



Eurographics 2013

May 6-10, Girona (Spain)

Lazy Visibility Evaluation for Exact Soft Shadows

Frédéric Mora, Lilian Aveneau,
Oana Apostu, Djamchid Ghazanfarpour



Soft shadows

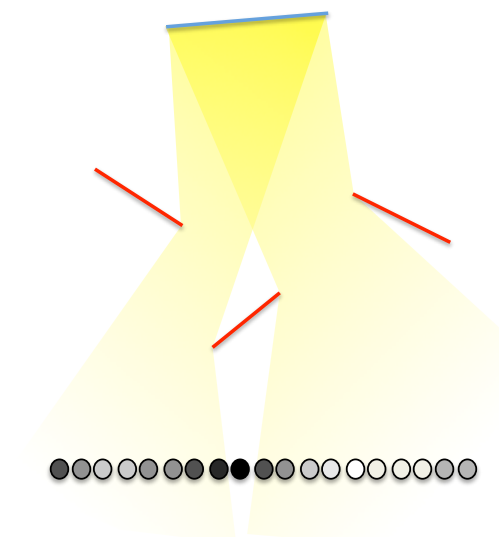
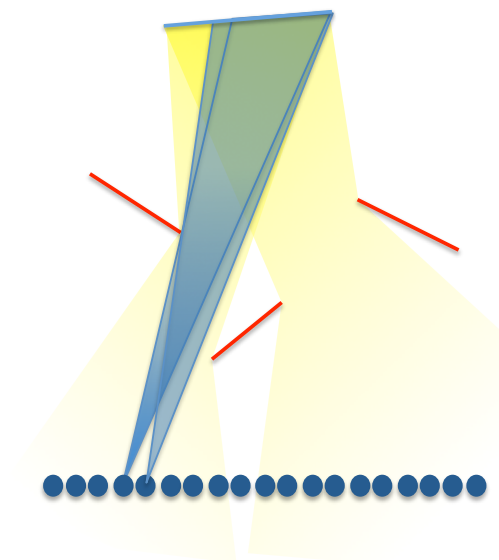
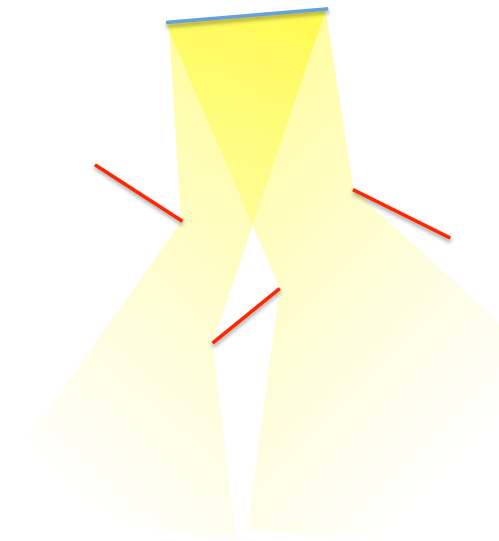
- underlying problem: area light source visibility
- visibility accuracy / soft shadows quality

A common solution: sampling

- noise sensitive
- can become expensive

Introduction > our approach, main ideas

- coherent visibility representation from/to an area light source
- contains all the possible light views
- extract light visibility to compute analytic direct illumination



Exact from-polygon visibility

- a 4D problem, complex
- 2002 Nirenstein / Bittner (target scenario: exact PVS)
- CSG computations (subtractions of 5D convex polyhedrons)

Common drawbacks

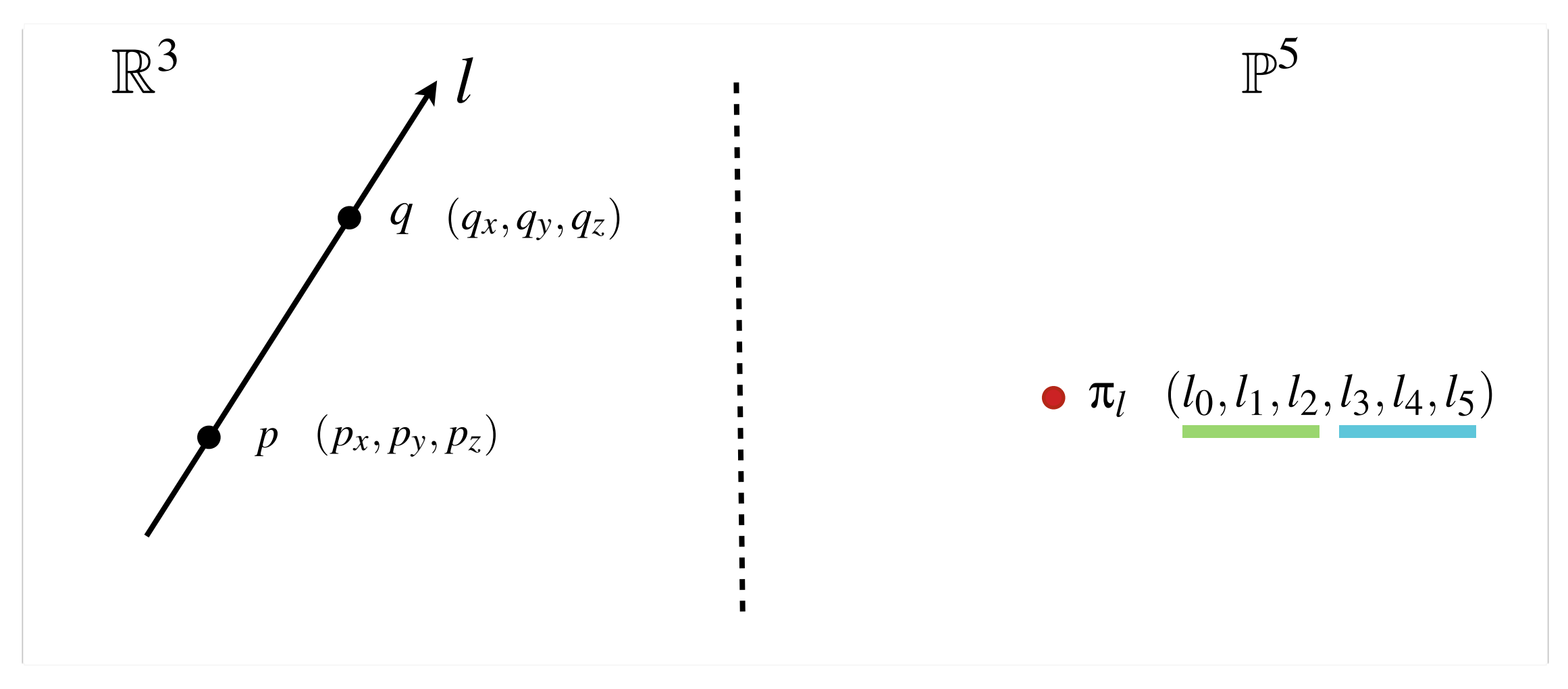
- very expensive (preprocess)
- robustness issues
- implementation is complicated

Analytic from-polygon visibility

- starts over from the theory
- no 5D CSG computations \Rightarrow easy, robust, efficient
- no preprocess

- **Geometric basis**
- **Visibility algorithm**
- **Results**

Geometrical basis > plucker coordinates of an oriented line



\vec{pq}



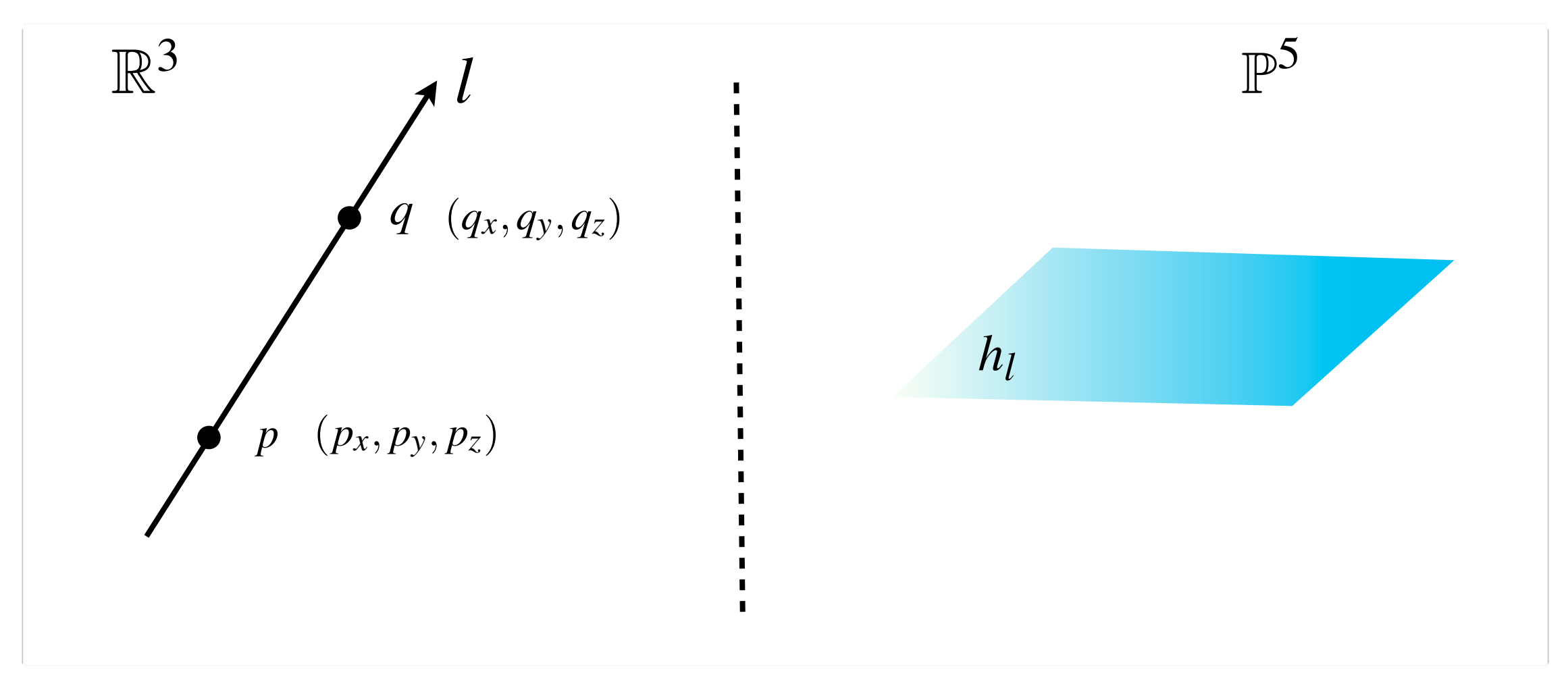
$$\begin{cases} l_0 = q_x - p_x \\ l_1 = q_y - p_y \\ l_2 = q_z - p_z \end{cases}$$

$$\begin{cases} l_3 = q_z p_y - q_y p_z \\ l_4 = q_x p_z - q_z p_x \\ l_5 = q_y p_x - q_x p_y \end{cases}$$

$\vec{op} \times \vec{oq}$



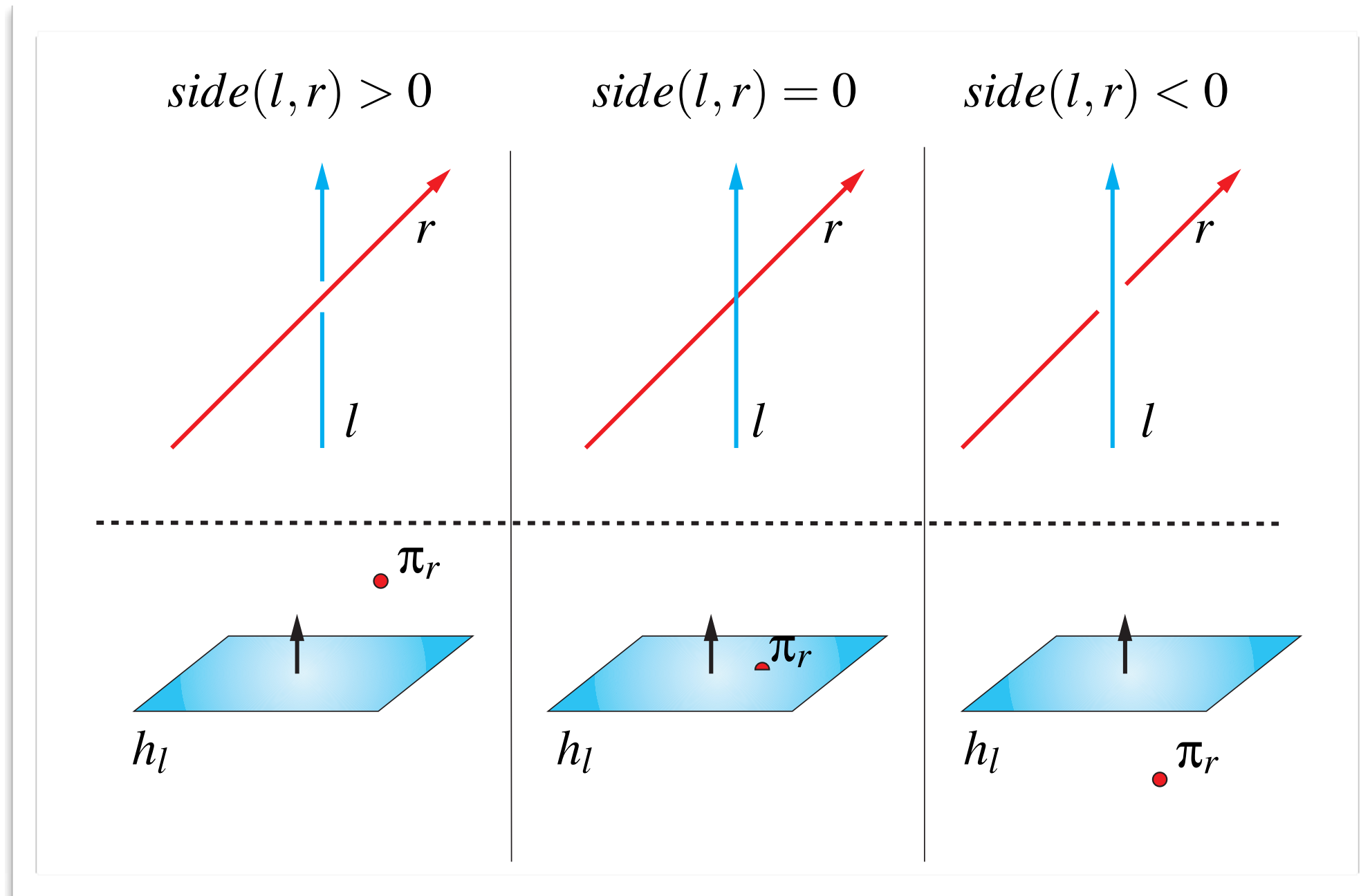
Geometrical basis > point - hyperplane duality



● $\pi_l (l_0, l_1, l_2, l_3, l_4, l_5)$

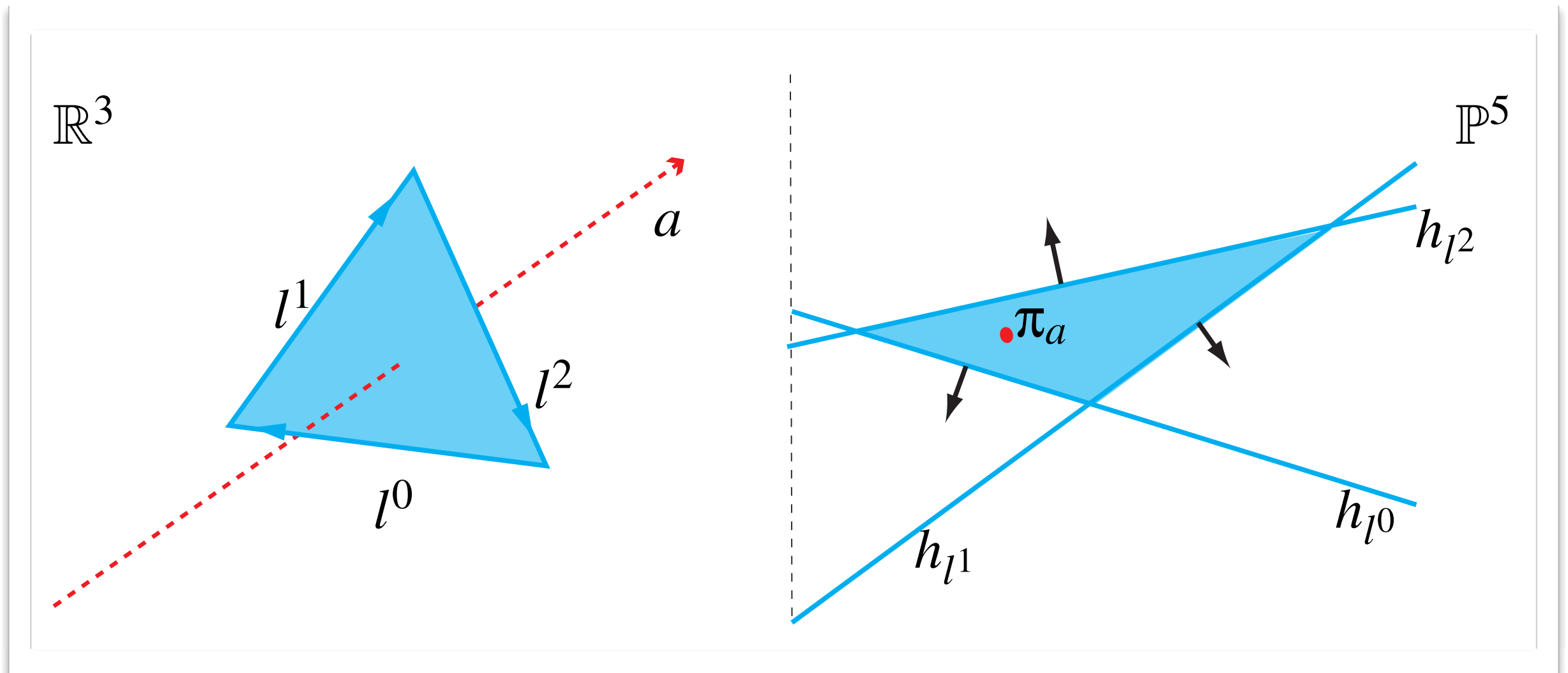
▮ $h_l(x) = l_3x_0 + l_4x_1 + l_5x_2 + l_0x_3 + l_1x_4 + l_2x_5 = 0$

