Collision Detection

Jane Li

Assistant Professor
Mechanical Engineering Department, Robotic Engineering Program
Worcester Polytechnic Institute

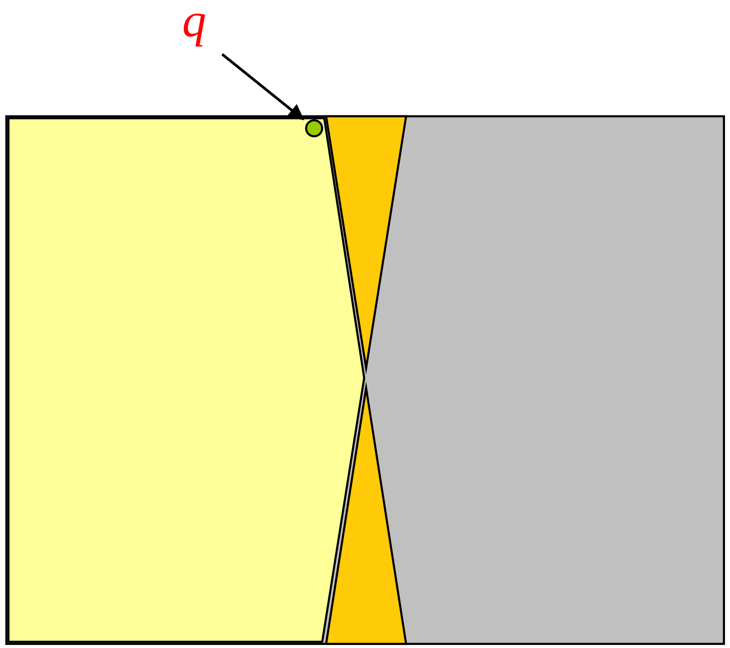
Quiz (10 pts)

- (6 pts) What does it mean if the free space *F* is $(\varepsilon, \alpha, \beta)$-expansive?

- (4 pts) Why is RRT probabilistically complete?

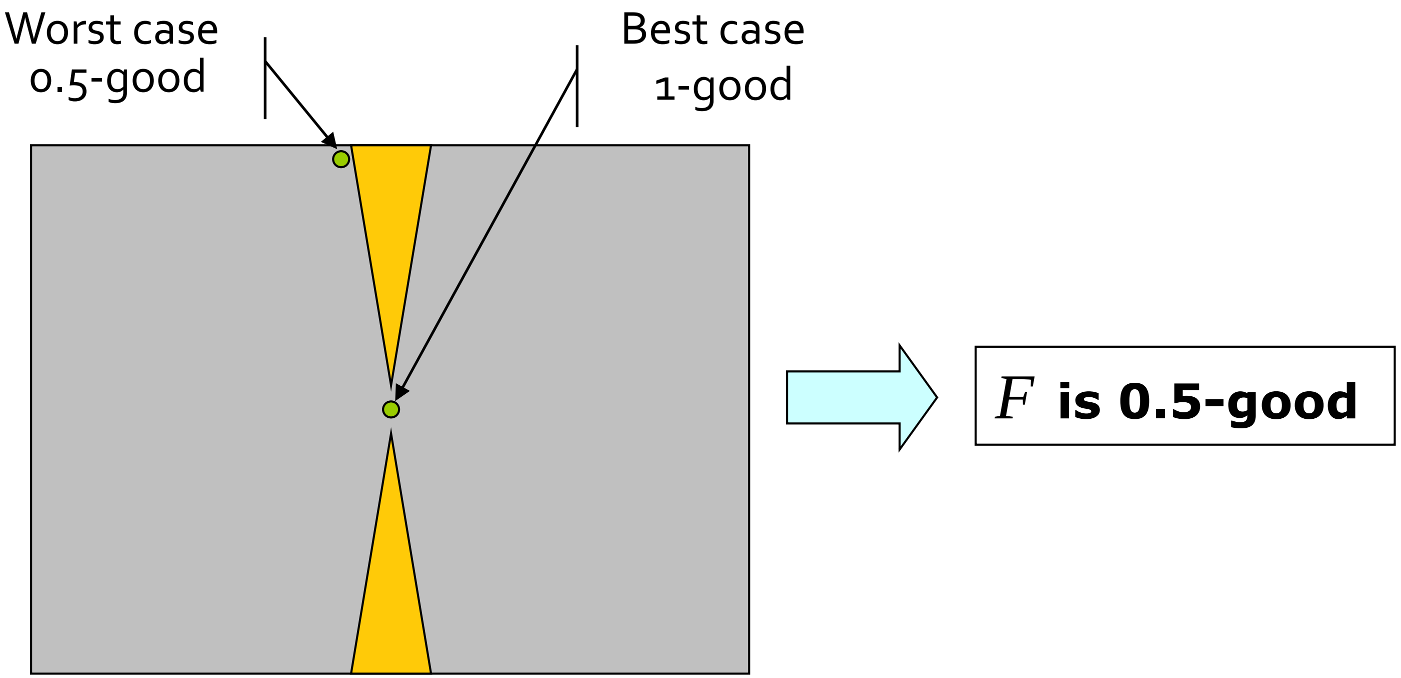
Definition: Visibility Set

- Visibility set of q
  - All configurations in F that can be connected to q by a straight-line path in F
  - All configurations seen by q



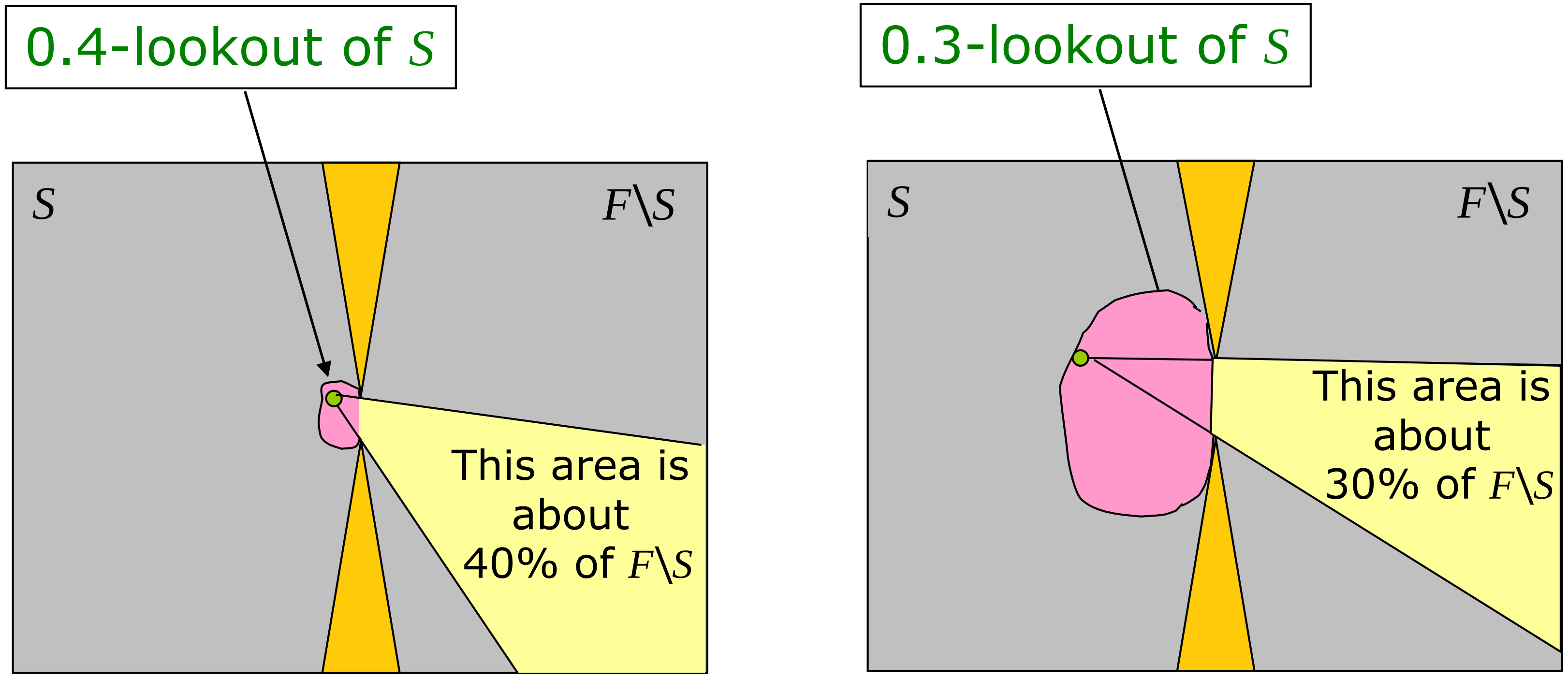
Definition: Є-good

- **Every** free configuration sees **at least** ϵ fraction of the free space, ϵ in (0,1].



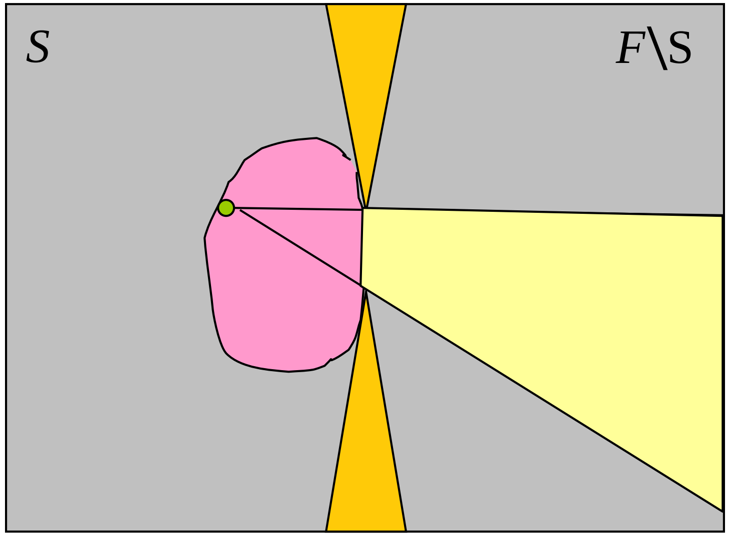
Definition: Lookout of a Subset S

- **Subset** of points in S that can see **at least β fraction** of $F \backslash S$, β is in (0,1].



Definition: (ε, α, β) - Expansive

- The free space F is $(\varepsilon, \alpha, \beta)$-expansive if
  - Free space F is ε-good
  - For each subset S of F, its β-lookout is at least α fraction of S. $\varepsilon, \alpha, \beta$ are in (0,1]



F is ε-good → ε=0.5

β-lookout → β=0.4

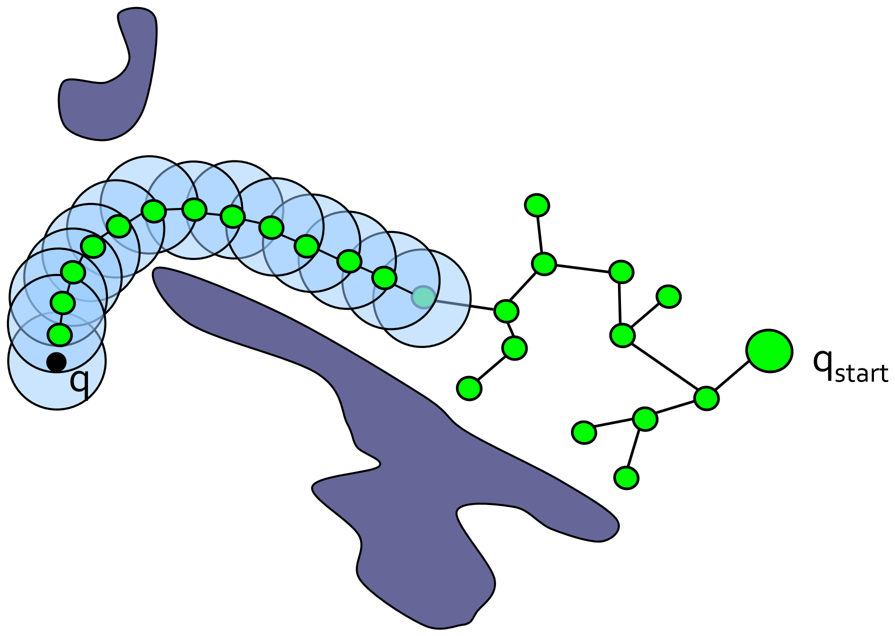
$$\frac{\text{Volume}(\beta\text{-lookout})}{\text{Volume}(S)} \rightarrow \alpha=0.2$$

F is (ε, α, β)-expansive, where ε=0.5, α=0.2, β=0.4.

