# Global Illumination I

Shih-Chin Weng
shihchin.weng@gmail.com

# What is Global Illumination?

*Photo by Abigail Keenan*


Photo by Kaushik Panchal

Ray tracing is everywhere in VFX & animation industry!

# The State of Rendering


Walt Disney
ANIMATION STUDIOS
Hyperion Renderer

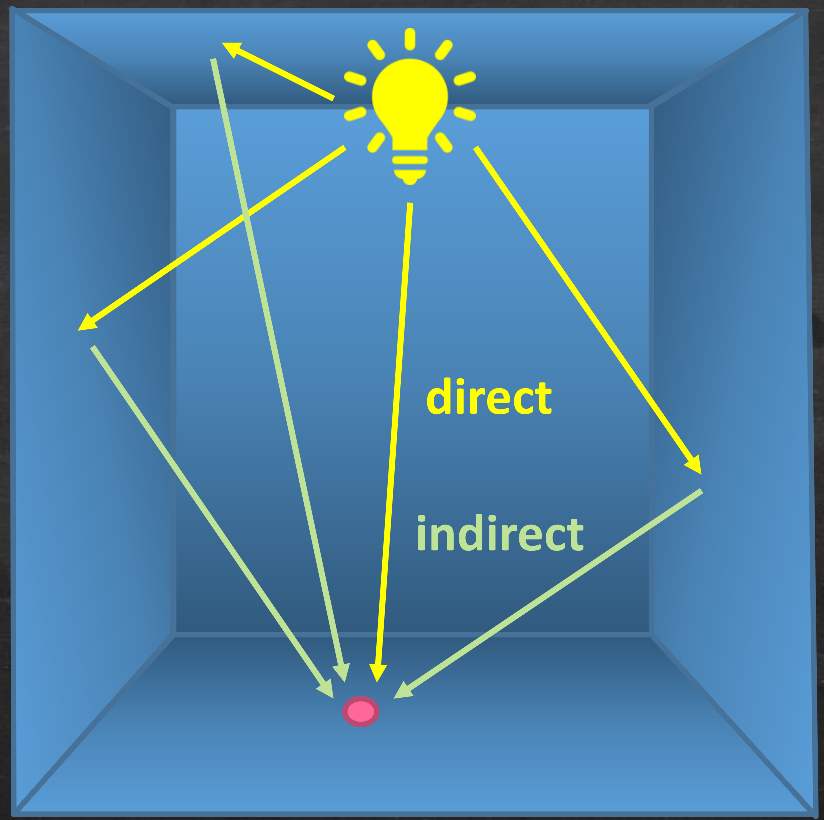

Manuka Renderer


Disney · PIXAR
RenderMan

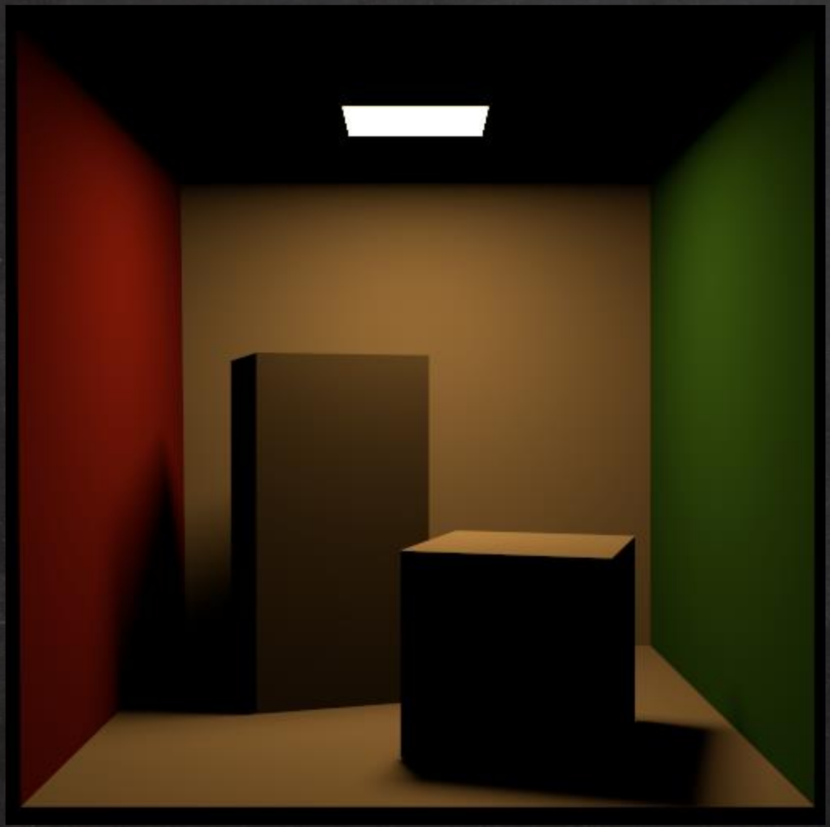

arnold renderer

# Basic Concepts

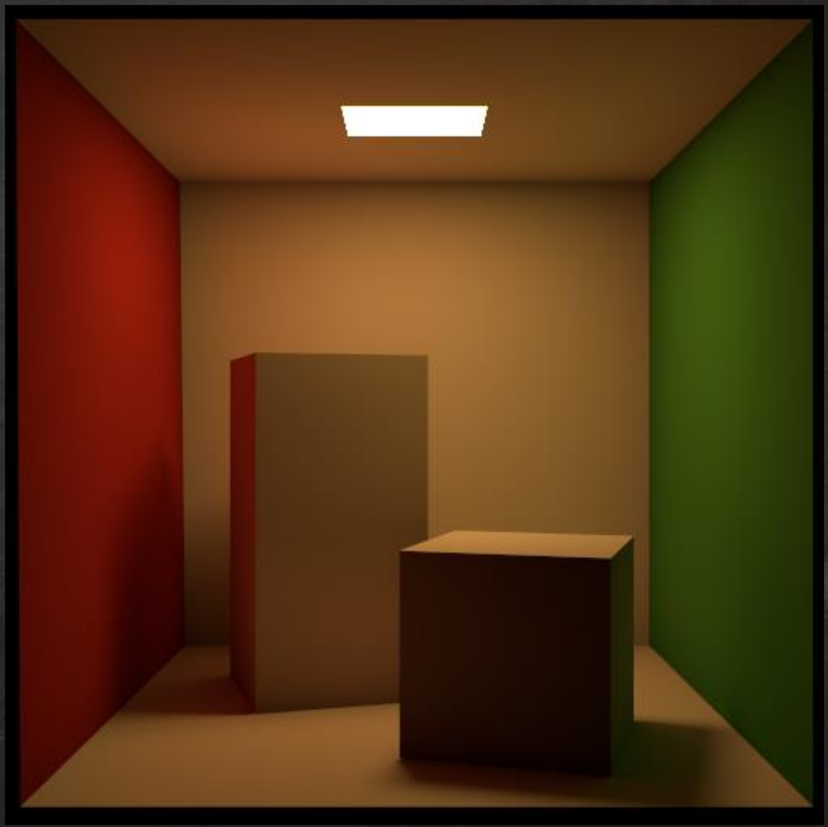
# Where Does Light Come From?