Geometrical basis > relative orientation of lines



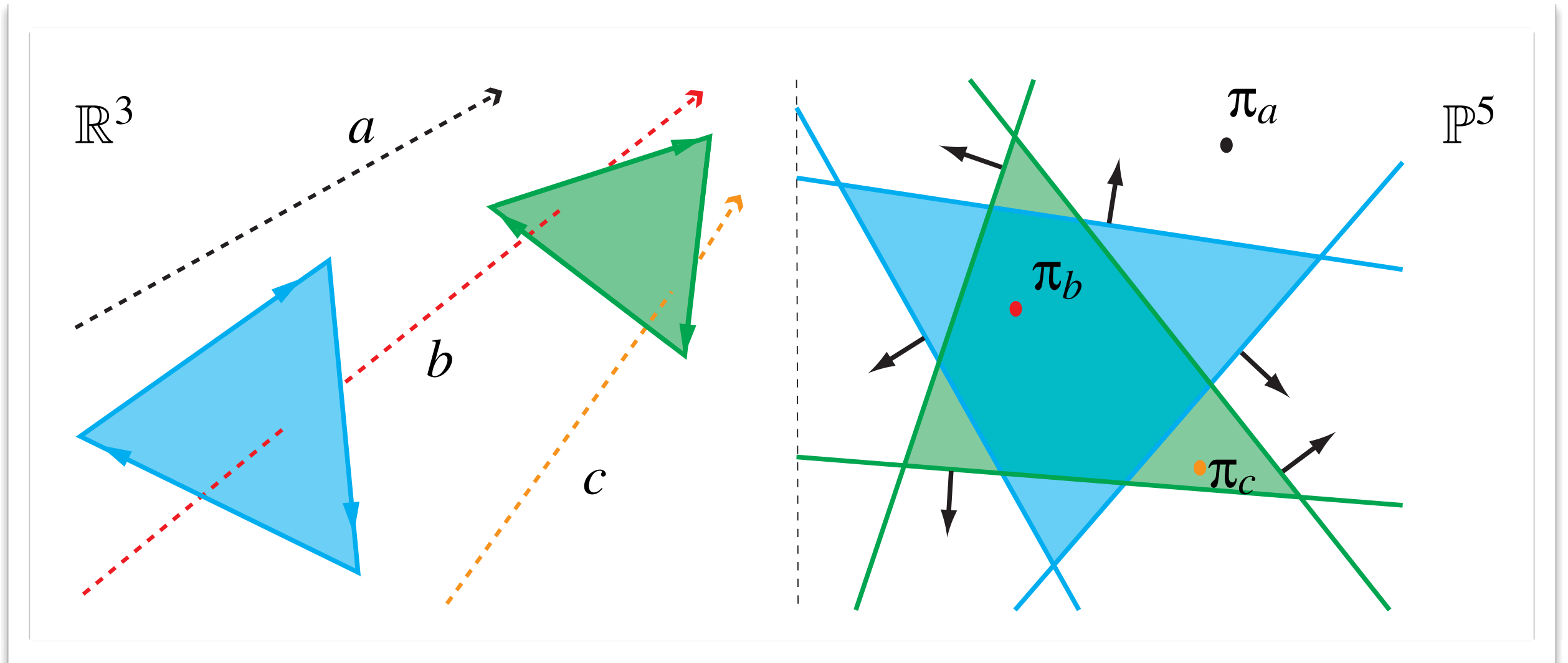
$$\begin{aligned} side(l, r) &= l_3 r_0 + l_4 r_1 + l_5 r_2 + l_0 r_3 + l_1 r_4 + l_2 r_5 \\ &= h_l(\pi_r) \end{aligned}$$

Geometrical basis > line-triangle intersection test



- lines stabbing a triangle...
- ... have their plucker point inside the convex polyhedron...
- ... whose hyperplanes are the lines spanning the triangle edges

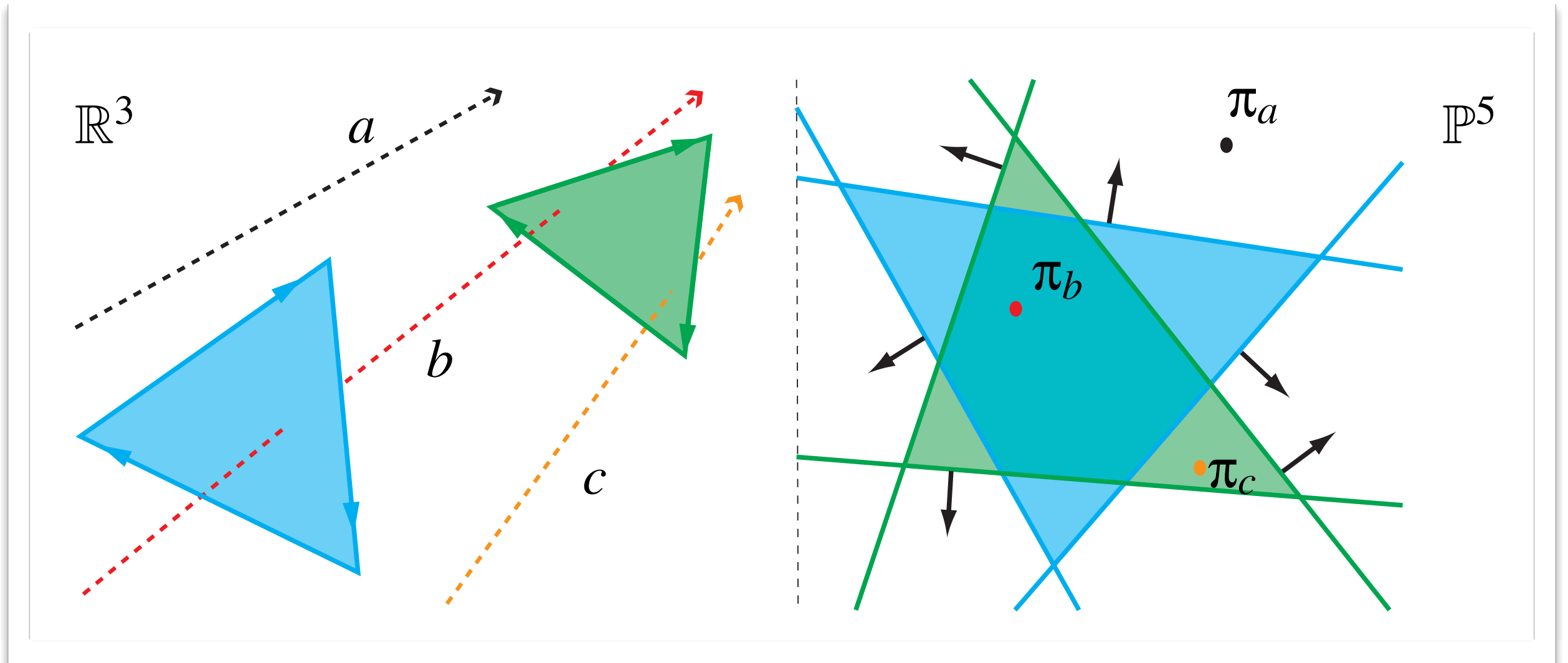
Geometrical basis > lines stabbing triangles



Pellegrini

- lines spanning triangle edges \Rightarrow arrangement of hyperplanes
- lines in a same cell intersect the same subset of triangles

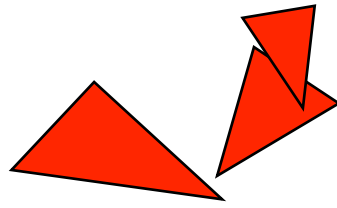
Geometrical basis > lines stabbing triangles



- equivalence relation on lines
- lines are grouped according to the geometry they intersect

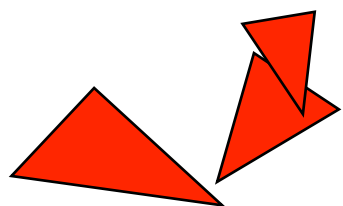
Geometrical basis \triangleright in short...

- take some triangles

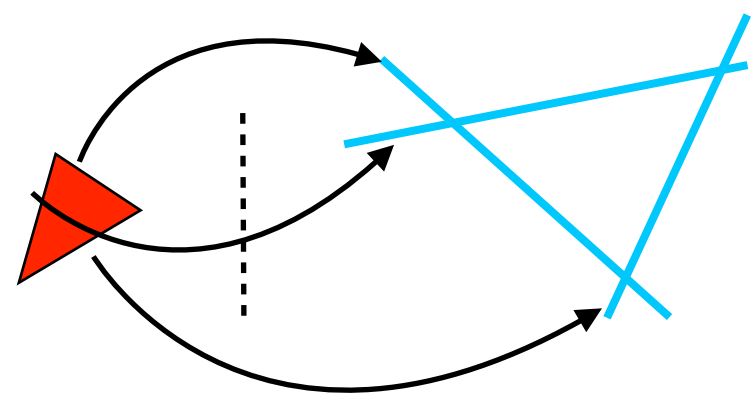


Geometrical basis > in short...

- take some triangles

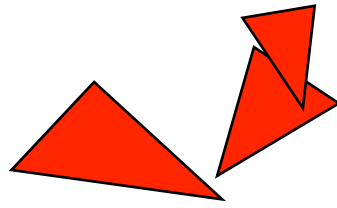


- map the lines spanning their edges to plucker hyperplanes

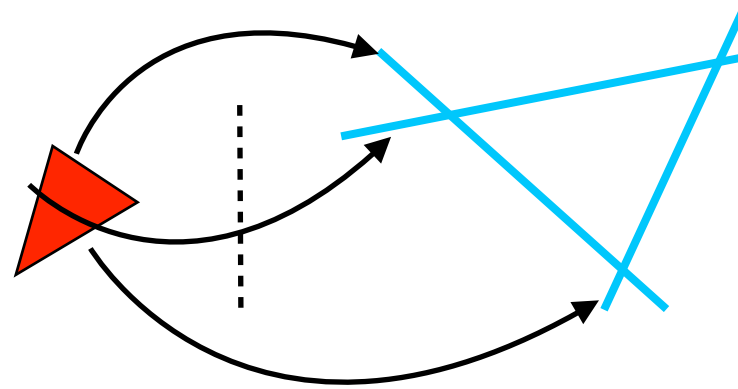


Geometrical basis \triangleright in short...

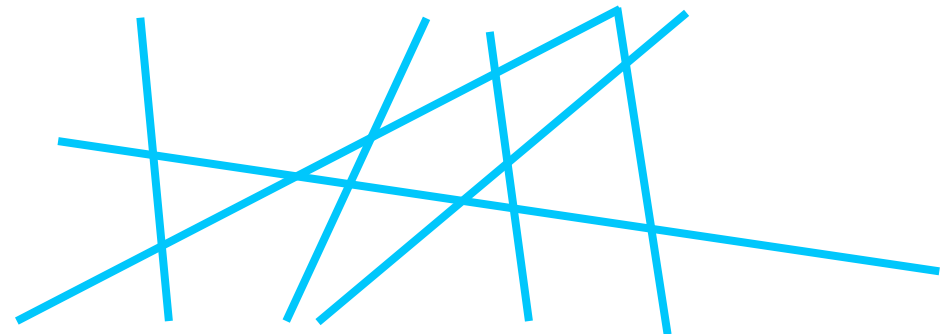
- take some triangles



- map the lines spanning their edges to plucker hyperplanes

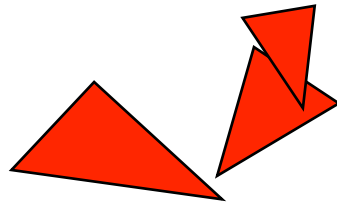


- this defines an arrangement of hyperplanes in Plucker space

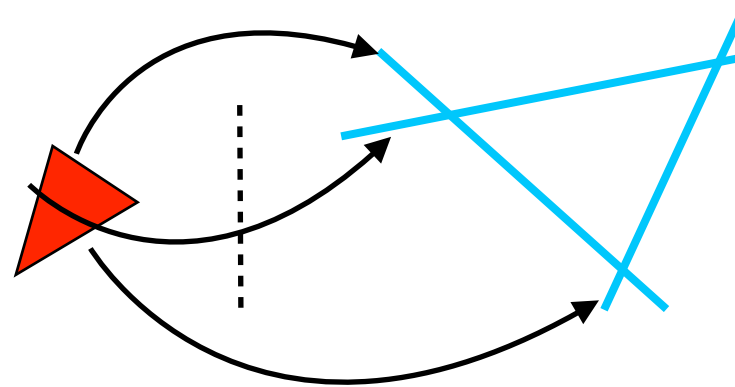


Geometrical basis \succ in short...

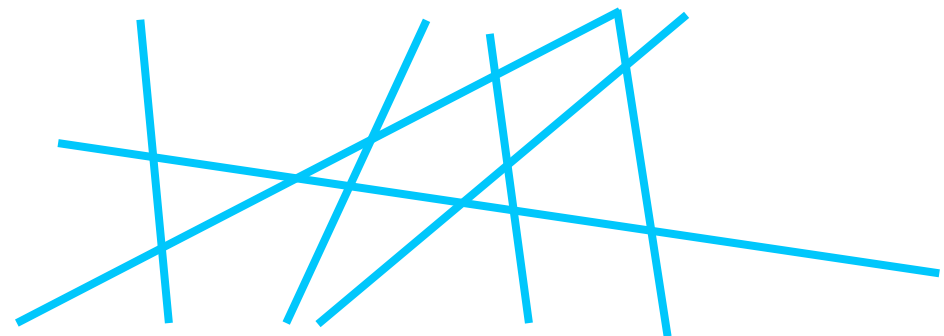
- take some triangles



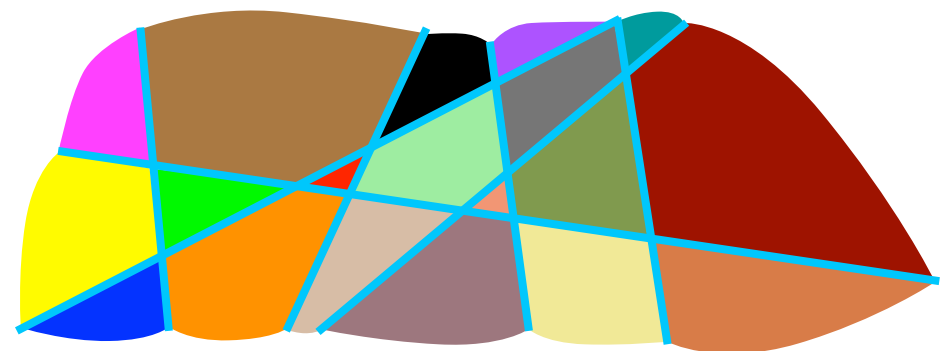
- map the lines spanning their edges to plucker hyperplanes



- this defines an arrangement of hyperplanes in Plucker space



- each cell contains lines that hit/miss the same subset of triangles

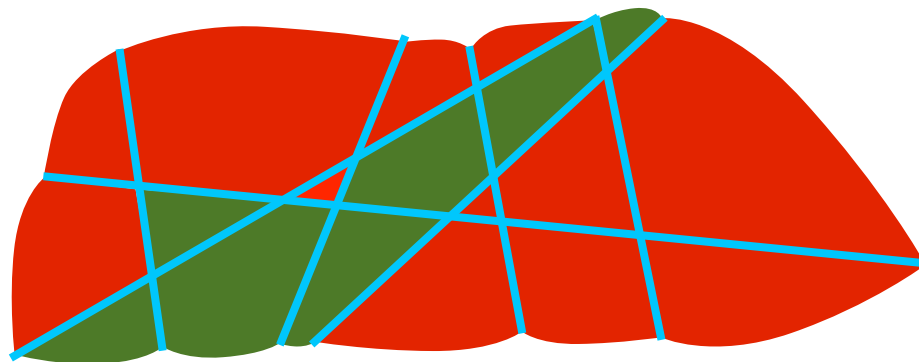


Geometrical basis > in short...

