

# Computer Graphics

MTAT.03.015

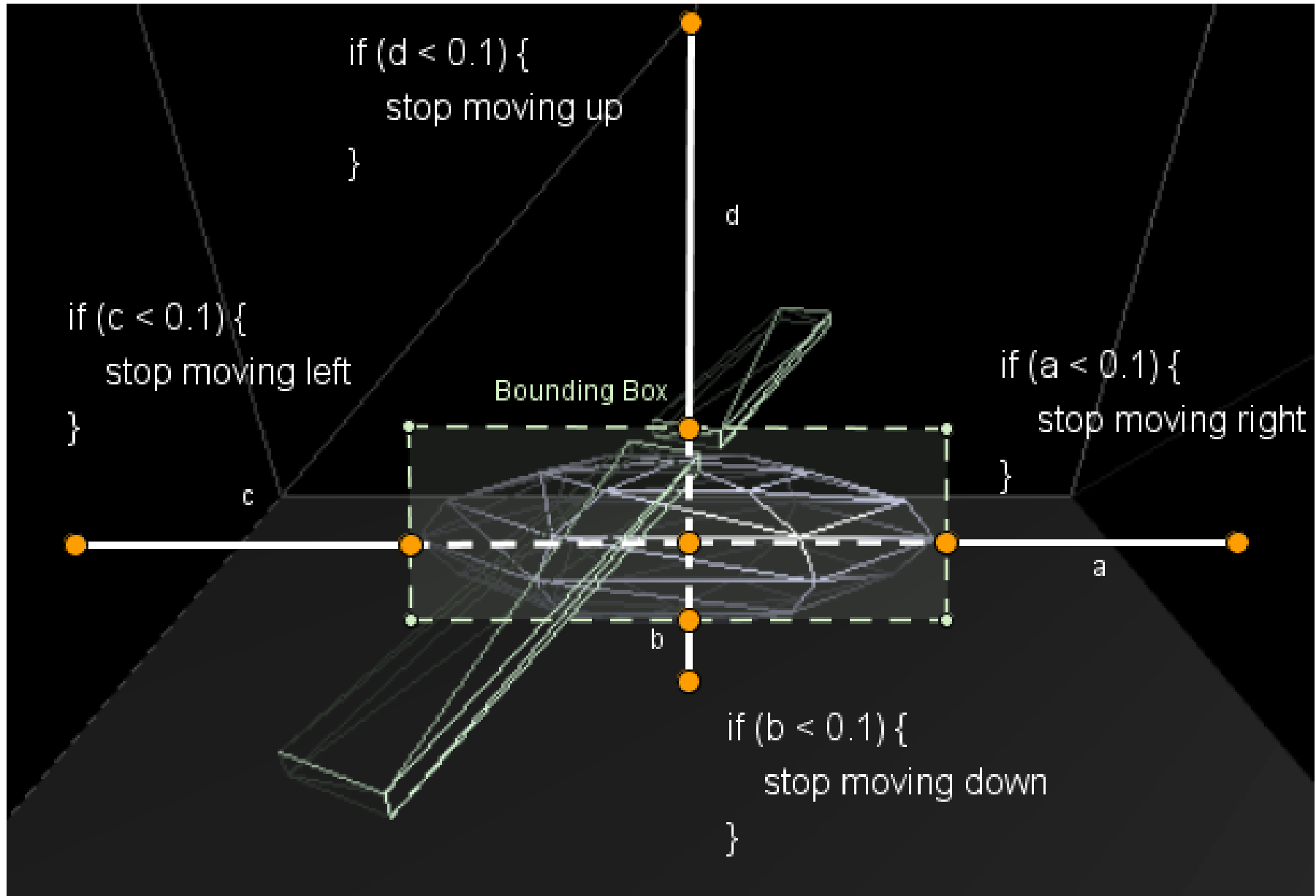
Raimond Tunnel



Study IT in .ee



# The Road So Far...

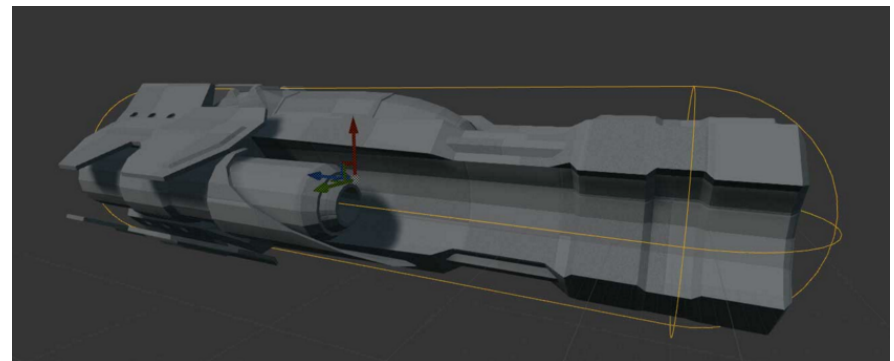
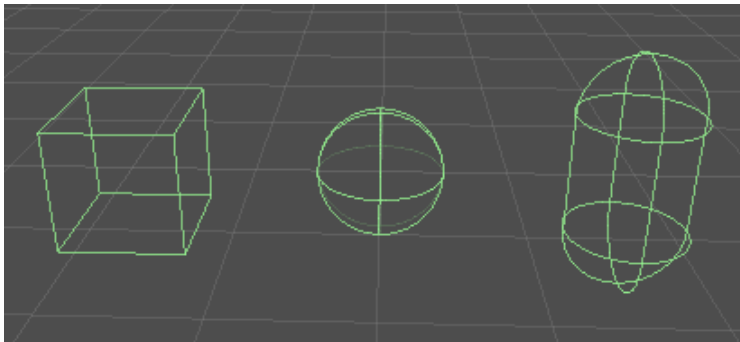


# Bounding Box

- With bounding boxes you can detect collisions between boxes.
- Our hangar just happens to be a box.
- The chopper is not a box, but the collision approximation with a bounding box seems ok.
- The bounding box is axis-aligned.
- Most of you wrote just 4 if-statements.  
That is a (kind of) bounding box collision detection for those specific boxes (around chopper, the hangar).

