COMPETITIVE QUERY STRATEGIES FOR MINIMISING THE PLY OF THE POTENTIAL LOCATIONS OF MOVING POINTS

Will Evans
David Kirkpatrick
Frank Staals
Maarten Löffler

A BRIEF ELUCIDATION OF THE TITLE

“MOVING POINTS”

“MOVING POINTS”

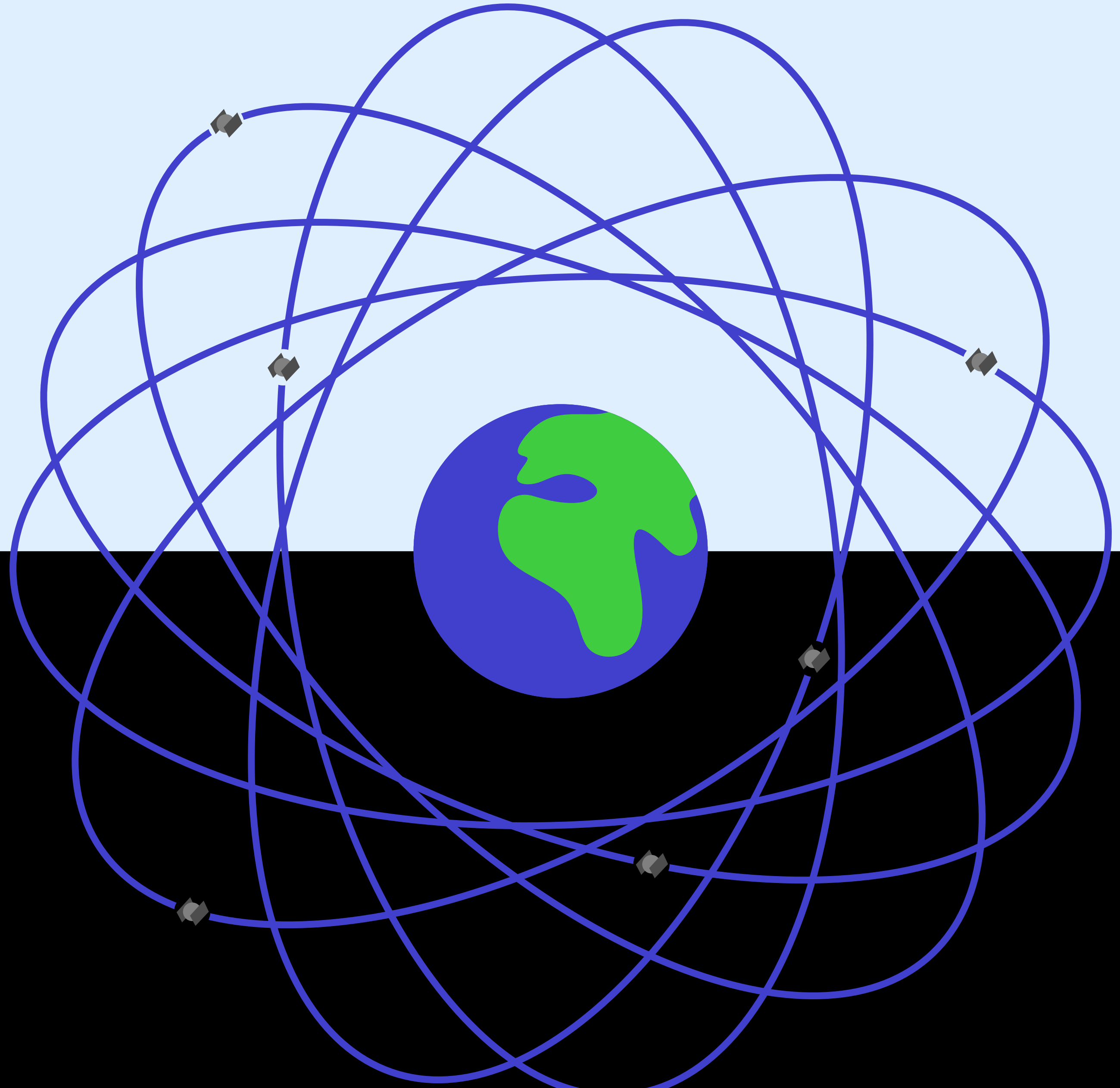
- Location-aware mobile devices

“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems

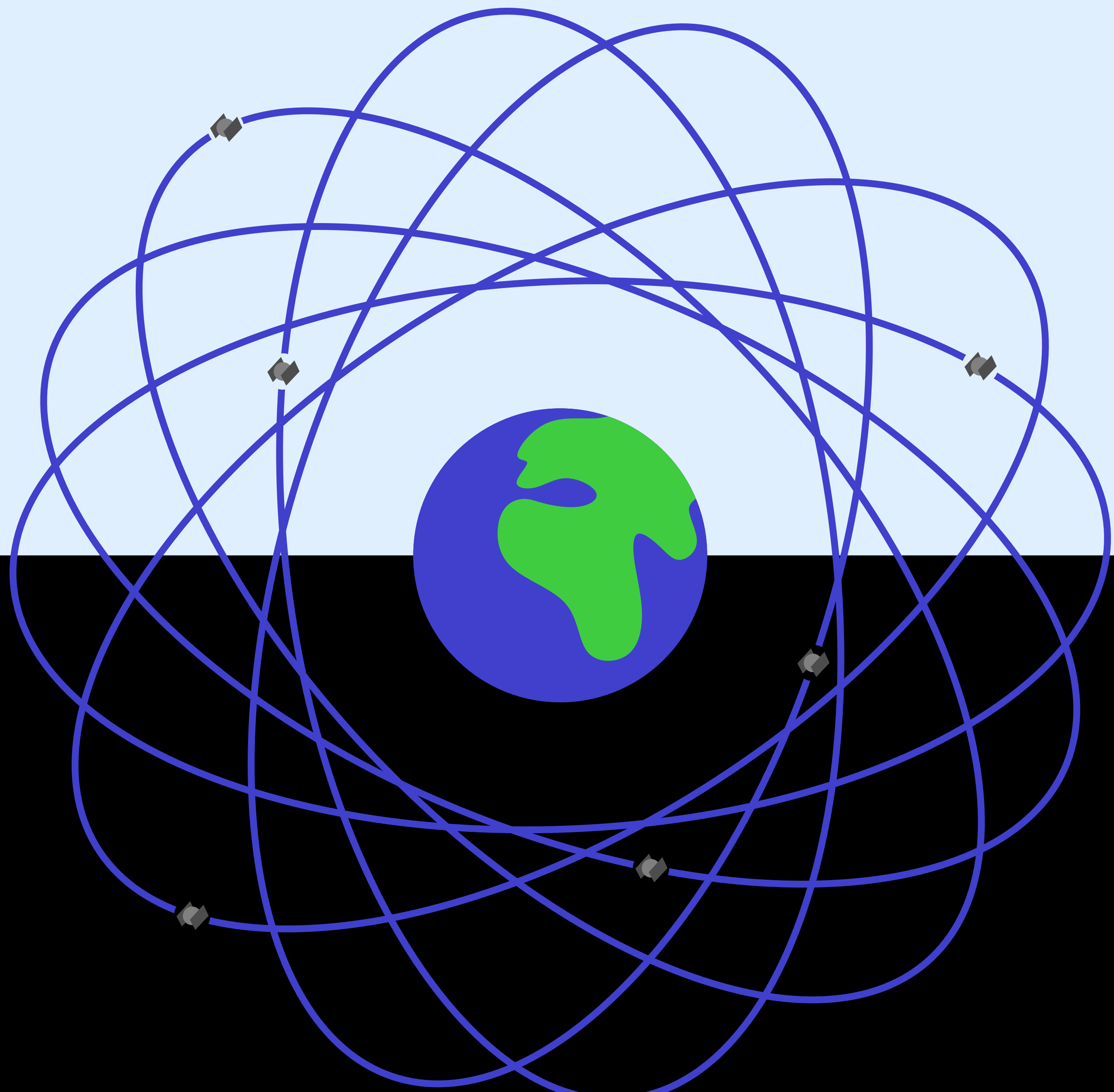
“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems



"MOVING POINTS"

- Location-aware mobile devices
  - GPS systems
  - Smart phones

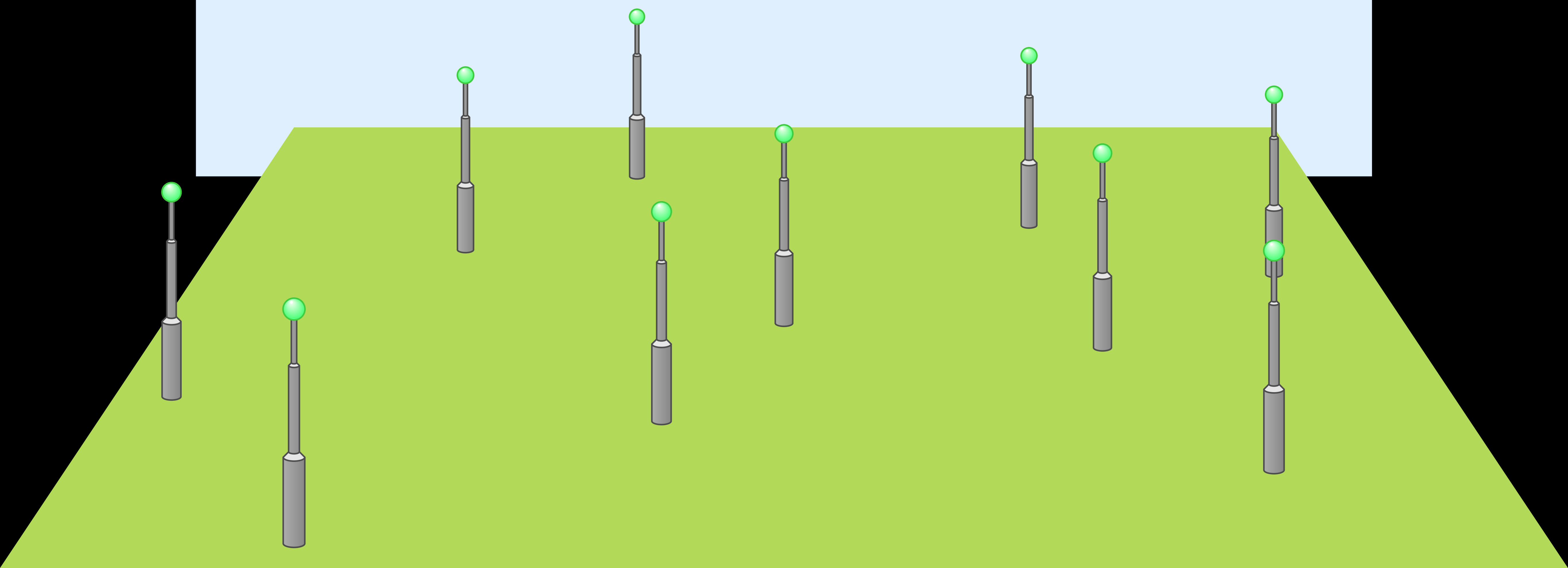


“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones

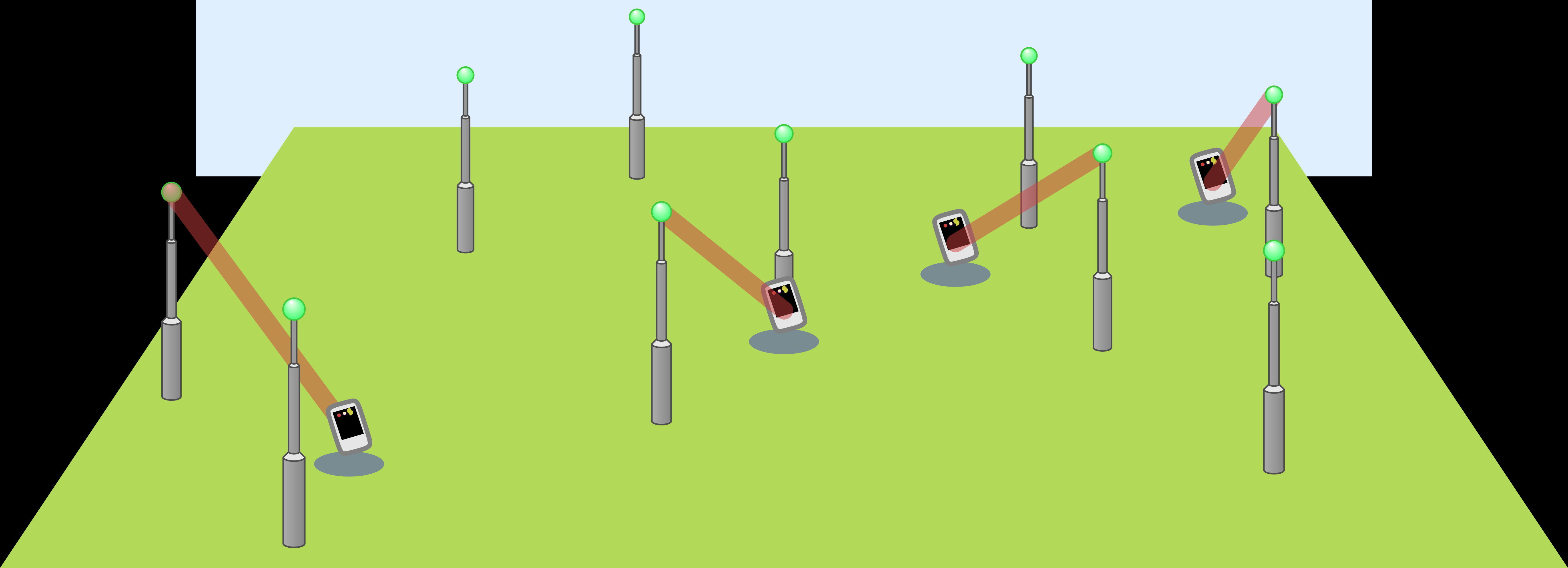
"MOVING POINTS"

- Location-aware mobile devices
  - GPS systems
  - Smart phones



“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones

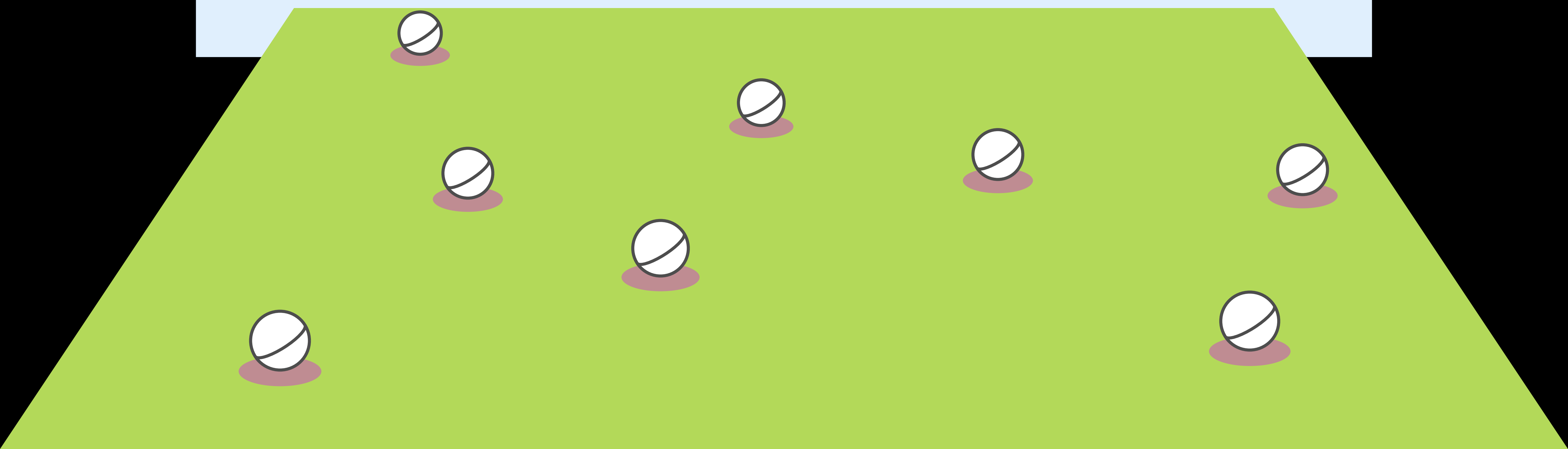


“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks

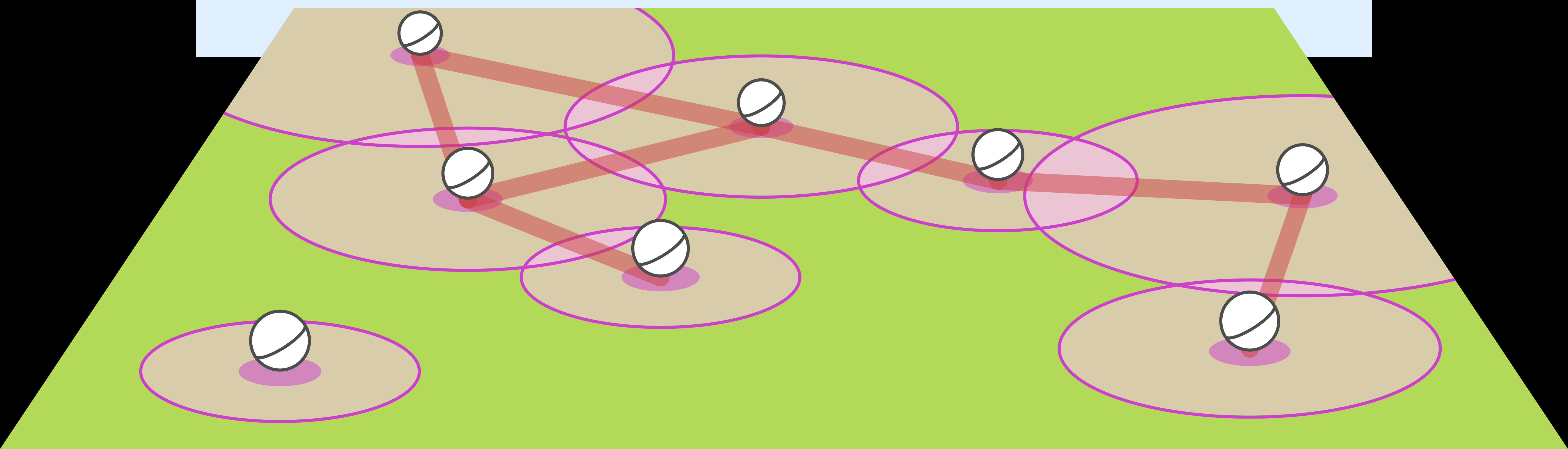
“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks



“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks

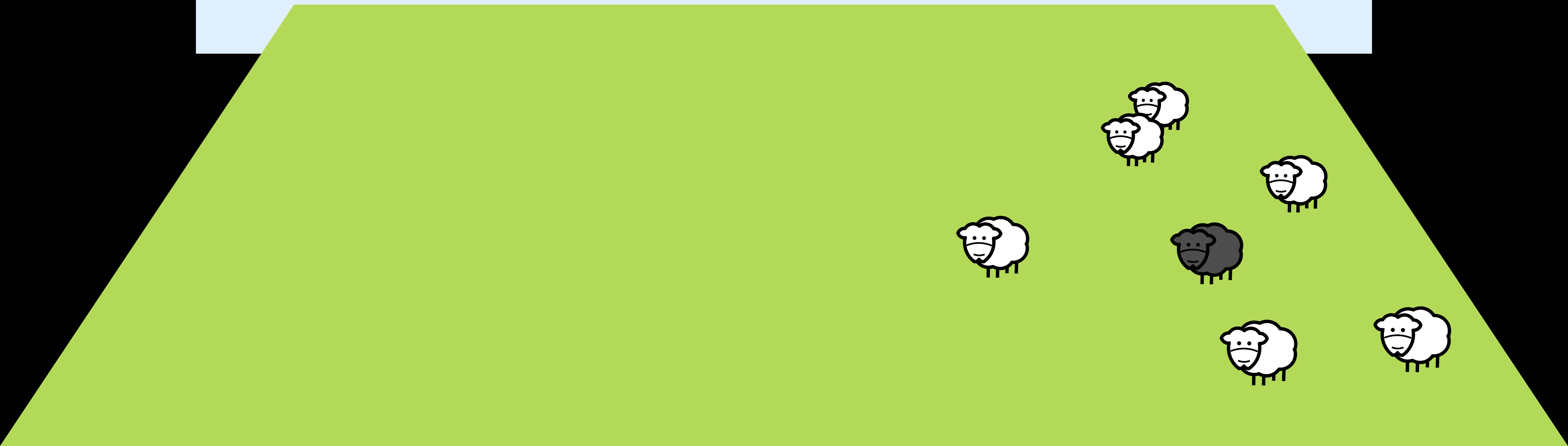


“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking

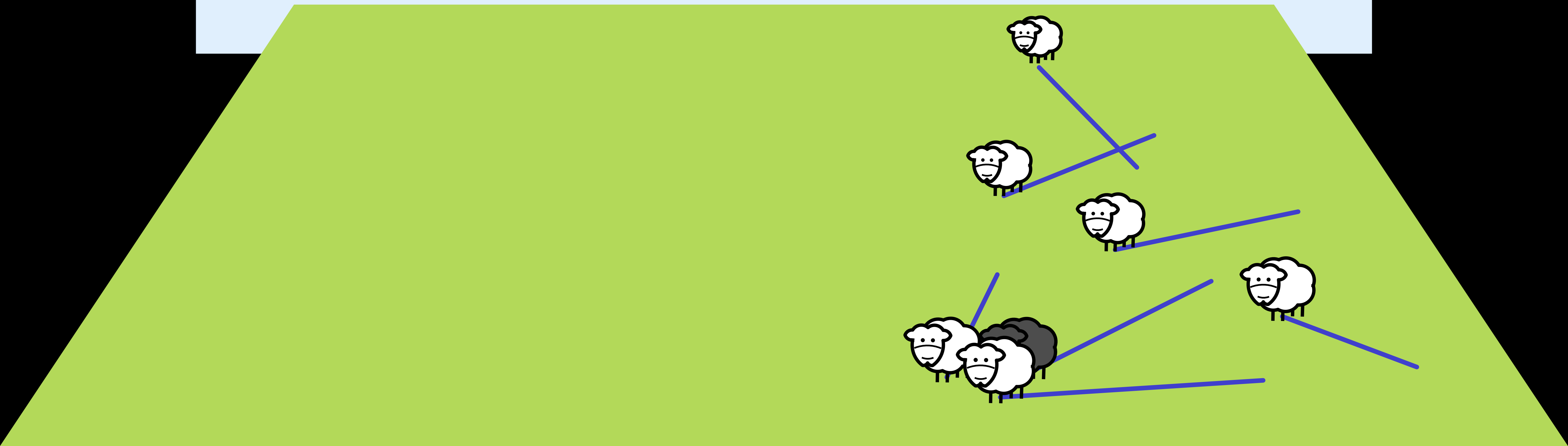
“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking



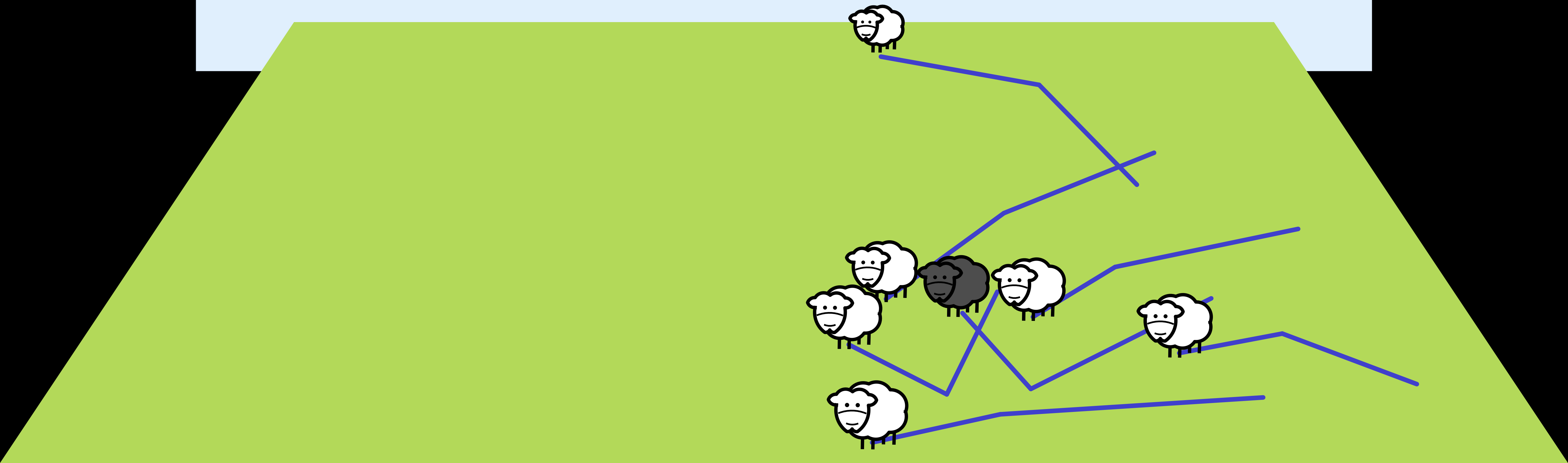
“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking



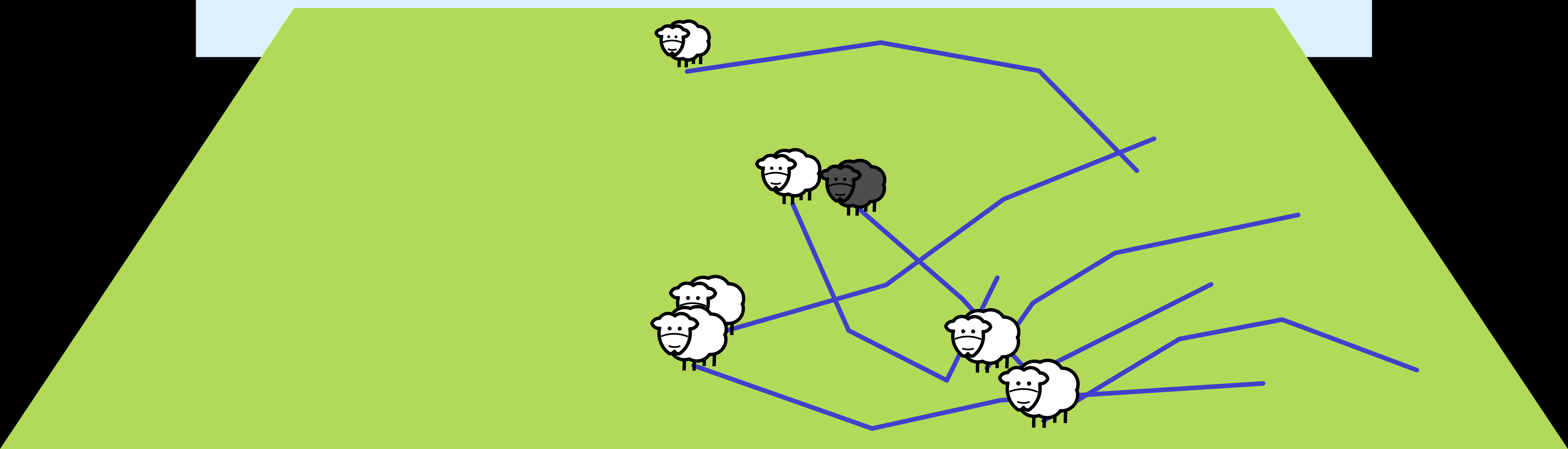
“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking



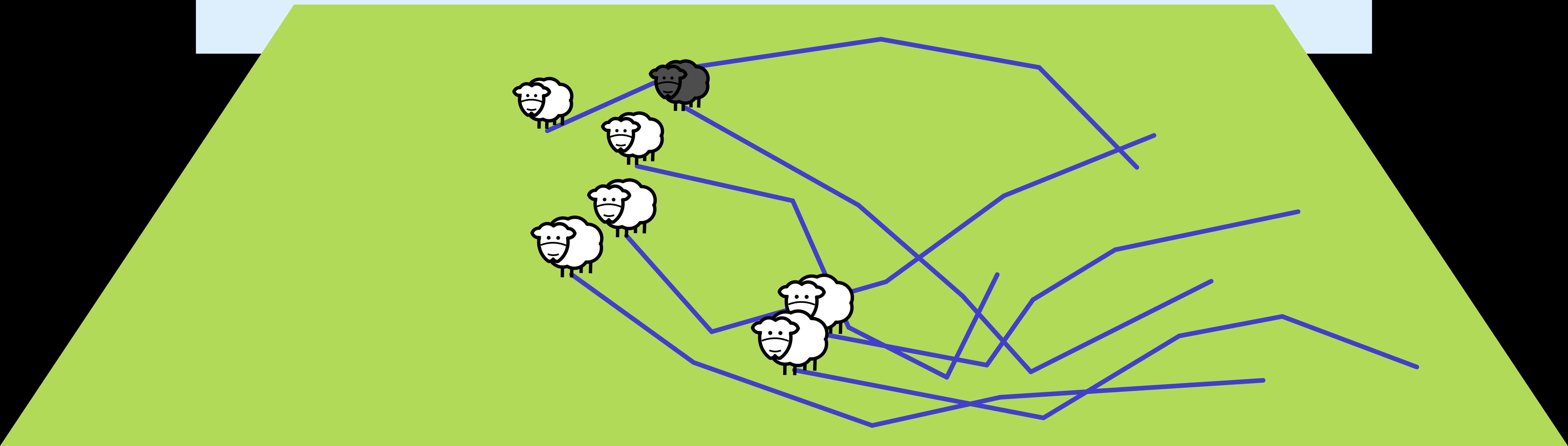
“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking



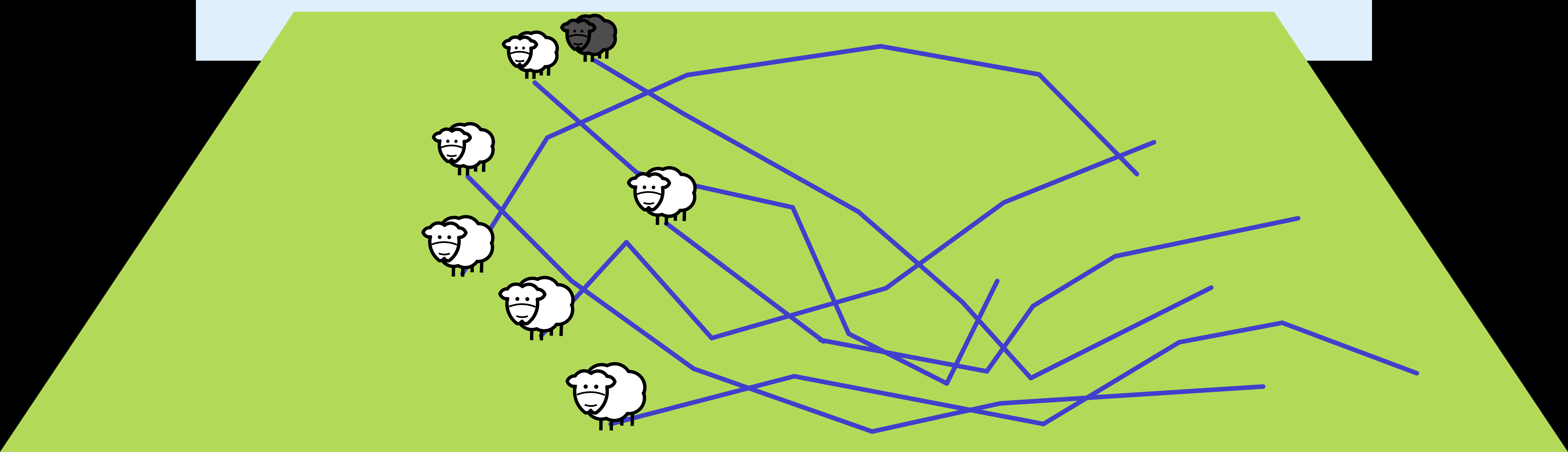
"MOVING POINTS"

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking



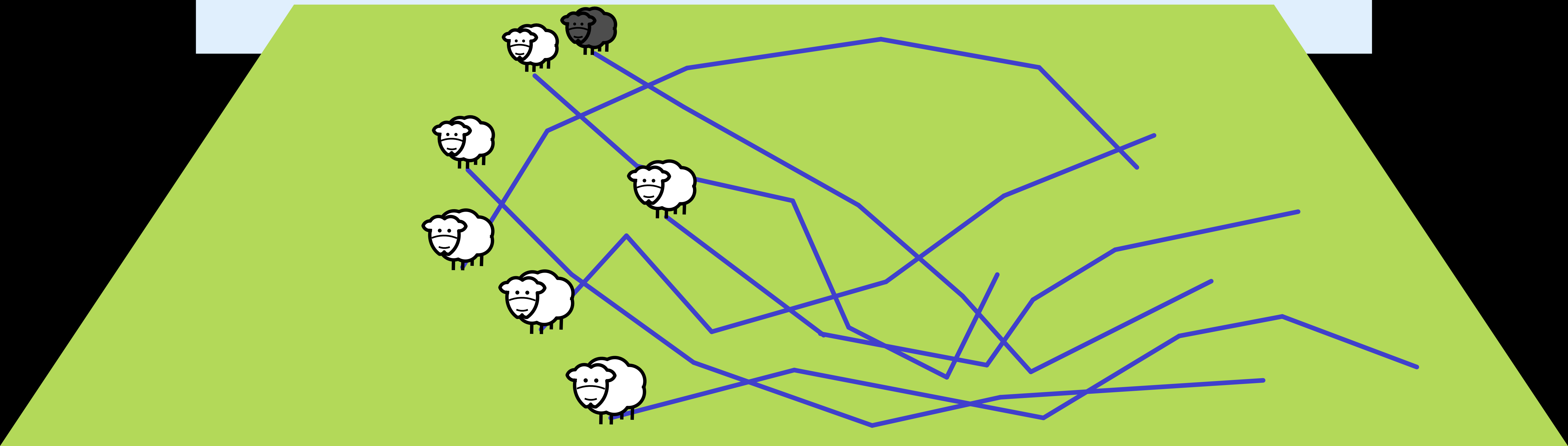
“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking



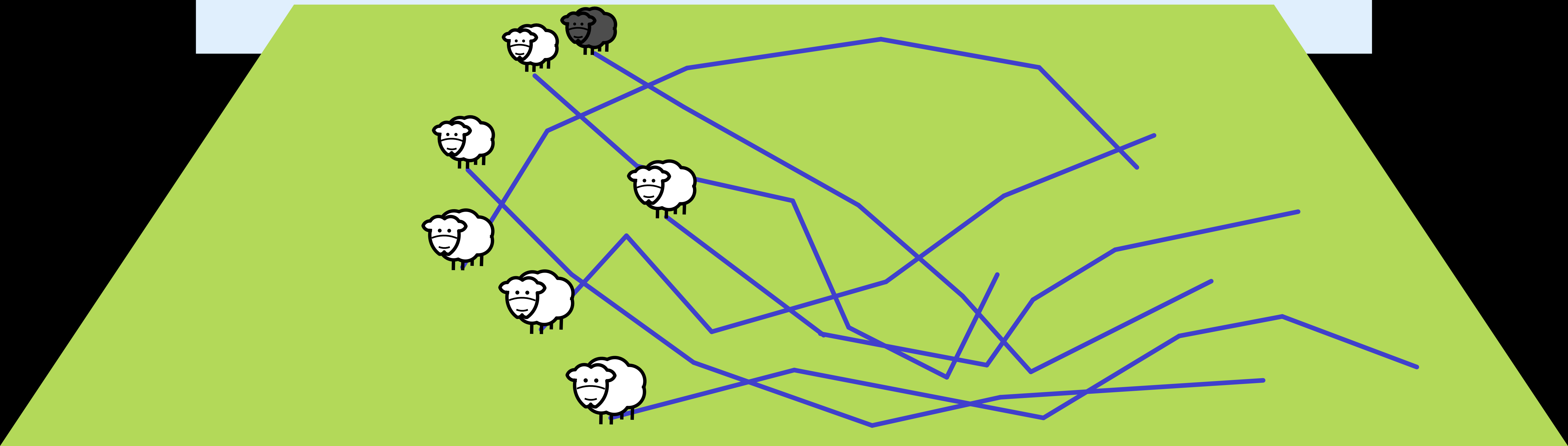
"MOVING POINTS"

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking
- New geometric data sets



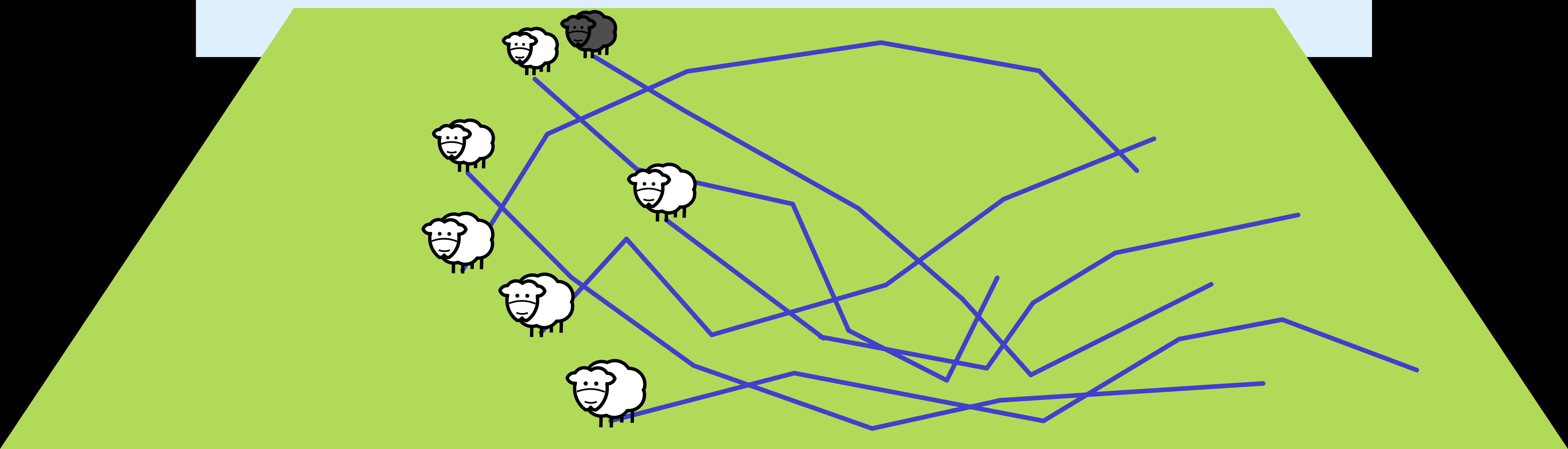
"MOVING POINTS"

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking
- New geometric data sets
  - Often very large



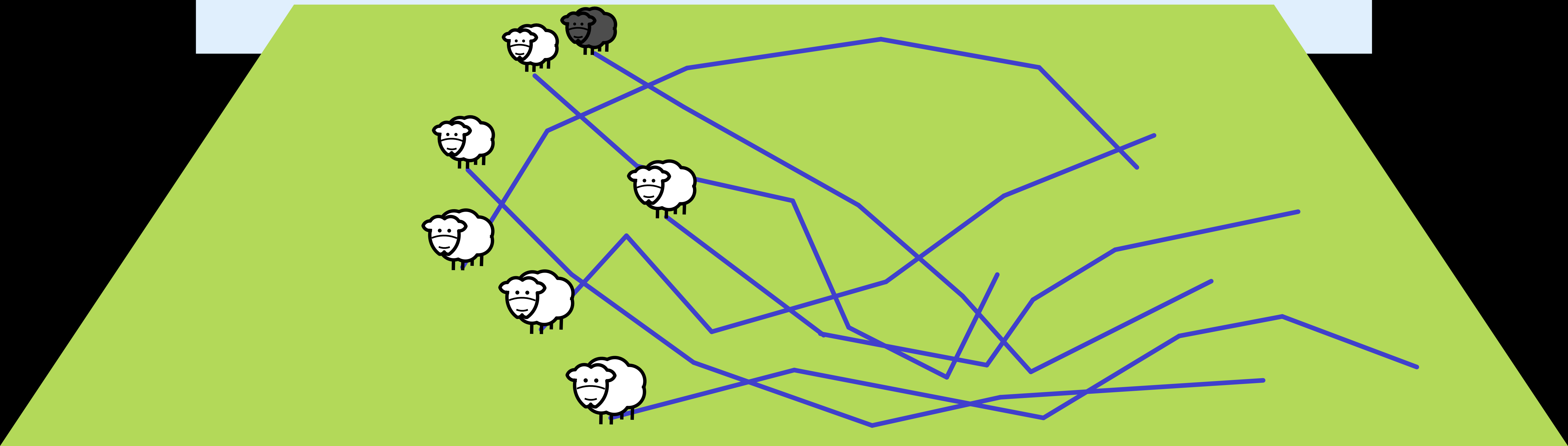
"MOVING POINTS"

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking
- New geometric data sets
  - Often very large
  - Imprecise in nature



“MOVING POINTS”

- Location-aware mobile devices
  - GPS systems
  - Smart phones
  - Wireless sensor networks
  - Flock tracking
- New geometric data sets
  - Often very large
  - Imprecise in nature
  - Dynamic / unpredictable



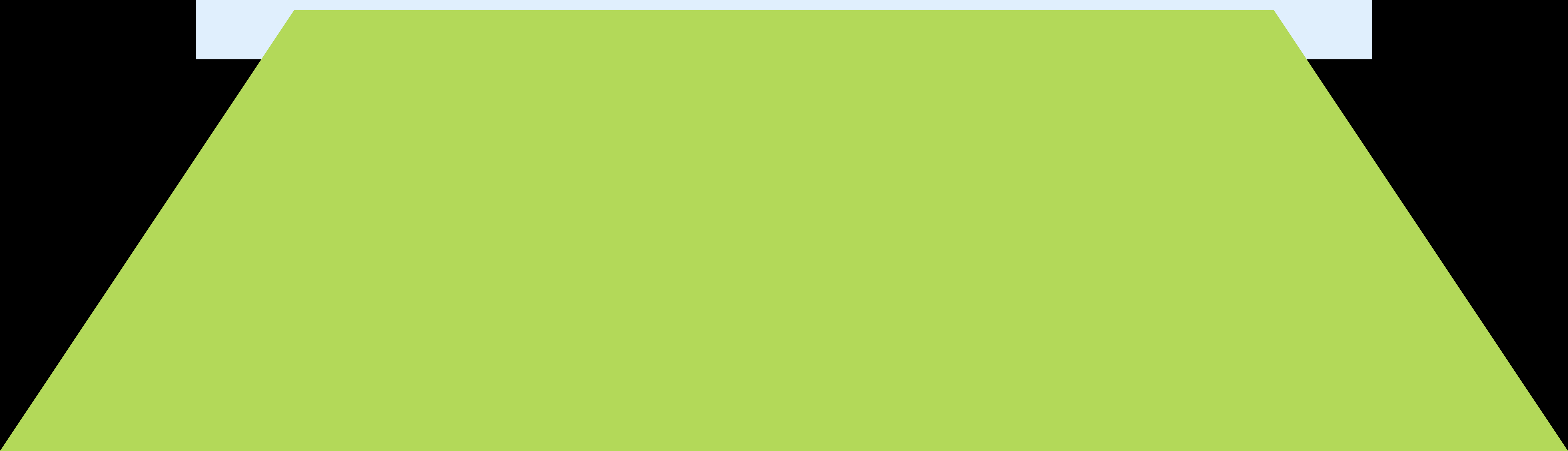
"POTENTIAL LOCATIONS"

“POTENTIAL LOCATIONS”

- Motion causes imprecision

"POTENTIAL LOCATIONS"

- Motion causes imprecision
  - Suppose we have a moving point p



"POTENTIAL LOCATIONS"

- Motion causes imprecision
  - Suppose we have a moving point p

"POTENTIAL LOCATIONS"

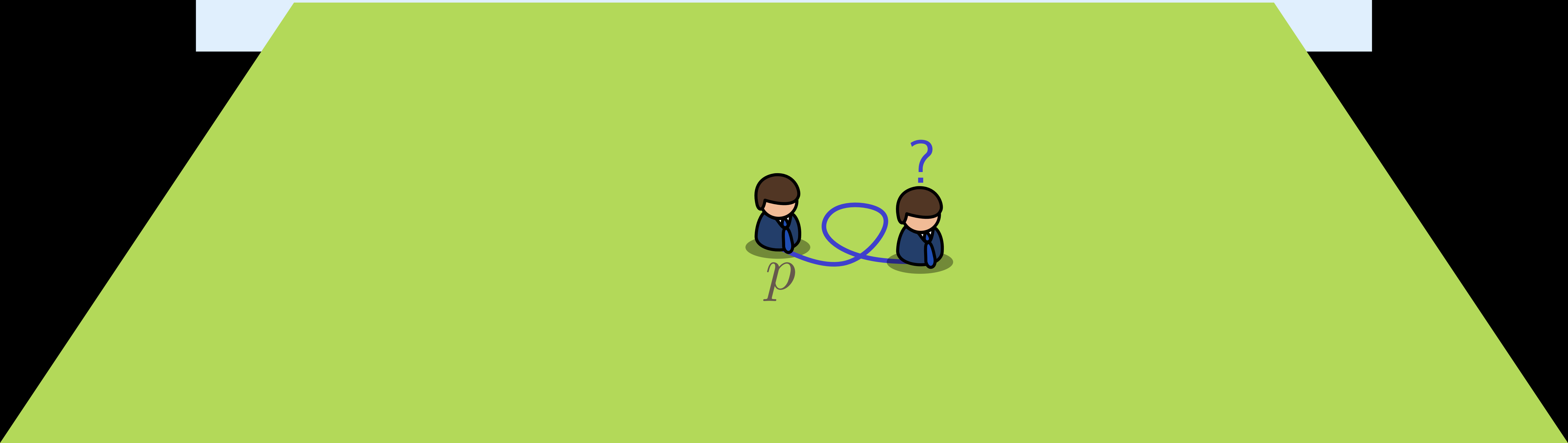
- Motion causes imprecision
  - Suppose we have a moving point p
  - We know an upper bound on its speed