Equivalence relation on lines

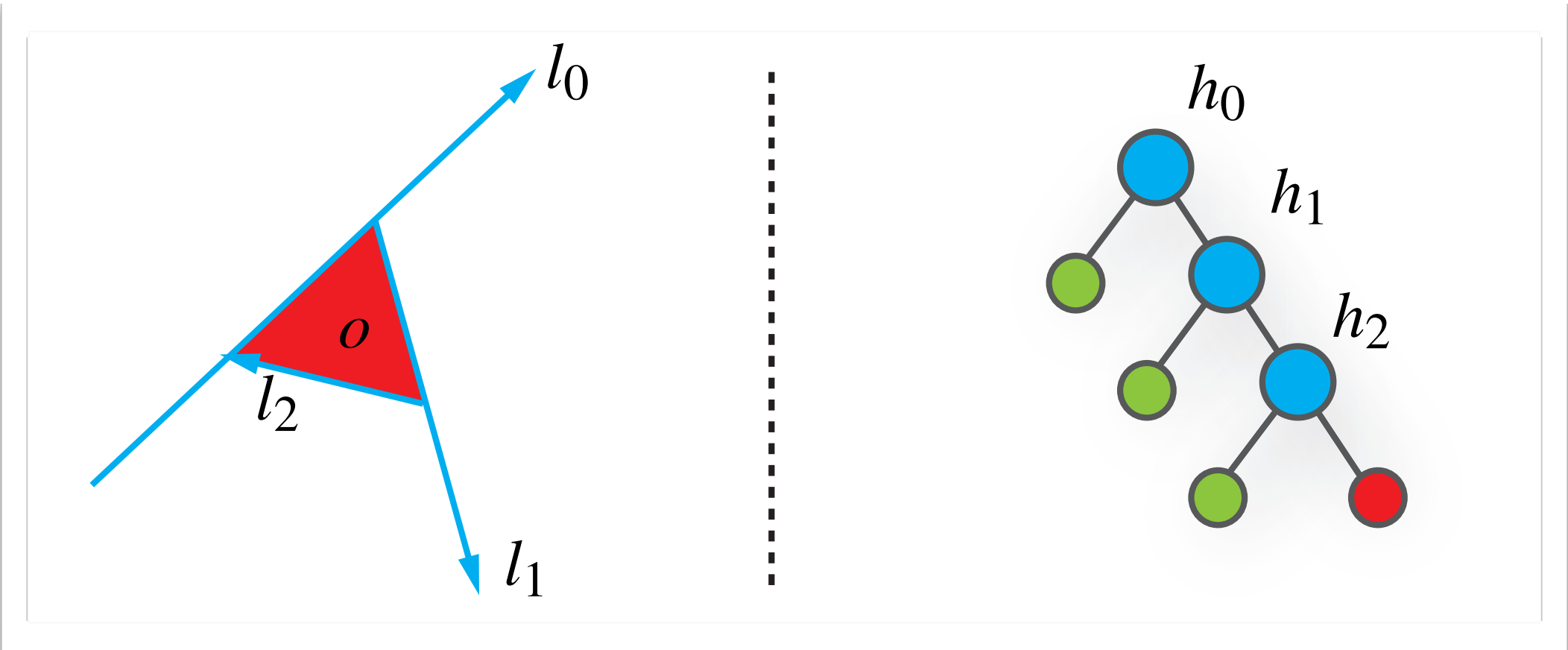
- one cell \Rightarrow one equivalence class
- equivalent lines are coherent paths through the triangles




We want to compute equivalence classes representing coherent paths from/to an area light source through its occluders



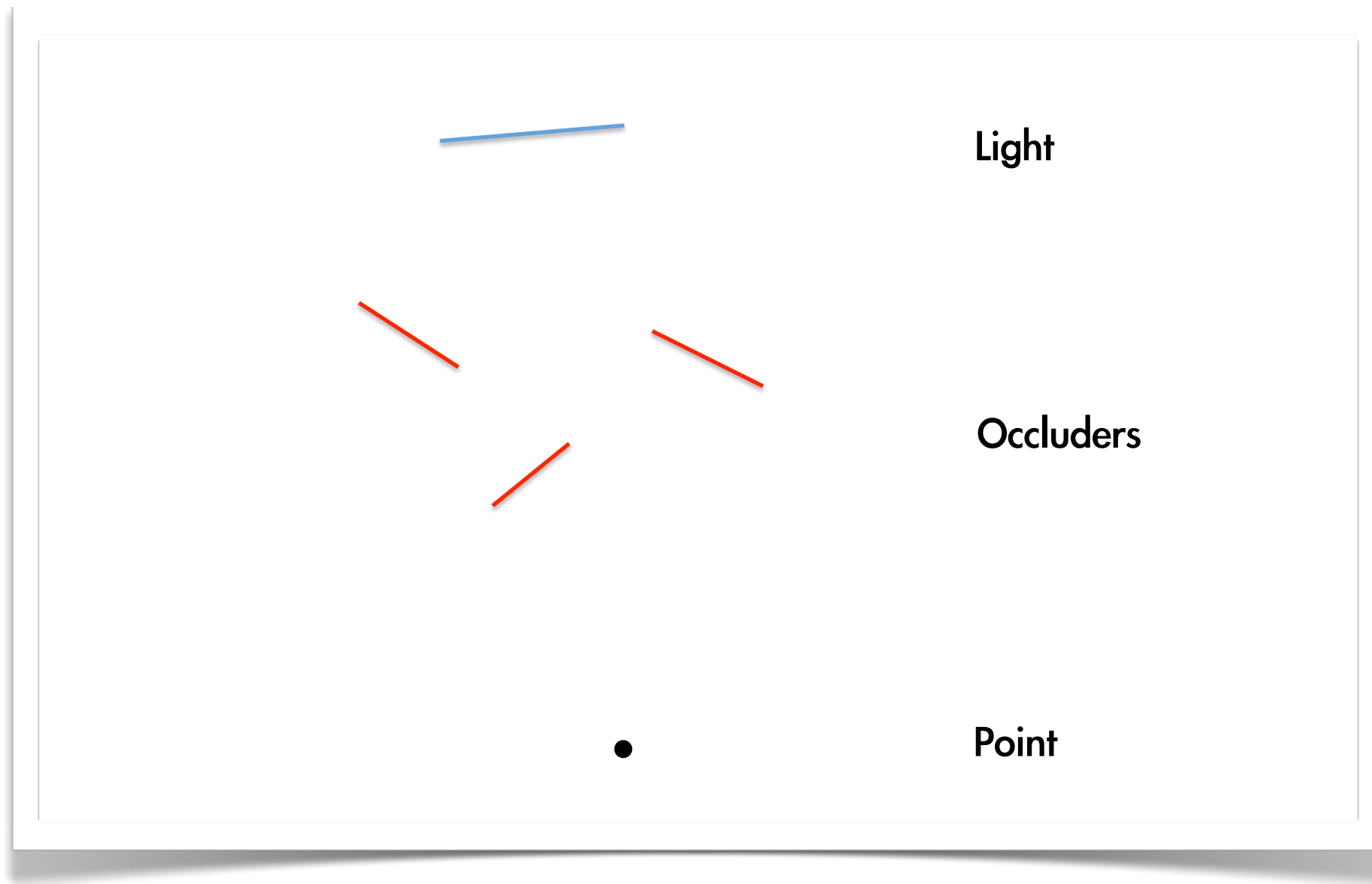
- **Geometric basis**
- **Visibility algorithm**
- **Results**

Visibility algorithm > BSP representation of an occluder



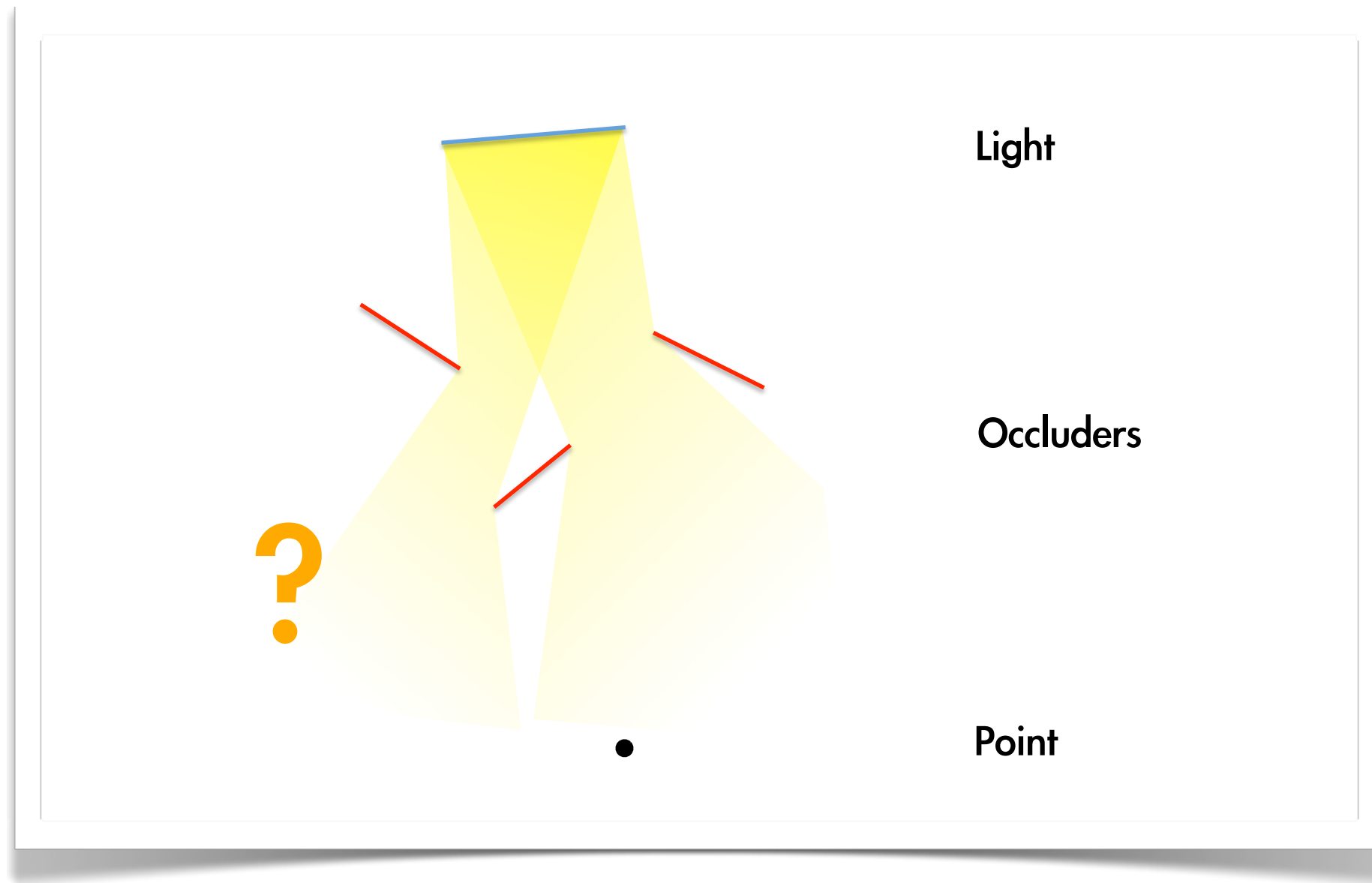
-  hyperplanes (inner nodes)
-  visible classes (leaves)
-  invisible classes (leaves)

Visibility algorithm > principle



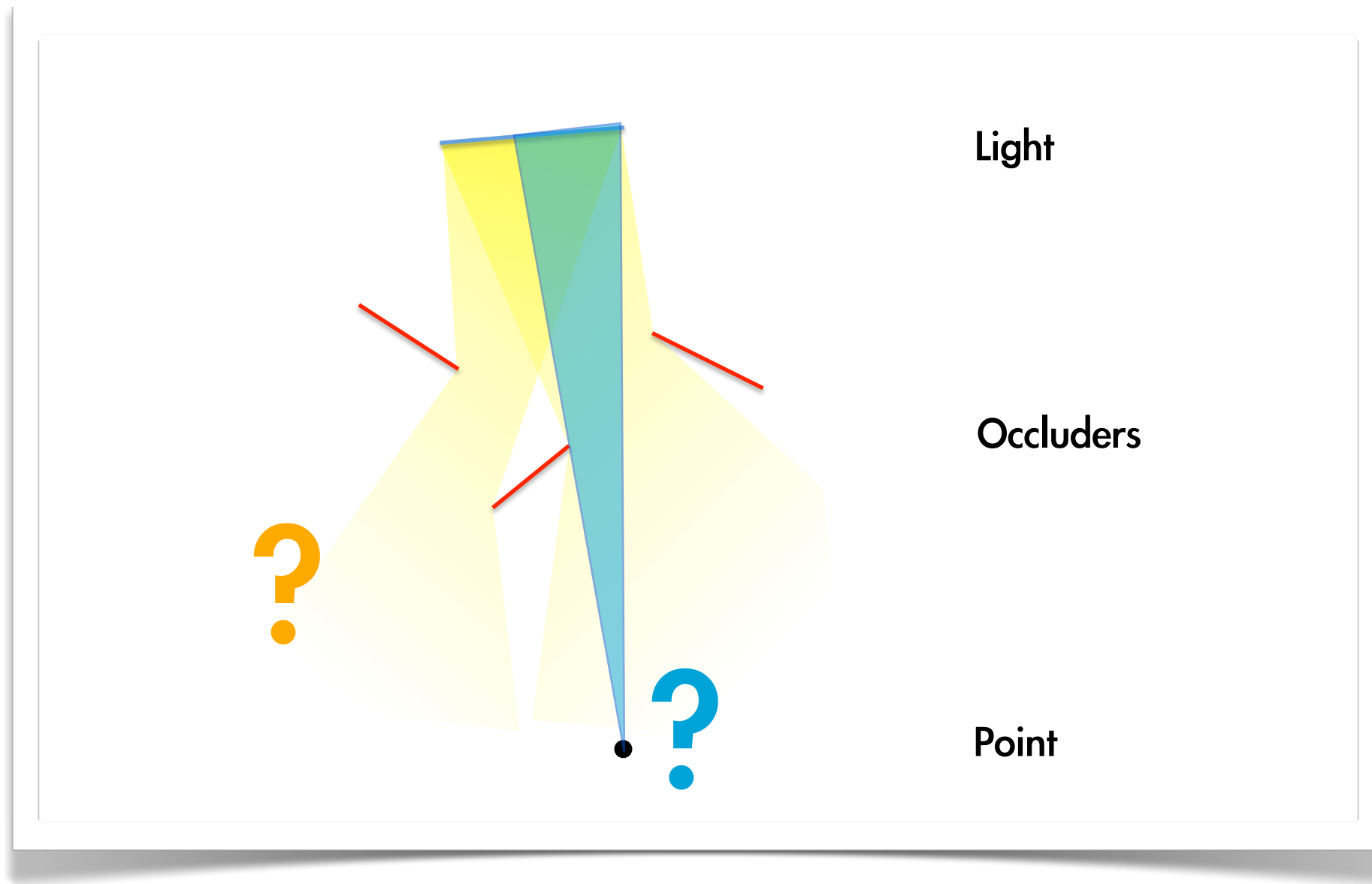
- visibility from the area light source ?
- light visibility from the point ?

Visibility algorithm > principle



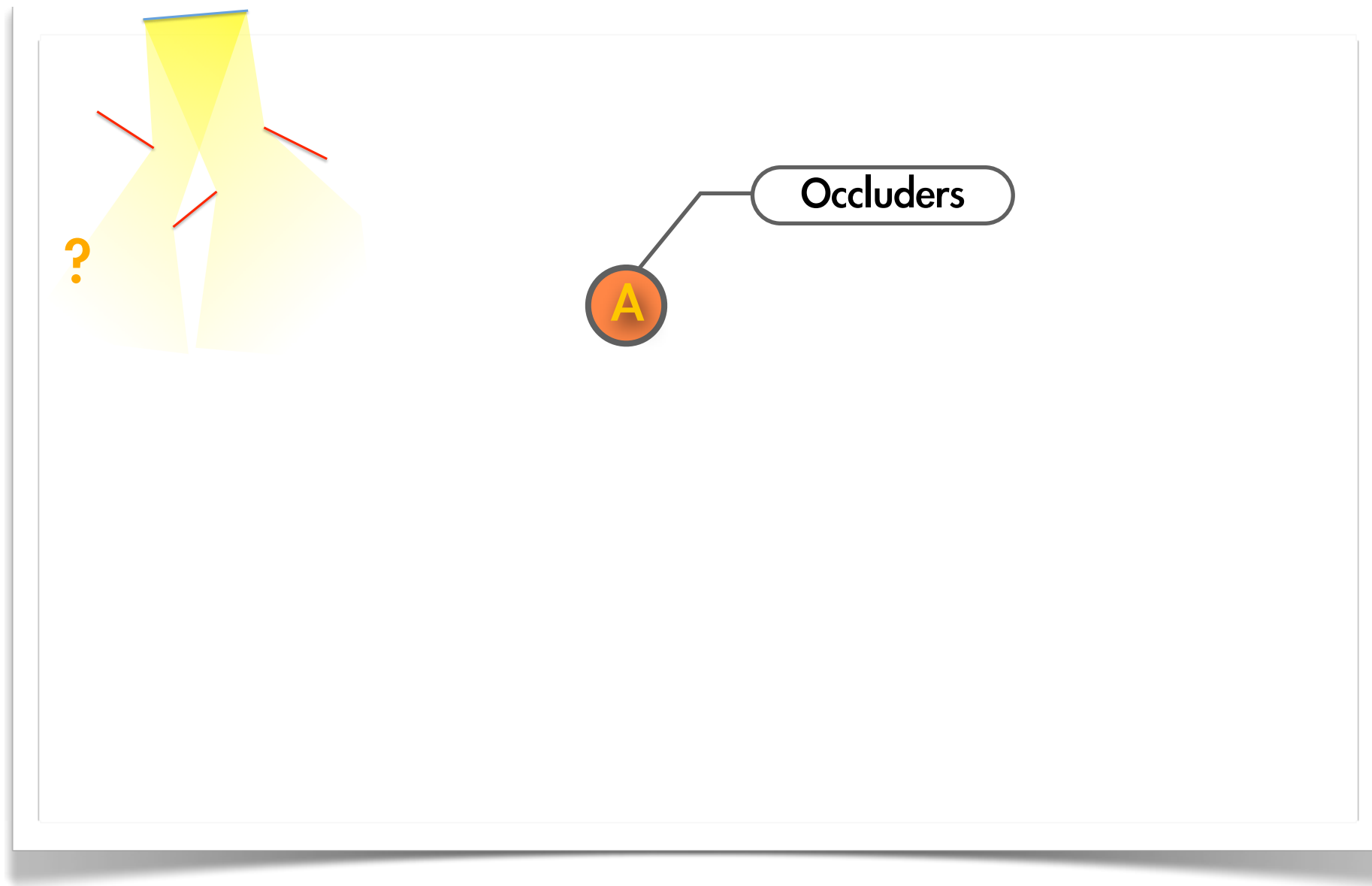
- visibility from the area light source ?
- light visibility from the point ?

Visibility algorithm > principle



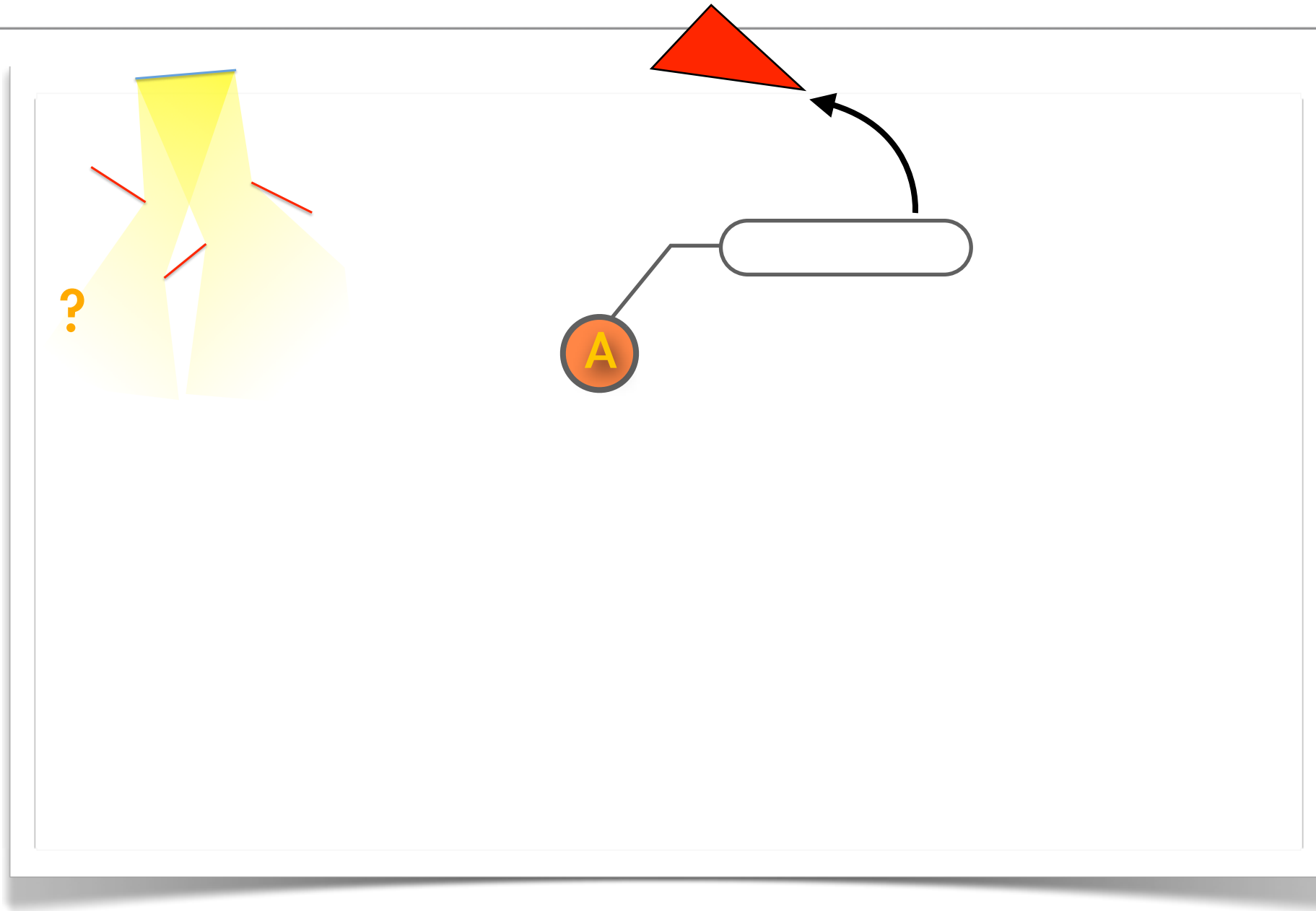
- visibility from the area light source ?
- light visibility from the point ?

