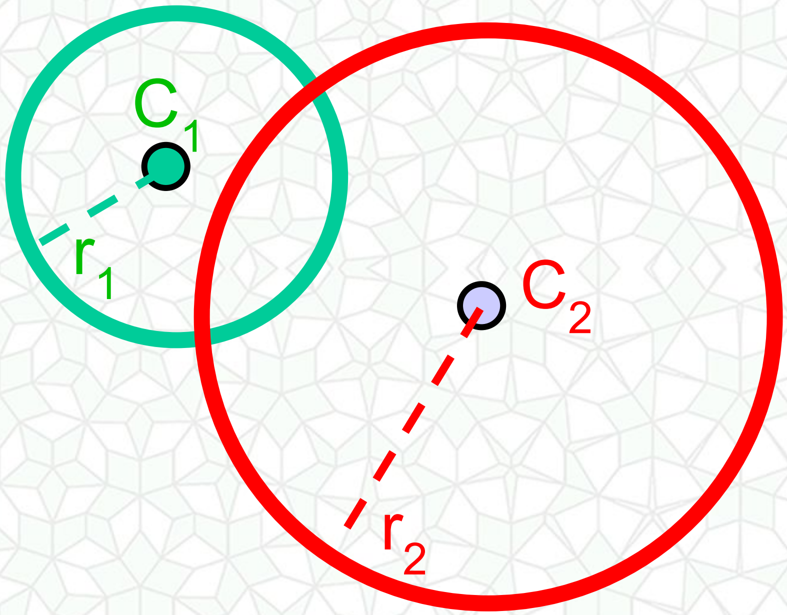
- Virtual Reality / Video Games
- Robotics
- Scientific Simulations

- Simulation over *time*
- *Detect* collisions
- Compute response:
  - Force of impact
  - Damage (deformation or fracture)
  - Bouncing / change of direction



Intersect Two Spheres

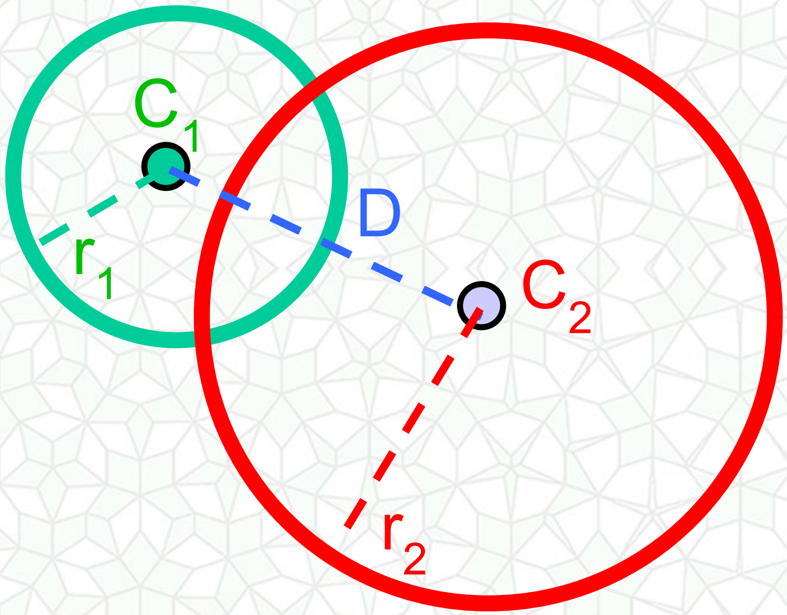
- Collision Detection / Overlap test between two spheres?



Intersect Two Spheres

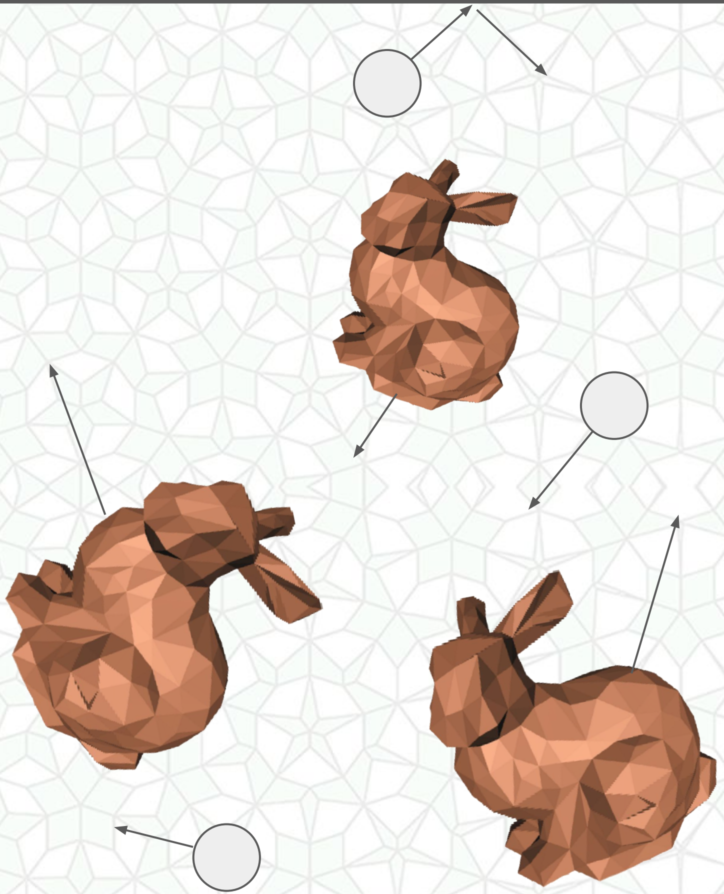
- Collision Detection / Overlap test between two spheres?

- Compute D, the distance between centers
- $D(C_1, C_2) < r_1 + r_2$



Cost of Collision Detection?

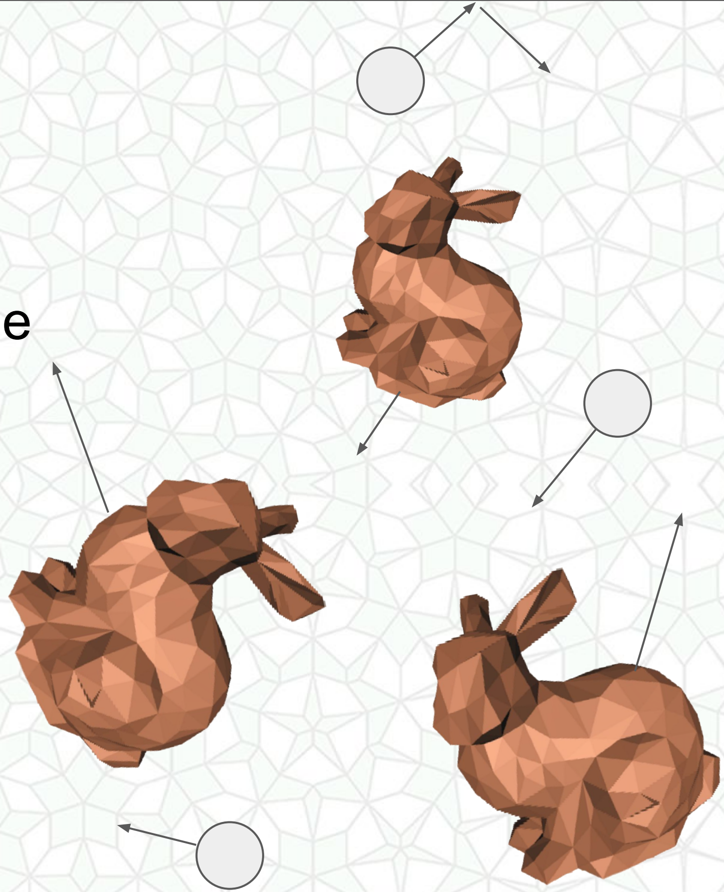
- If we have *n* bouncing ping pong balls inside of a box (6 quads)?
- If we add a stationary bunny statue (w/ $f=60,000$ faces) inside the box?
- What if we add *b* bunny statues bouncing around inside the box?



Naive Collision Detection

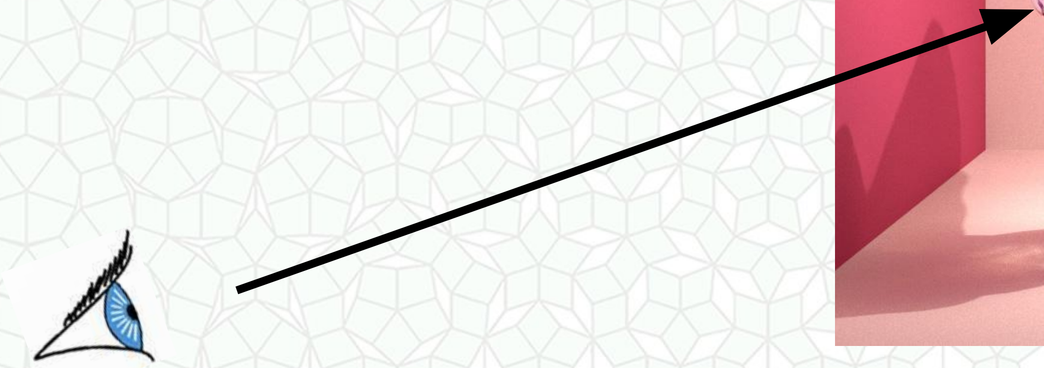
- Every frame of animation/simulation, intersect every sphere/triangle in motion with every other sphere/triangle (both stationary and in motion)

→ $O((n + b*f + 6) * (n + b*f))$



Application: Ray Tracing

- Cast *g = 1 gazillion* rays to simulate photons bouncing off of objects (& through objects!)
- Naive: Intersect every ray with every triangle