"POTENTIAL LOCATIONS"

- Motion causes imprecision
  - Suppose we have a moving point p
  - We know an upper bound on its speed
  - In 1 time step p can move at most d

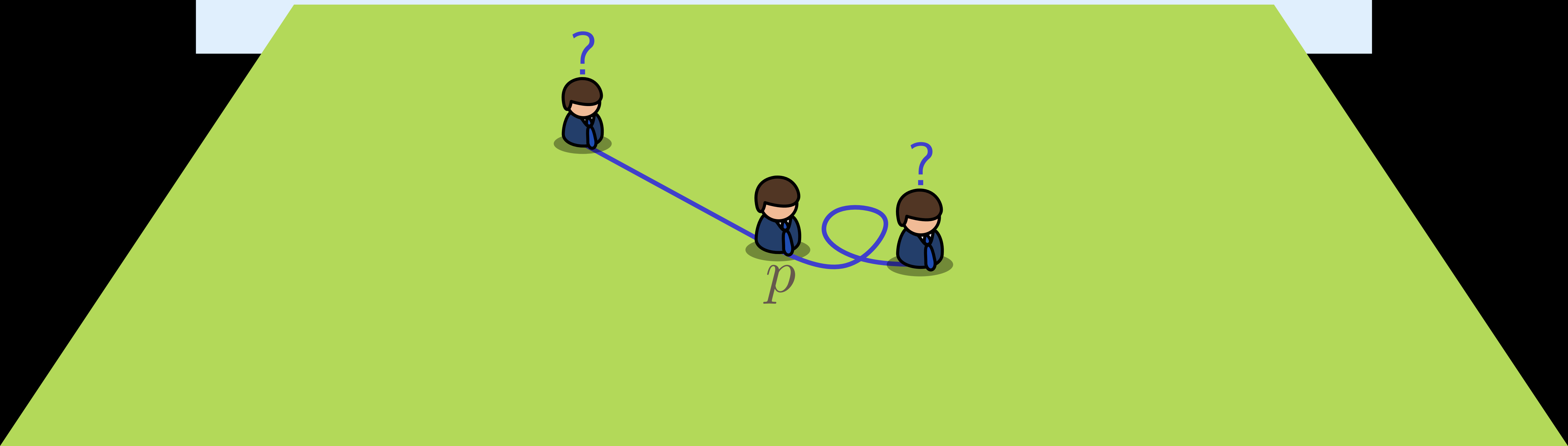
"POTENTIAL LOCATIONS"

- Motion causes imprecision
  - Suppose we have a moving point p
  - We know an upper bound on its speed
  - In 1 time step p can move at most d



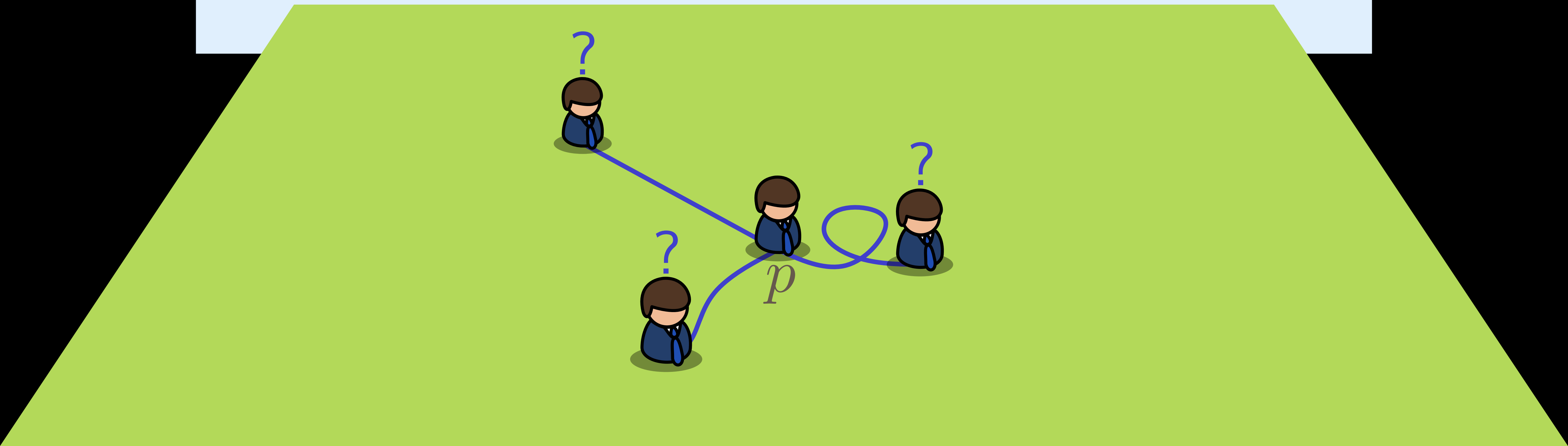
"POTENTIAL LOCATIONS"

- Motion causes imprecision
  - Suppose we have a moving point p
  - We know an upper bound on its speed
  - In 1 time step p can move at most d



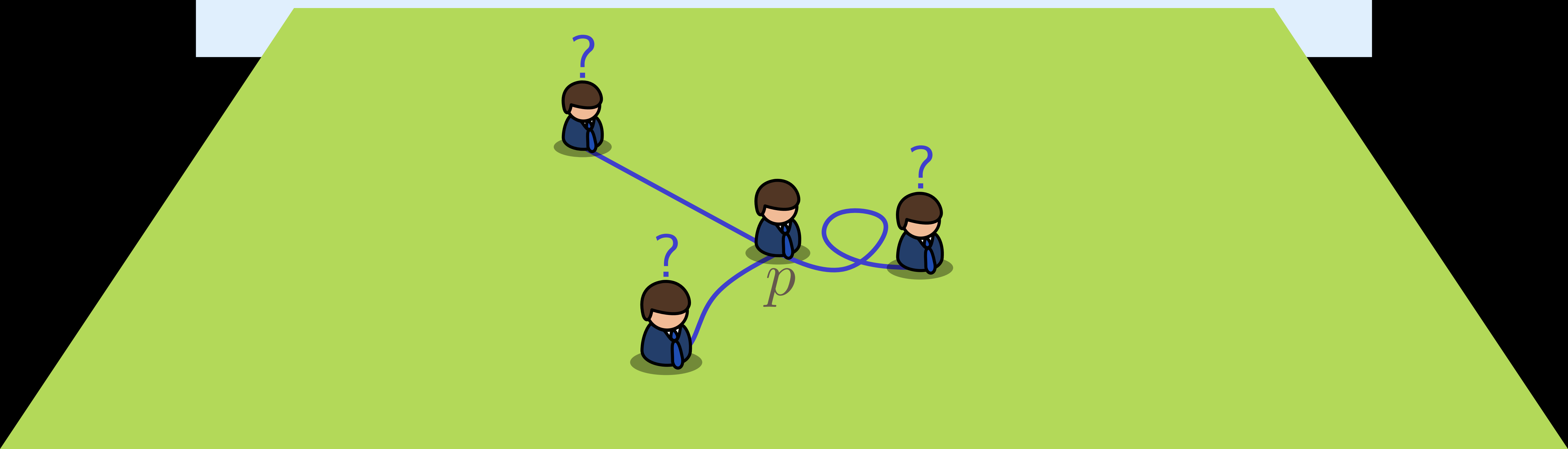
“POTENTIAL LOCATIONS”

- Motion causes imprecision
  - Suppose we have a moving point p
  - We know an upper bound on its speed
  - In 1 time step p can move at most d



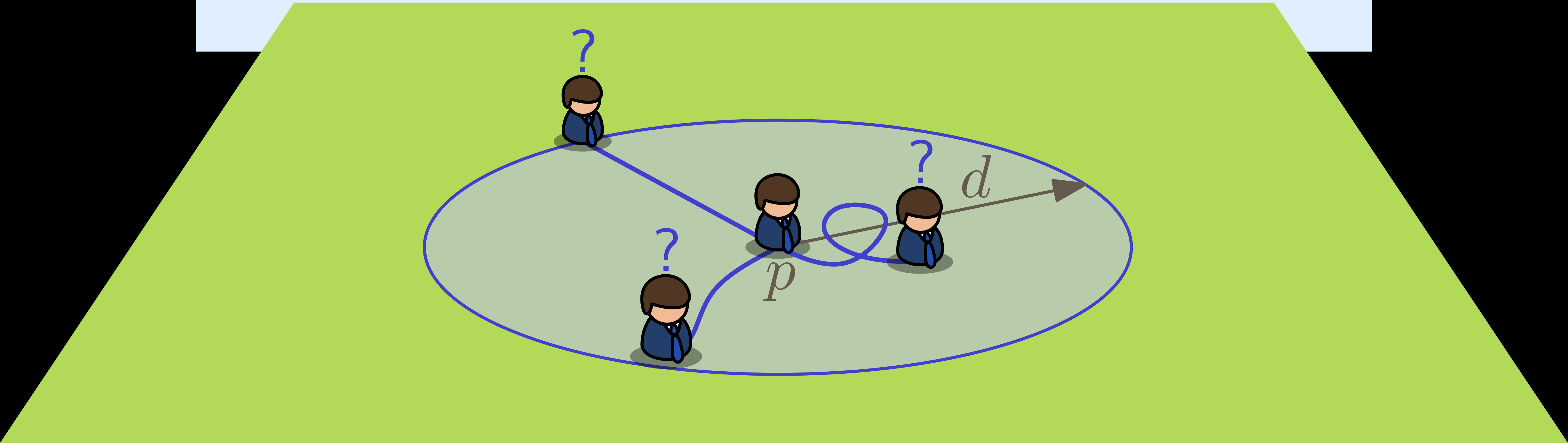
“POTENTIAL LOCATIONS”

- Motion causes imprecision
  - Suppose we have a moving point p
  - We know an upper bound on its speed
  - In 1 time step p can move at most d
  - We model the *potential locations* of p in the next time step by a disk of radius d



"POTENTIAL LOCATIONS"

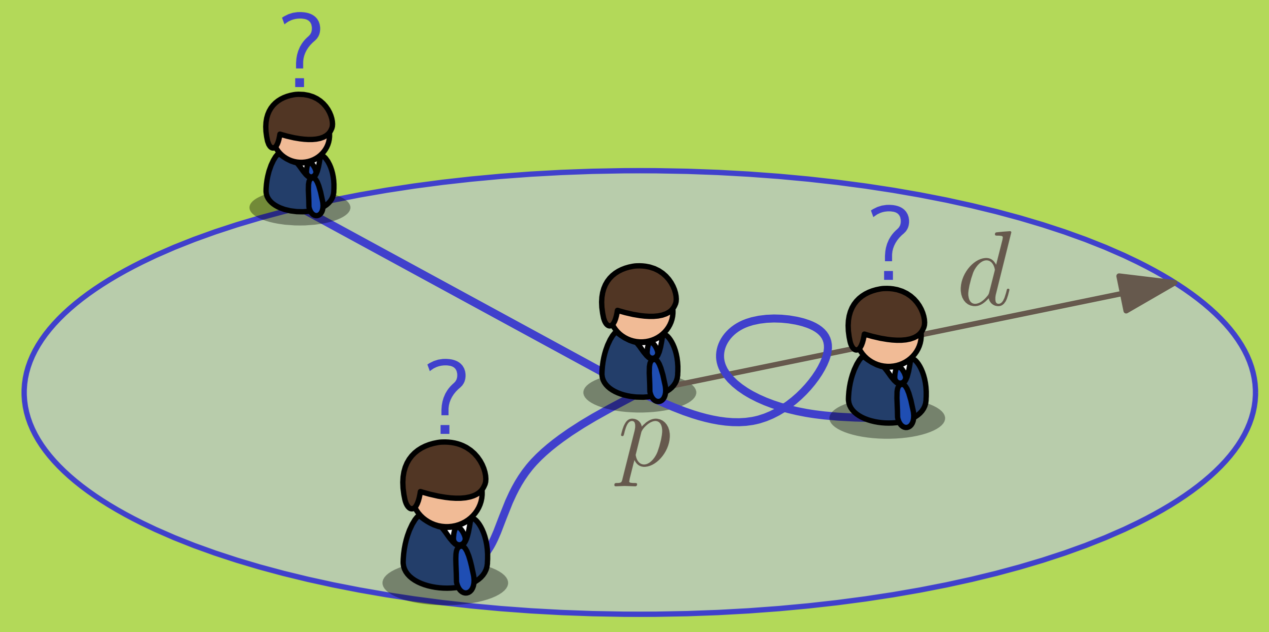
- Motion causes imprecision
  - Suppose we have a moving point p
  - We know an upper bound on its speed
  - In 1 time step p can move at most d
  - We model the *potential locations* of p in the next time step by a disk of radius d



“POTENTIAL LOCATIONS”

- Motion causes imprecision
  - Suppose we have a moving point p
  - We know an upper bound on its speed
  - In 1 time step p can move at most d
  - We model the *potential locations* of p in the next time step by a disk of radius d

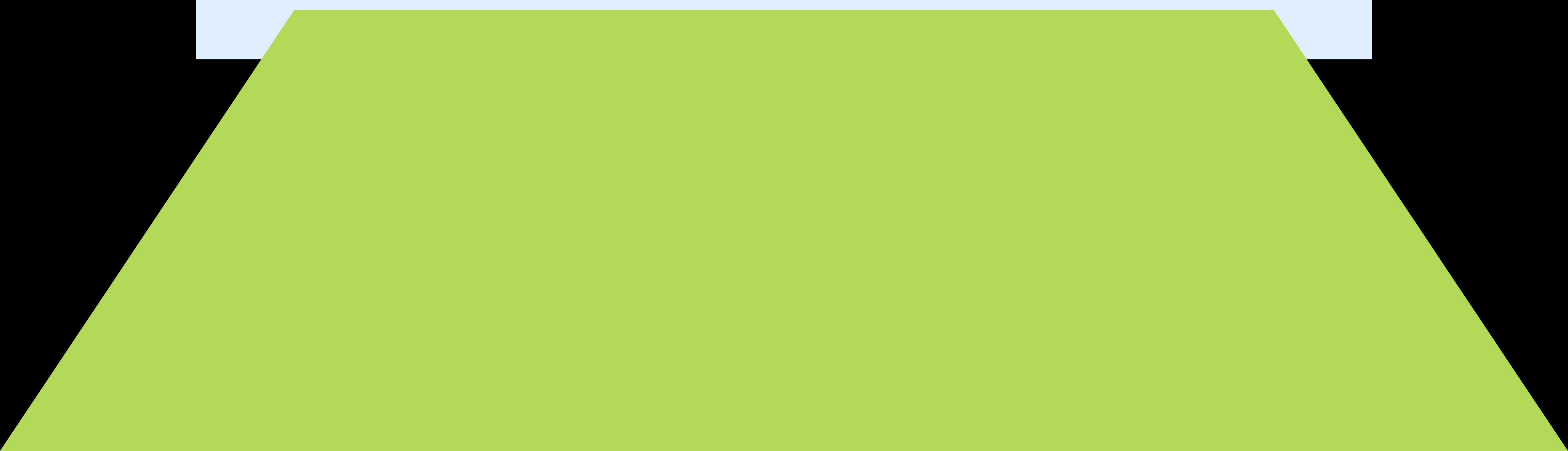
OBSERVATION: *If we knew where p was t time ago, we know it is now somewhere in a disk of radius td*



"QUERY"

“QUERY”

- Consider n moving points



“QUERY”

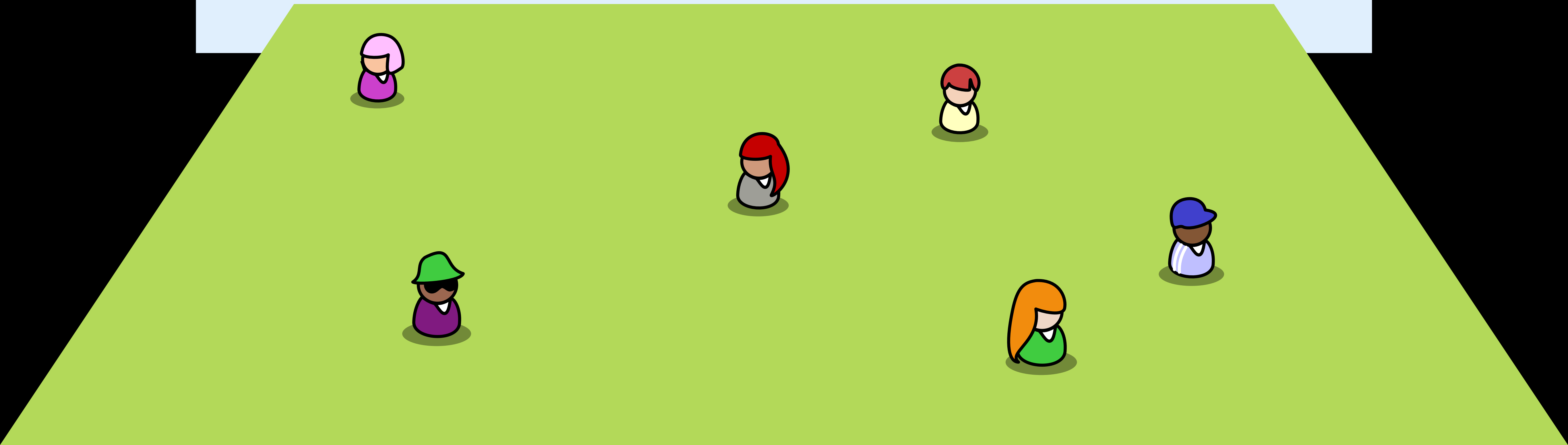
- Consider n moving points

“QUERY”

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i

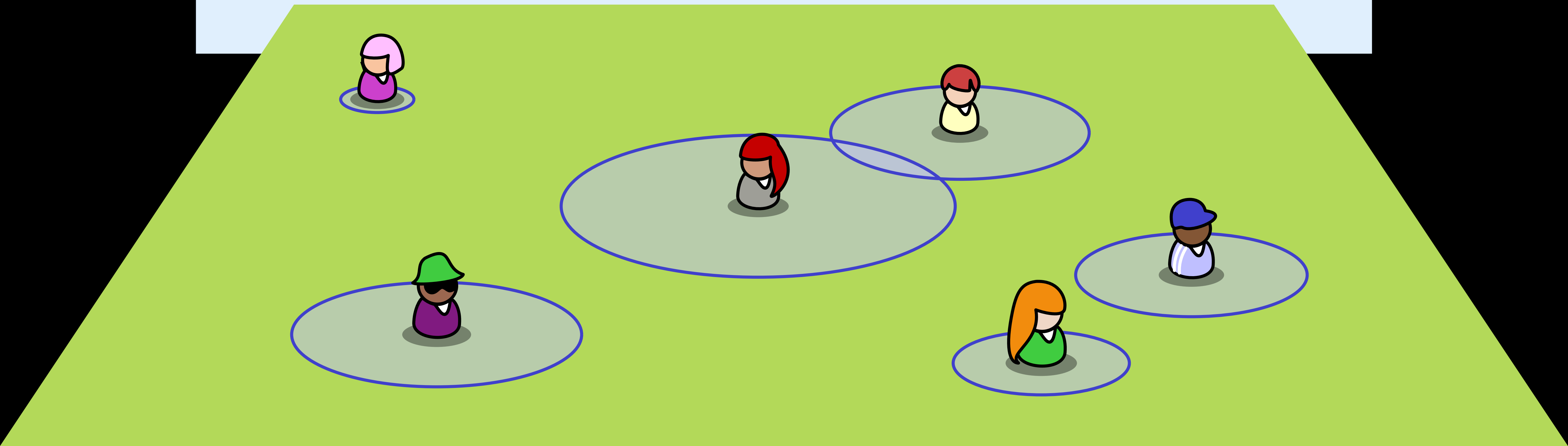
"QUERY"

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i



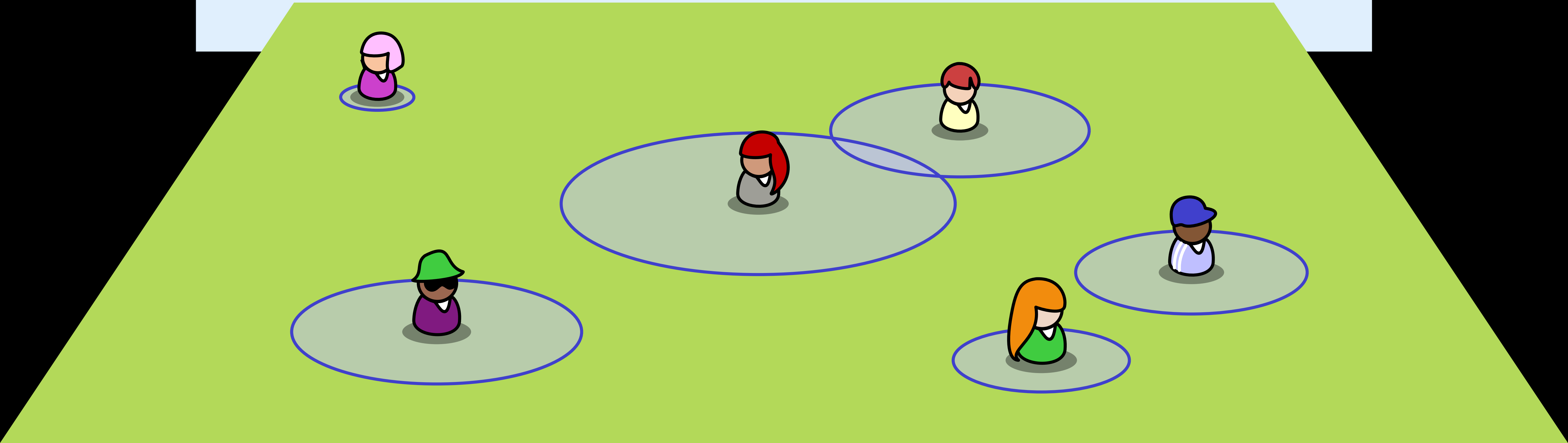
“QUERY”

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i



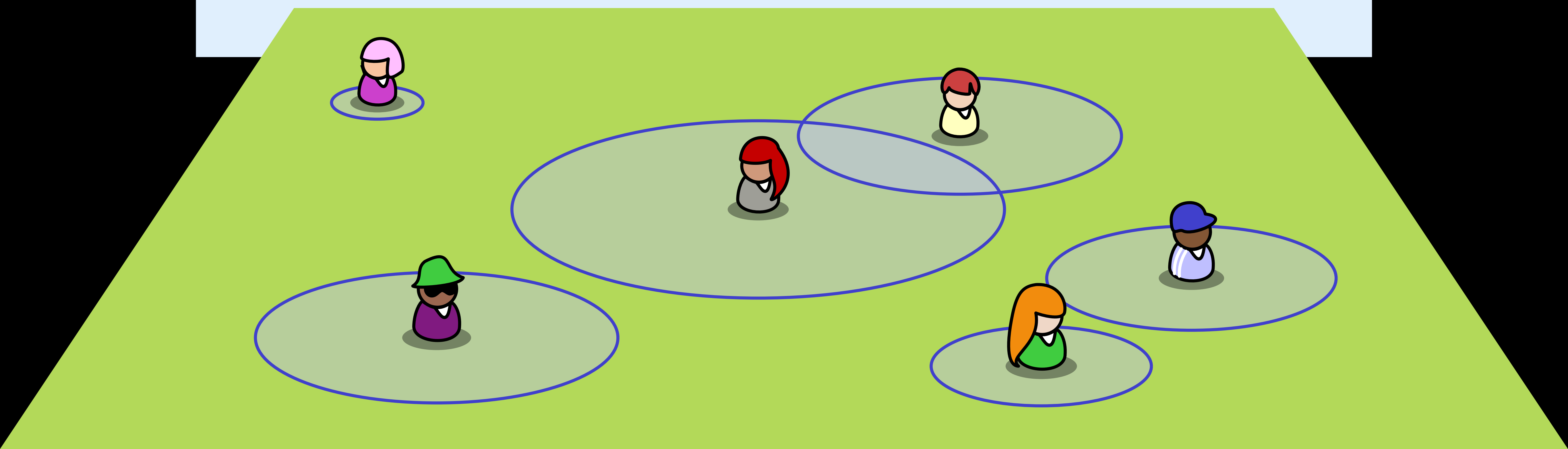
“QUERY”

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i
  - Each unit of time, all regions grow by d



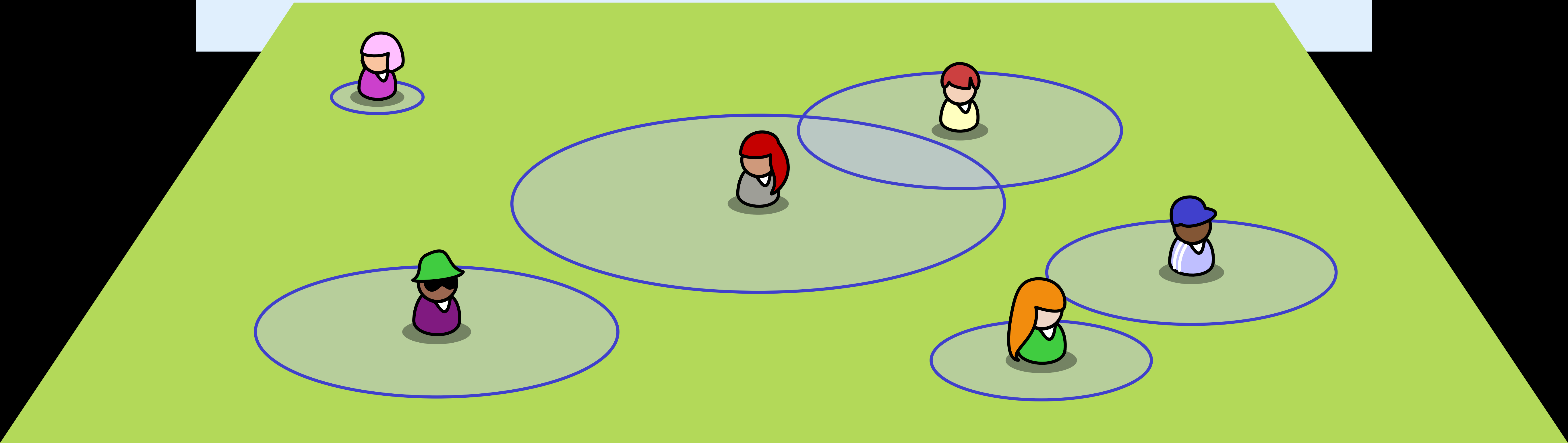
“QUERY”

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i
  - Each unit of time, all regions grow by d



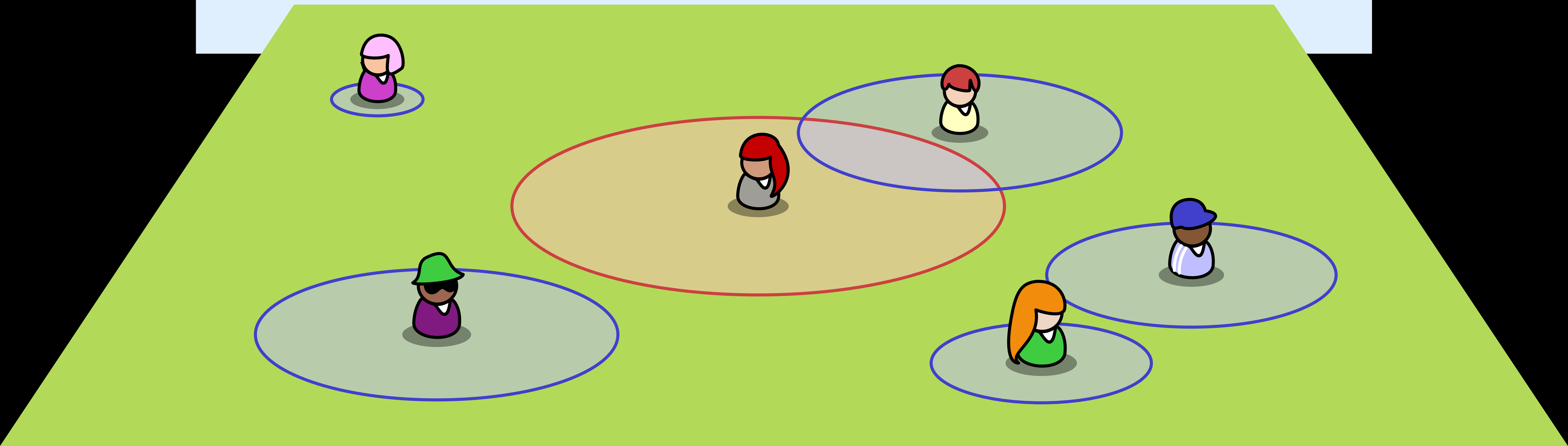
“QUERY”

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i
  - Each unit of time, all regions grow by d
- We can *query* a point to update it



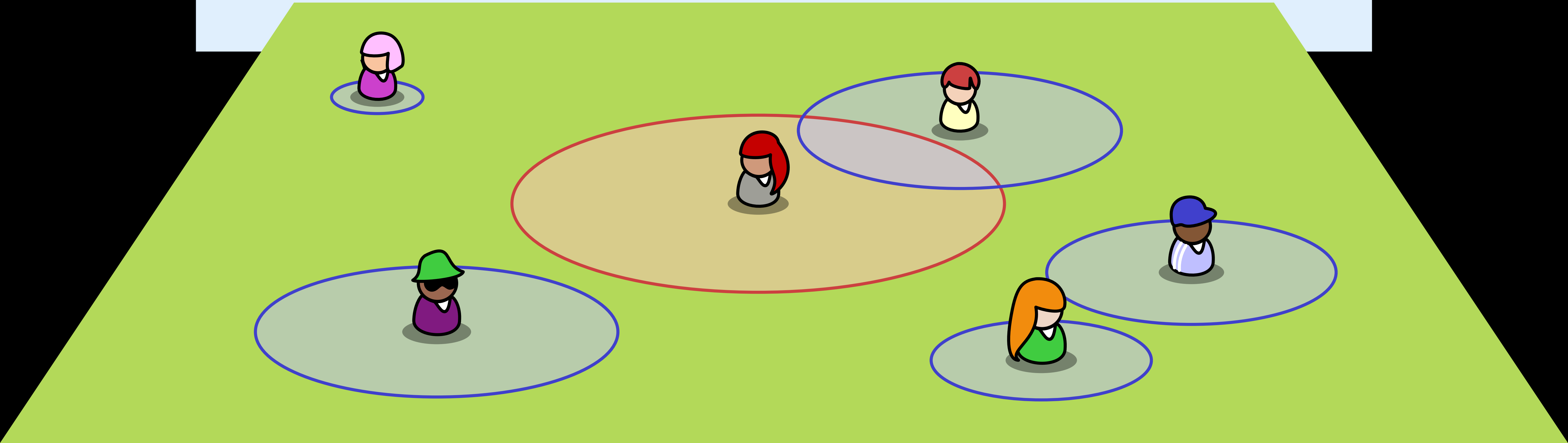
"QUERY"

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i
  - Each unit of time, all regions grow by d
- We can *query* a point to update it



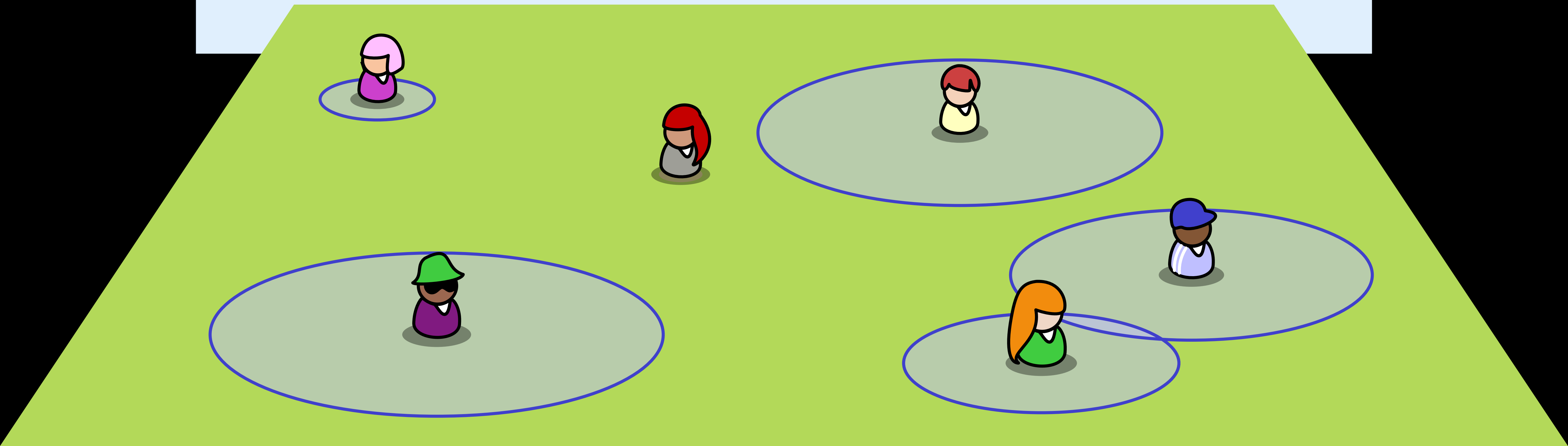
“QUERY”

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i
  - Each unit of time, all regions grow by d
- We can *query* a point to update it
  - Takes 1 unit of time to execute



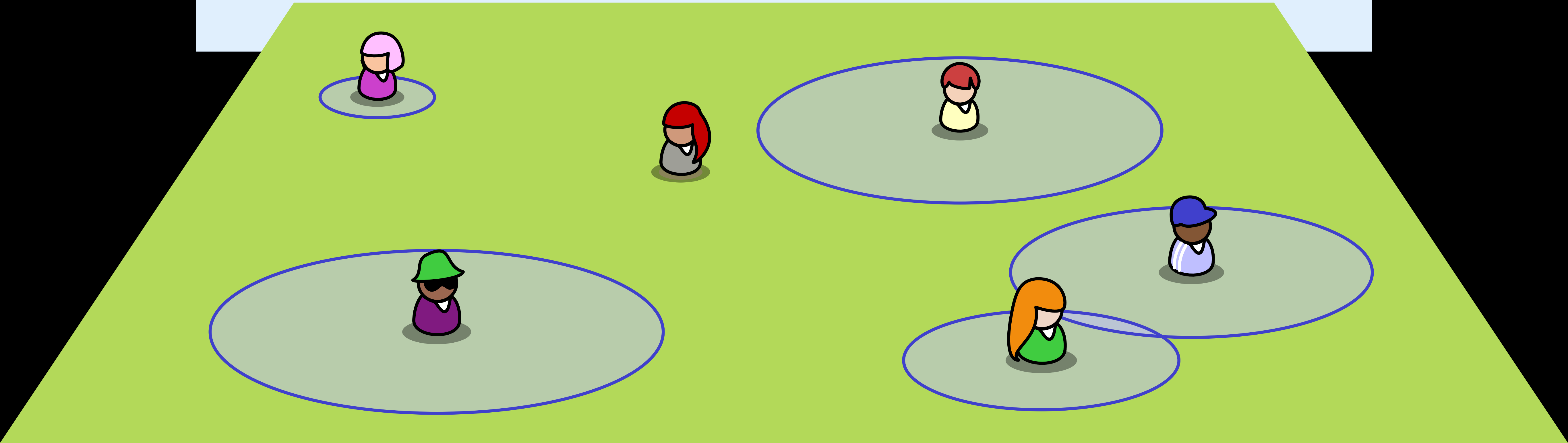
"QUERY"

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i
  - Each unit of time, all regions grow by d
- We can *query* a point to update it
  - Takes 1 unit of time to execute



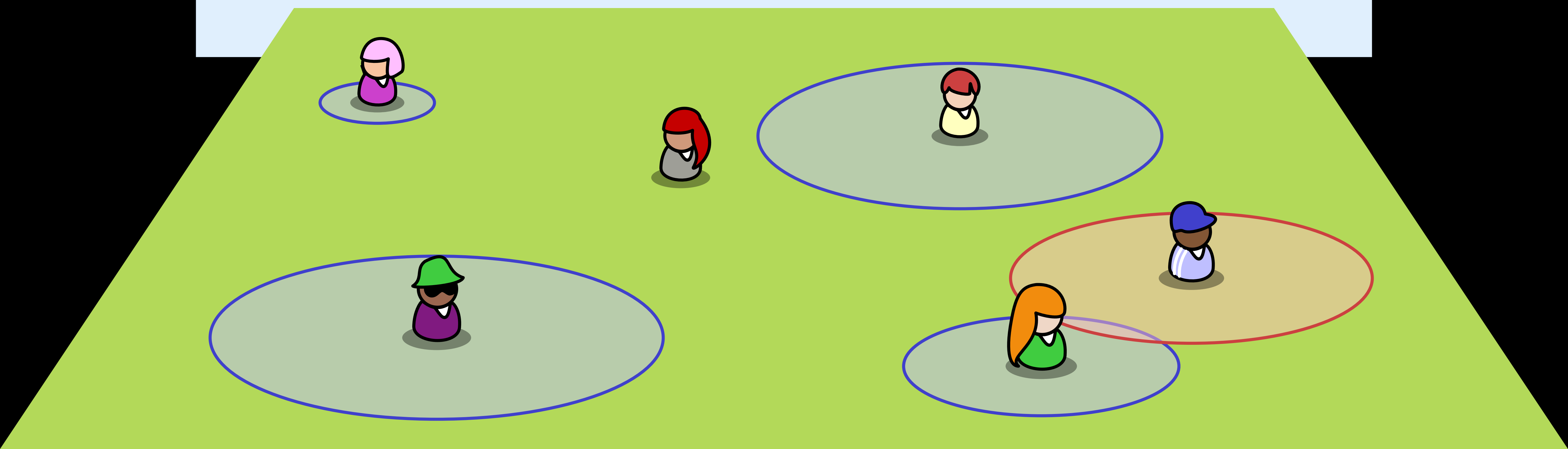
"QUERY"

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i
  - Each unit of time, all regions grow by d
- We can *query* a point to update it
  - Takes 1 unit of time to execute
  - After a query, one region collapses to a point, but all other regions grow



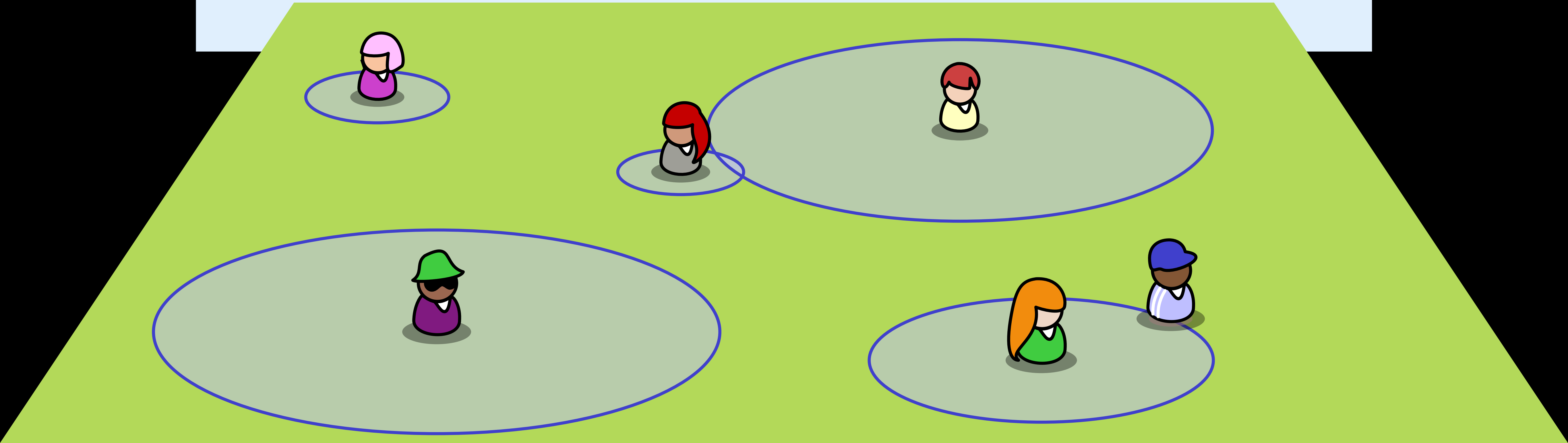
"QUERY"

- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i
  - Each unit of time, all regions grow by d
- We can *query* a point to update it
  - Takes 1 unit of time to execute
  - After a query, one region collapses to a point, but all other regions grow



“QUERY”

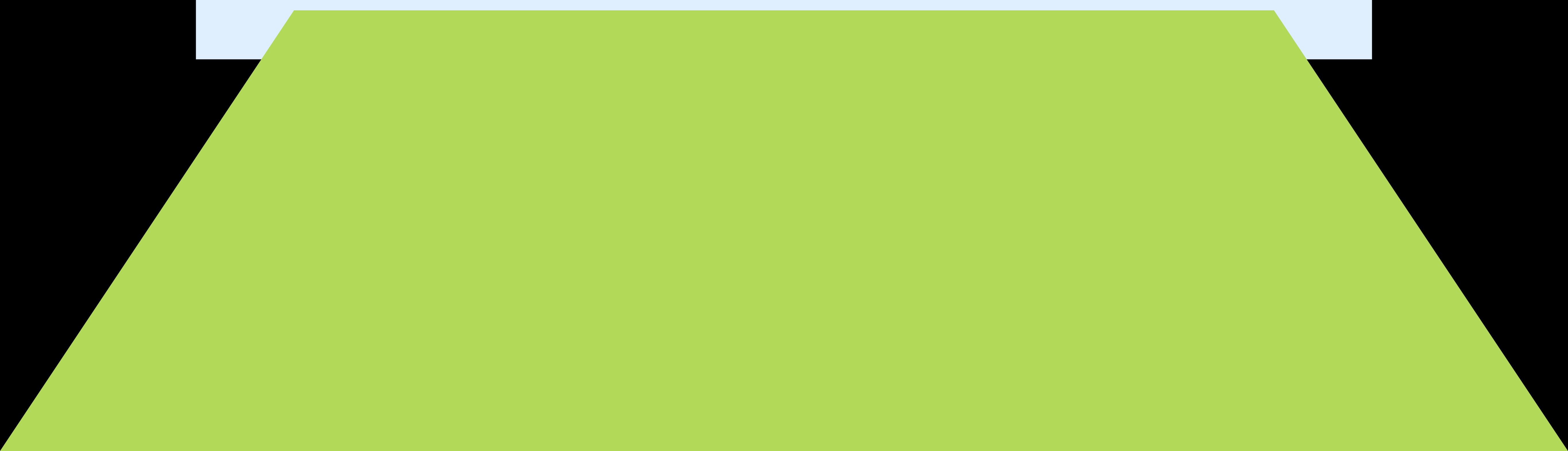
- Consider n moving points
  - Suppose that for each point we know a *location* p_i and an associated *time stamp* t_i
  - Then the current potential location region of each point is a disk centered at p_i
  - Each unit of time, all regions grow by d
- We can *query* a point to update it
  - Takes 1 unit of time to execute
  - After a query, one region collapses to a point, but all other regions grow



“PLY”

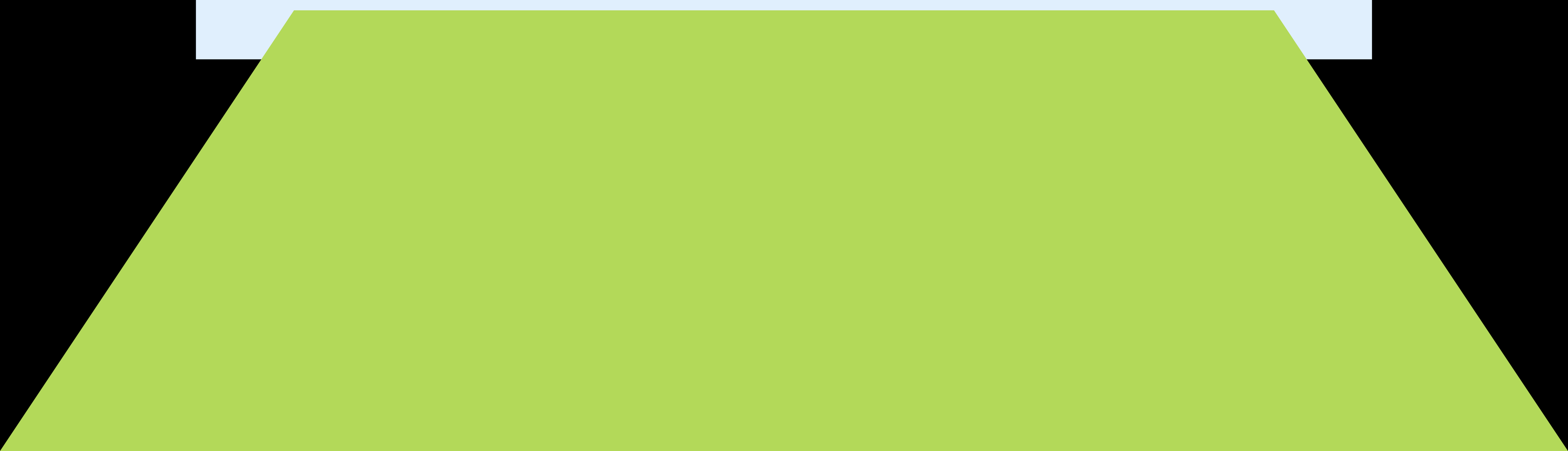
“PLY”