# Global = Direct + Indirect Lighting

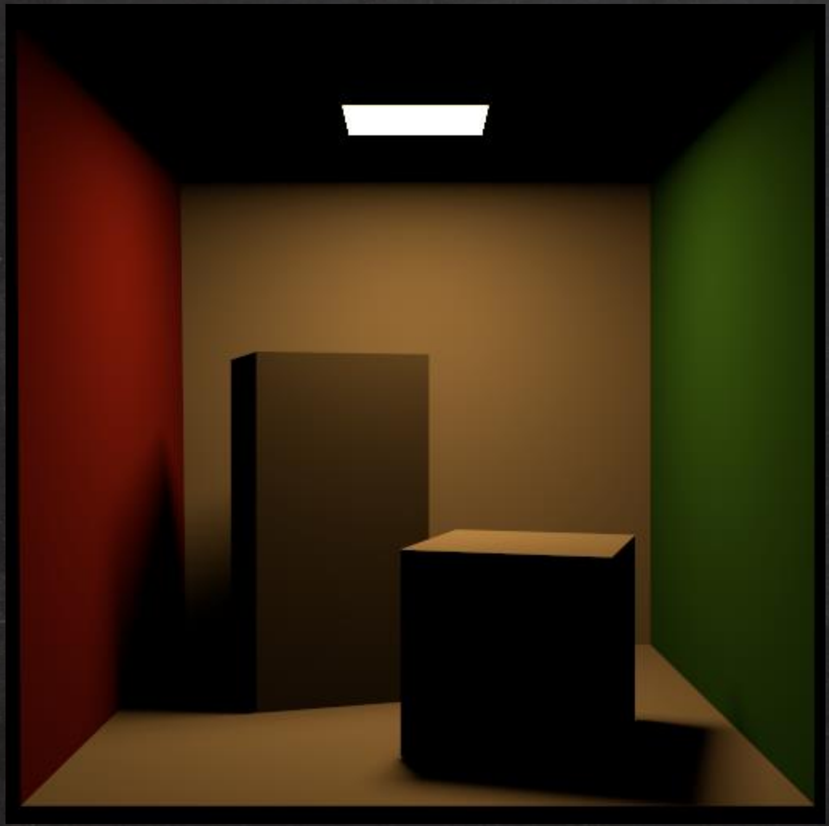


Direct Illumination

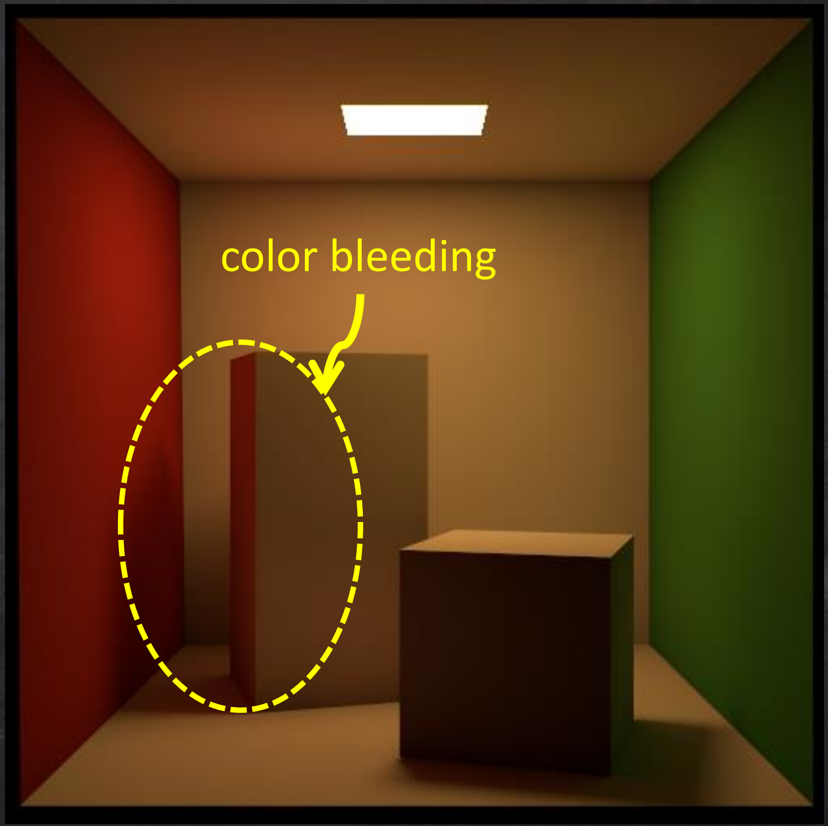


Global Illumination

# Global = Direct + Indirect Lighting

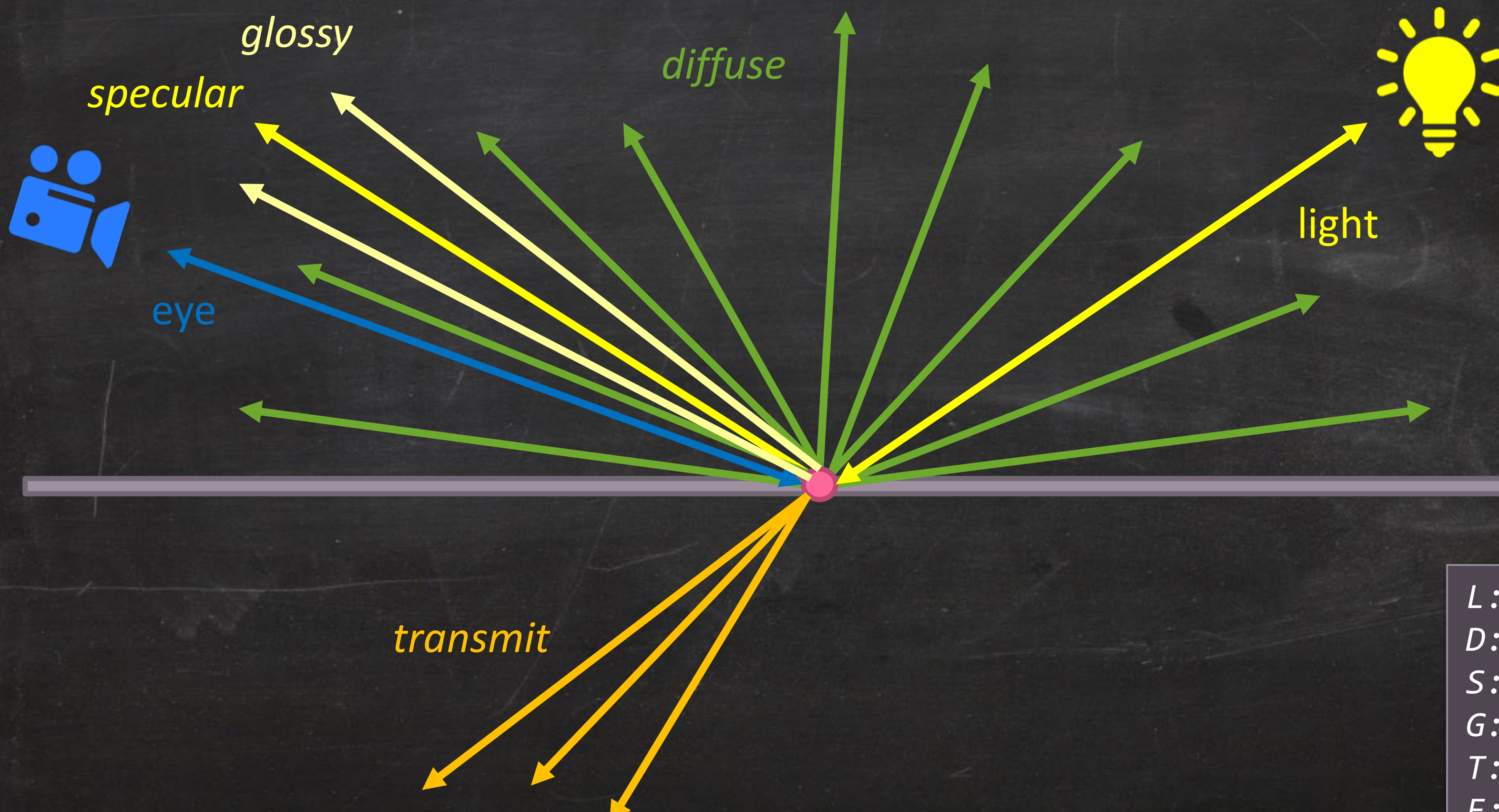


Direct Illumination



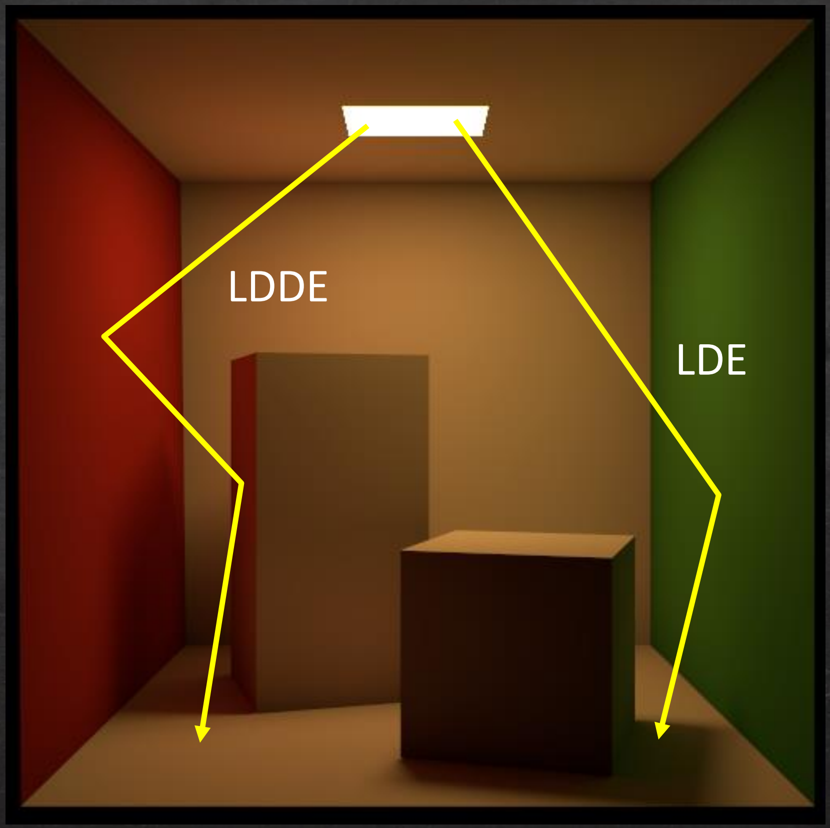

Global Illumination

# Light Path Expression




L: light
D: diffuse
S: specular
G: glossy
T: transmit
E: eye

# Light Path Expression (Cont'd)



Recap

# Radiometry

- Irradiance $E = \frac{d\Phi}{dA}$

  *pre area incoming flux at a surface*

- Radiance $L = \frac{d^2\Phi}{d\omega dA^{\perp}} = \frac{d^2\Phi}{d\omega dA \cos\theta}$

  *flux per solidangle per projected area*

$$L\cos\theta = \frac{d^2\Phi}{dA\,d\omega} = \frac{dE}{d\omega} \Rightarrow L\cos\theta\, d\omega = dE$$

Recap

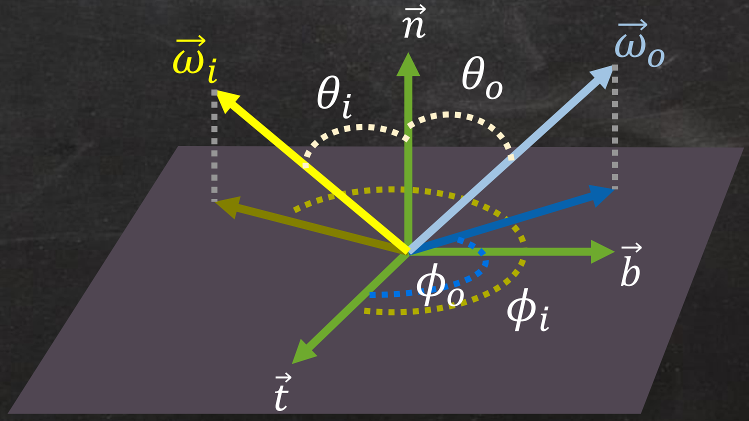
# BRDF Definition

$$\frac{spending}{income}$$

outgoing radiance

$$f(\vec{\omega}_i, \vec{\omega}_o) = \frac{dL_r(\vec{\omega}_o)}{dE_i(\vec{\omega}_i)} = \frac{dL_r(\vec{\omega}_o)}{L_i(\vec{\omega}_i)\cos\theta_i\, d\omega_i}$$

incoming irradiance



# Compute $L_r(\vec{\omega}_o)$ from BRDF (temp)

$$f(\vec{\omega}_i, \vec{\omega}_o) = \frac{dL_r(\vec{\omega}_o)}{dE_i(\vec{\omega}_i)} = \frac{dL_r(\vec{\omega}_o)}{L_i(\vec{\omega}_i)(\vec{\omega}_i \cdot \vec{n})d\vec{\omega}_i}$$

$$L_i(\vec{\omega}_i)f(\vec{\omega}_i, \vec{\omega}_o)(\vec{\omega}_i \cdot \vec{n}) = L_i(\vec{\omega}_i)\frac{dL_r(\vec{\omega}_o)}{L_i(\vec{\omega}_i)(\vec{\omega}_i \cdot \vec{n})d\vec{\omega}_i}(\vec{\omega}_i \cdot \vec{n}) = \frac{dL_r(\vec{\omega}_o)}{d\vec{\omega}_i}$$

$$\int_\Omega \frac{dL_r(\vec{\omega}_o)}{d\vec{\omega}_i} d\vec{\omega}_i = L_r(\vec{\omega}_o)$$

hemisphere

reflected radiance

# ★Render Equation

$$L_o(x, \vec{\omega}_o) = L_e(x, \vec{\omega}_o) + \int_\Omega L_i(x, \vec{\omega}_i) f(\vec{\omega}_i, \vec{\omega}_o)(\vec{\omega}_i \cdot \vec{n}) d\vec{\omega}_i$$