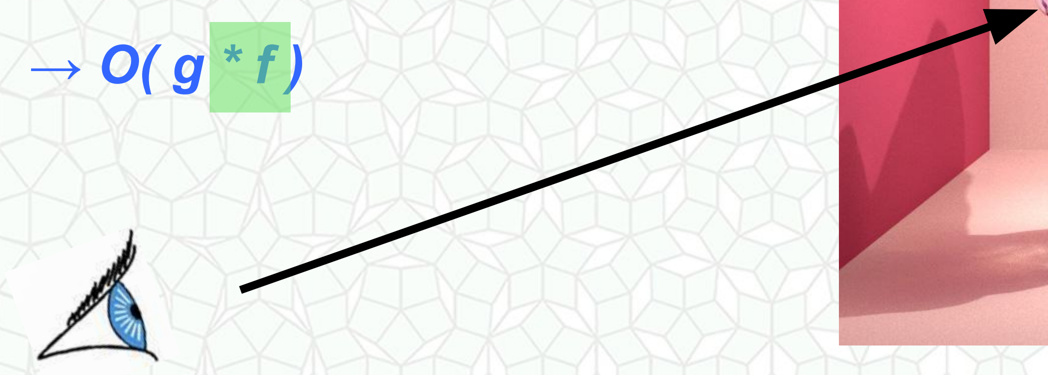


Laura Lediaev
http://www.omnigraphica.com/classes/cs6620/index.html

Application: Ray Tracing

- Cast *g = 1 gazillion* rays to simulate photons bouncing off of objects (& through objects!)
- Naive: Intersect every ray with every triangle

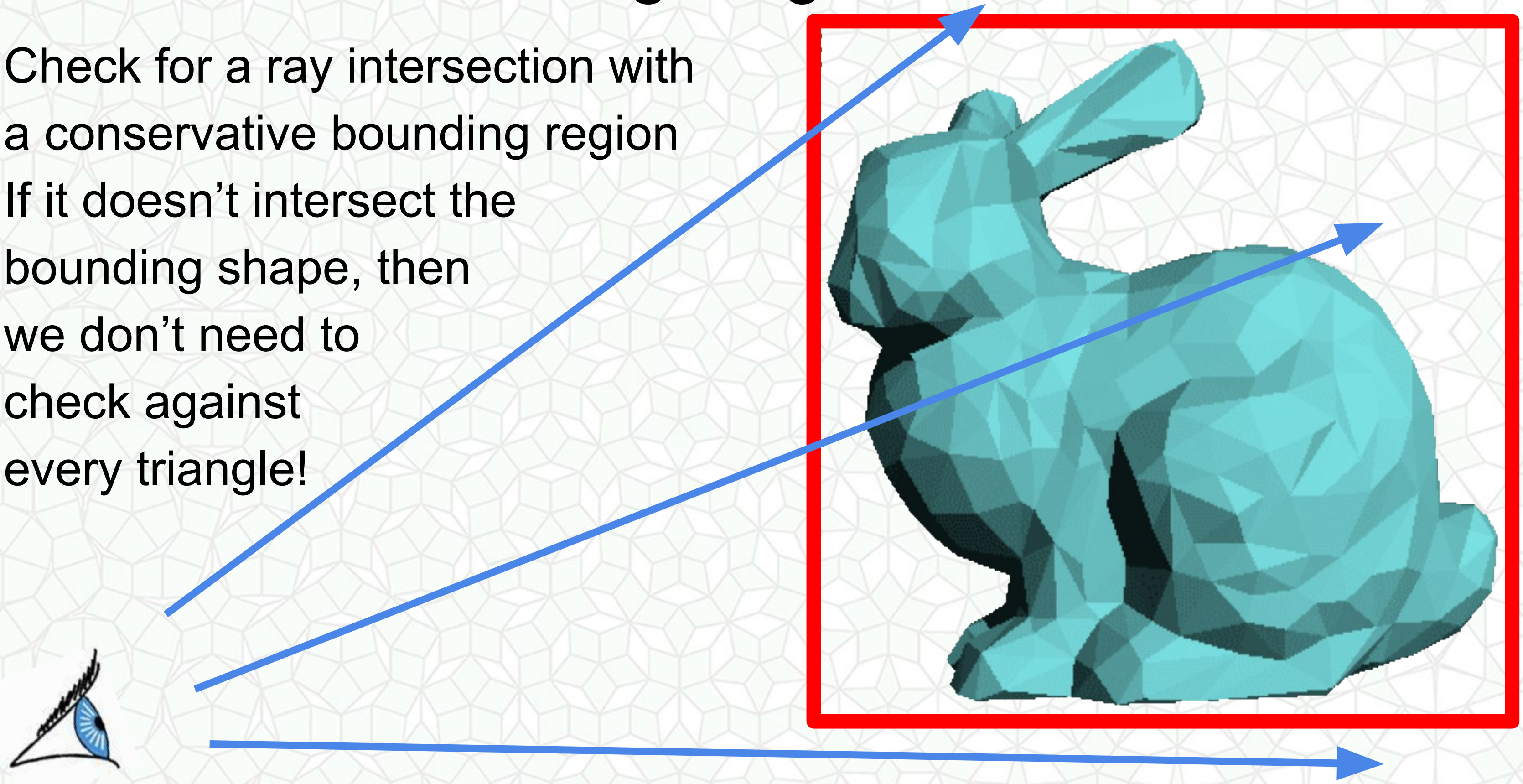
→ *O(g * f)*



Laura Lediaev
http://www.omnigraphica.com/classes/cs6620/index.html

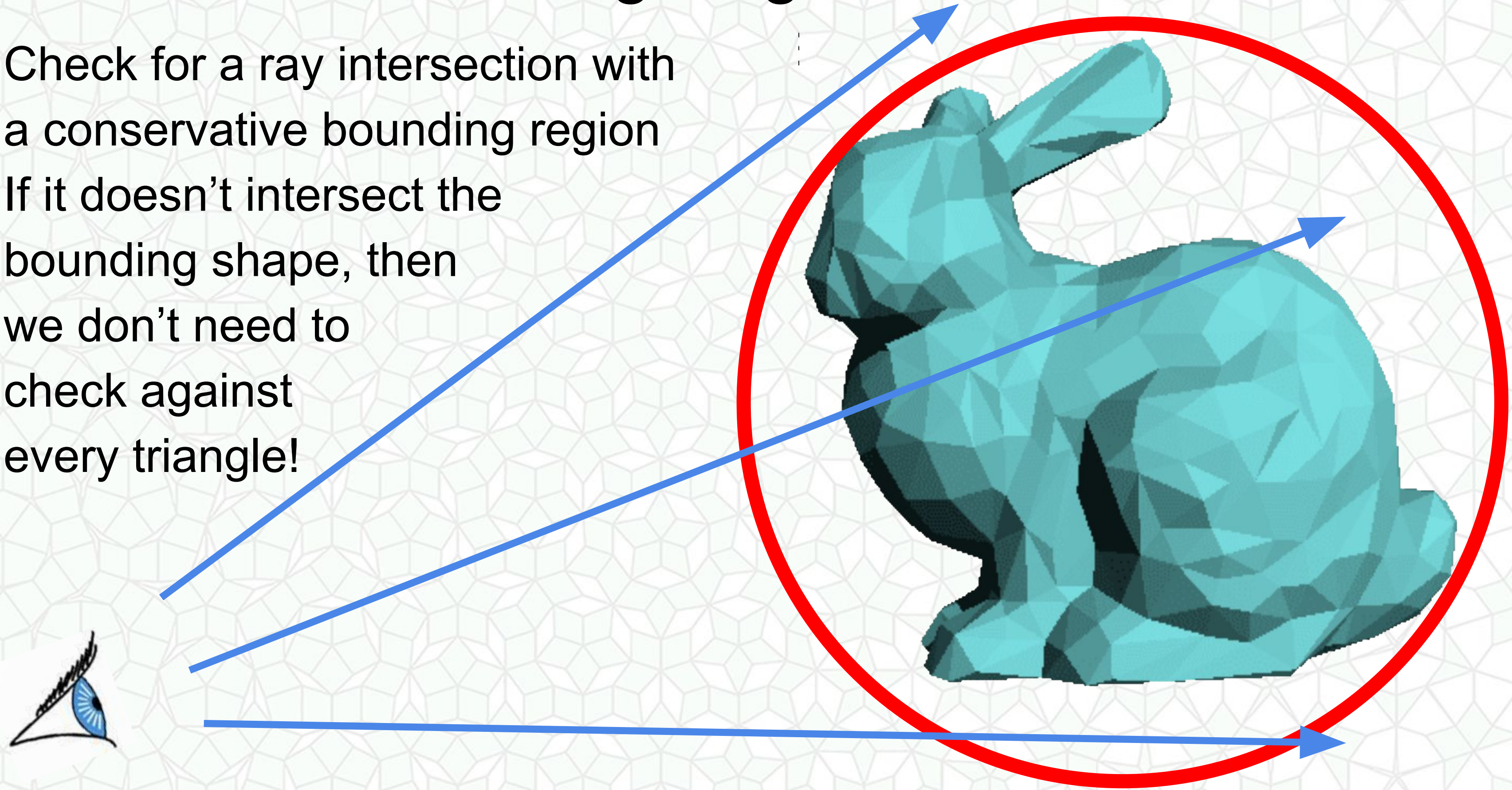
Conservative Bounding Region

- Check for a ray intersection with a conservative bounding region
- If it doesn't intersect the bounding shape, then we don't need to check against every triangle!



