CS380: Computer Graphics

Distributed Ray Tracing

Sung-Eui Yoon
(윤성의)

Course URL:
http://sgvr.kaist.ac.kr/~sungeui/CG/

KAIST

Class Objectives

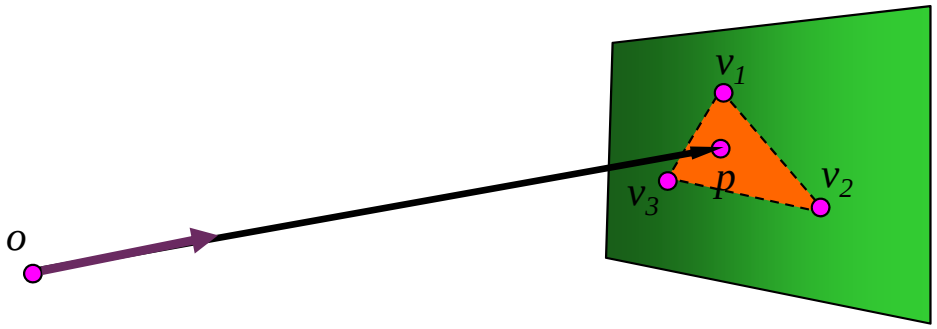
- **Support various effects based on distributed ray tracing**
  - **Acceleration methods**
  - **Random sampling, jittering, in each pixel**
- **At the last time:**
  - **Ray generations of ray tracing**
  - **Intersection tests w/ implicit equations**

Questions

- **it seems ray tracing simulates reflection and refraction of light. However, there are also other properties of light, like diffraction or interference. Is there any technique simulating those properties?**

Generalizing to Triangles

- Find of the point of intersection on the plane containing the triangle
- Determine if the point is inside the triangle
  - Barycentric coordinate method
  - Many other methods

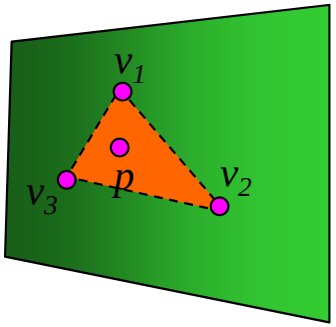


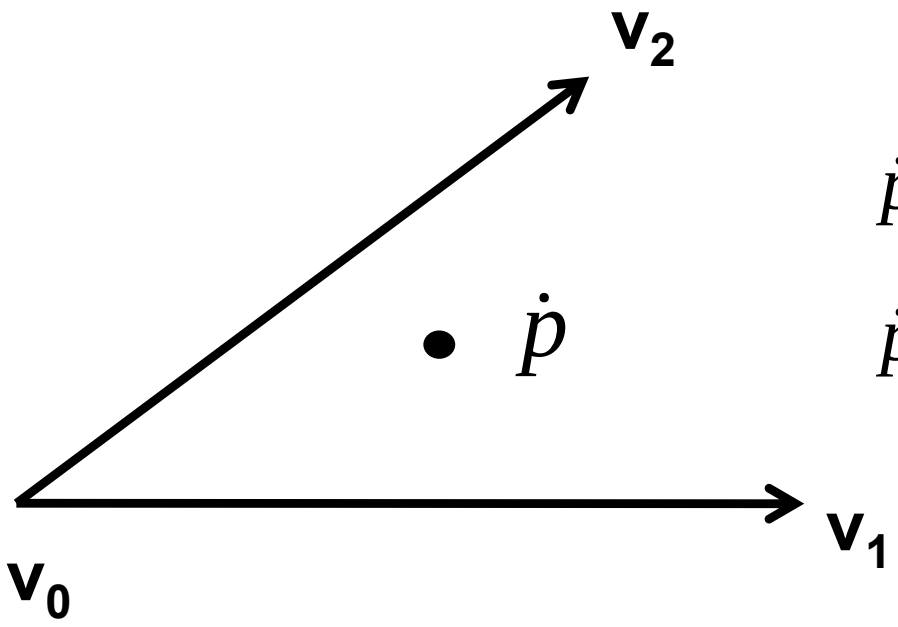
KAIST

Barycentric Coordinates

- **Points in a triangle have positive barycentric coordinates:**

$$\dot{p} = \alpha\dot{v}_0 + \beta\dot{v}_1 + \gamma\dot{v}_2 \quad \text{,where} \quad \alpha + \beta + \gamma = 1$$





$$\dot{p} = \dot{v}_0 + \beta(\dot{v}_1 - \dot{v}_0) + \gamma(\dot{v}_2 - \dot{v}_0)$$