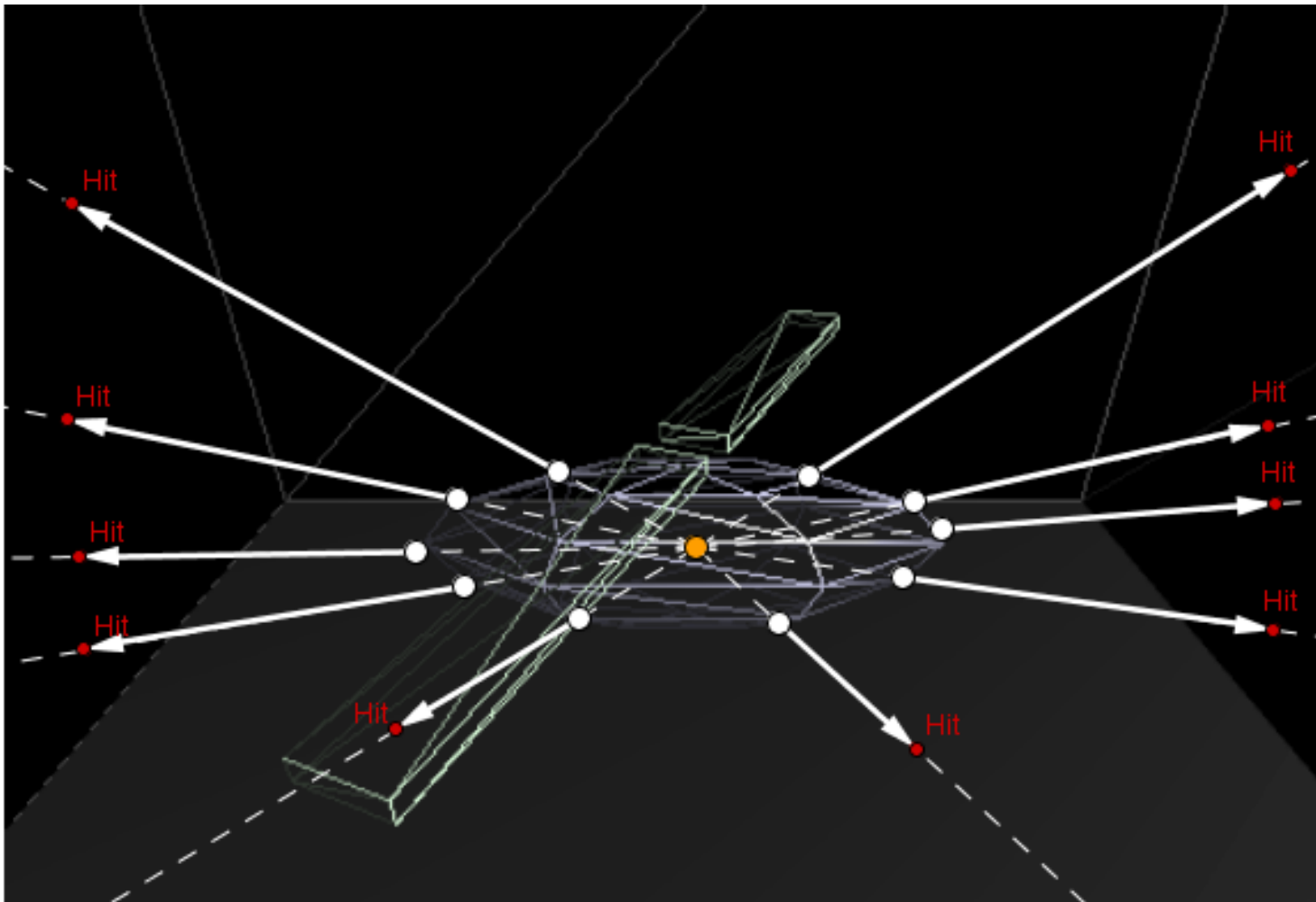
# Collision Detection

- What if the hangar walls were rotated? Can't assume, that all walls are always axis-aligned.
- What if the chopper rotated?
- The rotating blades actually would need a cylinder to minimally bound them.
- Bounding objects provide a fast and rough approximation.



# Ray Casting

- Cast rays out of some vertices, following the vertex normal.



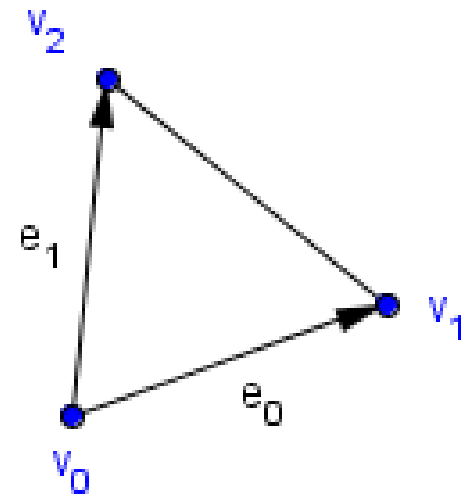
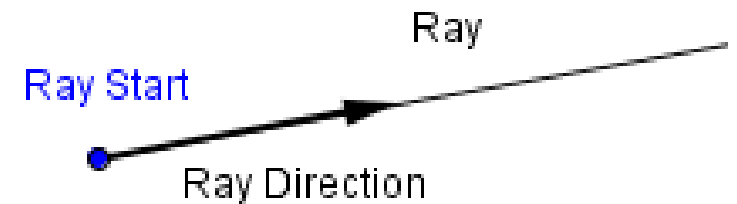
# Ray Casting

- Detect the first hit of ray and scene geometry.
- Measure the distance from the vertex to the hit.
- If the distance is too small, change the chopper's position, speed, acceleration, in order to avoid a collision.
- Intersection testing:
  - Intersection testing between a variety of objects:  
<http://www.realtimerendering.com/intersections.html>

# Möller-Trumbore Ray Triangle

- $Ray(t) = Start + t \cdot Direction$
- $Triangle(u, v) = v_0 + u \cdot e_0 + v \cdot e_1$
- The  $u$  and  $v$  are actually Barycentric coordinates of vertices  $v_1$  and  $v_2$ .
- What is the coordinate of  $v_0$ ?
- Goal is to find a solution to the following equation:

$$Ray(t) = Start + t \cdot Direction = v_0 + u \cdot e_0 + v \cdot e_1 = Triangle(u, v)$$



# Möller-Trumbore Ray Triangle

- Let us call  $S$  the *Start* and  $D$  the *Direction*.
- We can rearrange the terms to see better.