$$= L_e(x, \vec{\omega}_o) + L_r(x, \vec{\omega}_o)$$

emission　　reflection

# ★ Render Equation

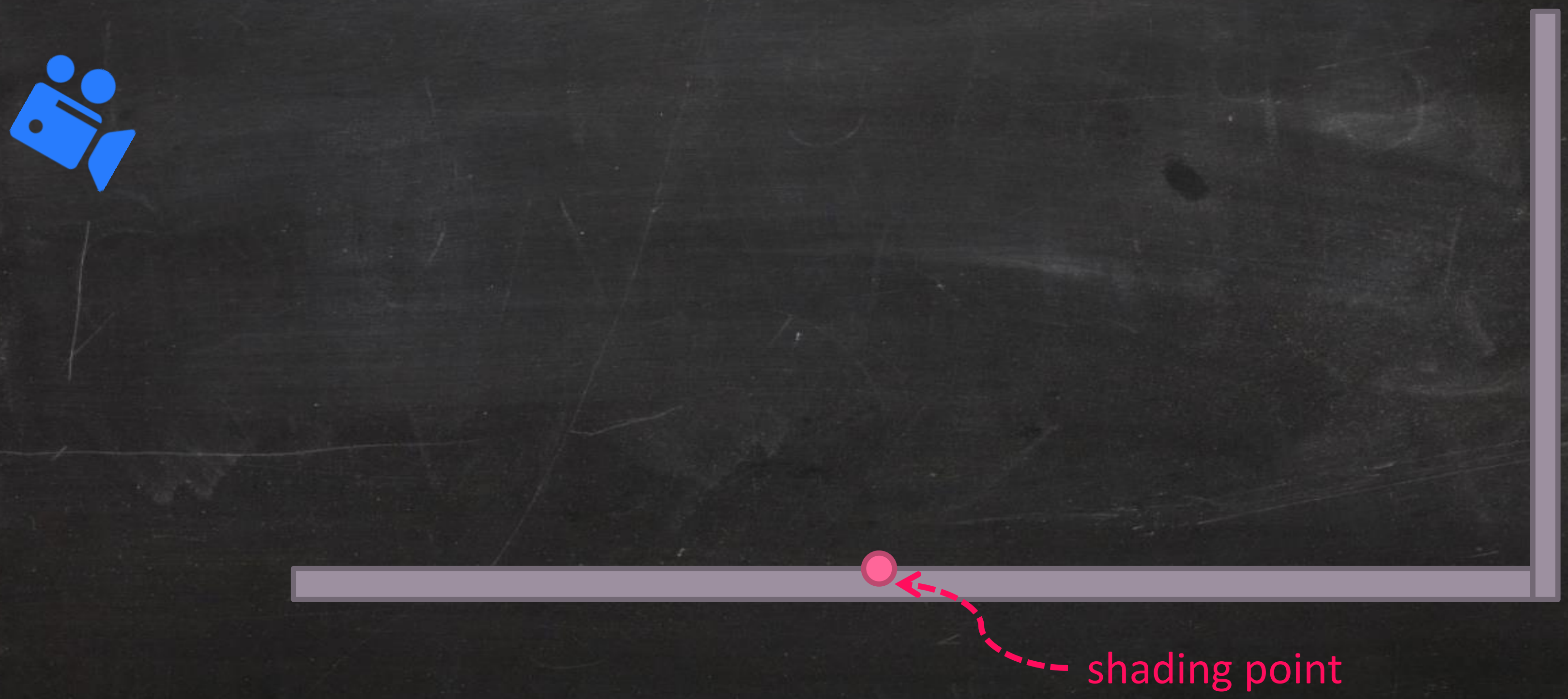
*Oops! There is another render equation nested inside!!*

$$L_o(x,\vec{\omega}_o) = L_e(x,\vec{\omega}_o) + \int_\Omega L_i(x,\vec{\omega}_i) f(\vec{\omega}_i,\vec{\omega}_o)(\vec{\omega}_i \cdot \vec{n}) d\vec{\omega}_i$$

$$= L_e(x,\vec{\omega}_o) + L_r(x,\vec{\omega}_o)$$

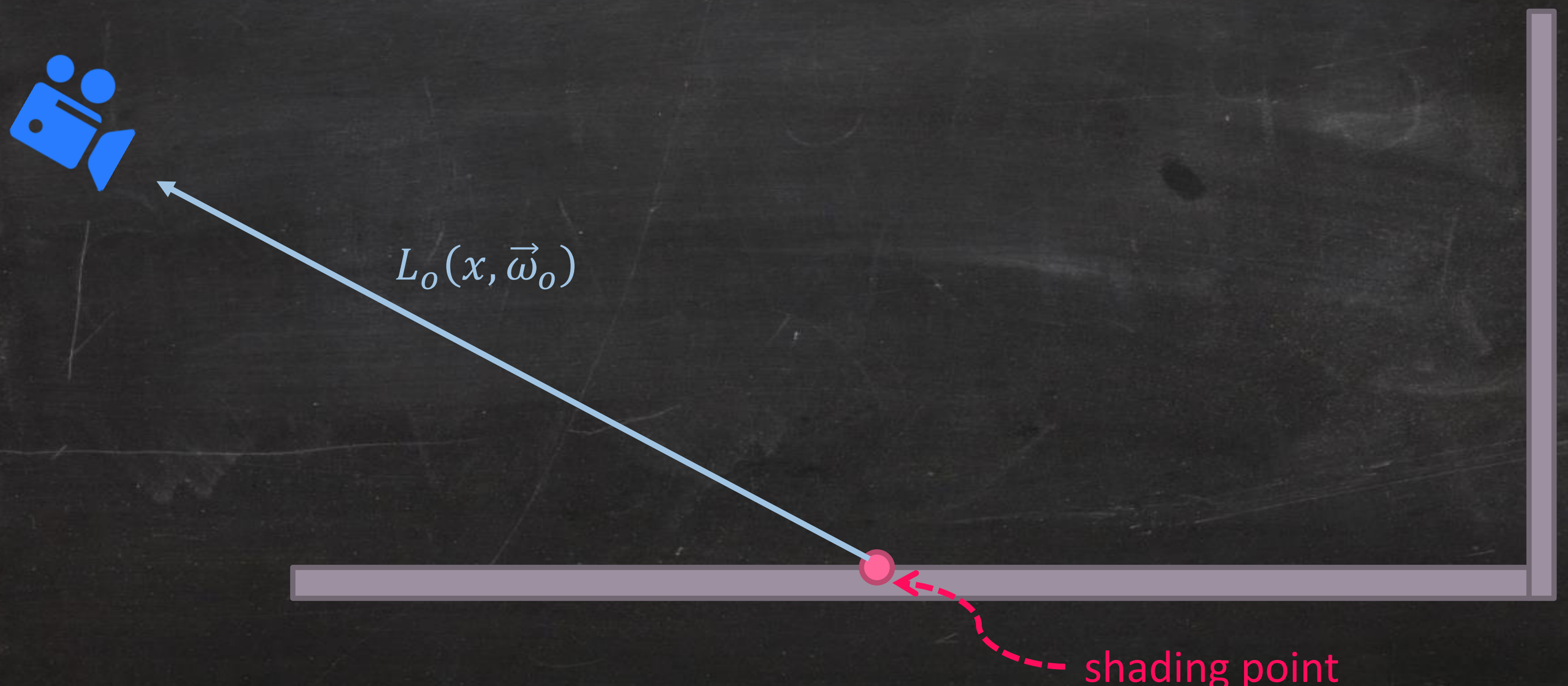
emission reflection

# Recursive Ray Tracing



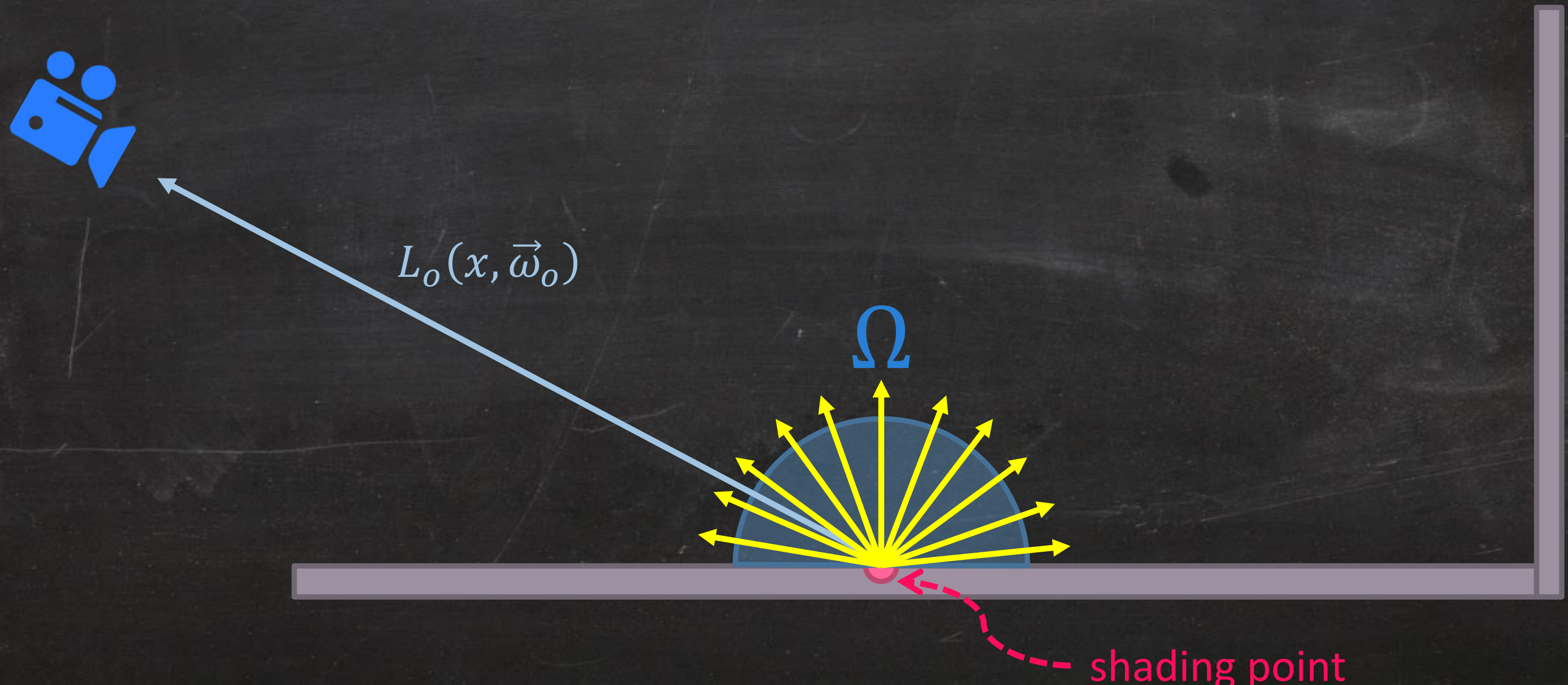

[Inspired by Křivánek, '10]

# Recursive Ray Tracing



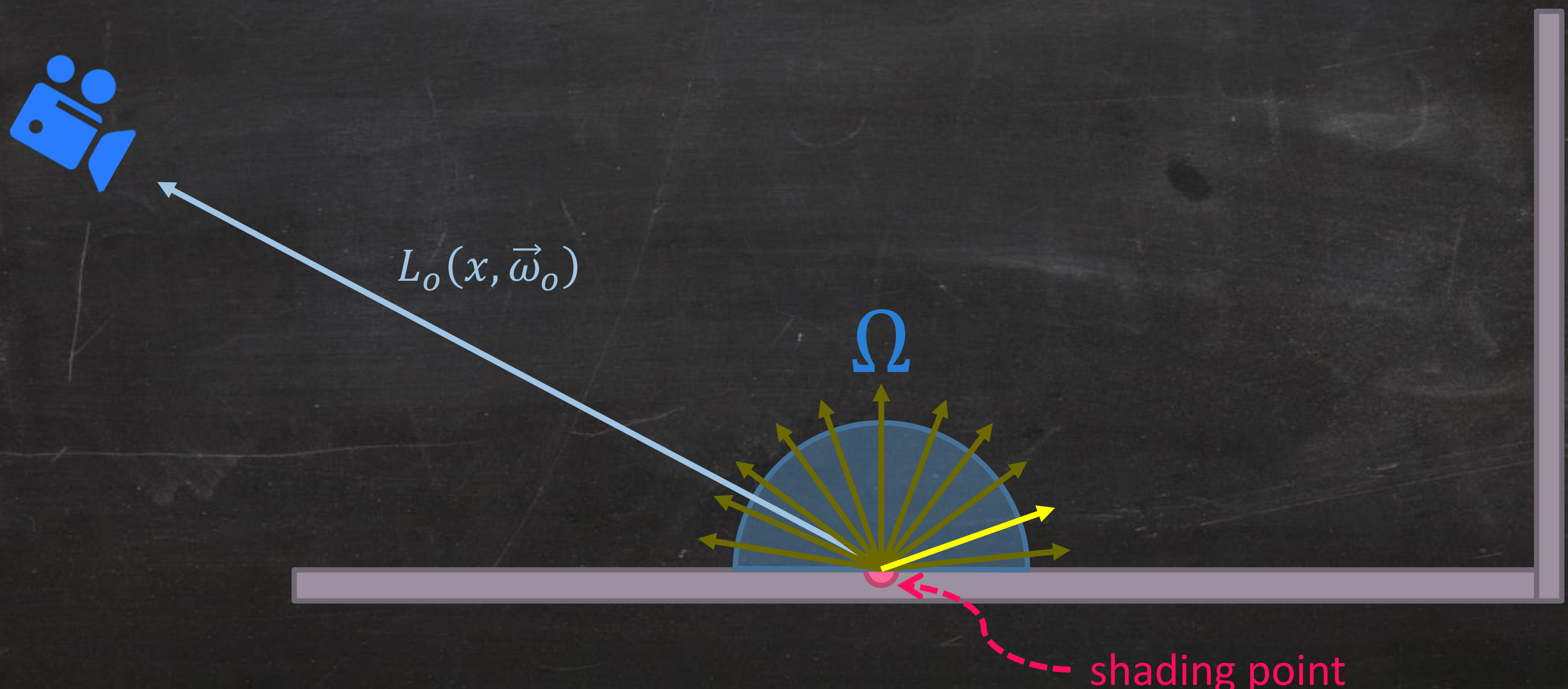

[Inspired by Křivánek, '10]