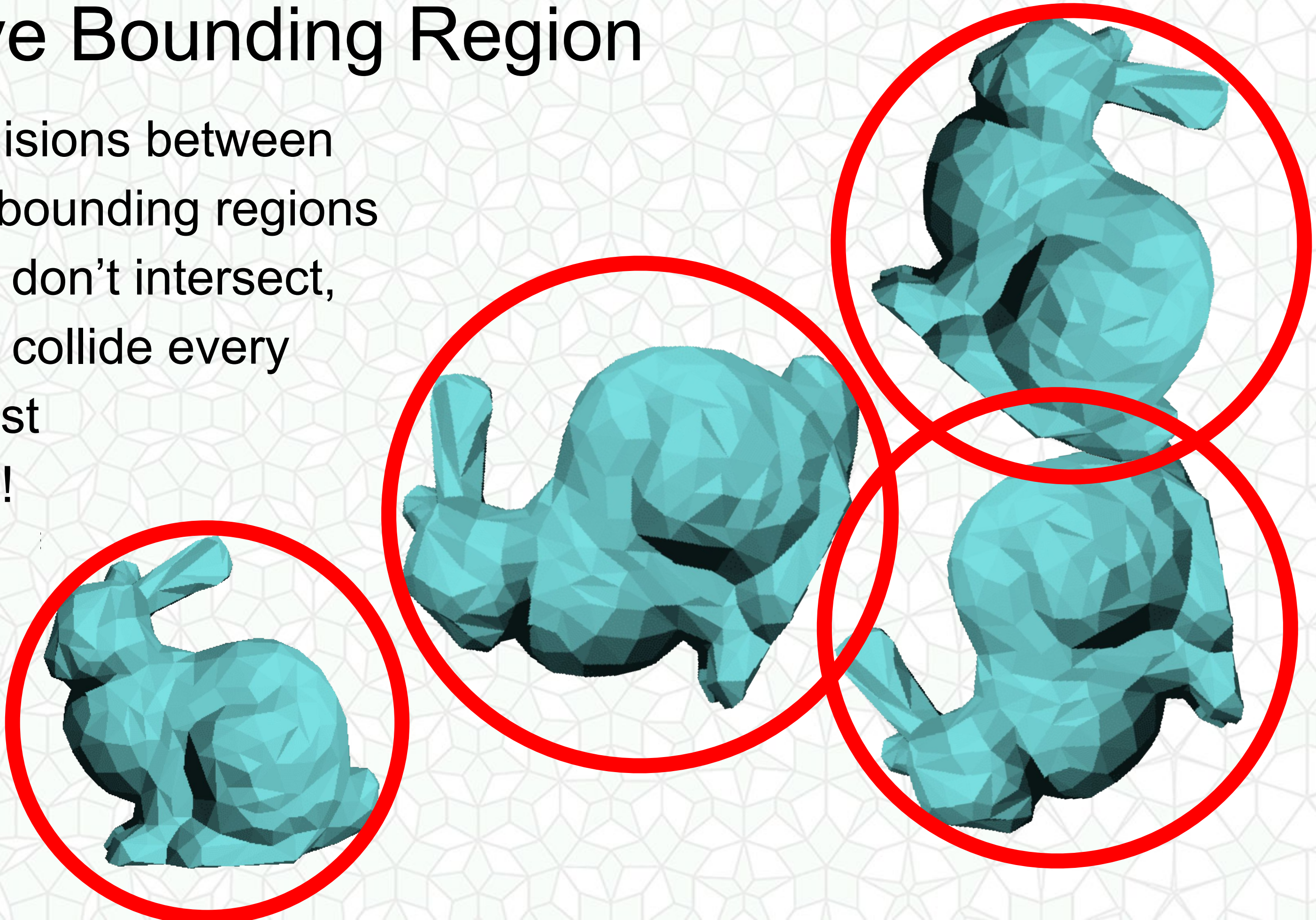
Conservative Bounding Region

- Check for a ray intersection with a conservative bounding region
- If it doesn't intersect the bounding shape, then we don't need to check against every triangle!



Conservative Bounding Region

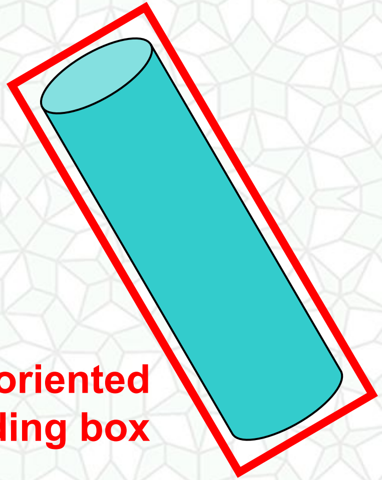
- Check for collisions between conservative bounding regions
- If two regions don't intersect, then we don't collide every triangle against every triangle!

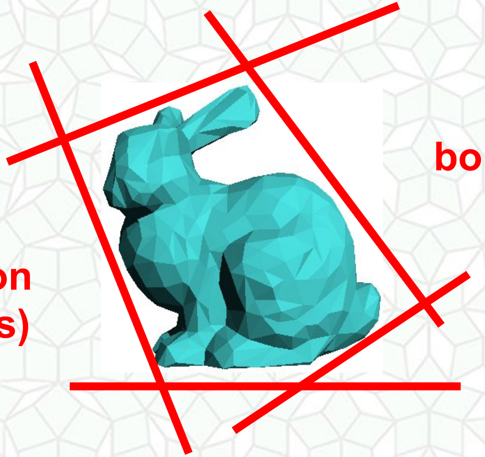


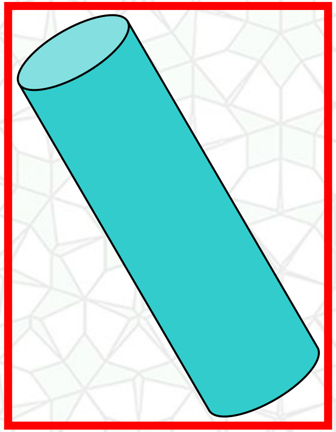
Conservative Bounding Regions

Requirements:

- tight → avoid false positives
- fast to intersect
- easy/fast/perfect construction (*less important*)

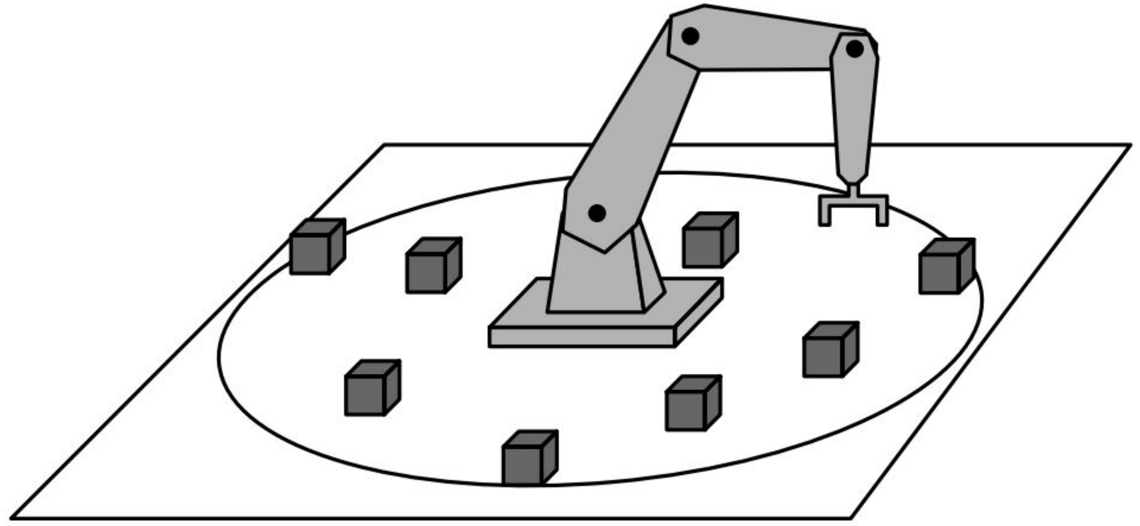






Another Application: Robot Placement

- We need a fixed-base robot to reach a bunch of objects from a set of *n* a known positions



- What is the smallest robot necessary (minimum arm length)?
- Where should the robot base be located?

Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 4

Outline for Today

- Homework…
- Last Time: Half-Space Intersections & Randomized Incremental Construction
- A Sample Quiz Problem?
- Motivation/Application: Smallest Bounding Sphere
  - Collision Detection, Ray Tracing, Robot Placement
- Brute Force Minimal Smallest Bounding Circle
- Bounding Circle by Center of Mass
- Incremental Construction of Smallest Bounding Circle
- Randomized Incremental Construction
- Next Time: Point Location & Orthogonal Range Searching

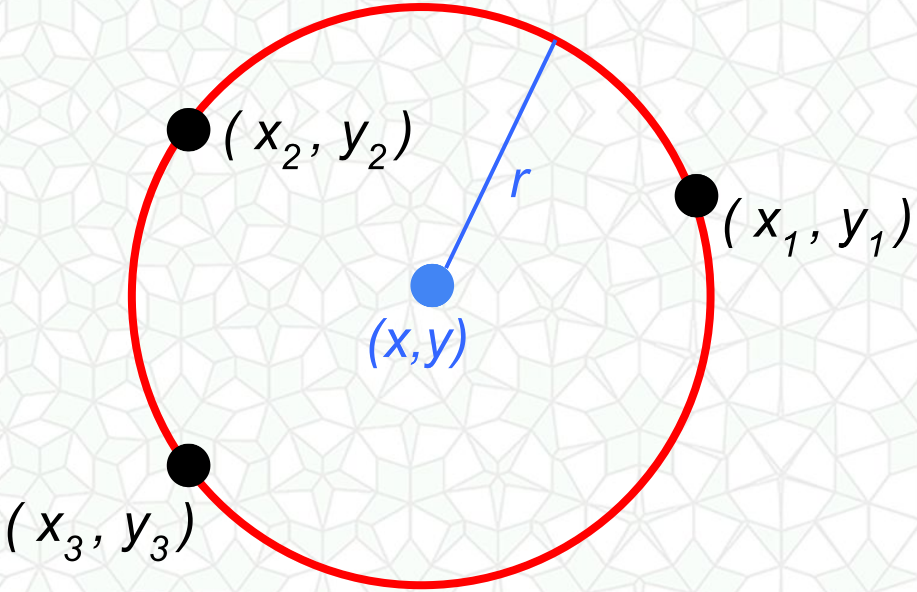
Problem: Minimal Bounding ~~Sphere~~ Circle

- Input: *n* vertices in ~~3D~~ 2D
- Assume (for convenience):
  "General Position"
  - No 3 points are collinear
  - No 4 points lie on the same circle
- Output: 3 of those vertices uniquely define a circle such that all other points lie inside of that circle