- Ply of a point



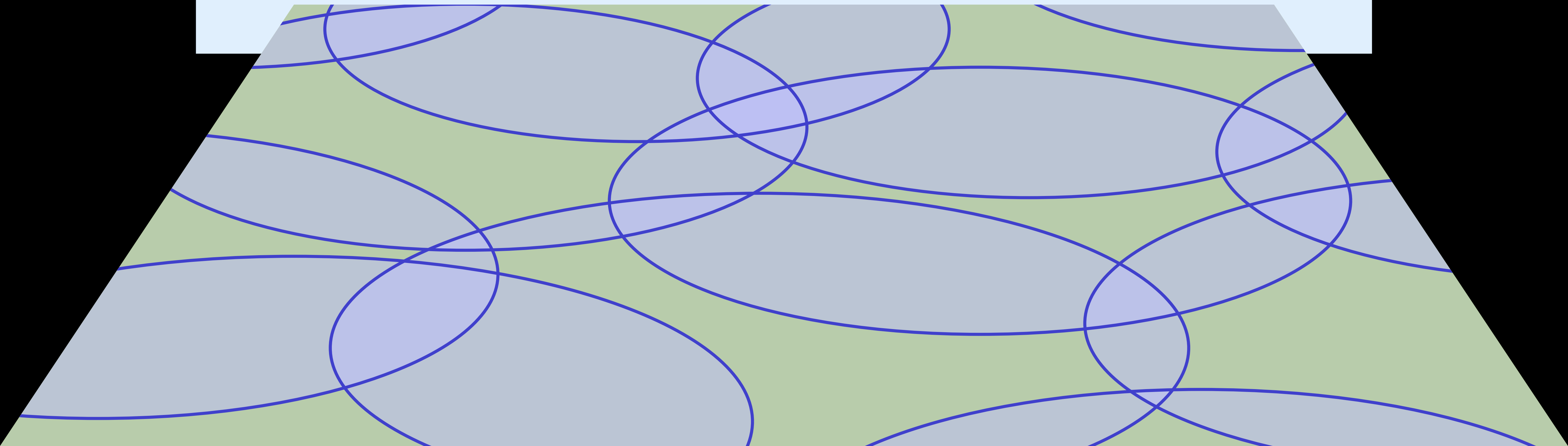
“PLY”

- Ply of a point
  - Given a set of regions



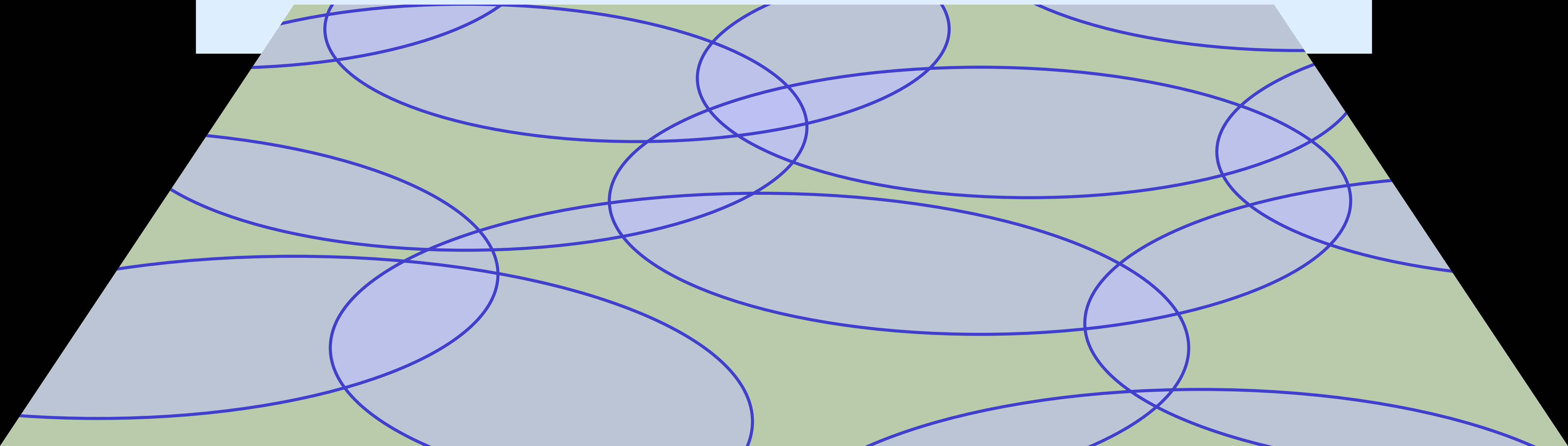
“PLY”

- Ply of a point
  - Given a set of regions



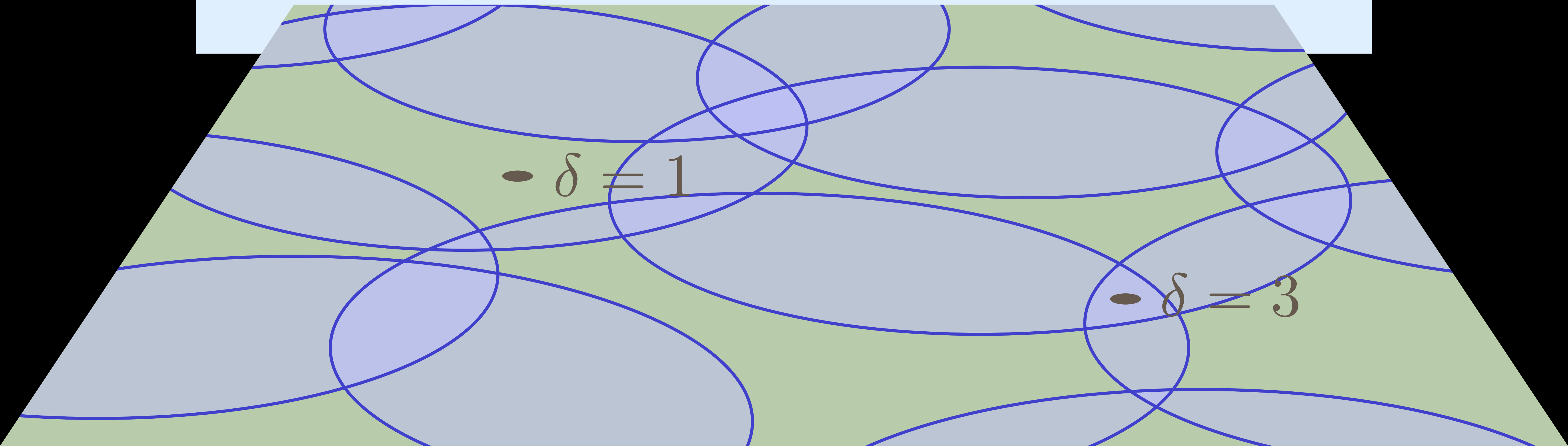
"PLY"

- Ply of a point
  - Given a set of regions
  - $\delta(p) = |\{R : p \in R\}|$



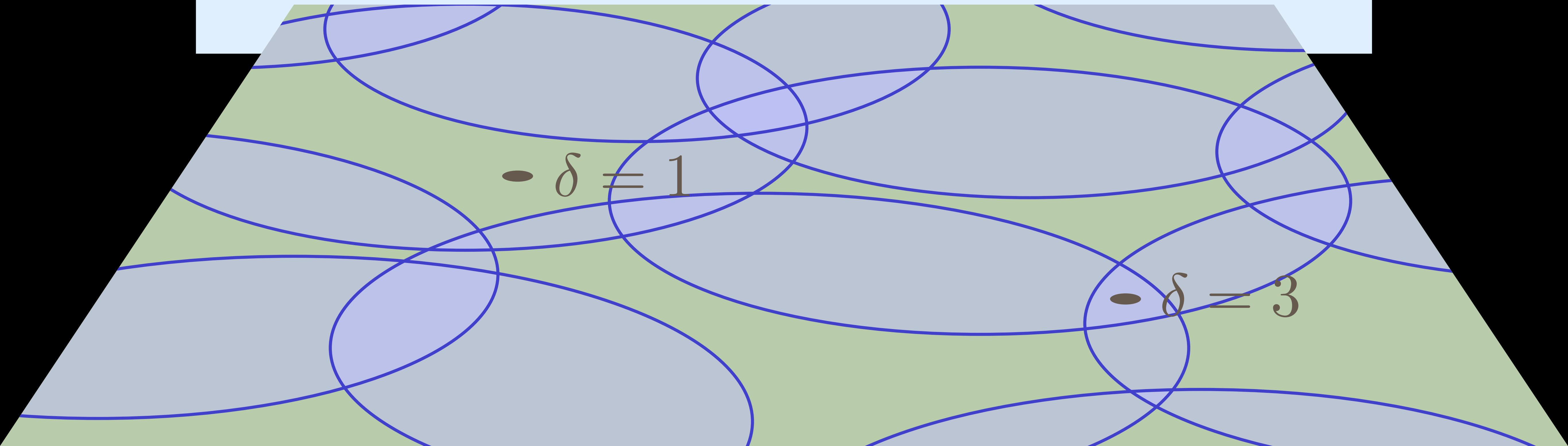
“PLY”

- Ply of a point
  - Given a set of regions
  - $\delta(p) = |\{R : p \in R\}|$



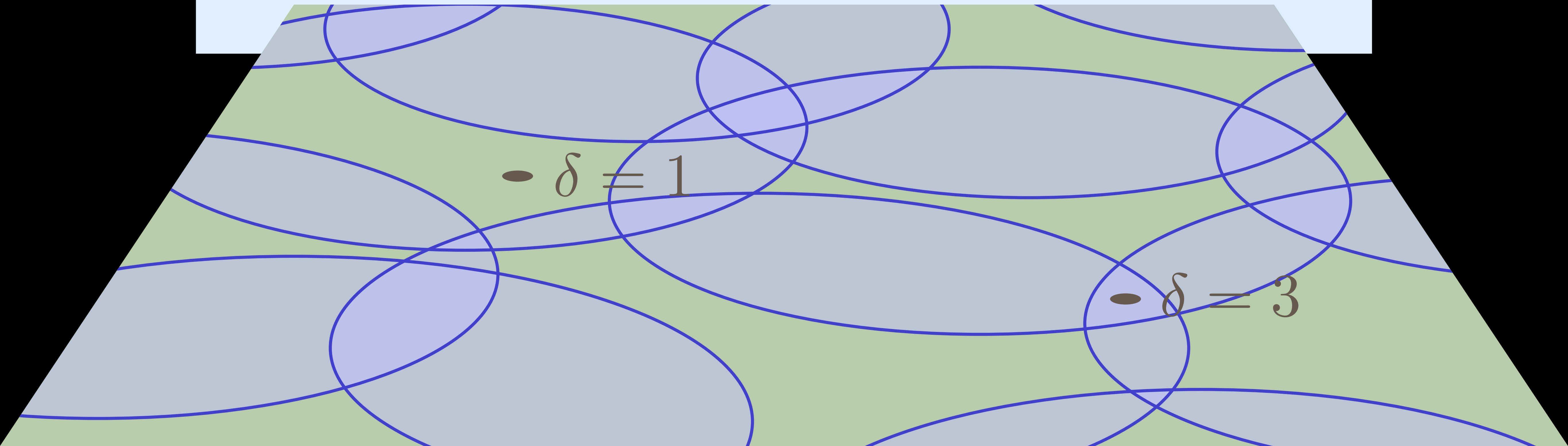
"PLY"

- Ply of a point
  - Given a set of regions
  - $\delta(p) = |\{R : p \in R\}|$
- Ply of the set of regions



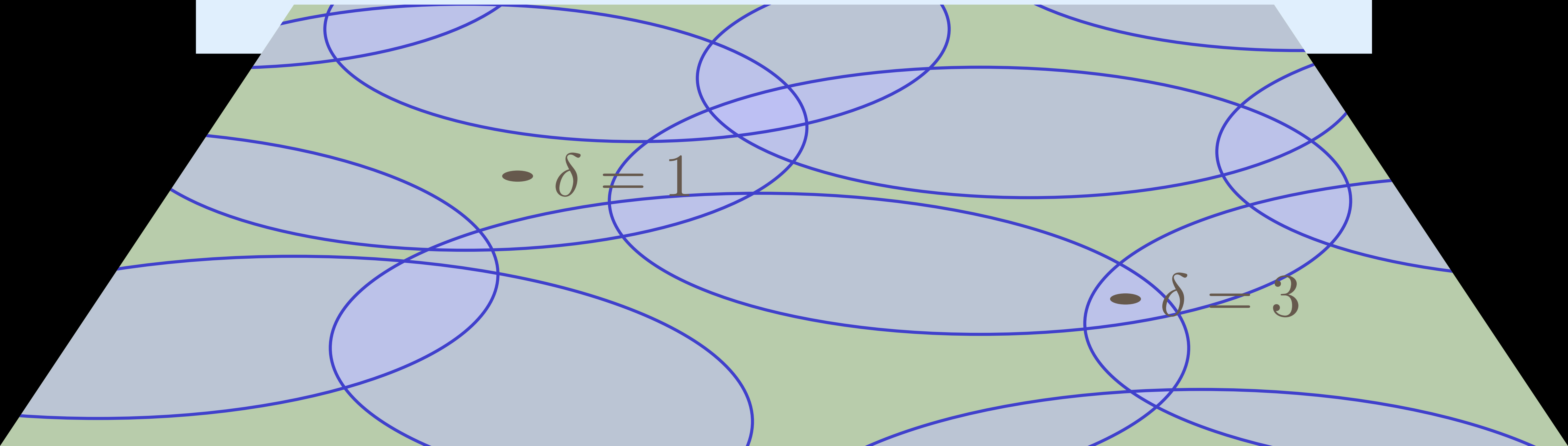
“PLY”

- Ply of a point
  - Given a set of regions
  - $\delta(p) = |\{R : p \in R\}|$
- Ply of the set of regions
  - Maximum ply of all points



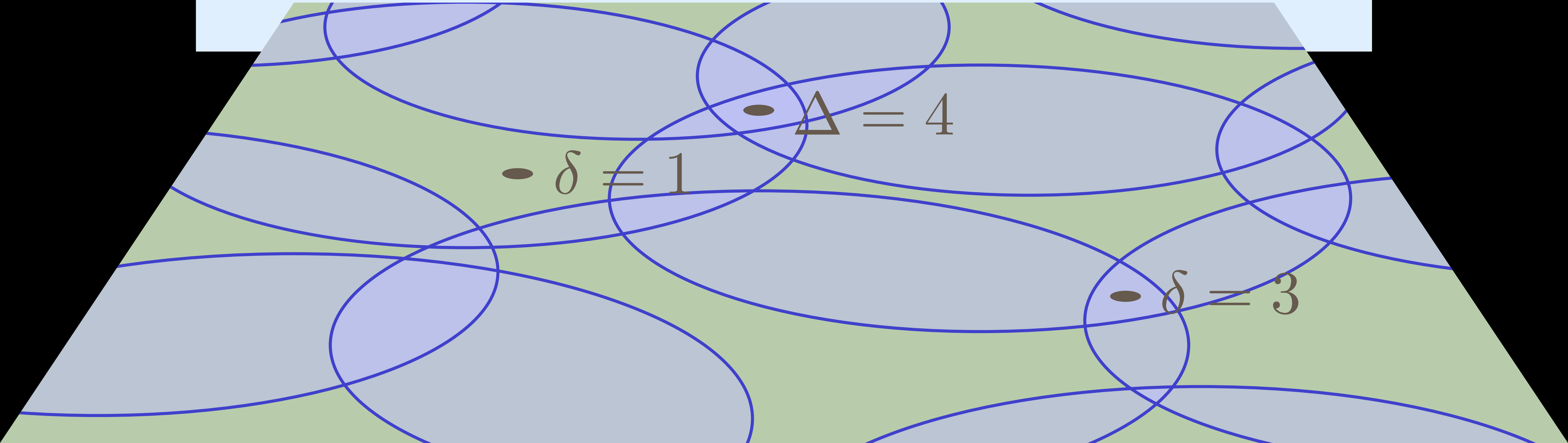
“PLY”

- Ply of a point
  - Given a set of regions
  - $\delta(p) = |\{R : p \in R\}|$
- Ply of the set of regions
  - Maximum ply of all points
  - $\Delta = \max_p \delta(p)$



"PLY"

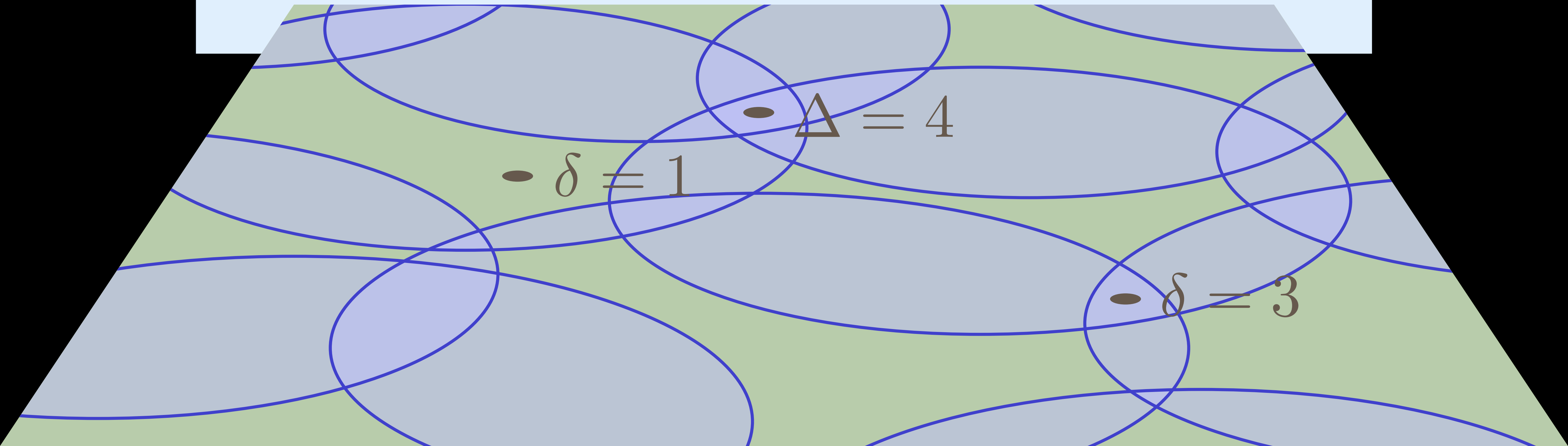
- Ply of a point
  - Given a set of regions
  - $\delta(p) = |\{R : p \in R\}|$
- Ply of the set of regions
  - Maximum ply of all points
  - $\Delta = \max_p \delta(p)$



“PLY”

- Ply of a point
  - Given a set of regions
  - $\delta(p) = |\{R : p \in R\}|$
- Ply of the set of regions
  - Maximum ply of all points
  - $\Delta = \max_p \delta(p)$

POSTULATION: *Low ply is good!*



“COMPETITIVE”

“COMPETITIVE”

- Classical, “offline” algorithm

“COMPETITIVE”

- Classical, “offline” algorithm
  - Receives the whole input at once

“COMPETITIVE”

- Classical, “offline” algorithm
  - Receives the whole input at once
- Online algorithm

"COMPETITIVE"

- Classical, "offline" algorithm
  - Receives the whole input at once
- Online algorithm
  - Receives a sequence of input *events*

“COMPETITIVE”

- Classical, “offline” algorithm
  - Receives the whole input at once
- Online algorithm
  - Receives a sequence of input *events*
  - Needs to react to each event before knowing what the next event will be

“COMPETITIVE”

- Classical, “offline” algorithm
  - Receives the whole input at once
- Online algorithm
  - Receives a sequence of input *events*
  - Needs to react to each event before knowing what the next event will be
- Competitive ratio

“COMPETITIVE”

- Classical, “offline” algorithm
  - Receives the whole input at once
- Online algorithm
  - Receives a sequence of input *events*
  - Needs to react to each event before knowing what the next event will be
- Competitive ratio
  - Performance measure for online algorithms

“COMPETITIVE”

- Classical, “offline” algorithm
  - Receives the whole input at once
- Online algorithm
  - Receives a sequence of input *events*
  - Needs to react to each event before knowing what the next event will be
- Competitive ratio
  - Performance measure for online algorithms
  - Cost of online algorithm divided by cost of best possible offline algorithm

COMPETITIVE QUERY STRATEGIES FOR MINIMISING THE PLY OF THE POTENTIAL LOCATIONS OF MOVING POINTS

Will Evans
David Kirkpatrick
Frank Staals
Maarten Löffler

COMPETITIVE QUERY STRATEGIES FOR MINIMISING THE PLY OF THE POTENTIAL LOCATIONS OF MOVING POINTS

Will Evans
David Kirkpatrick
Frank Staals
Maarten Löffler

PROBLEM STATEMENT

PROBLEM STATEMENT

- Input

PROBLEM STATEMENT

- Input
  - Set of n moving points with initial regions

PROBLEM STATEMENT

- Input
  - Set of n moving points with initial regions
  - Target time t^*

PROBLEM STATEMENT

- Input
  - Set of n moving points with initial regions
  - Target time t^*
- Objective

PROBLEM STATEMENT

- Input
  - Set of n moving points with initial regions
  - Target time t^*
- Objective
  - Select one region to query each time step

PROBLEM STATEMENT

- Input
  - Set of n moving points with initial regions
  - Target time t^*
- Objective
  - Select one region to query each time step
  - Minimize the ply of the regions at time t^*

PROBLEM STATEMENT

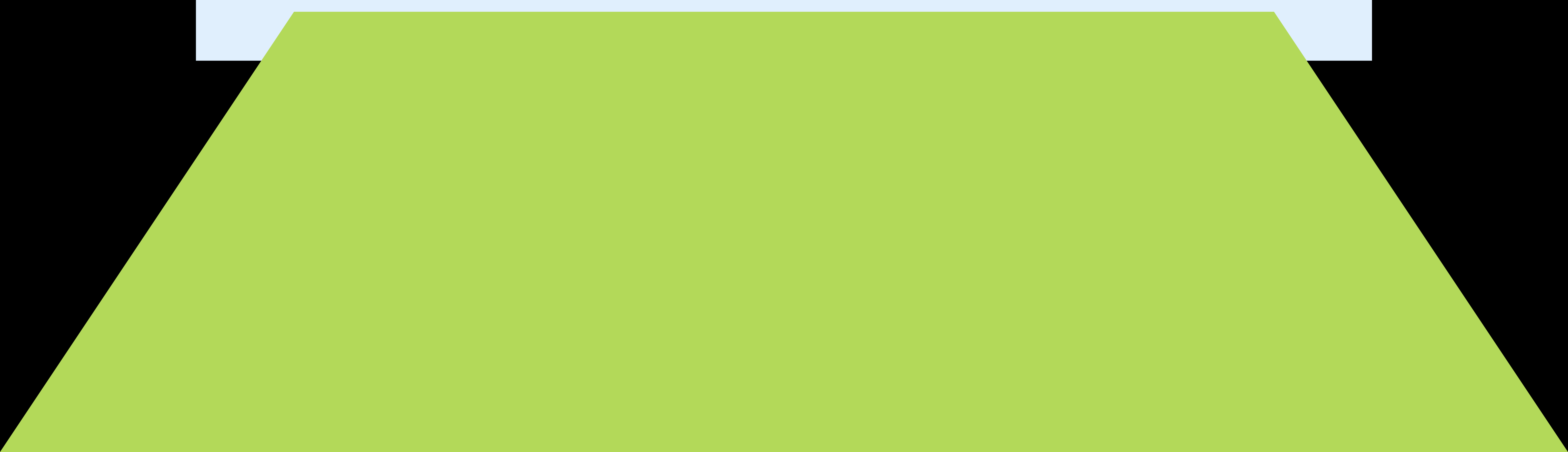
- Input
  - Set of n moving points with initial regions
  - Target time t^*
- Objective
  - Select one region to query each time step
  - Minimize the ply of the regions at time t^*
- Competitive analysis

PROBLEM STATEMENT

- Input
  - Set of n moving points with initial regions
  - Target time t^*
- Objective
  - Select one region to query each time step
  - Minimize the ply of the regions at time t^*
- Competitive analysis
  - Compare against adversary that knows the full trajectories of the points in advance

TECHNICAL STUFF

PROJECTED REGIONS

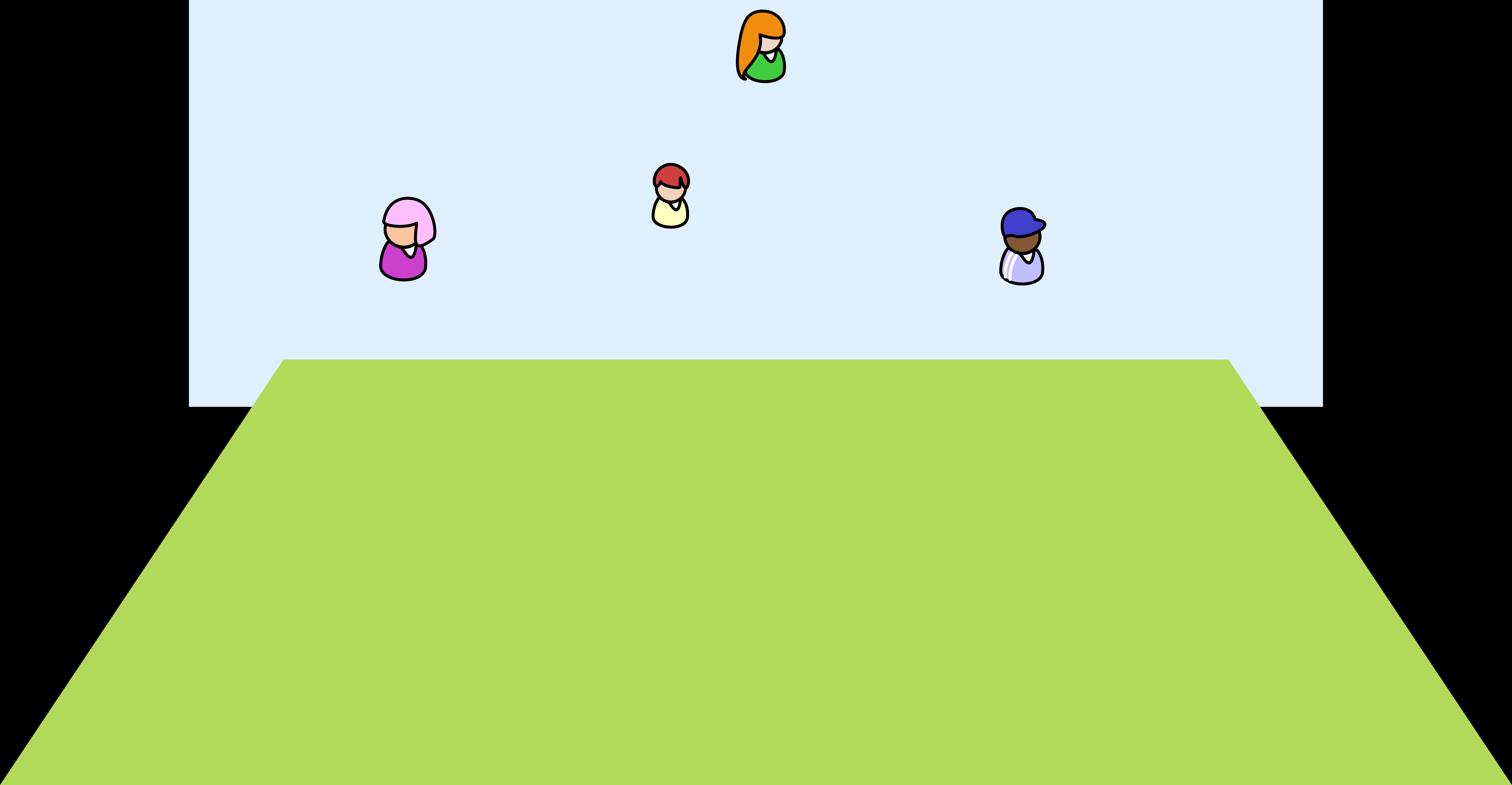


PROJECTED REGIONS

- View points in time-space

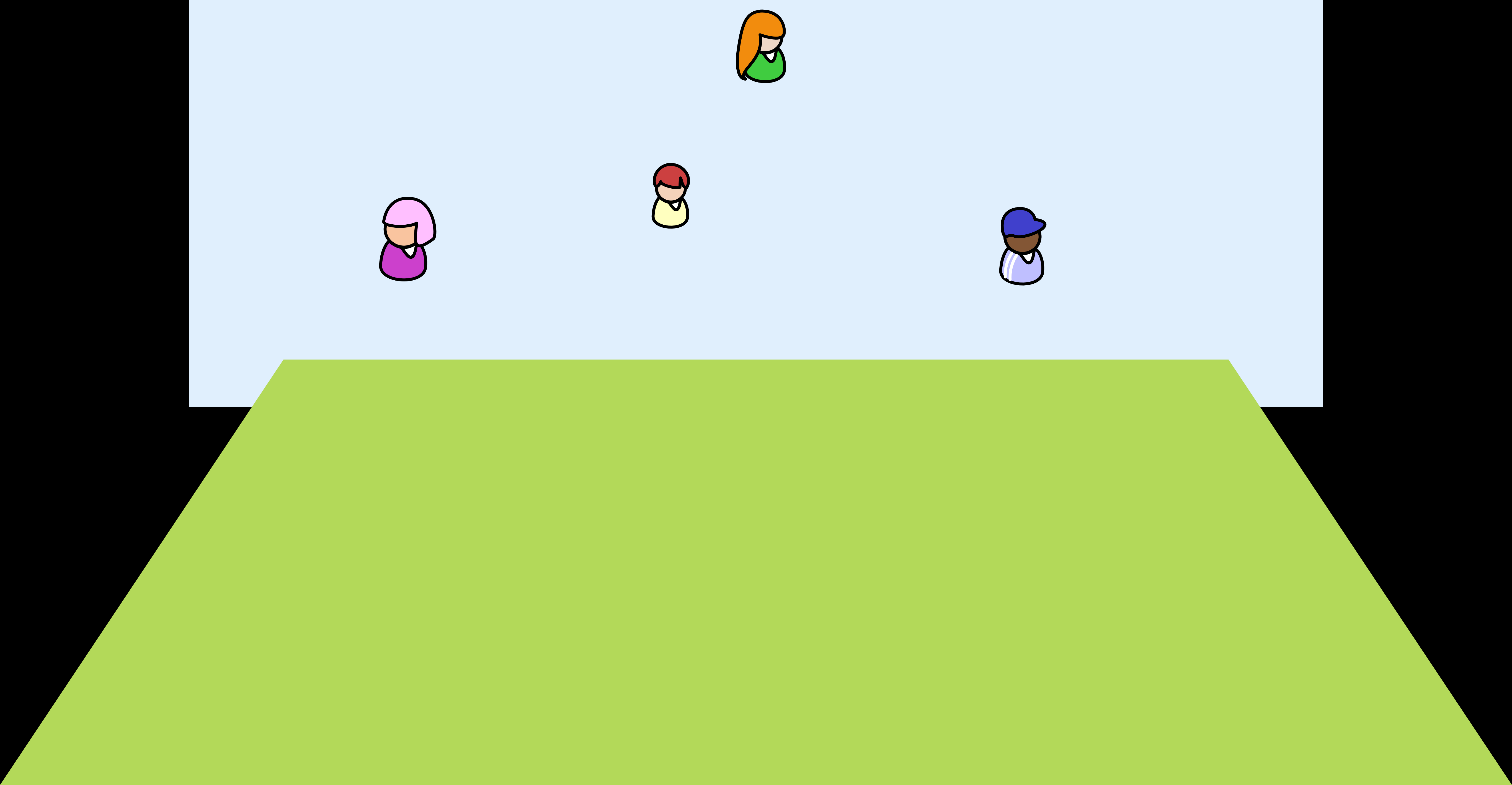
PROJECTED REGIONS

- View points in time-space



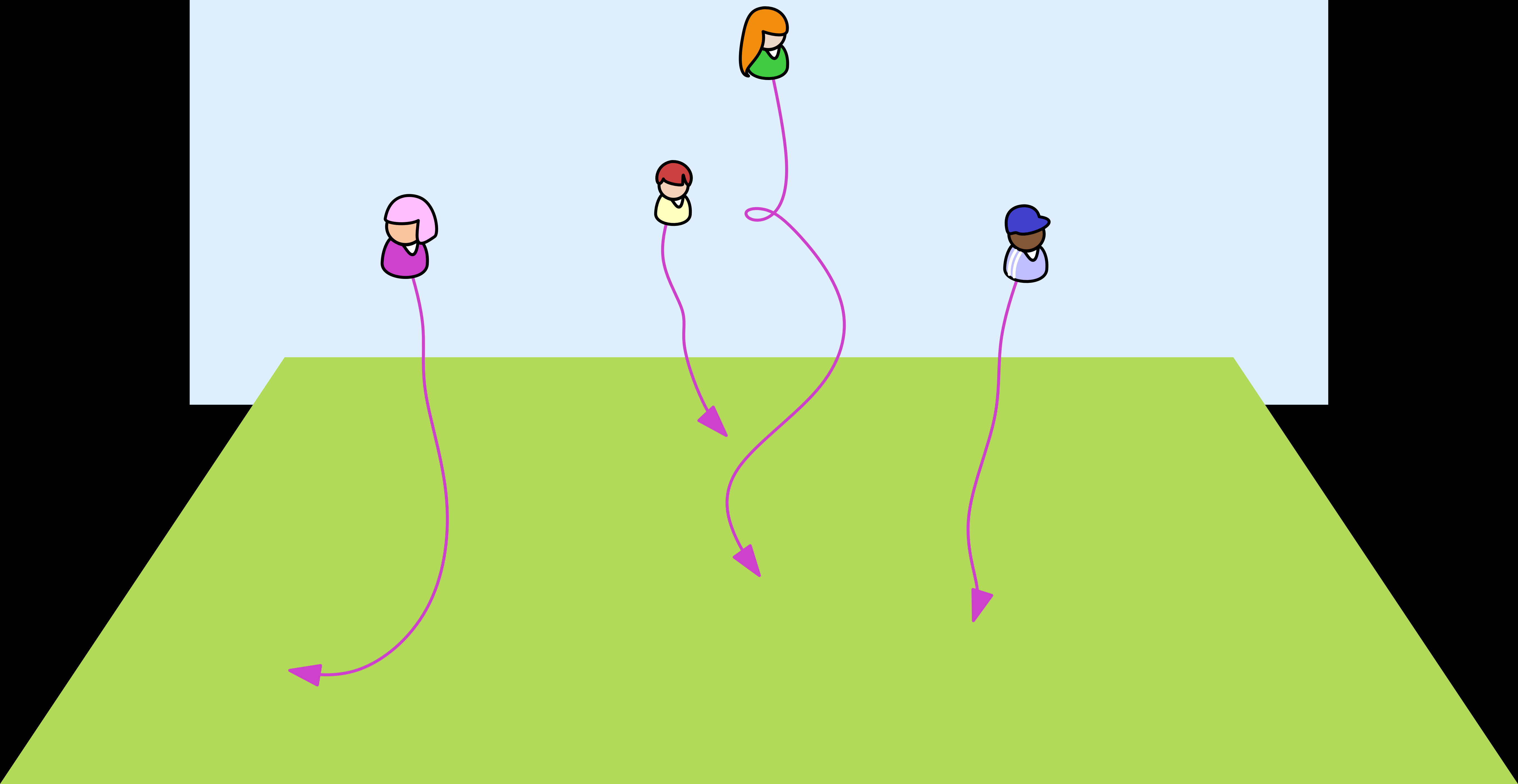
PROJECTED REGIONS

- View points in time-space
  - Points follow some unknown z-monotone trajectory



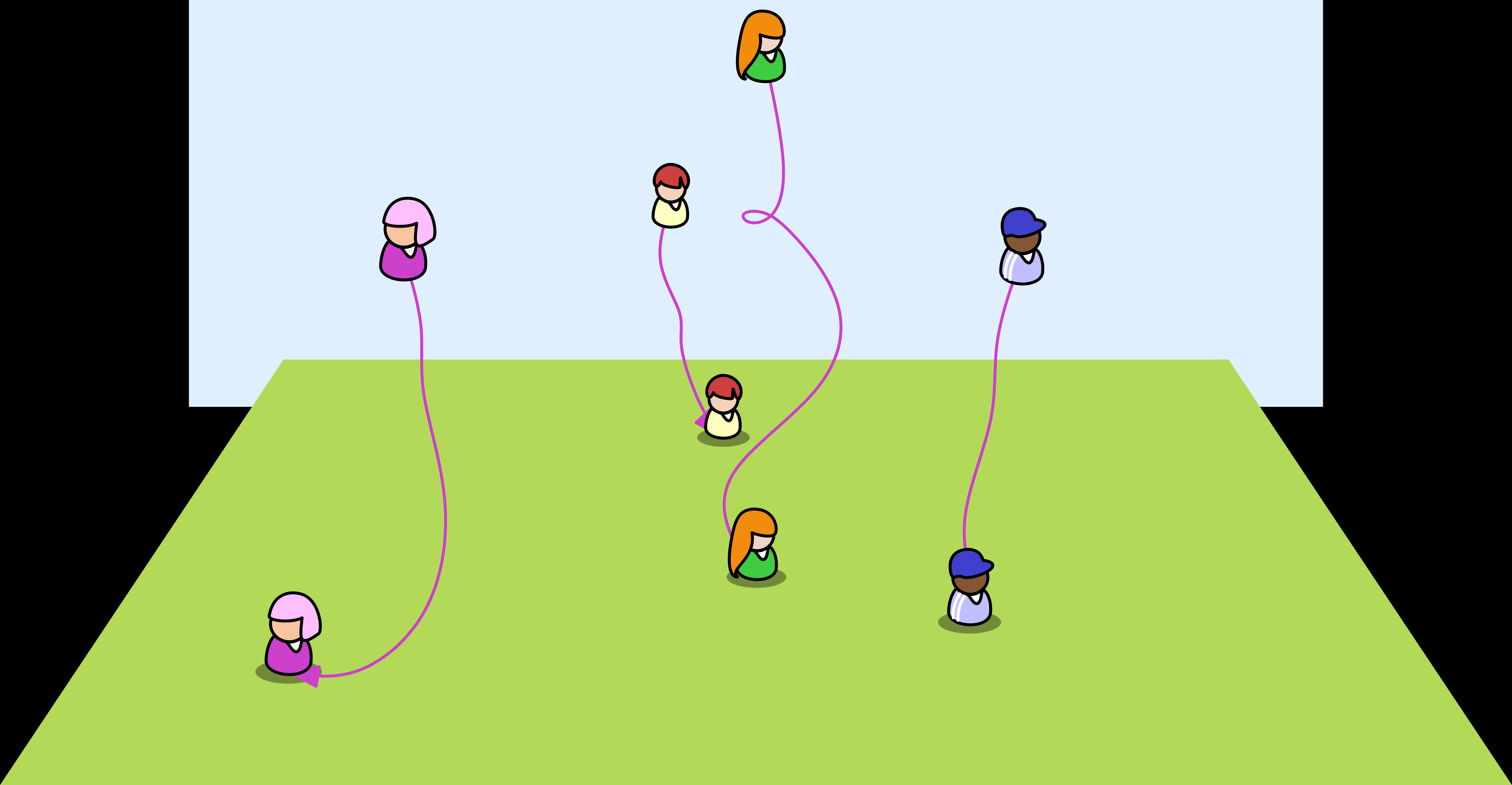
PROJECTED REGIONS

- View points in time-space
  - Points follow some unknown z-monotone trajectory



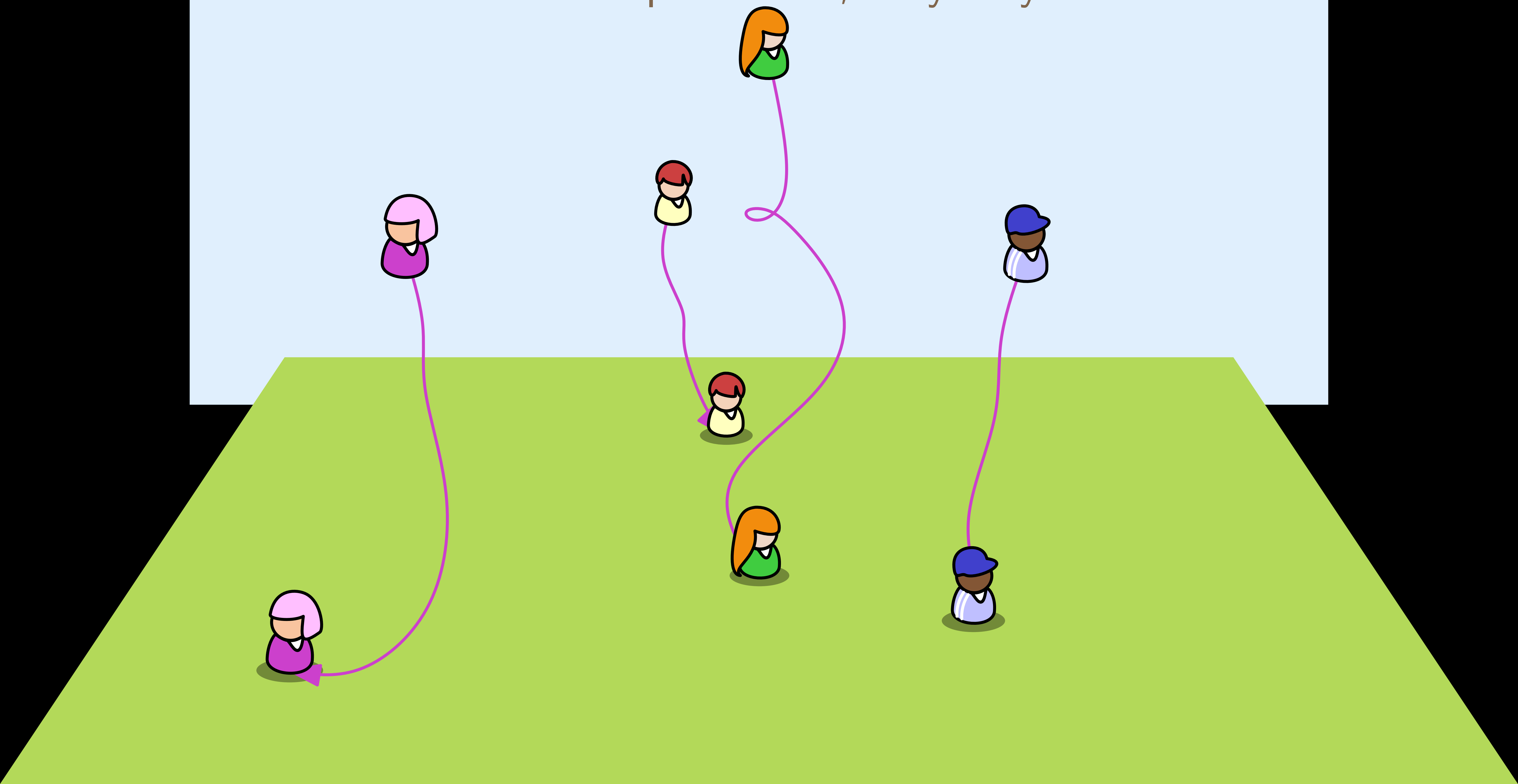
PROJECTED REGIONS

- View points in time-space
  - Points follow some unknown z-monotone trajectory



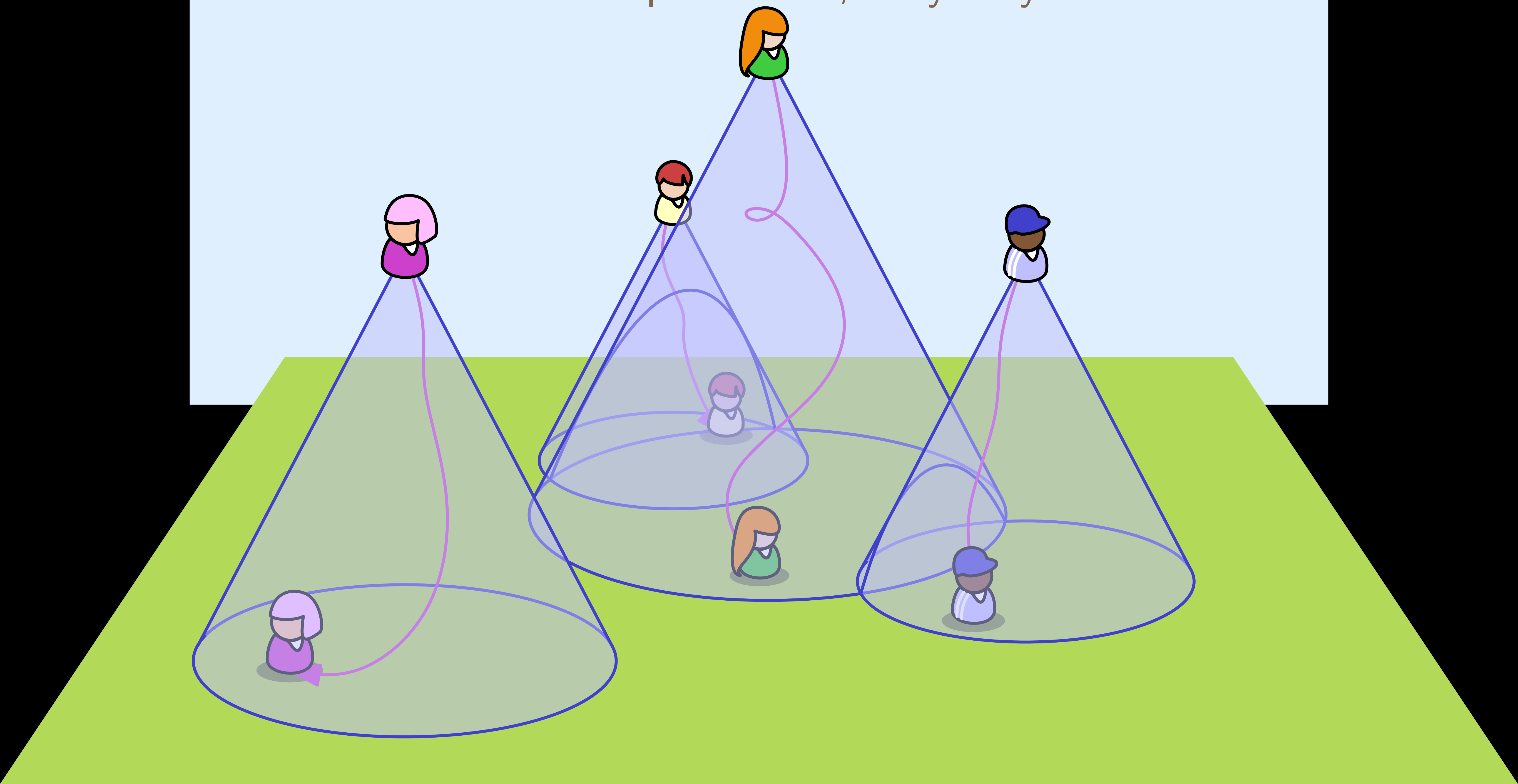
PROJECTED REGIONS

- View points in time-space
  - Points follow some unknown z-monotone trajectory
  - Because of speed limit, they stay in cones