Proof of RRT Probabilistic Completeness

- As the RRT reaches all of Q_{free},
  - The probability that q_{rand} immediately becomes a new vertex approaches 1.

RRT probabilistically complete

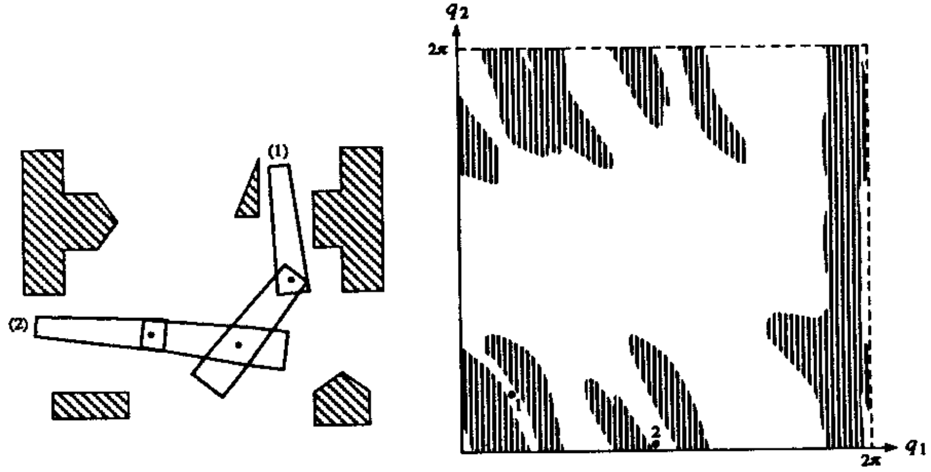


Collision Detection

Motivation

- Find a path in C-space
  - Compute C_{obs} – Hard
  - Check if a configuration is collision – Easy

- Collision detection
  - For a single configuration
  - Along a path/trajectory



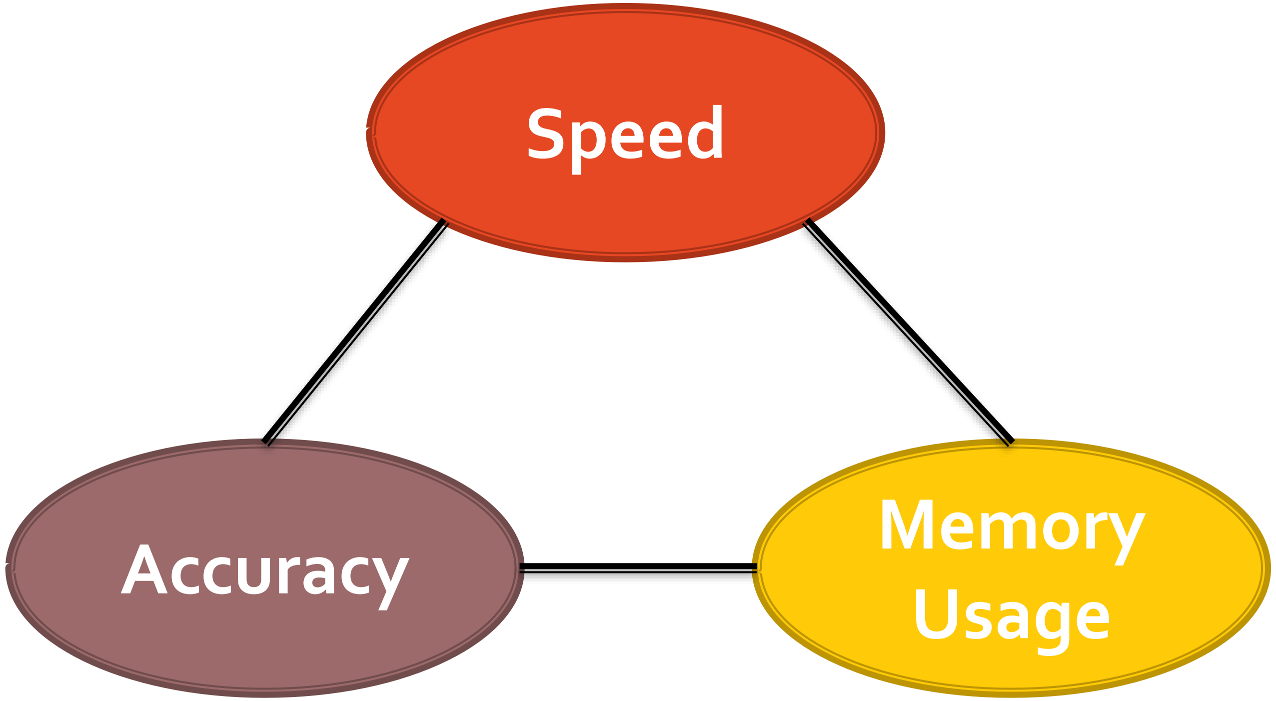
Fast collision detection

- Speed is **very** important
  - Need to check collision for **large number of** configurations

  - For most planners, runtime for real-world task depends **heavily** on the speed of collision checking

Tradeoff

- Increase speed → more memory, less accuracy



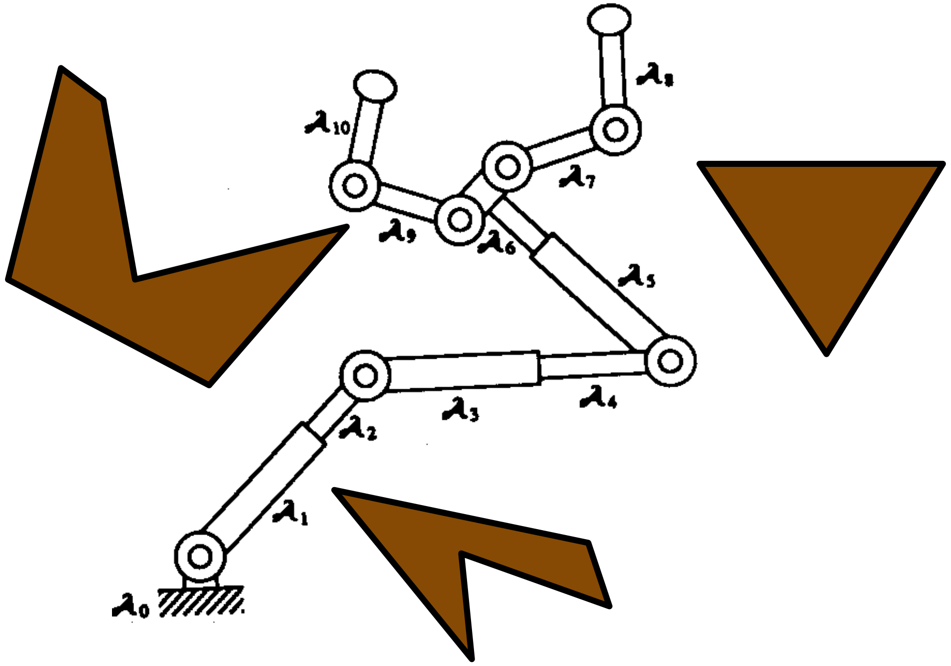
Crowd simulation

Figure from Kanyuk, Paul. "Brain Springs: Fast Physics for Large Crowds in WALLdr E." IEEE Computer Graphics and Applications 29.4 (2009).

Interactive Large-scale Crowd Simulation

Self-Collision Checking for Articulated Robot

- Self-collision is typically not an issue for mobile robots
- Articulated robots must avoid self-collision
  - Parent-child link – set proper joint angle limits
  - With root or other branches – e.g. Humanoid robot?



Self-Collision Checking For Humanoid Robot

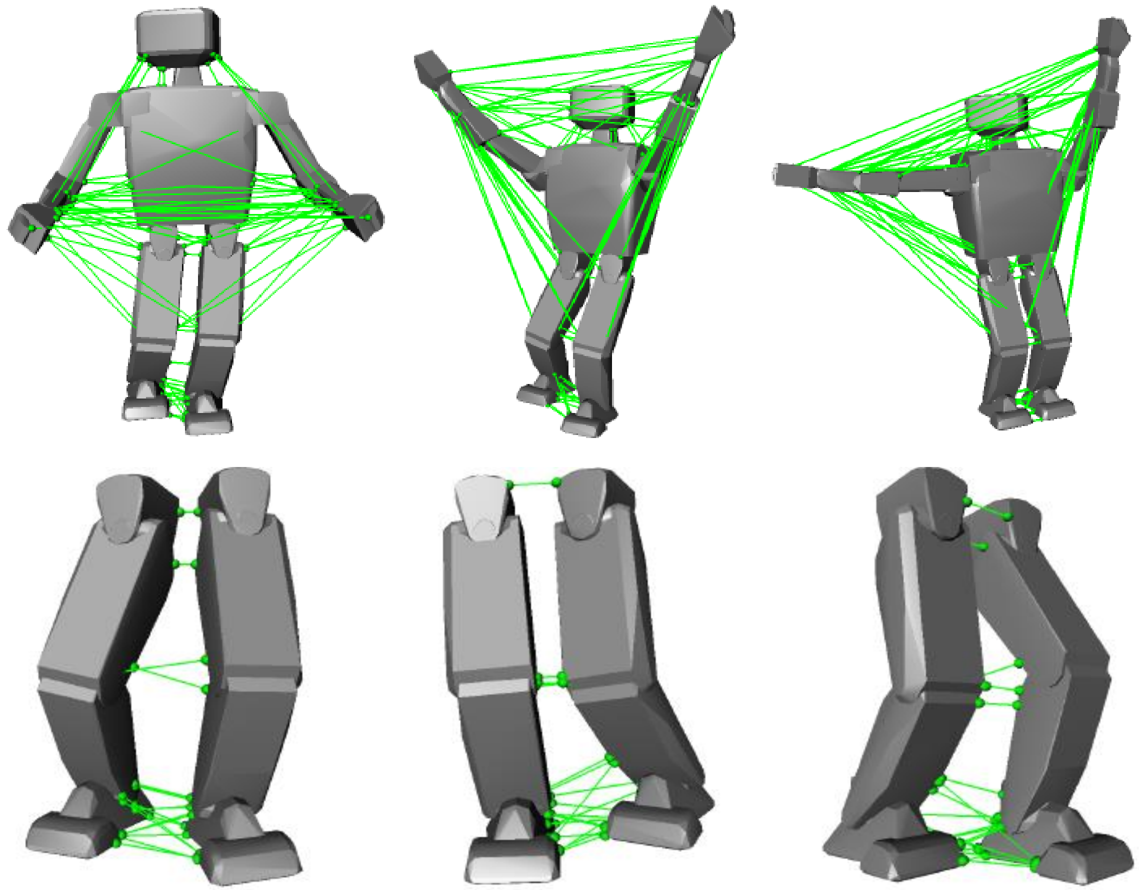
$$P = (\sum_{i=1}^{N-1} i) - (N-1) = \sum_{i=1}^{N-2} i$$

$$P = \frac{N^2 - 3N + 2}{2}$$

For $N = 31$,

$$P = 435.$$

(J. Kuffner et al. Self-Collision and Prevention for Humanoid Robots. Proc. IEEE Int. Conf. on Robotics and Automation, 2002)