# Recursive Ray Tracing



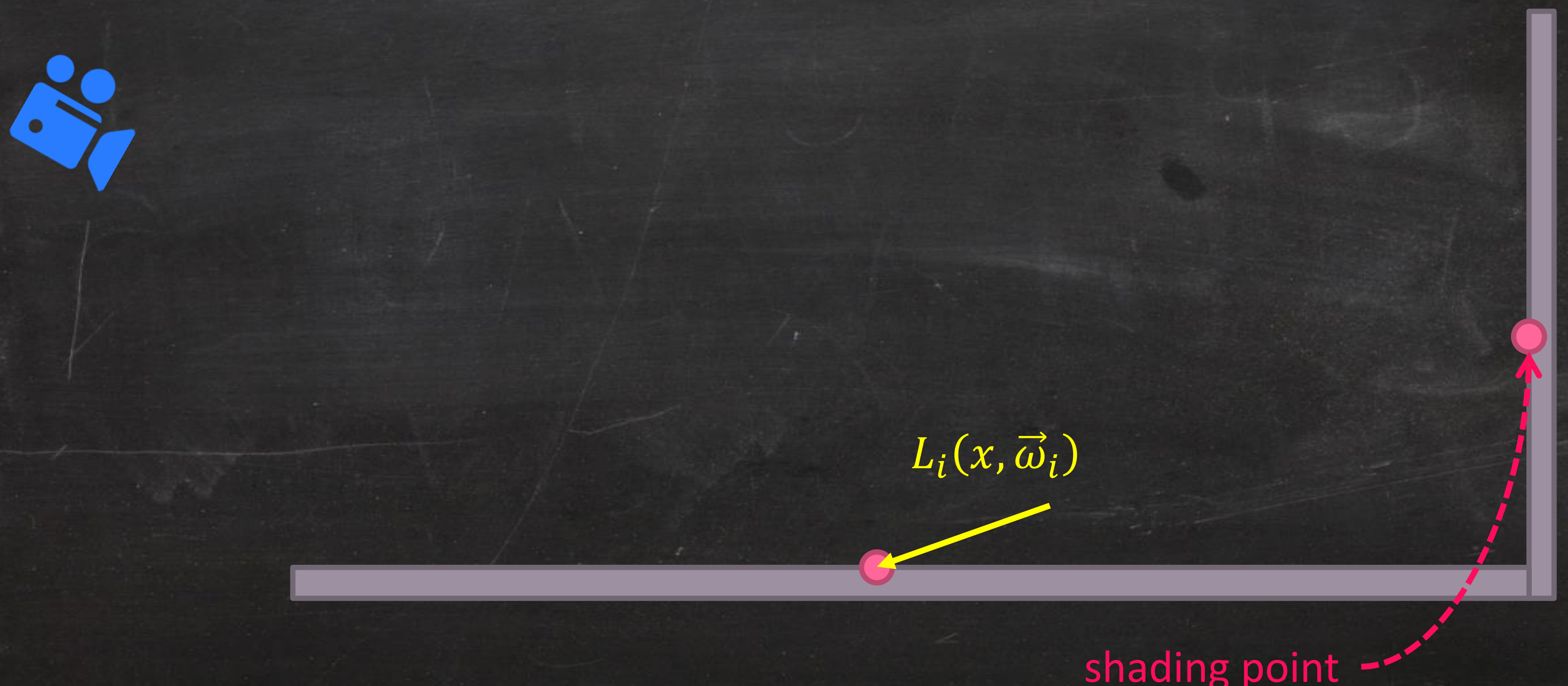

[Inspired by Křivánek, '10]

# Recursive Ray Tracing



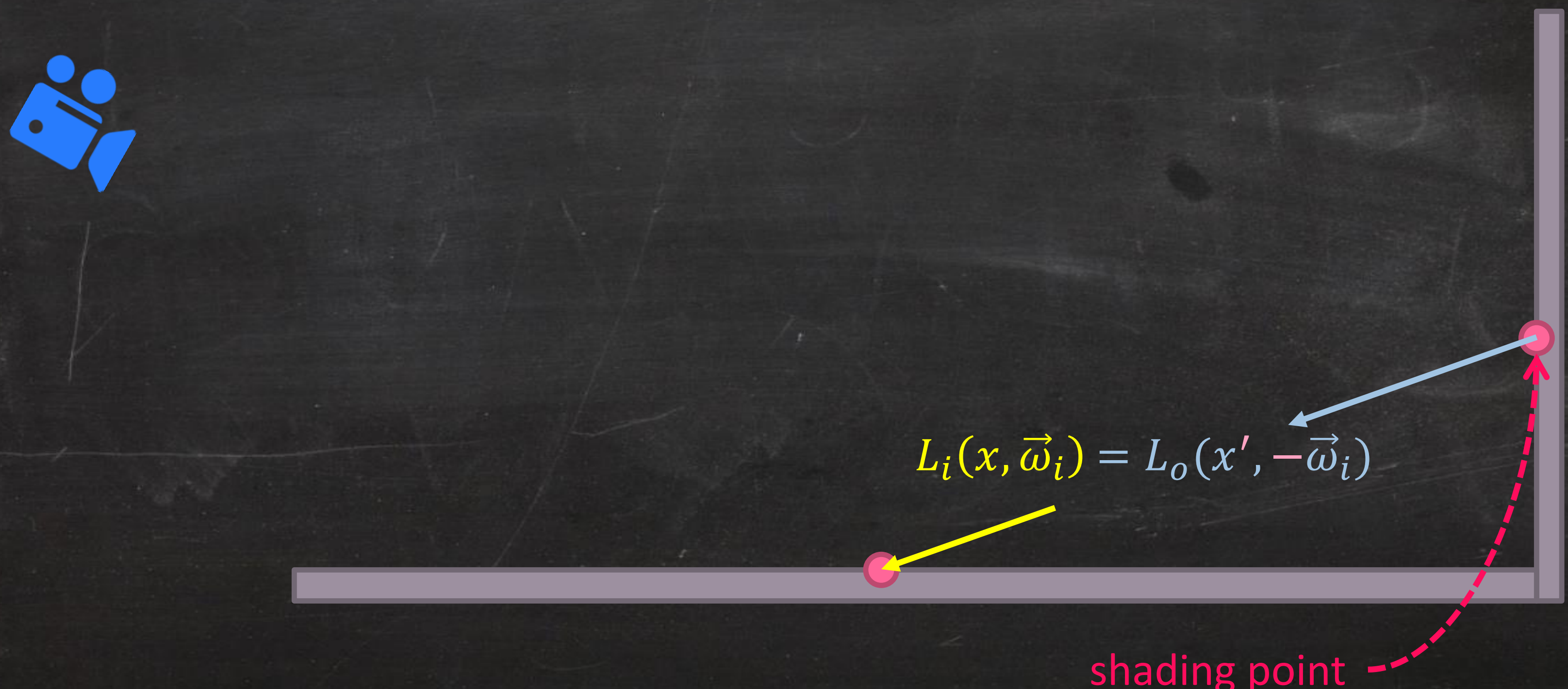

[Inspired by Křivánek, '10]

# Recursive Ray Tracing



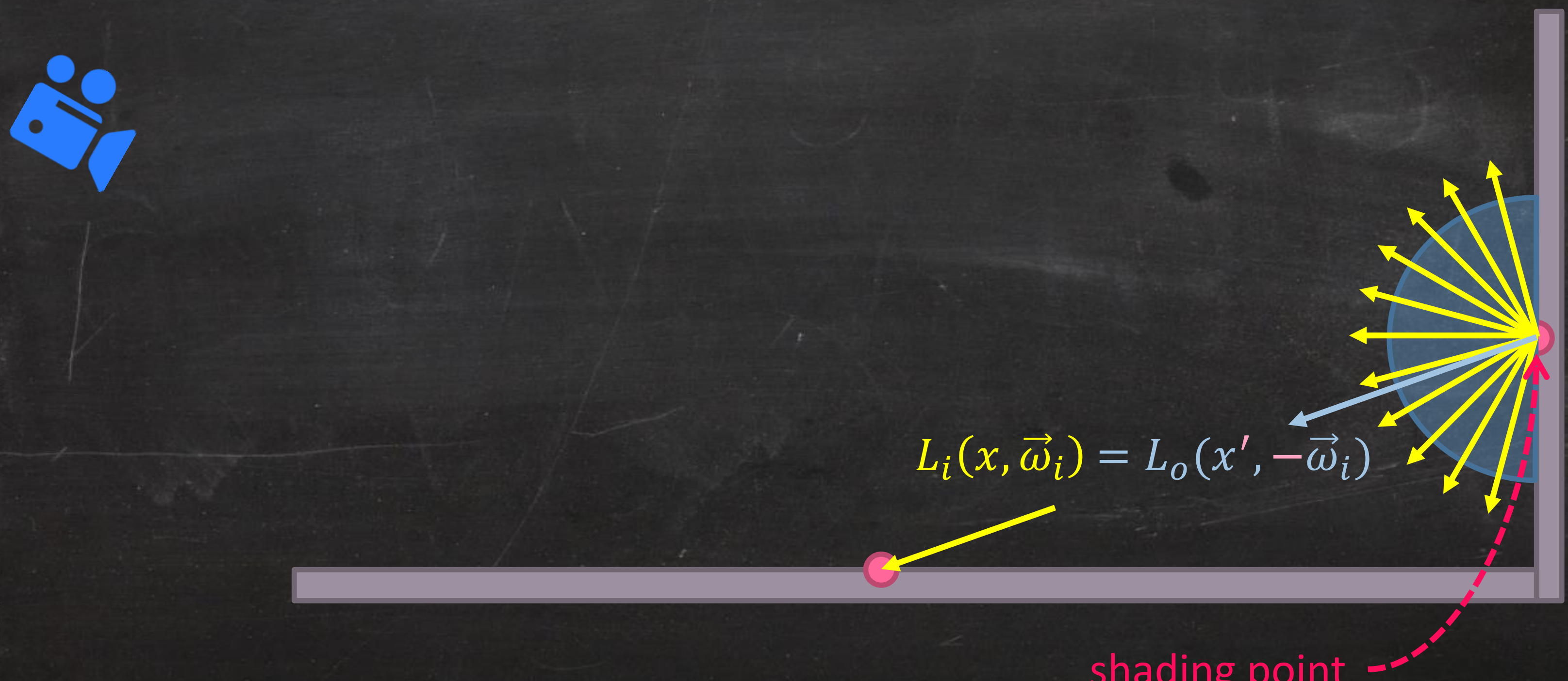

[Inspired by Křivánek, '10]