Note: In 3D, we would output 4 vertices (4 vertices uniquely define a sphere)

bounding sphere

How to Fit a Circle to 3 Points? *(not collinear)*



How to Fit a Circle to 3 Points? *(not collinear)*

Points: (x_1, y_1) (x_2, y_2) (x_3, y_3)

Solve for center = (x, y) and radius = r

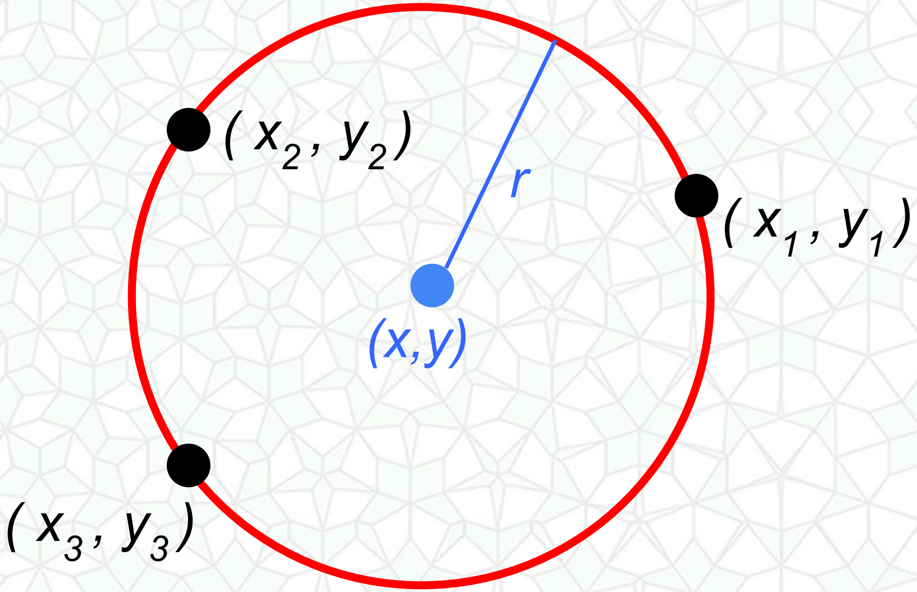
Solve system of equations:

3 equations, 3 unknowns

$$(x_1-x)^2 + (y_1-y)^2 = r^2$$

$$(x_2-x)^2 + (y_2-y)^2 = r^2$$

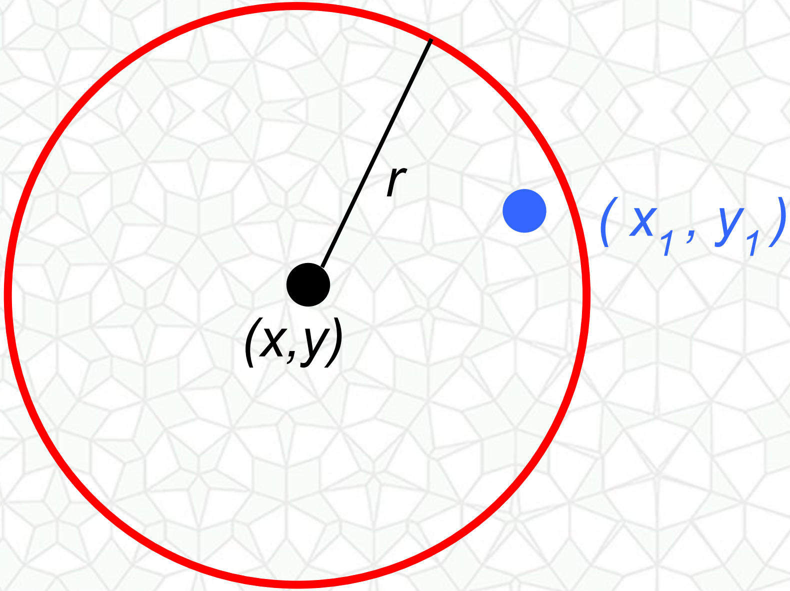
$$(x_3-x)^2 + (y_3-y)^2 = r^2$$



How to Test if Point is Inside/Outside Circle?

Point: (x_1, y_1)

Circle: center = (x, y) and radius = r



How to Test if Point is Inside/Outside Circle?

Point: (x_1, y_1)

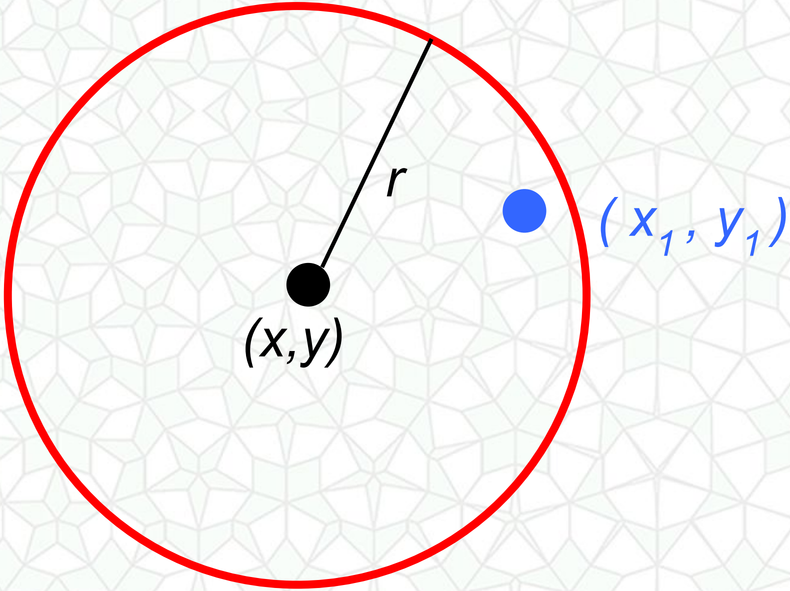
Circle: center = (x , y) and radius = r

Evaluate:

$(x_1-x)^2 + (y_1-y)^2 > r^2$ → *outside circle*

$(x_1-x)^2 + (y_1-y)^2 = r^2$ → *on edge of circle*

$(x_1-x)^2 + (y_1-y)^2 < r^2$ → *inside circle*



Brute Force Minimal Bounding Circle

- Input: *n* vertices in 2D

Brute Force Minimal Bounding Circle

- Input: *n* vertices in 2D
- For every triplet of those points
    - Compute circle
    - Check against all other points
        - Reject if any are outside circle

Overall Analysis:

Brute Force Minimal Bounding Circle

- Input: *n* vertices in 2D
- For every triplet of those points
  → " *n chose 3* " triplets = $n! / (3! * (n-3)!)$
  $= n*(n-1)*(n-2)/6 = O(n^3)$
  - Compute circle → $O(1)$
  - Check against all other points
    → $O(n)$
    - Reject if any are outside circle

Overall Analysis: → $O(n^4)$ *can we do better?*

Outline for Today

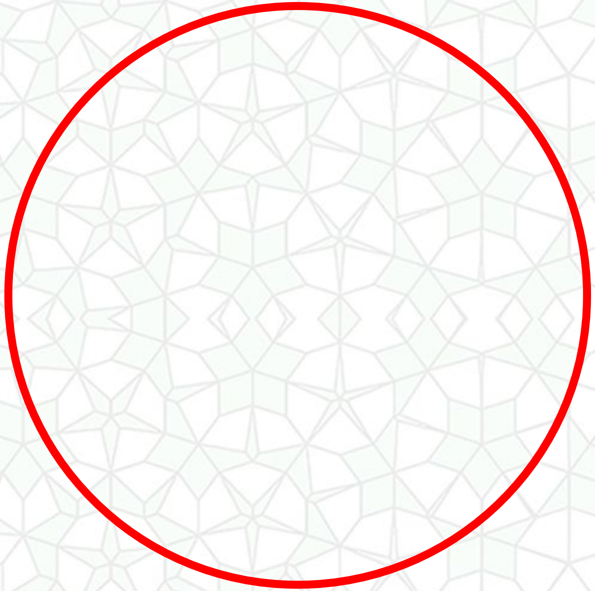
- Homework…
- Last Time: Half-Space Intersections & Randomized Incremental Construction
- A Sample Quiz Problem?
- Motivation/Application: Smallest Bounding Sphere
  - Collision Detection, Ray Tracing, Robot Placement
- Brute Force Minimal Smallest Bounding Circle
- Bounding Circle by Center of Mass
- Incremental Construction of Smallest Bounding Circle
- Randomized Incremental Construction
- Next Time: Point Location & Orthogonal Range Searching