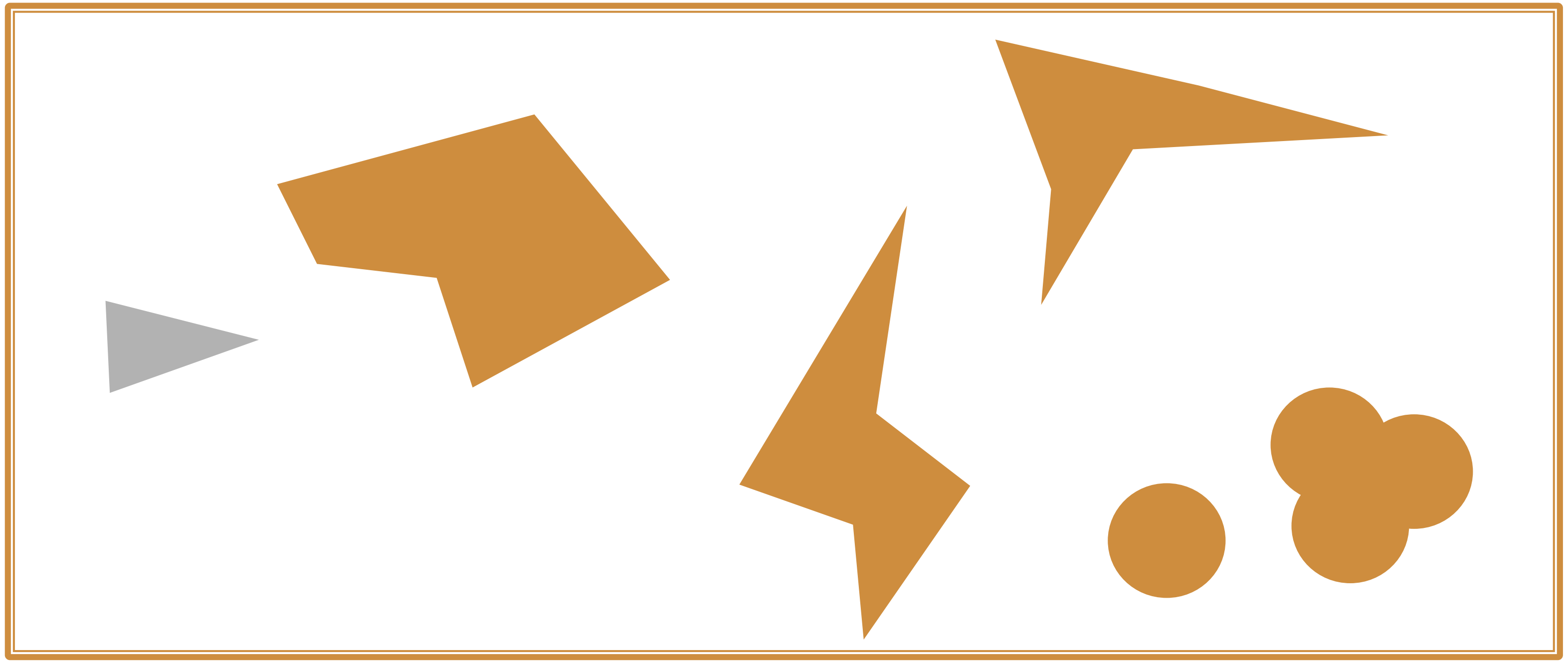


Outline

- Representing Geometry
- Methods
  - Bounding volumes
  - Bounding volume Hierarchy
- Dynamic collision detection
- Collision detection for Moving Objects
  - Feature tracking, swept-volume intersection, etc.

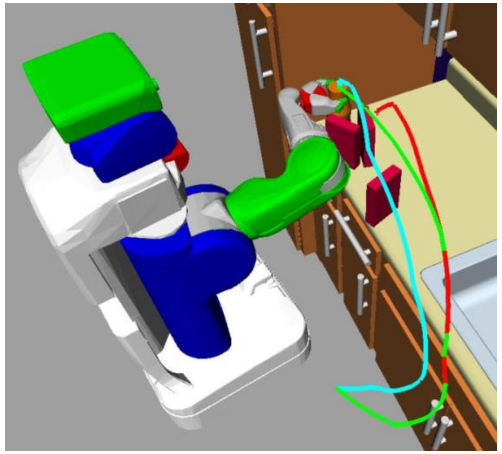
2D Representation

- 2D robots and obstacles are usually represented as
  - Polygons
  - Composites of discs

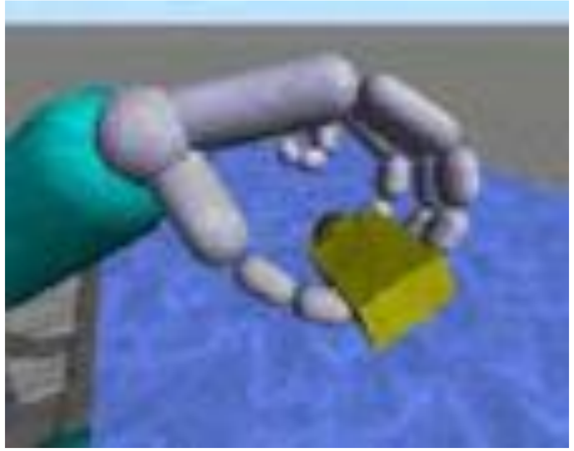


3D Representation

- Many representations - most popular for motion planning are



Triangle meshes

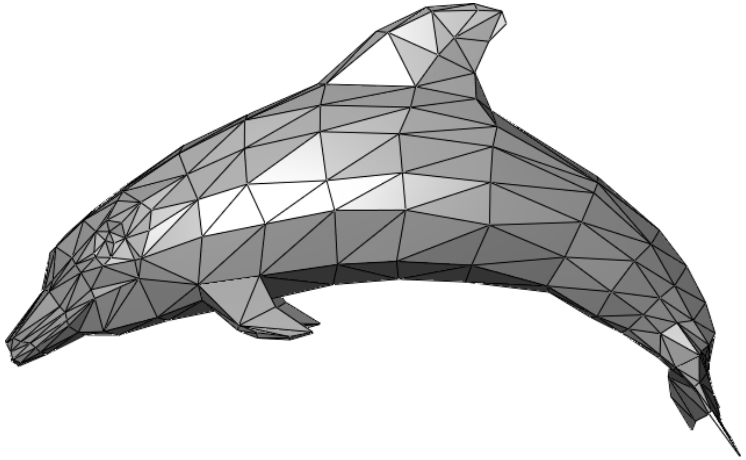


Composite of primitives

Voxel grid

3D Representation: Triangle Meshes

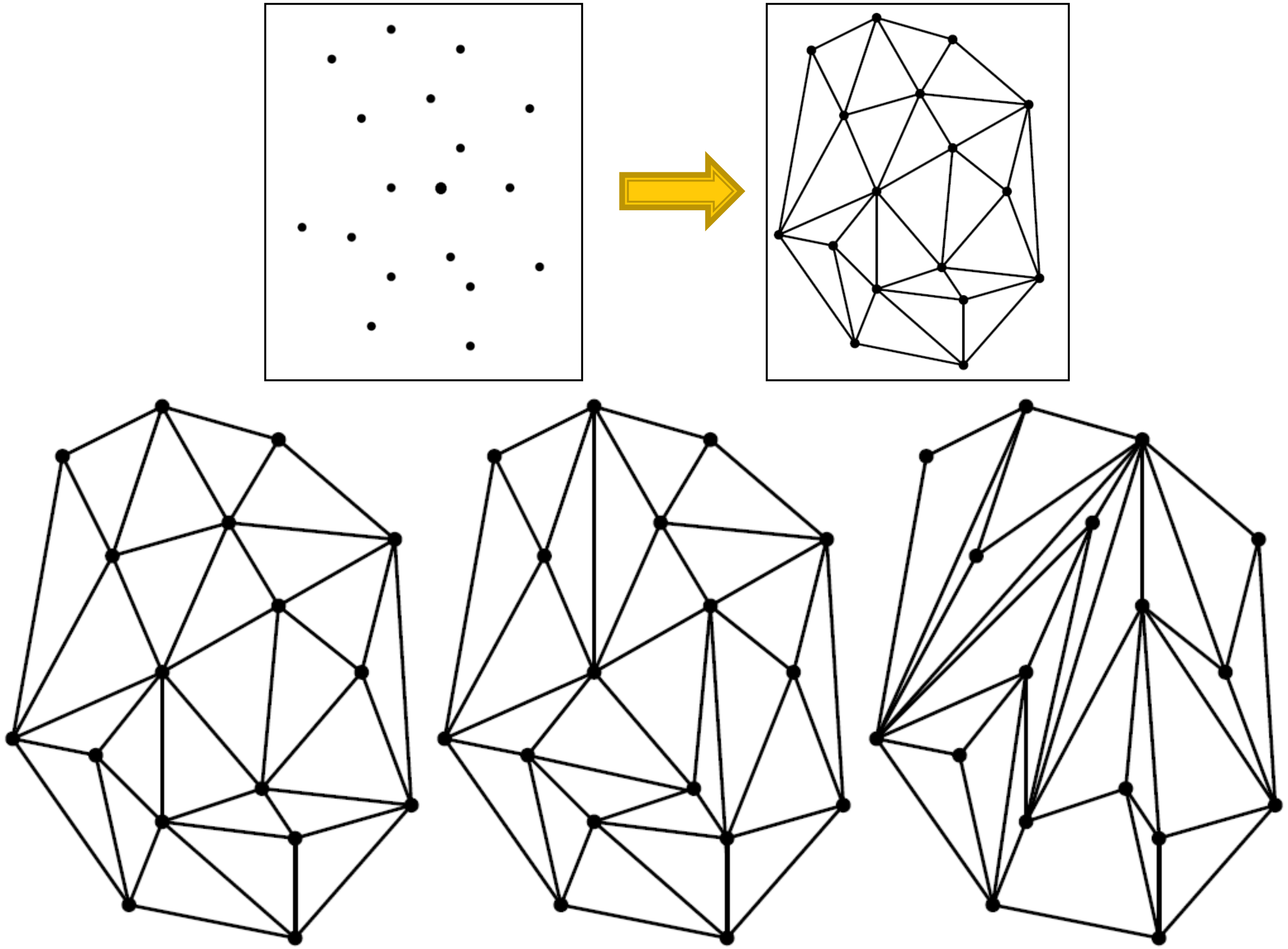
- Triangle mesh
  - A set of triangles in 3D that share common vertices and/or edges
- Most real-world shapes and be represented as triangle meshes



- Delaunay Triangulation
  - A good way to generate such meshes (there are several algorithms)

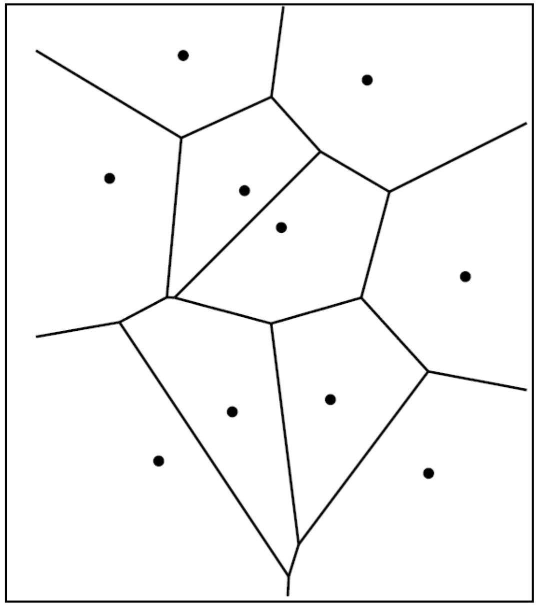
Delaunay Triangulation

- Method
  - Sample on the terrain
  - Connect Sample points
  - Which triangulation?