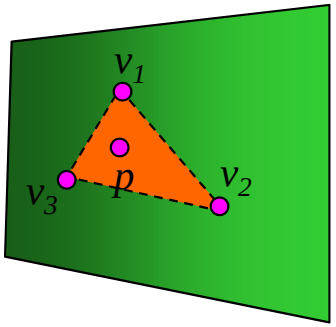
$$\dot{p} = (1 - \beta - \gamma)\dot{v}_0 + \beta\dot{v}_1 + \gamma\dot{v}_2$$

KAIST

Barycentric Coordinates

- **Points in a triangle have positive barycentric coordinates:**

$$\dot{p} = \alpha \dot{v}_0 + \beta \dot{v}_1 + \gamma \dot{v}_2 \quad \text{,where} \quad \alpha + \beta + \gamma = 1$$



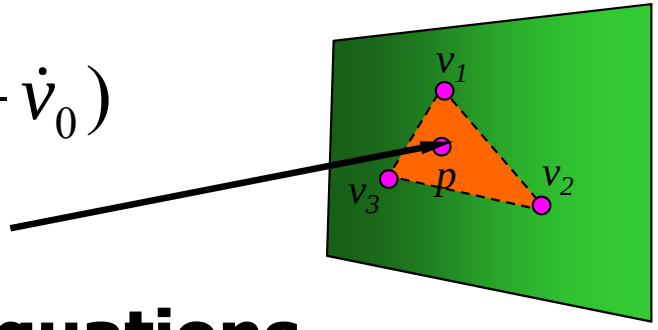
- **Benefits:**
  - **Barycentric coordinates can be used for interpolating vertex parameters (e.g., normals, colors, texture coordinates, etc)**

KAIST

Ray-Triangle Intersection

- **A point in a ray intersects with a triangle**

$$\dot{p}(t) = \dot{v}_0 + \beta(\dot{v}_1 - \dot{v}_0) + \gamma(\dot{v}_2 - \dot{v}_0)$$

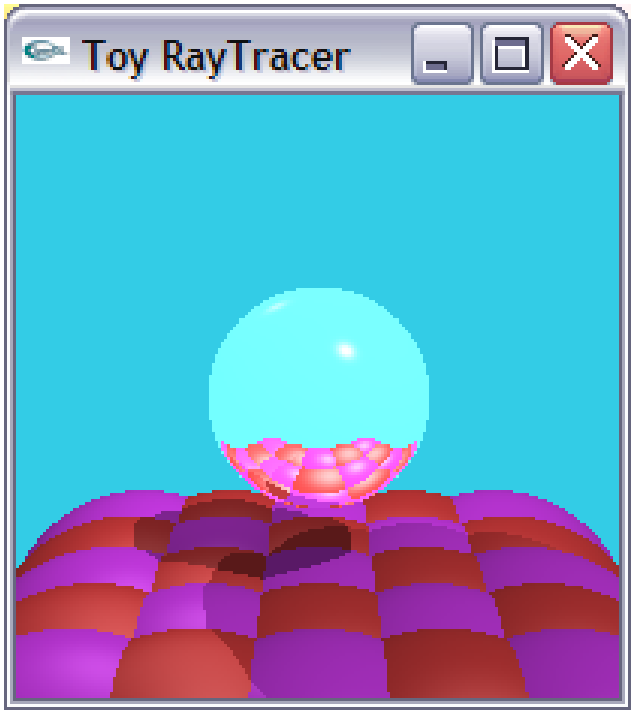


  - **Three unknowns, but three equations**
  - **Compute the point based on t**
  - **Then, check whether the point is on the triangle**

Pros and Cons of Ray Tracing

Advantages of Ray Tracing:

- **Very simple design**
- **Improved realism over the graphics pipeline**



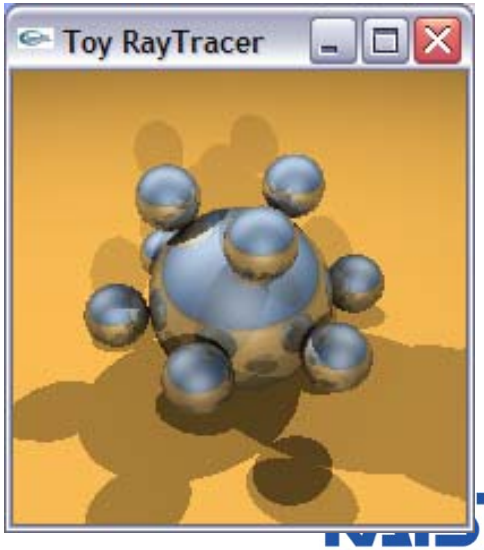
Disadvantages:

- **Very slow per pixel calculations**
- **Only approximates full global illumination**
- **Hard to accelerate with special-purpose H/W**

KAIST

Acceleration Methods

- Rendering time for a ray tracer depends on the number of ray intersection tests per pixel
  - The number of pixels X the number of primitives in the scene
- Early efforts focused on accelerating the ray-object intersection tests
- More advanced methods required to make ray tracing practical
  - Bounding volume hierarchies
  - Spatial subdivision



Bounding Volumes

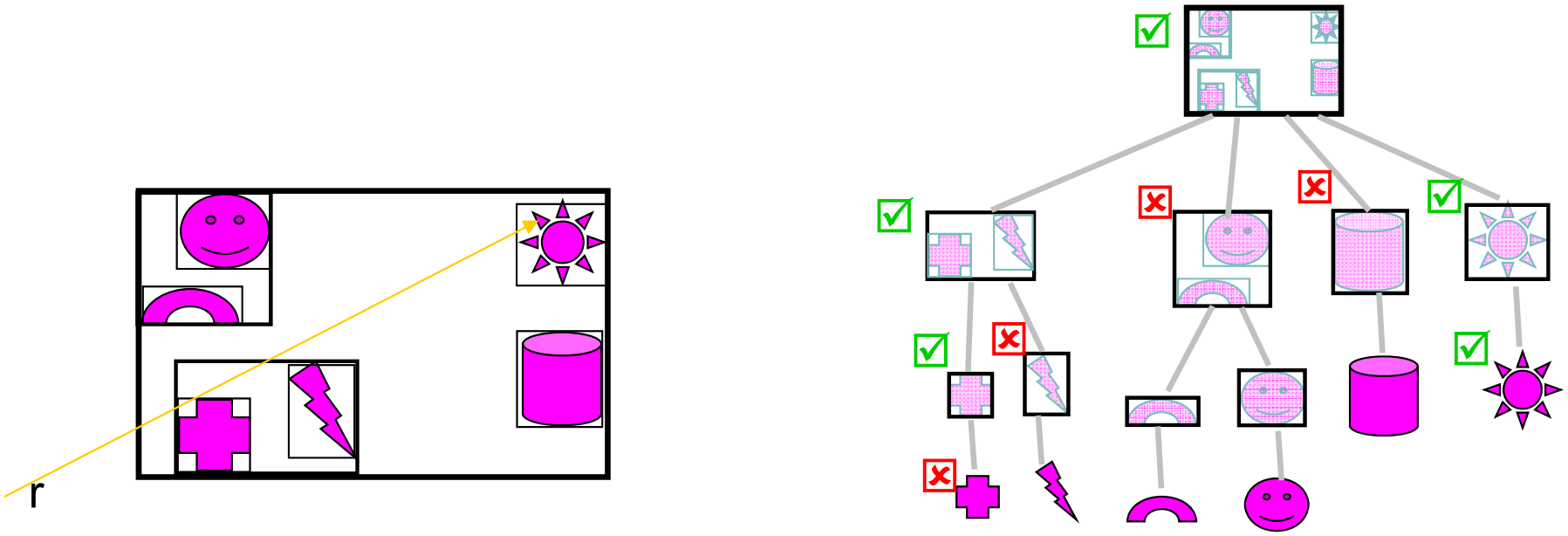
- **Enclose complex objects within a simple-to-intersect objects**
  - **If the ray does not intersect the simple object then its contents can be ignored**
  - **The likelihood that it will strike the object depends on how tightly the volume surrounds the object.**

Potentially tighter fit,
but with higher computation

KAIST

Hierarchical Bounding Volumes

- Organize bounding volumes as a tree
- Each ray starts with the root BV of the tree and traverses down through the tree