# Recursive Ray Tracing




[Inspired by Křivánek, ‘10]

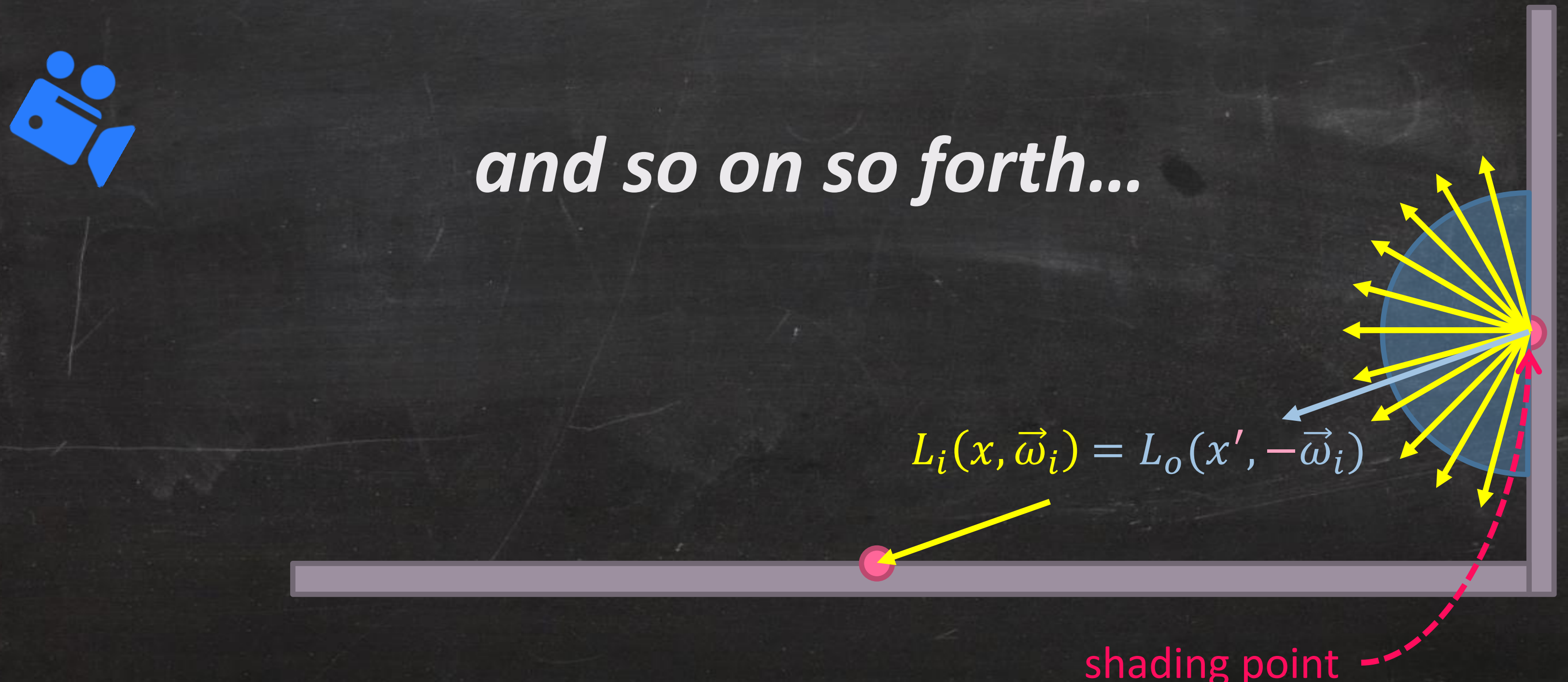
# Recursive Ray Tracing




[Inspired by Křivánek, '10]

# Recursive Ray Tracing

***and so on so forth…***




[Inspired by Křivánek, '10]

# Primary Visibility

**Rasterization**

- Surface to eye
- Visibility via depth buffer

**Ray Tracing**

- Eye to surface
- Visibility via ray casting

# Ray-Casting

- Find the nearest intersection from a ray
- Computed with different geometry representations
  - Explicit
    - Triangular meshes
    - Bezier curves for hair/fur
  - Implicit
    - Volume data (voxels)
    - Point cloud

# Acceleration Structures for Ray-Casting

# Ray-Casting Computation

```
for each ray in each pixel:
    for each geometry primitive in the scene:
        if intersect(ray, primitive):
            return closest point
```

# Ray-Casting Computation

```
for each ray in each pixel:
    for each geometry primitive in the scene:
        if intersect(ray, primitive):
            return closest point
```

$O(N)$ → spatial coherence → $O(\log N)$

# Spatial Coherence

- Geometry primitives only occupy a small portion of the ambient space
- Primitives can be ordered by their spatial locations
- A location in space is associated with a limited number of primitives

***Then, how should we do …***

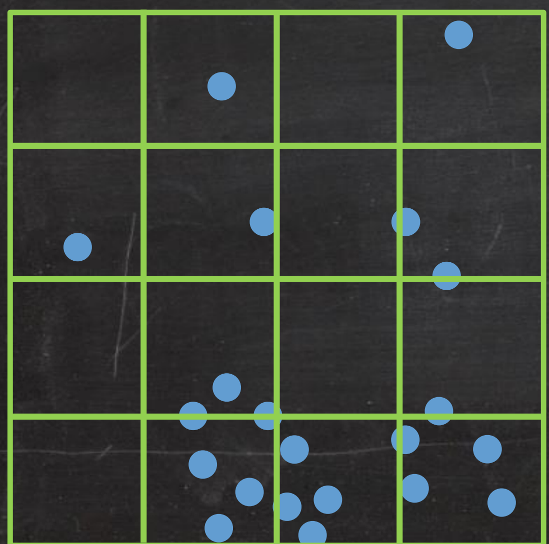
Photo by Amy Treasure

# Acceleration Structures

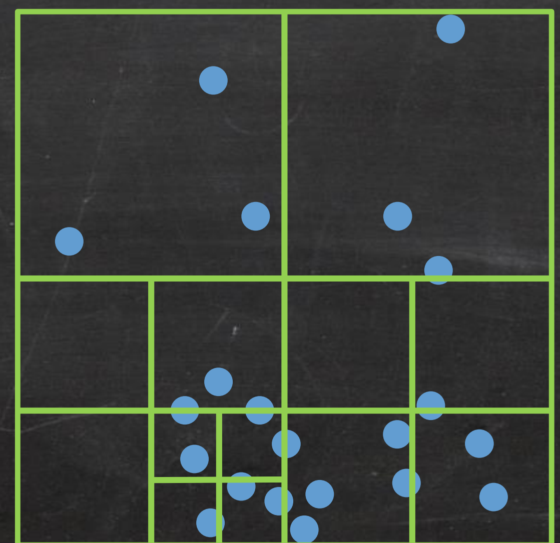
- Uniform grids
- Quadtree/Octree
- k-D tree
- BSP (Binary Space Partitioning) tree
- Bounding volume hierarchy (BVH)