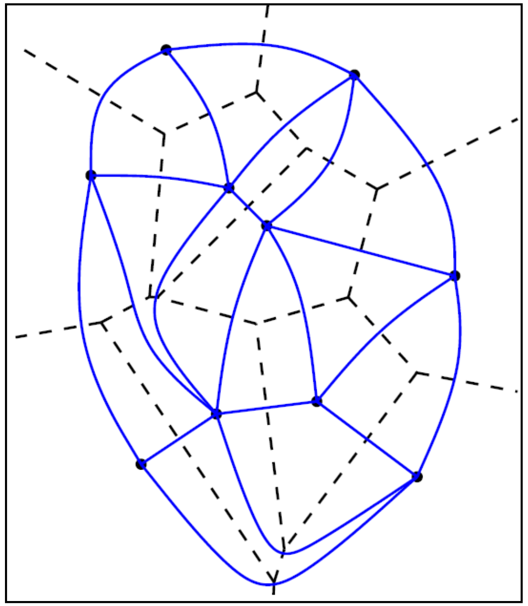


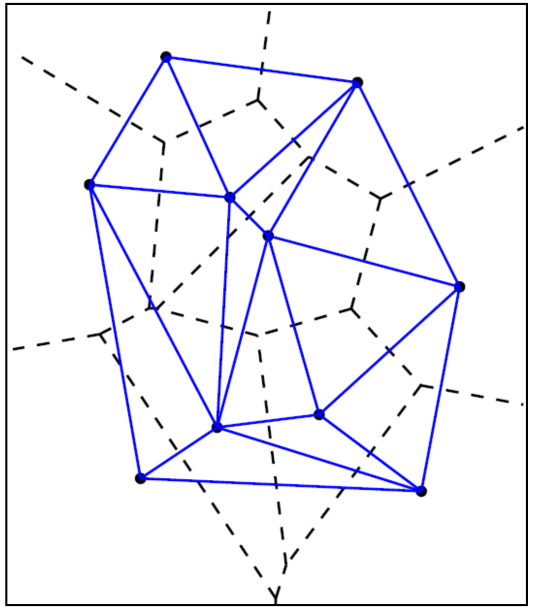
Delaunay Triangulation

- Goal – Avoid sliver triangle
  - Find the dual graph of Voronoi graph



Voronoi Graph

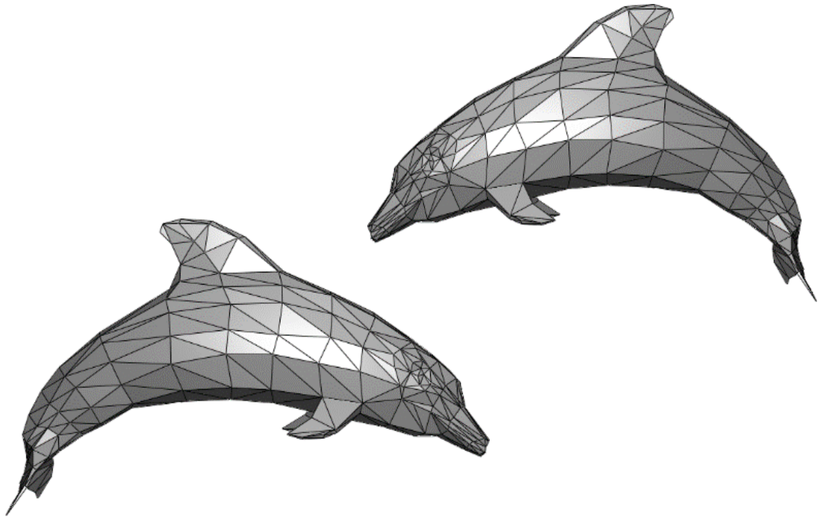




Delaunay Graph

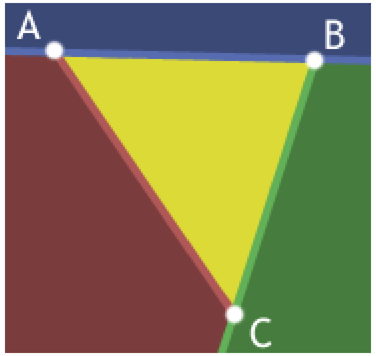
Collision Checking: Intersecting Triangle Meshes

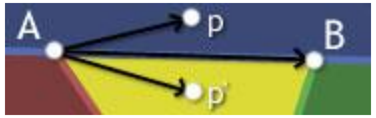
- The brute-force way
  - Check for intersection between every pair of triangles



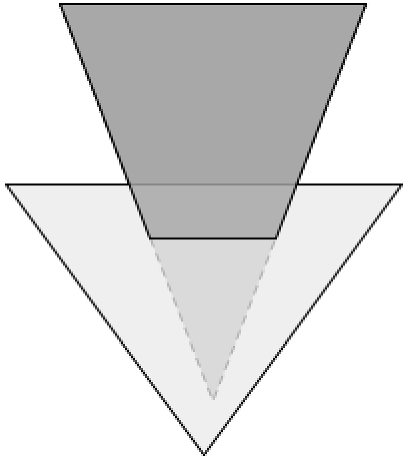
Collision Checking for Triangles

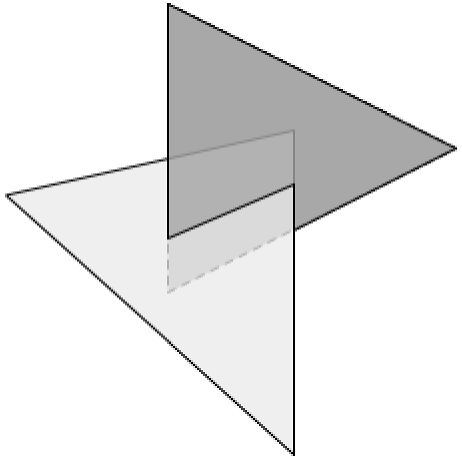
- Check if a point in a triangle





- Check if two triangles intersect





Object/Object Intersection

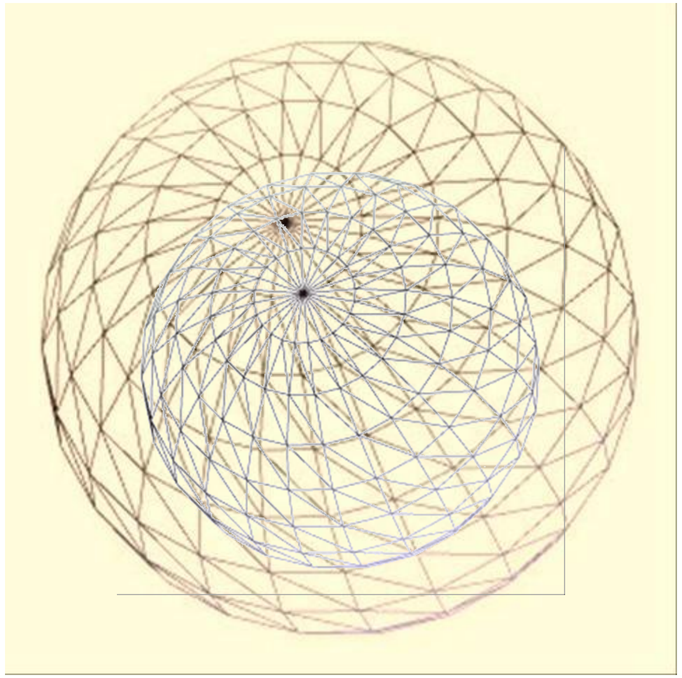
Real-Time Rendering

- Reference books
- Algorithms
  - *Link*

	ray	plane	sphere	cylinder	cone
ray	Gems p.304; SG; TGS; RTCD p.198; SoftSurfer; RTR2 p.618; RTR3 p.781	IRT p.50,88; SG; GTCG p.482; TGS; RTCD p.175; SoftSurfer (more); GPC	IRT p.39,91; Gems p.388; Held jgt 2(4); GTweb; 3DG p.16; GTCG p.501; TGS; RTCD p.127,177; RTR2 p.568; RTR3 p.738; GPC	IRT p.91; Gems IV p.356; Held jgt 2(4); GTweb; GTCG p.507; TGS; RTCD p.194	IRT p.91; Gems V p.227; Held jgt 2(4); GTweb doc; GTCG p.512
plane	IRT p.50,88; SG; GTCG p.482; TGS; RTCD p.175; SoftSurfer (more); GPC	GTweb; SG; GTCG p.529; RTCD p.207; GPC	distance of center to plane <= radius; GTweb; Gomez; GTCG p.548; TGS; RTCD p.160,219; GPC	GTweb; GTCG p.551; TGS; GTweb doc	GTCG p.563; RTCD p.164
sphere	IRT p.39,91; Gems p.388; Held jgt 2(4); GTweb; 3DG p.16; GTCG p.501; TGS; RTCD p.127,177; RTR2 p.568;	distance of center to plane <= radius; GTweb; Gomez; GTCG p.548; TGS; RTCD p.160,219;	If radii A+B >= center/center distance; Held jgt 2(4); GTweb; Gomez; GPG p.390; GTCG p.602; TGS; RTCD p.88,215,223;	If radii A+B >= center/axis distance; Held jgt 2(4); (GTCG p.602); TGS; RTCD p.114	GTweb; GTweb doc; (GTCG p.602)

Triangle Meshes

- Triangle Meshes are **hollow**!
- Be careful if you use them to represent solid bodies



One mesh inside another; no intersection

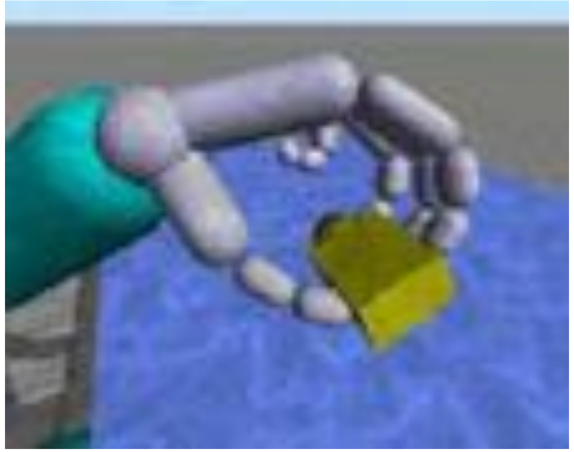
Triangle Meshes

- Complexity of collision checking increases with the number of triangles
  - Try to keep the number of triangles low

- Algorithms to simplify meshes exist but performance depends on shape

Composites of Primitives

- Advantages:
  - Can usually define these by hand
  - Collision checking is much faster/easier
  - Much better for simulation



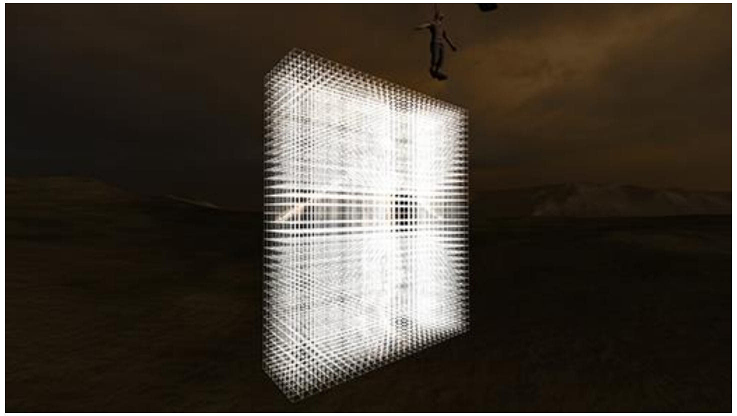
- Disadvantages
  - Hard to represent complex shapes
  - Usually conservative (i.e. overestimate true geometry)

Voxel Grids

- Voxel
  - Short for "volume pixel"
  - A single cube in a 3D lattice
- Not hollow like triangle meshes
  - Good for 'deep' physical simulations
  - E.g., heat diffusion, fracture, and soft physics

How to make a voxel model from triangle mesh?