Visibility algorithm > principle



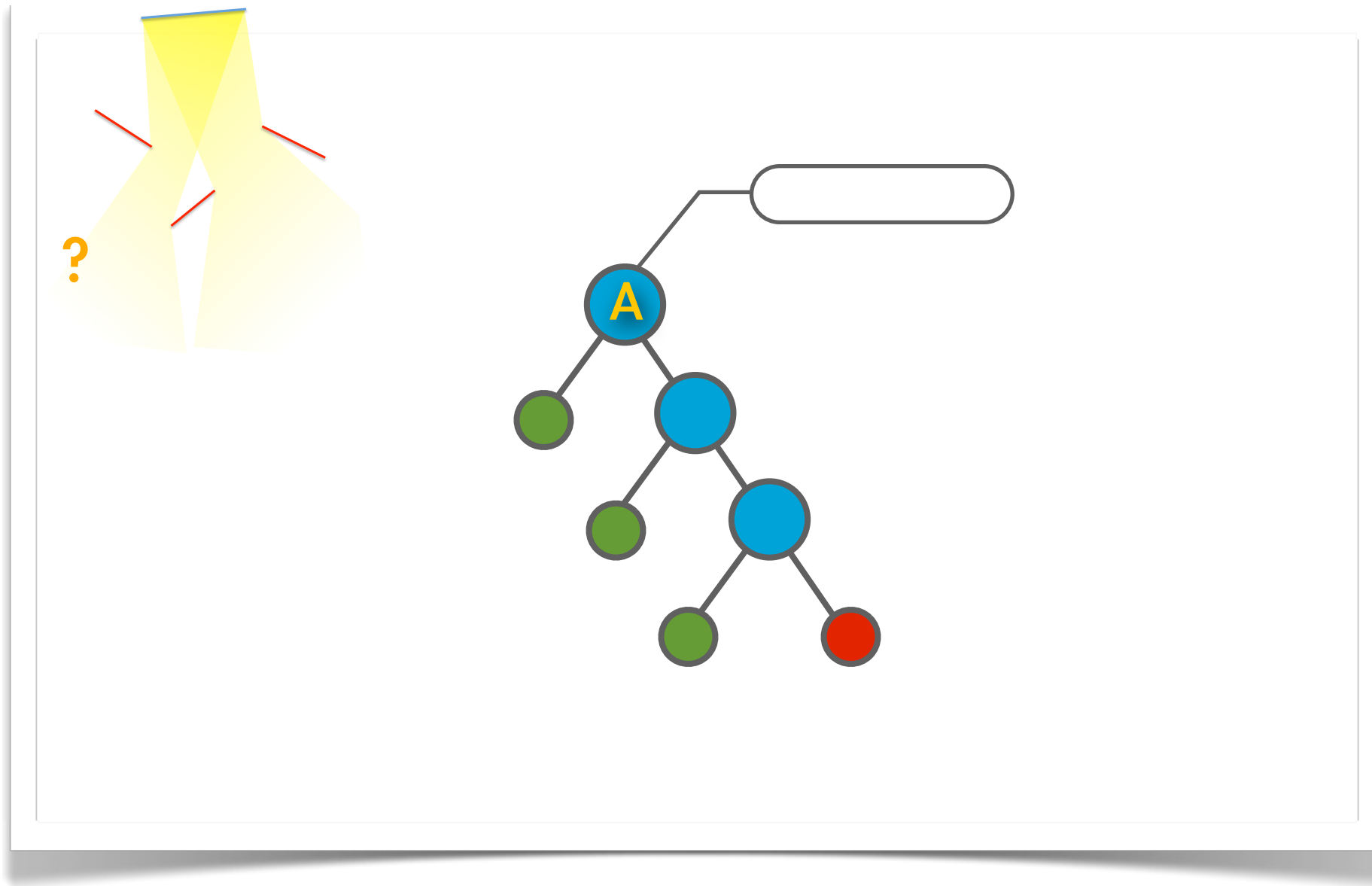
- start with an **undefined** class and all the occluders

Visibility algorithm > principle



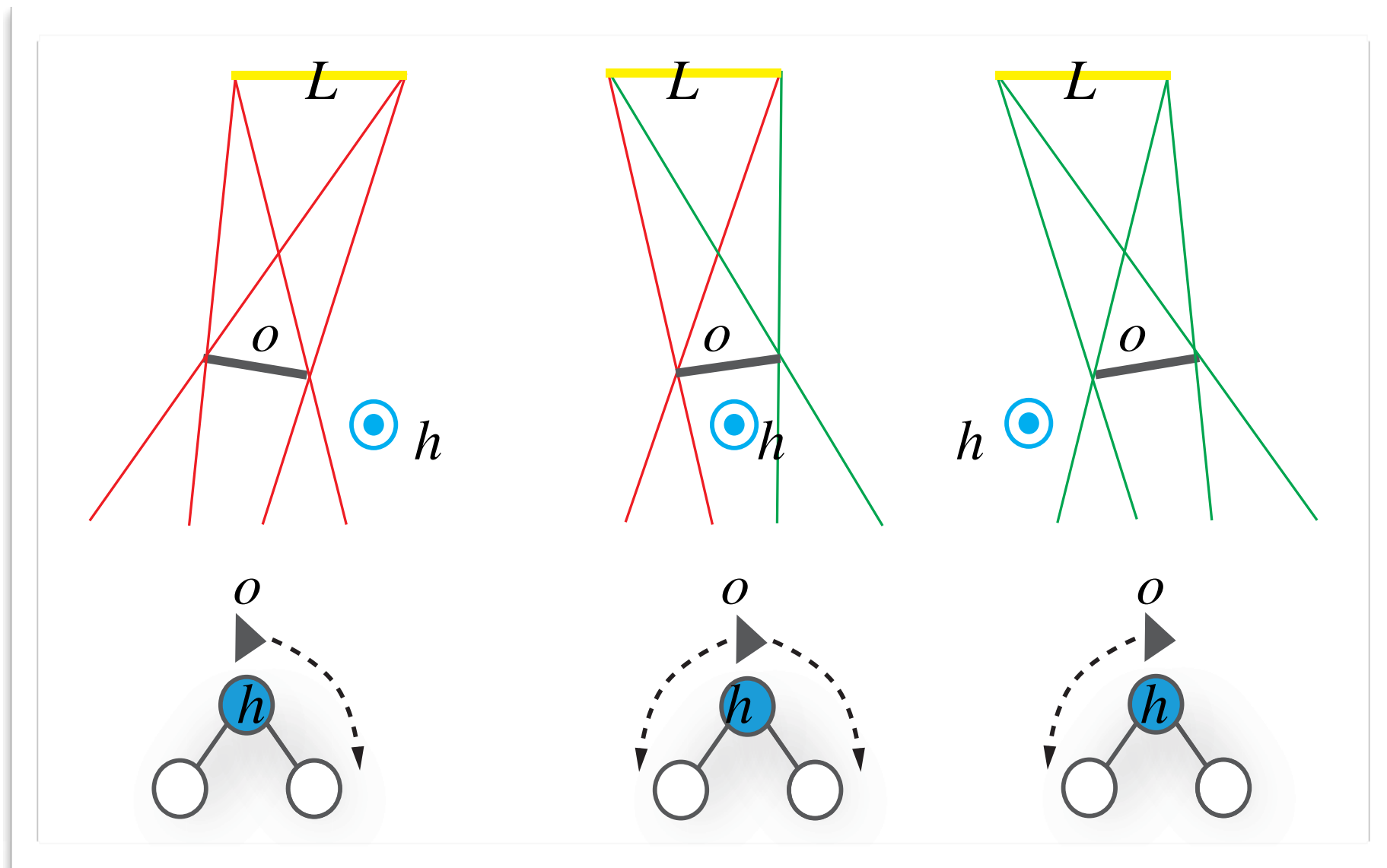
- select a random occluder

Visibility algorithm > principle



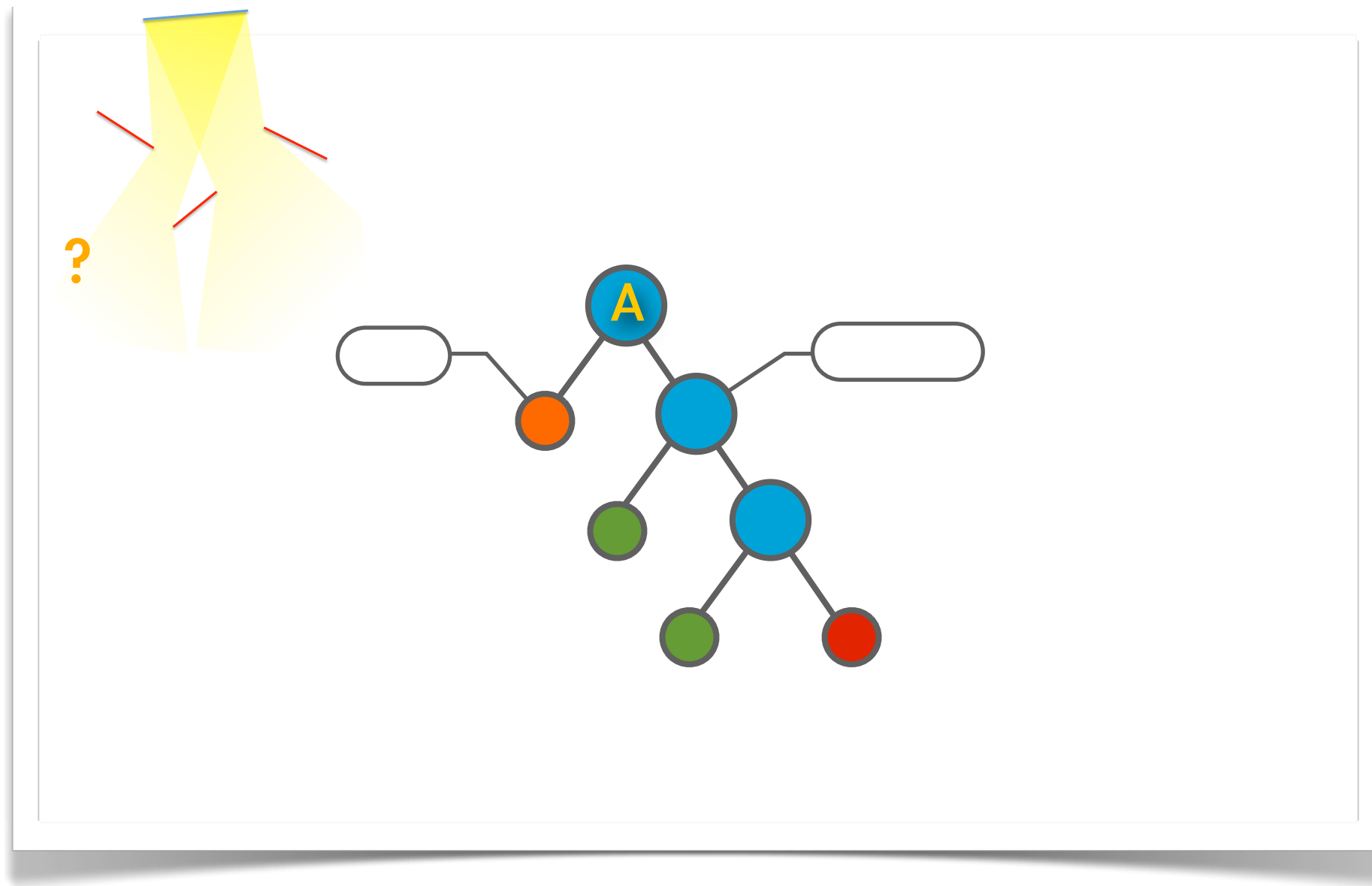
- replace the leaf with the occluder BSP representation
- remaining occluders can affect the visible classes

Visibility algorithm > occluded lines orientation



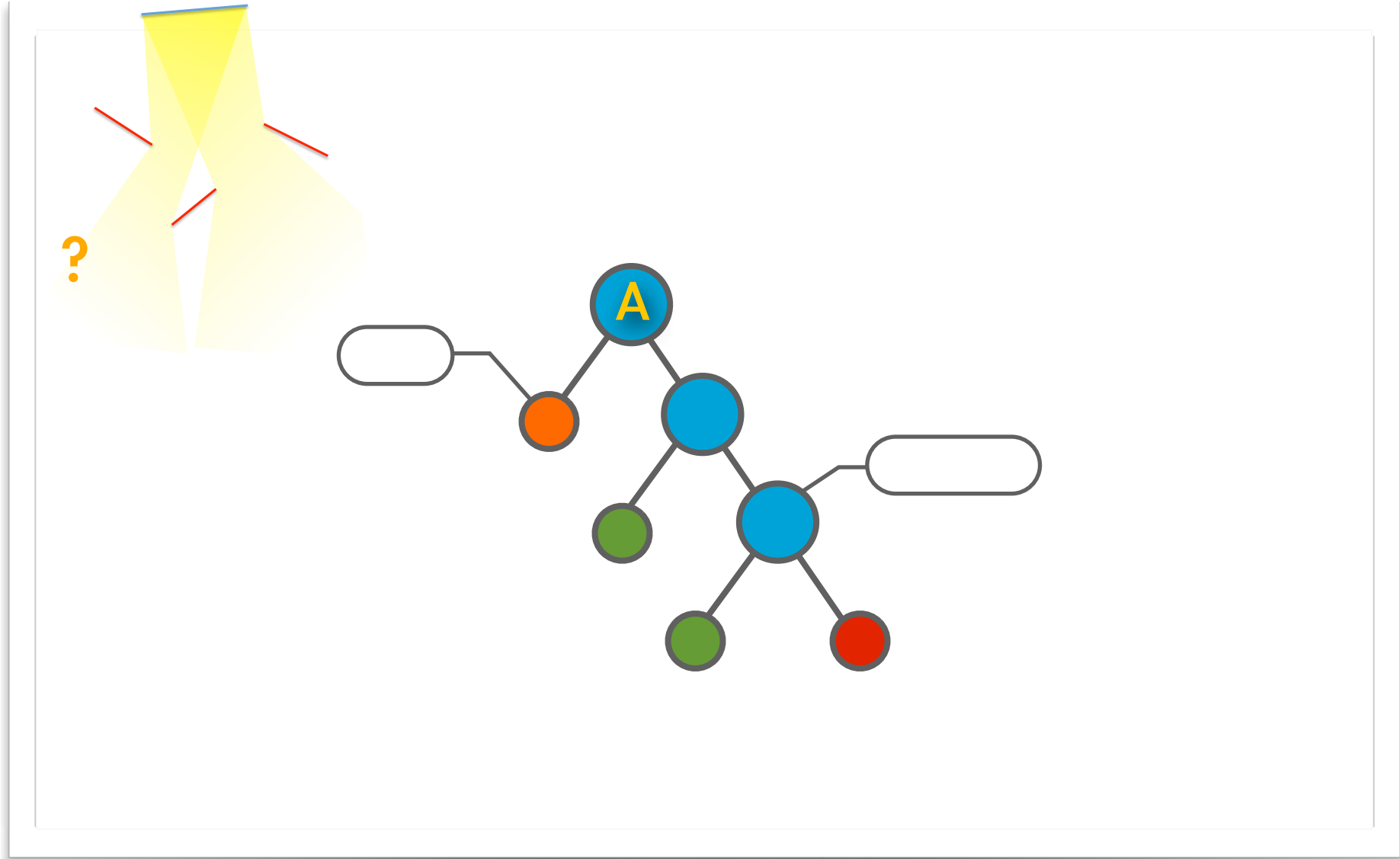
- orientation of the occluded lines with respect to h ?
- computing the vertex-to-vertex lines orientation is sufficient