$$S + t \cdot D = (1 - u - v) v_0 + u \cdot v_1 + v \cdot v_2$$

$$S - v_0 = u \cdot (v_1 - v_0) + v \cdot (v_2 - v_0) - t \cdot D$$

← This is the same thing  
that we started with.

$$\left( \begin{array}{ccc} (v_1 - v_0) & (v_2 - v_0) & -D \end{array} \right) \cdot \begin{pmatrix} u \\ v \\ t \end{pmatrix} = S - v_0$$

- We are looking for the unknown vector

$$\begin{pmatrix} u \\ v \\ t \end{pmatrix}$$



# Möller-Trumbore Ray Triangle

- We are in 3D, so we have 3 equations for each dimension.

$$\begin{pmatrix} e_0 & e_1 & -D \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ t \end{pmatrix} = S - v_0$$

- Cramer's rule

$$x = \frac{|A_x|}{|A|} \quad y = \frac{|A_y|}{|A|} \quad z = \frac{|A_z|}{|A|}$$

$$a_{0,0} \cdot x + a_{0,1} \cdot y + a_{0,2} \cdot z = b_0$$

$$a_{1,0} \cdot x + a_{1,1} \cdot y + a_{1,2} \cdot z = b_1$$

$$a_{2,0} \cdot x + a_{2,1} \cdot y + a_{2,2} \cdot z = b_2$$

$A_x$  - first column replaced by  $b$

$A_y$  - second column replaced by  $b$

$A_z$  - third column replaced by  $b$

# Möller-Trumbore Ray Triangle

- With Cramer's rule

$$\begin{pmatrix} e_0 & e_1 & -D \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ t \end{pmatrix} = S - v_0$$

$$u = \frac{\begin{vmatrix} S_x - v_{0x} & e_{1x} & -D_x \\ S_y - v_{0y} & e_{1y} & -D_y \\ S_z - v_{0z} & e_{1z} & -D_z \end{vmatrix}}{\begin{vmatrix} e_{0x} & e_{1x} & -D_x \\ e_{0y} & e_{1y} & -D_y \\ e_{0z} & e_{1z} & -D_z \end{vmatrix}}$$

- Denote columns

$$b = S - v_0$$

$$u = \frac{\begin{vmatrix} b & e_1 & -D \end{vmatrix}}{\begin{vmatrix} e_0 & e_1 & -D \end{vmatrix}} \quad v = \frac{\begin{vmatrix} e_0 & b & -D \end{vmatrix}}{\begin{vmatrix} e_0 & e_1 & -D \end{vmatrix}} \quad t = \frac{\begin{vmatrix} e_0 & e_1 & b \end{vmatrix}}{\begin{vmatrix} e_0 & e_1 & -D \end{vmatrix}}$$

How to find those determinants?

# Möller-Trumbore Ray Triangle

- Scalar triple product:  $a \cdot (b \times c) = \begin{vmatrix} a & b & c \end{vmatrix}$

$$u = \frac{b \cdot (e_1 \times -D)}{e_0 \cdot (e_1 \times -D)} \quad v = \frac{e_0 \cdot (b \times -D)}{e_0 \cdot (e_1 \times -D)} \quad t = \frac{e_0 \cdot (e_1 \times b)}{e_0 \cdot (e_1 \times -D)}$$

- Anticommutativity of the cross product:

$$u = \frac{b \cdot (D \times e_1)}{e_0 \cdot (D \times e_1)} \quad v = \frac{e_0 \cdot (D \times b)}{e_0 \cdot (D \times e_1)} \quad t = \frac{e_0 \cdot (e_1 \times b)}{e_0 \cdot (D \times e_1)}$$

- Circular shift invariance of scalar triple product

$$v = \frac{D \cdot (b \times e_0)}{e_0 \cdot (D \times e_1)} \quad t = \frac{e_1 \cdot (b \times e_0)}{e_0 \cdot (D \times e_1)}$$

# Möller-Trumbore Ray Triangle

- We can calculate only two cross products

$$u = \frac{b \cdot (D \times e_1)}{e_0 \cdot (D \times e_1)} \quad v = \frac{D \cdot (b \times e_0)}{e_0 \cdot (D \times e_1)} \quad t = \frac{e_1 \cdot (b \times e_0)}{e_0 \cdot (D \times e_1)}$$

$$u = \frac{b \cdot P}{\hat{P}} \quad v = \frac{D \cdot Q}{\hat{P}} \quad t = \frac{e_1 \cdot Q}{\hat{P}} \quad \begin{aligned} Q &= (b \times e_0) \\ P &= (D \times e_1) \\ \hat{P} &= e_0 \cdot P \end{aligned}$$

- What happens if:  $\hat{P} = e_0 \cdot (D \times e_1) \sim 0$

$$\hat{P} = e_0 \cdot (D \times e_1) < 0 \quad \hat{P} = e_0 \cdot (D \times e_1) > 0$$



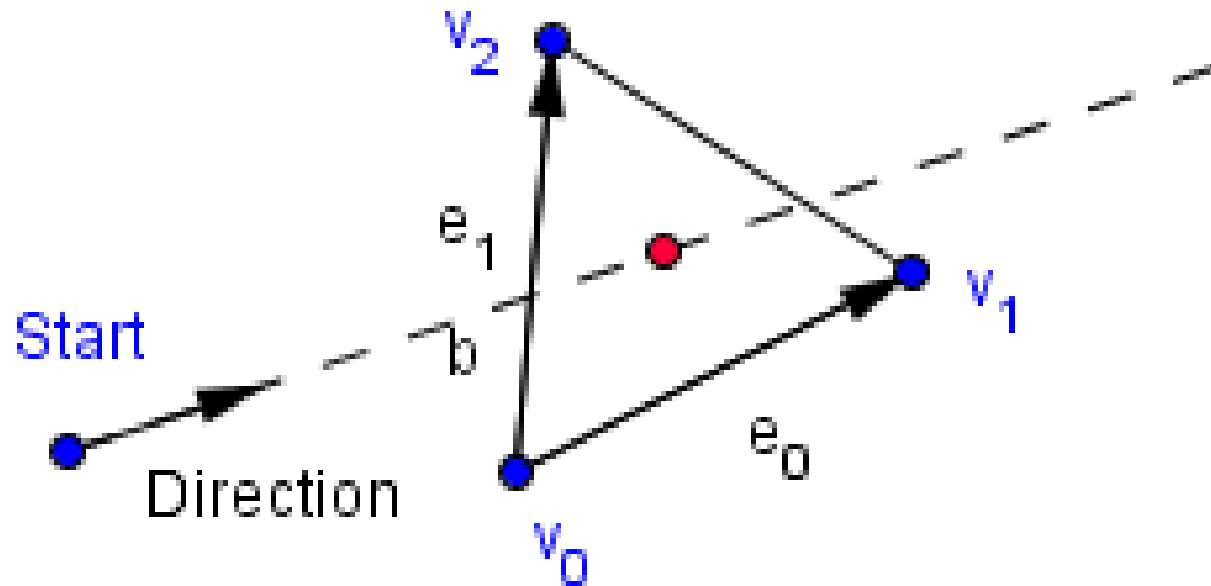
Circular shift can help to visualize this better...