- Step 1 - Grid the space
  - Grid resolution – without losing important details
  - Grid dimension – just enough to cover the model, but not bigger (for efficiency)



How to make a voxel model from triangle mesh?

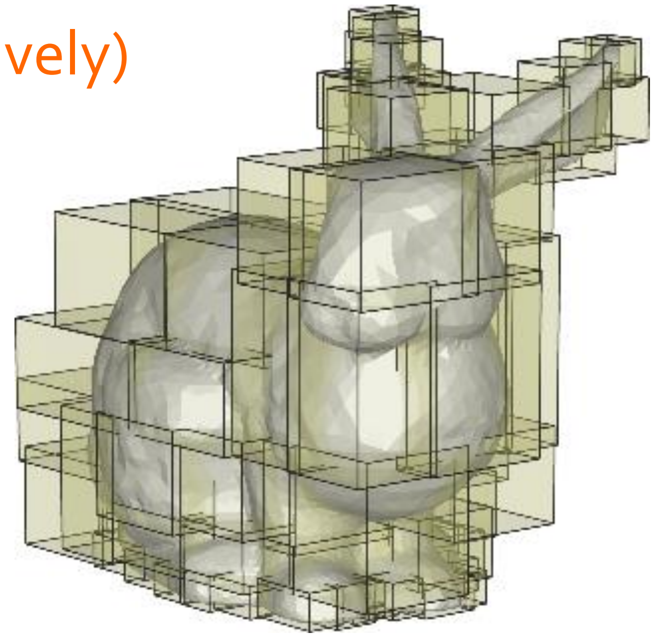
- Step 2 – Solidify a shell representing the surface
  - For every triangle, check every voxel in the triangle's bounding box to see if it intersects
  - If it does, the voxel is made solid.

How to make a voxel model from triangle mesh?

- Step 3 – Fill the shell using a scanline fill algorithm
  - Casting a ray from your point in any direction you want.
  - Count how many times the raycast intersects with your mesh.
  - Odd number count → inside of the mesh
  - Even number count →outside of the mesh

Bounding Volume

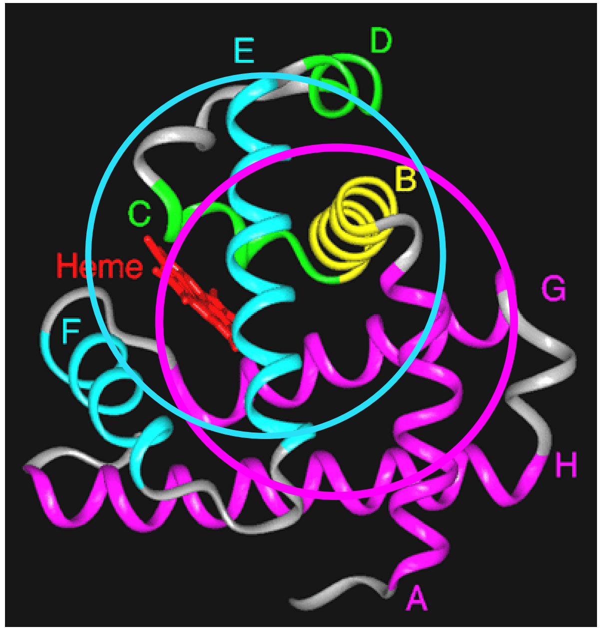
- Bounding Volume
  - A closed volume that completely contains the object (set).
  - If we don't care about getting the *true* collision,
    - Bounding volumes represents the geometry (conservatively)



- Various Bounding Volumes
  - Sphere
  - Axis-Aligned Bounding Boxes (AABBs)
  - Oriented Bound Boxes (OBBs)

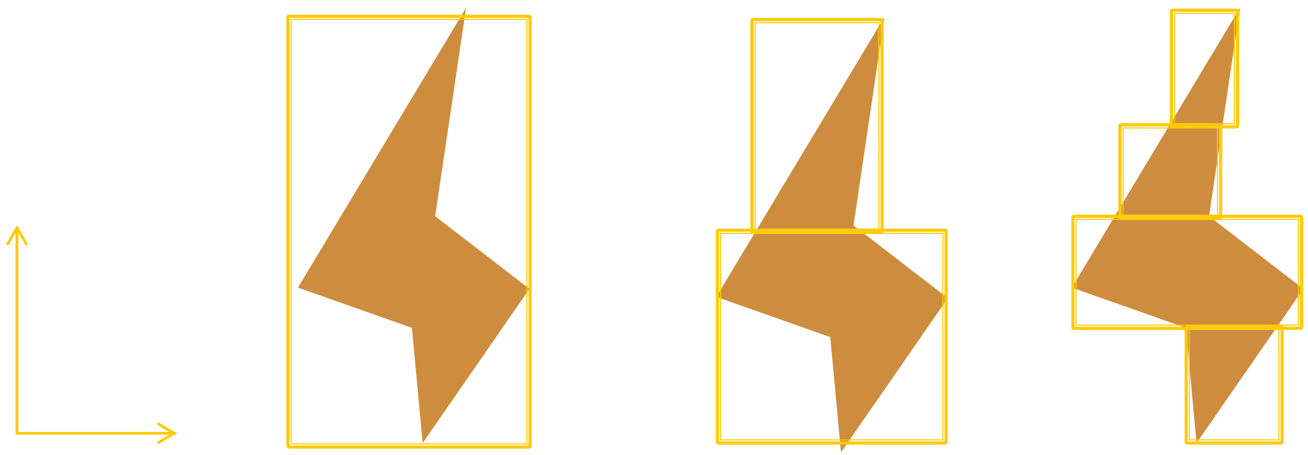
Spheres

- Invariant to rotation and translations,
  - Do not require being updated
- Efficient
  - Constructions and interference tests
- Tight?



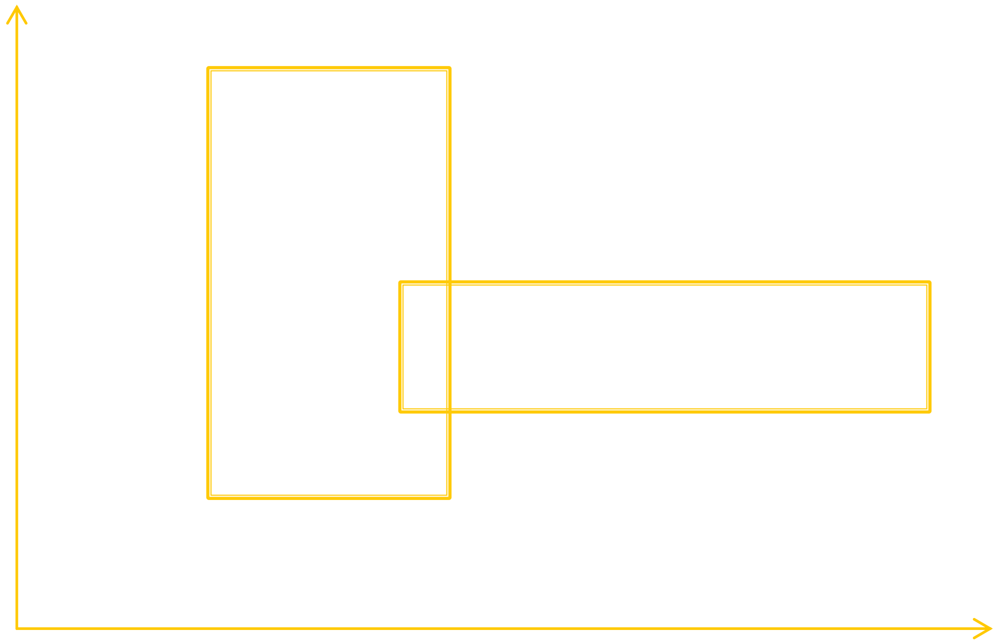
AABBs

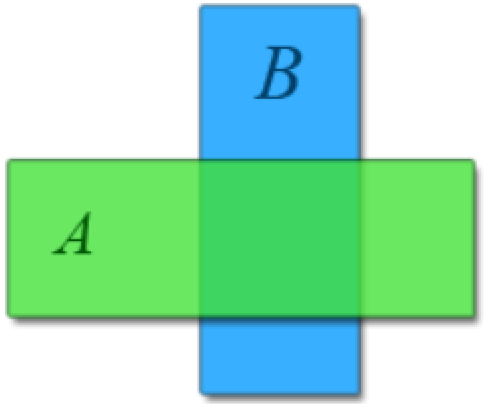
- Axis-Aligned Bounding Boxes (AABBs)
  - Bound object with one or more boxes oriented along the same axis



AABBs

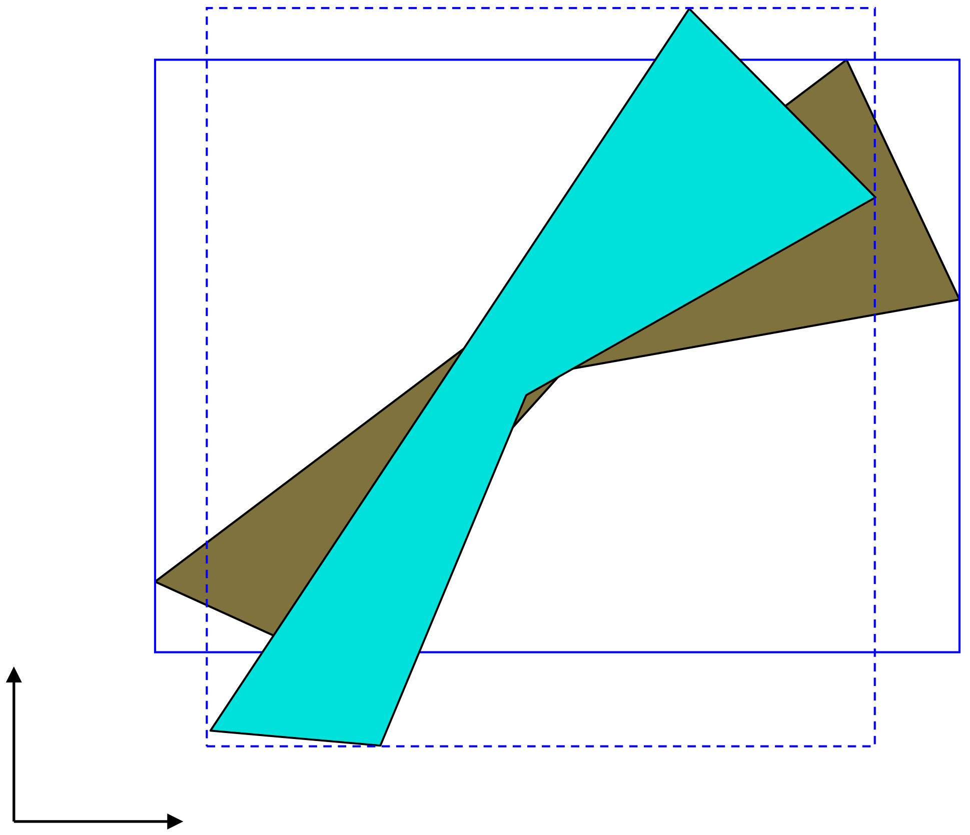
- How can you check for intersection of AABBs?