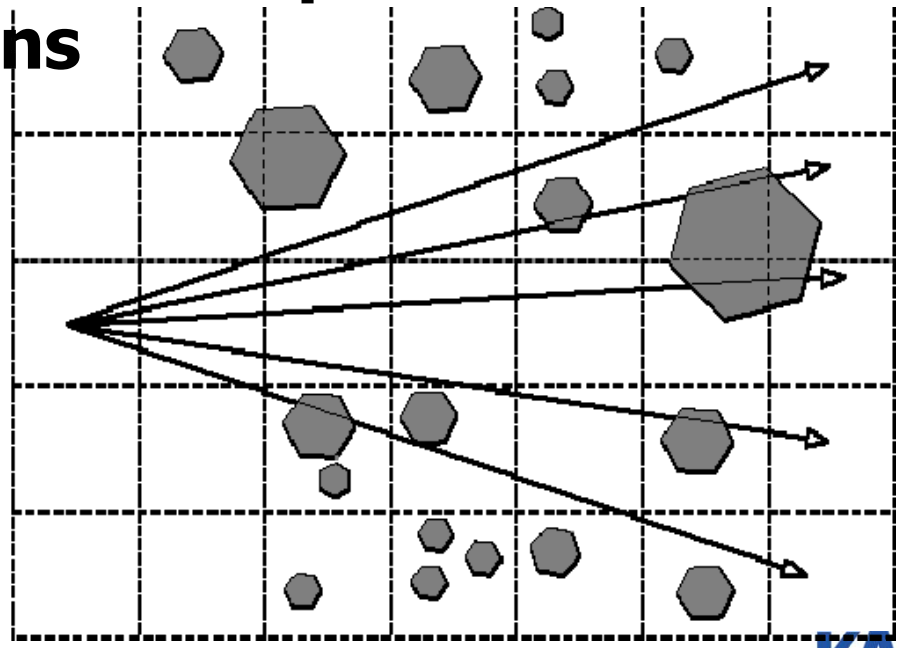


KAIST

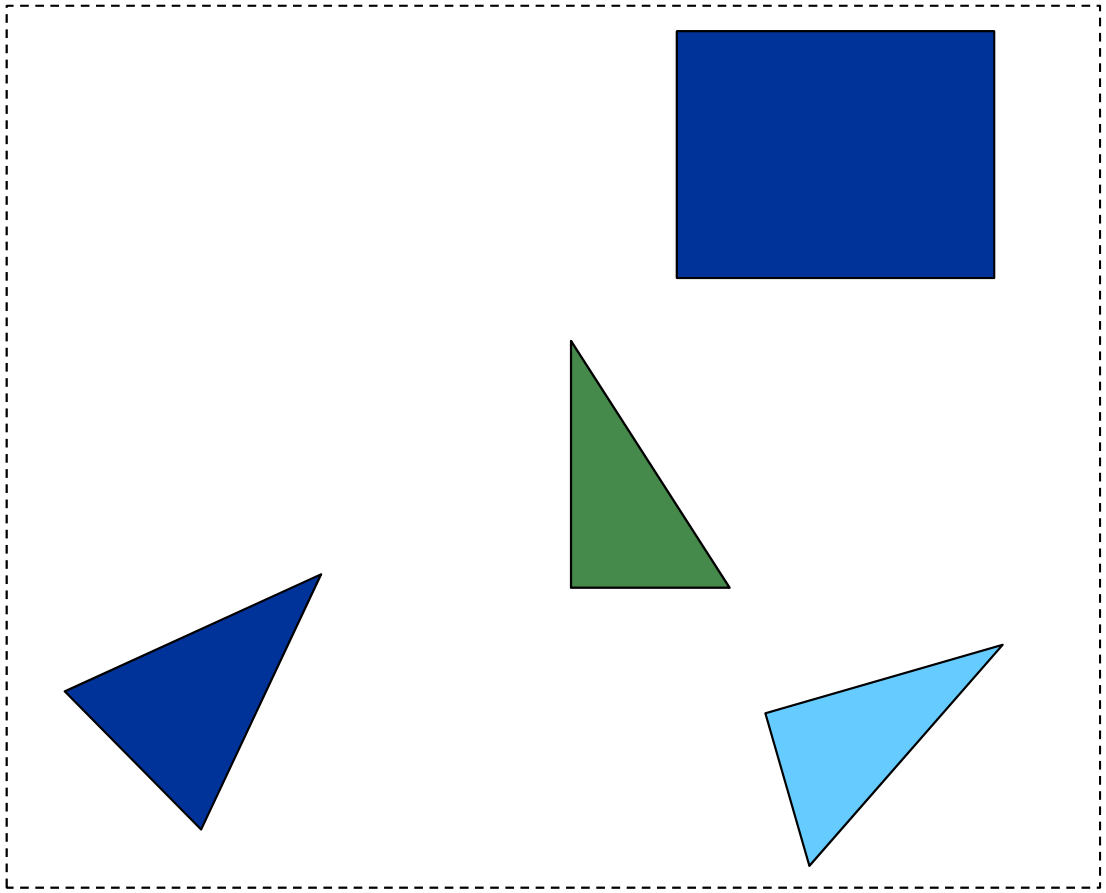
Spatial Subdivision

Idea: Divide space in to subregions

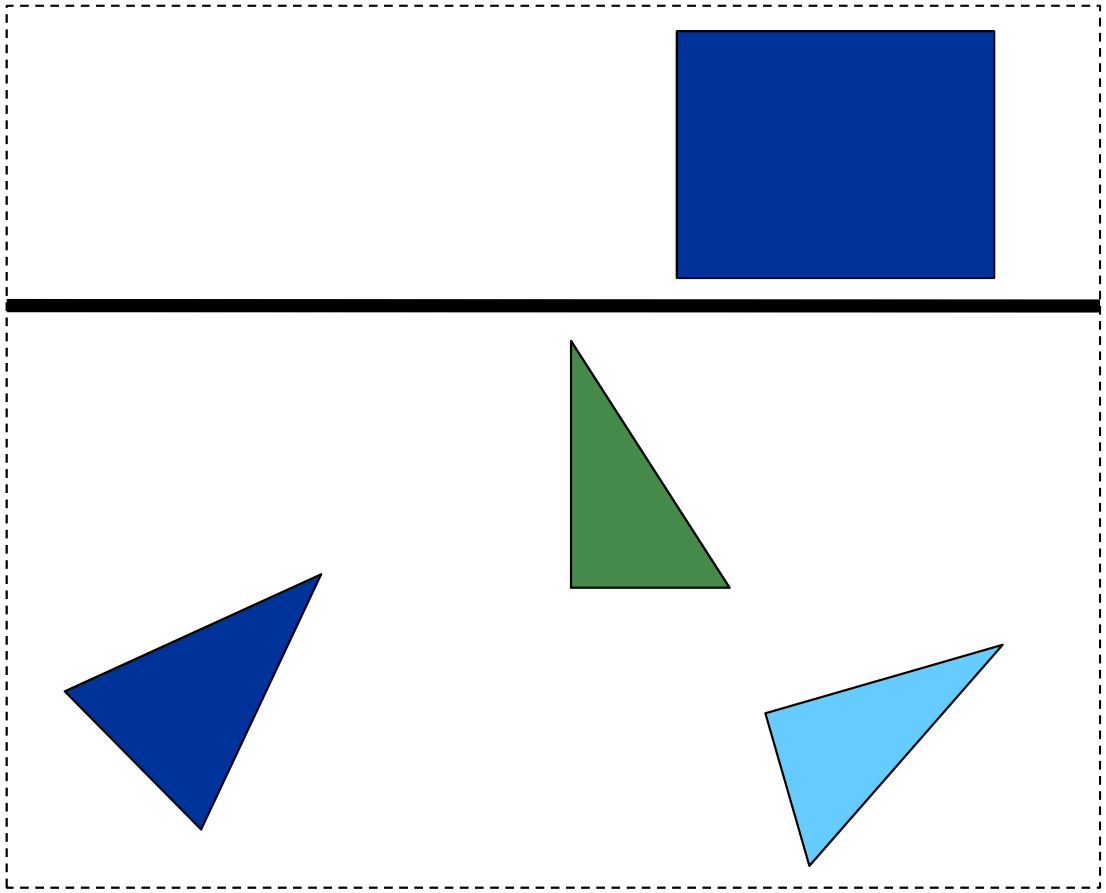
- **Place objects within a subregion into a list**
- **Only traverse the lists of subregions that the ray passes through**
- **"Mailboxing" used to avoid multiple test with objects in multiple regions**
- **Many types**
  - **Regular grid**
  - **Octree**
  - **BSP tree**
  - **kd-tree**