# Möller-Trumbore Ray Triangle

- Can it happen, and what does it mean?

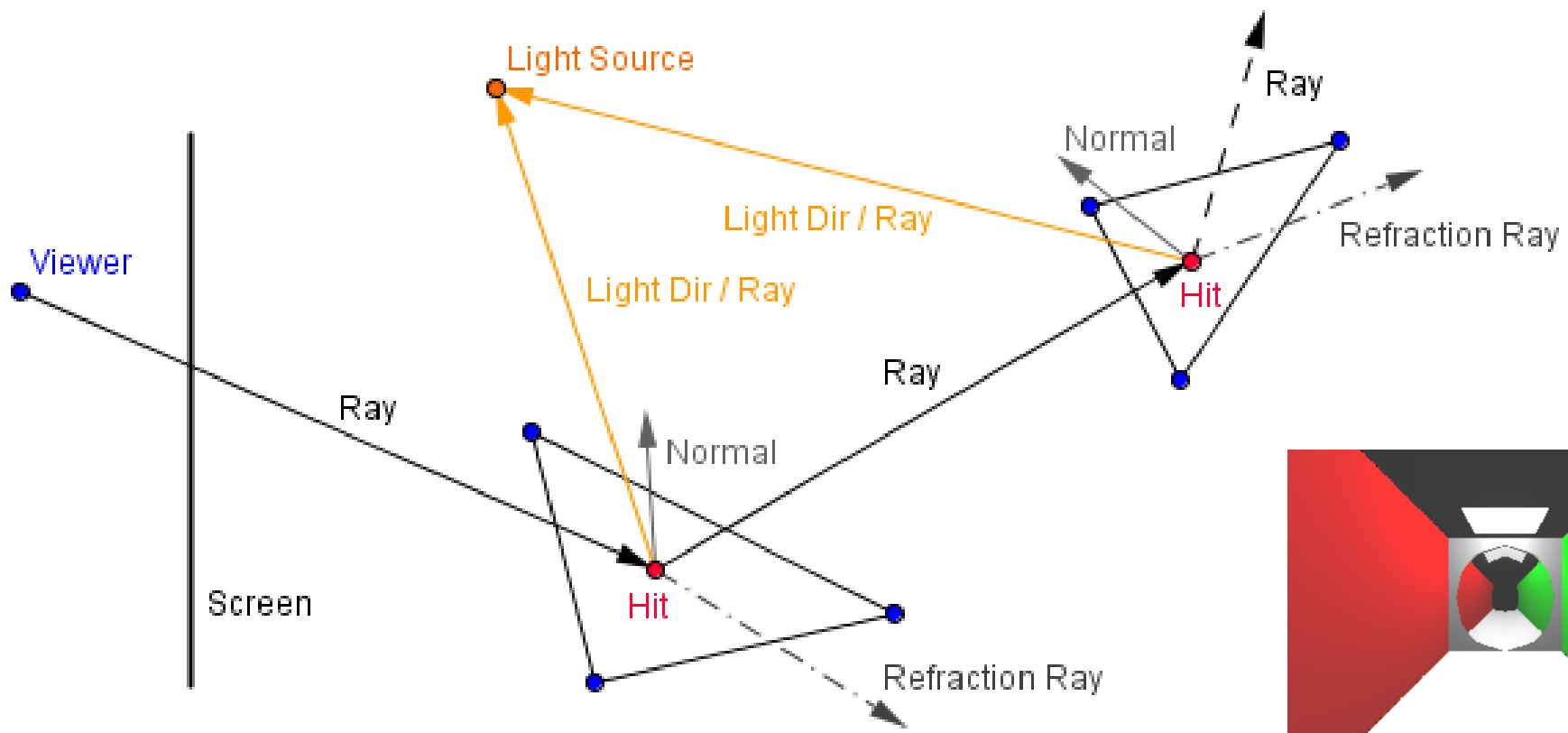
$$u < 0 \quad u > 1 \quad v < 0 \quad v > 1 \quad u + v > 1 \quad t \leq 0$$

$$\text{Start} + t \cdot \text{Direction} = v_0 + u \cdot e_0 + v \cdot e_1$$



# Ray Trace Rendering

- What is the origin of a ray?
- What about the direction?



# Ray Trace Rendering

- Accurate way to model reflective / refractive surfaces.
- Quite expensive, we need to test each ray against our geometry.

$$800 \cdot 600 \cdot 3 \cdot 700 = 1008000000$$

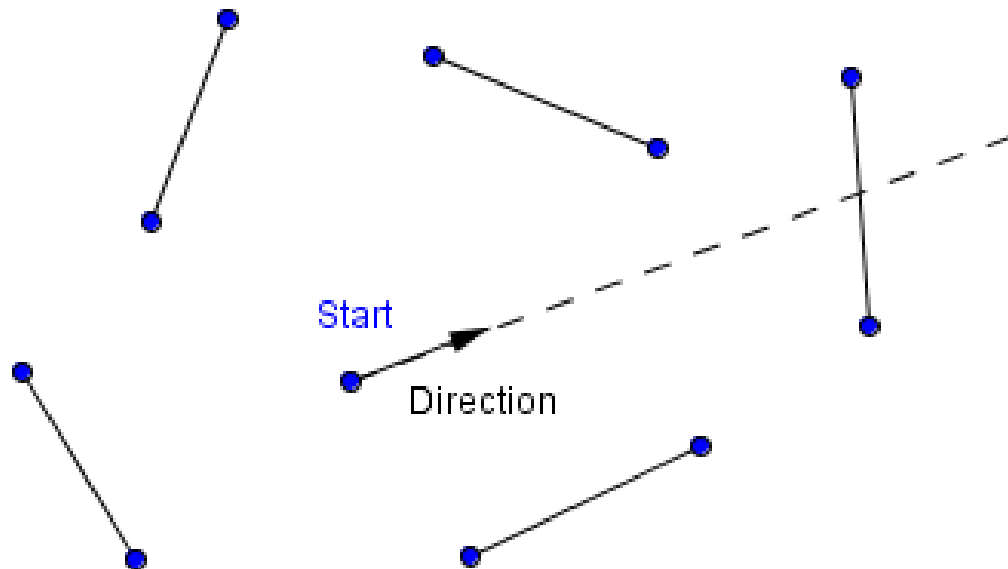
screen width, height      bounces      triangles (quite few)

number of rays

That is over a billion  
intersection tests each frame!