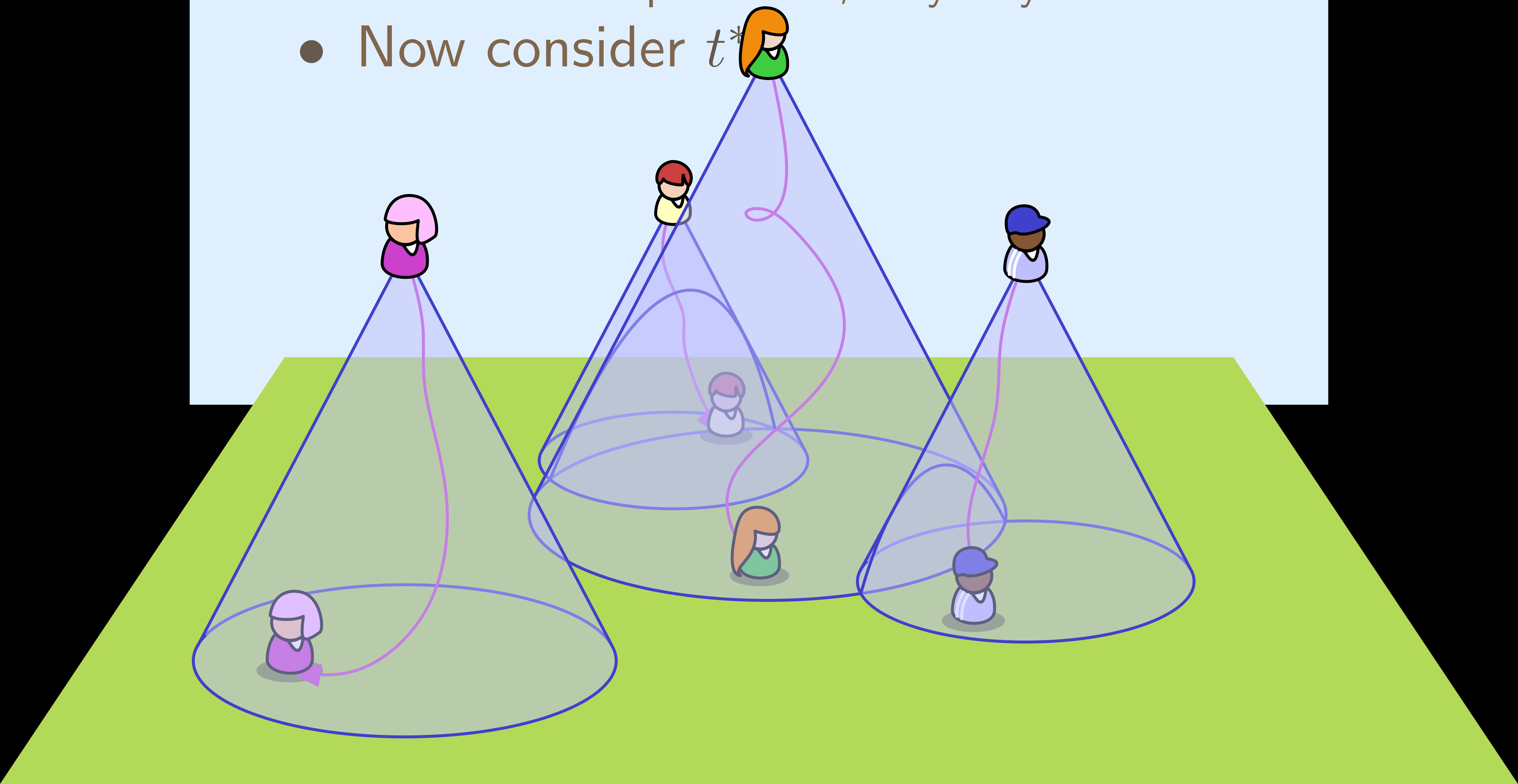
PROJECTED REGIONS

- View points in time-space
  - Points follow some unknown z-monotone trajectory
  - Because of speed limit, they stay in cones



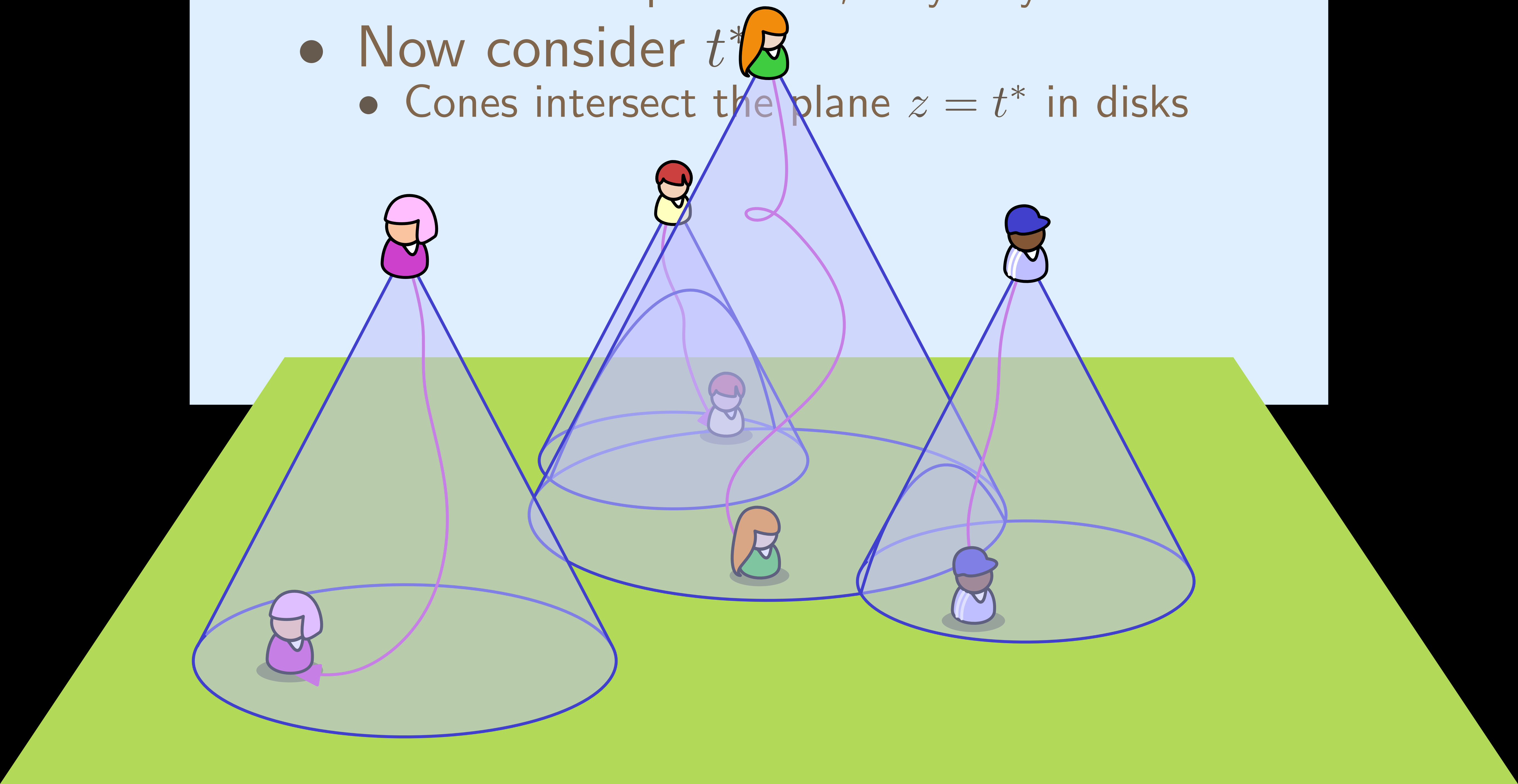
PROJECTED REGIONS

- View points in time-space
  - Points follow some unknown z-monotone trajectory
  - Because of speed limit, they stay in cones
- Now consider t^*



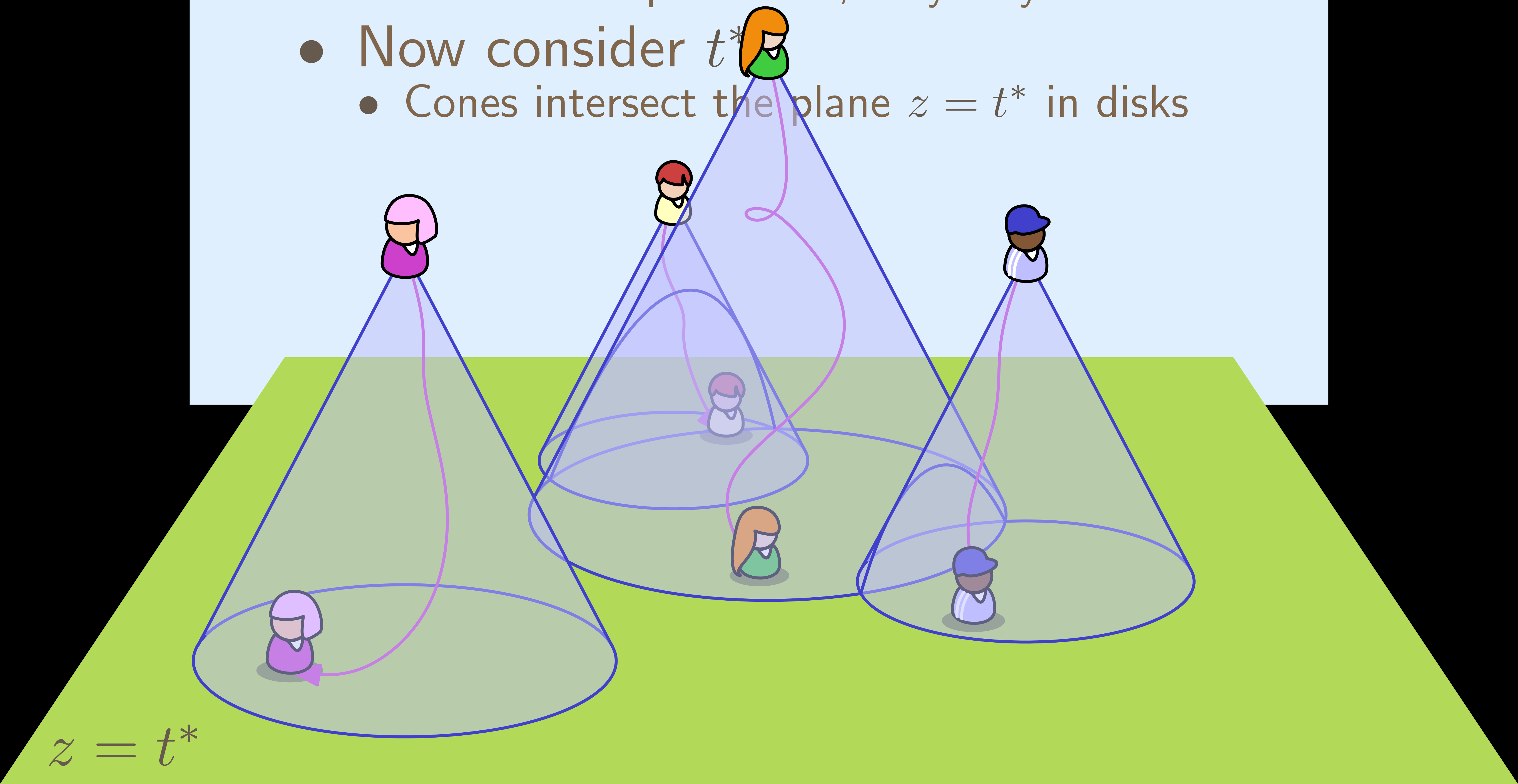
PROJECTED REGIONS

- View points in time-space
  - Points follow some unknown z-monotone trajectory
  - Because of speed limit, they stay in cones
- Now consider t^*
  - Cones intersect the plane $z = t^*$ in disks



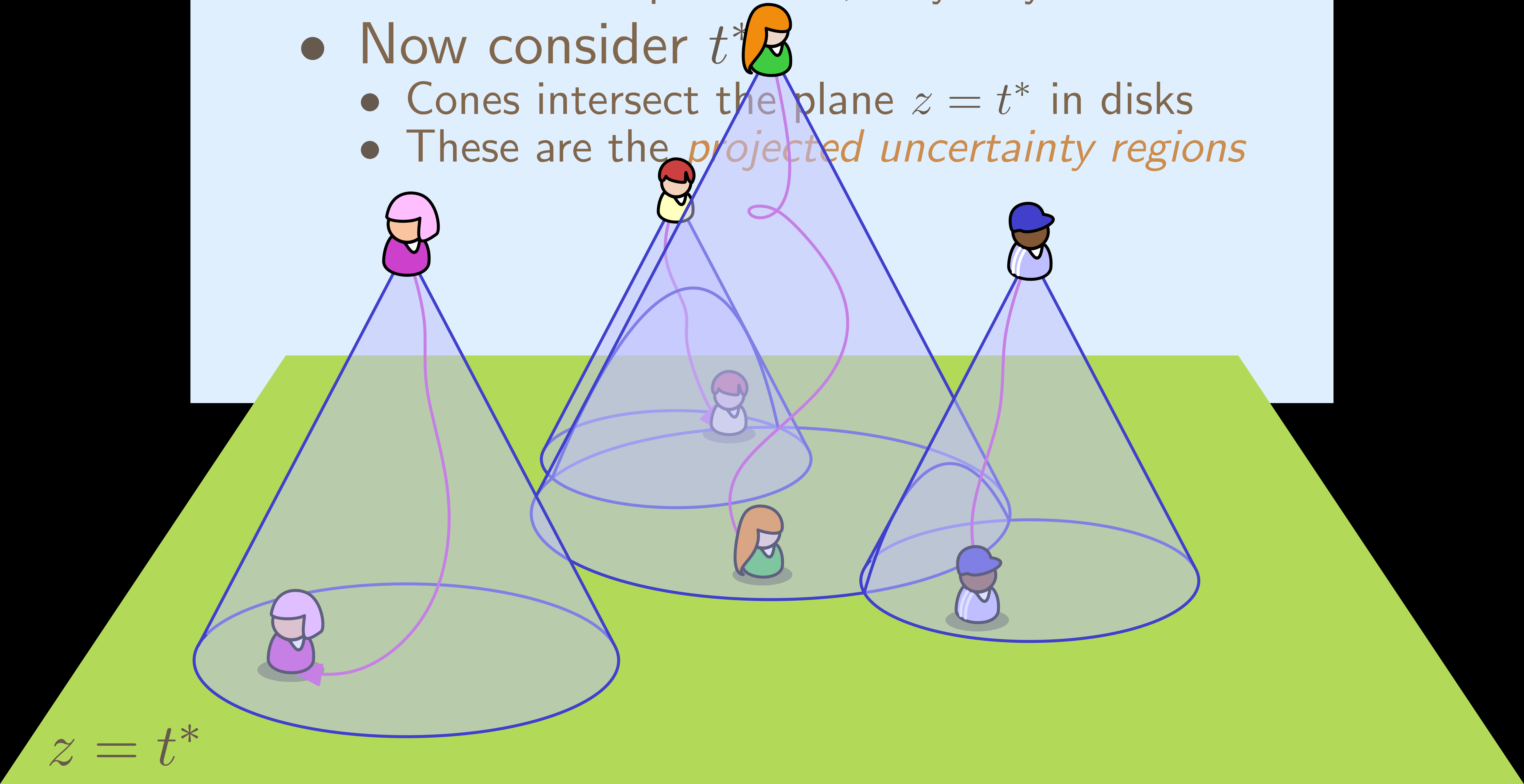
PROJECTED REGIONS

- View points in time-space
  - Points follow some unknown z-monotone trajectory
  - Because of speed limit, they stay in cones
- Now consider t^*
  - Cones intersect the plane $z = t^*$ in disks



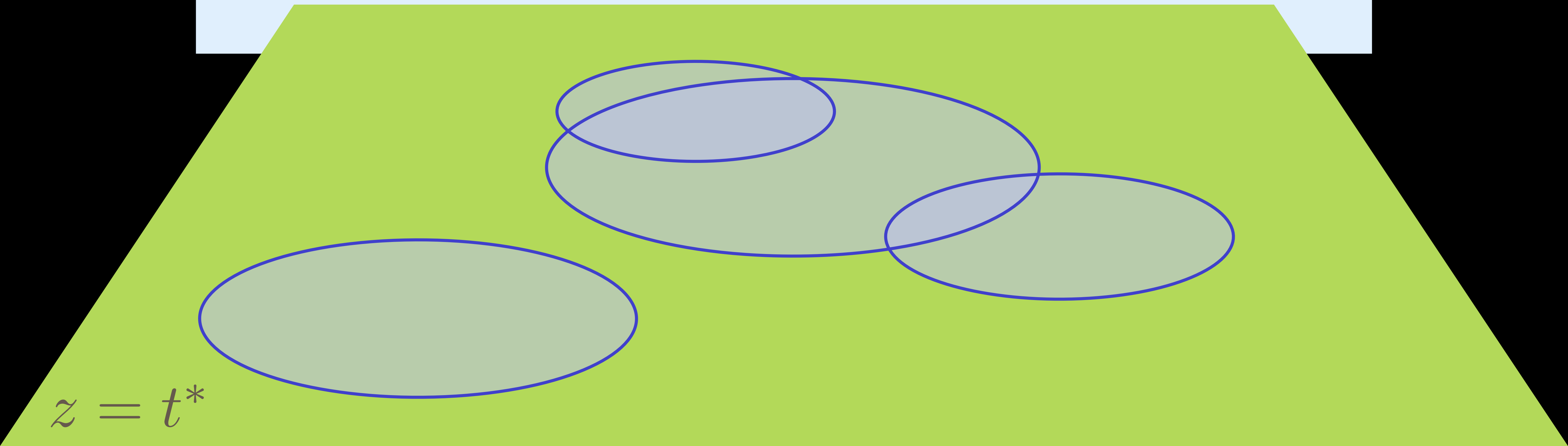
PROJECTED REGIONS

- View points in time-space
  - Points follow some unknown z-monotone trajectory
  - Because of speed limit, they stay in cones
- Now consider t^*
  - Cones intersect the plane $z = t^*$ in disks
  - These are the *projected uncertainty regions*



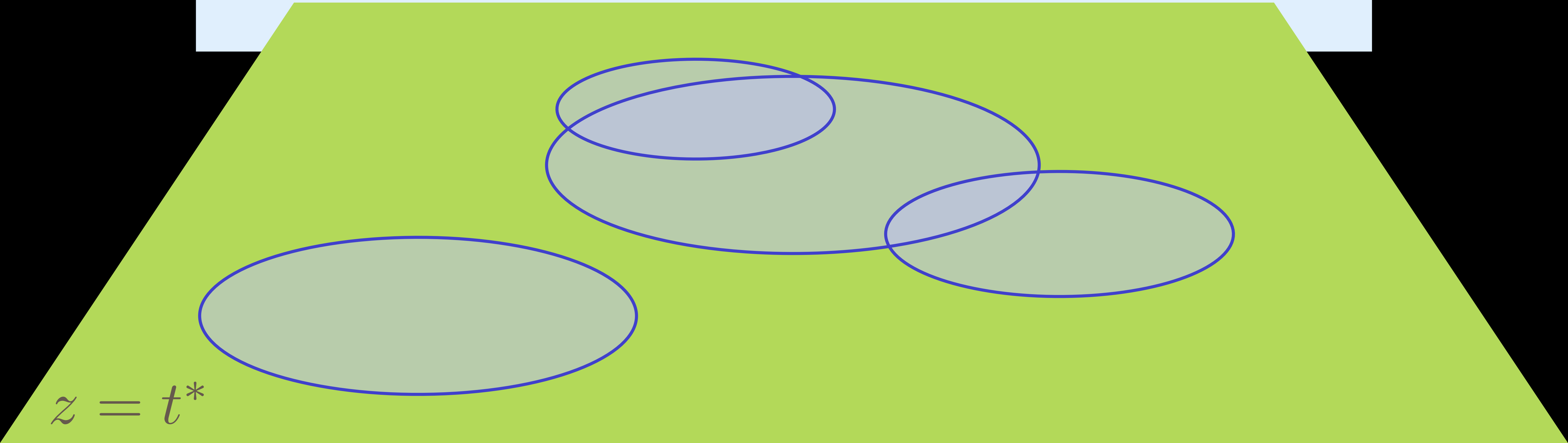
PROJECTED REGIONS

- View points in time-space
  - Points follow some unknown z-monotone trajectory
  - Because of speed limit, they stay in cones
- Now consider t^*
  - Cones intersect the plane $z = t^*$ in disks
  - These are the *projected uncertainty regions*



PROJECTED REGIONS

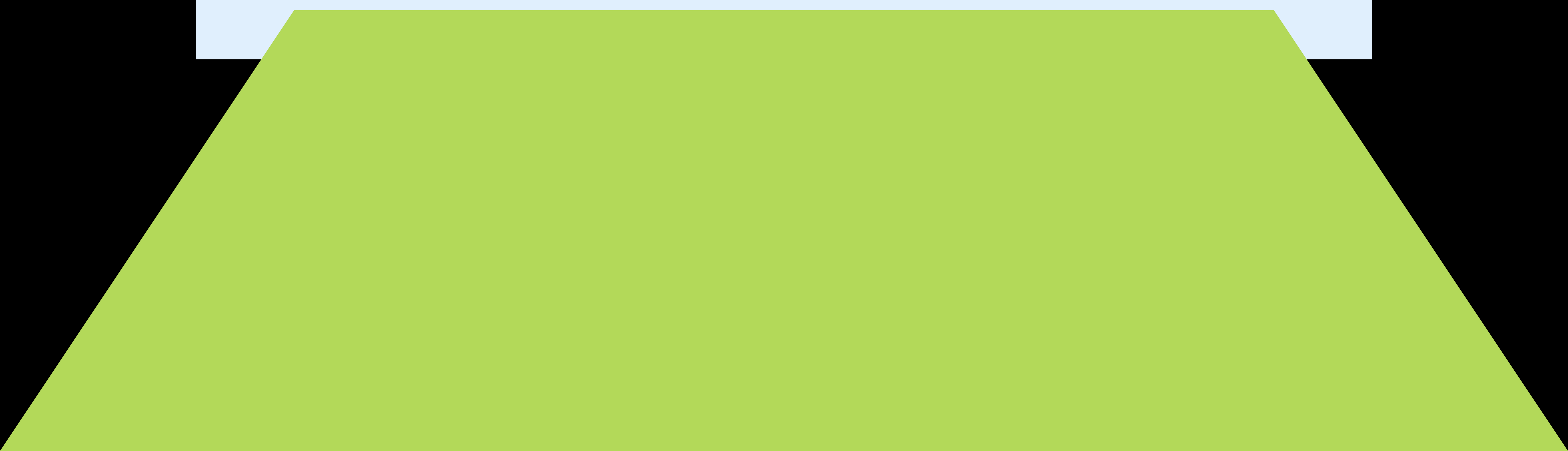
- View points in time-space
  - Points follow some unknown z-monotone trajectory
  - Because of speed limit, they stay in cones
- Now consider t^*
  - Cones intersect the plane $z = t^*$ in disks
  - These are the *projected uncertainty regions*
  - Projected regions never grow, and they shrink to radius $t^* - t$ when we query them



STRATEGY ATTEMPT 1

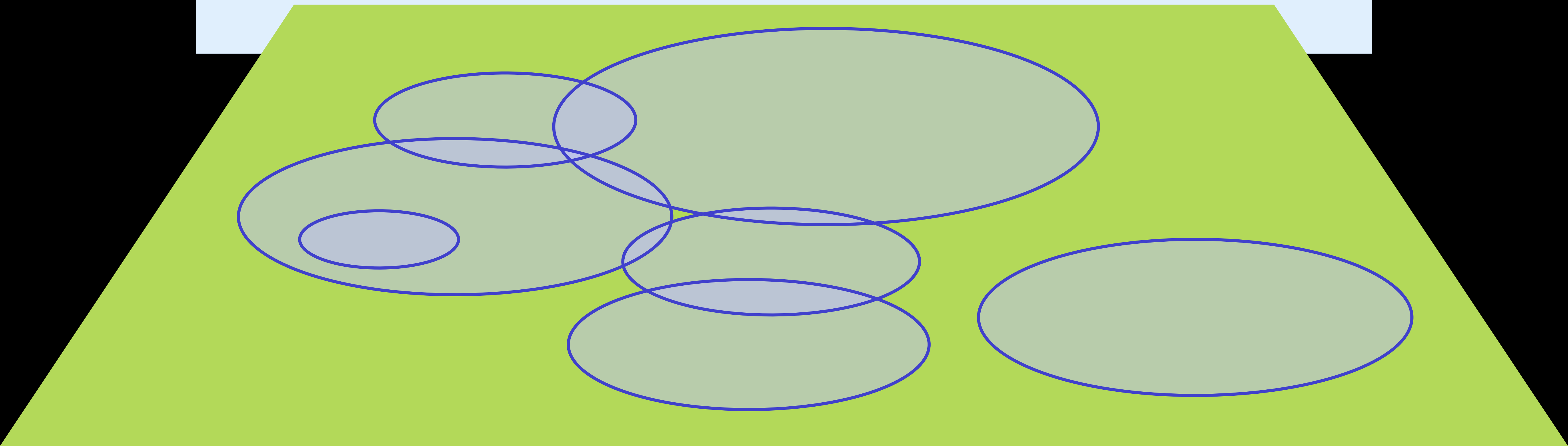
STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.



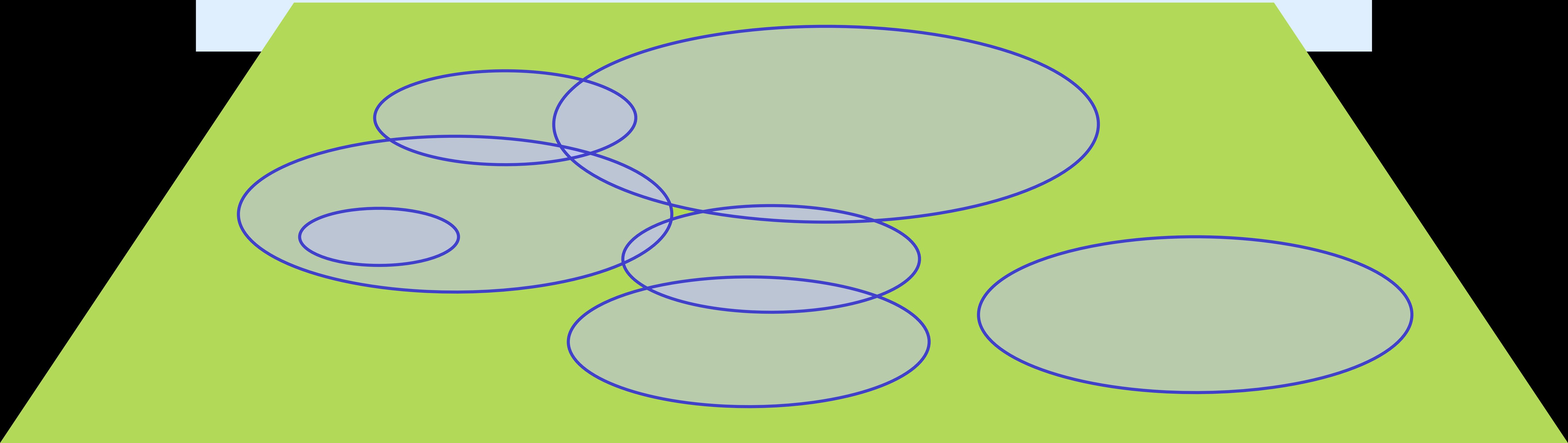
STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.



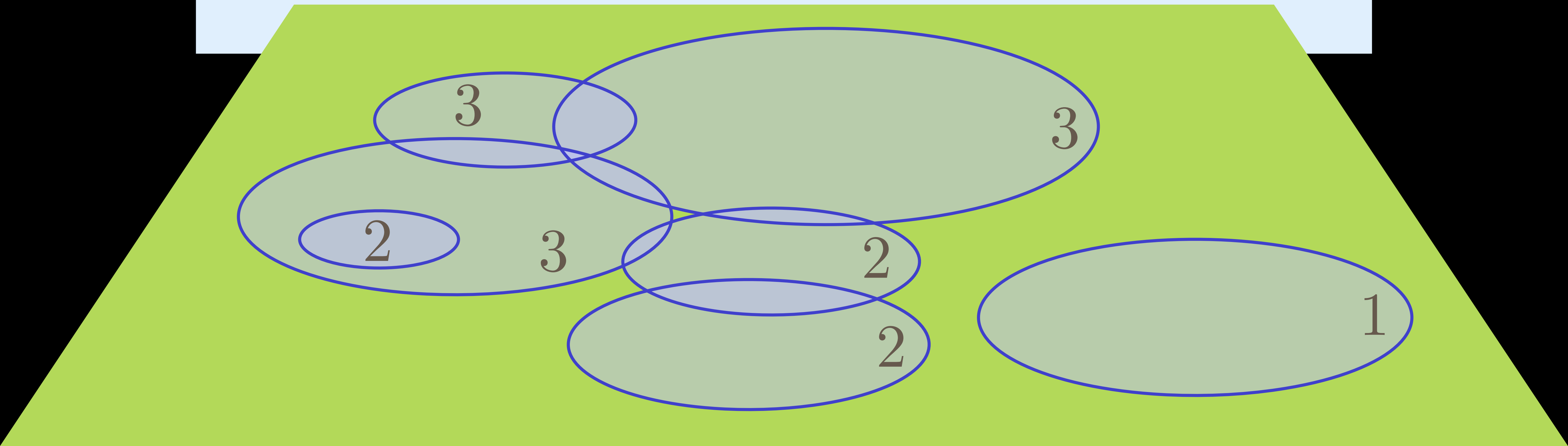
STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i



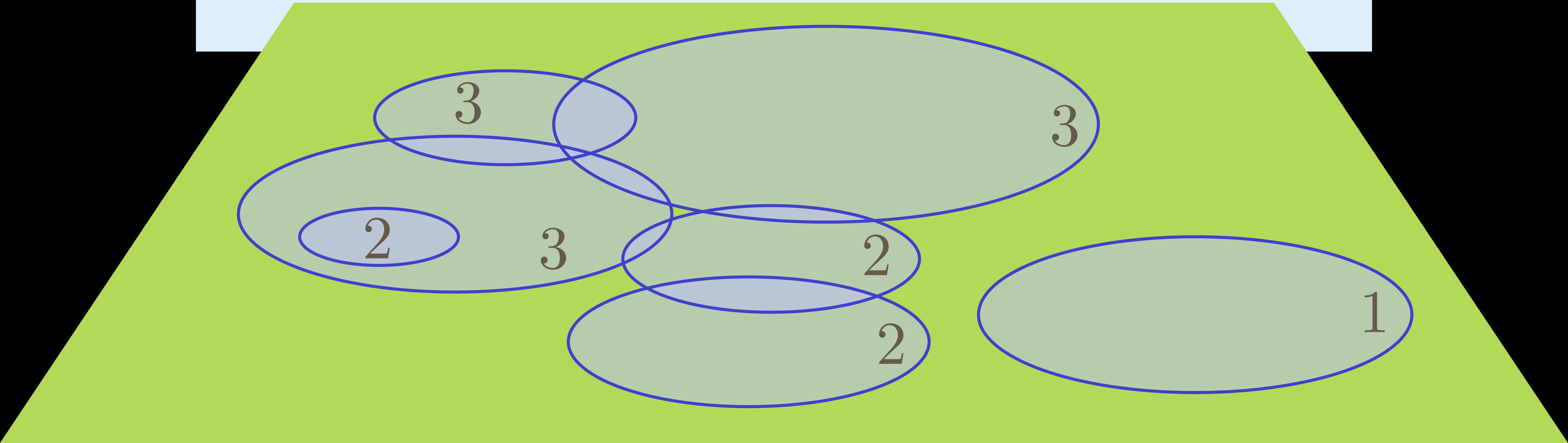
STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i



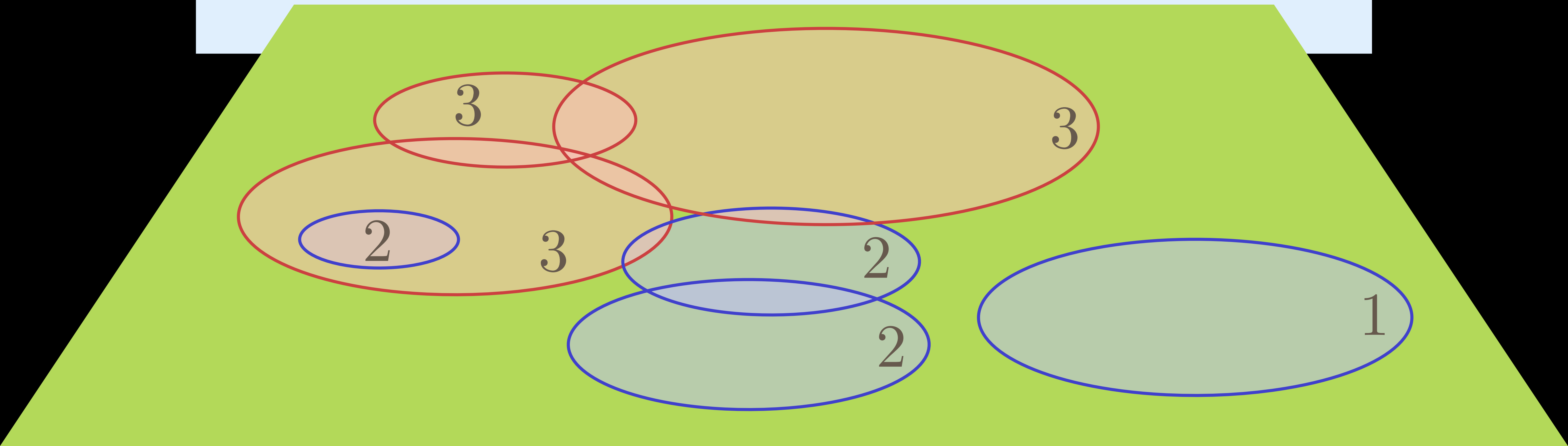
STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i



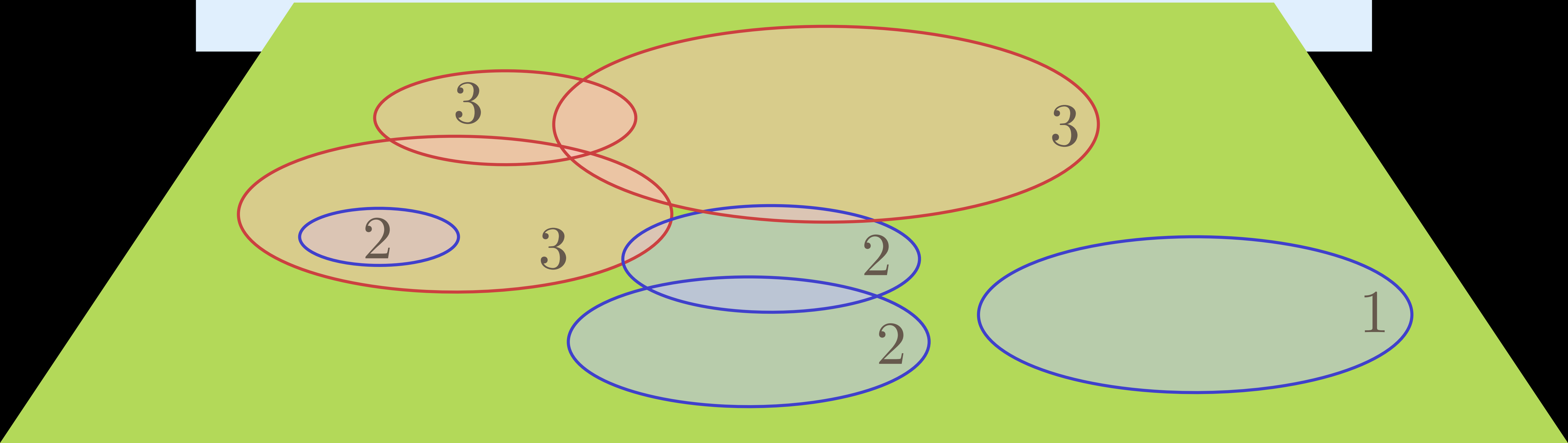
STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i



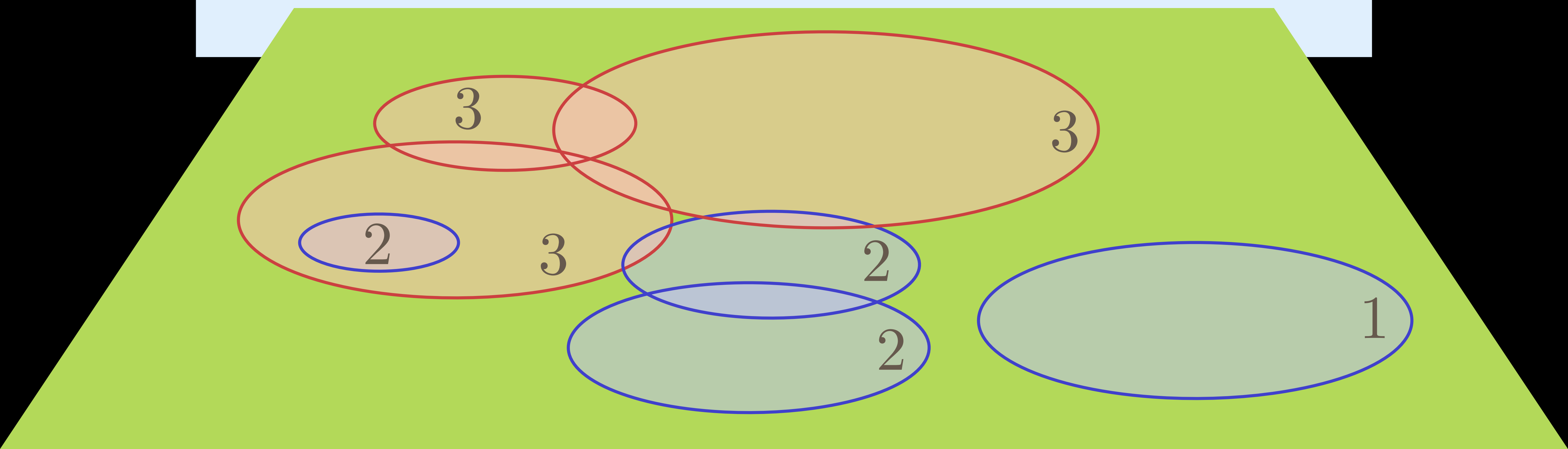
STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique



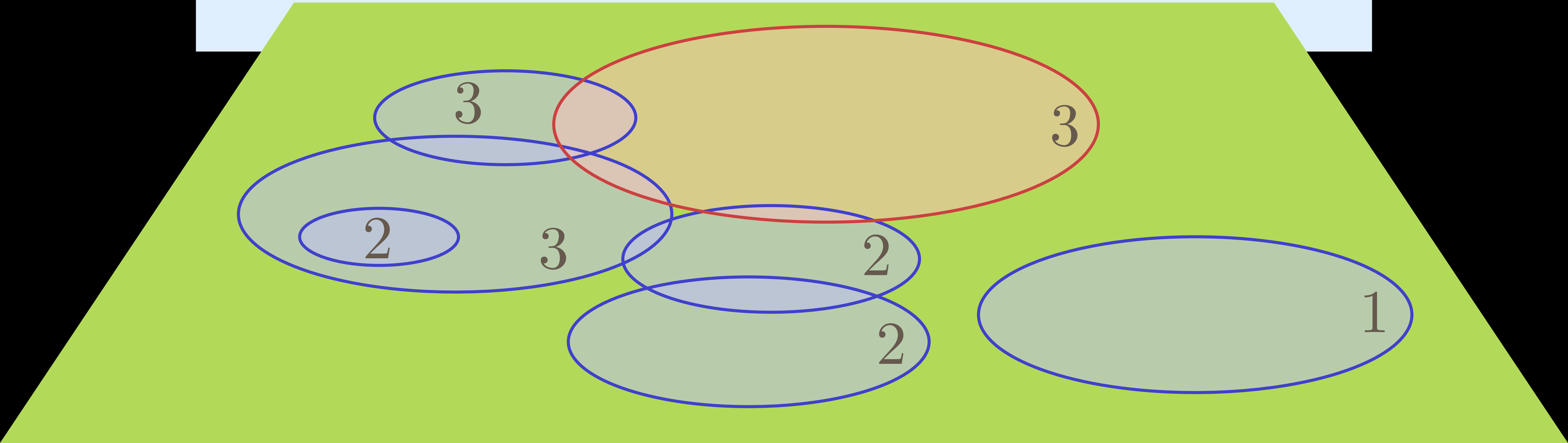
STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them



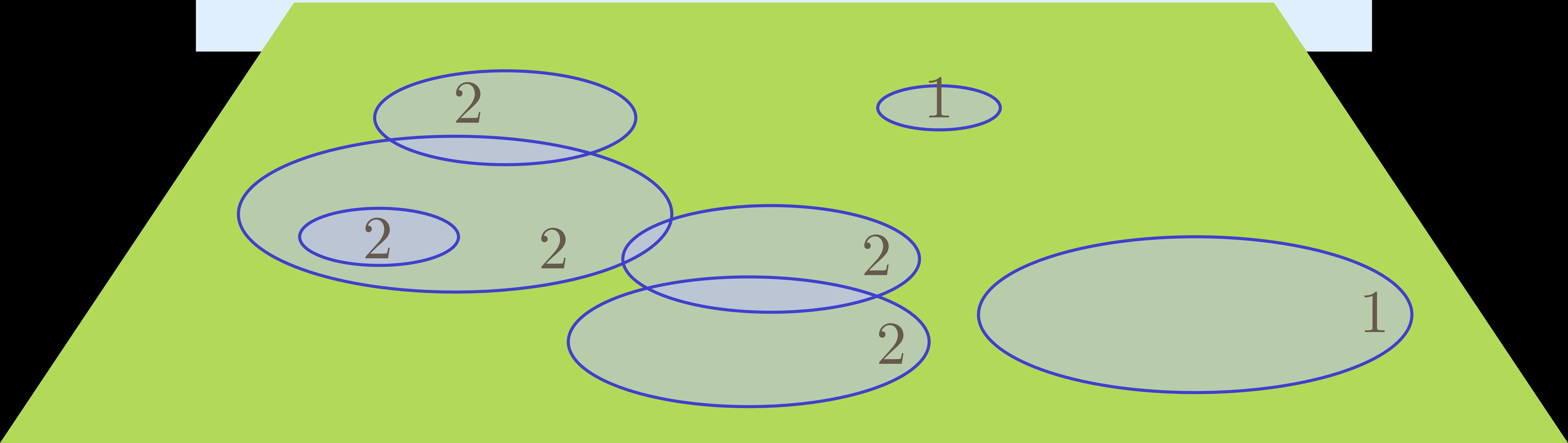
STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them