Bounding Circle *by Center of Mass*

- Let the center = average of all of the vertices

Bounding Circle *by Center of Mass*

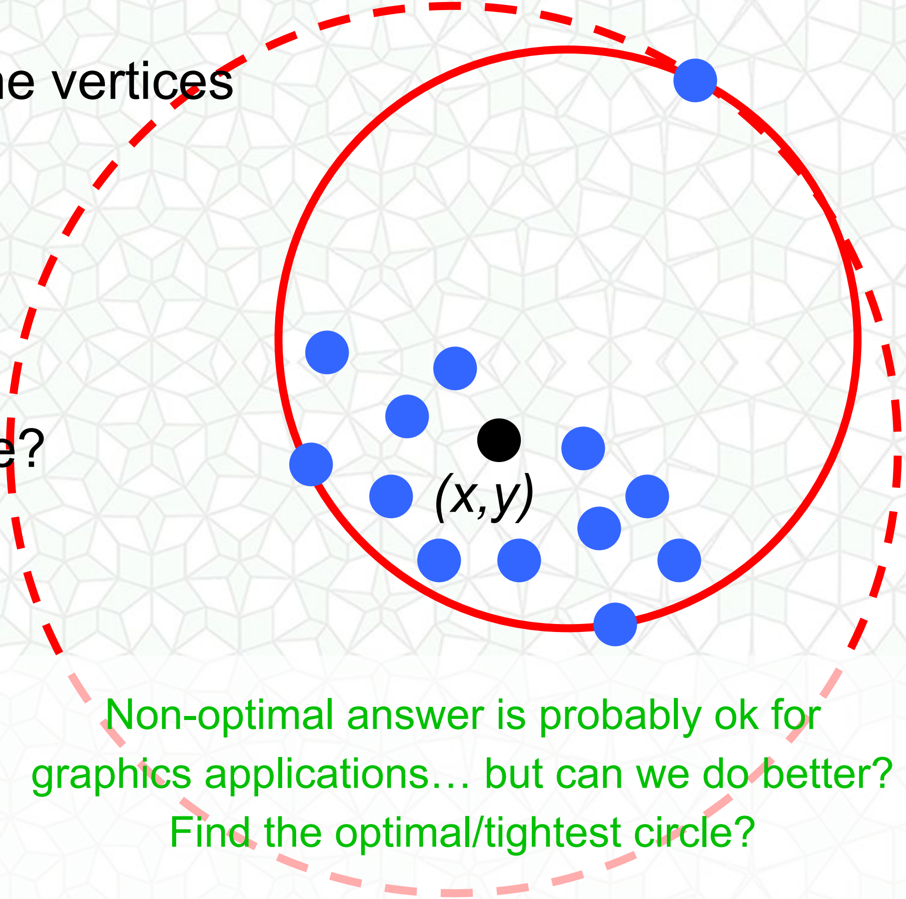
- Let the center = average of all of the vertices
- Find point furthest from center,
  use that to set the radius
- Are all points on or inside this circle?
- Overall running time?
- Is this optimal/tightest circle?



Bounding Circle *by Center of Mass*

- Let the center = average of all of the vertices
  → *O(n)*
- Find point furthest from center,
  use that to set the radius
  → *O(n)*
- Are all points on or inside this circle?
  → *yes!*
- Overall running time? → *O(n)*

- Is this optimal/tightest circle?
  Probably not, maybe only 1 point on circle



Non-optimal answer is probably ok for graphics applications… but can we do better? Find the optimal/tightest circle?

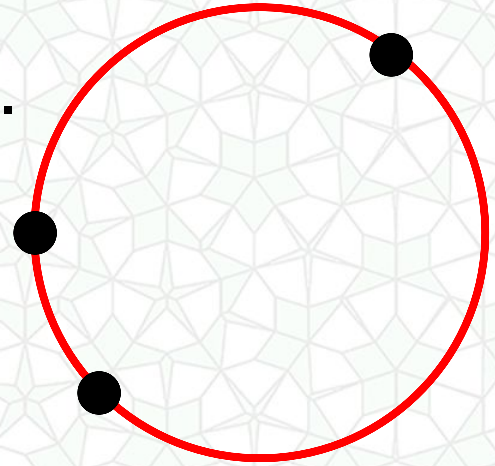
Outline for Today

- Homework …
- Last Time: Half-Space Intersections & Randomized Incremental Construction
- A Sample Quiz Problem?
- Motivation/Application: Smallest Bounding Sphere
  - Collision Detection, Ray Tracing, Robot Placement
- Brute Force Minimal Smallest Bounding Circle
- Bounding Circle by Center of Mass
- Incremental Construction of Smallest Bounding Circle
- Randomized Incremental Construction
- Next Time: Point Location & Orthogonal Range Searching

Let's Try Incremental Construction…

- Make a circle with the first 3 points p_1, p_2, p_3
- Loop over all of the remaining points

  For i = 4 … n

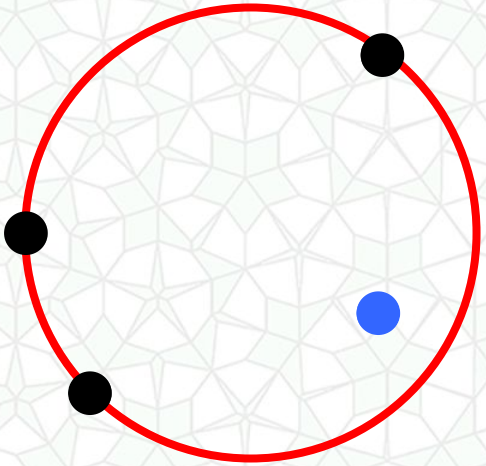


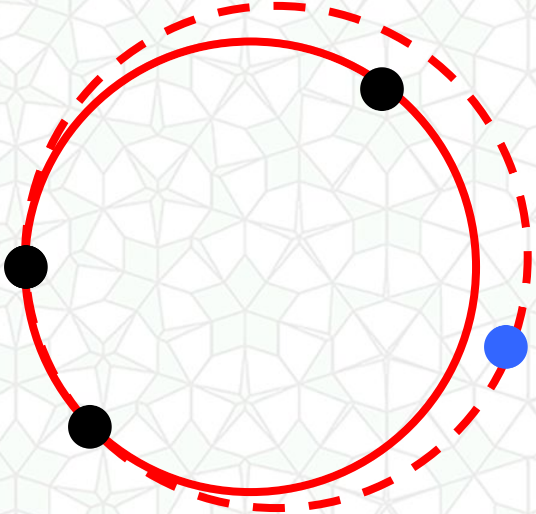
Incremental Construction

- Make a circle with the first 3 points p_1, p_2, p_3
- Loop over all of the remaining points

  For i = 4 … n

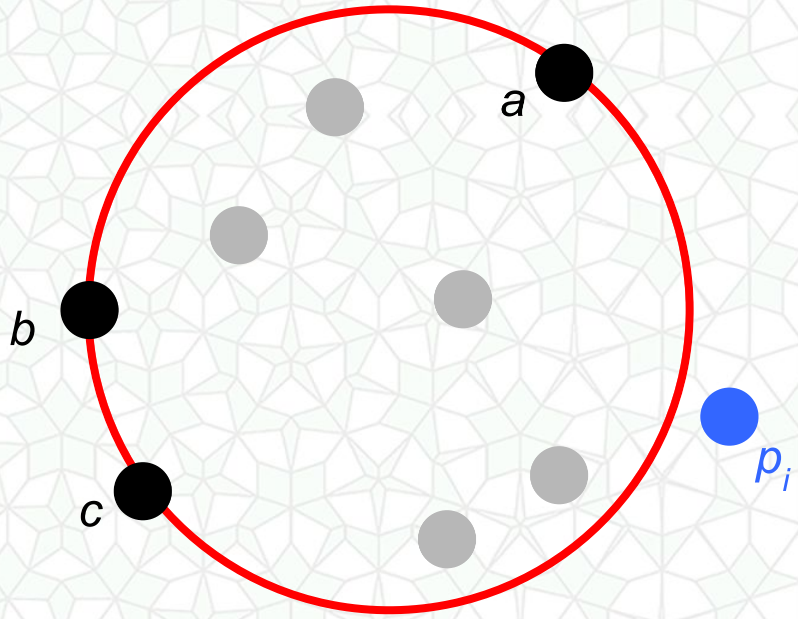
  - If the p_i is inside the circle, then
    the solution for points { $p_1 \rightarrow p_{i-1}$ }
    is also the solution for points { $p_1 \rightarrow p_i$ }
  - If p_i is outside the circle,
    then **solve for the new circle**
    NOTE: *p_i is definitely ON the circle solution for* { $p_1 \rightarrow p_i$ }





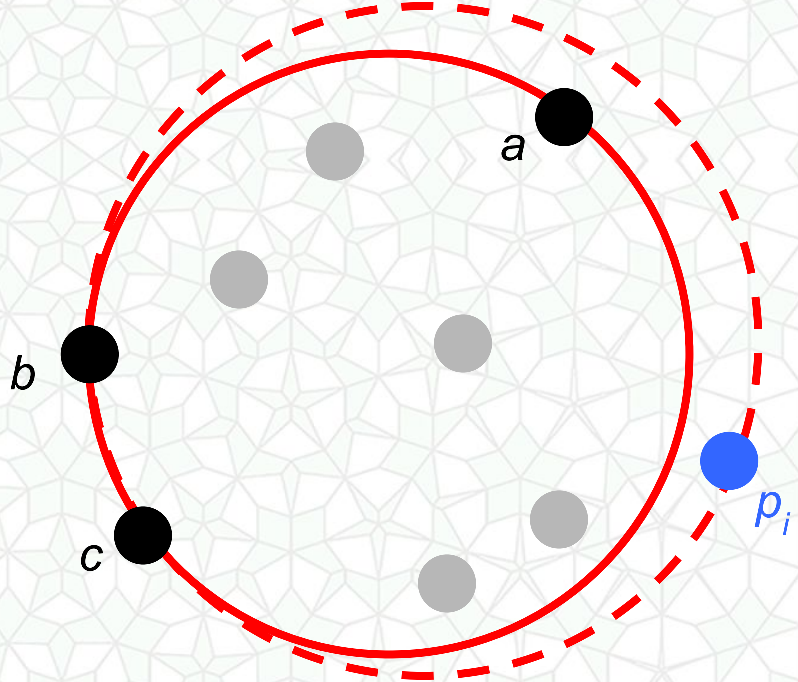
Complexity of Incremental Construction?