# Space Partitioning

- We can keep our objects in a structure, that lessens the number of intersections we need to test.
- Imagine in 2D a ray and a some line segments.

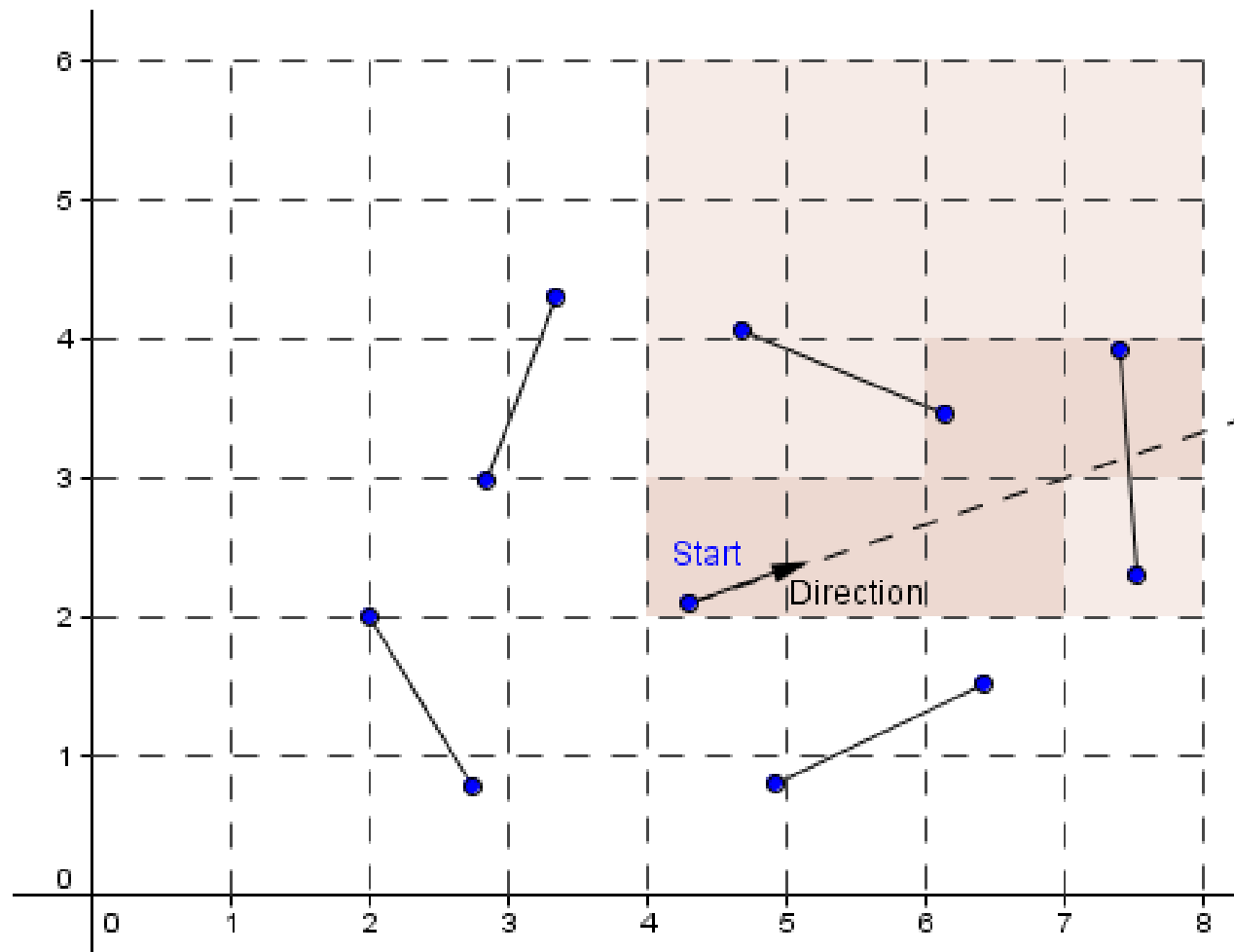


Any ideas, how to lessen the number of tests?