Visibility algorithm > principle

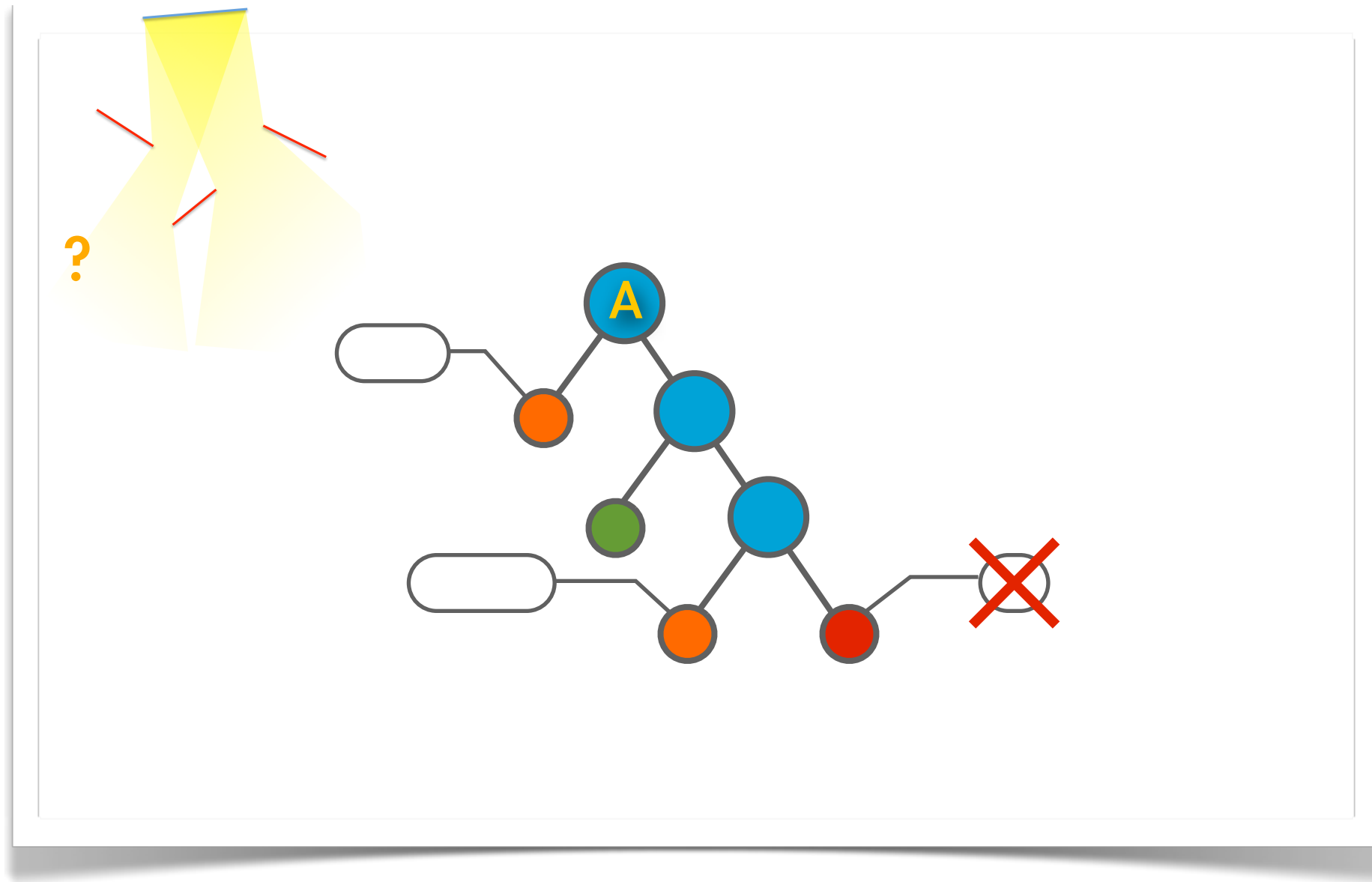


- insert the remaining occluders in the tree
- **visible** classes reached by occluders become **undefined**

Visibility algorithm > principle

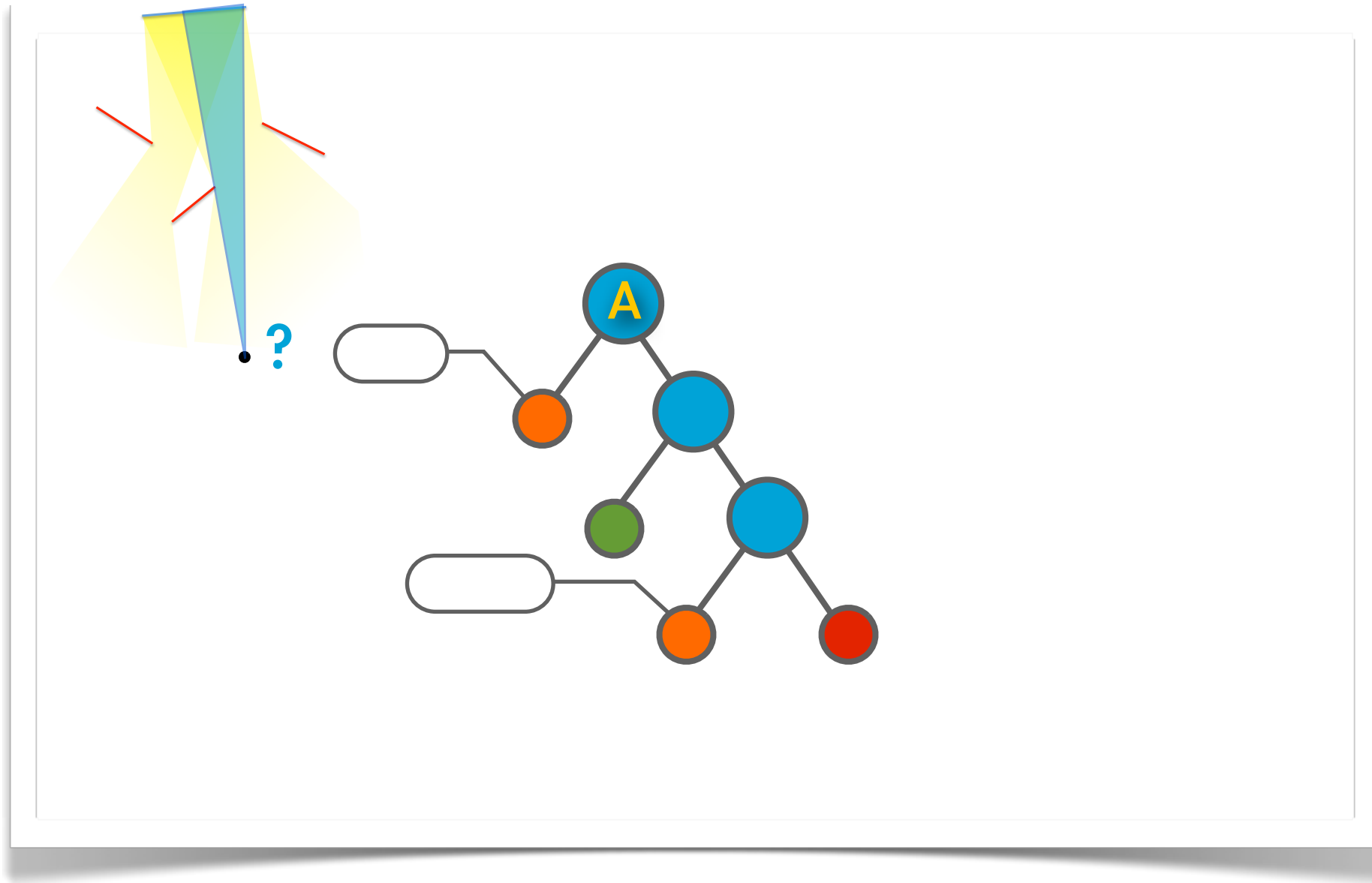


Visibility algorithm > principle



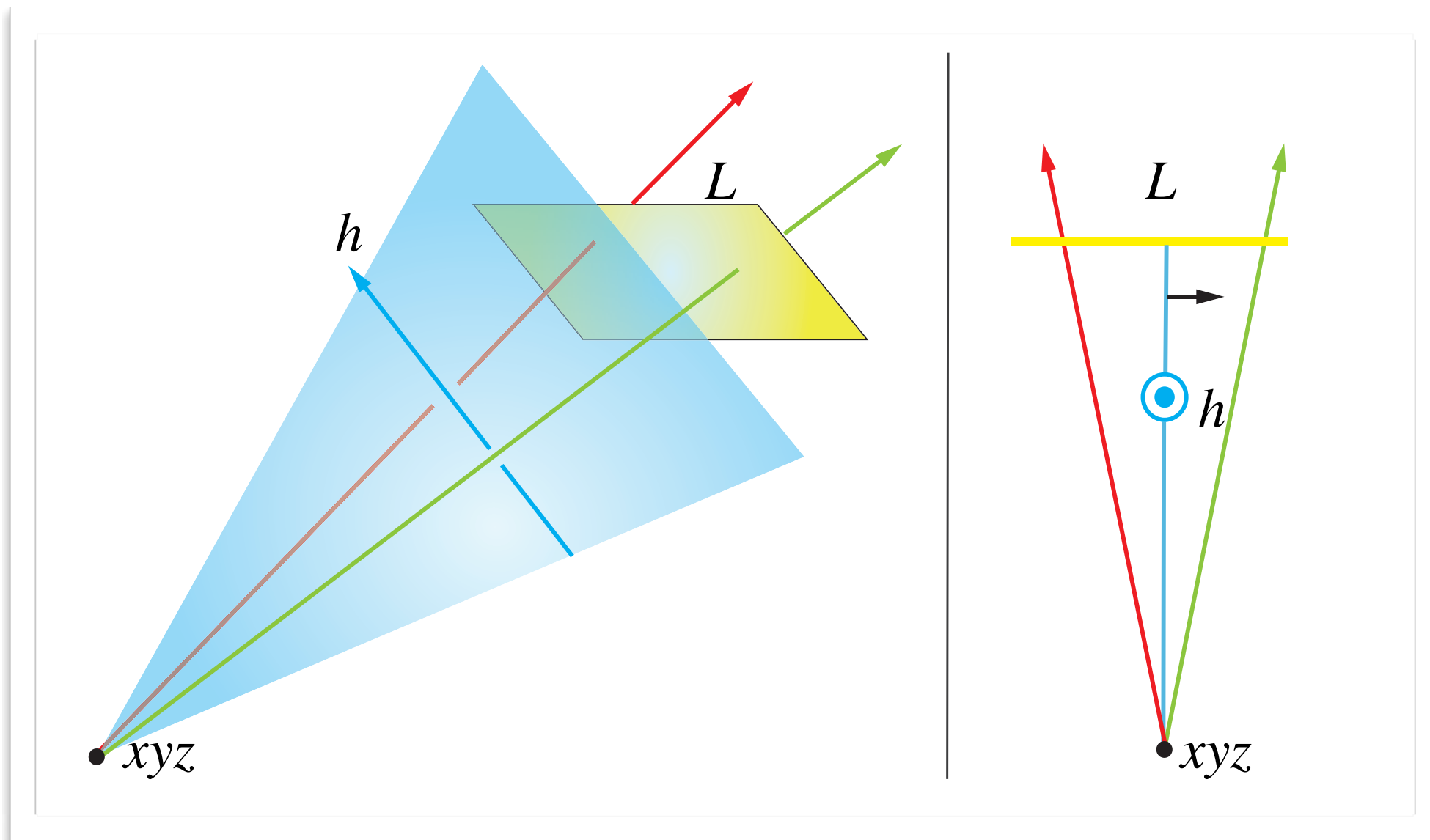
- occluders reaching **invisible** classes are discarded
- first problem is partially solved

Visibility algorithm > principle



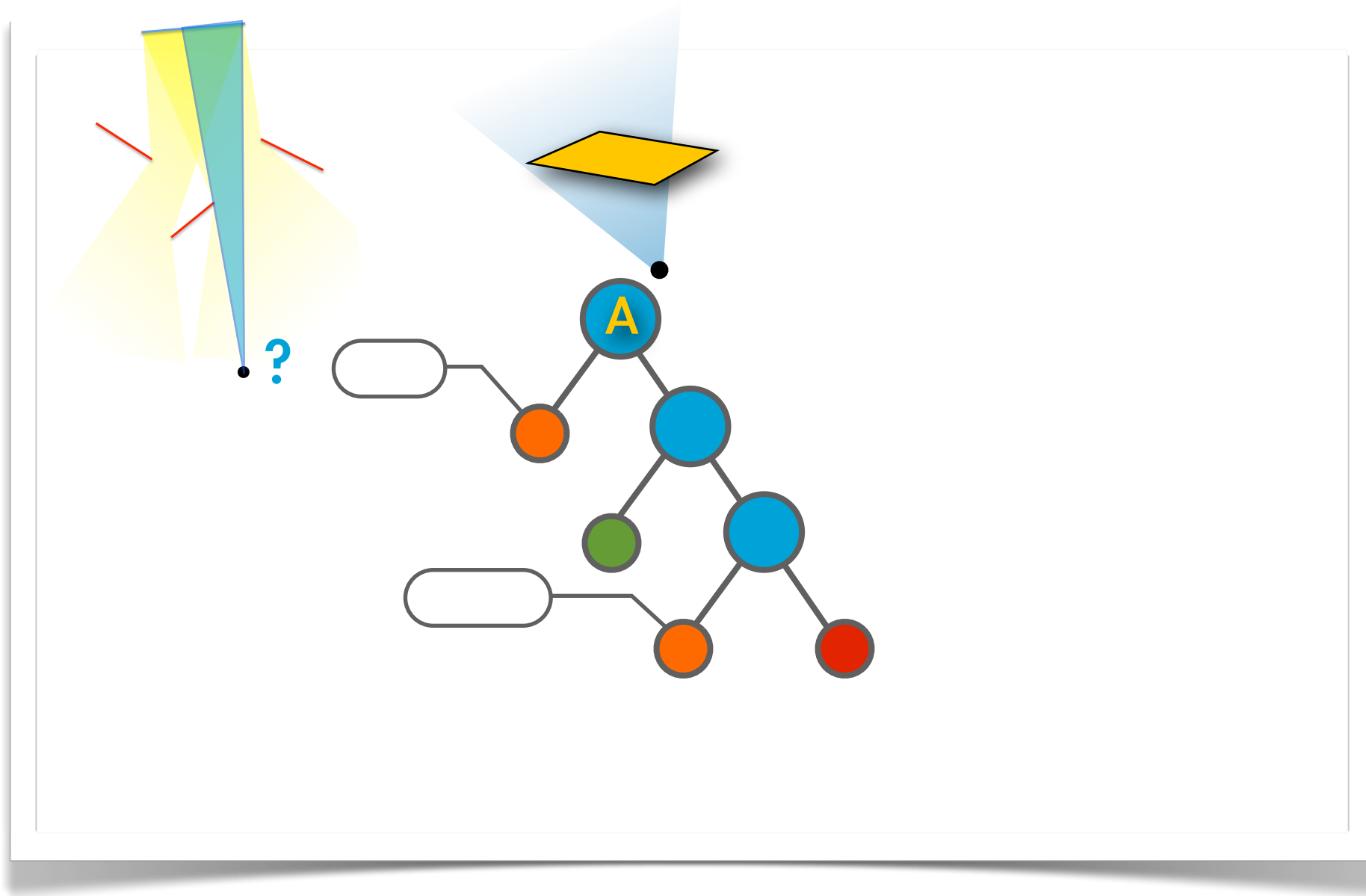
- light visibility from the point ?

Visibility algorithm > view beam orientation



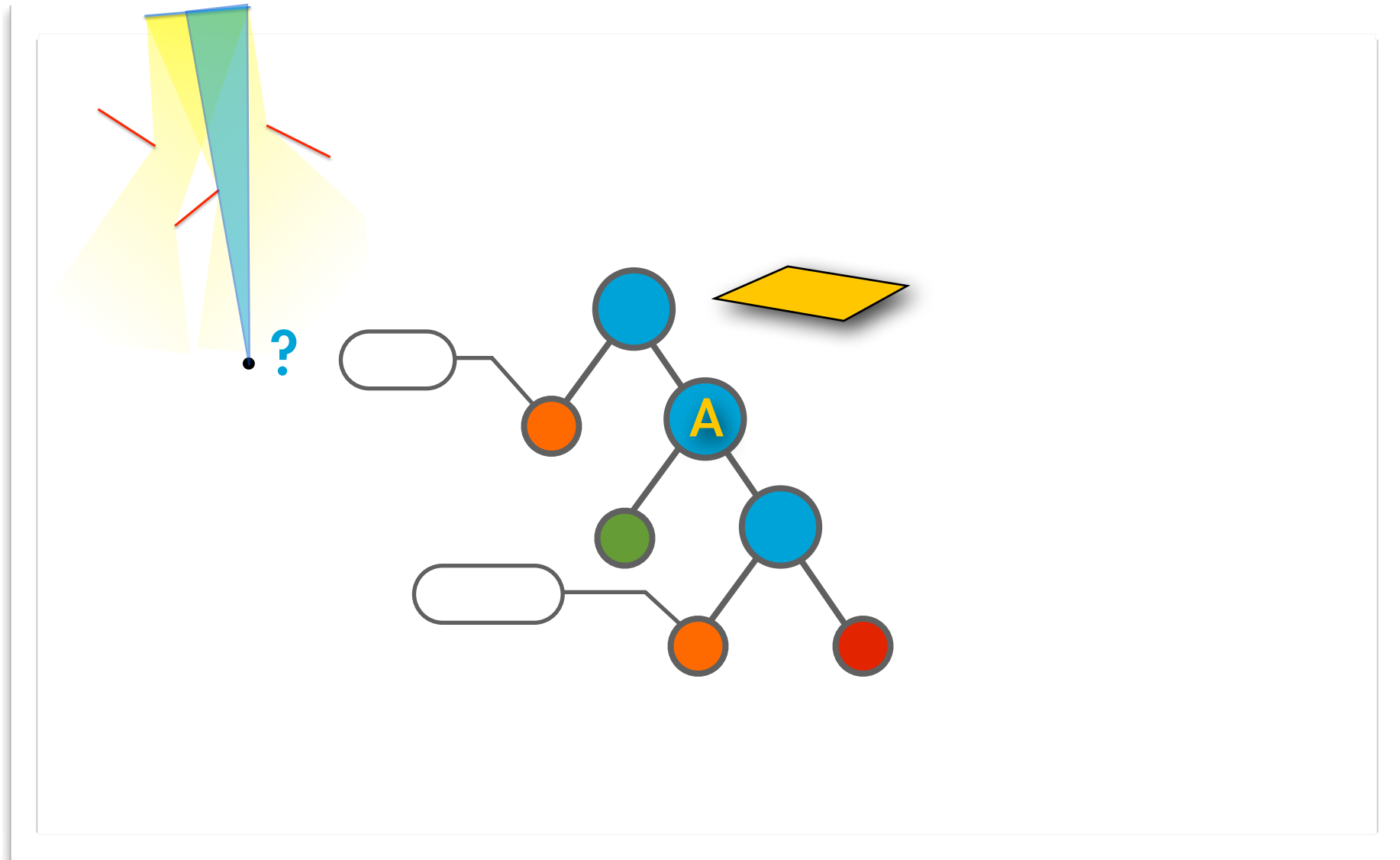
- view beam orientation with respect to a given line h
- not consistent ? Then split the light L !