- If the current circle is fit to points *a, b, c*…



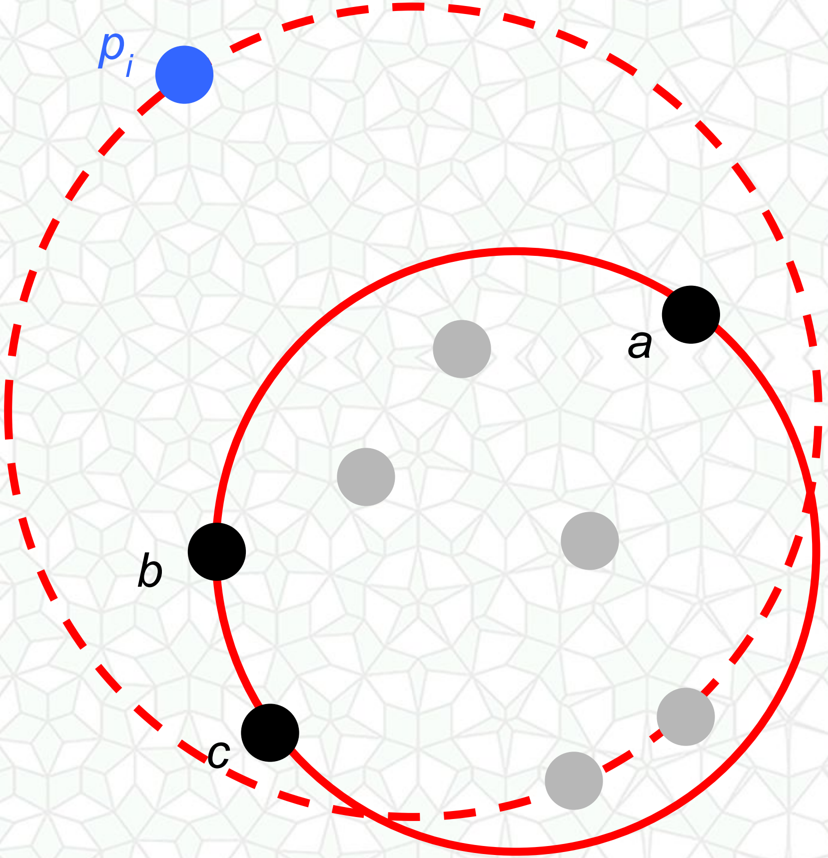
Complexity of Incremental Construction?

- If the current circle is fit to points *a, b, c*…
- Can we prove/disprove that adding p_i will be a circle fit to
  - a, b, p_i OR
  - b, c, p_i OR
  - a, c, p_i



Complexity of Incremental Construction?

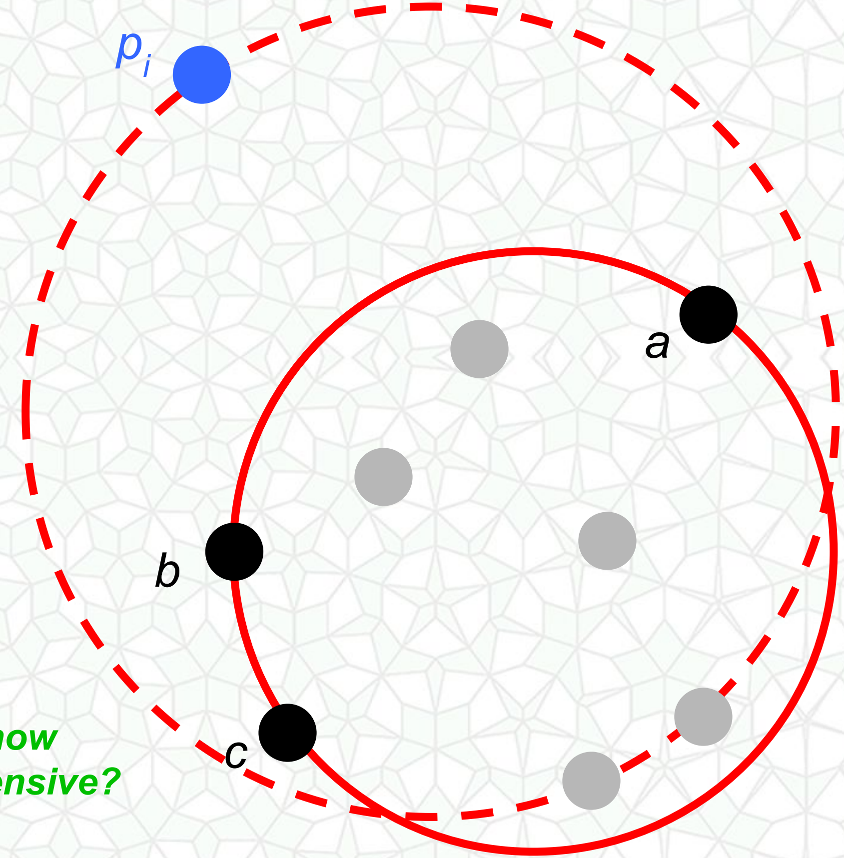
- If the current circle is fit to points *a, b, c*…
- Can we prove/disprove that adding p_i will be a circle fit to
  - a, b, p_i OR
  - a, c, p_i OR
  - b, c, p_i
- *Do we need to consider all other points? YES!!!*



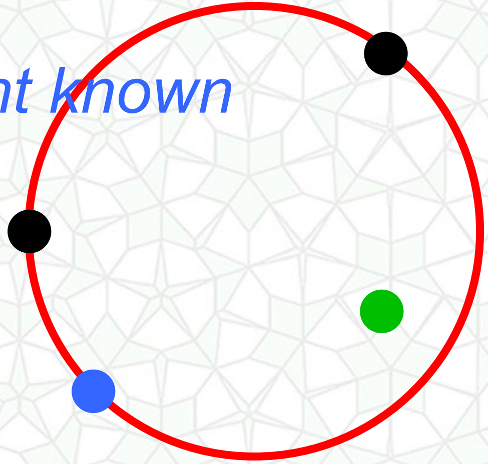
Complexity of Incremental Construction?

- If the current circle is fit to points *a, b, c…*
- Can we prove/disprove that adding p_i will be a circle fit to
  - a, b, p_i OR
  - a, c, p_i OR
  - b, c, p_i

  } ***would be constant time***
- *Do we need to consider all other points? YES!!!* } ***how expensive?***



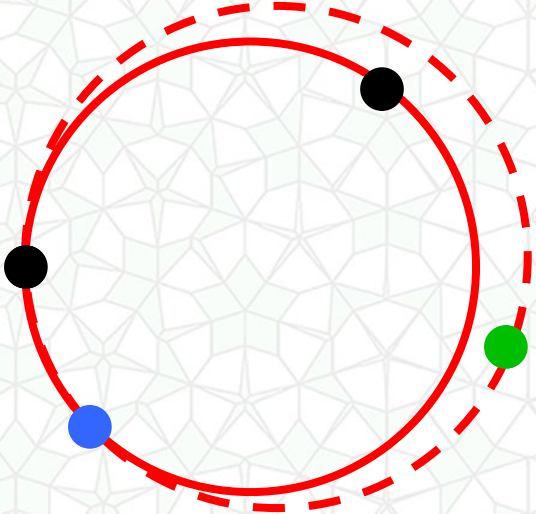
Incremental Construction *with one point known*



- Make a circle with the points p_i , p_1, p_2
- Loop over all of the remaining points

  For $j = 3 \ldots i-1$
  - If the p_j is inside the circle, then the solution for points { p_i , $p_1 \rightarrow p_{j-1}$ } is also the solution for points { p_i , $p_1 \rightarrow p_j$}
  - If the p_j is outside the circle, then **solve for the new circle**

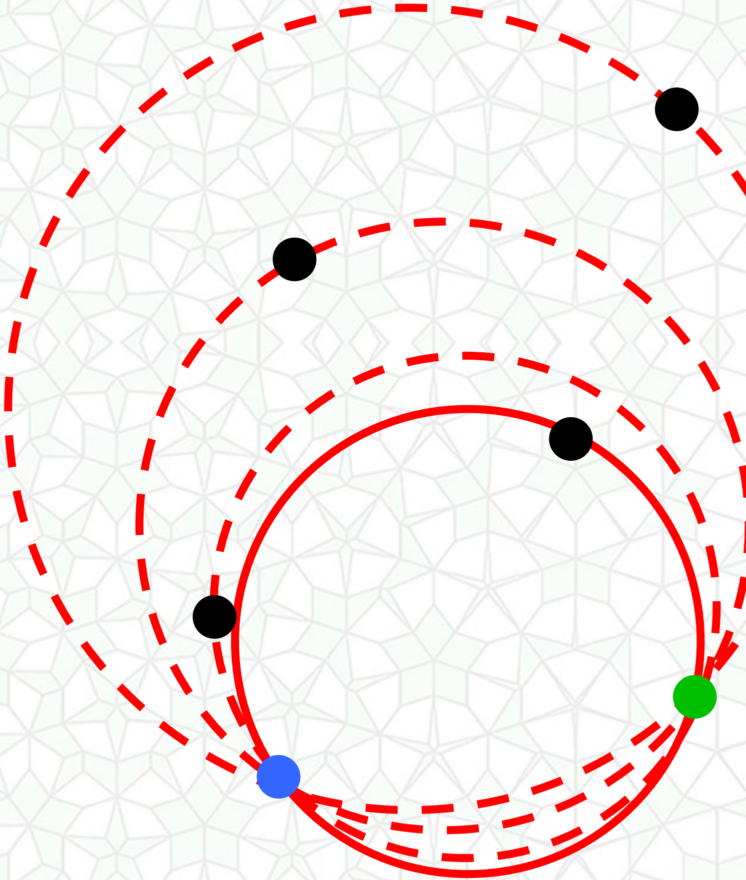
    NOTE: p_j *is definitely ON the circle solution for* { p_i , $p_1 \rightarrow p_j$ }



Incremental Construction *with two points known*

- Make a circle with the points p_i , p_j , p_1
- Loop over all of the remaining points

  For $k = 2 \ldots j-1$
  - If the p_k is inside the circle, then the solution for points $\{ p_i , p_j , p_1 \rightarrow p_{k-1} \}$ is also the solution for points $\{ p_i , p_j , p_1 \rightarrow p_k \}$
  - If the p_k is outside the circle, then the solution for $\{ p_i , p_j , p_1 \rightarrow p_k \}$ is the circle fit to p_i , p_j , p_k



Analysis of Incremental Construction

- Incremental Construction with two known points is:
  -
  -
- Incremental Construction with one known point is:
  - Worst case =
  - Best case =
- Overall, Incremental Construction is:
  - Worst case =
  - Best case =