# First Idea: Axis-Aligned Grid

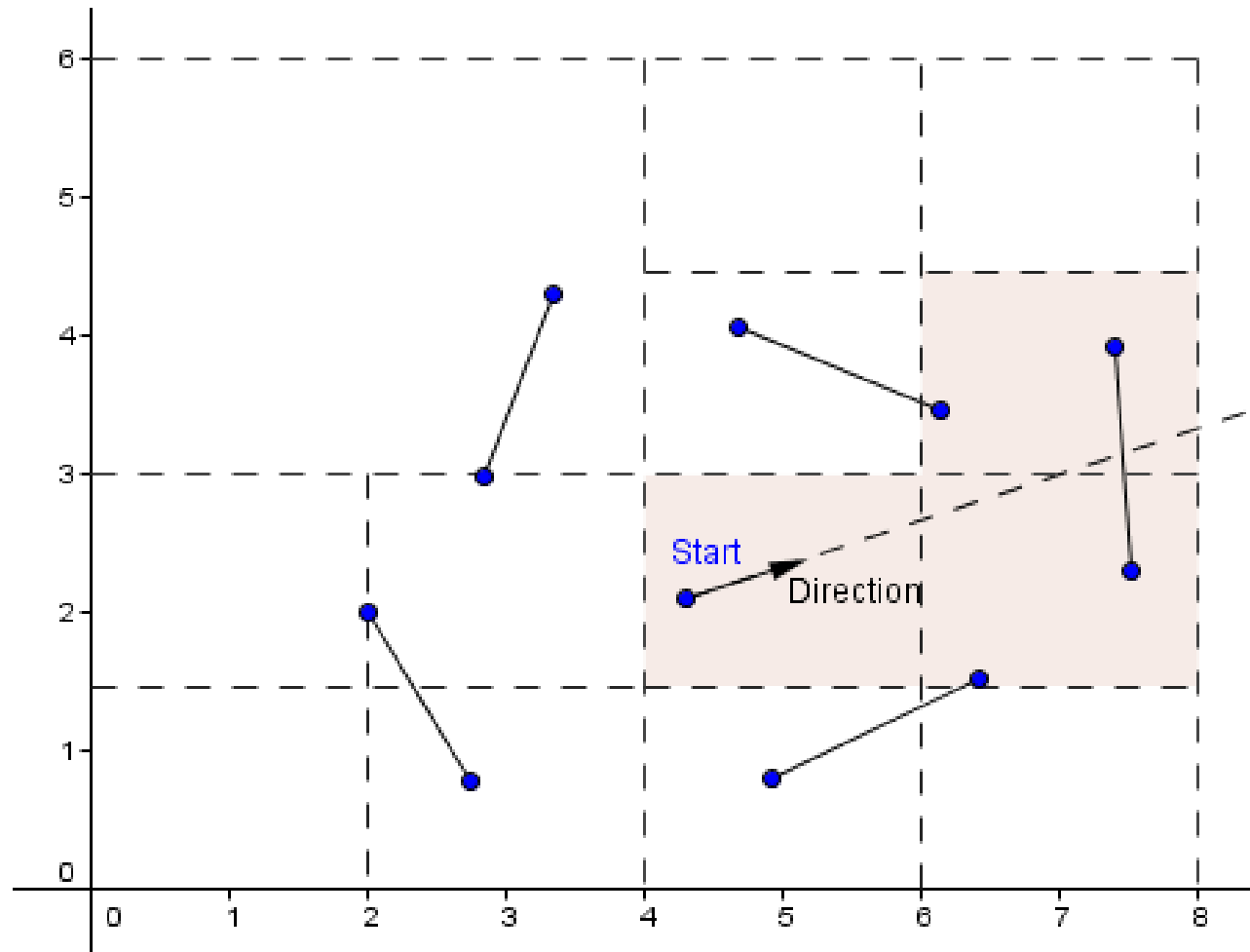
- We can limit the number of grid cells to check, by accounting for the ray's direction.





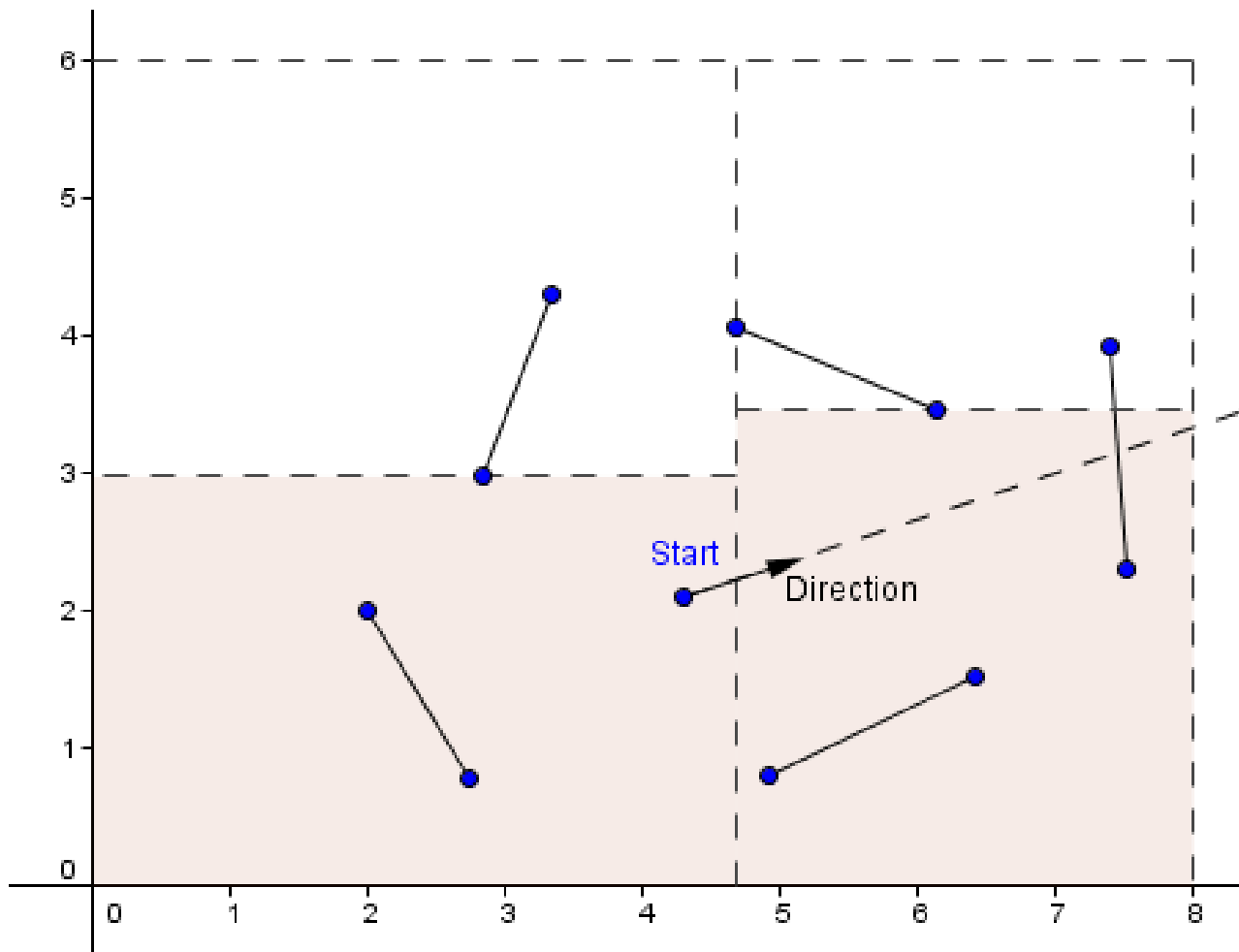
# Second Idea: Quadtree / Octree

- Make the cells divide, if there are more objects inside them. Start with one cell for the entire scene.