Visibility algorithm > principle

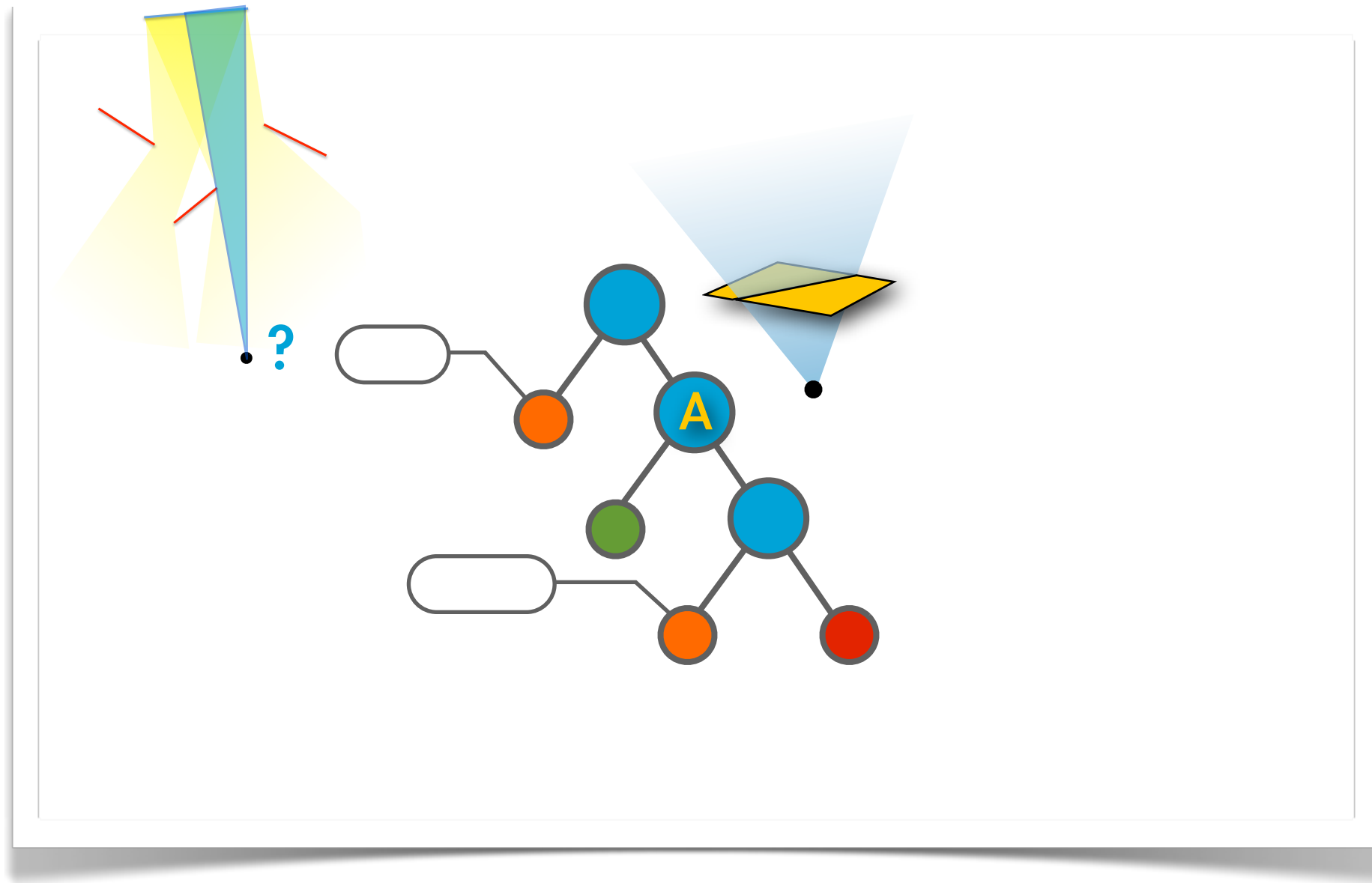


- negative orientation \Rightarrow right child

Visibility algorithm > principle

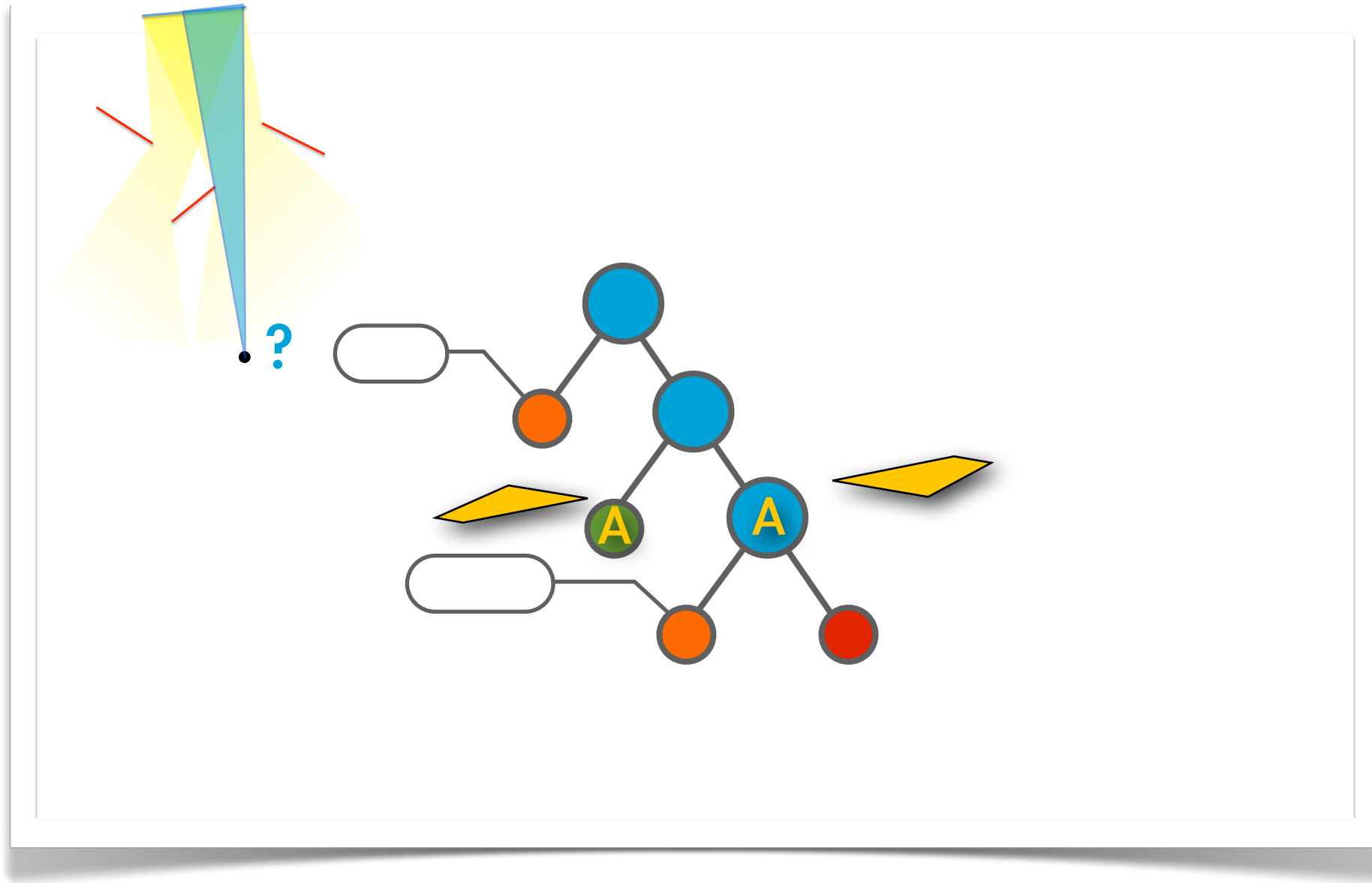


Visibility algorithm > principle



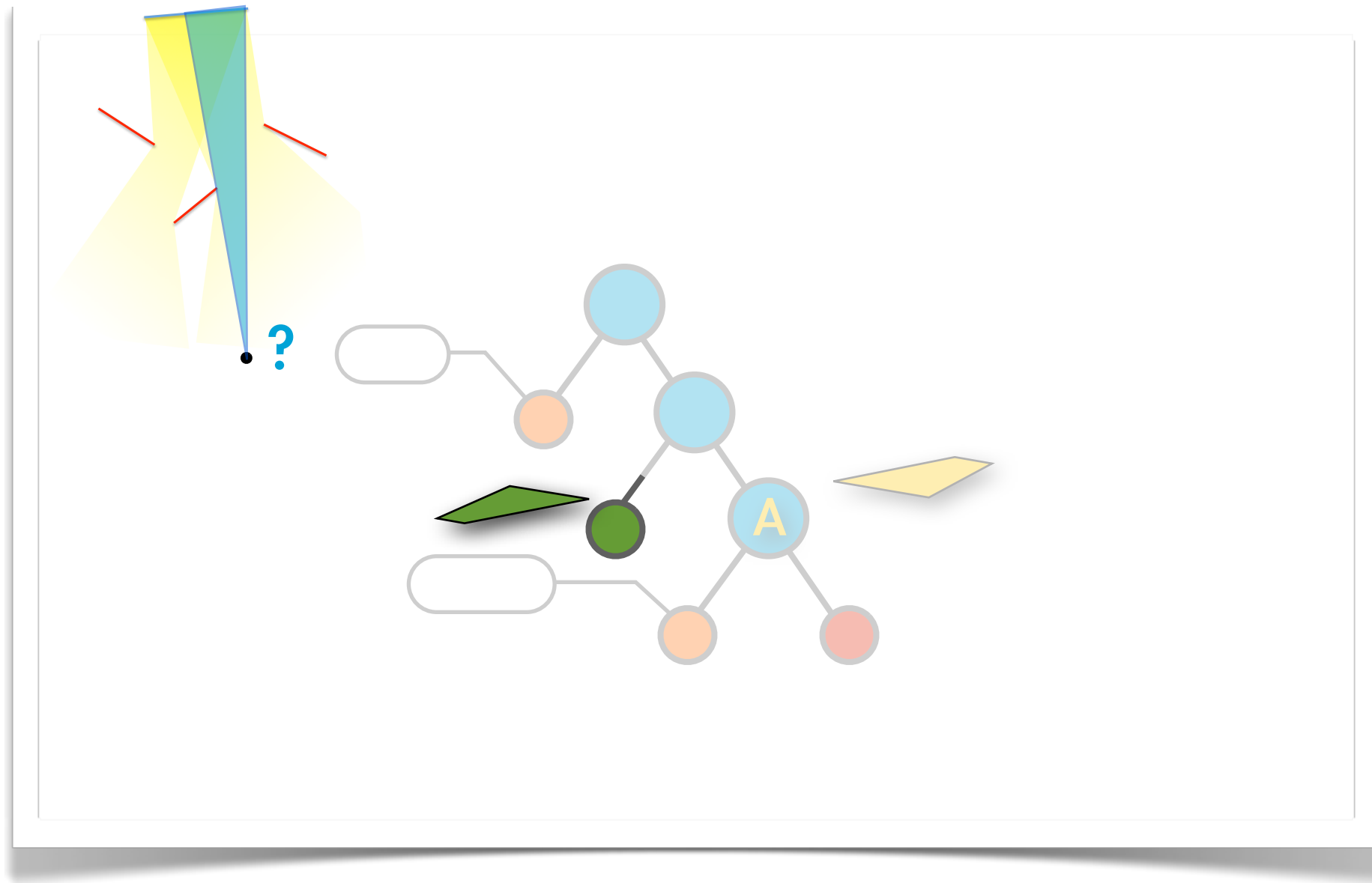
- heterogeneous orientation \Rightarrow split the light