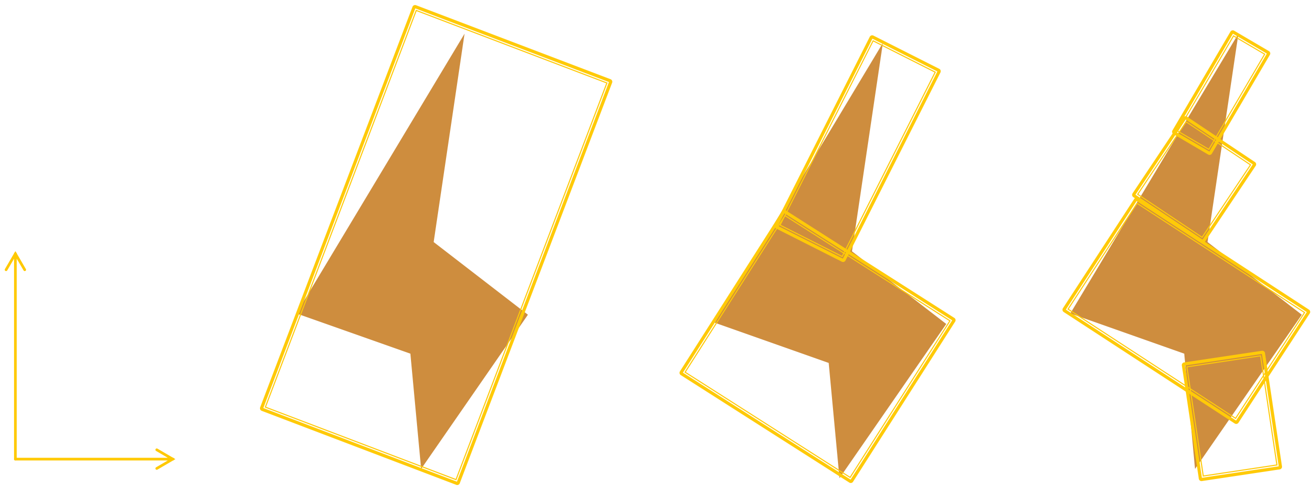
AABBs

- Not invariant
- Efficient
- Not tight



OBBs

- Oriented Bound Boxes (OBBs) are the same as AABBs except
  - The orientation of the box is not fixed

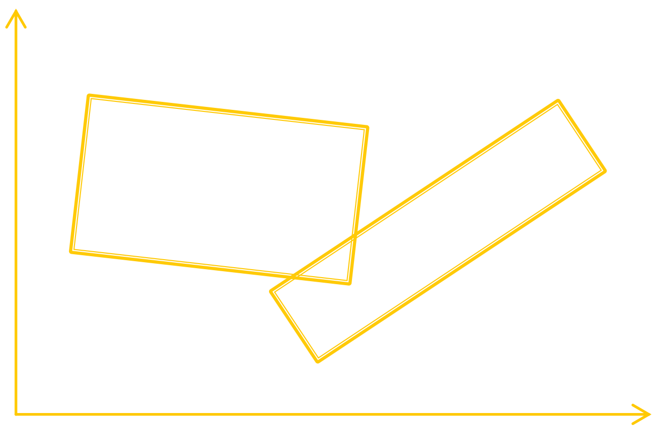


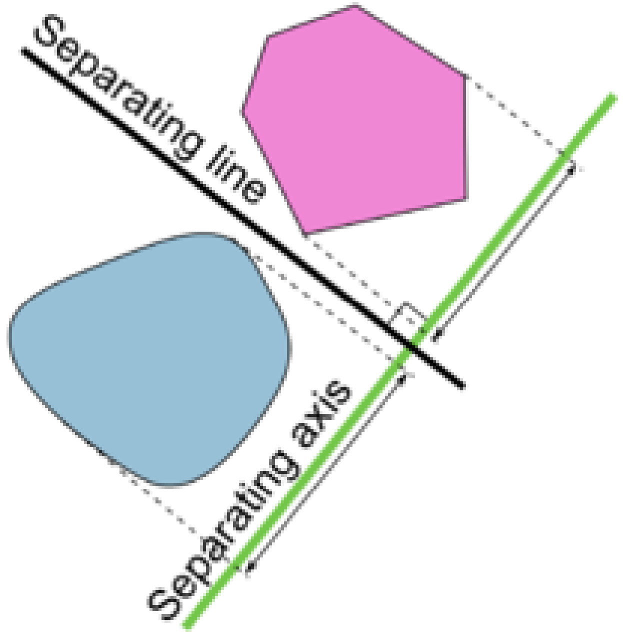
- OBBs can give you a tighter fit with fewer boxes

2D OBBs

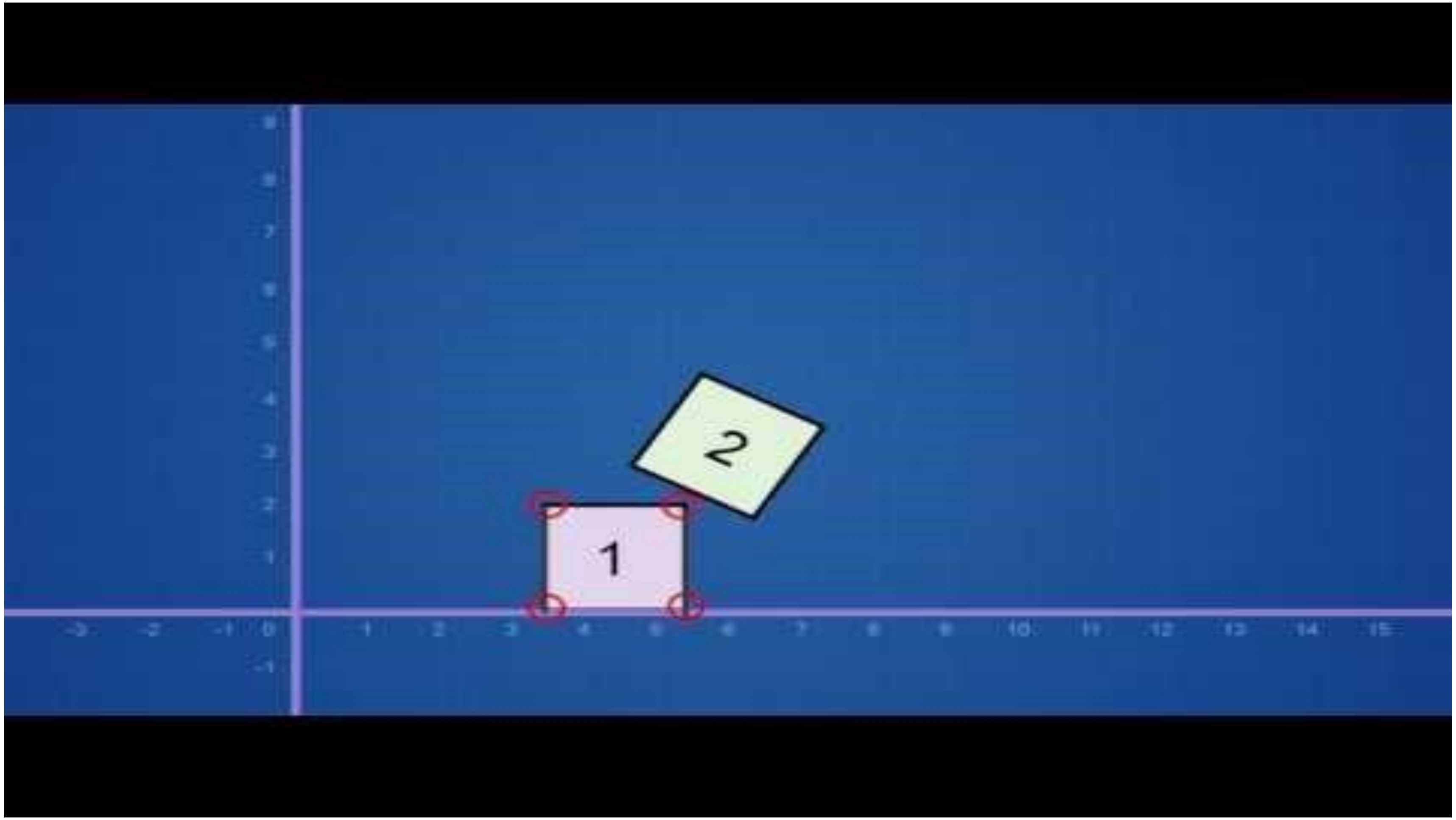
- How do you check for intersection of OBBs?
  - Hyperplane separation theorem

In 2D?