STRATEGY ATTEMPT 1

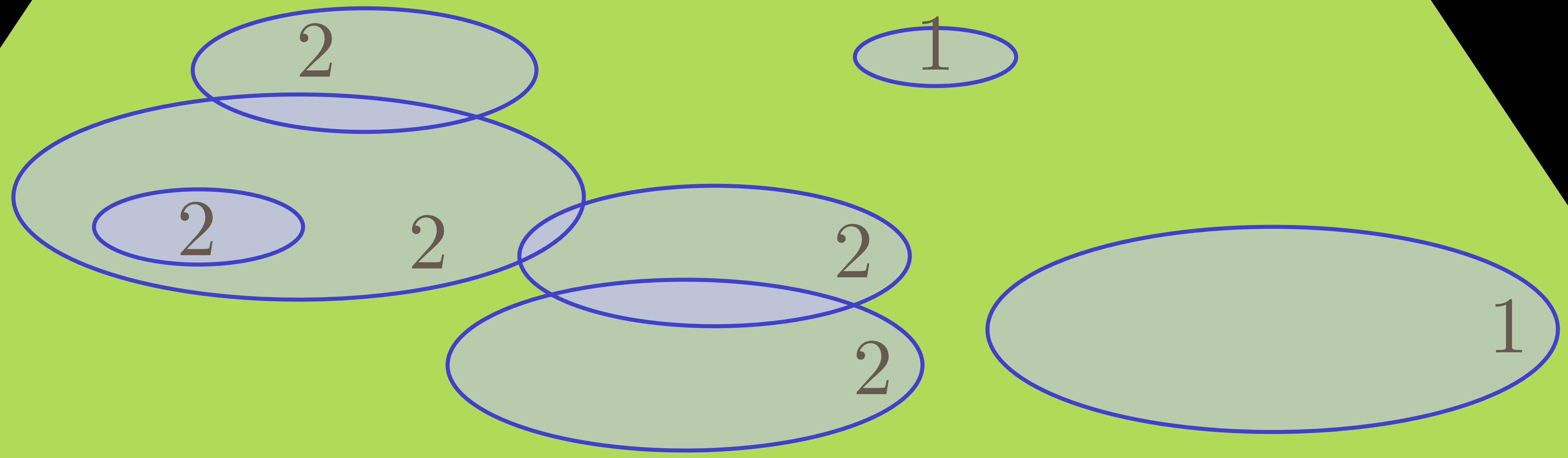
- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them



STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them

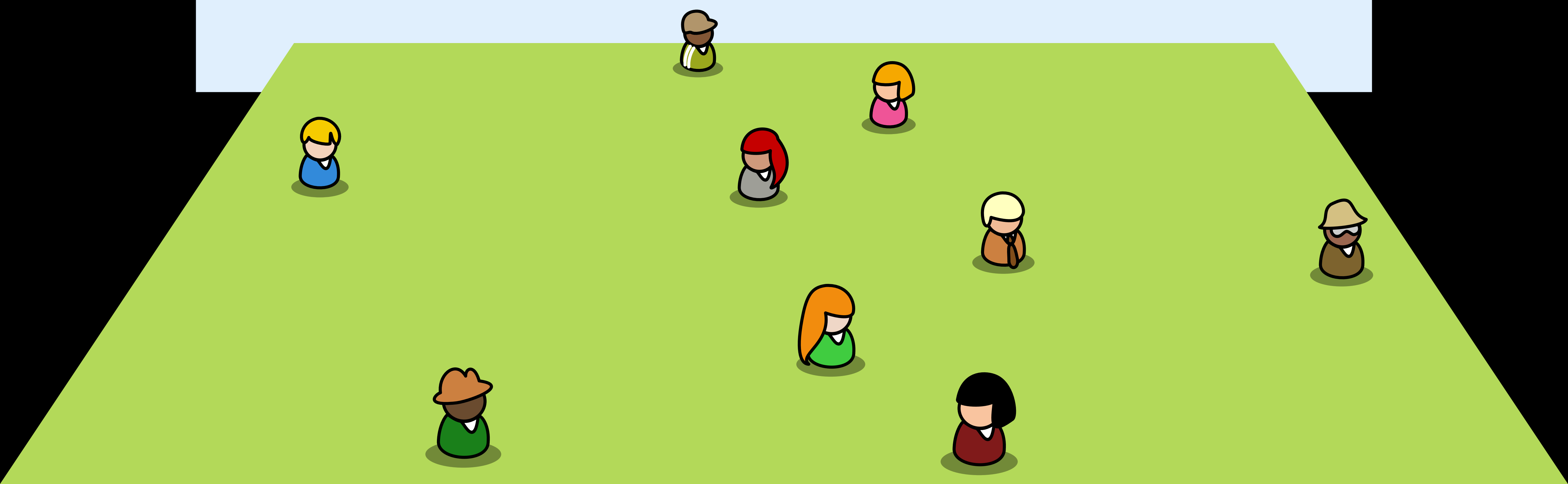
CLAIM: *Resulting ply Δ can be a factor n larger than optimal!*



STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them

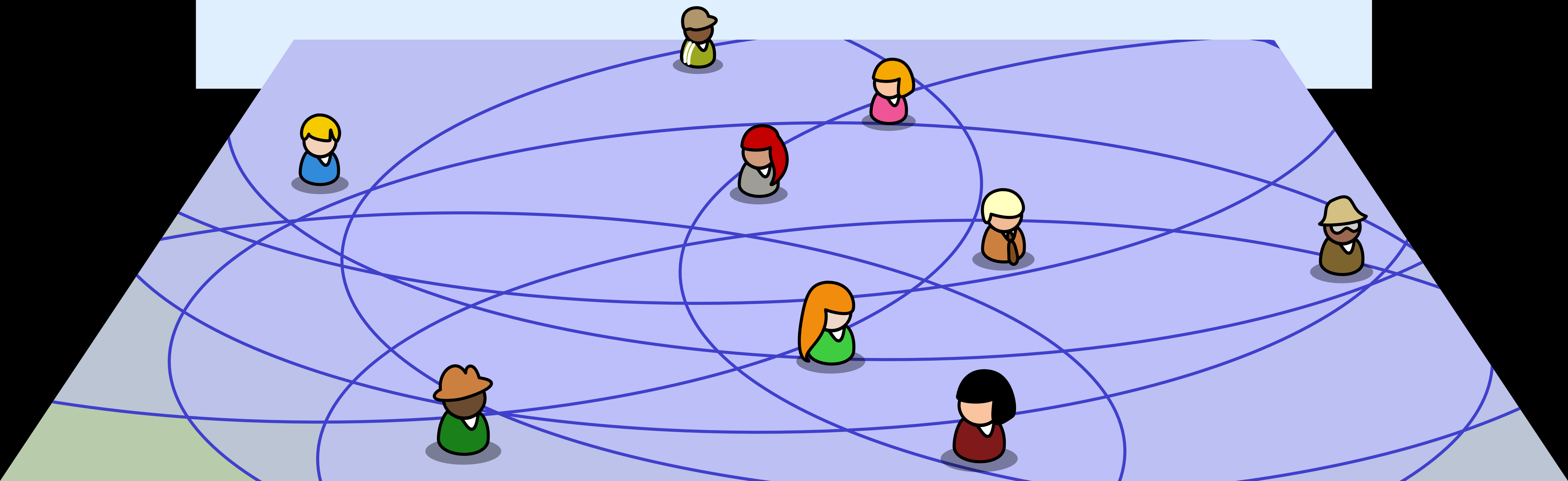
CLAIM: *Resulting ply Δ can be a factor n larger than optimal!*



STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them

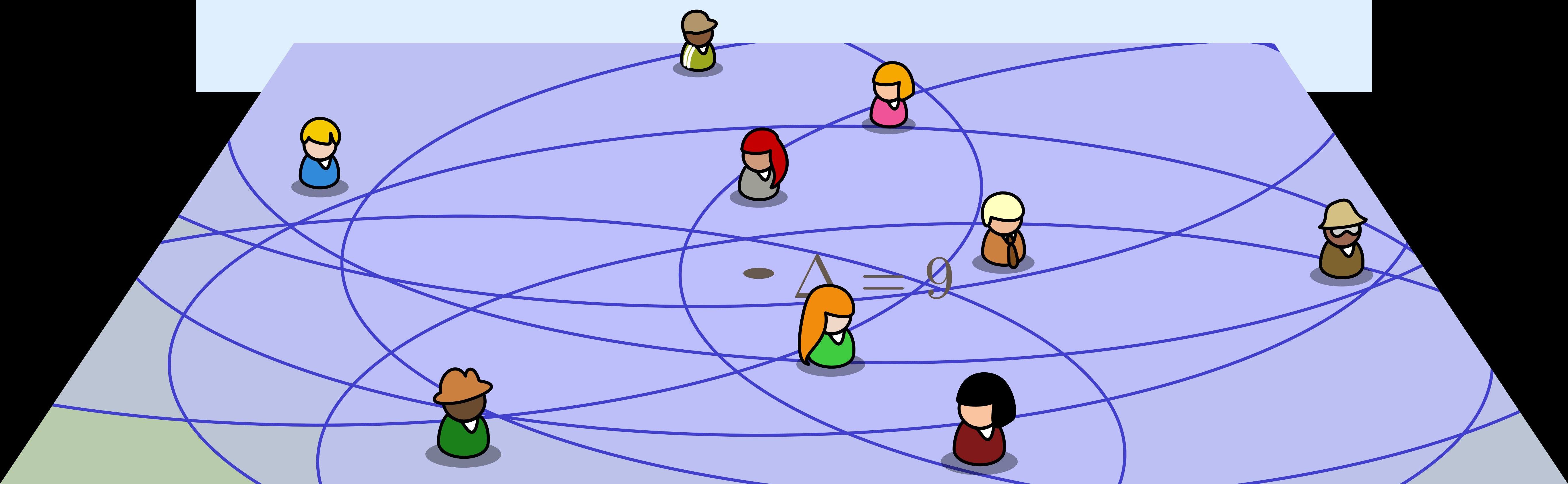
CLAIM: *Resulting ply Δ can be a factor n larger than optimal!*



STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them

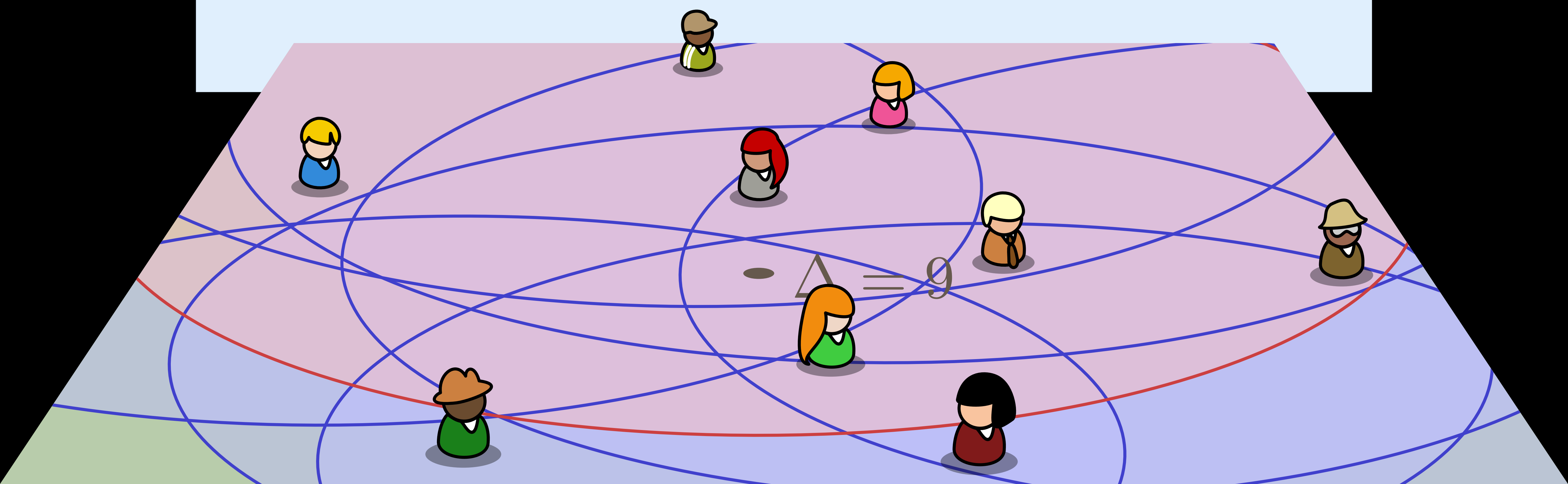
CLAIM: *Resulting ply Δ can be a factor n larger than optimal!*



STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them

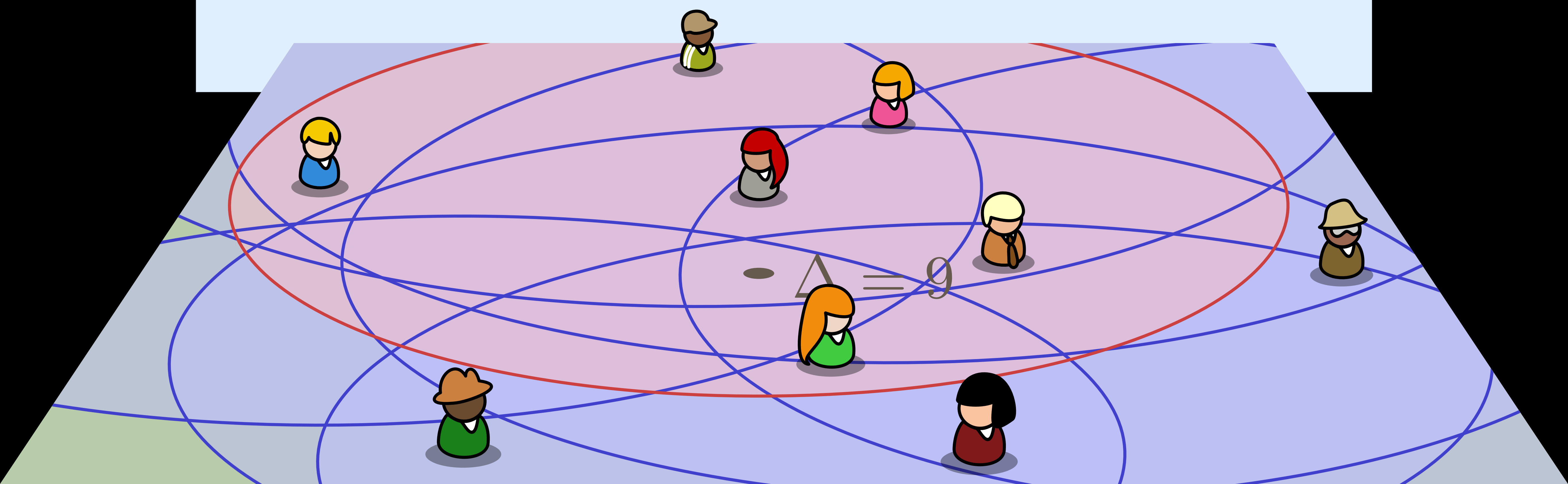
CLAIM: *Resulting ply Δ can be a factor n larger than optimal!*



STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them

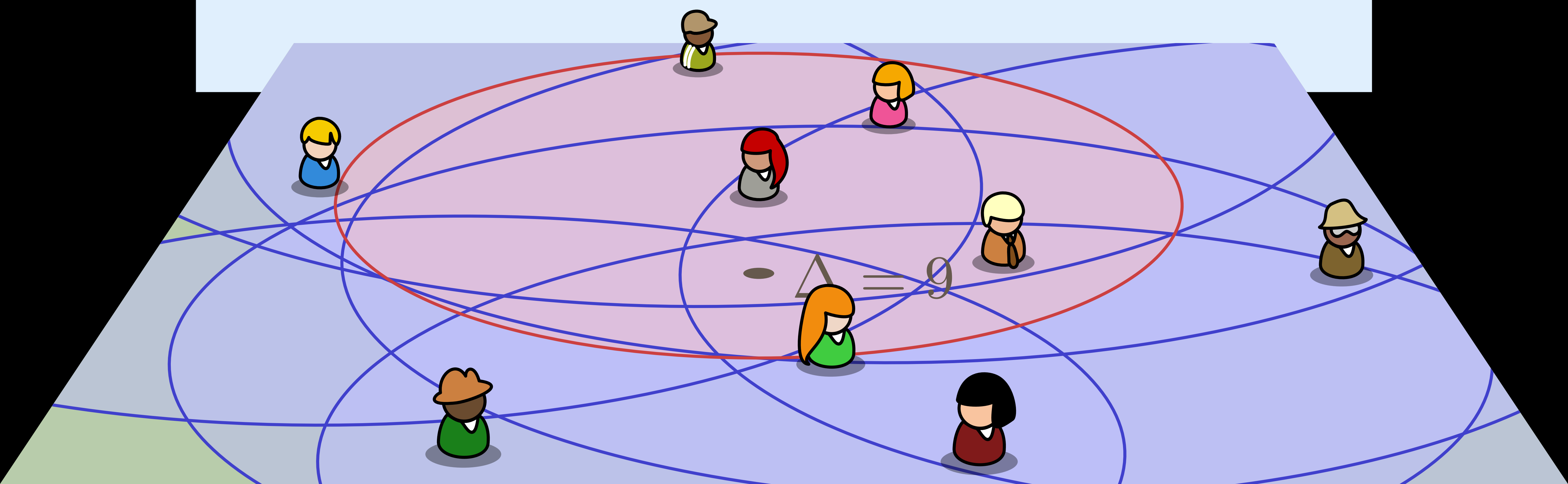
CLAIM: *Resulting ply Δ can be a factor n larger than optimal!*



STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them

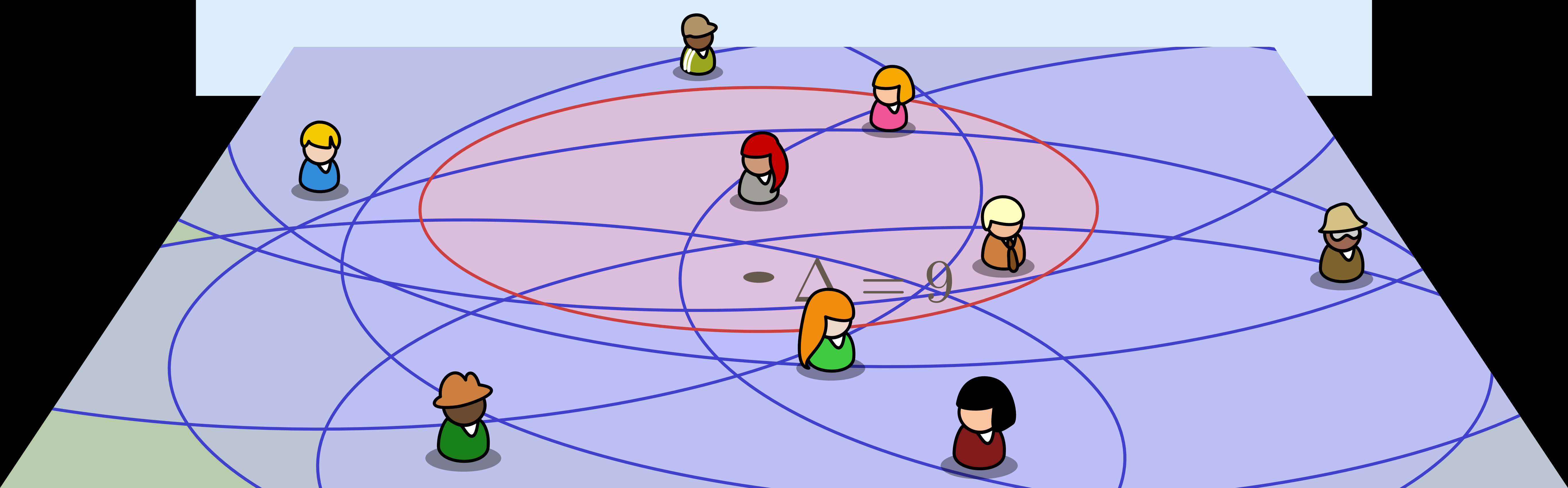
CLAIM: *Resulting ply Δ can be a factor n larger than optimal!*



STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them

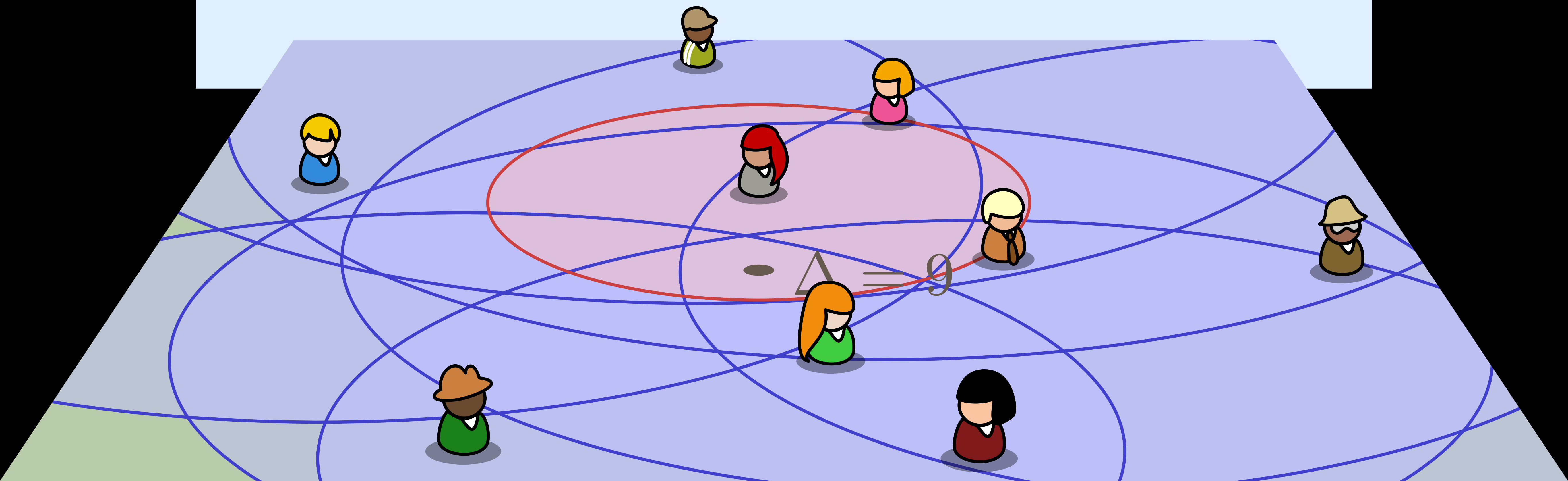
CLAIM: *Resulting ply Δ can be a factor n larger than optimal!*



STRATEGY ATTEMPT 1

- Let's just always query the region with the largest ply.
  - Let δ_i be the max ply in region R_i
  - Find region R_i with largest δ_i
  - Of course, R_i is not unique
  - Query any of them

CLAIM: *Resulting ply Δ can be a factor n larger than optimal!*



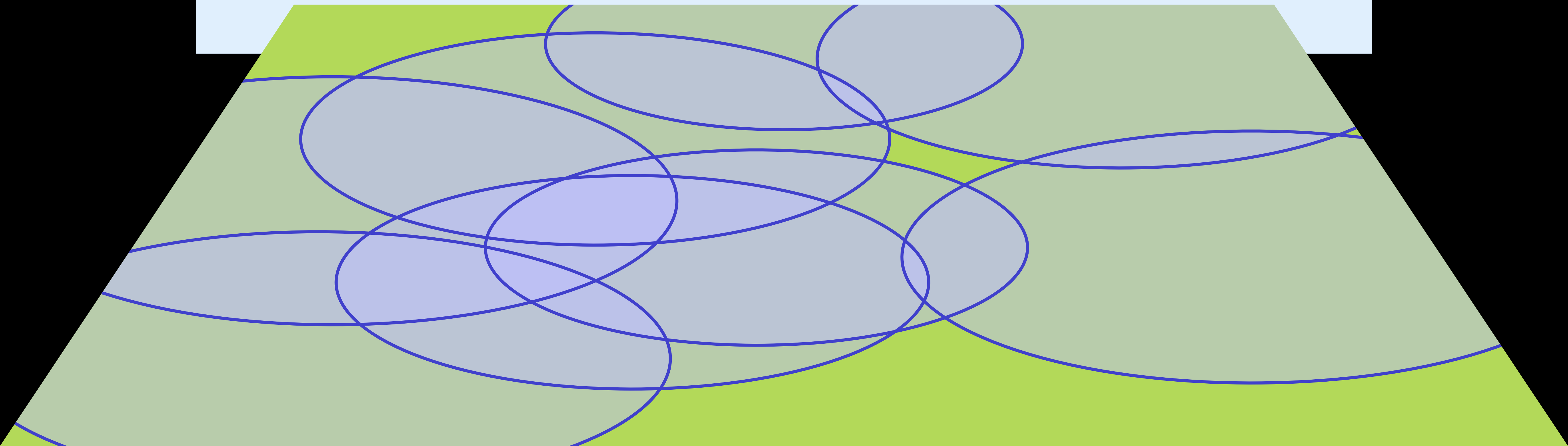
STRATEGY ATTEMPT 2

STRATEGY ATTEMPT 2

- Why not just always query the largest region?

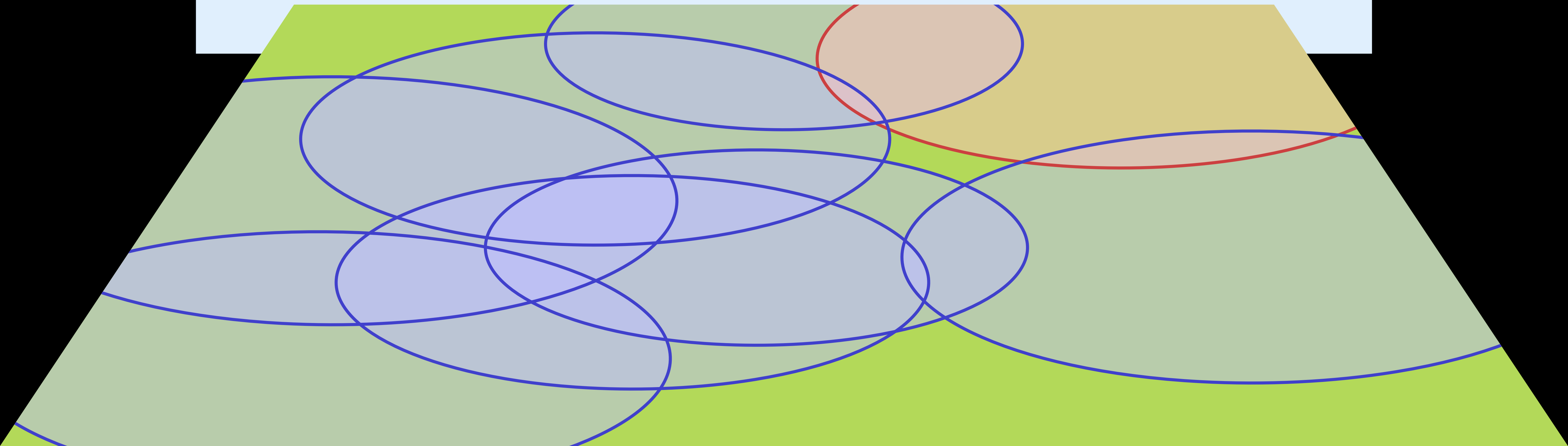
STRATEGY ATTEMPT 2

- Why not just always query the largest region?



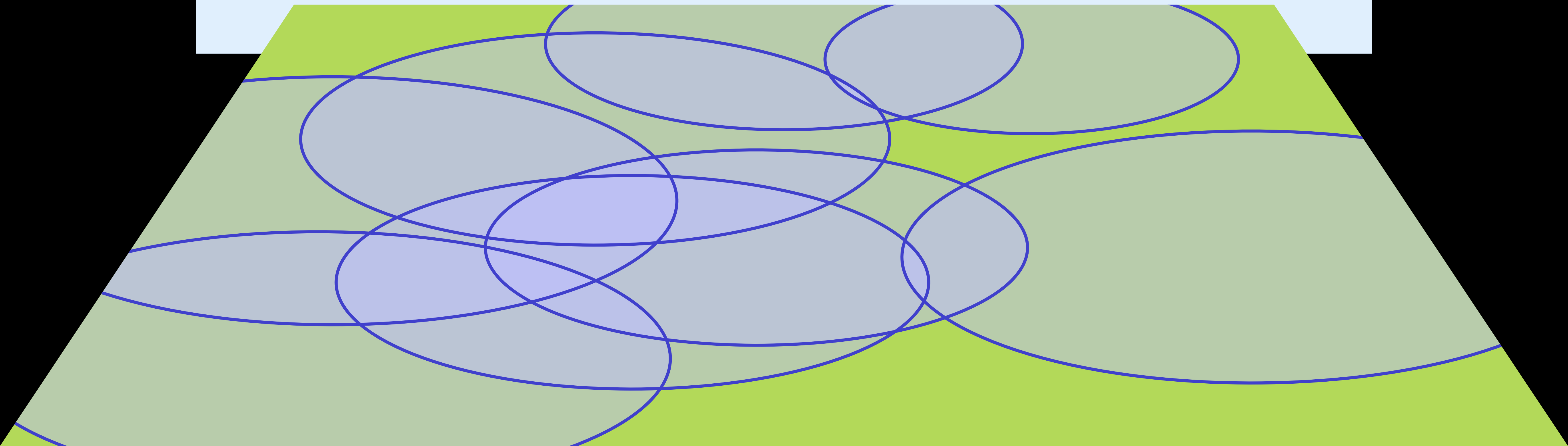
STRATEGY ATTEMPT 2

- Why not just always query the largest region?



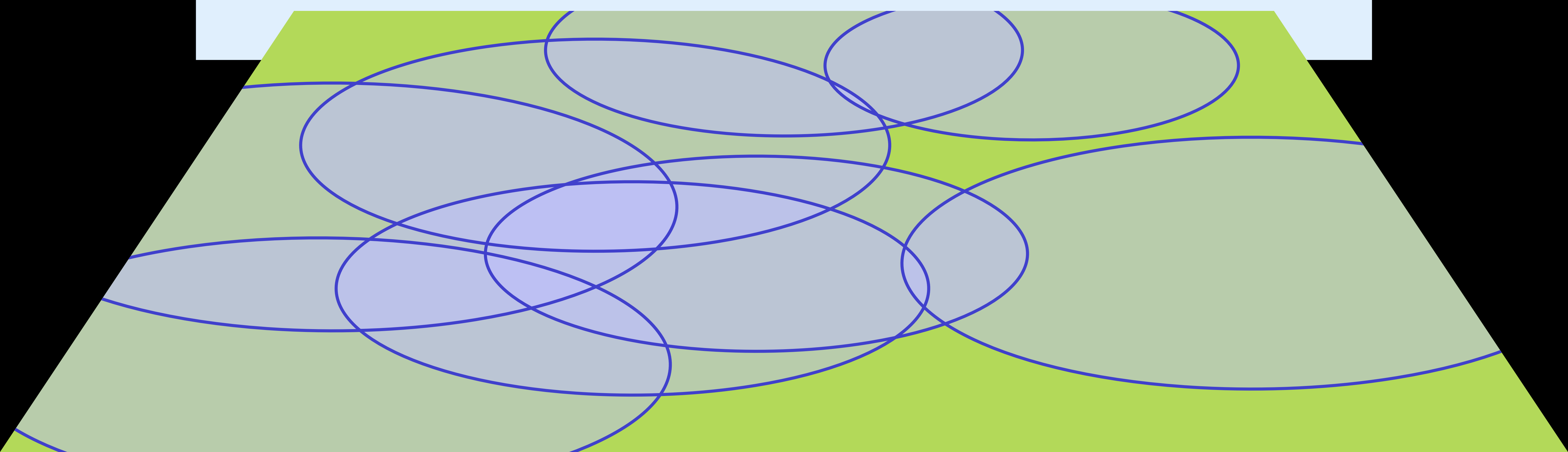
STRATEGY ATTEMPT 2

- Why not just always query the largest region?



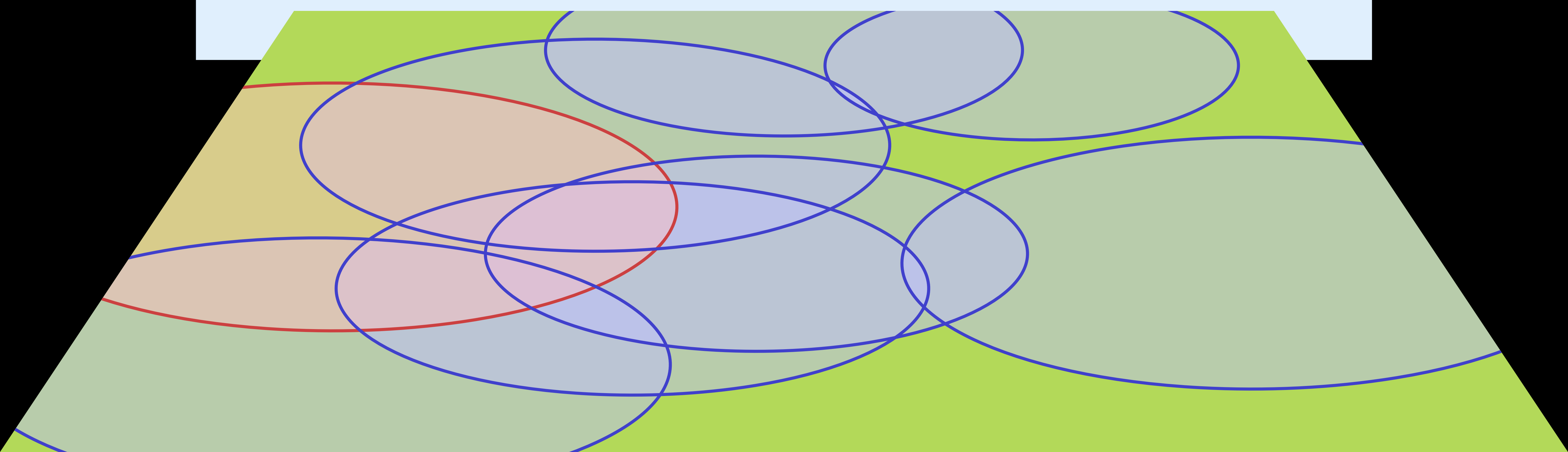
STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice



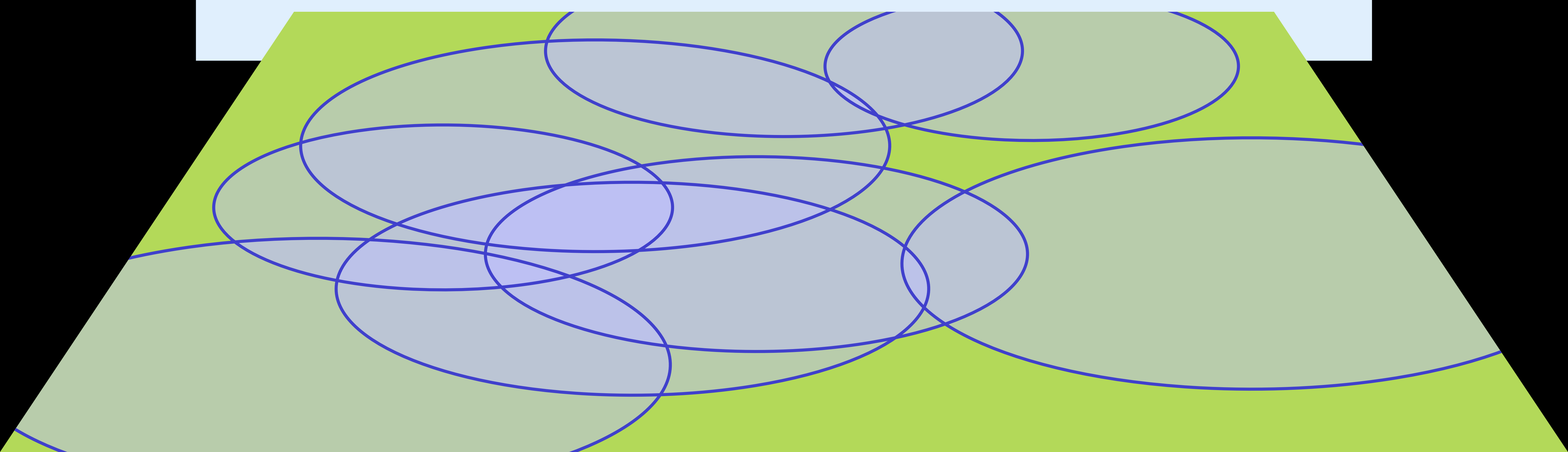
STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice



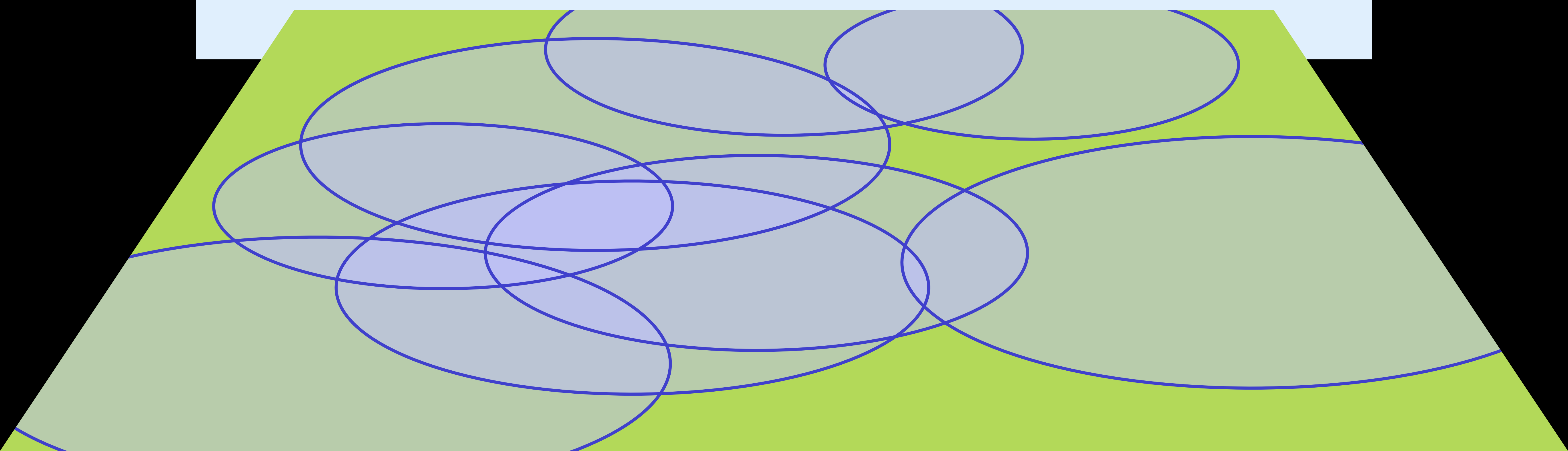
STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice



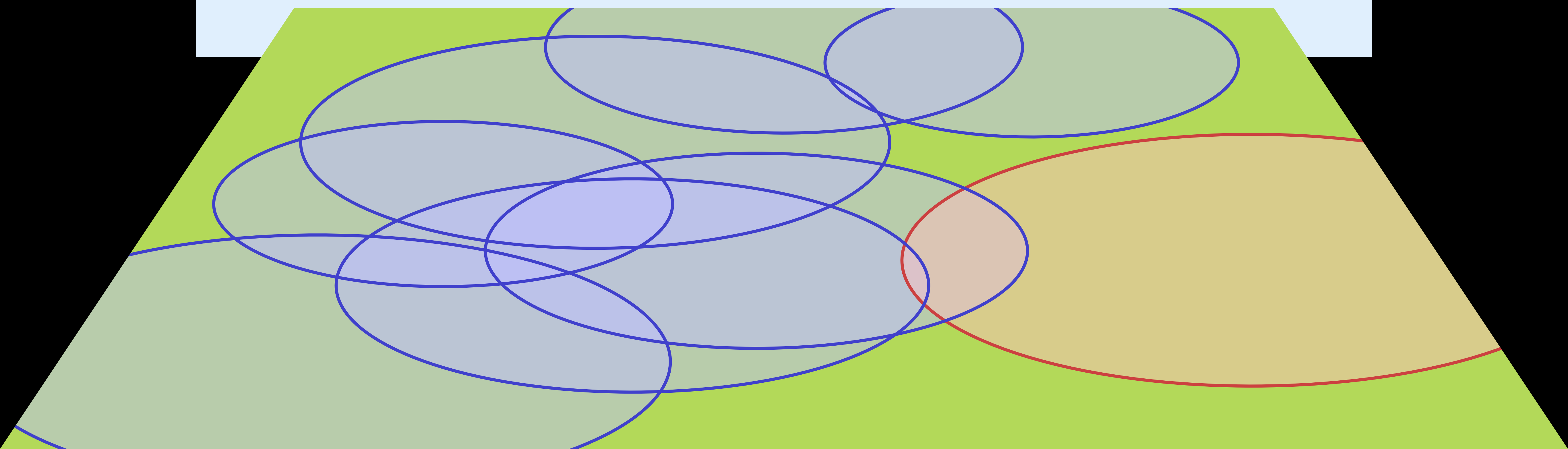
STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy



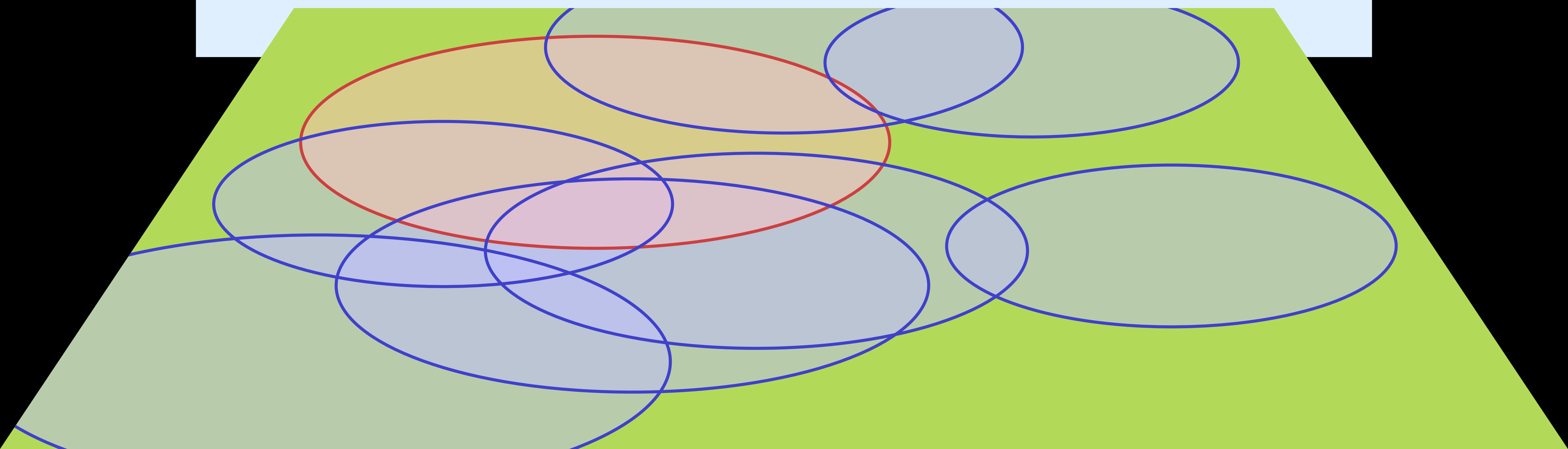
STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy



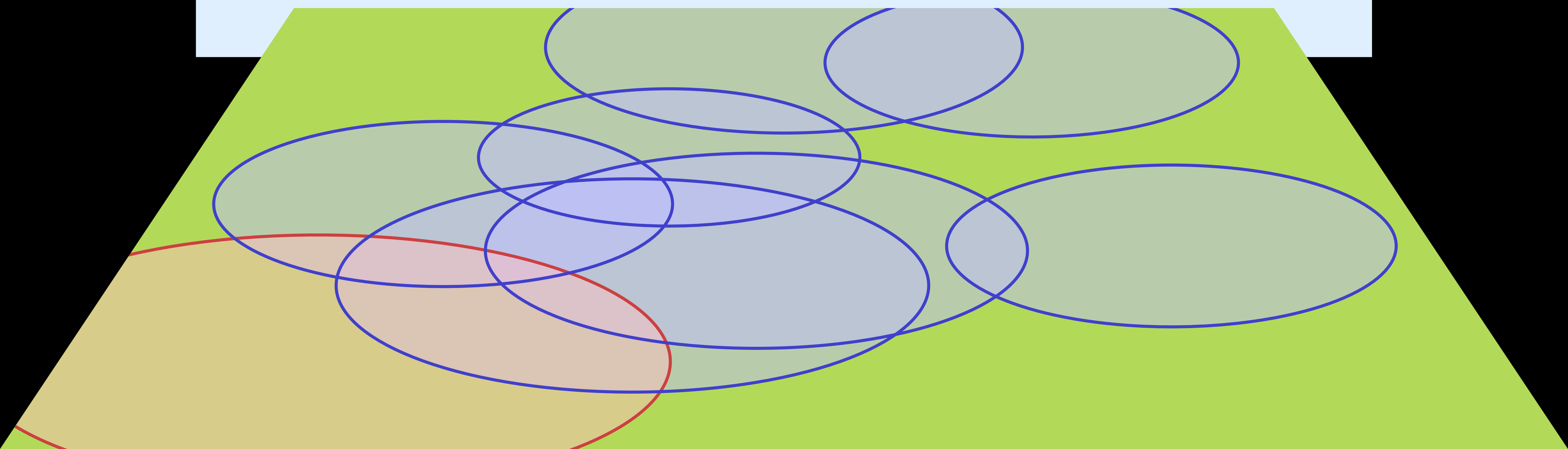
STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy



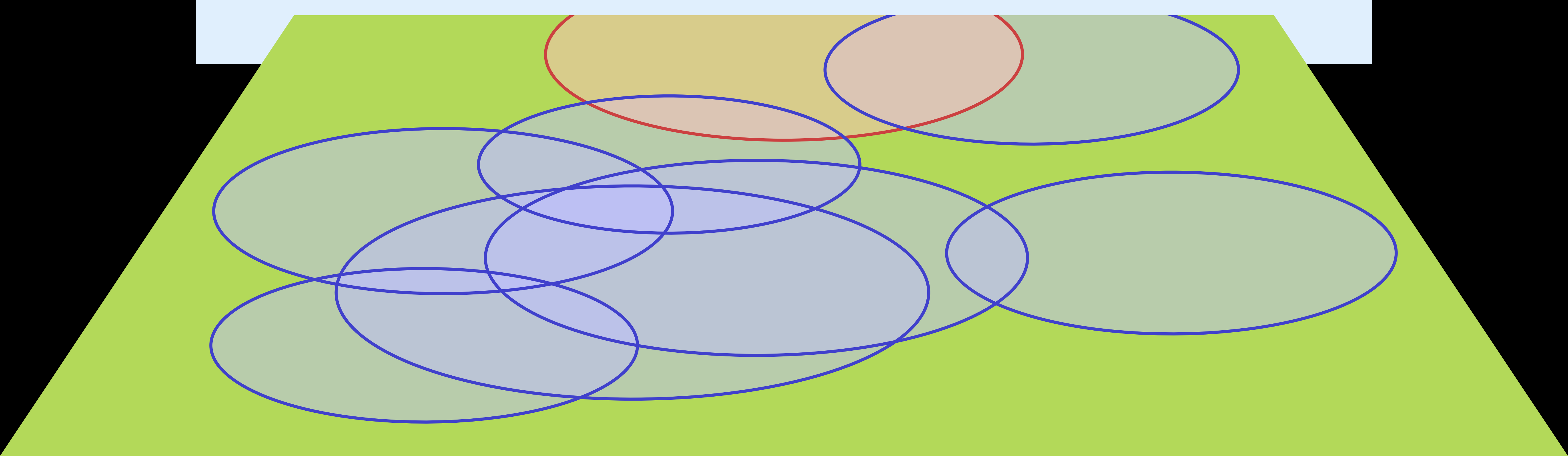
STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy



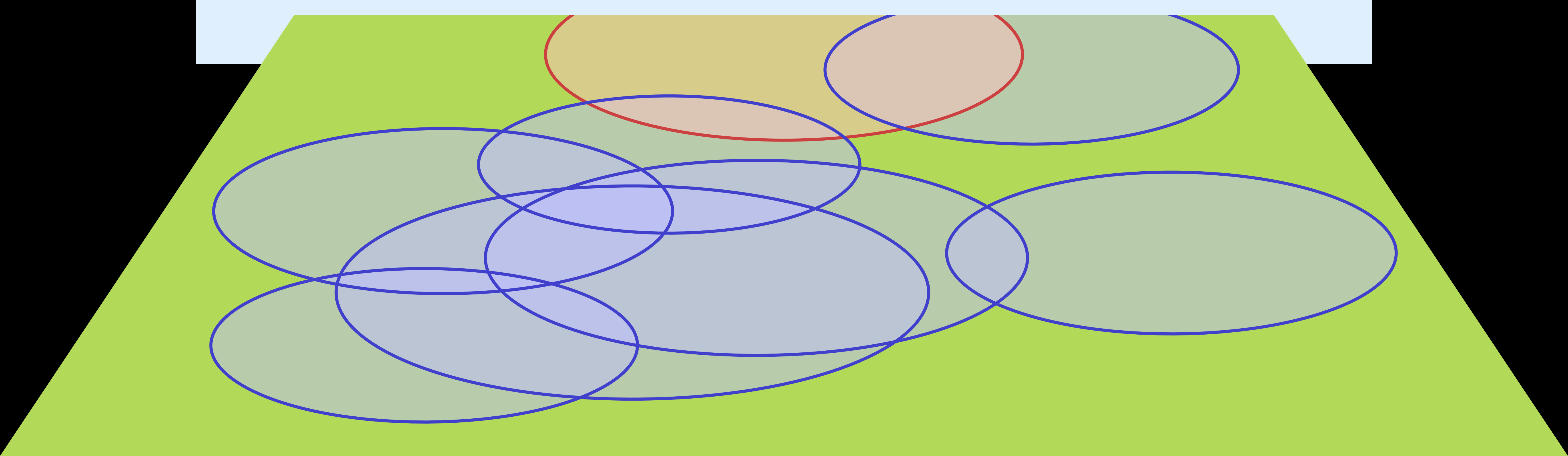
STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy



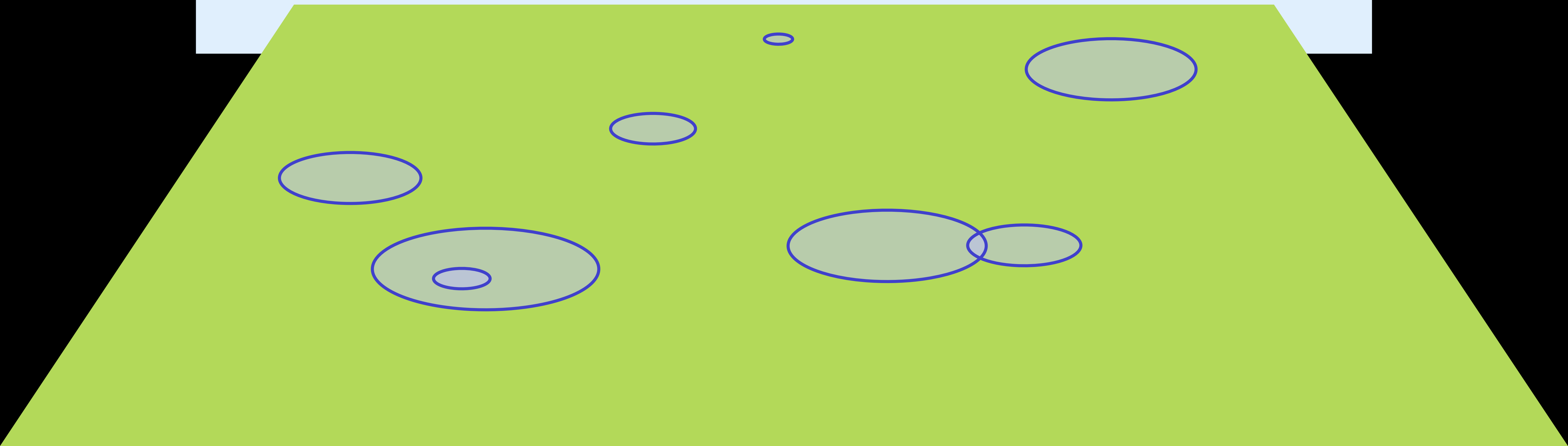
STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy
  - Final regions at t^* will have sizes 1 to n



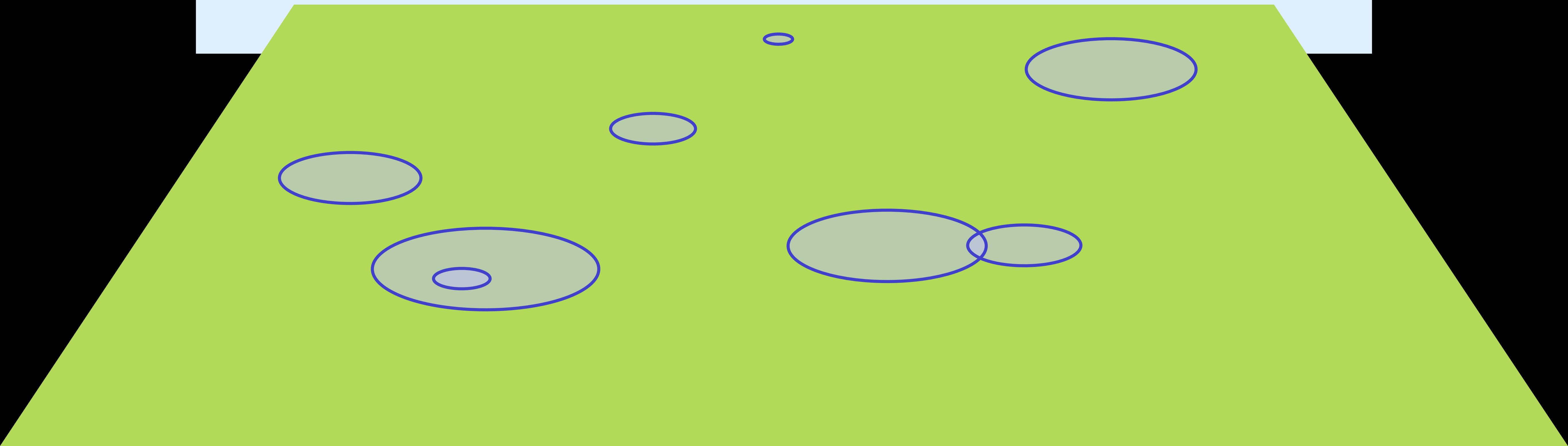
STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy
  - Final regions at t^* will have sizes 1 to n



STRATEGY ATTEMPT 2

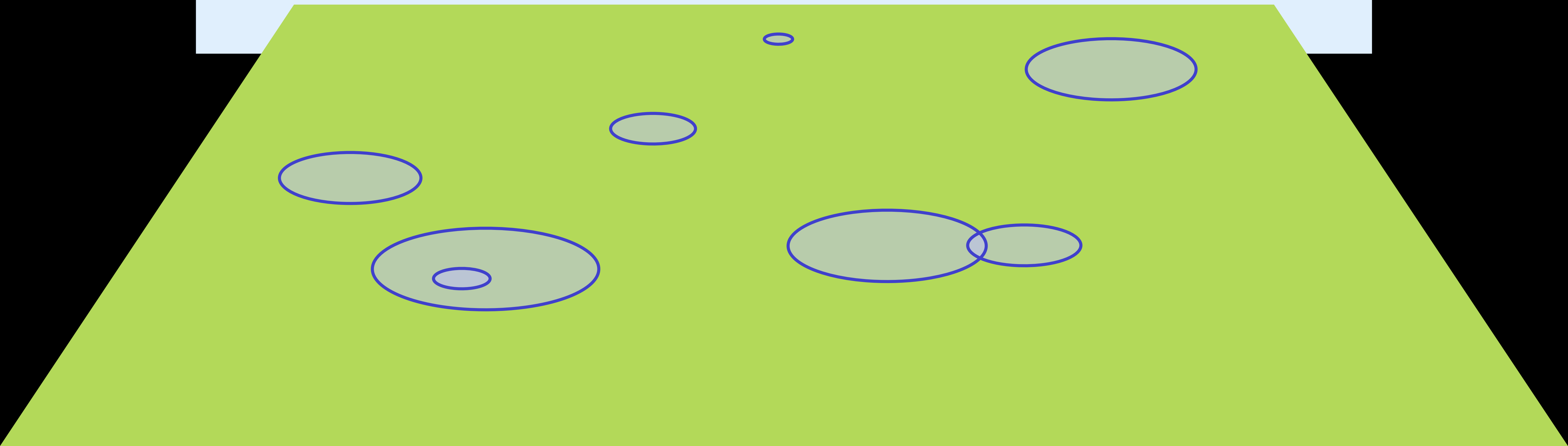
- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy
  - Final regions at t^* will have sizes 1 to n
  - Configuration depends on order



STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy
  - Final regions at t^* will have sizes 1 to n
  - Configuration depends on order

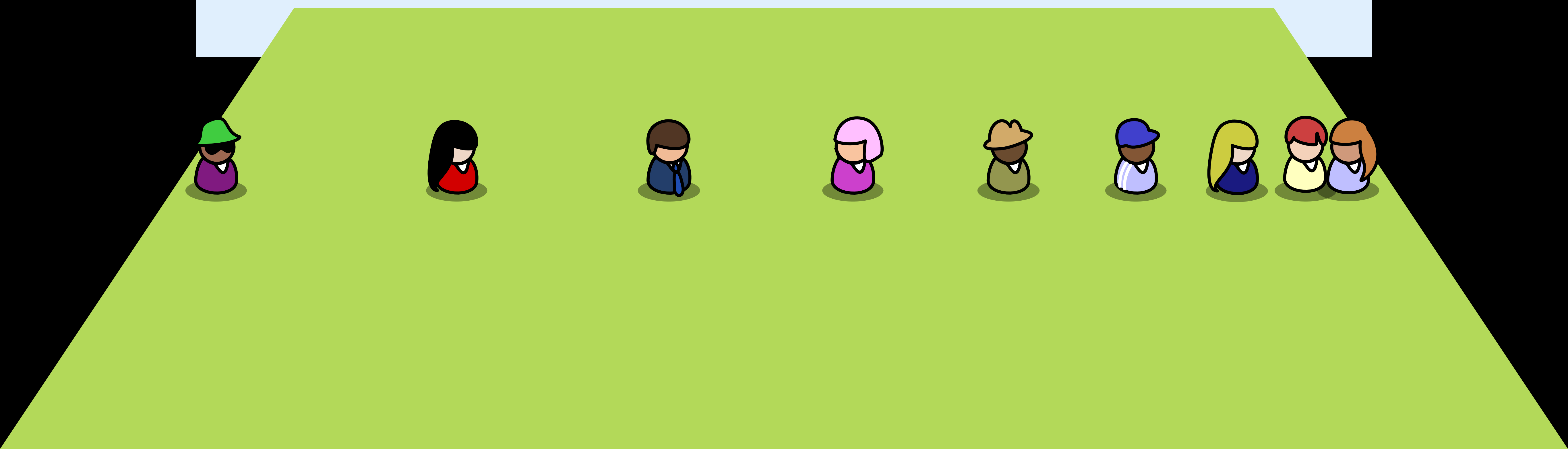
CLAIM: *Resulting ply Δ can be a factor $\sqrt{n}$ larger than optimal!*



STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy
  - Final regions at t^* will have sizes 1 to n
  - Configuration depends on order

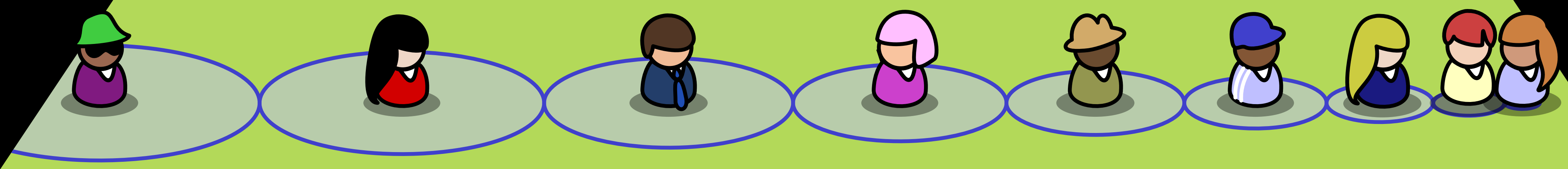
CLAIM: *Resulting ply Δ can be a factor $\sqrt{n}$ larger than optimal!*



STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy
  - Final regions at t^* will have sizes 1 to n
  - Configuration depends on order

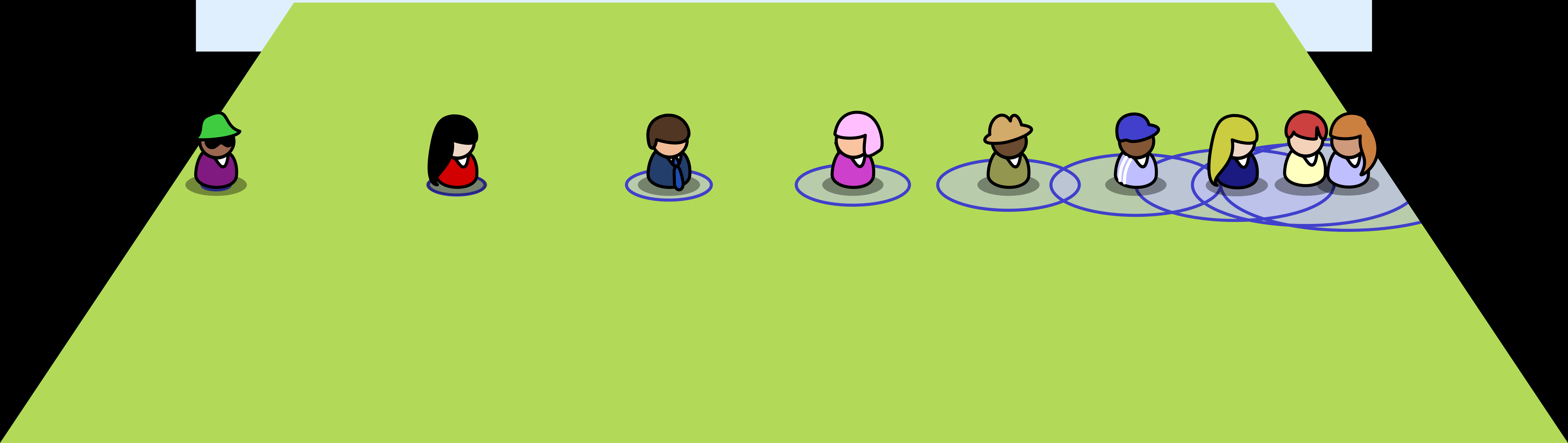
CLAIM: *Resulting ply Δ can be a factor $\sqrt{n}$ larger than optimal!*



STRATEGY ATTEMPT 2

- Why not just always query the largest region?
  - Never query the same region twice
  - Effectively, query cyclicy
  - Final regions at t^* will have sizes 1 to n
  - Configuration depends on order

CLAIM: *Resulting ply Δ can be a factor $\sqrt{n}$ larger than optimal!*



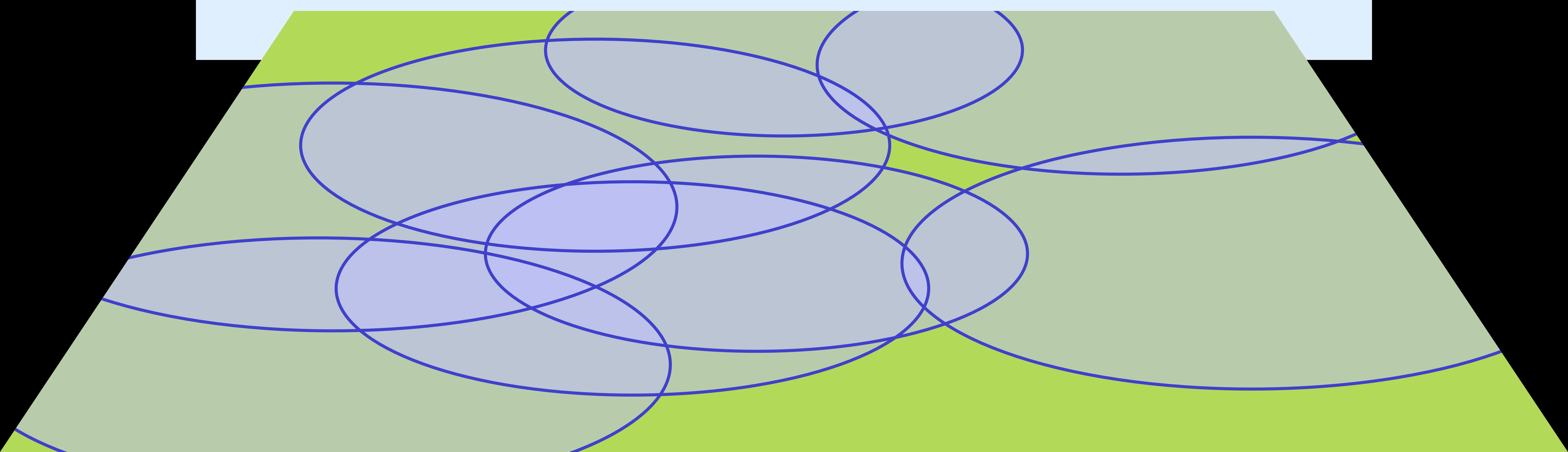
STRATEGY ATTEMPT 3

STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$

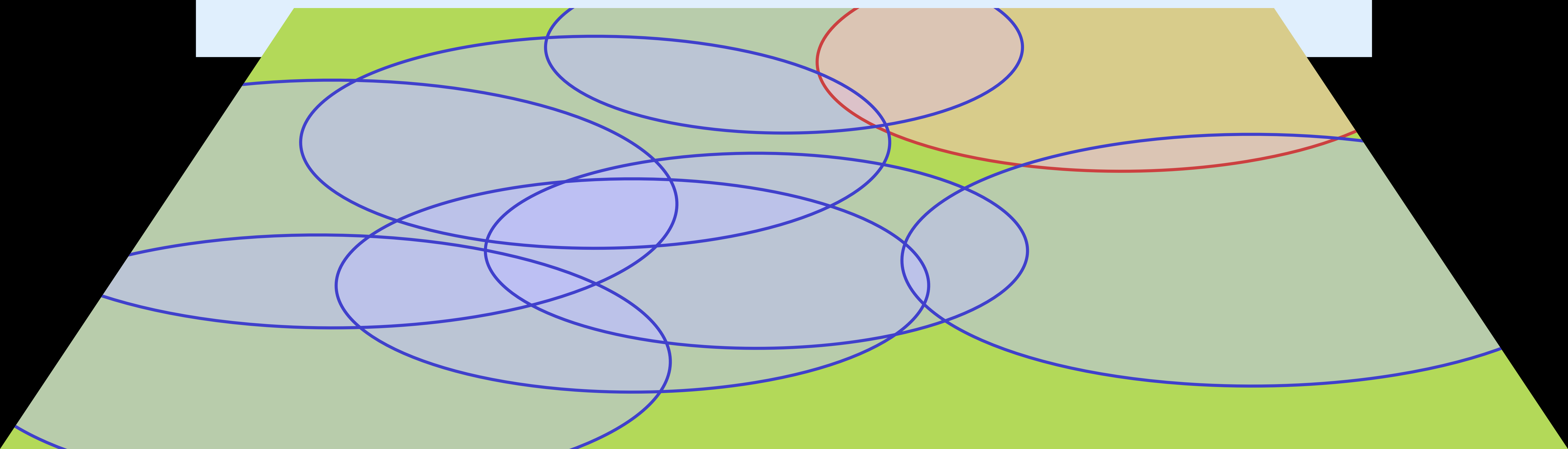
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



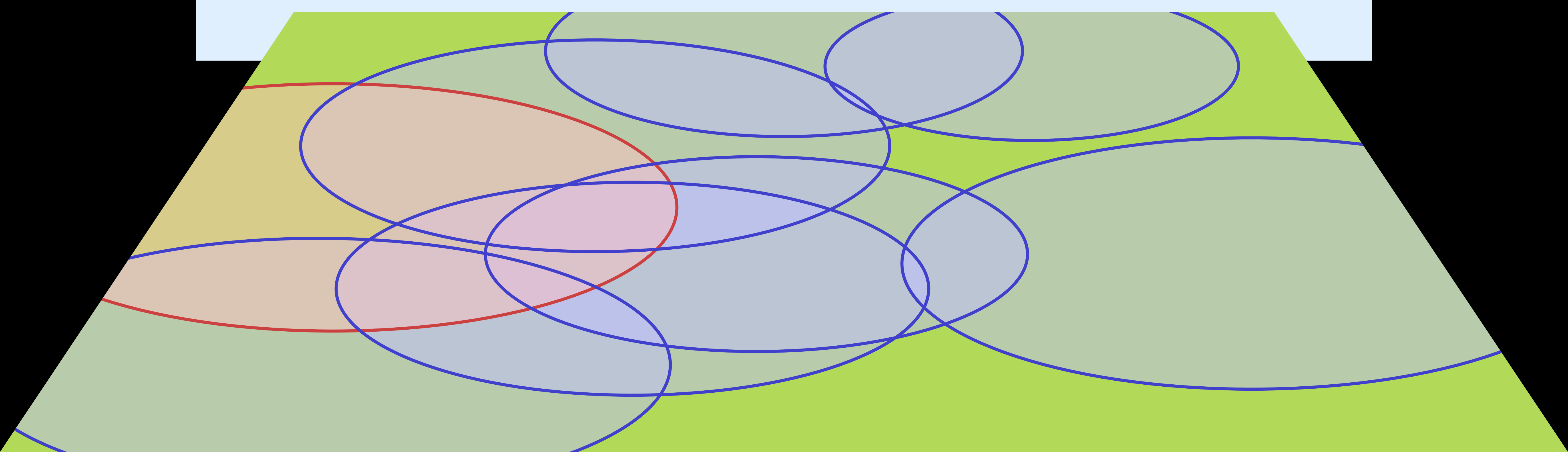
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



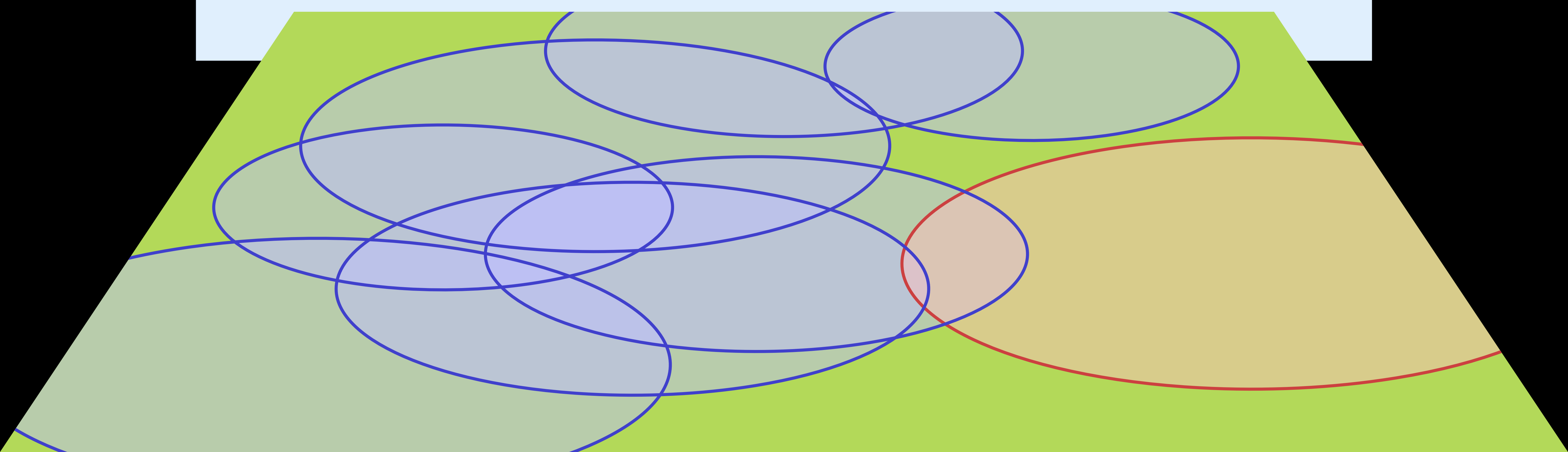
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



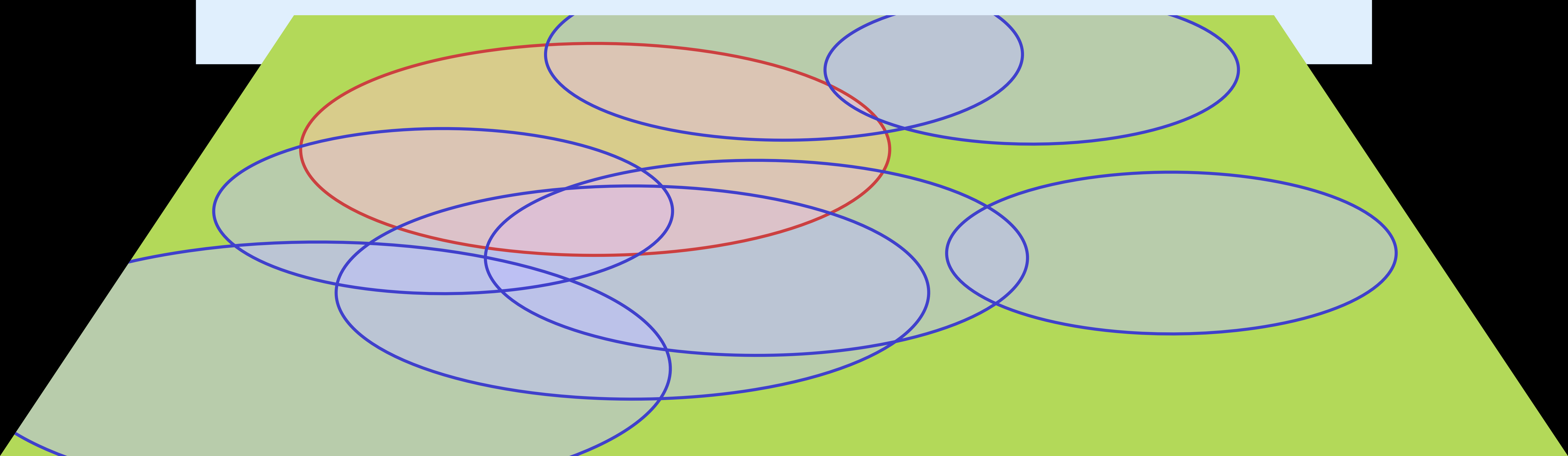
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



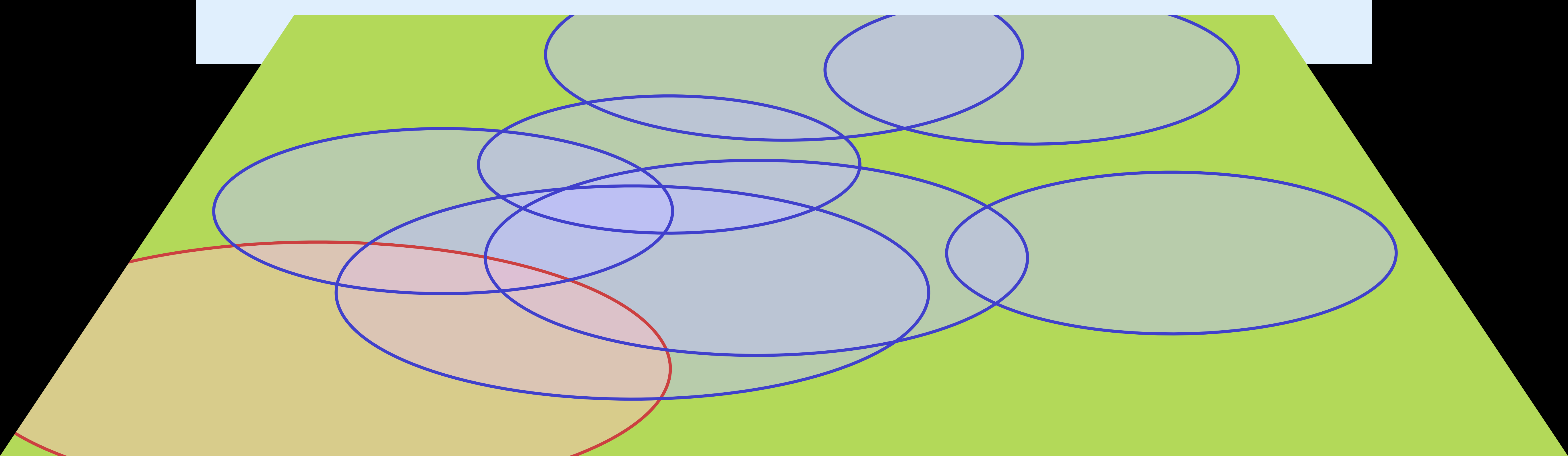
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



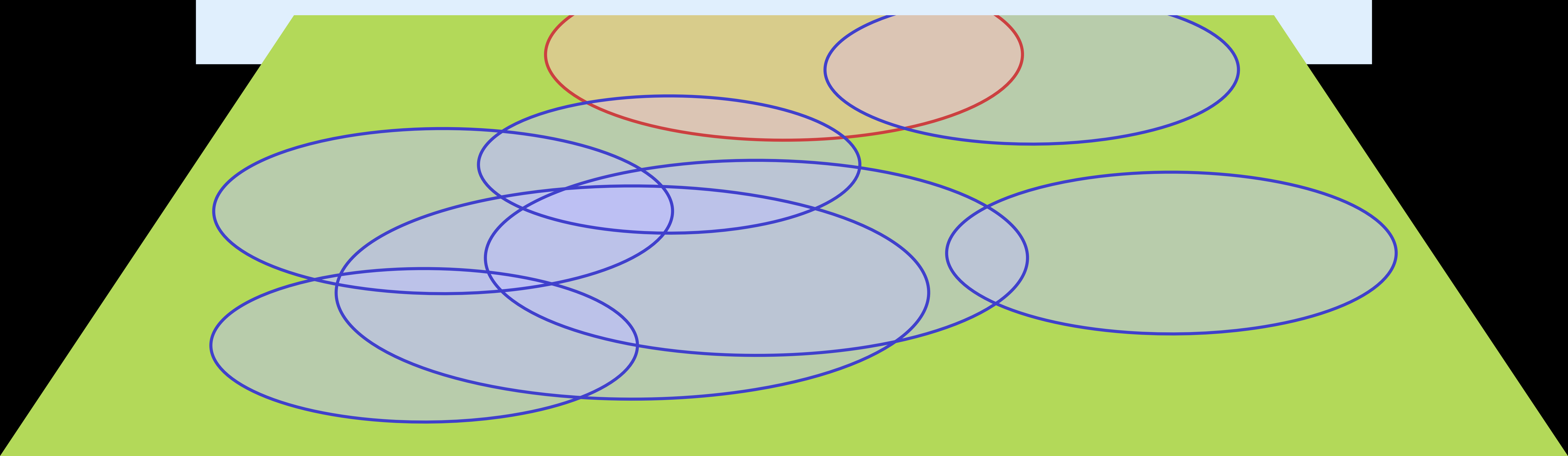
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



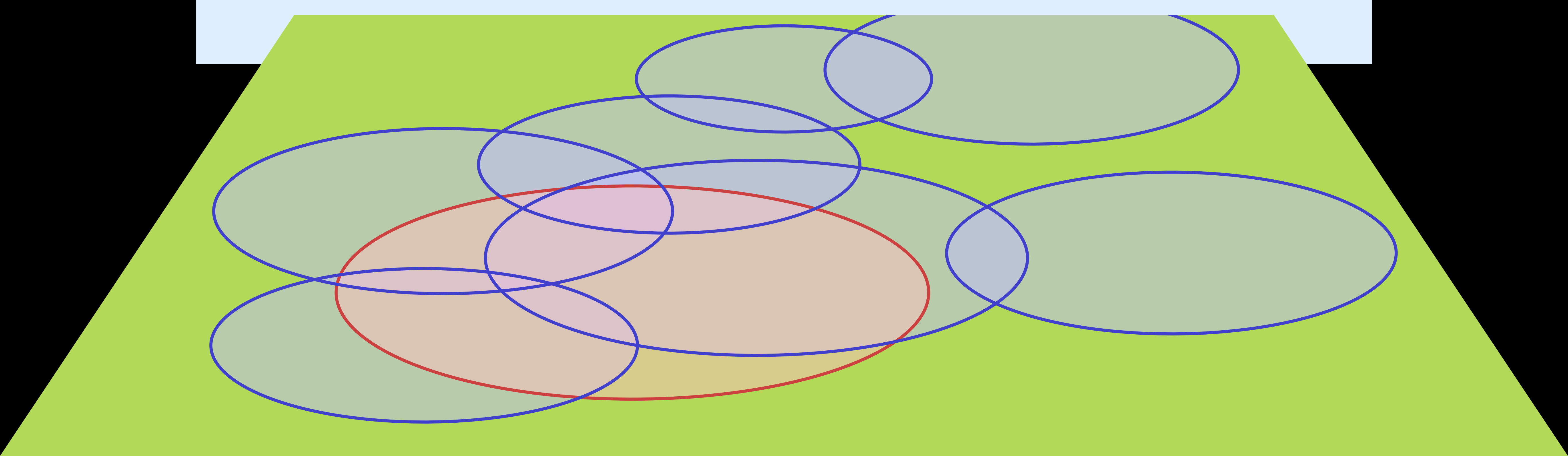
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



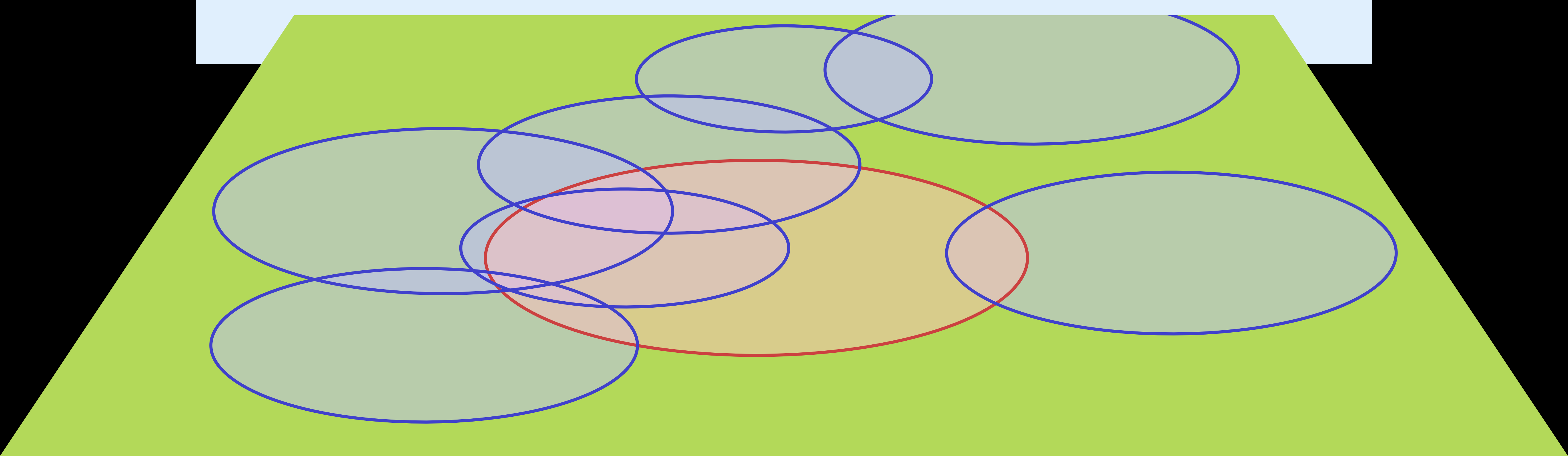
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



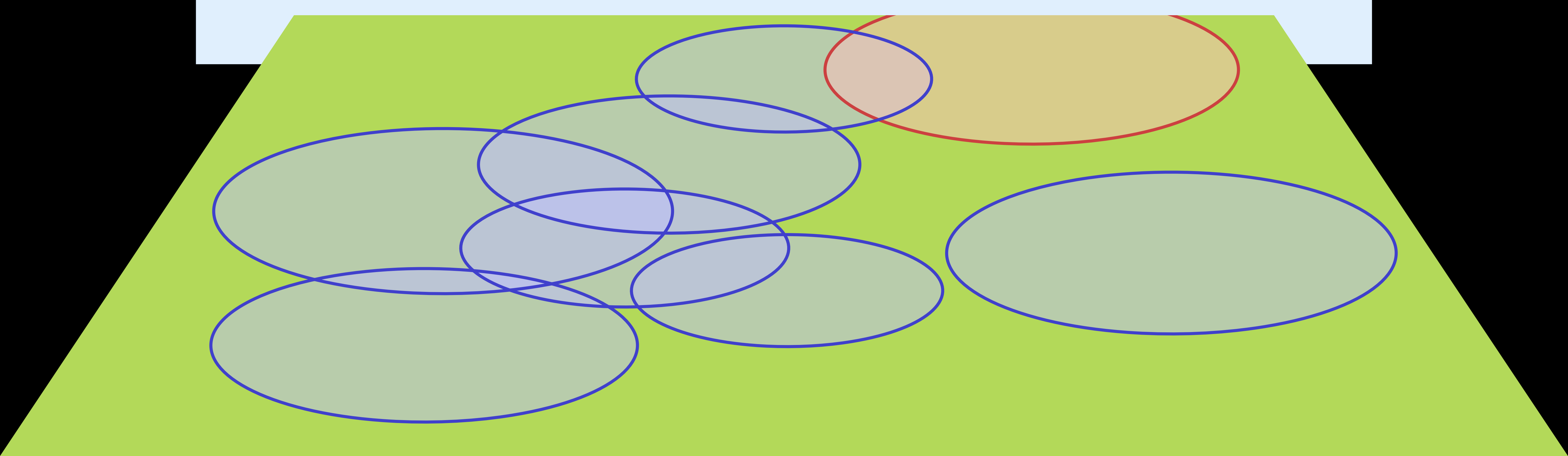
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



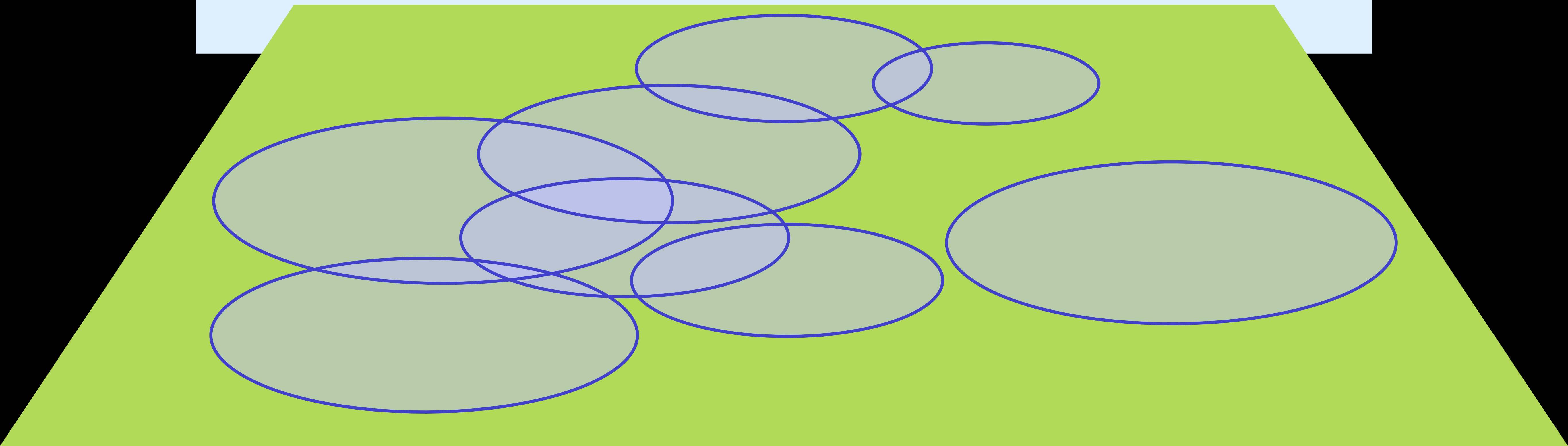
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



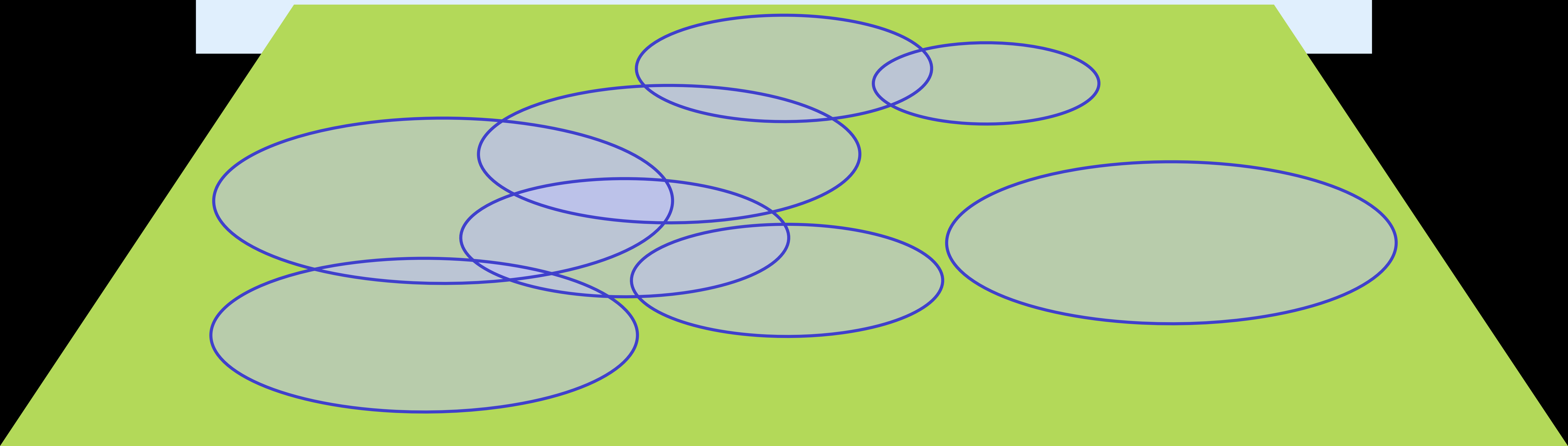
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$