Visibility algorithm > principle



- insert light fragments in the relevant child

Visibility algorithm > principle

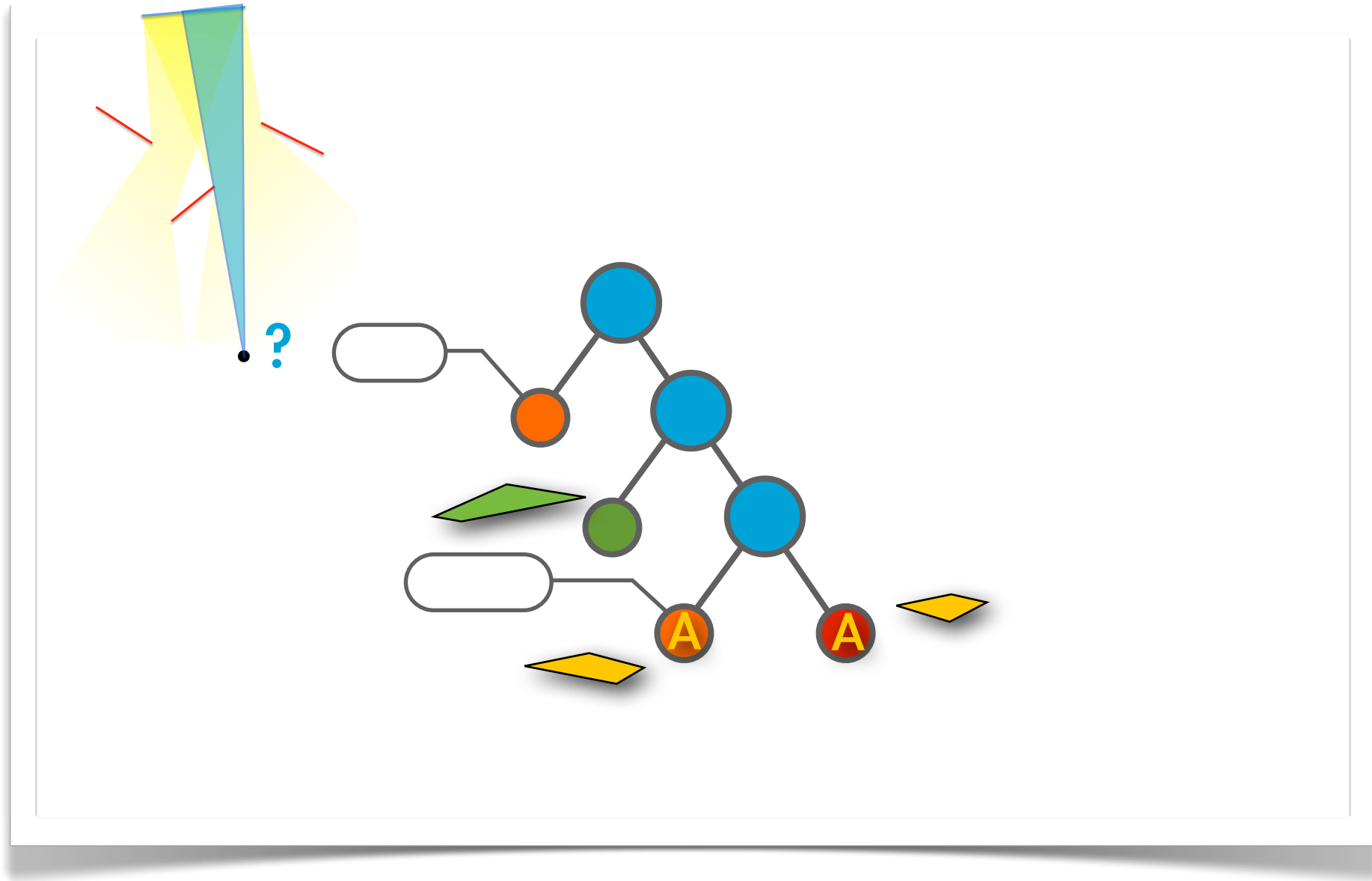


- light fragments reaching a **visible** class are visible

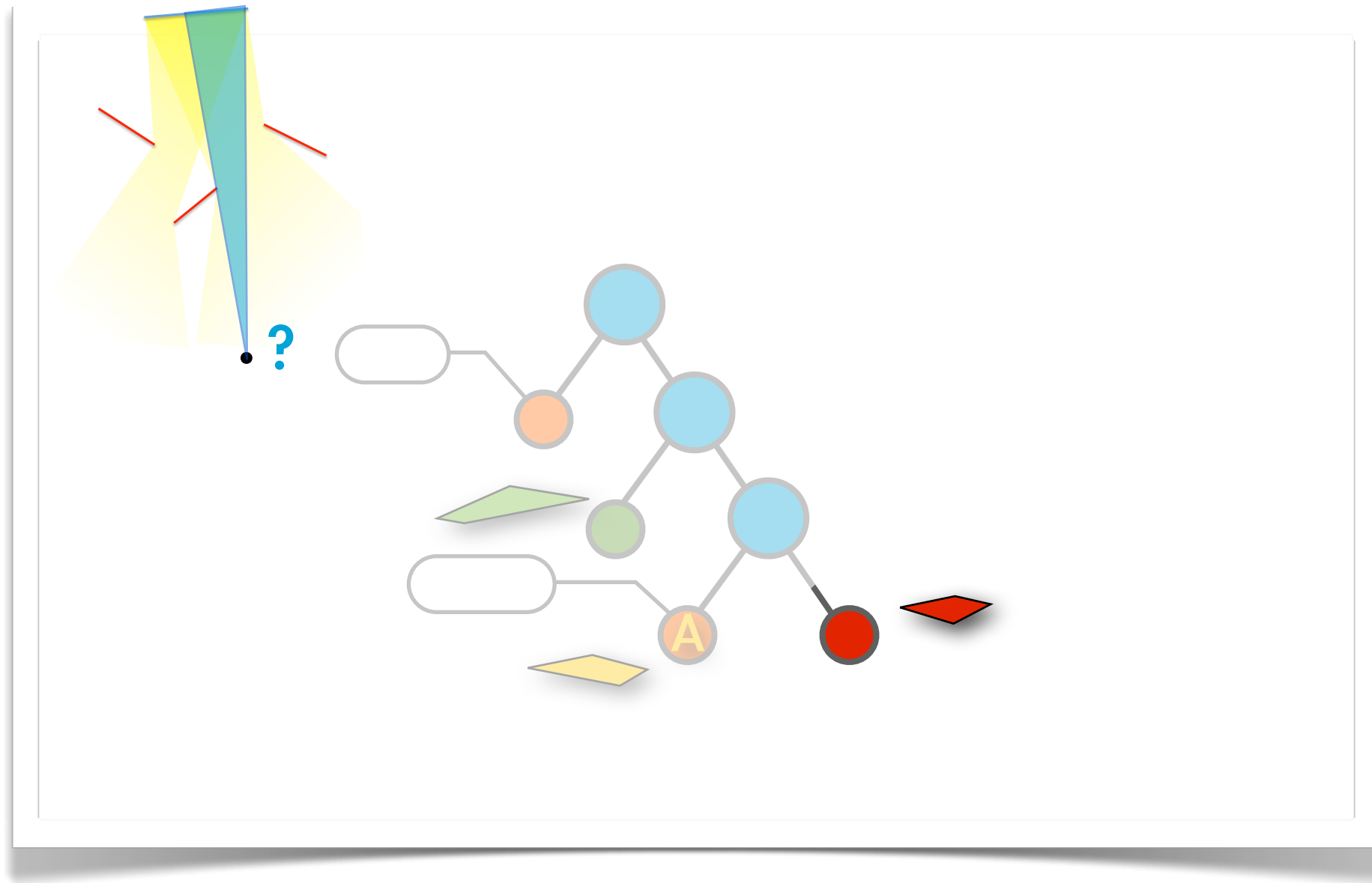
Visibility algorithm > principle



Visibility algorithm > principle

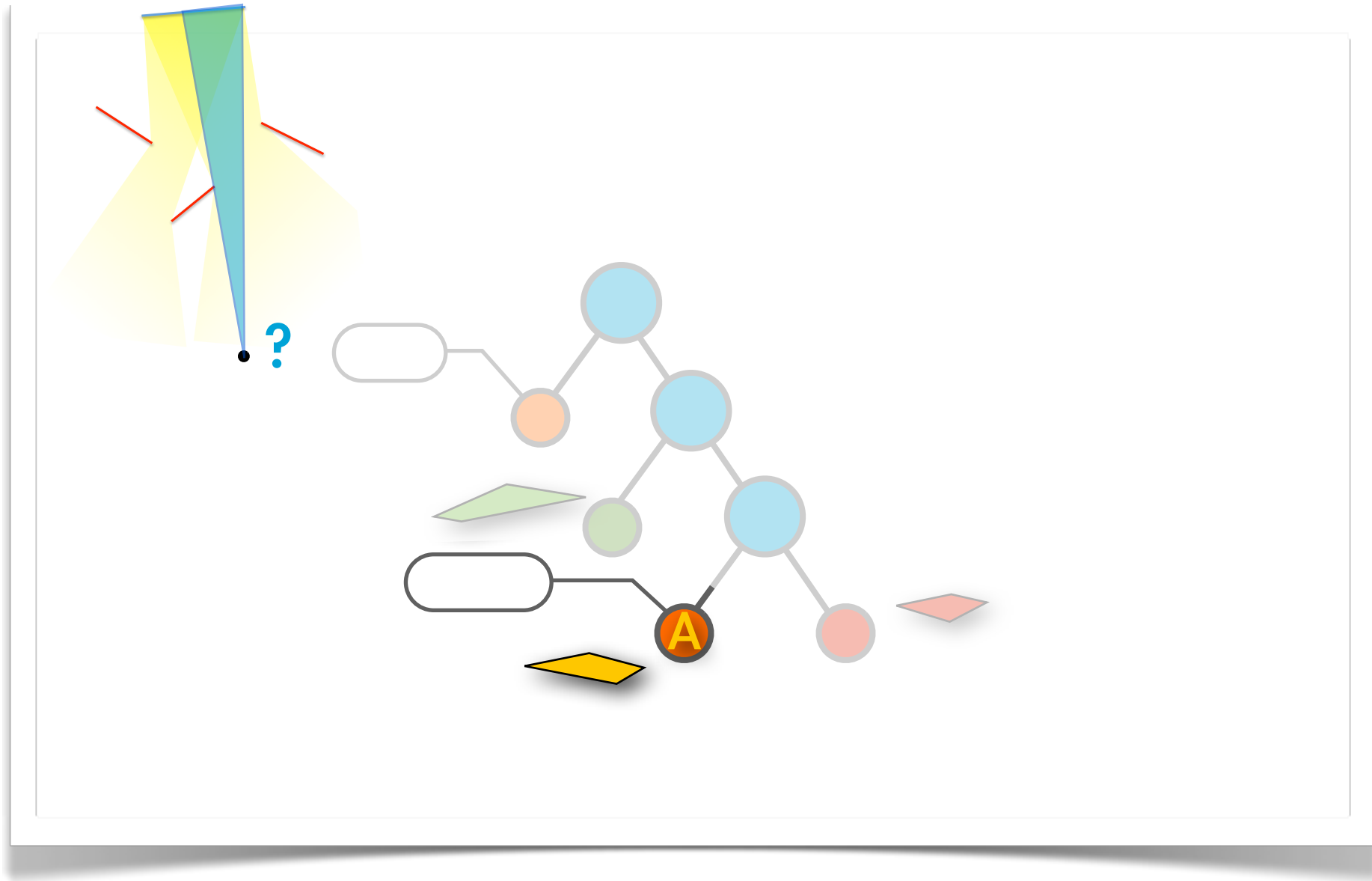


Visibility algorithm > principle



- Light fragments reaching **invisible** class are invisible

Visibility algorithm > principle

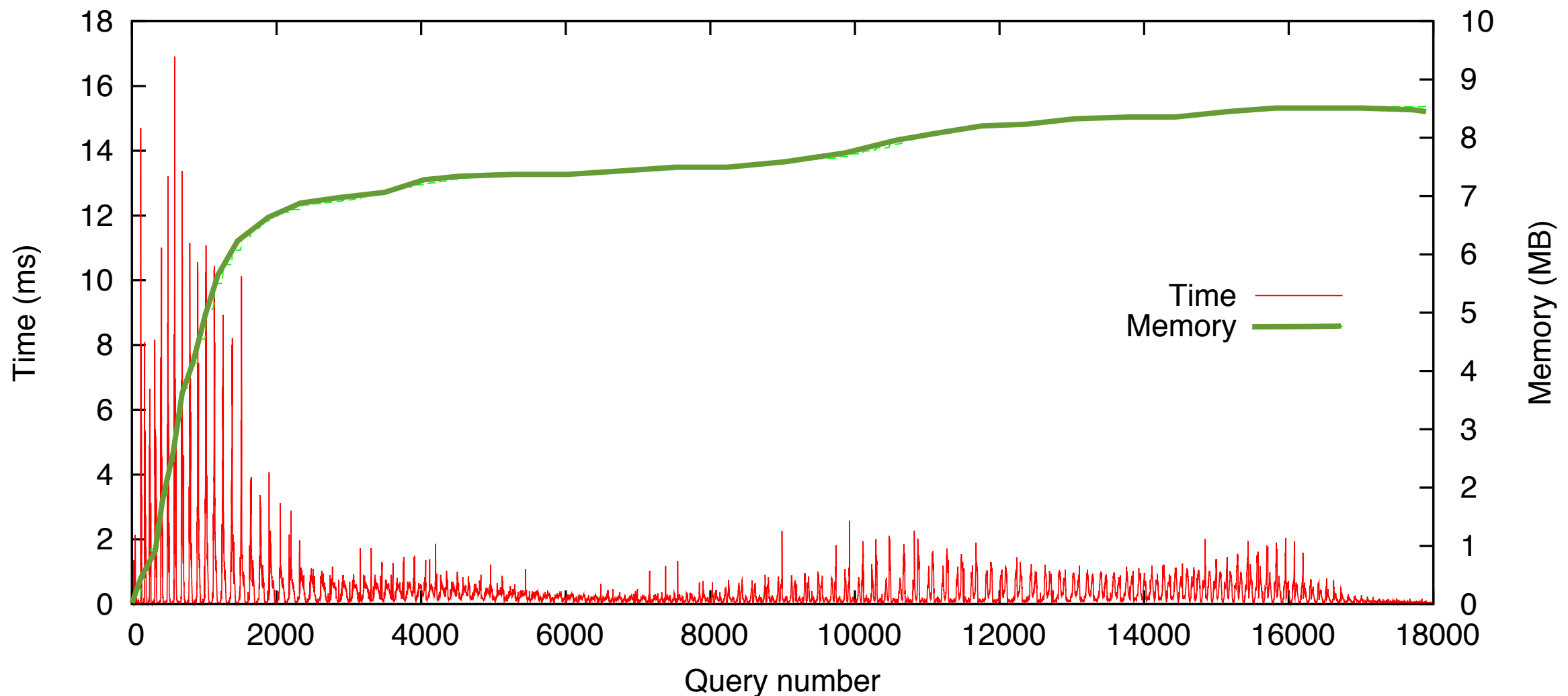


- a light fragment has reached an **undefined** class
- recurse !
- BSP tree grows on demand

Visibility algorithm > key points

Visibility coherence

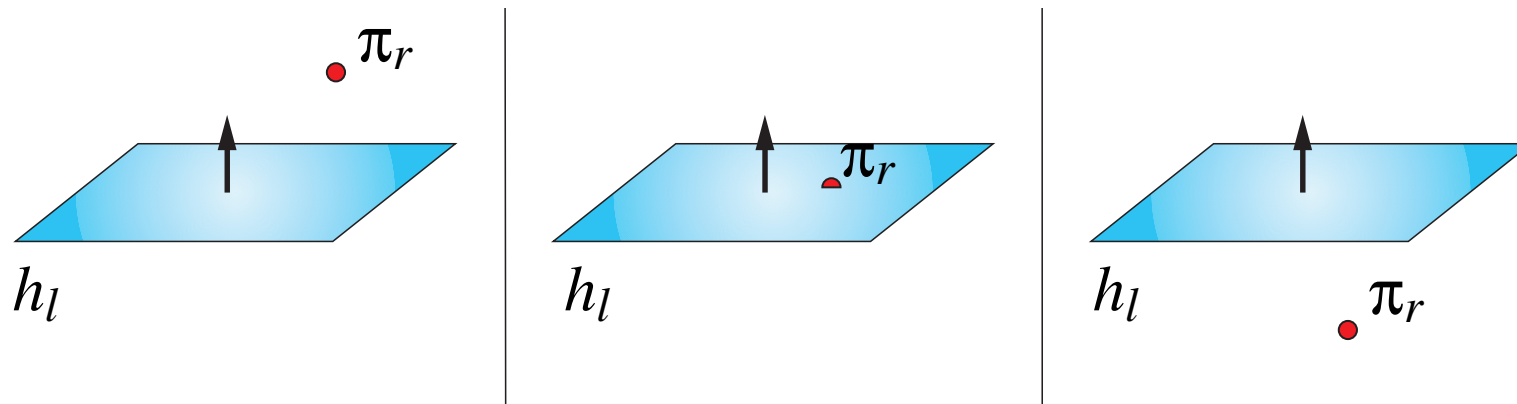
- first queries grow the tree
- next queries take advantage of the first ones



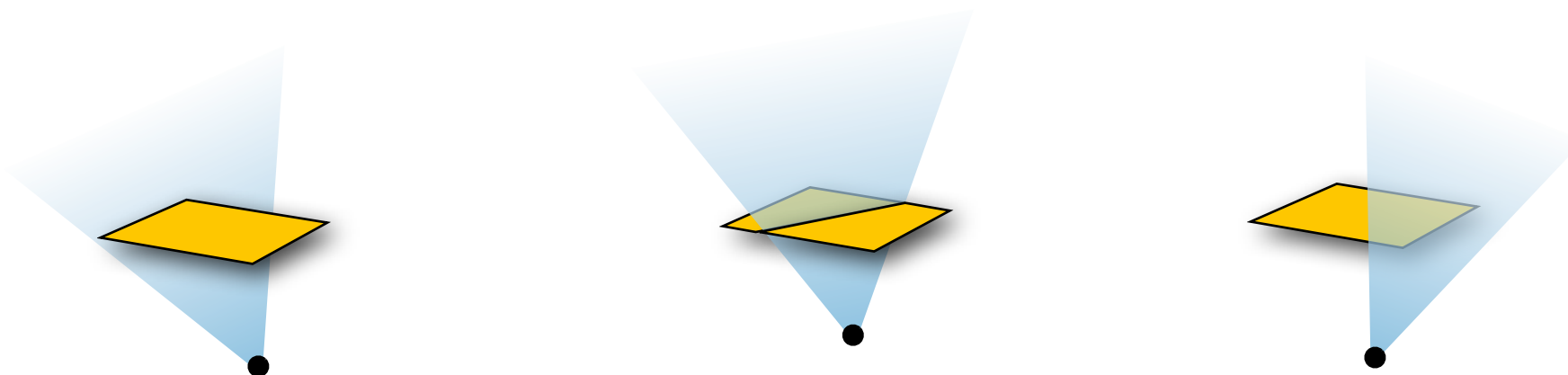
Visibility algorithm > key points

Robustness/efficiency

- BSP tree growth : point/hyperplane sign tests



- view beam filtering : at worst, plane-polygon intersection



- **Geometric basis**
- **Visibility algorithm**
- **Results**

Visibility algorithm > application to soft shadows

- group all image points per visible triangles
 - for each group
 - get the occluders and initialize a BSP tree
 - for each image point
 - run a point-light query
 - compute analytic direct illumination

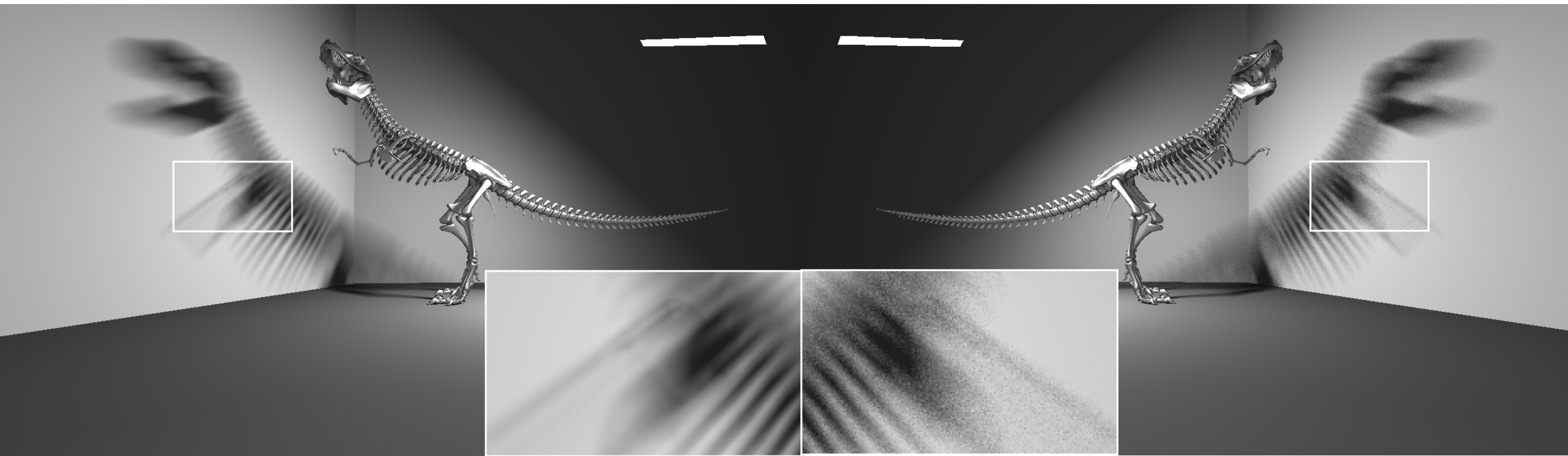
Ray-traced shadows

- optimized (CPU) ray-tracer (SAH kd-tree, SIMD, multithreading)
- 4 shadow rays at a time
- uncorrelated stratified sampling

Configuration

- 2.67 GHz Intel Core i7 920, 4GB of memory
- all picture at 1280 x 720, 1 primary ray per pixel
- all tests use 4 threads

Results > comparison on equivalent time



T-Rex (26K triangles)

Ours

Time : 6.5s

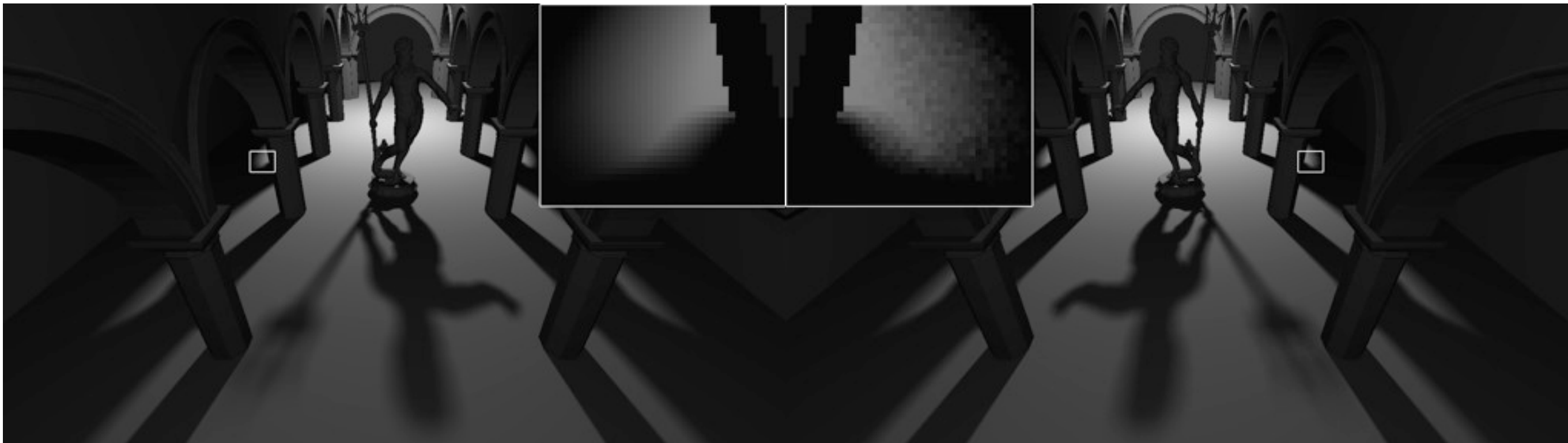
Memory : 19 MB

RT

Time : 7s

32 samples

Results > comparison on equivalent time



Sponza with Neptune (115K triangles)

