Spatial Data Structures
(For Acceleration)

Flocking - what if n gets big?
$O(n^2)$ - need fast near neighbor queries
$O(n)$ - avoid drawing - off screen culling

Idea:
Spatial Data Structures - organize for fast search/queries /....
Many data structures
Many applications

As # of dimensions go up, things get tricky
Points vs. Objects

Applications
Collisions (object sets)
nearest neighbor / k-NN / approximate k-NN / d-neighbor
Range search
Culling - View culling, Occlusion Culling
LOD
1D Case (boring, but ... ) for points

\( O(n) \) draw in range \( \| \) nearest neighbor

\( O(n^2) \) \( k \)-nearest neighbor for each

Sort \( n \log n \) cost

- Equivalent to building search tree
- \( \text{Knn} \) takes \( \log N + K \)
- All \( \text{Knn} \) takes \( n \log n \)
- Scan range - only considers in range

Update - resort?

- Fast update algorithms

Can you do better? (no \( n \log n \))

Radix Sort - buckets - \( O(n) \)

- Might not help (bad distribution)
- Always helps range / \( d \)-neighbor query

Updates?

Approximate nearest neighbor

- Closest in bucket \( b \times b \)
- At worst off by bucket size
- Better strategies
Higher Dimensions

- no analog to sorting
- search trees
- grids - don't work
  most grid cells empty (especially as dimension goes up)
  too long just to clear cells!
- spatial hashing
  locality preserving (close in space ⇒ close in hash)
  simple one: pick a dimension
  LSH: pick multiple dimensions
  as # of hashes approaches # dimension

Weird Fact:

as dimensions get high, linear search is not so bad (LSH)
low dimensions - highly tuned spatial data structures + tricks
ANN library
Hierarchical Data Structures for high-dimensions

Spatial Partitions VS. OBJECT GROUPING
- Quad trees / Octrees
- K-D Trees
- Arbitrary BSP

Bounding Volume Hierarchy
- Box Trees
- R-Trees

Dynamic / Kinetic Data Structures
VS. static

BVH choices
- Geometry: spheres, axis-aligned boxes AABB, object OBB
- Balancing, branchiness