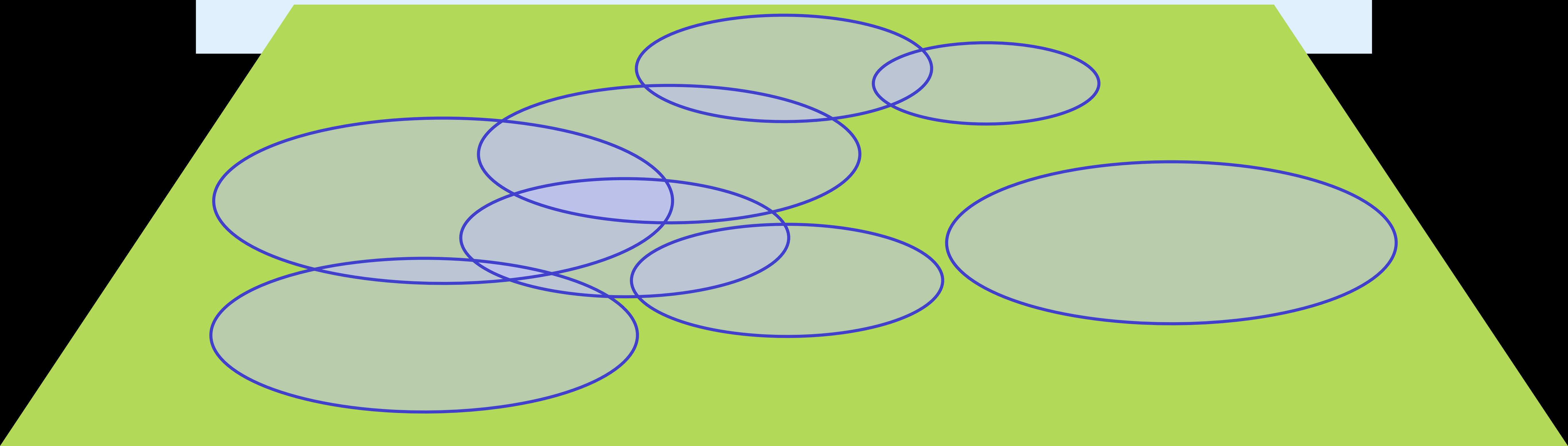
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$



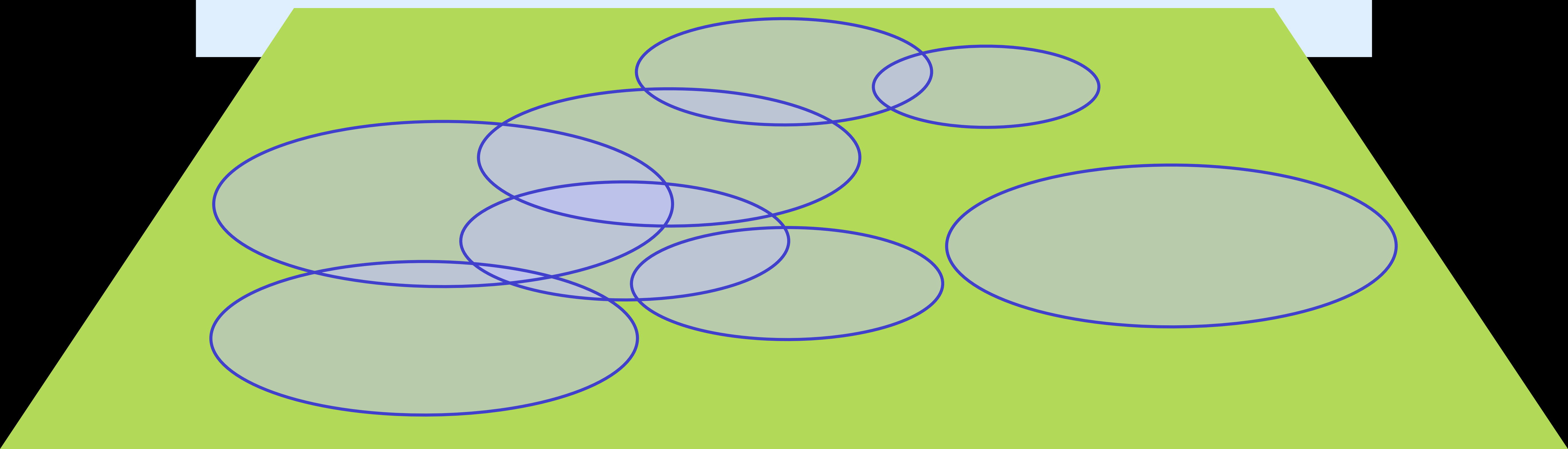
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n + 1$ to $2n$
- While there are still regions left



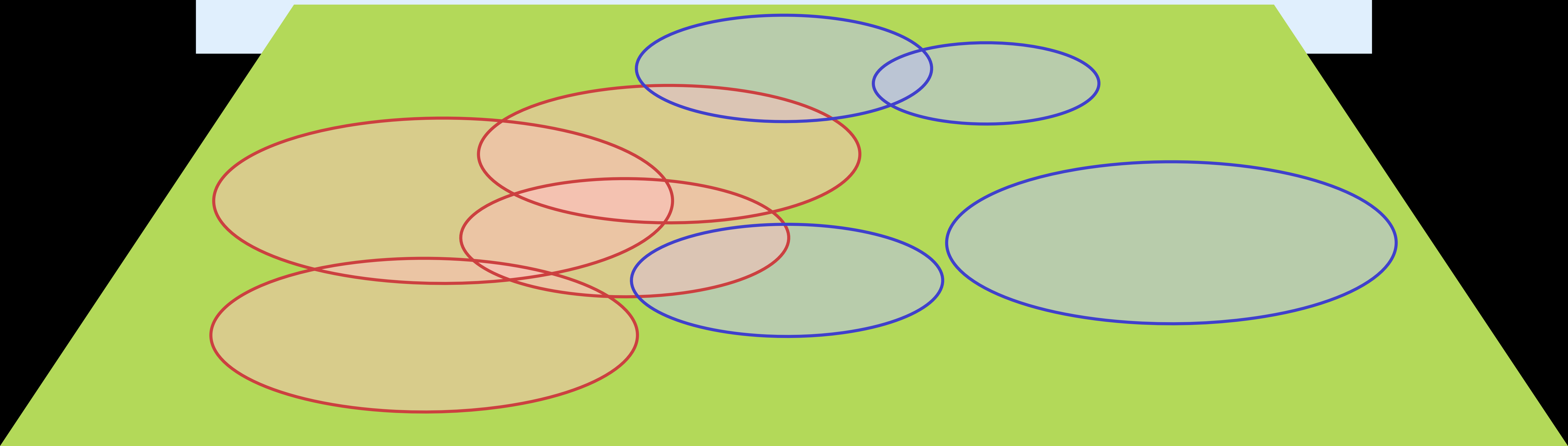
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply



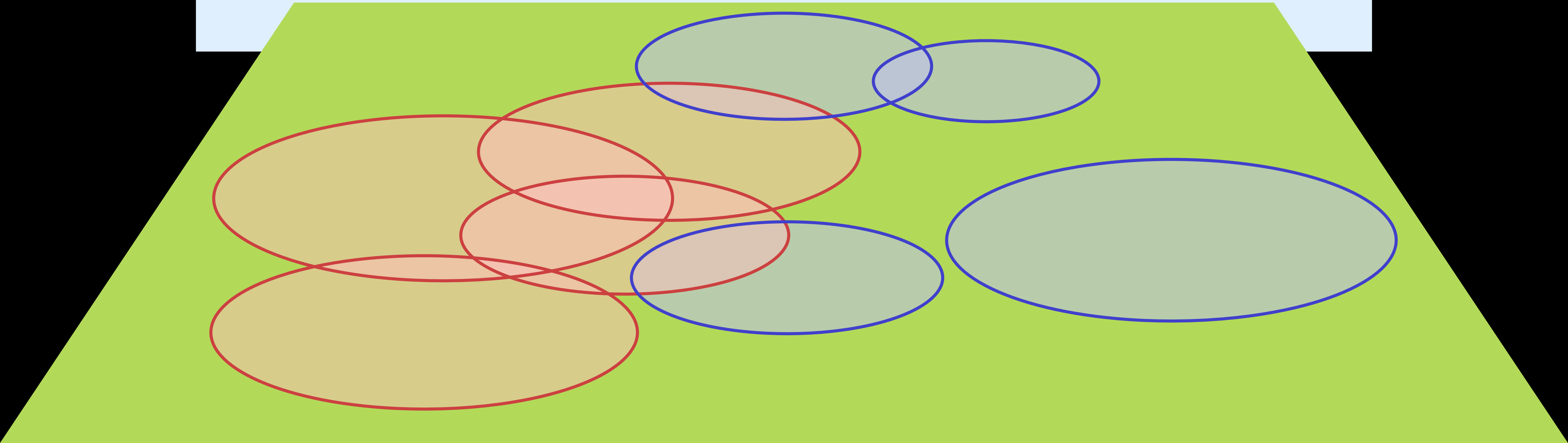
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply



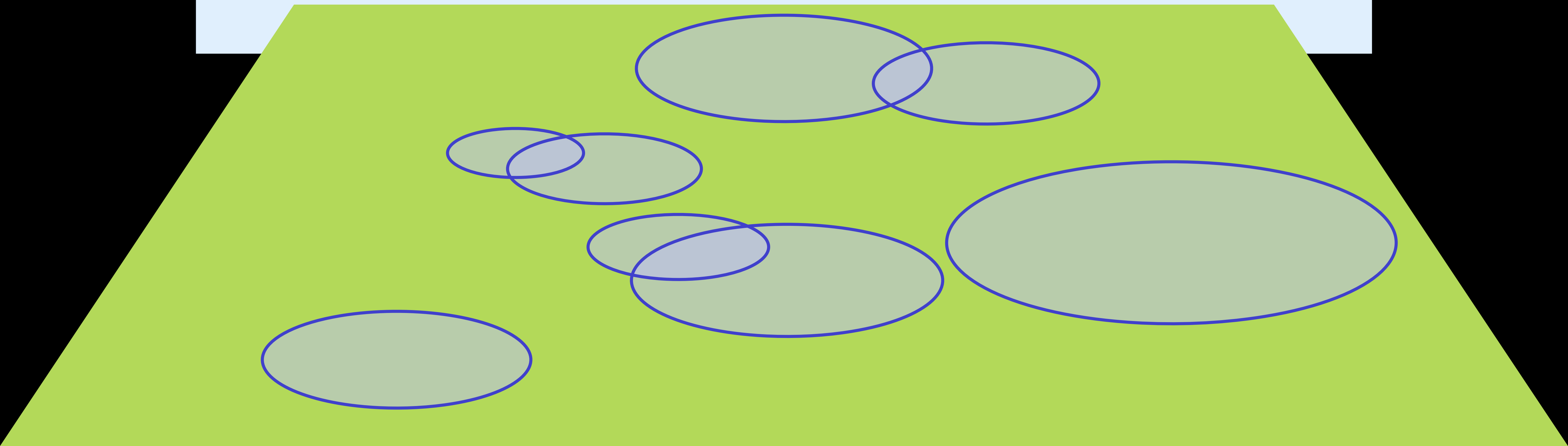
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n + 1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply
  - Query all of them once, in any order



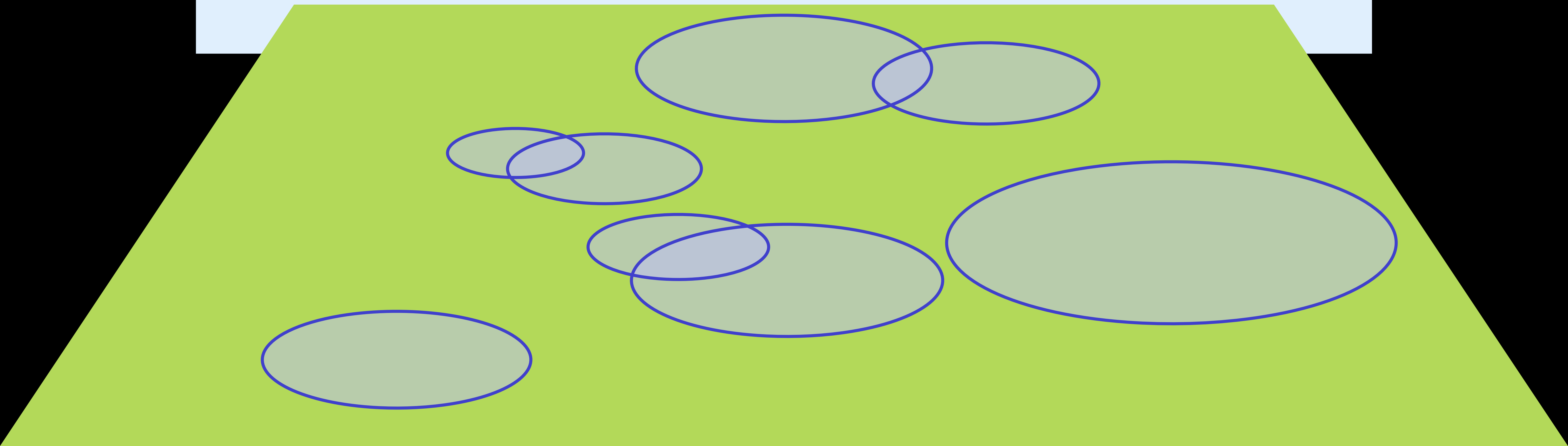
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply
  - Query all of them once, in any order



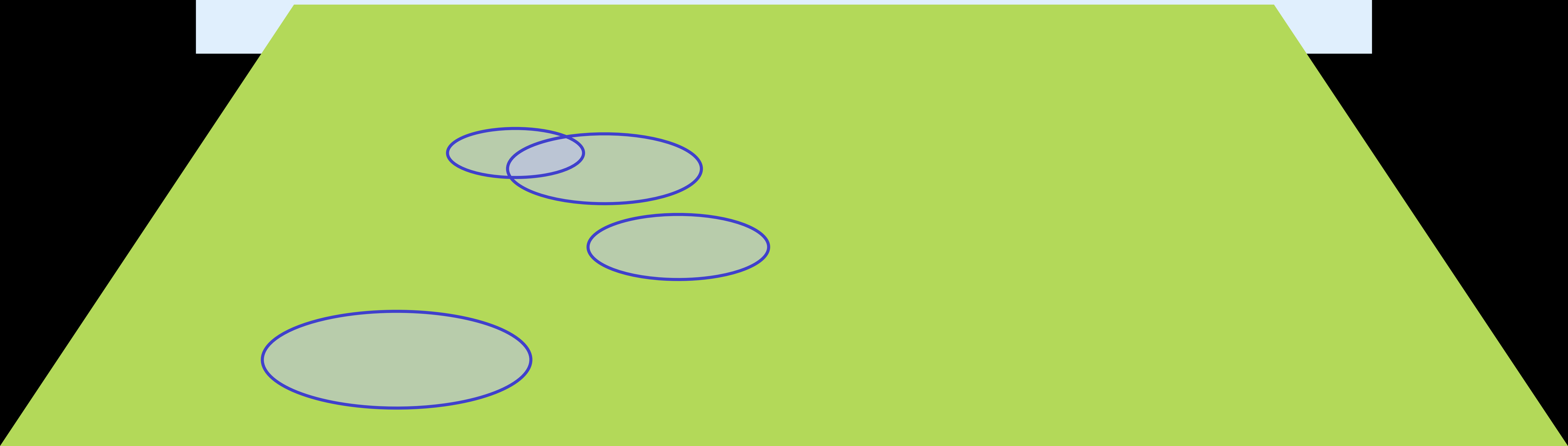
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply
  - Query all of them once, in any order
  - Throw away the others; set $n = \frac{1}{2}n$



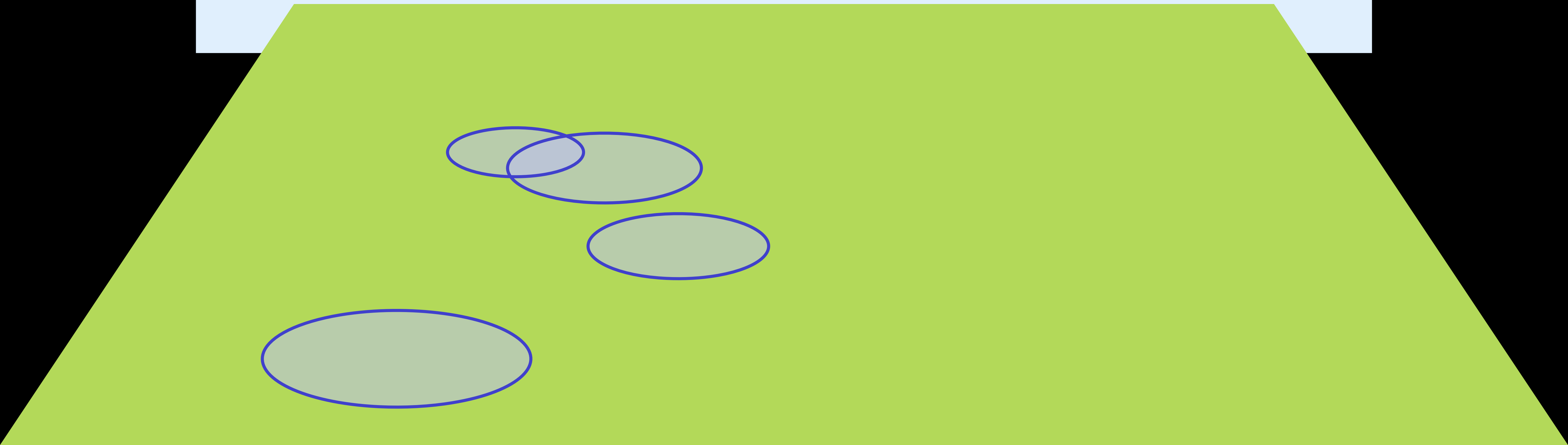
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply
  - Query all of them once, in any order
  - Throw away the others; set $n = \frac{1}{2}n$



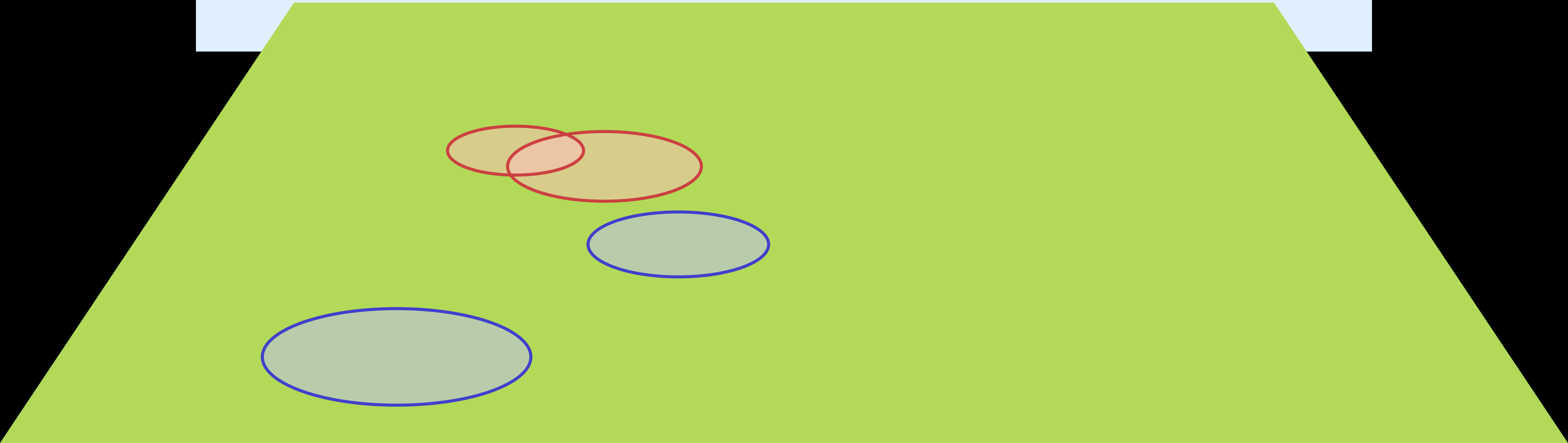
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply
  - Query all of them once, in any order
  - Throw away the others; set $n = \frac{1}{2}n$
  - Repeat



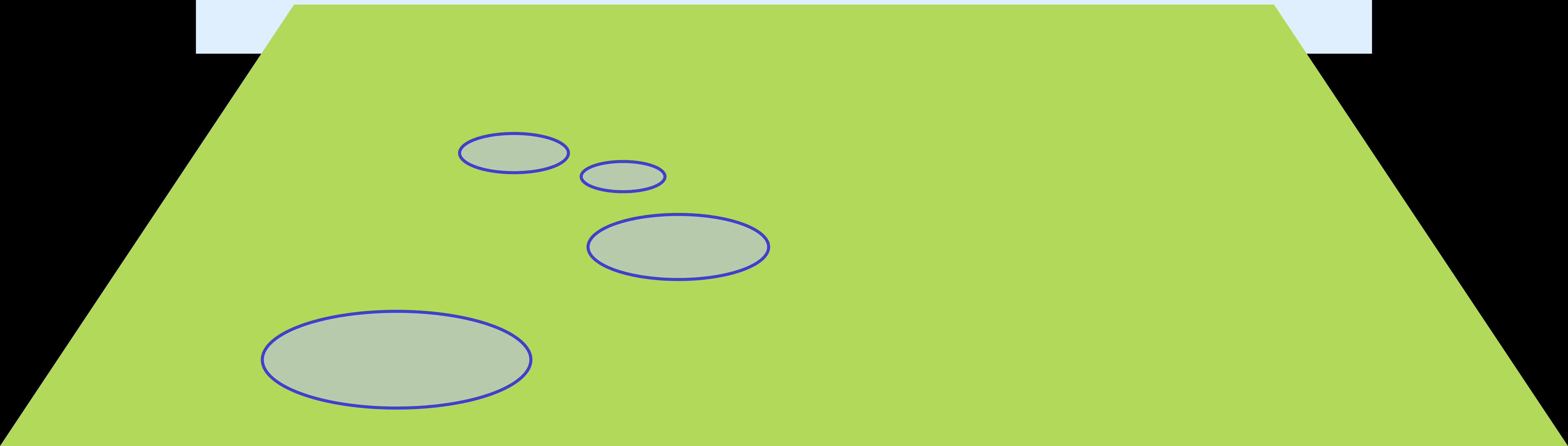
STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply
  - Query all of them once, in any order
  - Throw away the others; set $n = \frac{1}{2}n$
  - Repeat



STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply
  - Query all of them once, in any order
  - Throw away the others; set $n = \frac{1}{2}n$
  - Repeat

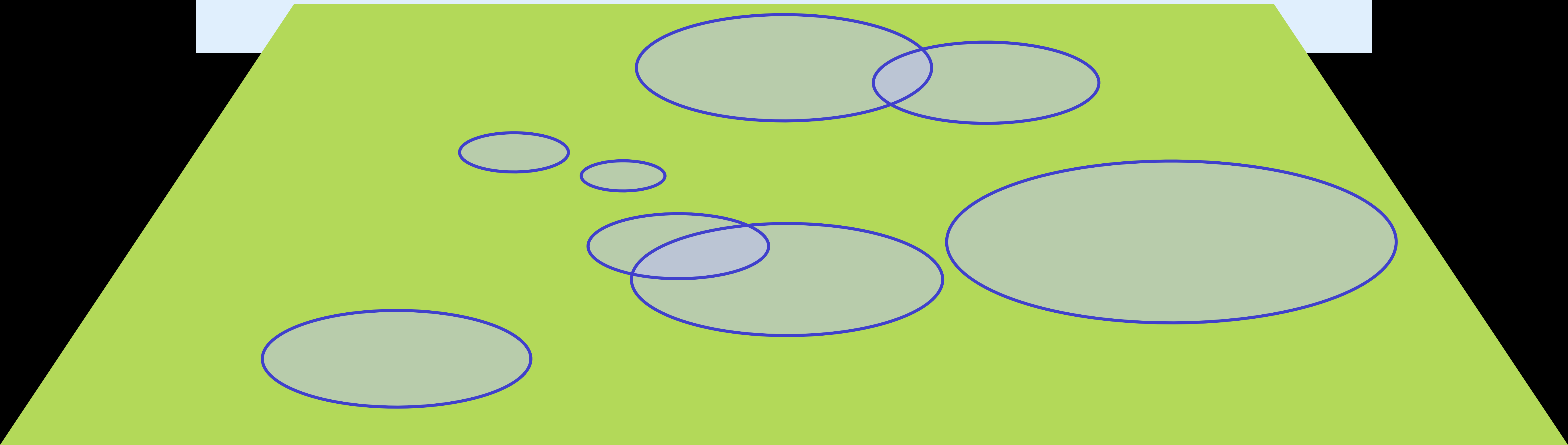


STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n + 1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply
  - Query all of them once, in any order
  - Throw away the others; set $n = \frac{1}{2}n$
  - Repeat

STRATEGY ATTEMPT 3

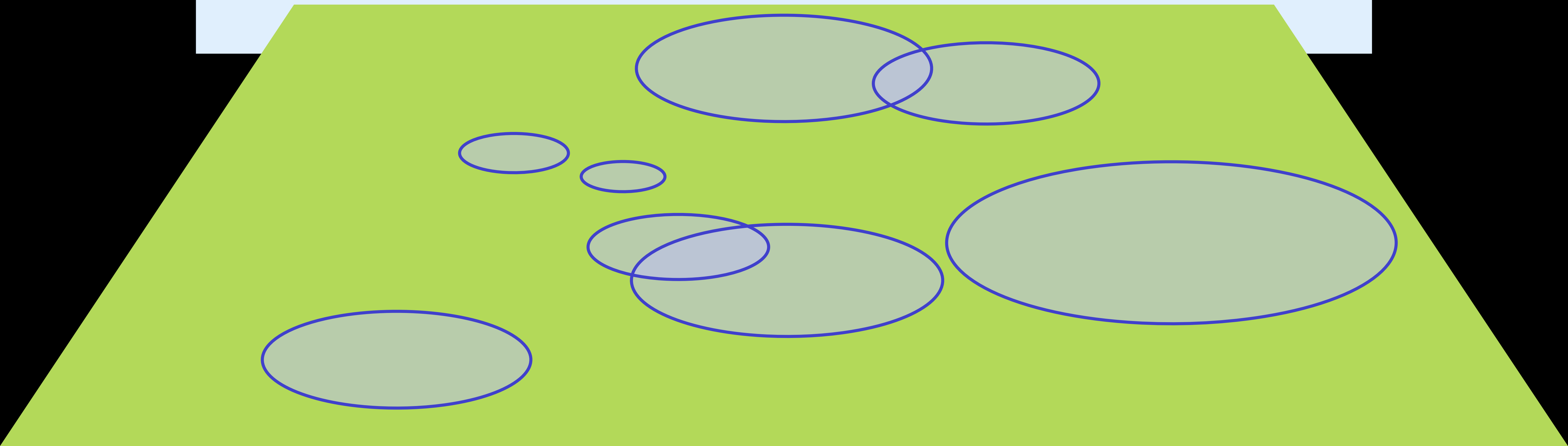
- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply
  - Query all of them once, in any order
  - Throw away the others; set $n = \frac{1}{2}n$
  - Repeat



STRATEGY ATTEMPT 3

- Keep querying cyclicly until $\tau = n$
  - Now regions have sizes $n+1$ to $2n$
- While there are still regions left
  - Find the $\frac{1}{2}n$ regions with highest ply
  - Query all of them once, in any order
  - Throw away the others; set $n = \frac{1}{2}n$
  - Repeat

CLAIM: *Resulting ply Δ is within a constant factor of optimal!*



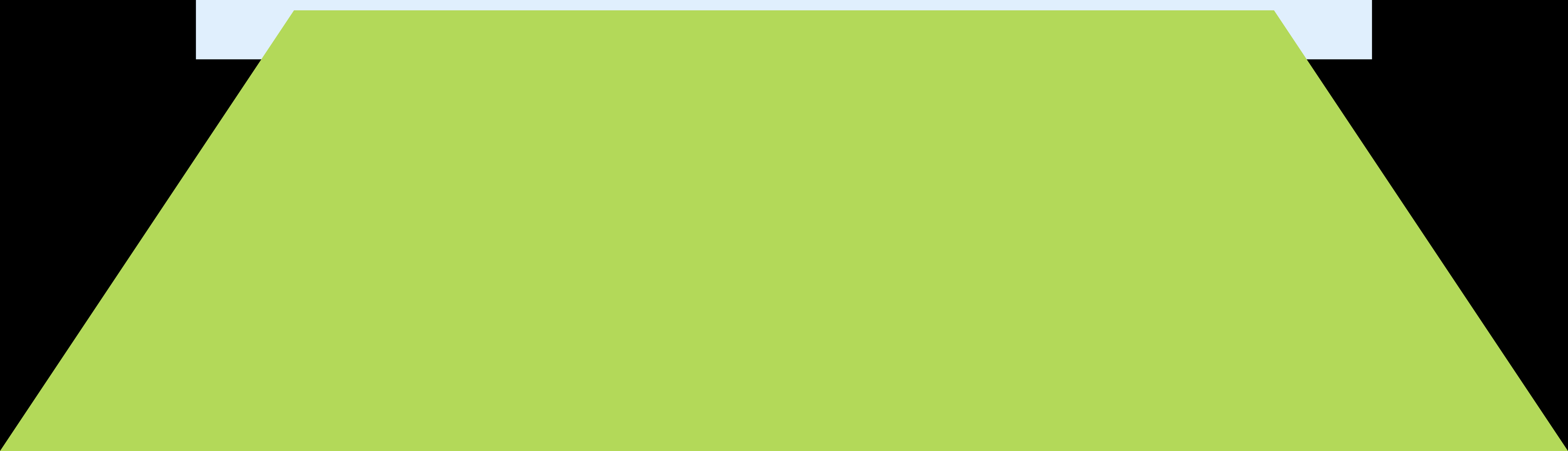
...WHY WOULD THAT WORK?

. . . WHY WOULD THAT WORK?

- We may need a bit of notation

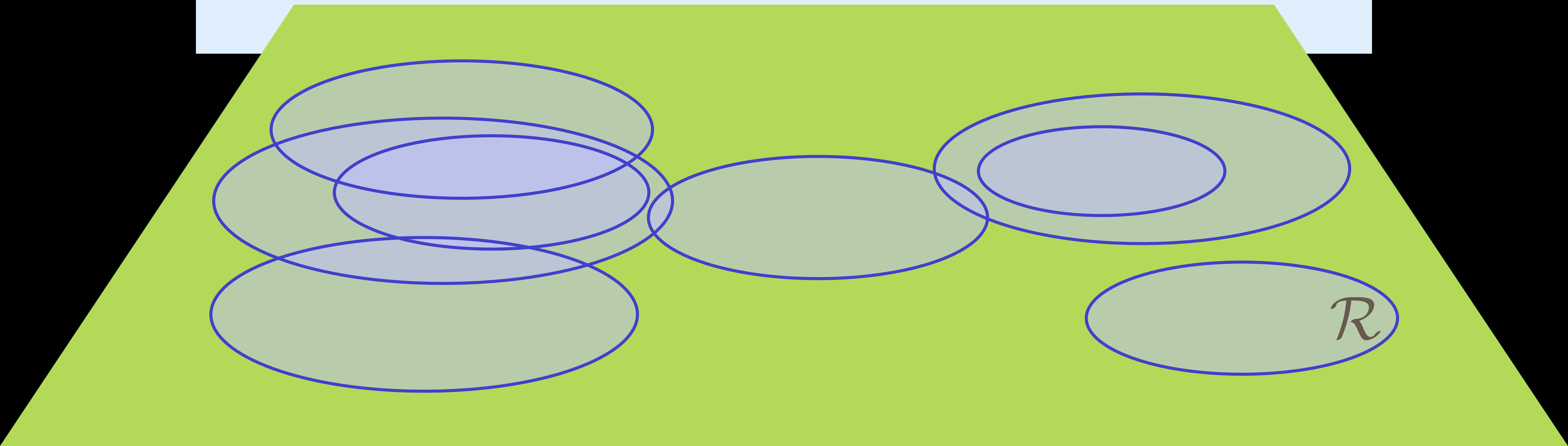
...WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1 \ldots 2n$



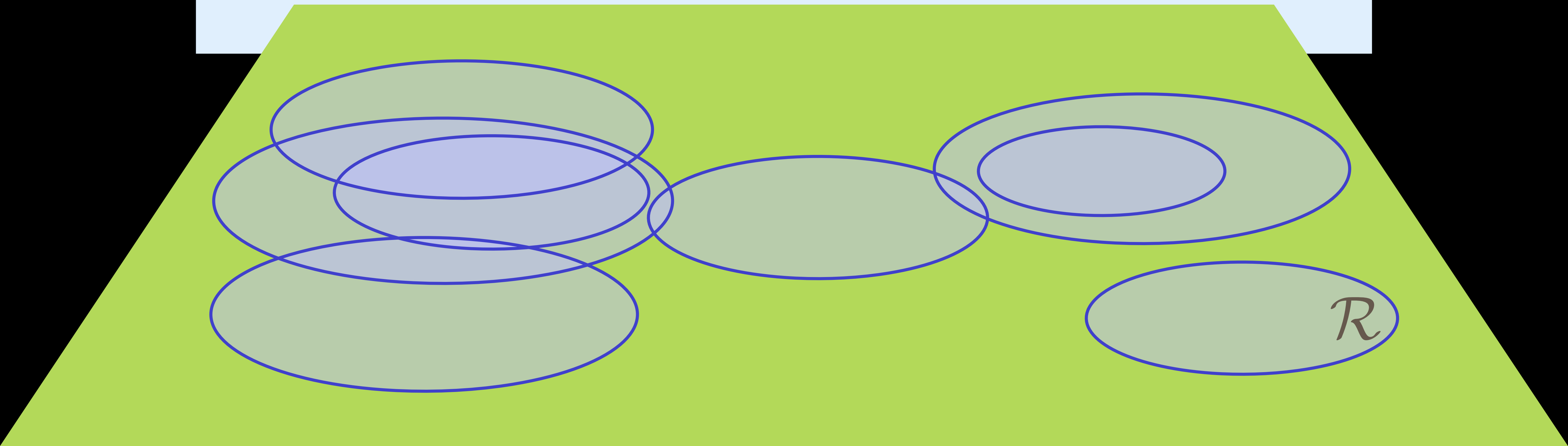
...WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1\dots 2n$



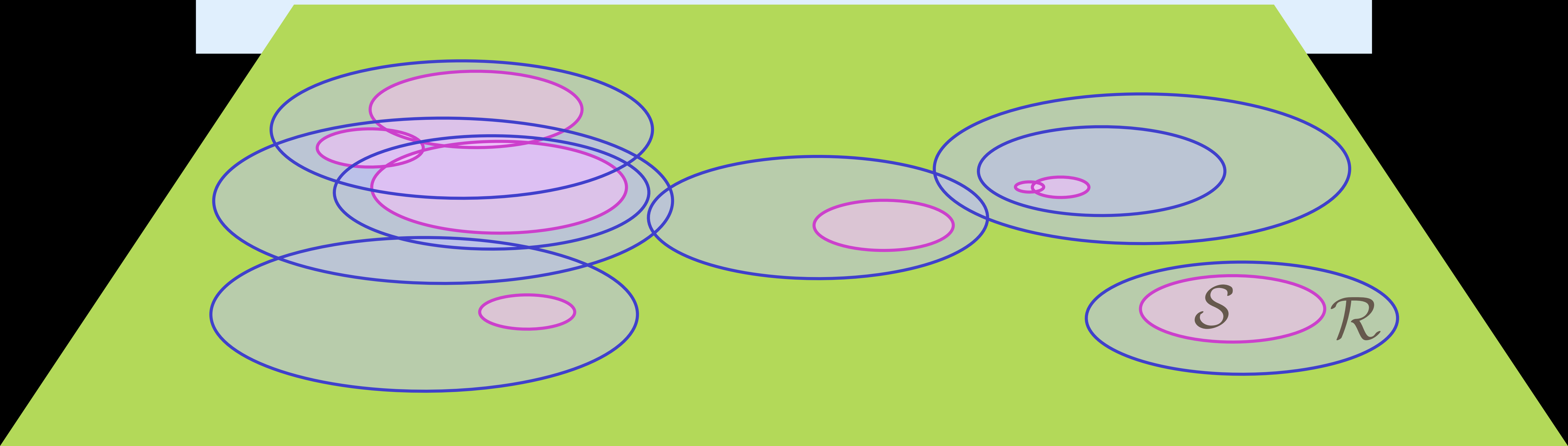
...WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1\ldots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1\ldots n$



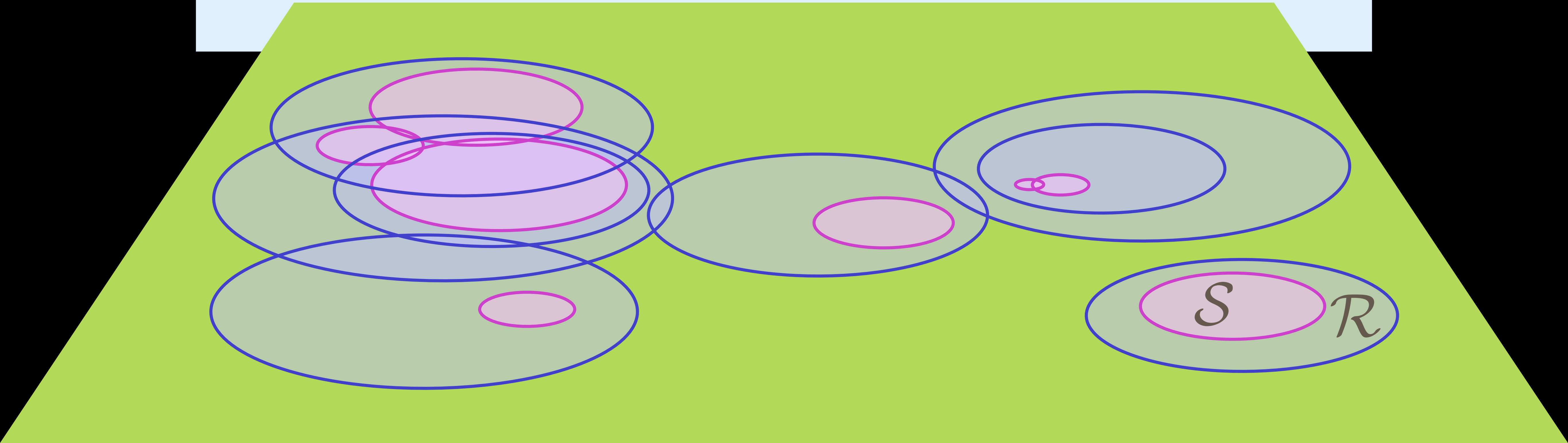
...WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1 \ldots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1 \ldots n$



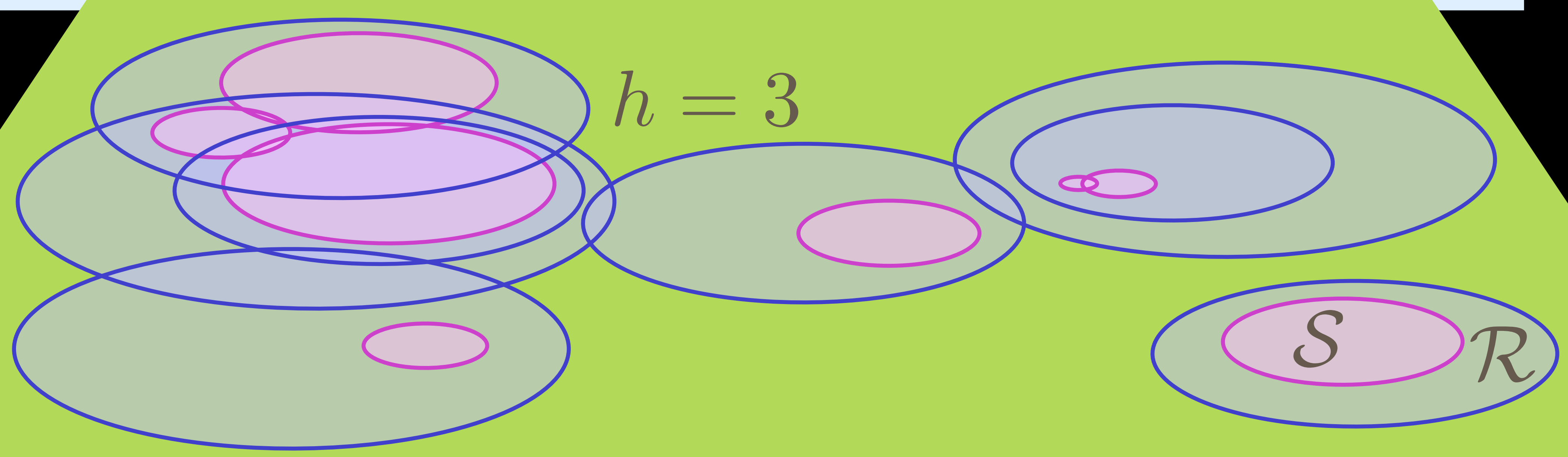
...WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1 \ldots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1 \ldots n$
  - h: half of the $\mathcal{R}$ has ply at least h



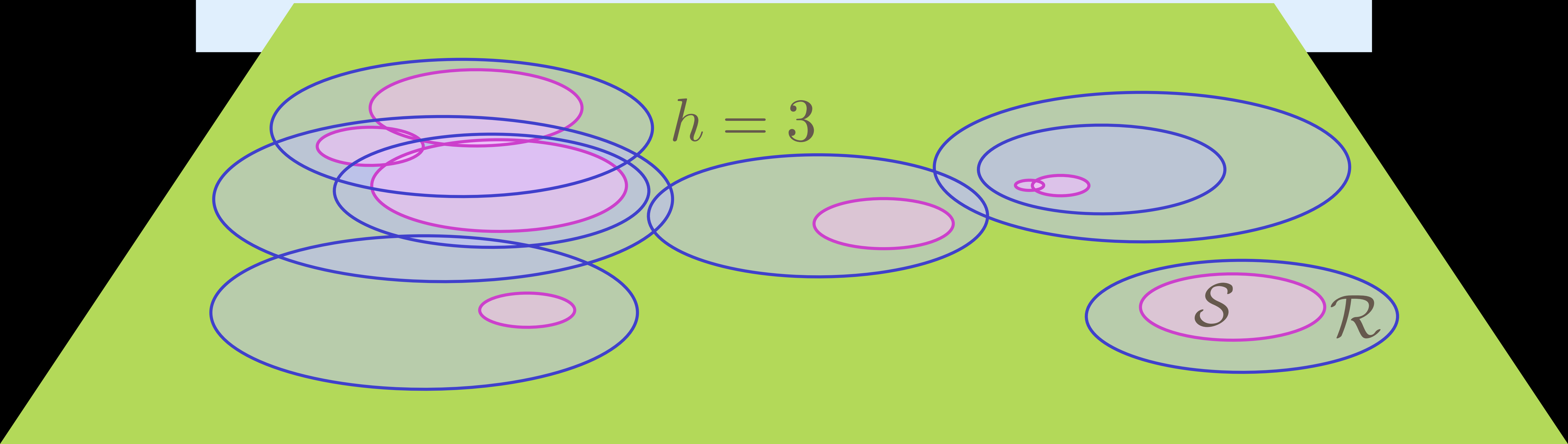
. . . WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1\ldots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1\ldots n$
  - h: half of the $\mathcal{R}$ has ply at least h