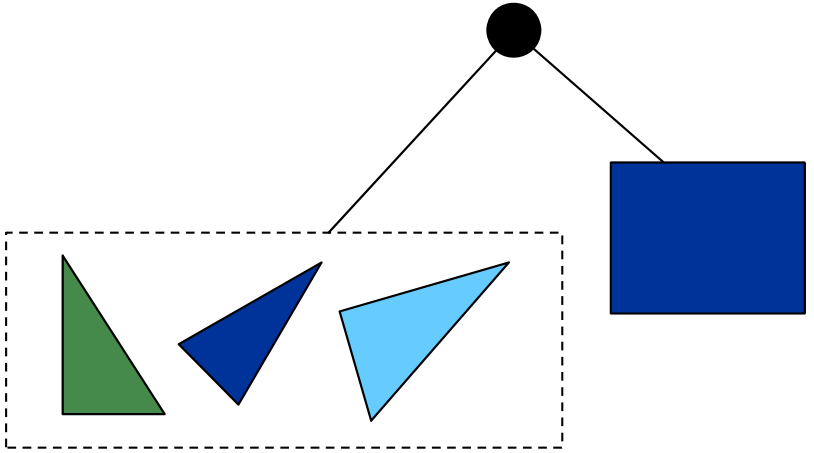


Kd-tree: Example



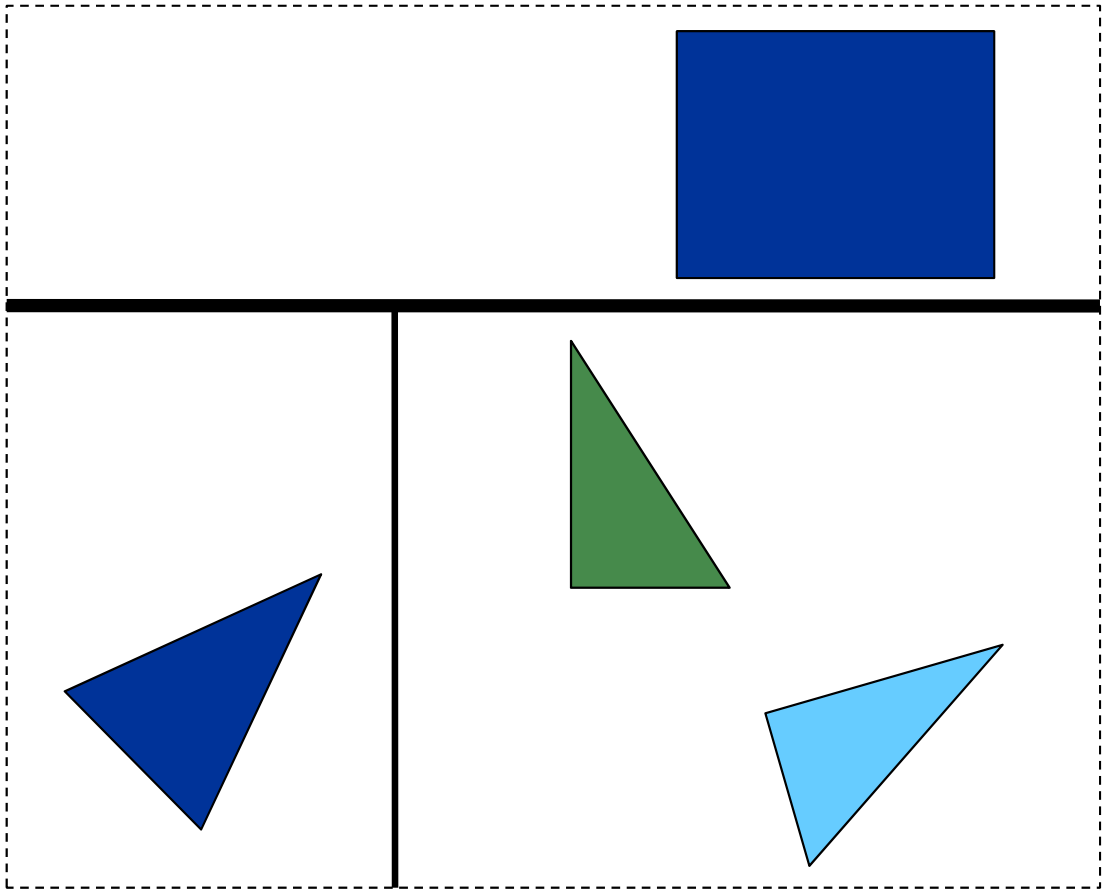