# Spatial Partition

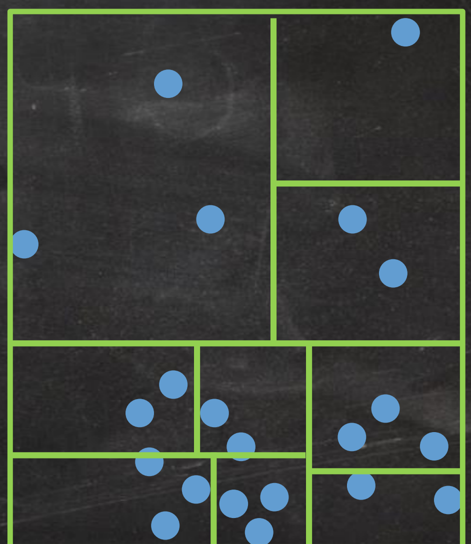
**Uniform Grids**



**Quad Tree**

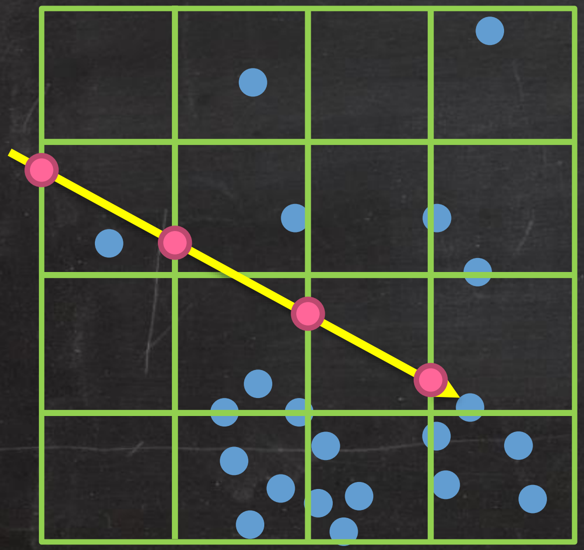


**k-D Tree**

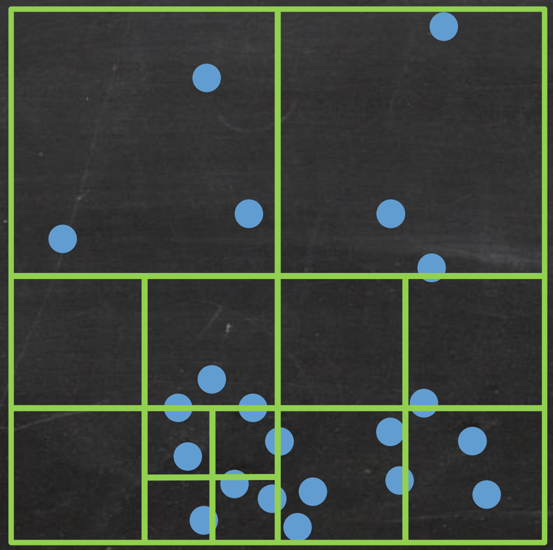


# Spatial Partition

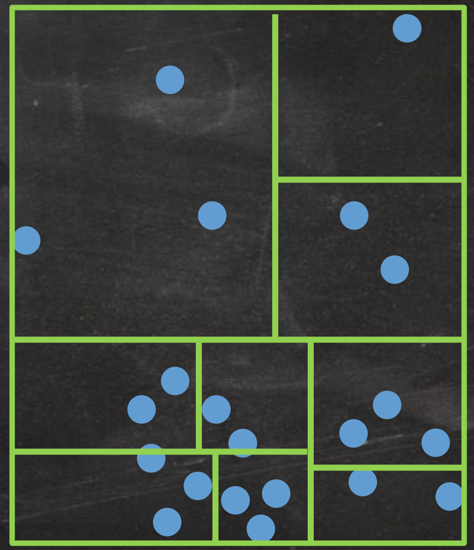
**Uniform Grids**



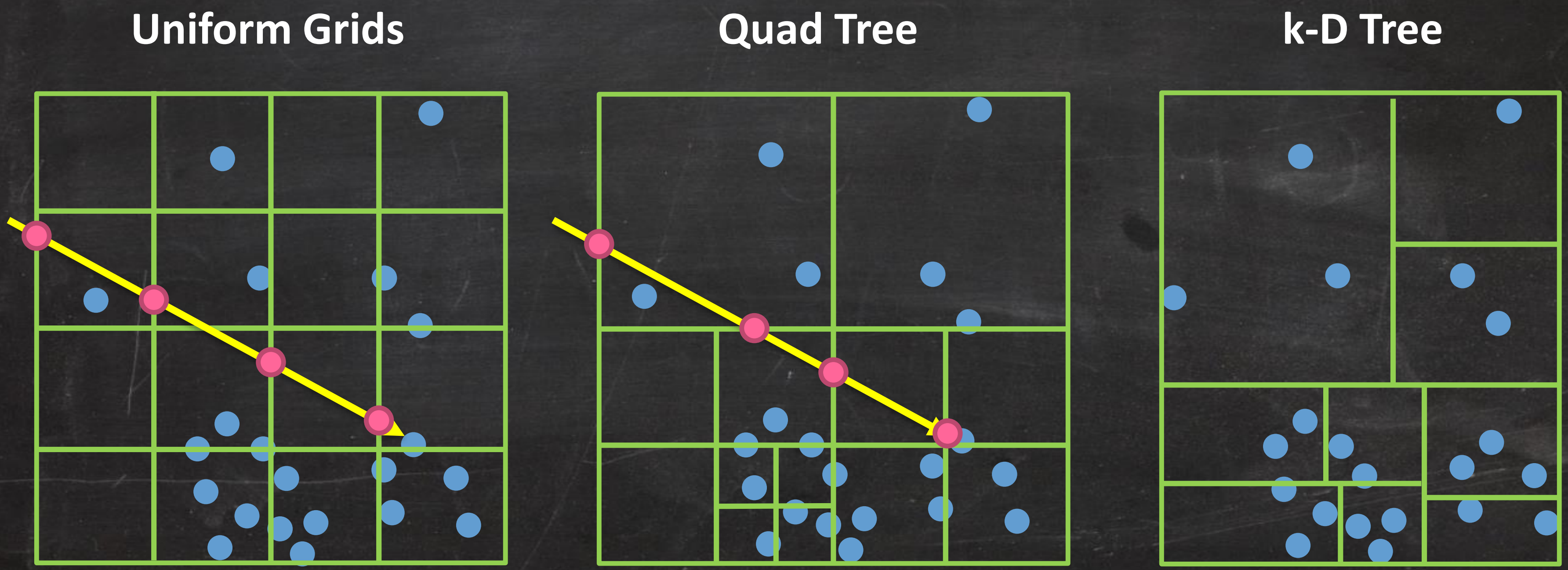
**Quad Tree**



**k-D Tree**

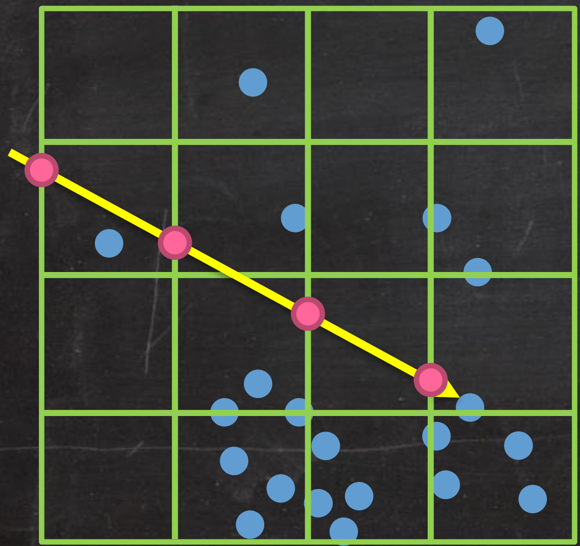


# Spatial Partition

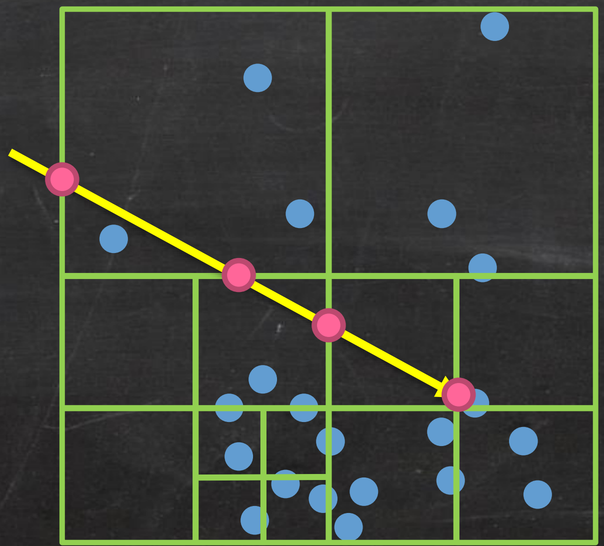


# Spatial Partition

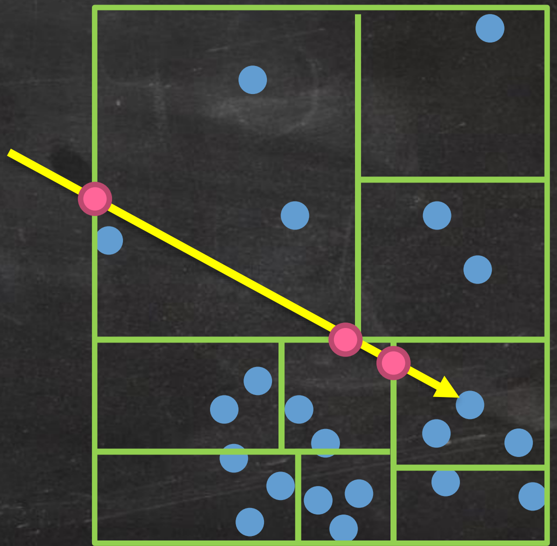
**Uniform Grids**



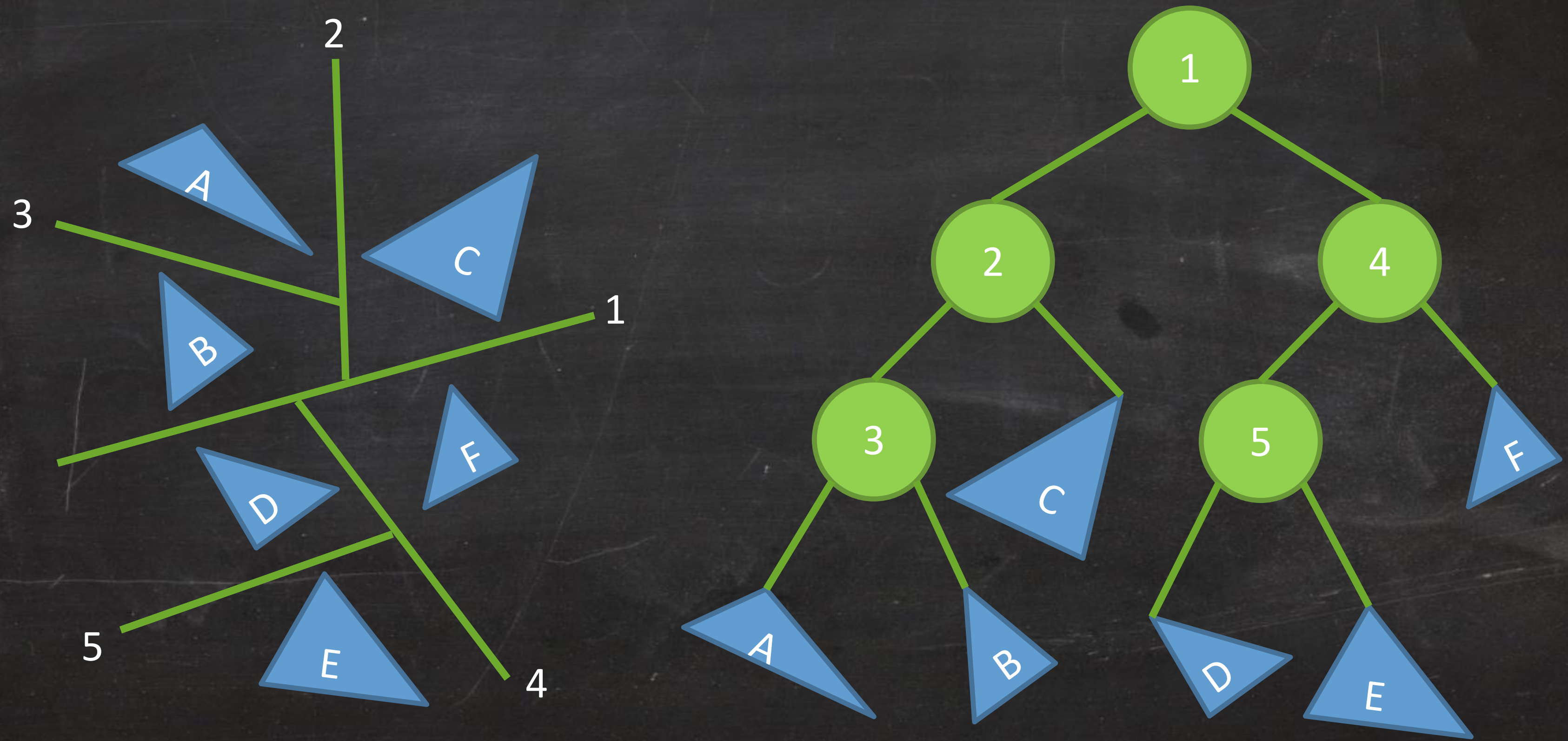
**Quad Tree**



**k-D Tree**

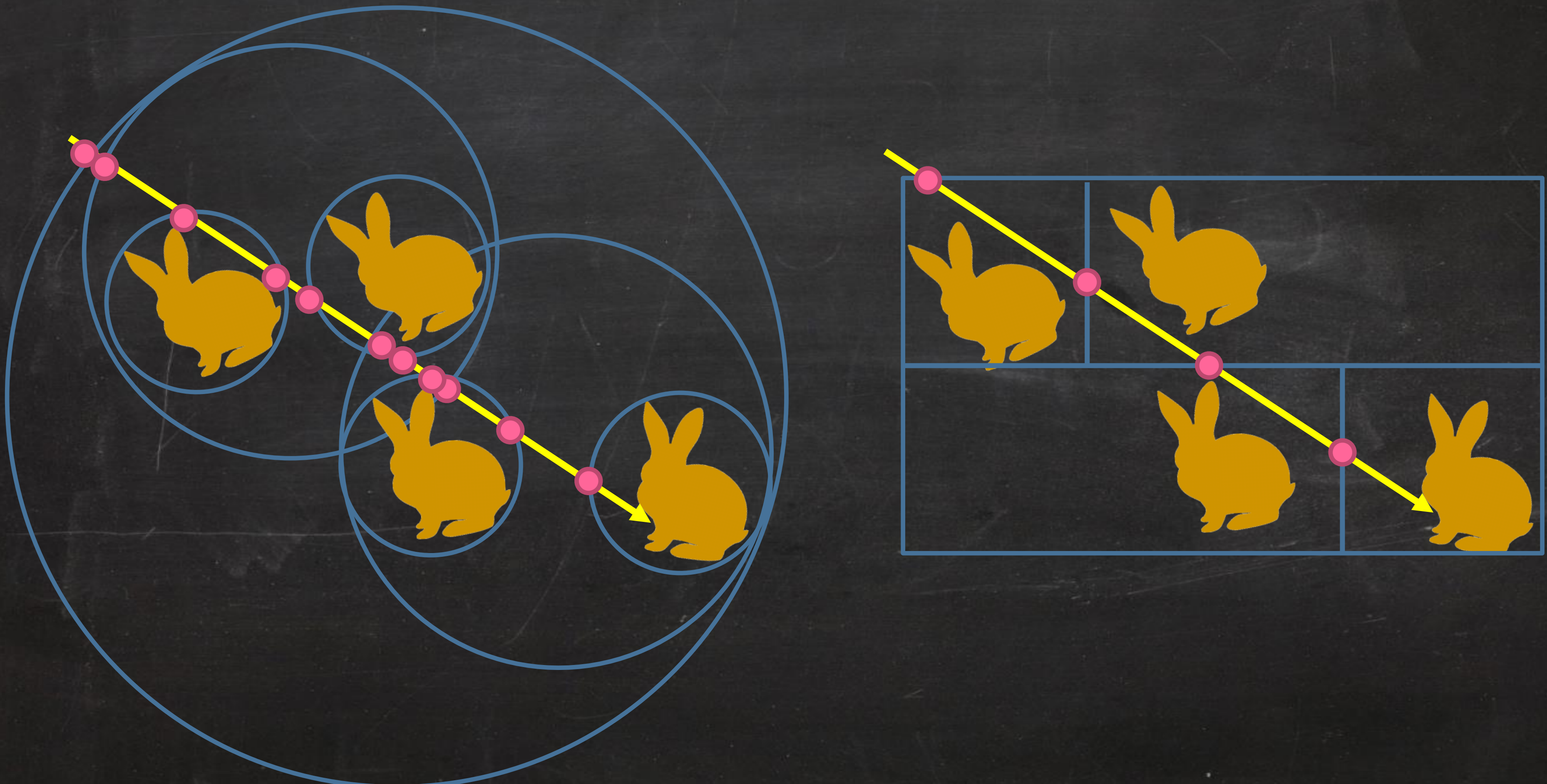


# Binary Space Partitioning Tree



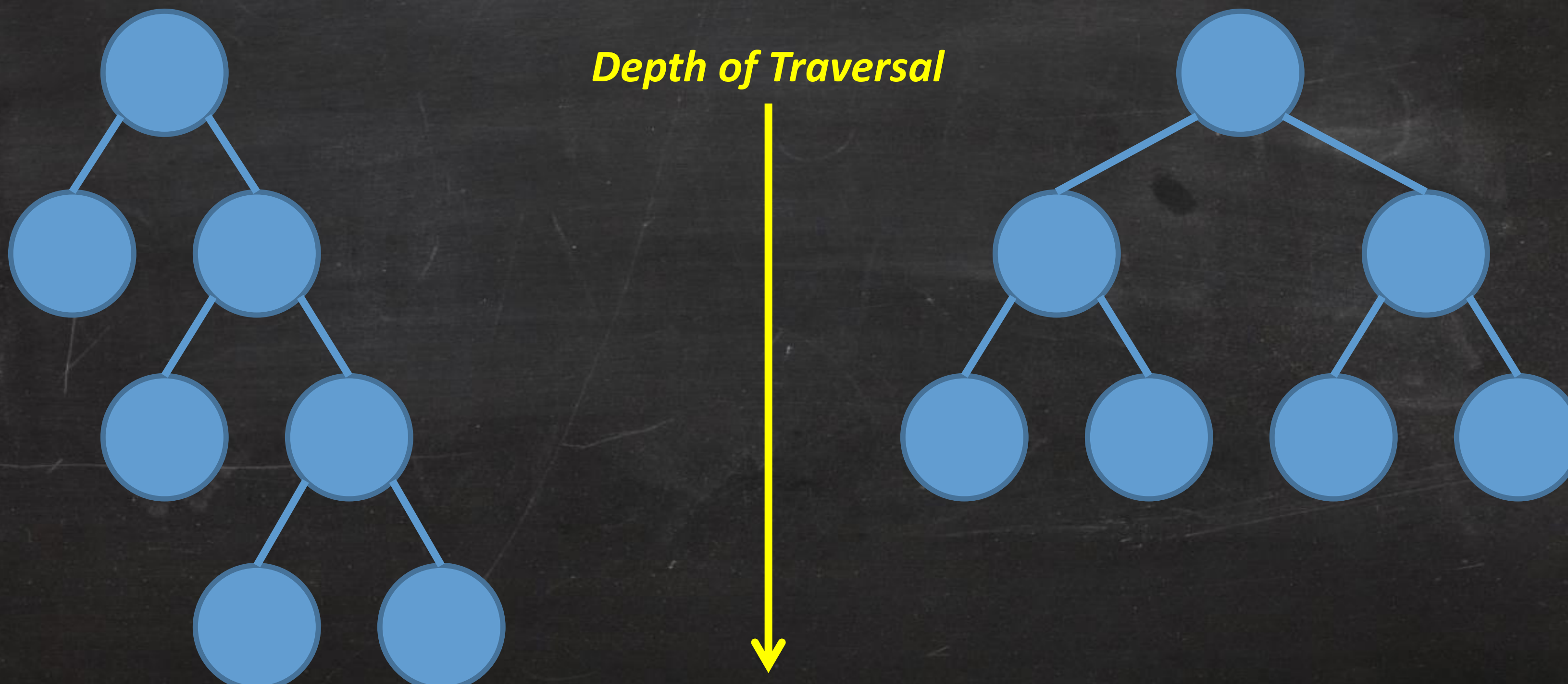
# Types of Boundary Volumes


# Hierarchy Traversal

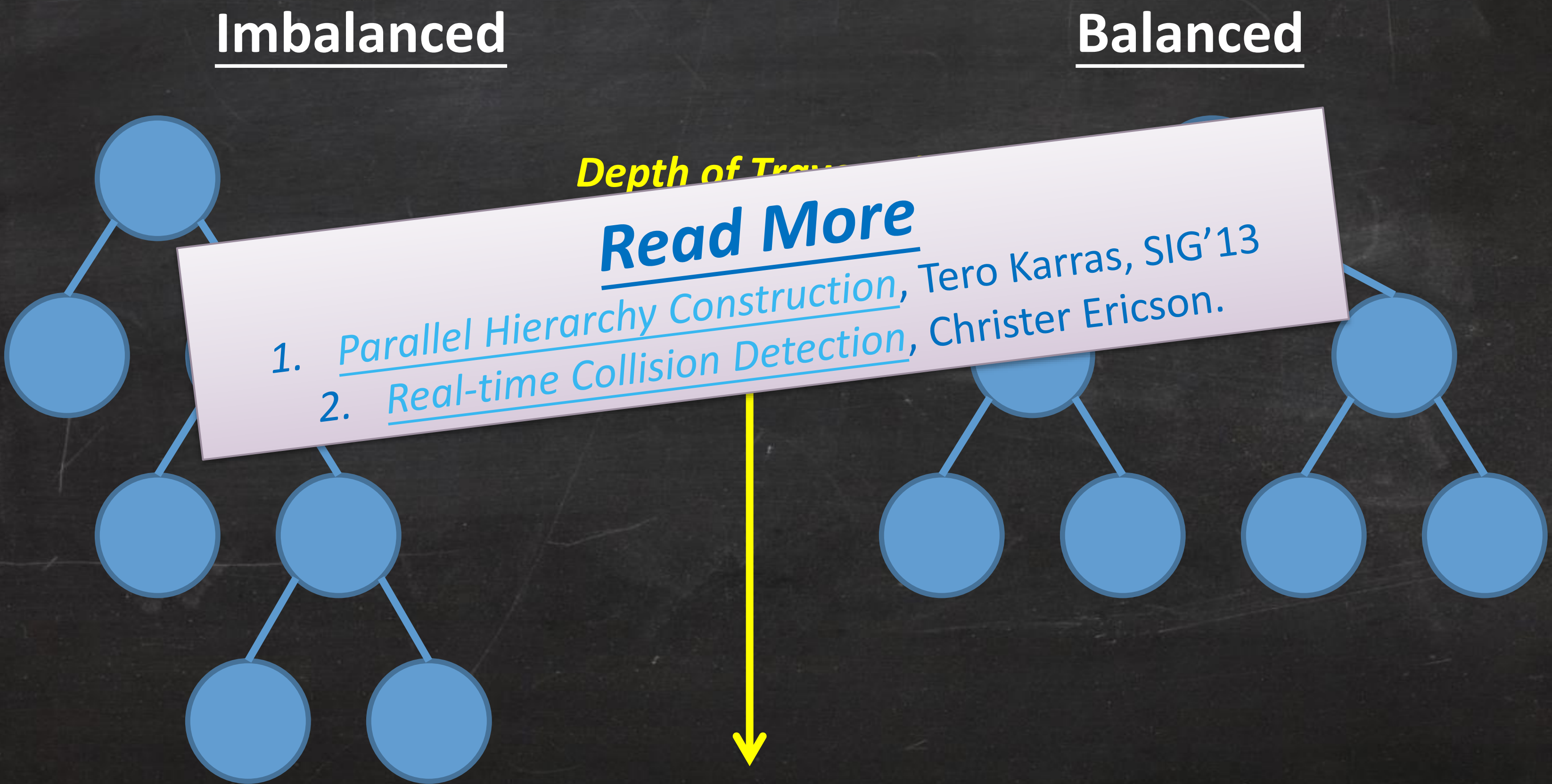


# Balance = Query Performance

**Imbalanced** **Balanced**



# Balance = Query Performance



# Ray-Object Intersection

**Real-Time Rendering**

http://www.realtimerendering.com/intersections.html

Blog | Book Information | Graphics Books | **Intersections** | Portal | Resources

## Object/Object Intersection

Last changed: February 19, 2016

This page gives a grid of intersection routines for various popular objects, pointing to resources in books and on the web. For a unified static and dynamic object intersection and distance library (non-commercial use only, though), see the TGS collis system. The most comprehensive books on the subject are *Geometric Tools for Computer Graphics* (GTCG) and *Real-Time Collision Detection* (RTCD); the former is all-encompassing, the latter more approachable and focused.

Guide to source abbreviations:

- **3DG** - *3D Games: Real-time Rendering and Software Technology*, Alan Watt and Fabio Policarpo, Addison-Wesley, 2001.
- **GPG** - *Game Programming Gems*, ed. Mark DeLoura, Charles River Media, 2000.
- **GTCG** - *Geometric Tools for Computer Graphics*, Philip J. Schneider and David H. Eberly, Morgan Kaufmann Publishers, 2002. Good, comprehensive book on this topic.
- **Gems** - The *Graphics Gems* series. See the book's website for individual book links and code.
- **GTweb** - Geometric Tools, Dave Eberly's online computer graphics related software repository. His book *3D Game Engine Design* also covers these, in a readable format, as well as many other object/object intersection tests.
- **IRT** - *An Introduction to Ray Tracing*, ed. Andrew Glassner, Academic Press, 1989.
- **JCGT** - *The Journal of Computer Graphics Techniques*.
- **jgt** - *journal of graphics tools*. A partial code repository is available.
- **RTCD** - *Real-Time Collision Detection*, by Christer Ericson, Morgan Kaufmann Publishers, 2004.
- **RTR** - *Real-Time Rendering, Third Edition*, by Tomas Möller, Eric Haines, and Naty Hoffman, A.K. Peters Ltd., 2008.
- **RTR2** - *Real-Time Rendering, Second Edition*, by Tomas Akenine-Möller and Eric Haines, A.K. Peters Ltd., 2002.
- **SG** - Simple Geometry library, Steve Baker's vector, matrix, and quaternion manipulation library.
- **TGS** - Teikitu Gaming System Collision, Andrew Aye's object/object intersection/distance and sweep/penetration software (non-commercial use only).
- **TVCG** - IEEE Transactions on Visualization and Computer Graphics.

Individual article references follow after the table.