Analysis of Incremental Construction

- Incremental Construction with two known points is: ***O(n)***
  - We have to check ***O(1)*** each of the *n* points
  - Computing a new circle ***O(1)*** will be done at most *n* times
- Incremental Construction with one known point is:
  - Worst case =
  - Best case =
- Overall, Incremental Construction is:
  - Worst case =
  - Best case =

Analysis of Incremental Construction

- Incremental Construction with two known points is: ***O(n)***
  - We have to check ***O(1)*** each of the *n* points
  - Computing a new circle ***O(1)*** will be done at most *n* times
- Incremental Construction with one known point is:
  - Worst case = $O(n^2)$ – if we compute a new circle, calling two known points function, *n* times
  - Best case = ***O(n)*** – never or rarely call the two known points function
- Overall, Incremental Construction is:
  - Worst case =
  - Best case =

Analysis of Incremental Construction

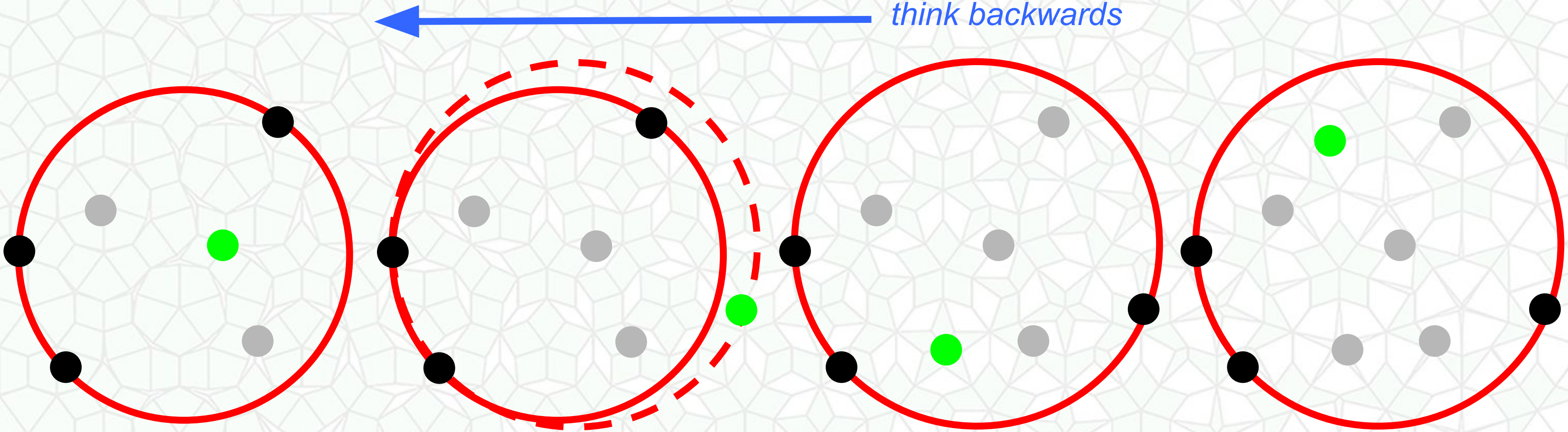
- Incremental Construction with two known points is: ***O(n)***
  - We have to check ***O(1)*** each of the *n* points
  - Computing a new circle ***O(1)*** will be done at most *n* times
- Incremental Construction with one known point is:
  - Worst case = $O(n^2)$ – if we compute a new circle, calling two known points function, *n* times
  - Best case = ***O(n)*** – never or rarely call the two known points function
- Overall, Incremental Construction is:
  - Worst case = $O(n^3)$ – if we compute a new circle, calling the one known point function, *n* times
  - Best case = ***O(n)*** – never or rarely call the one known point function

Outline for Today

- Homework…
- Last Time: Half-Space Intersections & Randomized Incremental Construction
- A Sample Quiz Problem?
- Motivation/Application: Smallest Bounding Sphere
    - Collision Detection, Ray Tracing, Robot Placement
- Brute Force Minimal Smallest Bounding Circle
- Bounding Circle by Center of Mass
- Incremental Construction of Smallest Bounding Circle
- Randomized Incremental Construction
- Next Time: Point Location & Orthogonal Range Searching

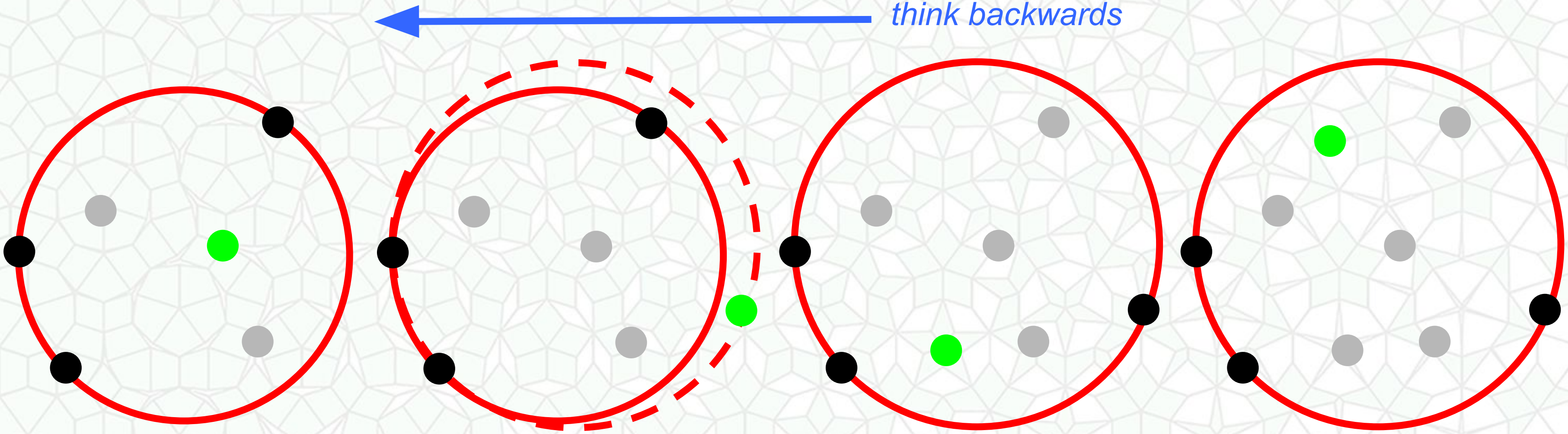
Randomized Incremental Construction

- If we randomize the initial order of the points, we will *RARELY* need to call the helper functions to compute the circles… Why???
- Let's think backwards… about removing points one at a time.



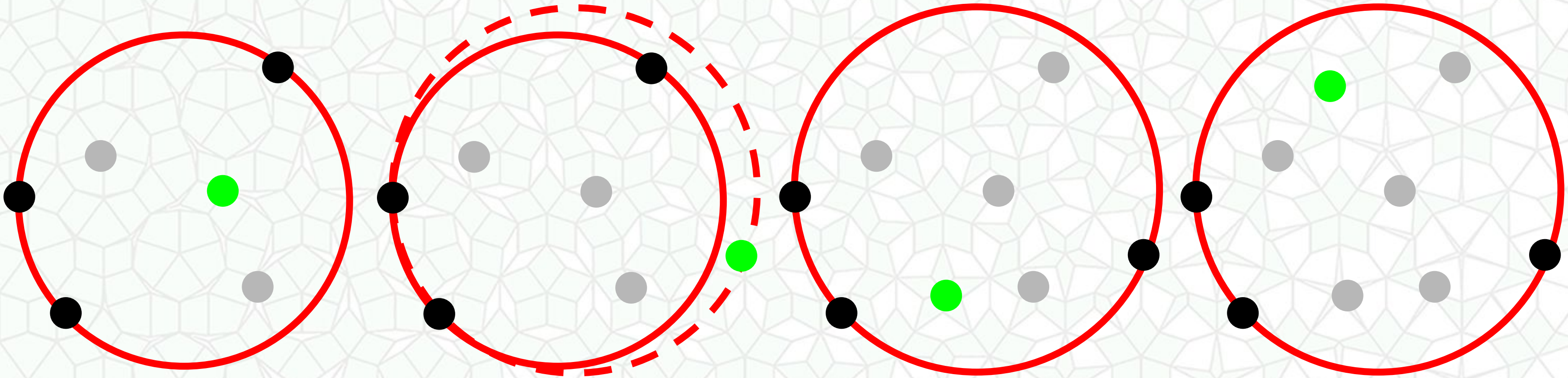
Randomized Incremental Construction

- We start with all *n* points and the optimal minimal bounding circle, which is defined by 3 of those points.
- Each step, we randomly choose one of *n* points to remove.



Randomized Incremental Construction

- Do we need to tighten & recompute the minimal bounding circle? Only *when / if* we remove one of the 3 circle-defining points.
- Expected chance we pick a point *on the circle*: *3/n* each step
- Expected: *O(1)* circle recomputes * *O(n)* per recompute → *O(n)*



Is Randomized Incremental Construction Magic?

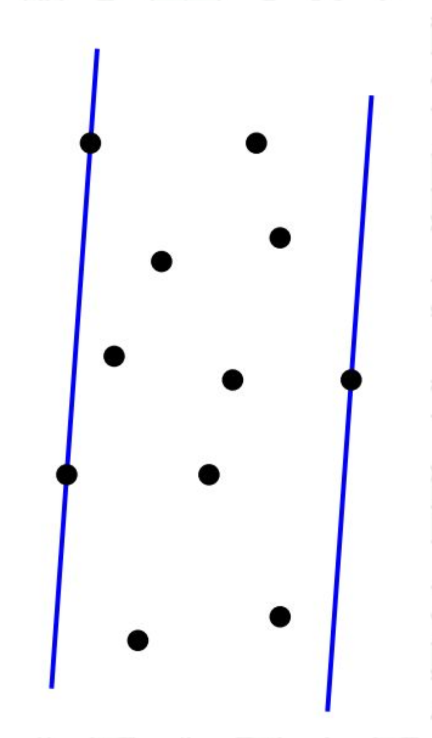
- Can we use it for every problem? ***No!***

- It only works if:
  - Fast to test if new item works with the current optimal solution
  - When new item does not work,
    - Current solution can be used to compute the new optimal
    - And it will be *faster than starting over from scratch*

Frank Staals,
http://www.cs.uu.nl/docs/vakken/ga/2021/

A Counter-Example: Minimum Strip Width

- Input: A set of 2D points
- Output: Two parallel lines that define the narrowest strip that contains all of the input points.