Kd-tree: Example

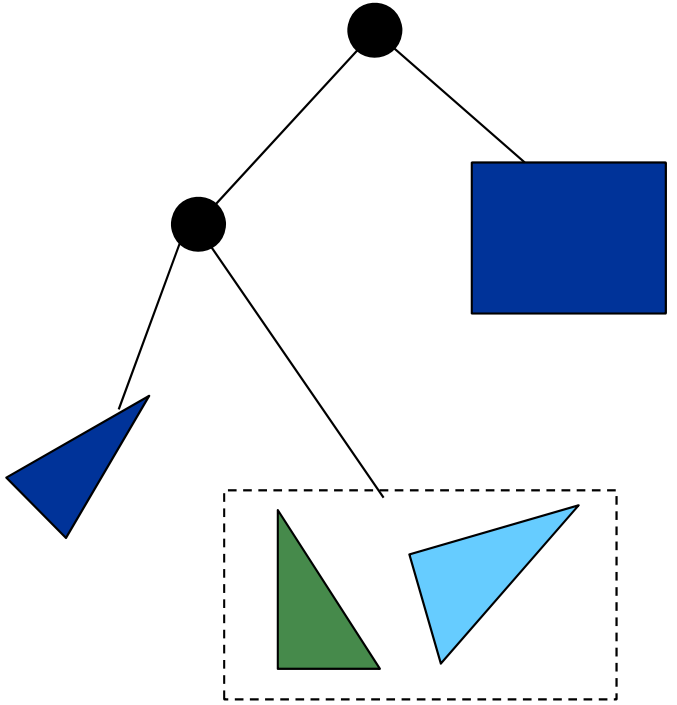




KAIST

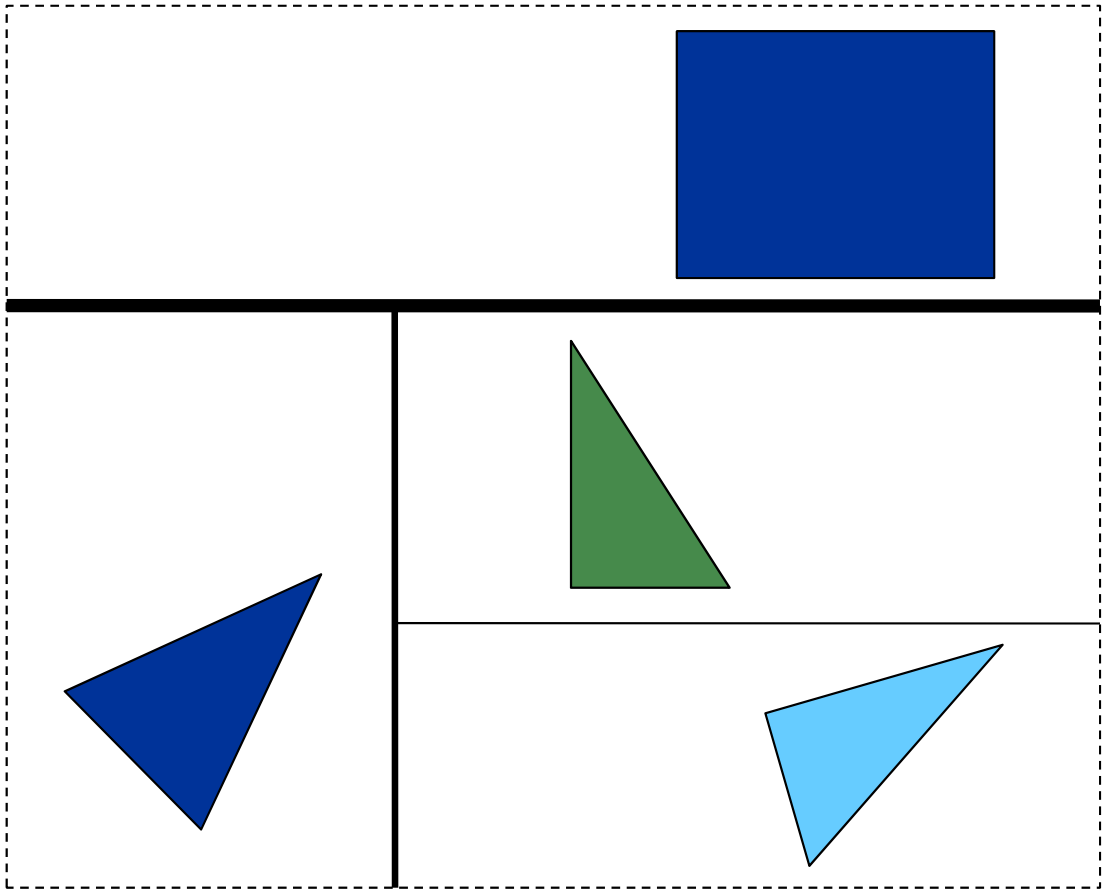
Kd-tree: Example