# Third Idea: K-D Tree

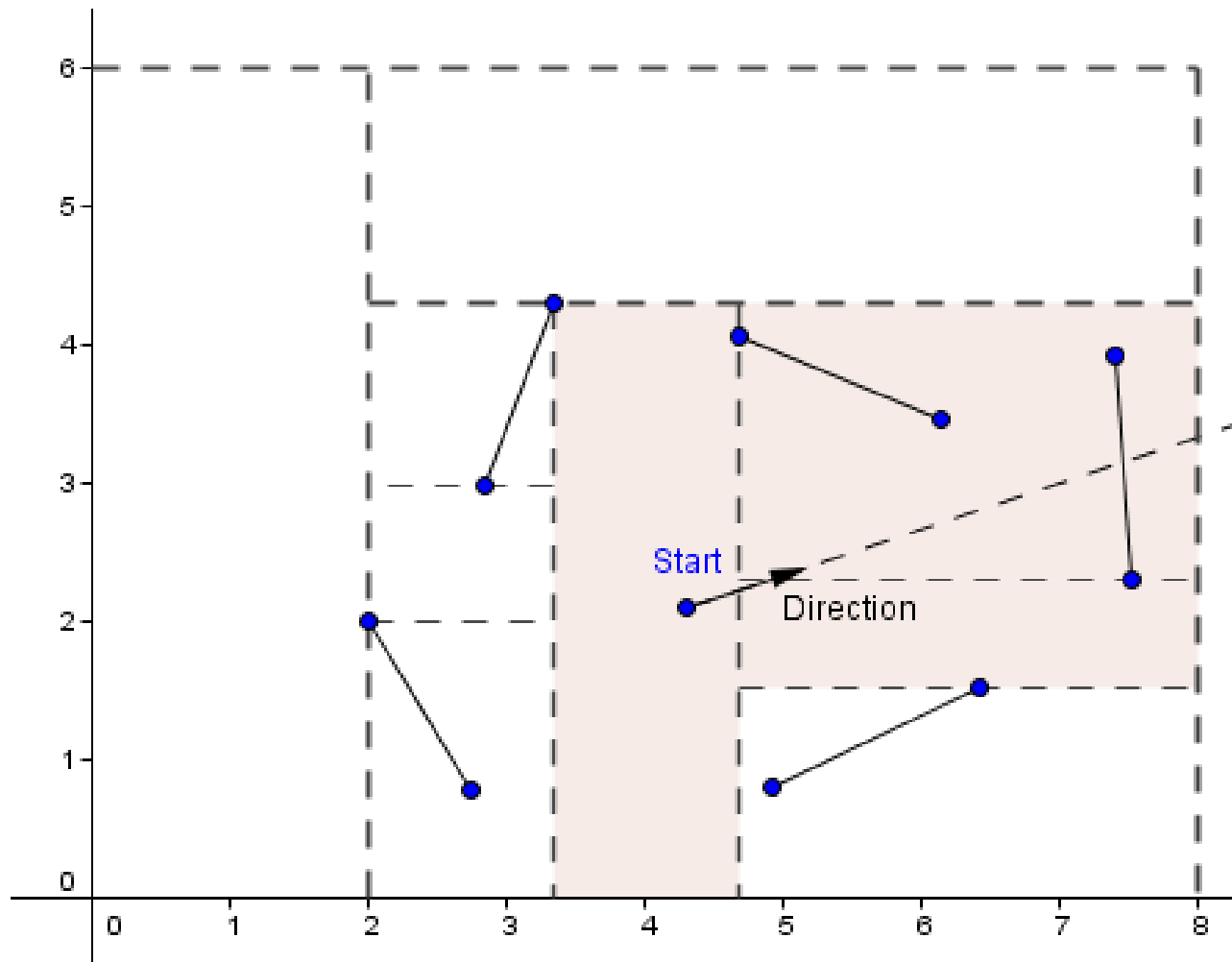
- Split according to the geometry. Traditionally by the median value.



No node will be empty.

# Third Idea: K-D Tree

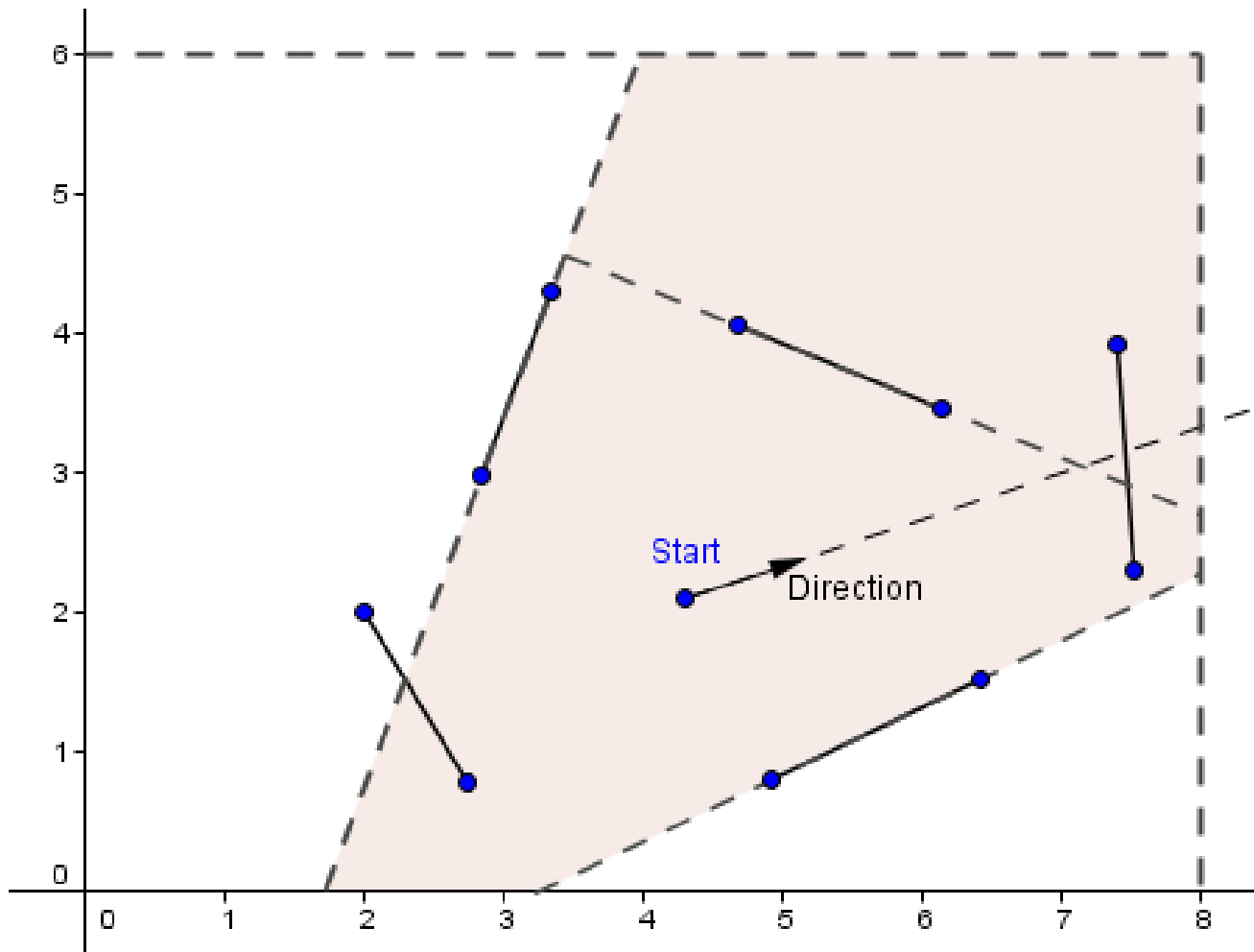
- Split with a rule to maximize the occurrence of empty nodes.