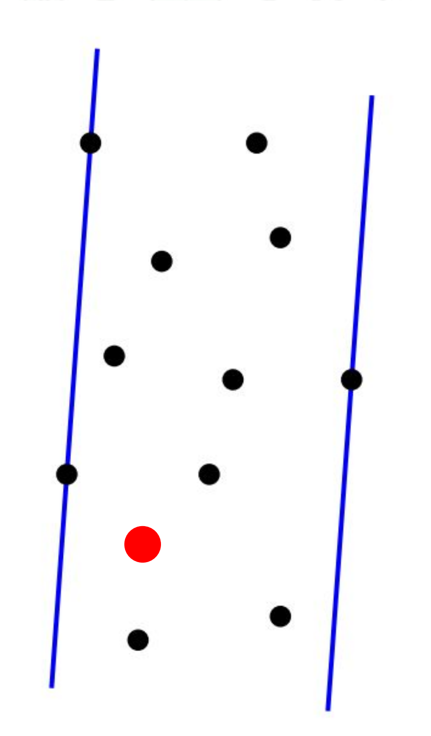


Frank Staals,http://www.cs.uu.nl/docs/vakken/ga/2021/

A Counter-Example: Minimum Strip Width

- Input: A set of 2D points
- Output: Two parallel lines that define the narrowest strip that contains all of the input points.

- *It is fast to test if a new point is contained in the strip*

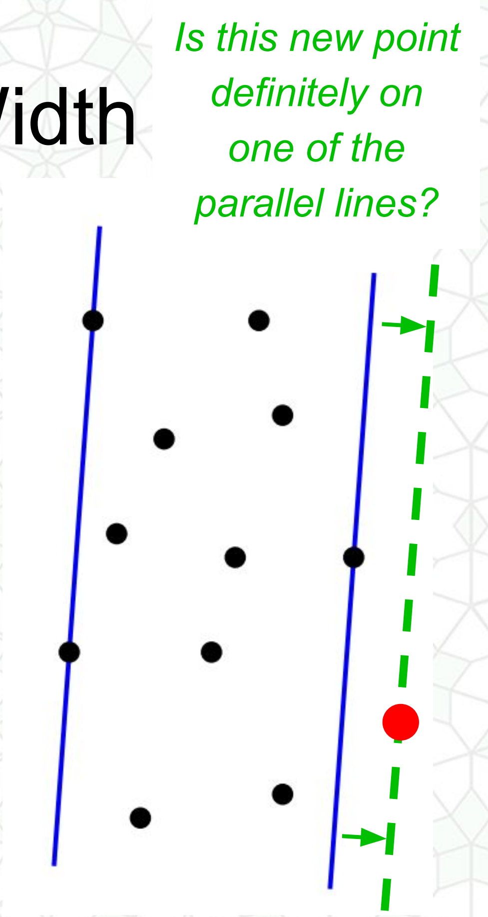


Frank Staals,http://www.cs.uu.nl/docs/vakken/ga/2021/

A Counter-Example: Minimum Strip Width

- Input: A set of 2D points
- Output: Two parallel lines that define the narrowest strip that contains all of the input points.

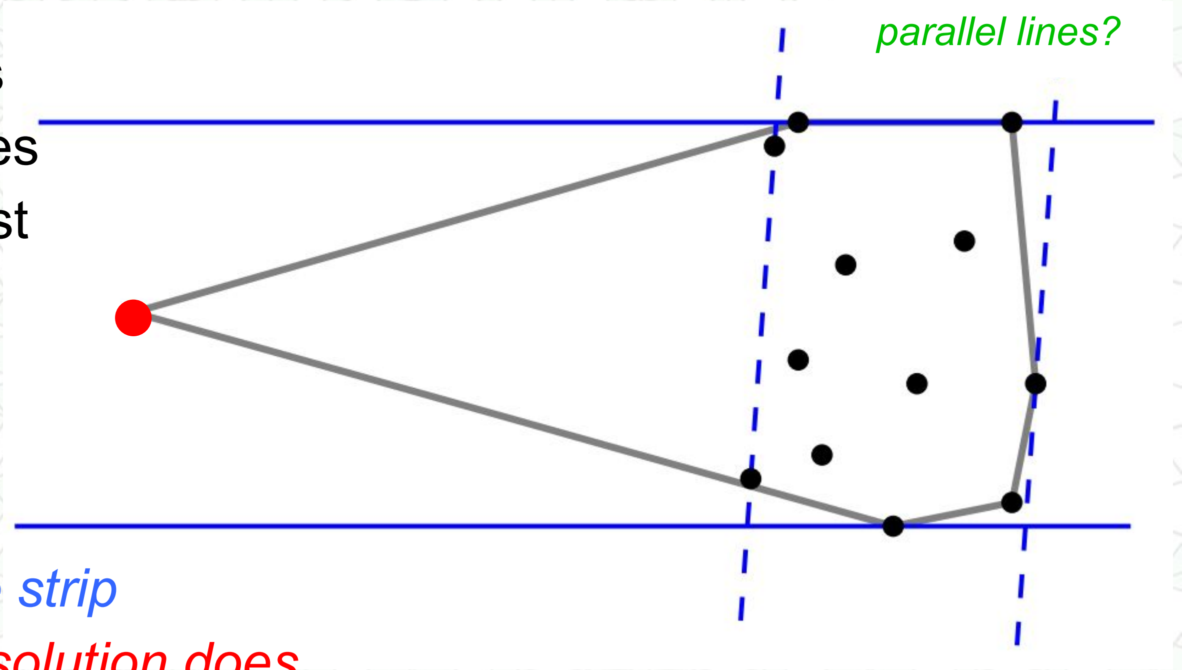
- *It is fast to test if a new point is contained in the strip*



Frank Staals,http://www.cs.uu.nl/docs/vakken/ga/2021/

A Counter-Example: Minimum Strip Width

- Input: A set of 2D points
- Output: Two parallel lines that define the narrowest strip that contains all of the input points.



- *It is fast to test if a new point is contained in the strip*
- *However, the previous solution does not help us find a new optimal solution*

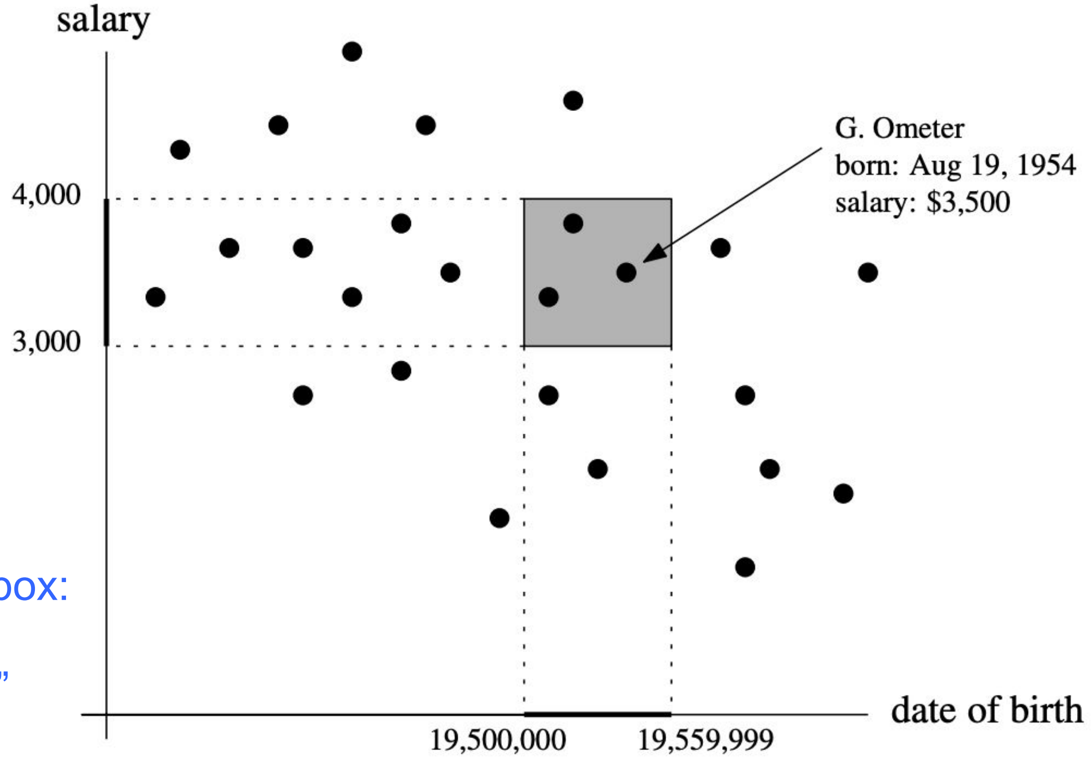
Frank Staals,http://www.cs.uu.nl/docs/vakken/ga/2021/

Outline for Today

- Homework…
- Last Time: Half-Space Intersections & Randomized Incremental Construction
- A Sample Quiz Problem?
- Motivation/Application: Smallest Bounding Sphere
  - Collision Detection, Ray Tracing, Robot Placement
- Brute Force Minimal Smallest Bounding Circle
- Bounding Circle by Center of Mass
- Incremental Construction of Smallest Bounding Circle
- Randomized Incremental Construction
- Next Time: Point Location & Orthogonal Range Searching

Motivating Application: 2D Database Queries

- Return all data points with *x* value in range $[x_0, x_1]$ and *y* value in range $[y_0, y_1]$



Find all values in an axis parallel box:
a "*rectangular range query*"
a.k.a. "*orthogonal range query*"

Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 5