### Static Object Intersections

Entries are listed from oldest to newest, so often the *last* entry is the best. This table covers objects not moving; see the next section for dynamic objects.

| | ray | plane | sphere | cylinder | cone | triangle | AABB | OBB | frustum | polyhedron |
|---|---|---|---|---|---|---|---|---|---|---|
| **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) | 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 | 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; GTCG p.512 | Möller-Trumbore jgt 2(1): code (mirror), paper draft; IRT p.53,102; Gems IV p.24; Held jgt 2(4); GTweb; 3DG p.17; Möller (mirror); GTCG p.485; TGS; RTCD p.153,184; Löfstedt jgt 10(2): code, paper draft Chirkov jgt 10(3): code; Lagae jgt 10(4): code, paper draft; SoftSurfer; RTR2 p.578; RTR3 p.746; Havel TVCG June 2009; Woop JCGT 2(1) | IRT p.65,104; Gems p.395; Smits; 3DG p.20; Terdiman (optimized Woo); Schroeder; GTCG p.626; TGS; RTCD p.179; Mahovsky jgt 9(1); Williams jgt 10(1) (code); Eisemann jgt 12(4) (code); RTR2 p.572; RTR3 p.742; Shirley 2016 | (IRT p.104; Gems II p.247); GTweb; Gomez; GTCG p.630; TGS; RTCD p.179; RTR2 p.572; RTR3 p.743 | (IRT p.104; Gems II p.247) | IRT p.104; Gems II p.247; GTCG p.493; Platis jgt 8(4); RTCD p.198; SoftSurfer |

# Practical Issues

- Construction costs in space and time
  - Use float for scalar data instead of double
  - Pointers are costly in x64 platform
    - Might point to incontiguous memory location in heap
    - To save storage, try using int32 or short for indexing
  - Geometry compression?
    - Unit vector quantization
    - Store local position in float, only use double for their transform matrix
- Parallelism and locality are key factors for parallel processing

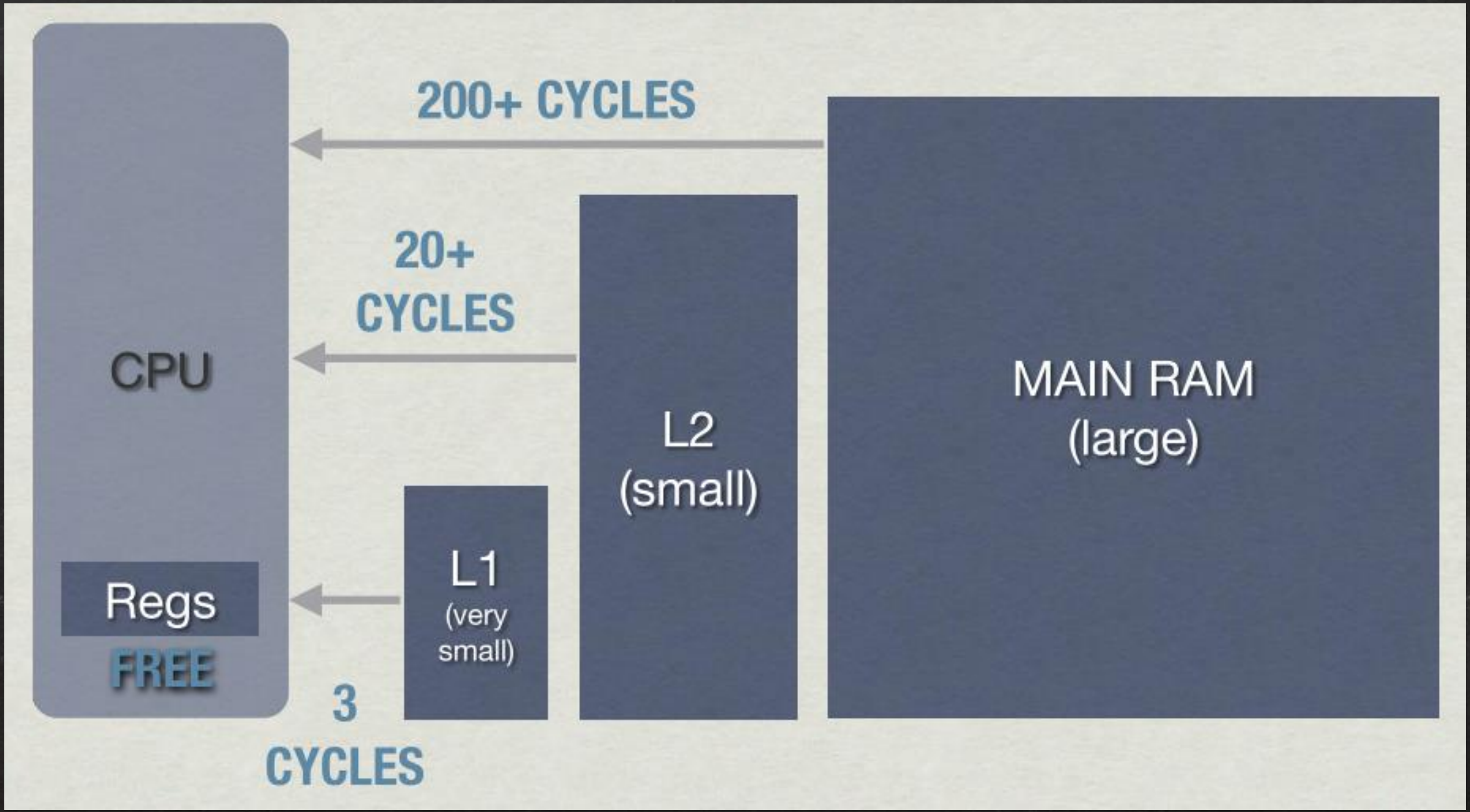
# Practical Issues

- Construction costs in space and time
  - Use float for scalar data instead of double
  - Pointers are costly in x64 platfo
    - Mi
  - 
    - ...it vector quantization
    - Store local position in float, only use double for their transform matrix
- Parallelism and locality are key factors for parallel processing

**Read More**

1. *A Survey of Efficient Representations*, Zina H. Cigolle et al., JCGT'14.
2. *Geometry Compression*, Michael Deering, Computer Graphics '95.

# Memory Caching




[Dogged Determination: Technology and Process at Naughty Dog Inc.](), Jason Gregory. SINFO XXI 2014 Keynote

# Cache Line

- The fixed size data block transferred between memory and cache
- Might take hundreds of clocks to move around
- False sharing
  - Different threads access elements which reside in the same cache line

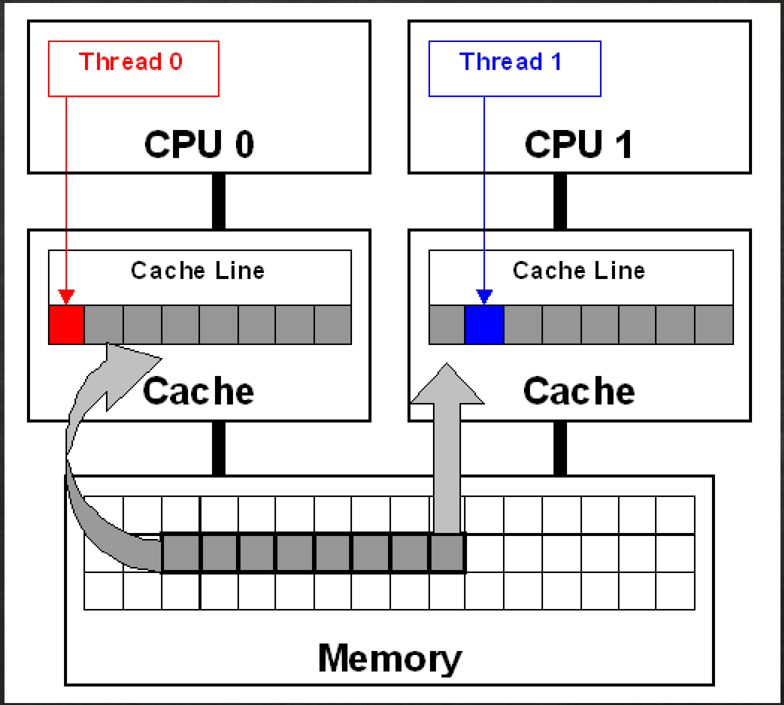
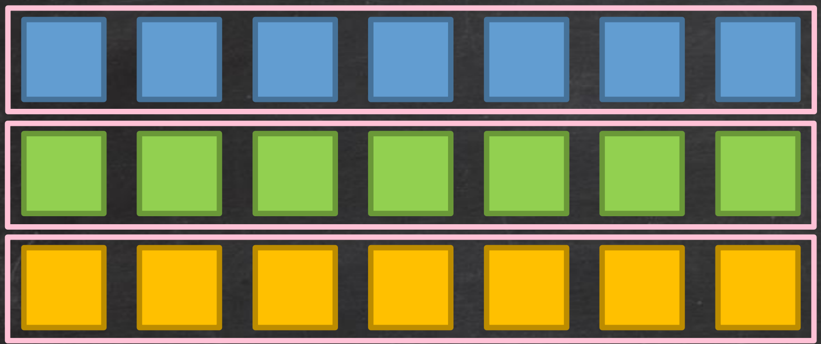



*figure from [https://software.intel.com](https://software.intel.com)*

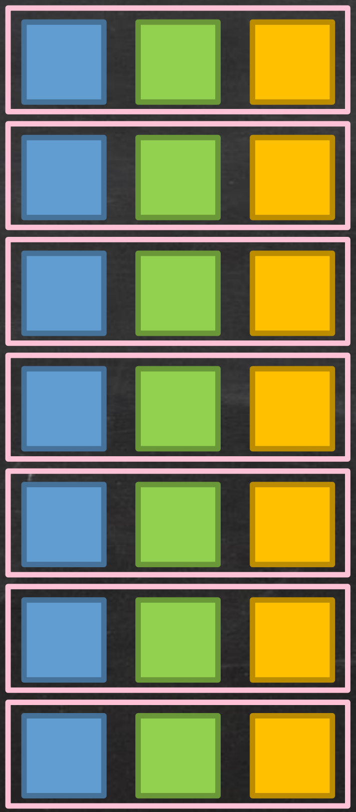
# SOA vs. AOS



***Structures of Arrays (SOA)***

- Easily aligned cache boundaries
- Easier to utilize SIMD
- Chance for hardware prefetching

***Array of Structures (AOS)***



- Intuitively match the object abstraction
- Might cause cache alignment problems
- Hard to vectorized

# SOA vs. AOS

***Structures o***

- Easily alig cache boundaries
- Easier to utilize SIMD
- Chance for hardware prefetching

***Array of Structures (AOS)***

- Intuitively match the object abstraction
  use cache
  t problems
  ectorized

## ***Read More***

1. *CPU Caches and Why You Care*, Scott Meyers.
2. *Cache Aware Components*, Randy Gaul.

# Object-Oriented or Data-Oriented Design?

- Abstraction is good for modeling
  - But over-abstraction is harmful for performance
- Memory access pattern is crucial for parallel processing
  - Structure of arrays (SOA) vs. Array of structures (AOS)
  - Hot/cold splitting
- 80/20 principle
  - Optimizing after profiling!!
  - Don’t optimizing the insignificant parts

# Object-Oriented or Data-Oriented Design?

- Abstraction is good for modeling
  - But over-abstraction is harmful for performance
- Memory access patt... cessing
  - Str...
  - Ho...
- 80/20 principle
  - Optimizing after profiling!!
  - Don't optimizing the insignificant parts

**Read More**

1. Data Oriented Design, Richard Fabian.
2. Data-Oriented Design and C++, Mike Acton, CppCon'14.

# Out-of-Core Algorithms

- What if the data are too large to fit the main memory?
  - Conventional algorithm doesn't work!
  - Reduce the times of data reading as many as possible
    - Avoid rewinding all data elements
  - Dice one computation task into several sub-tasks
    - Need to estimate the memory consumption for each sub-task
  - Apply the concept of 'paging'
    - Use memory mapped file during computation