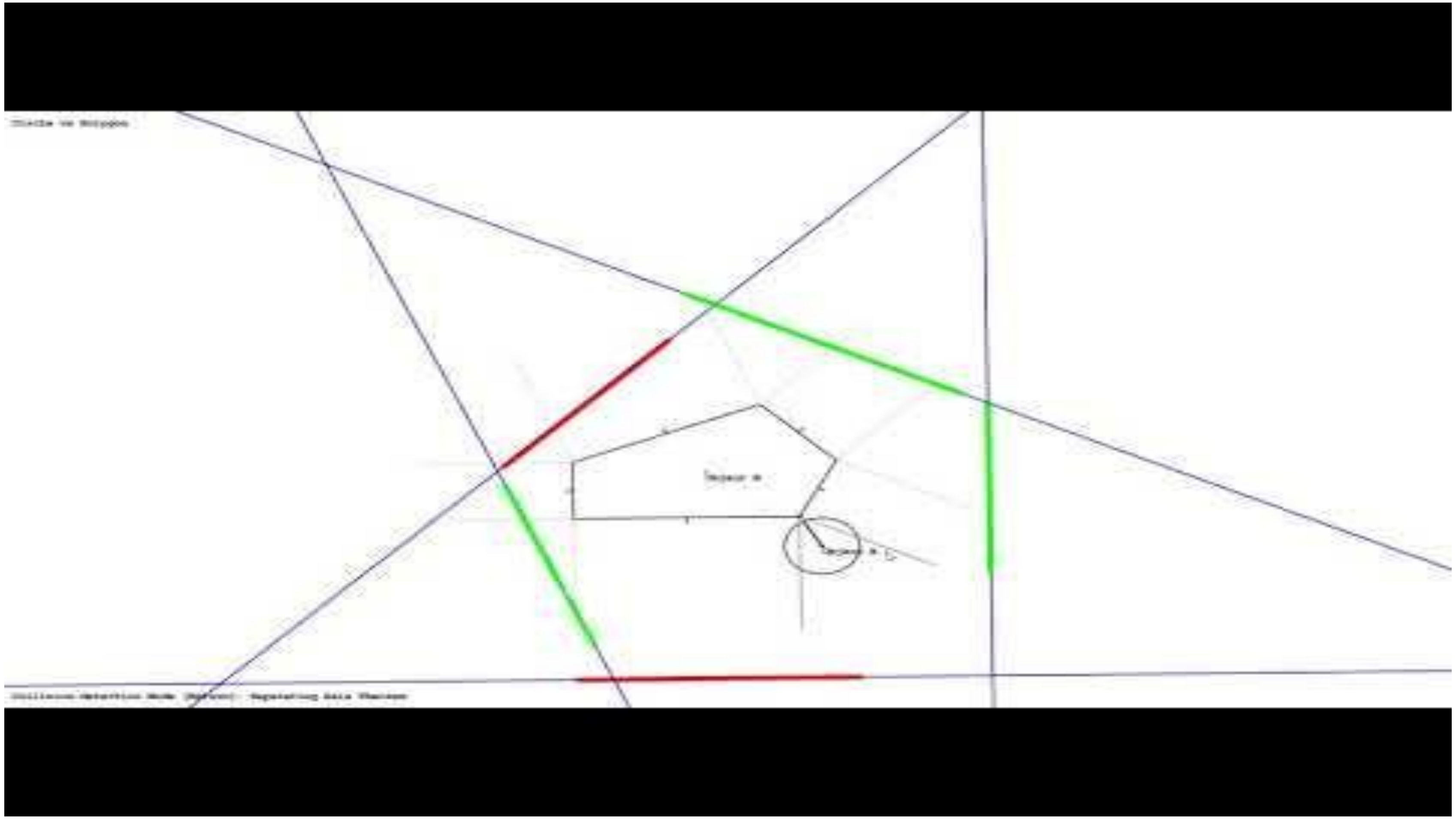




2D OBBs



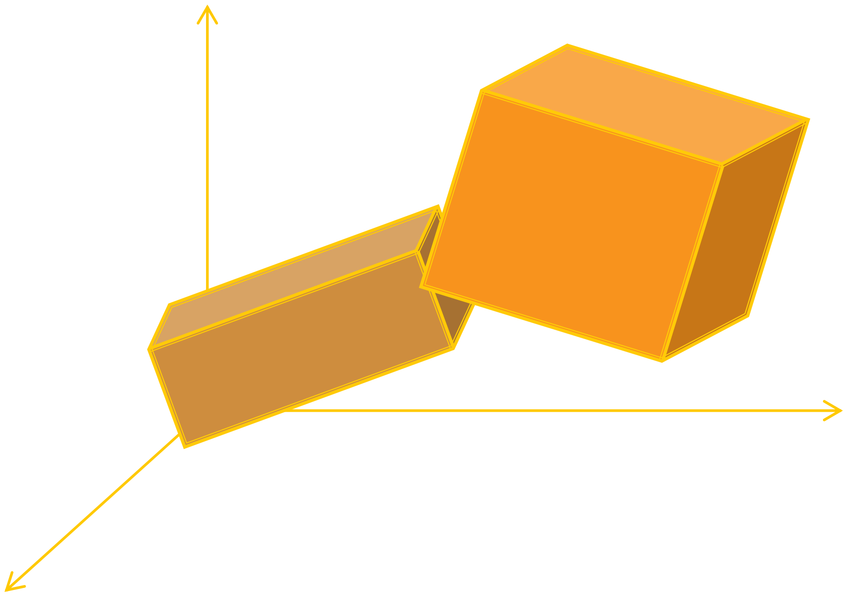
2D OBBs



3D OBBs

- Use cubical bounding box
  - OBBs have 6 faces (2 are parallel each)
  - 3 normal per OBB X 2 bounding boxes = 6
  - Additional Hyperplanes = 3x3 for the cross products
- Totally, **15** hyperplanes to project on for testing
  - If the projections have no overlap, you have no contact

In 3D?

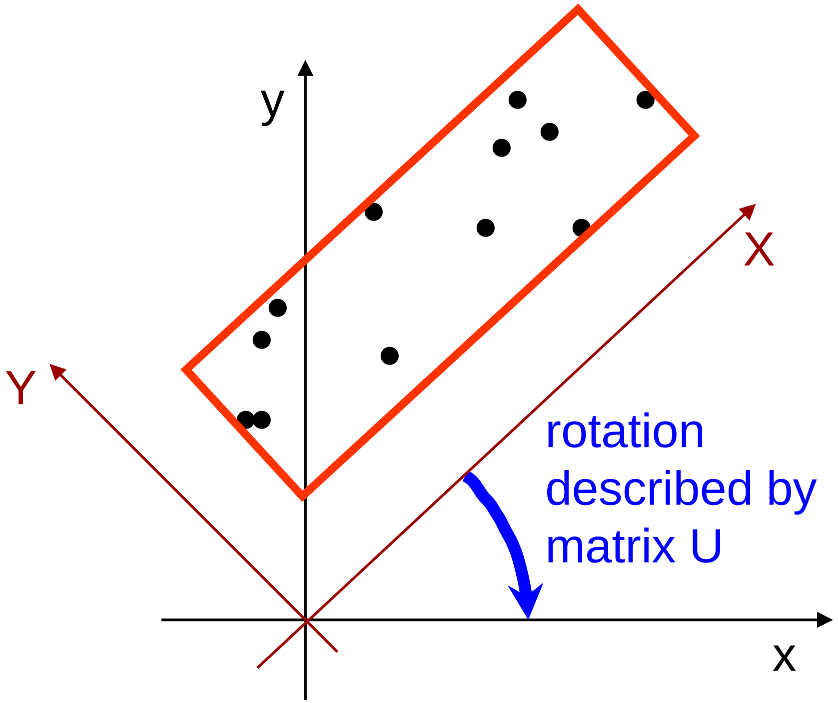


Compute OBBs

- N points $\mathbf{a}_i = (x_i, y_i, z_i)^T$, i = 1,…, N
- SVD of $A = (\mathbf{a}_1\ \mathbf{a}_2 \ldots \mathbf{a}_N)$

  $A = UDV^T$ where

  - $D = \text{diag}(s_1, s_2, s_3)$ such that $s_1 \geq s_2 \geq s_3 \geq 0$
  - U is a 3x3 rotation matrix that defines the principal axes of variance of the $\mathbf{a}_i$'s → OBB's directions



- The OBB is defined by max and min coordinates of the $\mathbf{a}_i$'s along these directions

OBBs

- Invariant
- Less efficient to test
- Tight

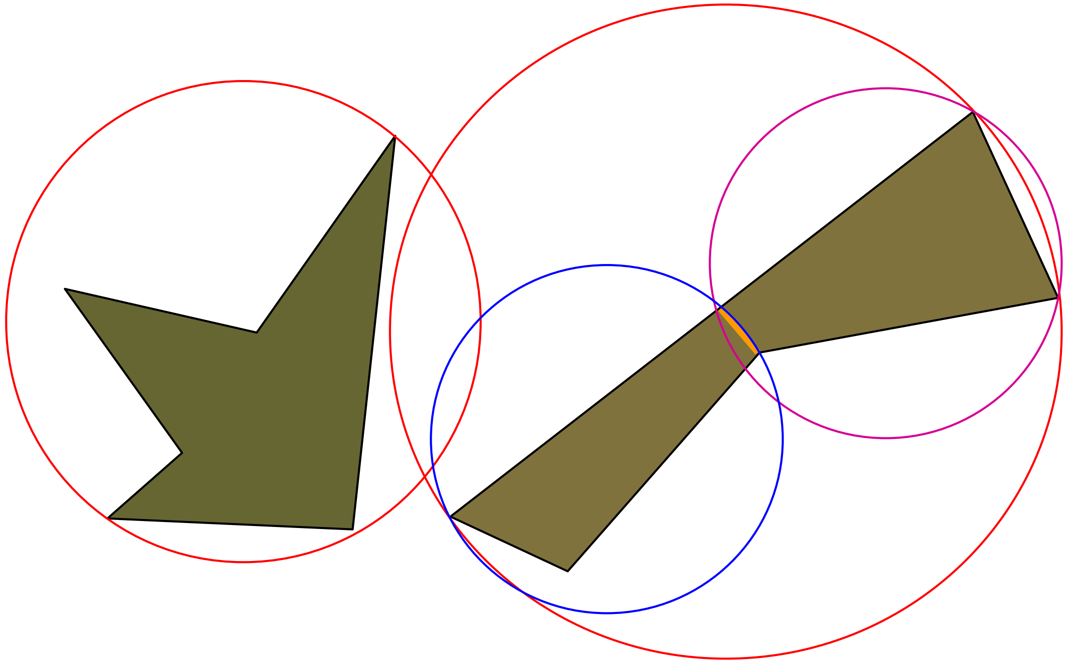
Comparison of BVs

	Sphere	AABB	OBB
Tightness	-	--	+
Testing	+	+	o
Invariance	yes	no	yes

No type of BV is optimal for all situations

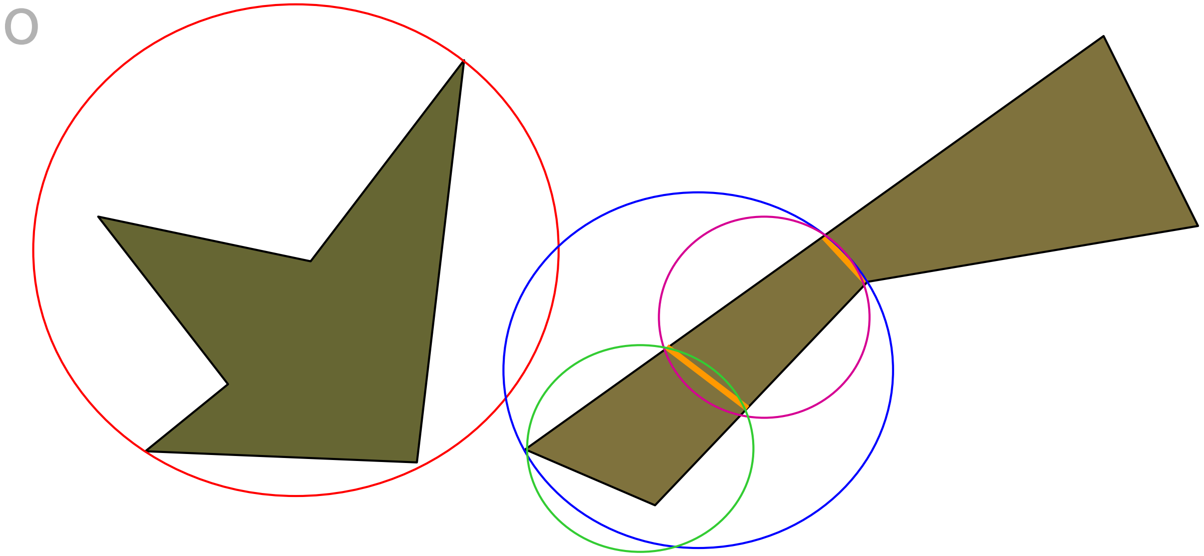
Bounding Volume Hierarchy (BVH)

- Bounding Volume Hierarchy method
  - Enclose objects into bounding volumes (spheres or boxes)
  - Check the bounding volumes first
  - Decompose an object into two



Bounding Volume Hierarchy (BVH)

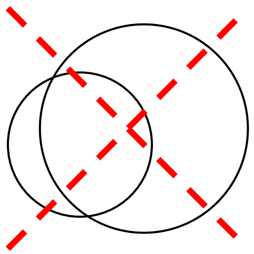
- Bounding Volume Hierarchy method
  - Enclose objects into bounding volumes (spheres or boxes)
  - Check the bounding volumes first
  - Decompose an object into two
  - **Proceed hierarchically**

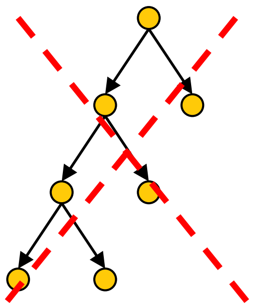


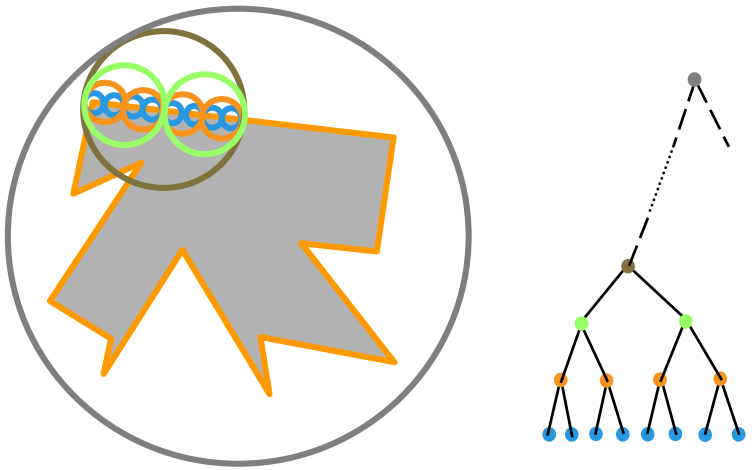
Bounding Volume Hierarchy (BVH)