Why is this good?

# Fourth Idea: BSP Tree

- Binary Space Partitioning divides the space with existing polygons.

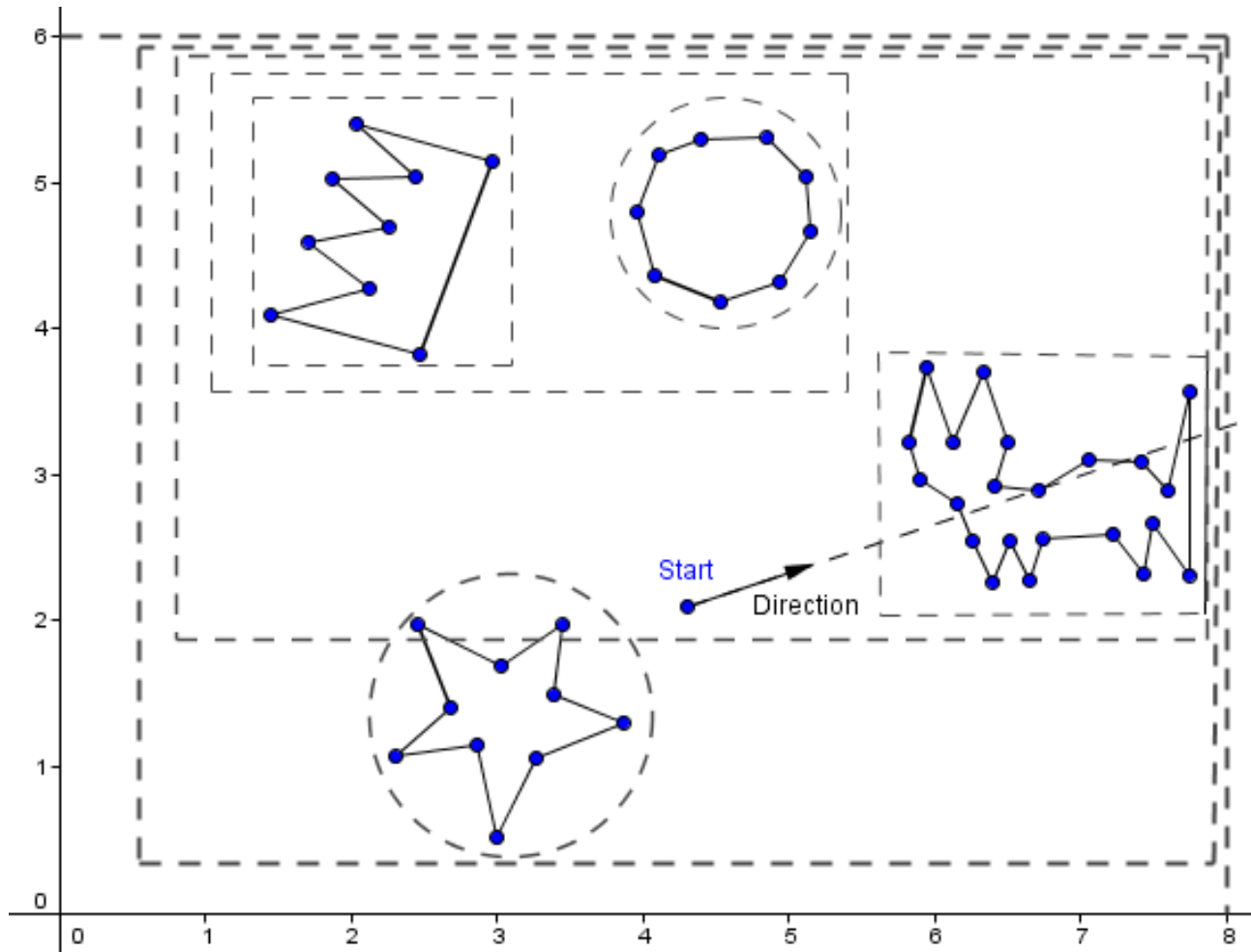


Not that useful for ray tracing.

Works well for geometry ordering (front to back).

# Fifth Idea: BVH

- Bounding Volume Hierarchy – create a tree of bounding polygons around objects.



Bounding objects  
also useful for  
collision detection.

Axis-aligned  
bounding boxes.

Bounding spheres.

# Space Partitioning

- Possible to combine different methods.
- Create structures, based on your own rules.
- Some better for dynamic, some for static scene.
- *Ray Tracing Acceleration Data Structures:*  
[http://www.cse.iitb.ac.in/~paragc/teaching/2009/cs475/notes/accelerating\\_raytracing\\_sumair.pdf](http://www.cse.iitb.ac.in/~paragc/teaching/2009/cs475/notes/accelerating_raytracing_sumair.pdf)
- *Octree vs BVH:*  
<http://thomasdiewald.com/blog/?p=1488>



What did you learn today?

What more would you like to know?

Next time

Global Illumination