Ours

Time : 7s

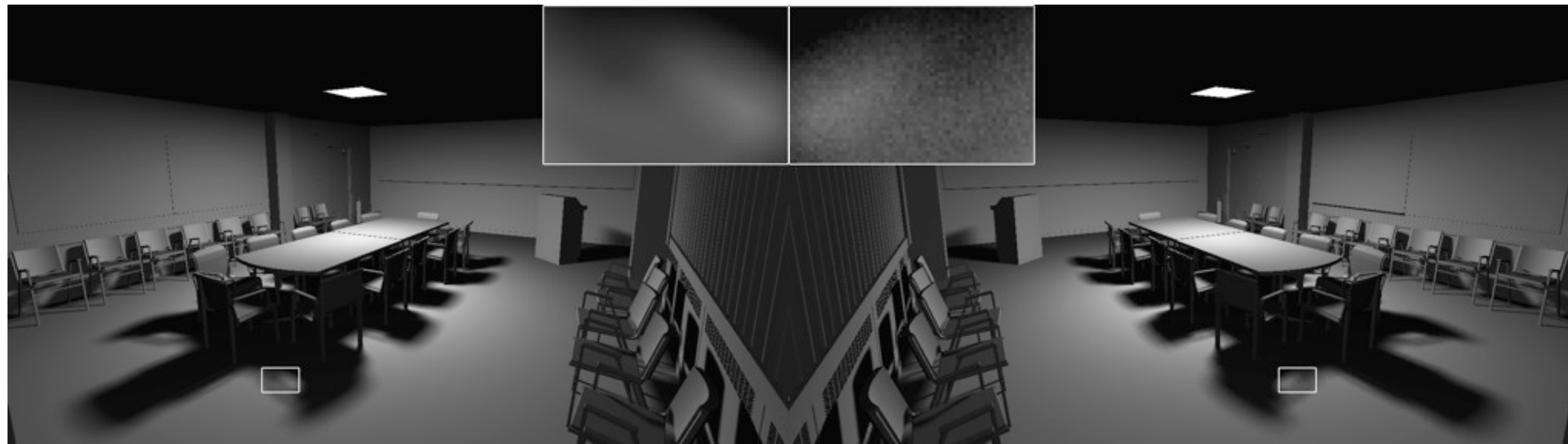
Memory : 16 MB

RT

Time : 7s

32 samples

Results > comparison on equivalent time



Conference (282K triangles)

Ours

Time : 6s

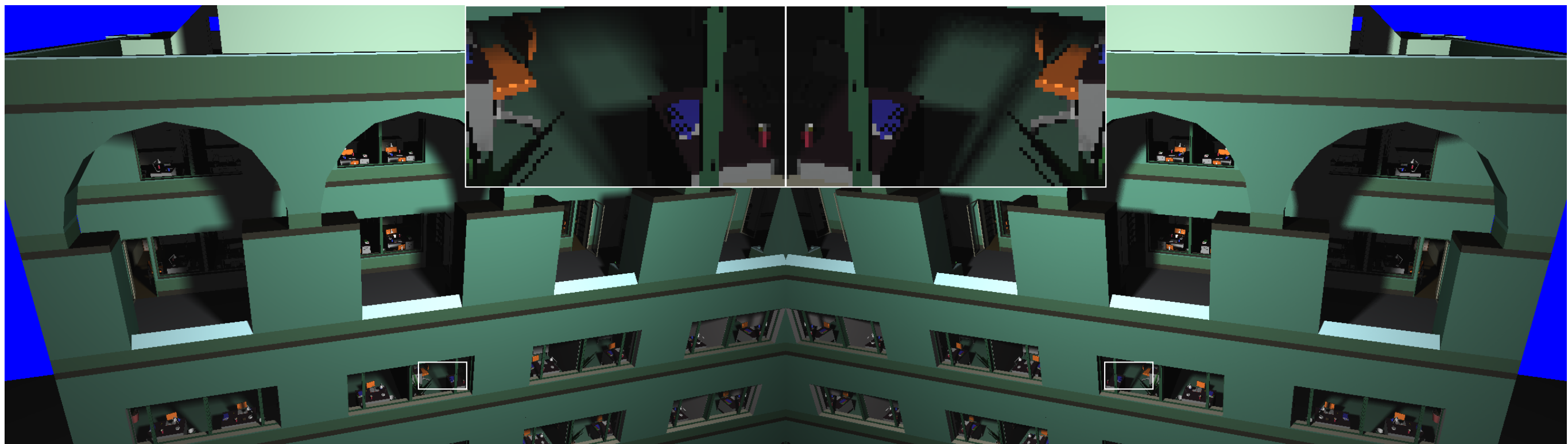
Memory : 20 MB

RT

Time : 6s

32 samples

Results > comparison on equivalent time



Soda Hall (2147K triangles)

Ours

Time : 5s

Memory : 20 MB

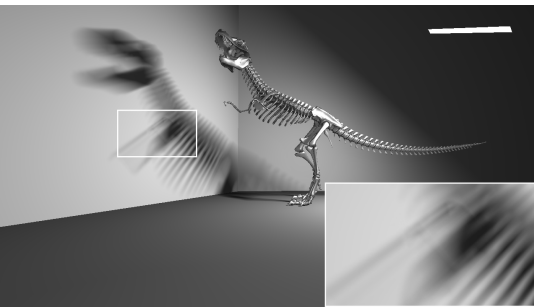
RT

Time : 8s

32 samples

Results > comparison on equivalent quality

Ours RT



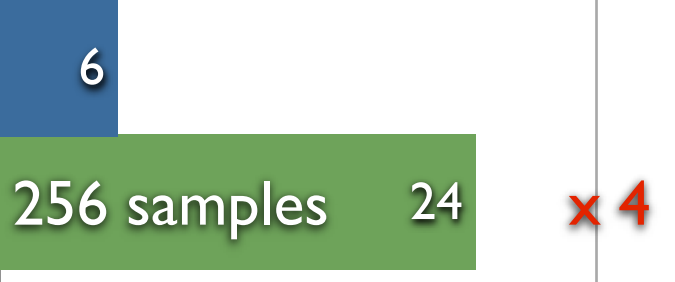
T-Rex



Sponza



Conf.



Soda.



0 30 60 90

Conclusion >

What we have shown:

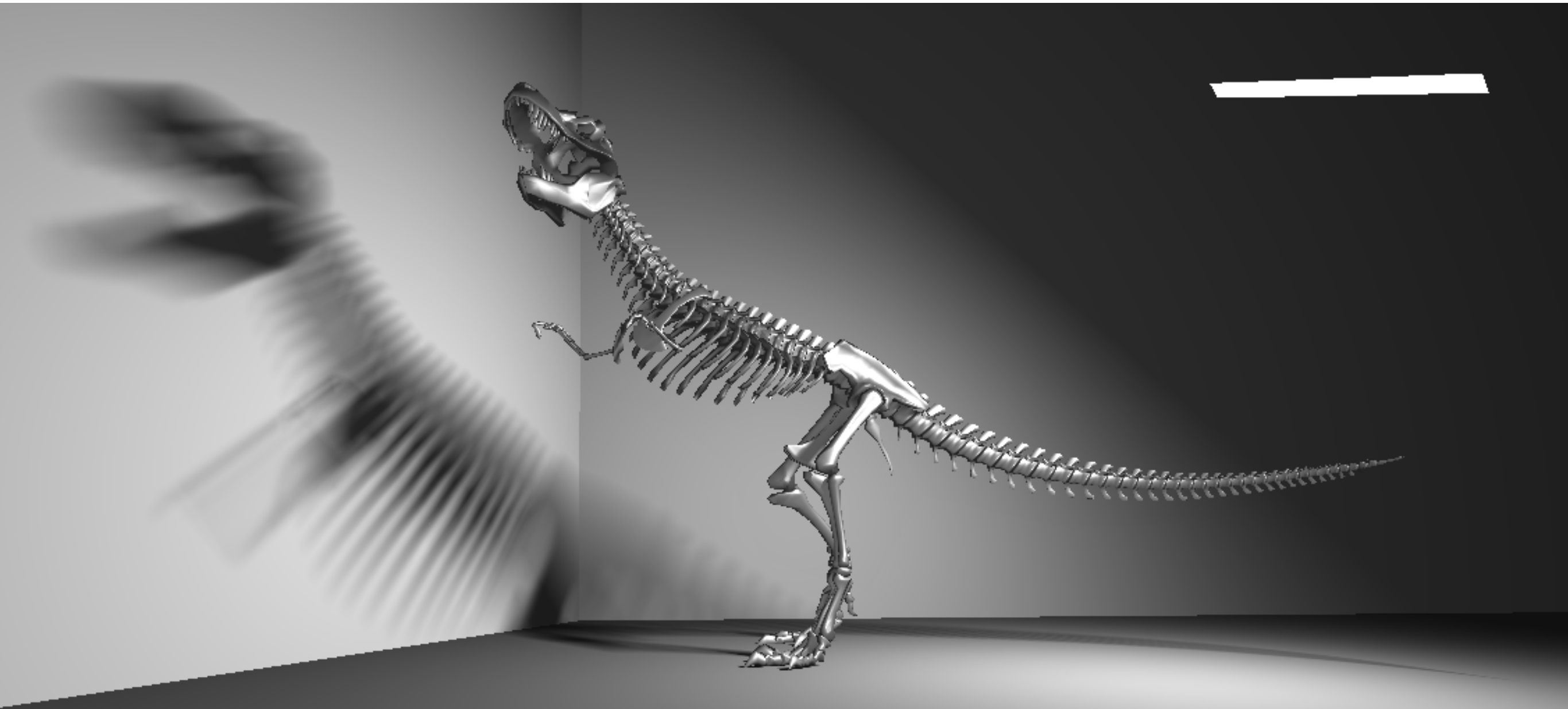
- coherent from-polygon visibility representation
- exact and analytic soft shadows

What we hope:

- analytic (from polygon) visibility can be robust and efficient
- may be useful for many other problems

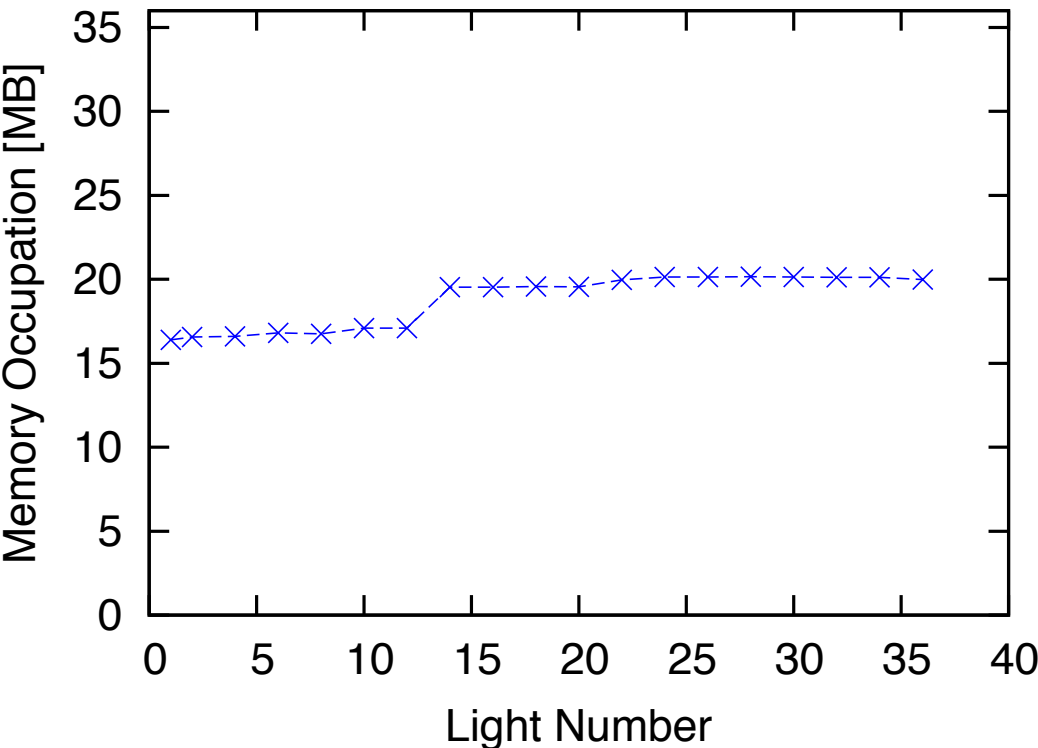
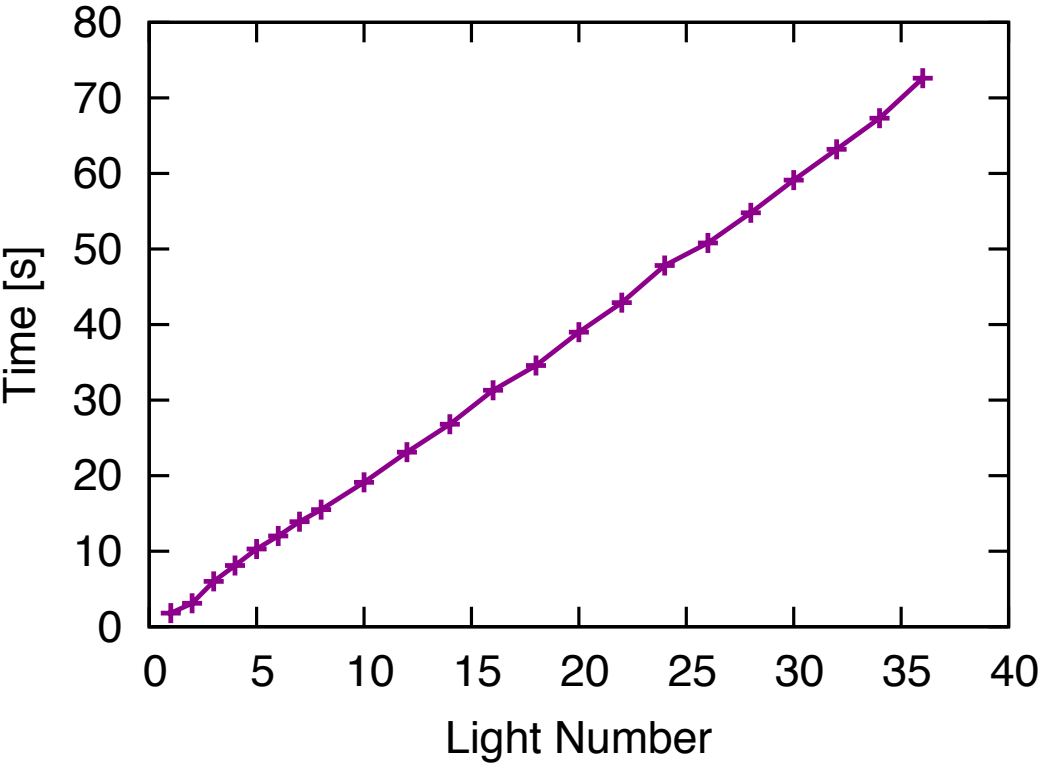
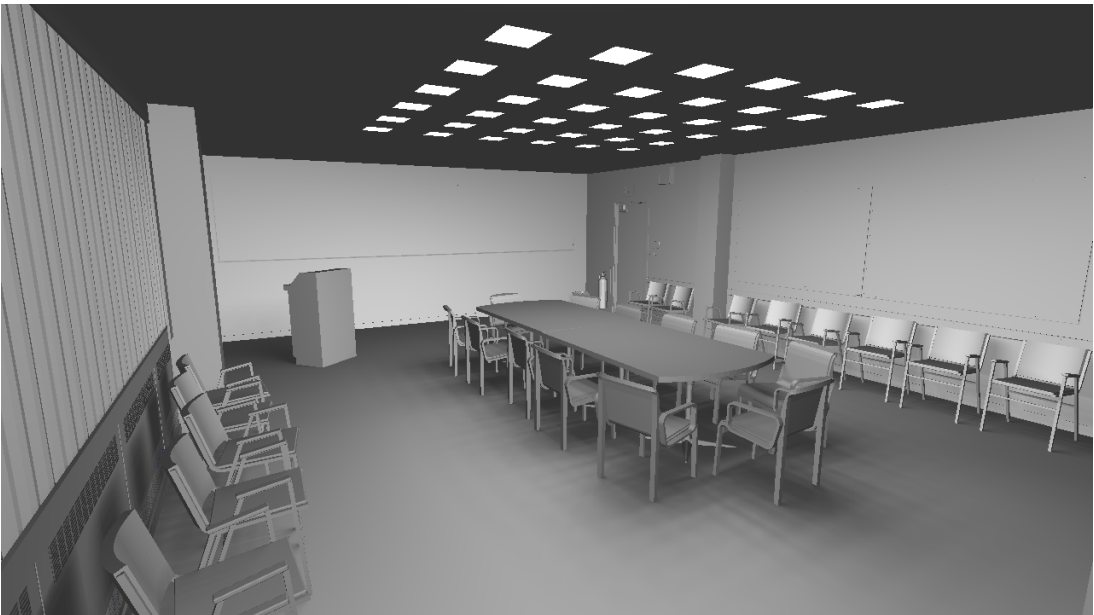
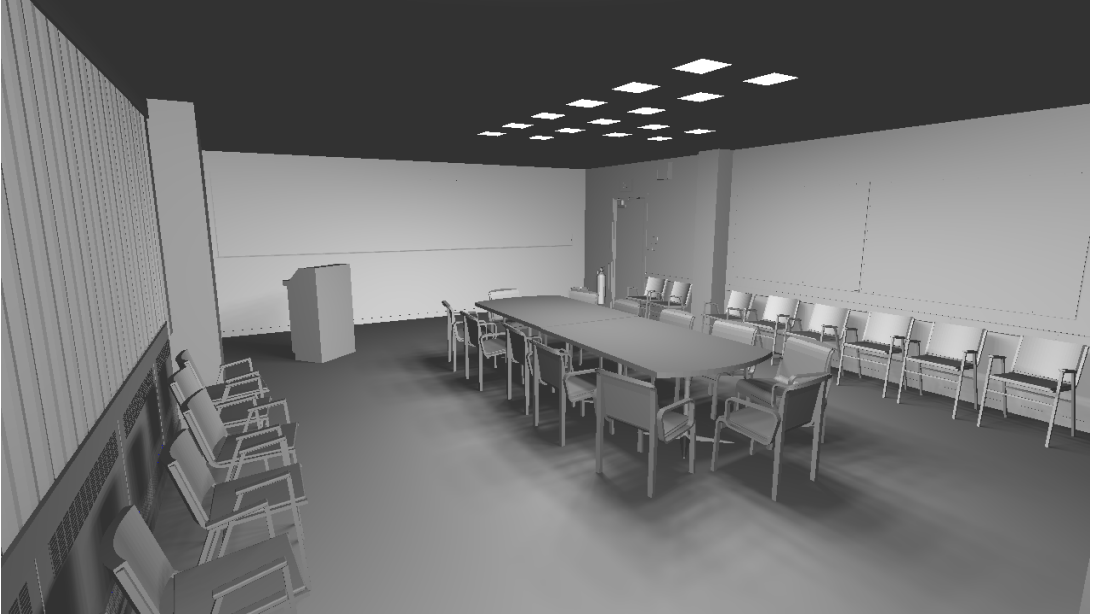
What we plane

- occlusion  visibility ? (done)



Questions ?

Results > increasing the number of lights



Results > increasing the area light source