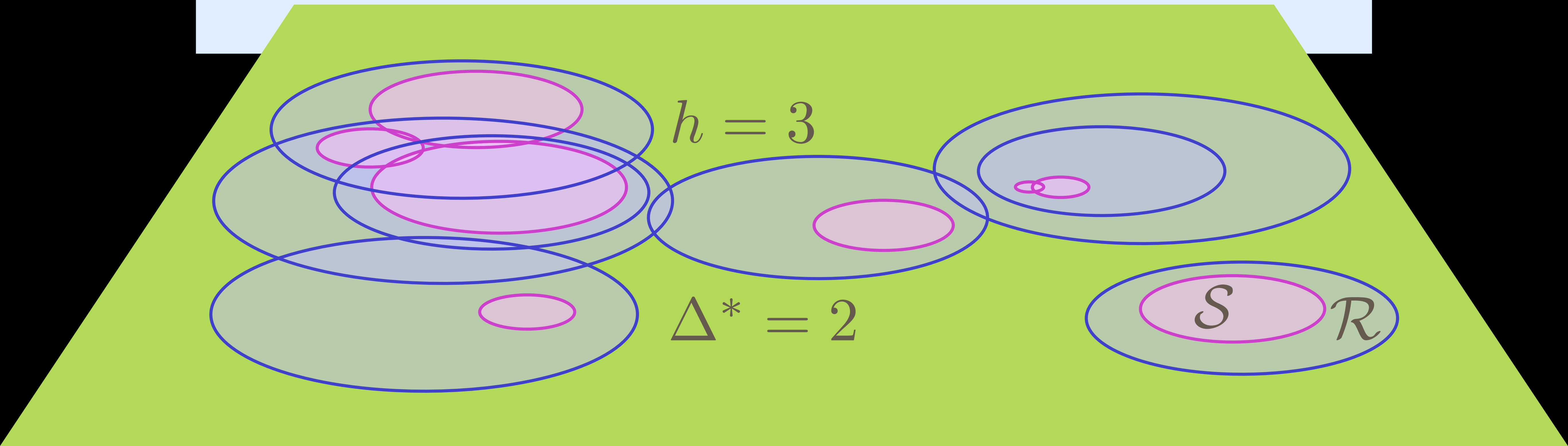
. . . WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1\ldots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1\ldots n$
  - h: half of the $\mathcal{R}$ has ply at least h
  - Δ^*: optimal ply (the ply of $\mathcal{S}$)



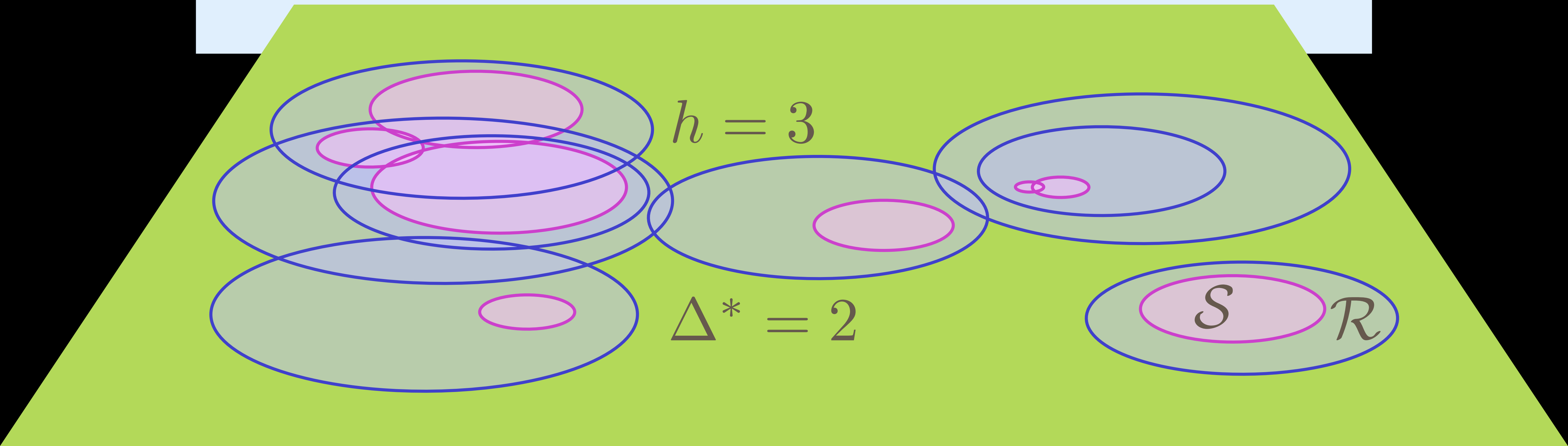
. . . WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1 \ldots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1 \ldots n$
  - h: half of the $\mathcal{R}$ has ply at least h
  - Δ^*: optimal ply (the ply of $\mathcal{S}$)



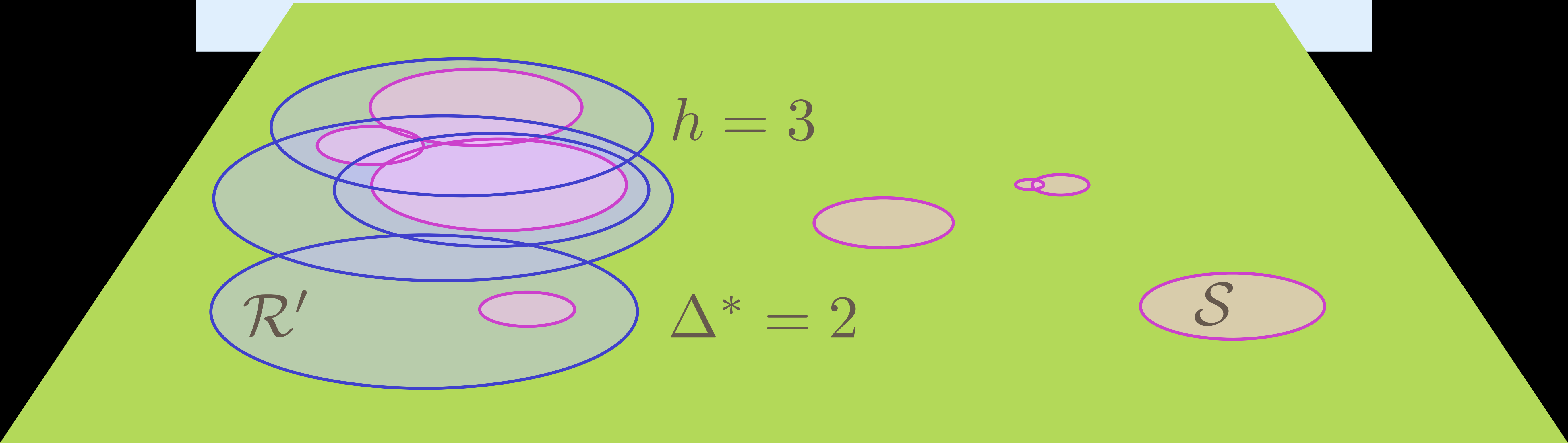
...WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1\ldots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1\ldots n$
  - h: half of the $\mathcal{R}$ has ply at least h
  - Δ^*: optimal ply (the ply of $\mathcal{S}$)
  - $\mathcal{R}'$: half of $\mathcal{R}$ with highest ply (at least h)



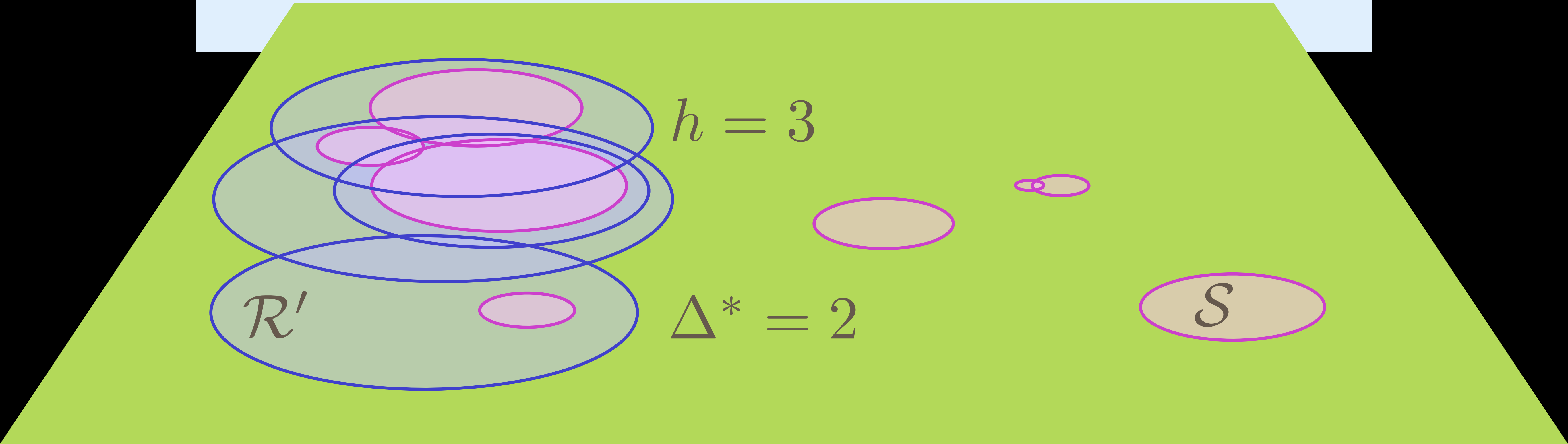
...WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1\ldots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1\ldots n$
  - h: half of the $\mathcal{R}$ has ply at least h
  - Δ^*: optimal ply (the ply of $\mathcal{S}$)
  - $\mathcal{R}'$: half of $\mathcal{R}$ with highest ply (at least h)



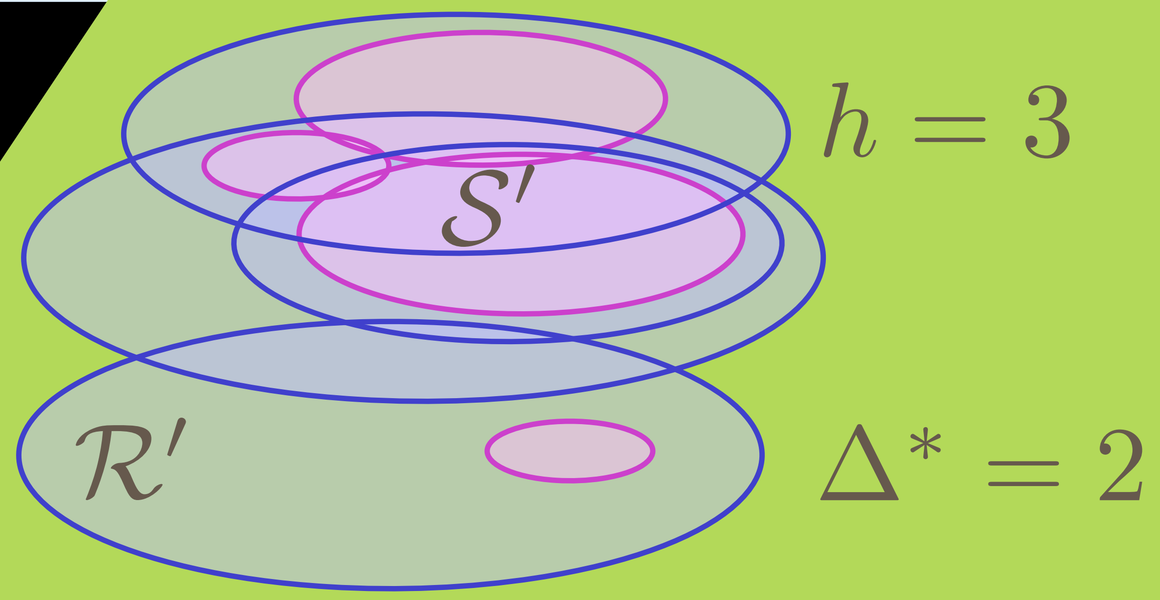
...WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1\ldots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1\ldots n$
  - h: half of the $\mathcal{R}$ has ply at least h
  - Δ^*: optimal ply (the ply of $\mathcal{S}$)
  - $\mathcal{R}'$: half of $\mathcal{R}$ with highest ply (at least h)
  - $\mathcal{S}'$: optimal regions corresponding to $\mathcal{R}'$



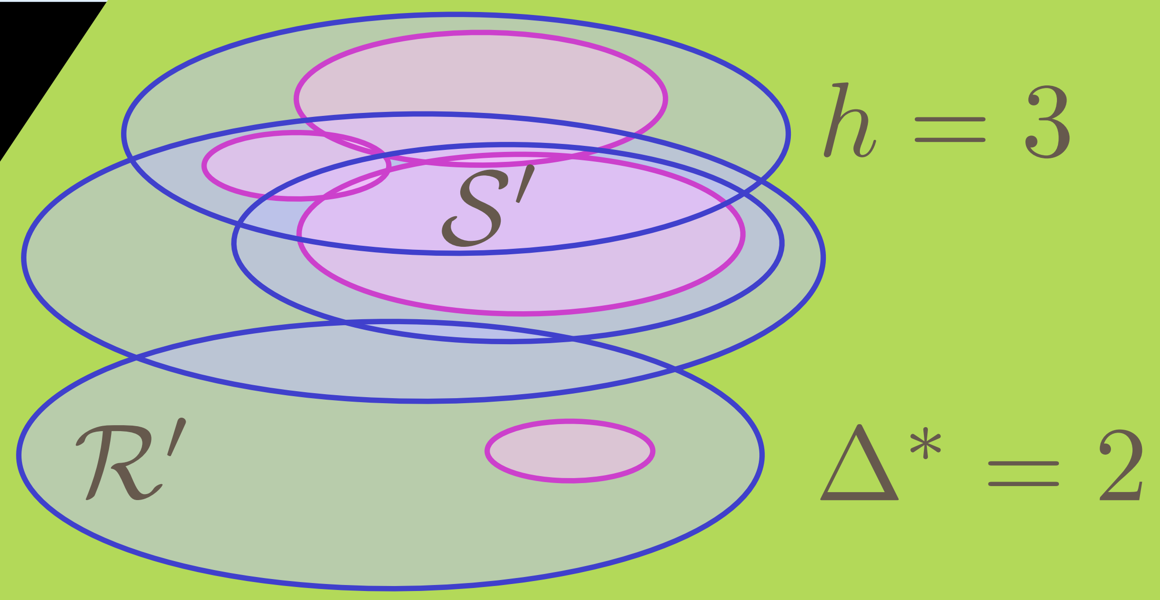
...WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1\dots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1\dots n$
  - h: half of the $\mathcal{R}$ has ply at least h
  - Δ^*: optimal ply (the ply of $\mathcal{S}$)
  - $\mathcal{R}'$: half of $\mathcal{R}$ with highest ply (at least h)
  - $\mathcal{S}'$: optimal regions corresponding to $\mathcal{R}'$



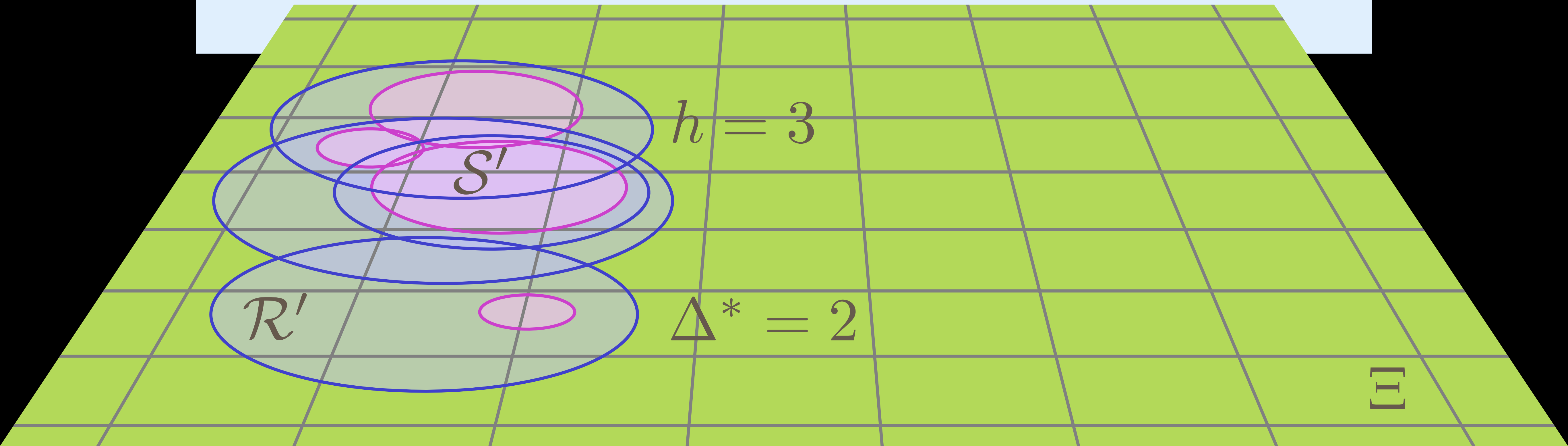
...WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1\ldots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1\ldots n$
  - h: half of the $\mathcal{R}$ has ply at least h
  - Δ^*: optimal ply (the ply of $\mathcal{S}$)
  - $\mathcal{R}'$: half of $\mathcal{R}$ with highest ply (at least h)
  - $\mathcal{S}'$: optimal regions corresponding to $\mathcal{R}'$
  - Ξ: a grid with cells of size $\Theta(n)$



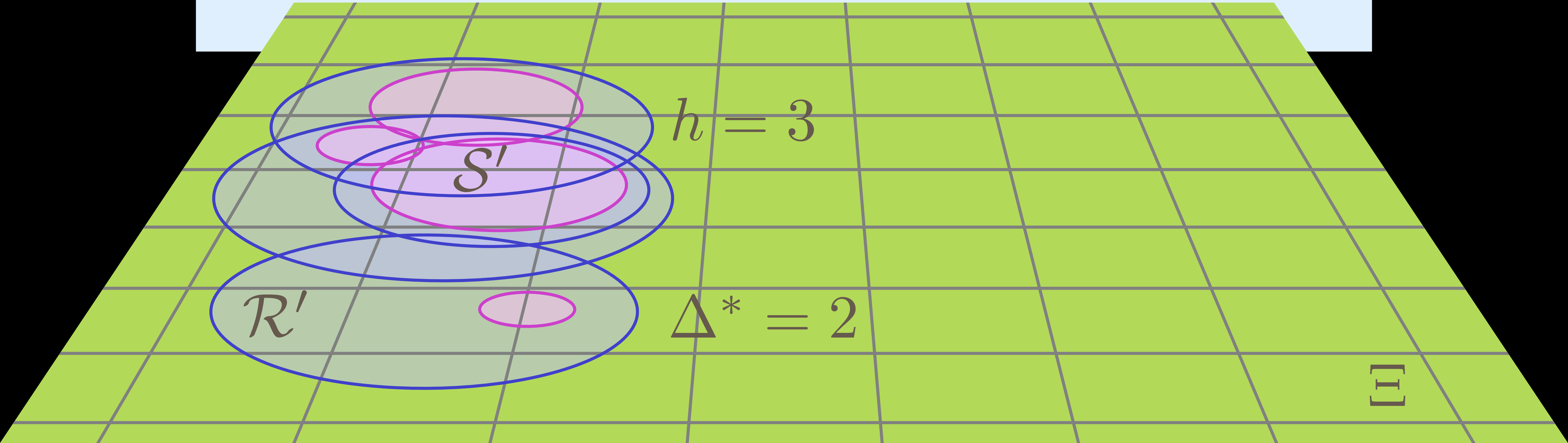
. . . WHY WOULD THAT WORK?

- We may need a bit of notation
  - $\mathcal{R}$: n regions of sizes $n+1\dots 2n$
  - $\mathcal{S}$: *optimal* final regions of sizes $1\dots n$
  - h: half of the $\mathcal{R}$ has ply at least h
  - Δ^*: optimal ply (the ply of $\mathcal{S}$)
  - $\mathcal{R}'$: half of $\mathcal{R}$ with highest ply (at least h)
  - $\mathcal{S}'$: optimal regions corresponding to $\mathcal{R}'$
  - Ξ: a grid with cells of size $\Theta(n)$



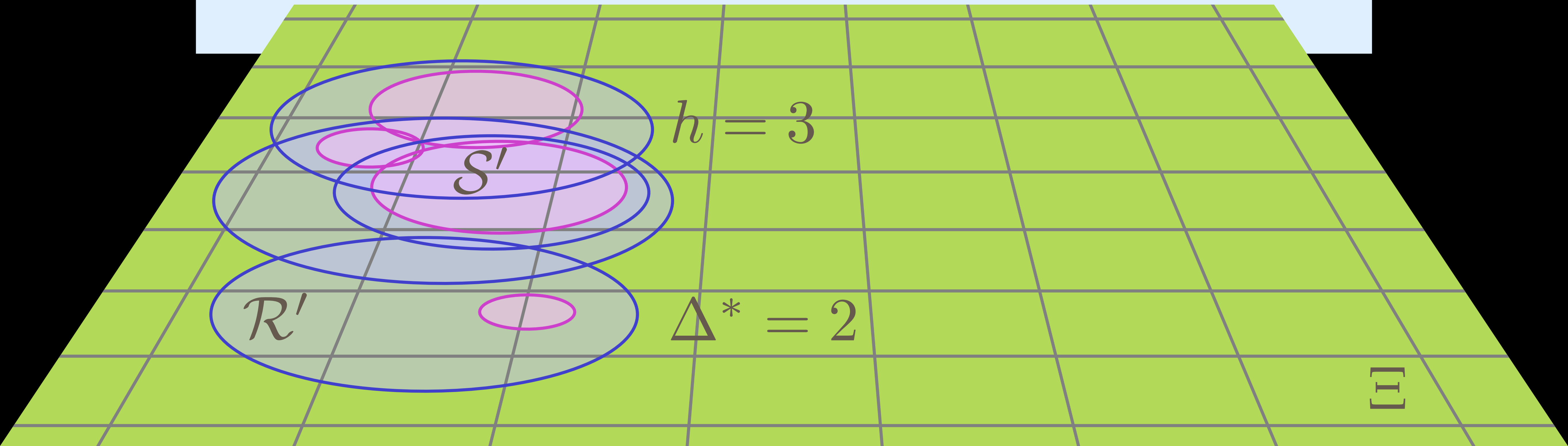
...WHY WOULD THAT WORK?

- Now for the argument...



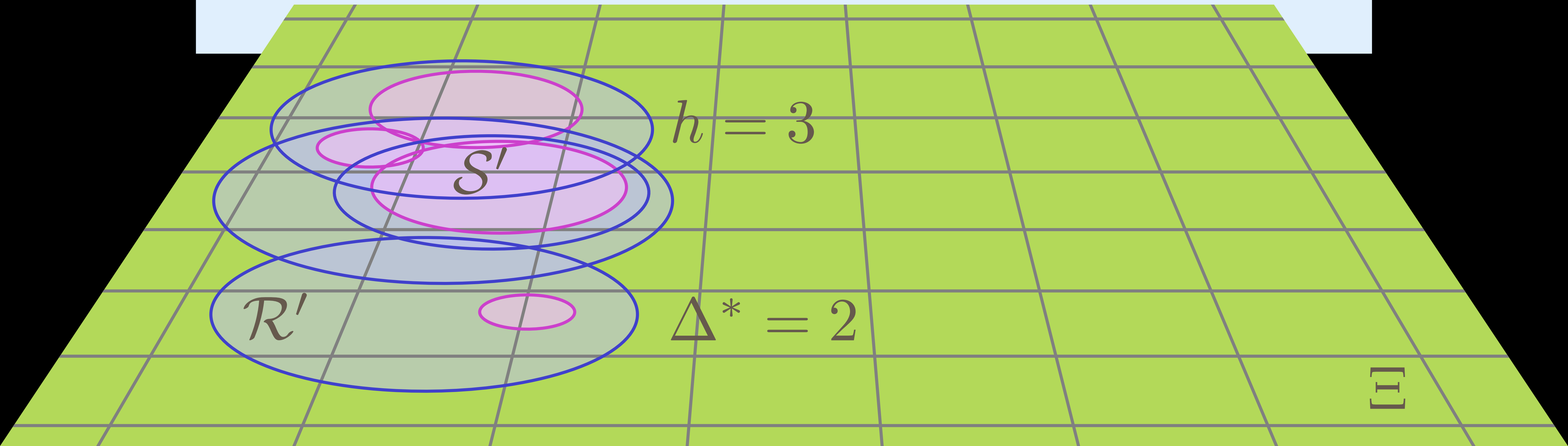
...WHY WOULD THAT WORK?

- Now for the argument...
  - $\mathcal{S}'$ covers an area of at least $\Omega(n^3/\Delta^*)$



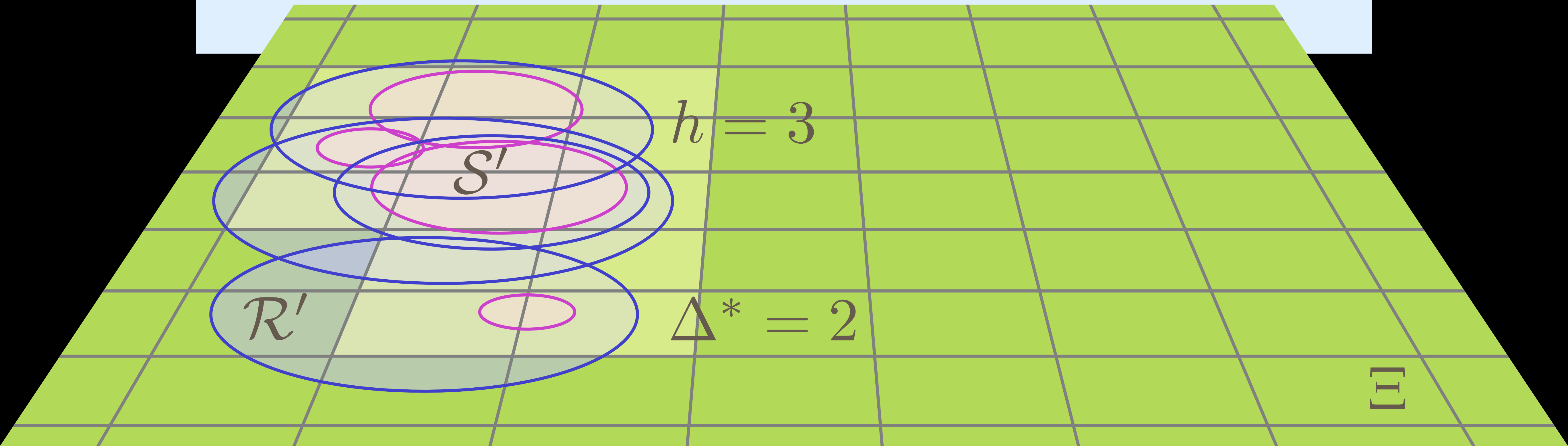
. . . WHY WOULD THAT WORK?

- Now for the argument. . .
  - $\mathcal{S}'$ covers an area of at least $\Omega(n^3/\Delta^*)$
  - $\mathcal{S}'$ intersects at least $\Omega(n/\Delta^*)$ cells of Ξ



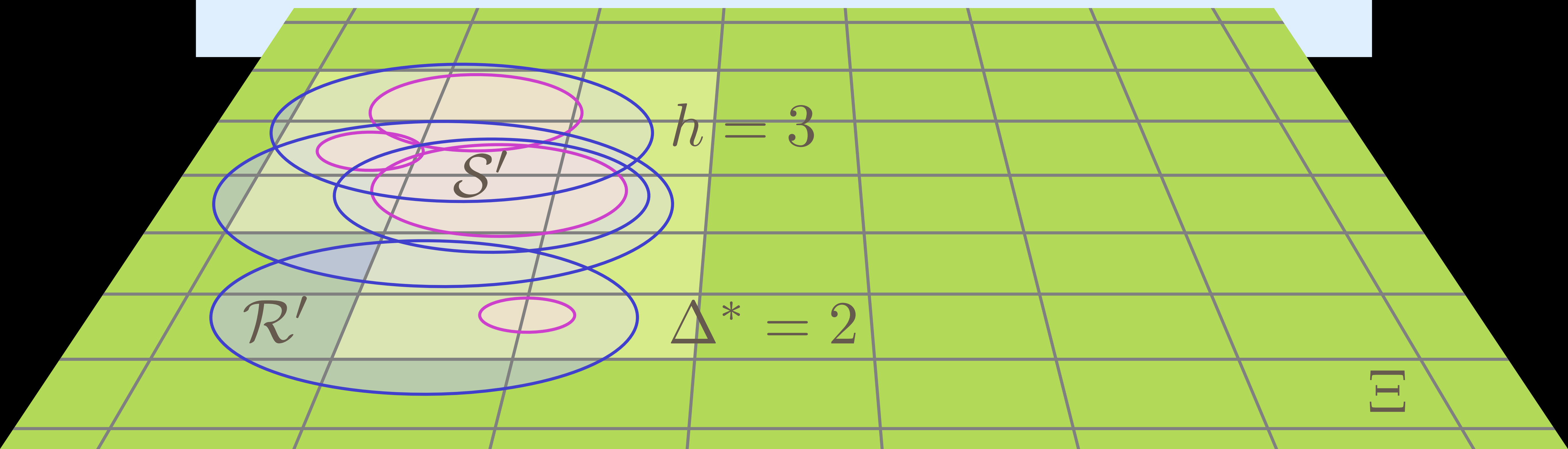
...WHY WOULD THAT WORK?

- Now for the argument...
  - $\mathcal{S}'$ covers an area of at least $\Omega(n^3/\Delta^*)$
  - $\mathcal{S}'$ intersects at least $\Omega(n/\Delta^*)$ cells of Ξ



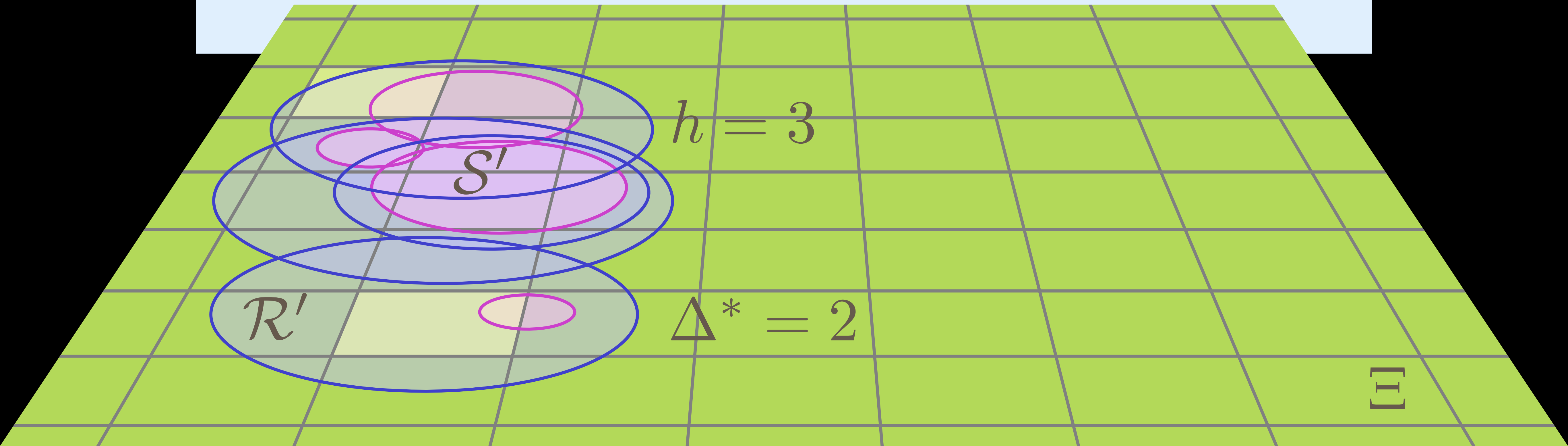
...WHY WOULD THAT WORK?

- Now for the argument...
  - $\mathcal{S}'$ covers an area of at least $\Omega(n^3/\Delta^*)$
  - $\mathcal{S}'$ intersects at least $\Omega(n/\Delta^*)$ cells of Ξ
  - There is an independent set I of cells of Ξ that are intersected by $\mathcal{S}'$ of size $\Omega(n/\Delta^*)$



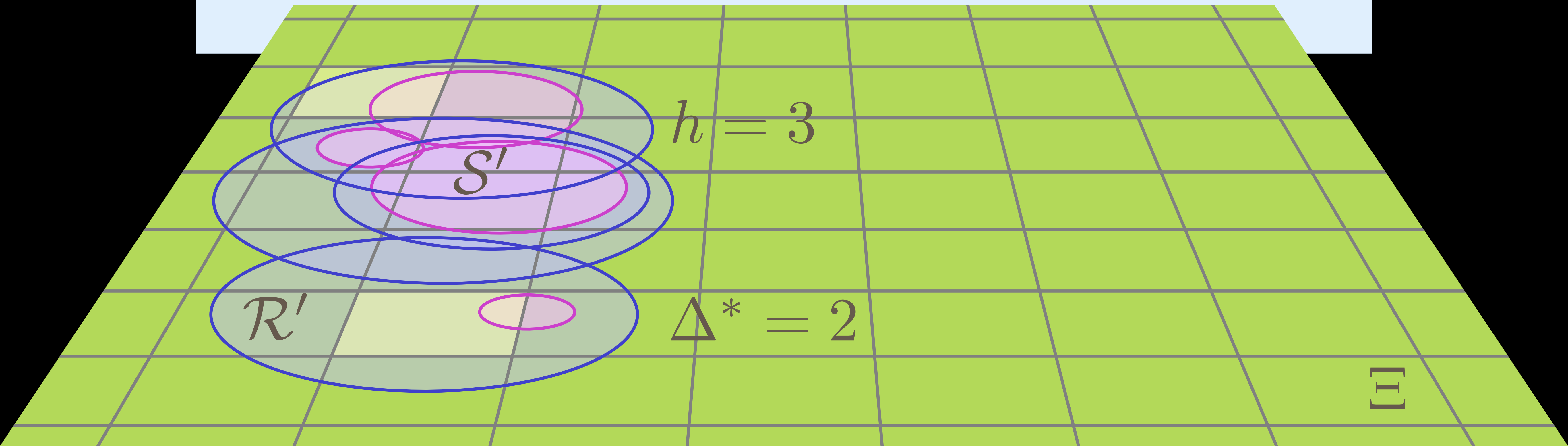
...WHY WOULD THAT WORK?

- Now for the argument...
  - $\mathcal{S}'$ covers an area of at least $\Omega(n^3/\Delta^*)$
  - $\mathcal{S}'$ intersects at least $\Omega(n/\Delta^*)$ cells of Ξ
  - There is an independent set I of cells of Ξ that are intersected by $\mathcal{S}'$ of size $\Omega(n/\Delta^*)$



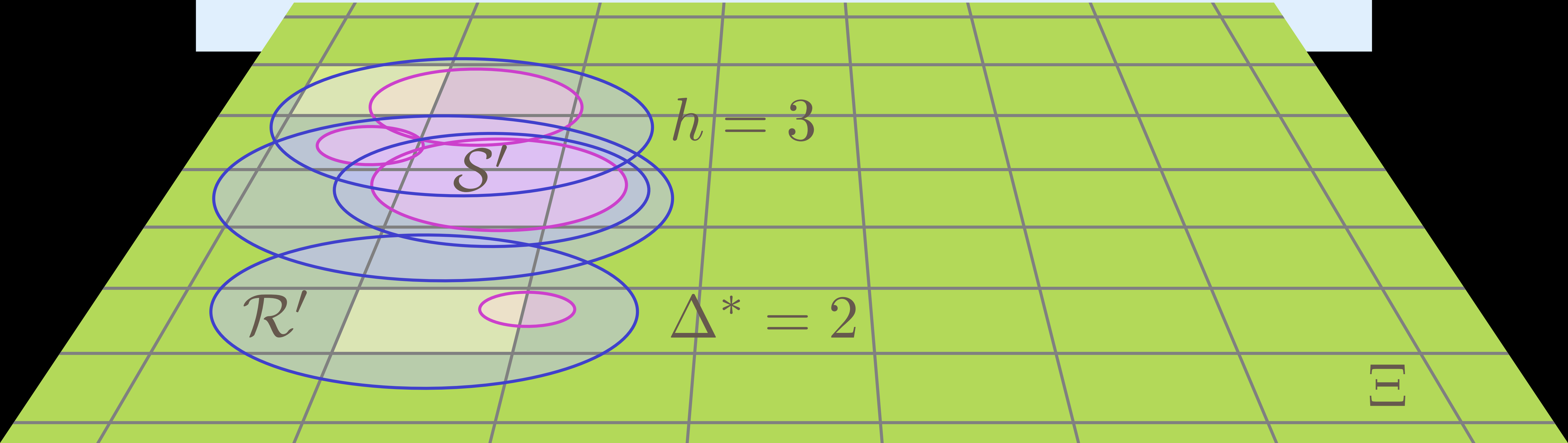
. . . WHY WOULD THAT WORK?

- Now for the argument. . .
  - $\mathcal{S}'$ covers an area of at least $\Omega(n^3/\Delta^*)$
  - $\mathcal{S}'$ intersects at least $\Omega(n/\Delta^*)$ cells of Ξ
  - There is an independent set I of cells of Ξ that are intersected by $\mathcal{S}'$ of size $\Omega(n/\Delta^*)$
  - Each cell of I is intersected by at least h unique regions of $\mathcal{R}'$



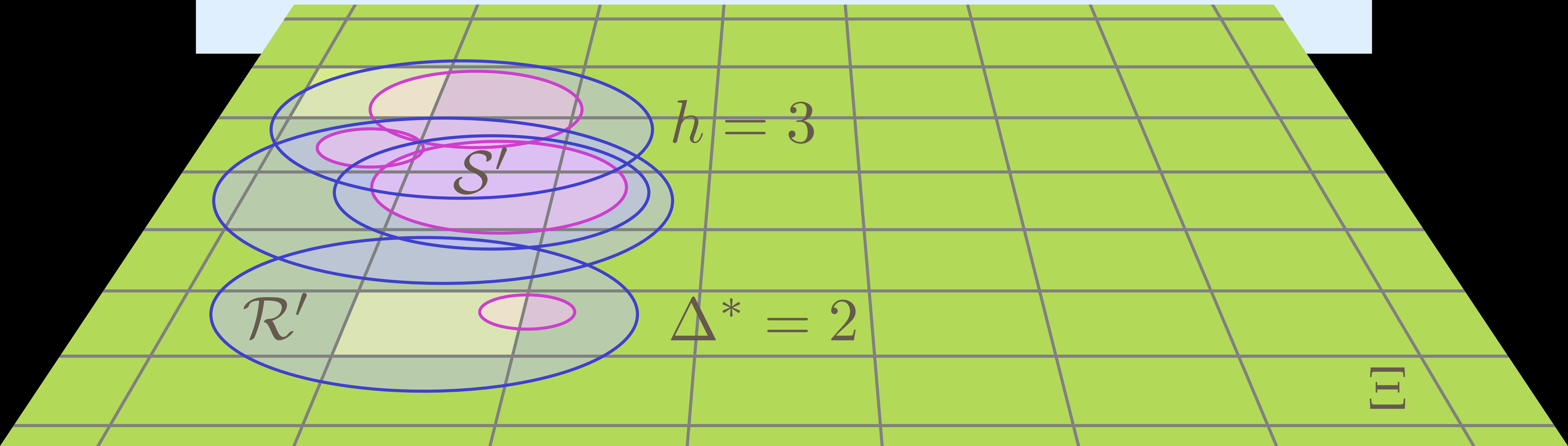
...WHY WOULD THAT WORK?

- Now for the argument...
  - $\mathcal{S}'$ covers an area of at least $\Omega(n^3/\Delta^*)$
  - $\mathcal{S}'$ intersects at least $\Omega(n/\Delta^*)$ cells of Ξ
  - There is an independent set I of cells of Ξ that are intersected by $\mathcal{S}'$ of size $\Omega(n/\Delta^*)$
  - Each cell of I is intersected by at least h unique regions of $\mathcal{R}'$
  - There are $\Omega(h \cdot n/\Delta^*)$ regions in $\mathcal{R}'$



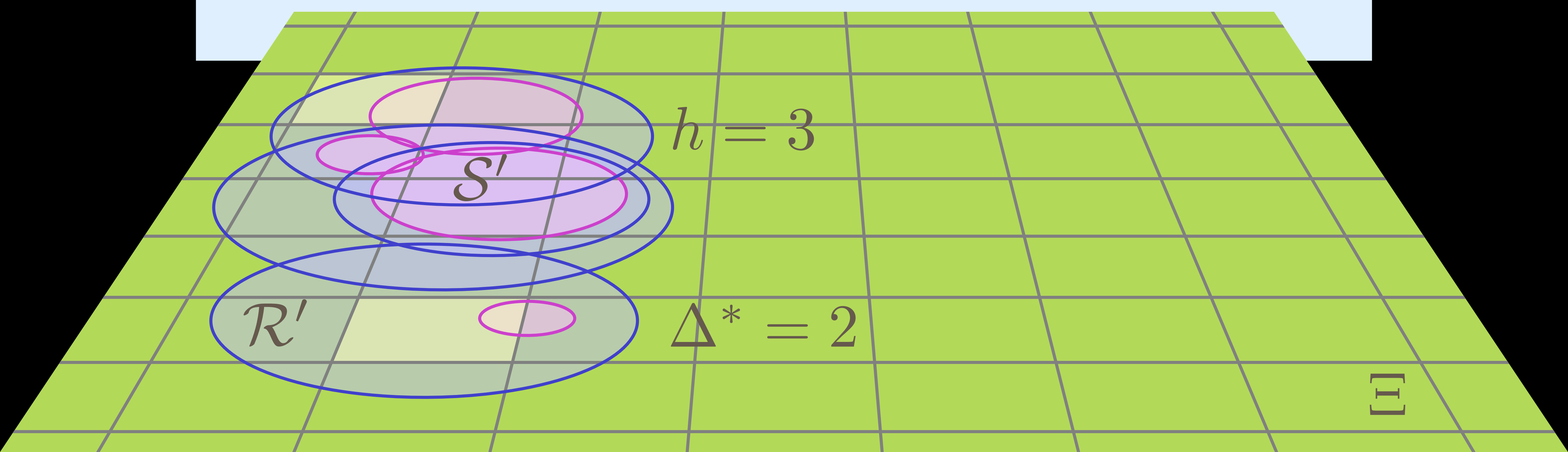
. . . WHY WOULD THAT WORK?

- Now for the argument. . .
  - $\mathcal{S}'$ covers an area of at least $\Omega(n^3/\Delta^*)$
  - $\mathcal{S}'$ intersects at least $\Omega(n/\Delta^*)$ cells of Ξ
  - There is an independent set I of cells of Ξ that are intersected by $\mathcal{S}'$ of size $\Omega(n/\Delta^*)$
  - Each cell of I is intersected by at least h unique regions of $\mathcal{R}'$
  - There are $\Omega(h \cdot n/\Delta^*)$ regions in $\mathcal{R}'$
  - So. . . $h \in O(\Delta^*)$



…WHY WOULD THAT WORK?

- Now for the argument…
  - $\mathcal{S}'$ covers an area of at least $\Omega(n^3/\Delta^*)$
  - $\mathcal{S}'$ intersects at least $\Omega(n/\Delta^*)$ cells of Ξ
  - There is an independent set I of cells of Ξ that are intersected by $\mathcal{S}'$ of size $\Omega(n/\Delta^*)$
  - Each cell of I is intersected by at least h unique regions of $\mathcal{R}'$
  - There are $\Omega(h \cdot n/\Delta^*)$ regions in $\mathcal{R}'$
  - So… $h \in O(\Delta^*)$
  - !!!



FURTHER RESULTS

FURTHER RESULTS

- Limited time τ

FURTHER RESULTS

- Limited time τ
  - Depends average time ℓ since last query

FURTHER RESULTS

- Limited time τ
  - Depends average time ℓ since last query
  - Competitive ratio: $O(\sqrt{\ell/\tau + 1})$

FURTHER RESULTS

- Limited time τ
  - Depends average time ℓ since last query
  - Competitive ratio: $O(\sqrt{\ell/\tau + 1})$
- Dimension d

FURTHER RESULTS

- Limited time τ
  - Depends average time ℓ since last query
  - Competitive ratio: $O(\sqrt{\ell/\tau + 1})$
- Dimension d
  - Analysis goes through mostly unchanged

FURTHER RESULTS

- Limited time τ
  - Depends average time ℓ since last query
  - Competitive ratio: $O(\sqrt{\ell/\tau + 1})$
- Dimension d
  - Analysis goes through mostly unchanged
  - Competitive ratio: $O((\ell/\tau + 1)^{d - \frac{d}{d+1}})$

FURTHER RESULTS

- Limited time τ
  - Depends average time ℓ since last query
  - Competitive ratio: $O(\sqrt{\ell/\tau + 1})$
- Dimension d
  - Analysis goes through mostly unchanged
  - Competitive ratio: $O((\ell/\tau + 1)^{d - \frac{d}{d+1}})$
- Lower bounds

FURTHER RESULTS

- Limited time τ
  - Depends average time ℓ since last query
  - Competitive ratio: $O(\sqrt{\ell/\tau + 1})$
- Dimension d
  - Analysis goes through mostly unchanged
  - Competitive ratio: $O((\ell/\tau + 1)^{d - \frac{d}{d+1}})$
- Lower bounds
  - Ratios are tight up to constant factor

FURTHER RESULTS

- Limited time τ
  - Depends average time ℓ since last query
  - Competitive ratio: $O(\sqrt{\ell/\tau+1})$
- Dimension d
  - Analysis goes through mostly unchanged
  - Competitive ratio: $O((\ell/\tau+1)^{d-\frac{d}{d+1}})$
- Lower bounds
  - Ratios are tight up to constant factor
- NP-hardness

FURTHER RESULTS

- Limited time τ
  - Depends average time ℓ since last query
  - Competitive ratio: $O(\sqrt{\ell/\tau + 1})$
- Dimension d
  - Analysis goes through mostly unchanged
  - Competitive ratio: $O((\ell/\tau + 1)^{d-\frac{d}{d+1}})$
- Lower bounds
  - Ratios are tight up to constant factor
- NP-hardness
  - Even knowing the full trajectories

THANK YOU!