- Construction
  - Not all levels of hierarchy need to have the same type of bounding volume
    - Highest level could be a sphere
    - Lowest level could be a triangle mesh

- Ideal BVH
  - Separation
  - Balanced tree

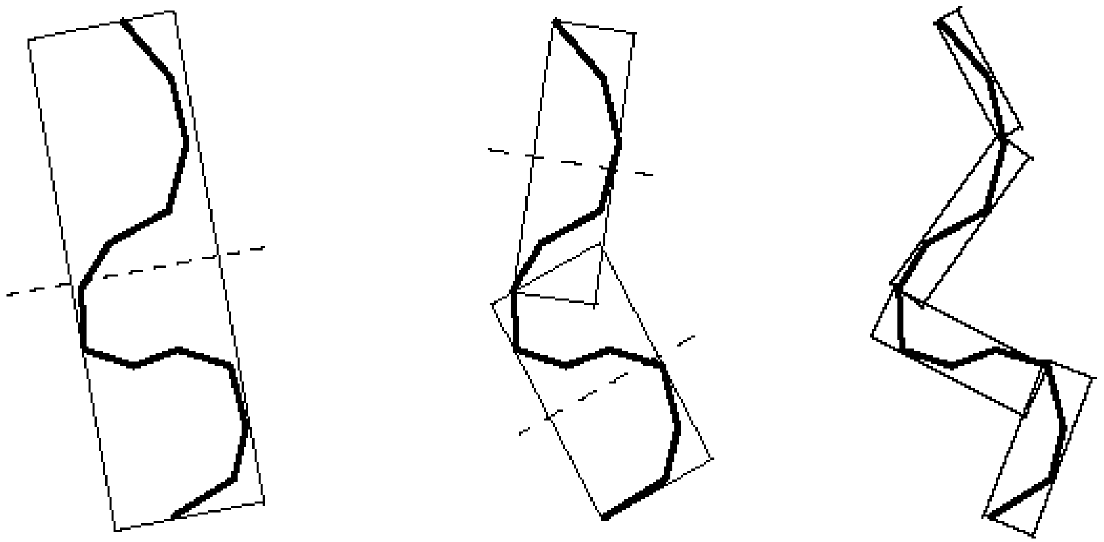




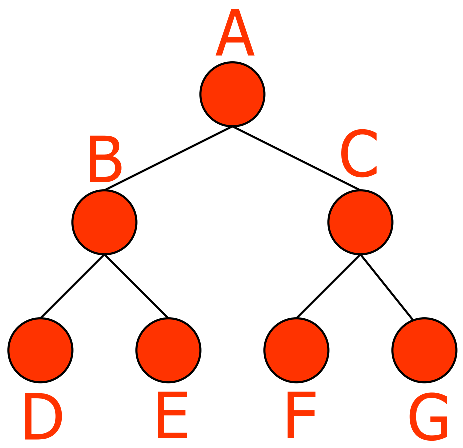


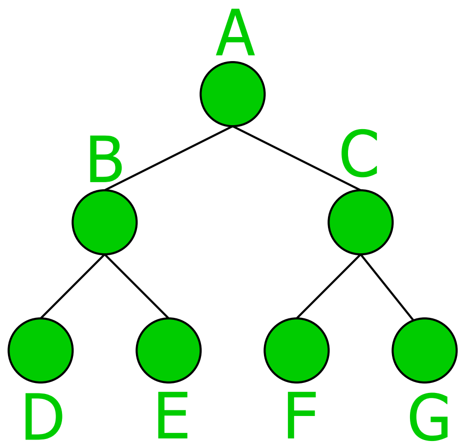
Construction of a BVH

- Top-down construction
  - At each step, create the two children of a BV
- Example
  - For OBB, split longest side at midpoint



Collision Detection using BVH

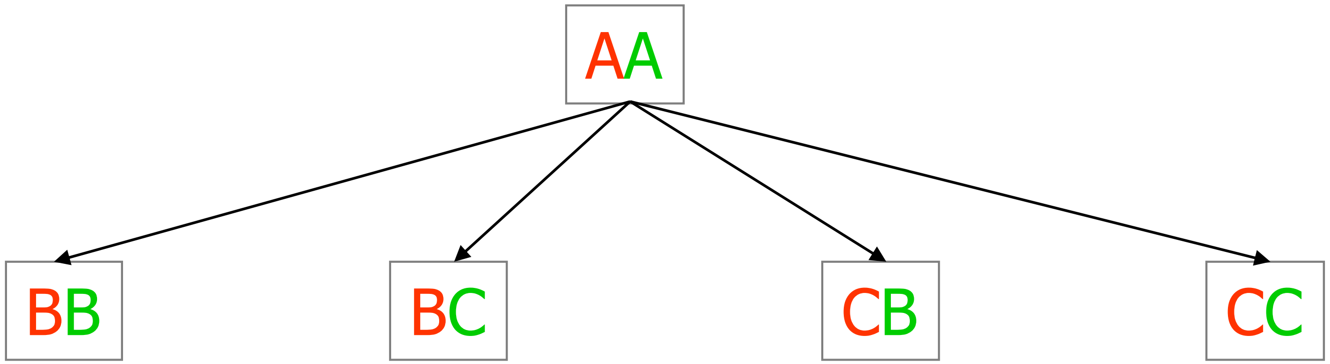


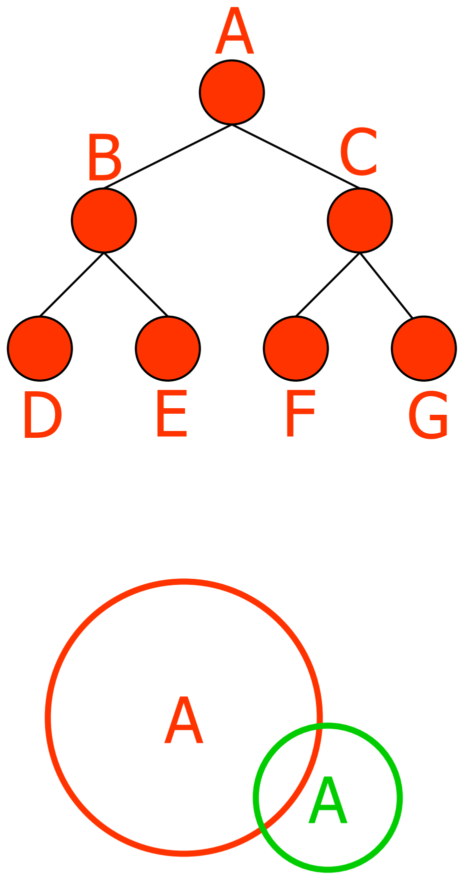


Two objects described by their precomputed BVHs

Collision Detection using BVH

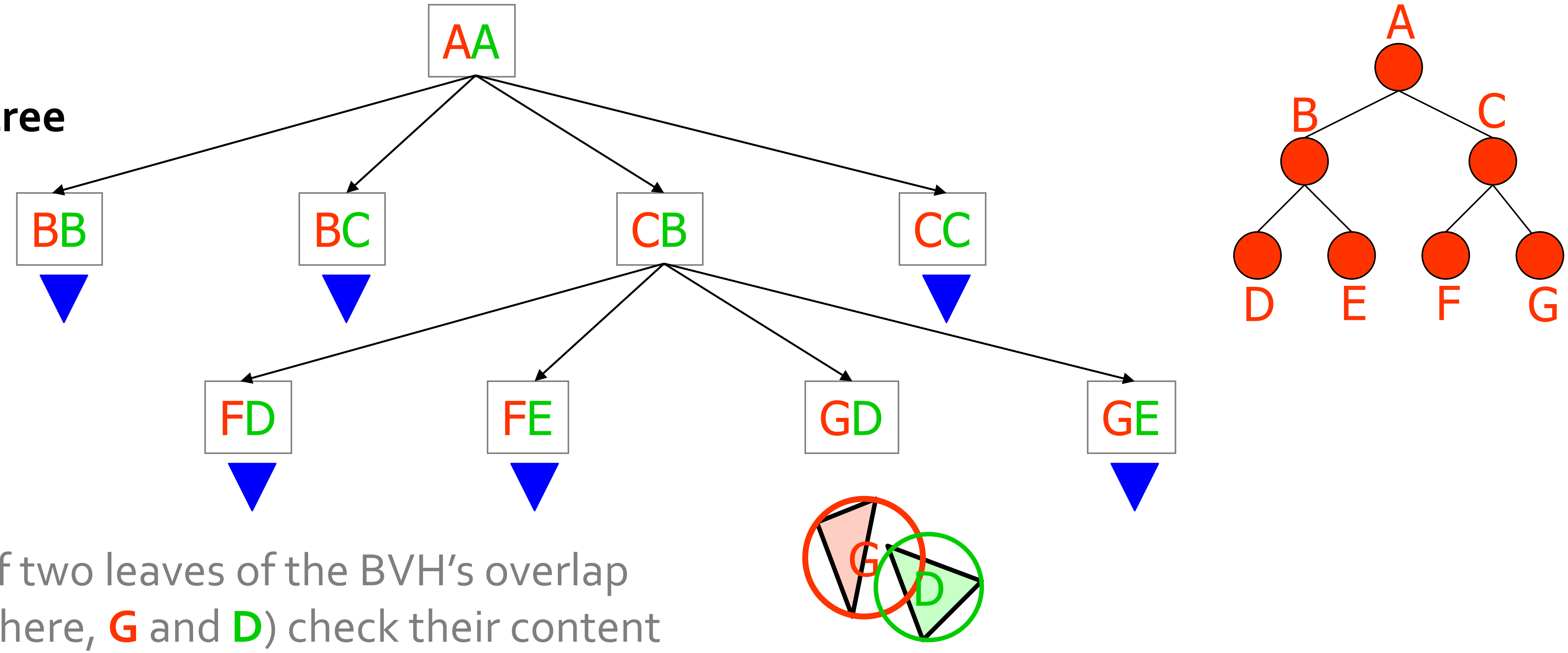
Search tree





Collision Detection with BVH

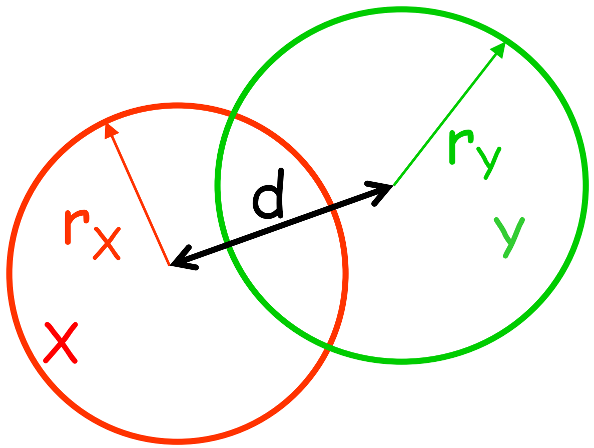
Search tree



If two leaves of the BVH's overlap (here, **G** and **D**) check their content for collision

Search Strategy

- If there is collision
  - It is desirable to detect it as quickly as possible
- **Greedy best-first search strategy** with
  - Expand the node XY with largest relative overlap (most likely to contain a collision)
  - Many ways to compute distance **d**



$$f(N) = d/(r_X+r_Y)$$

End