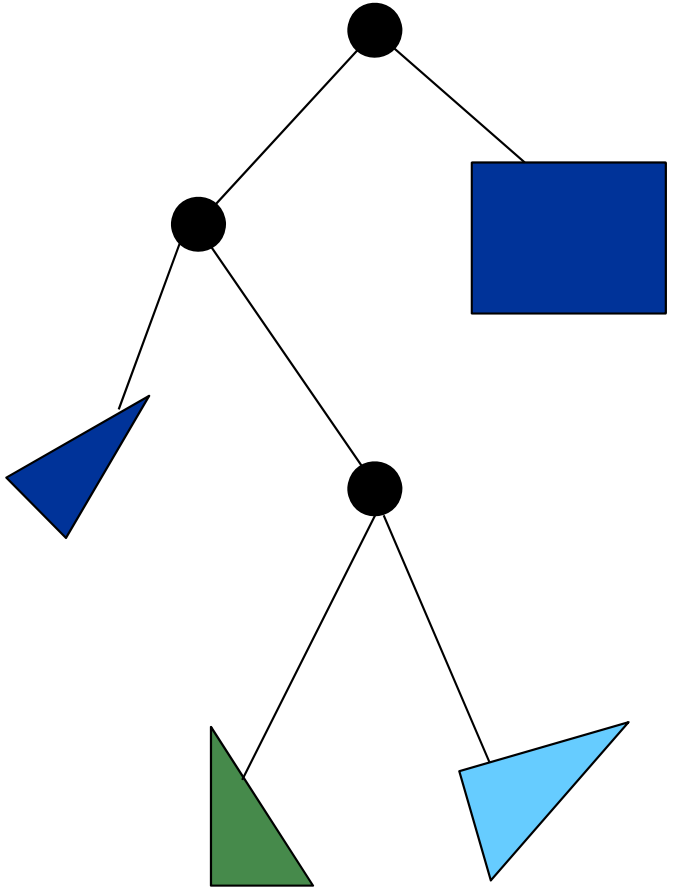




KAIST

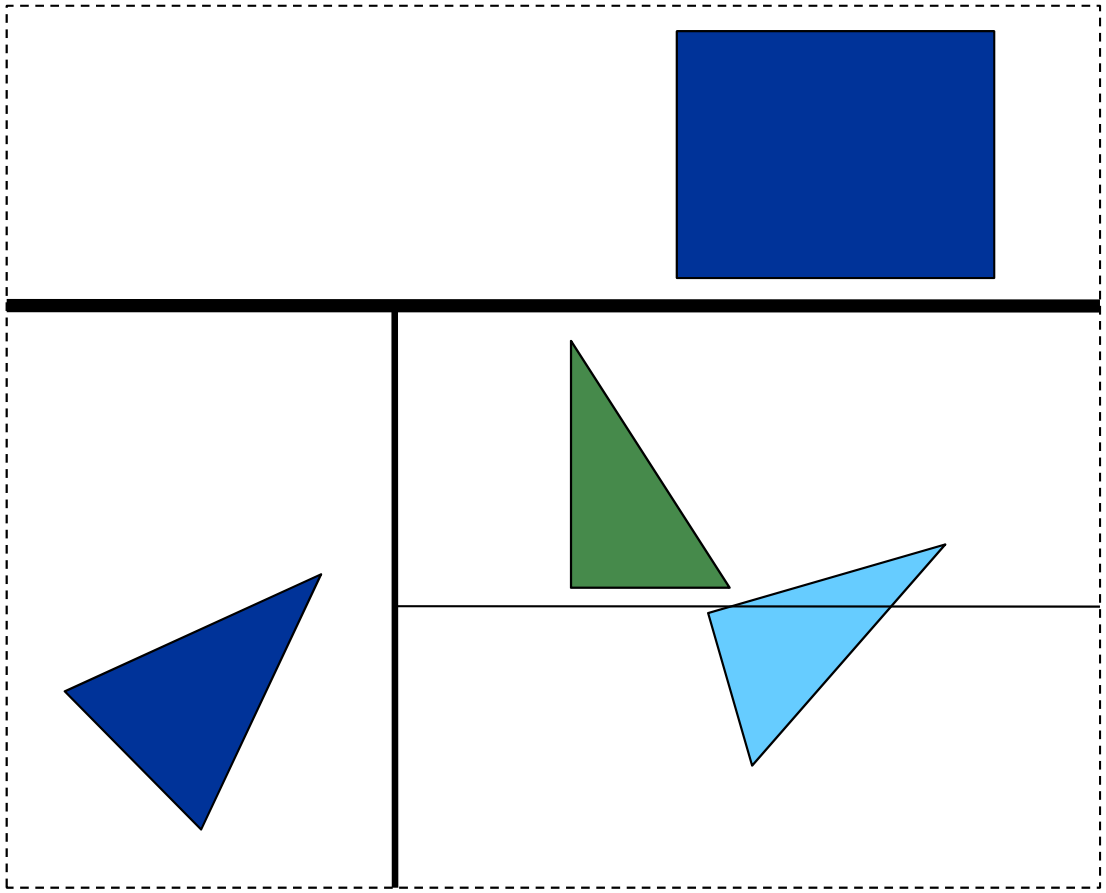
Example





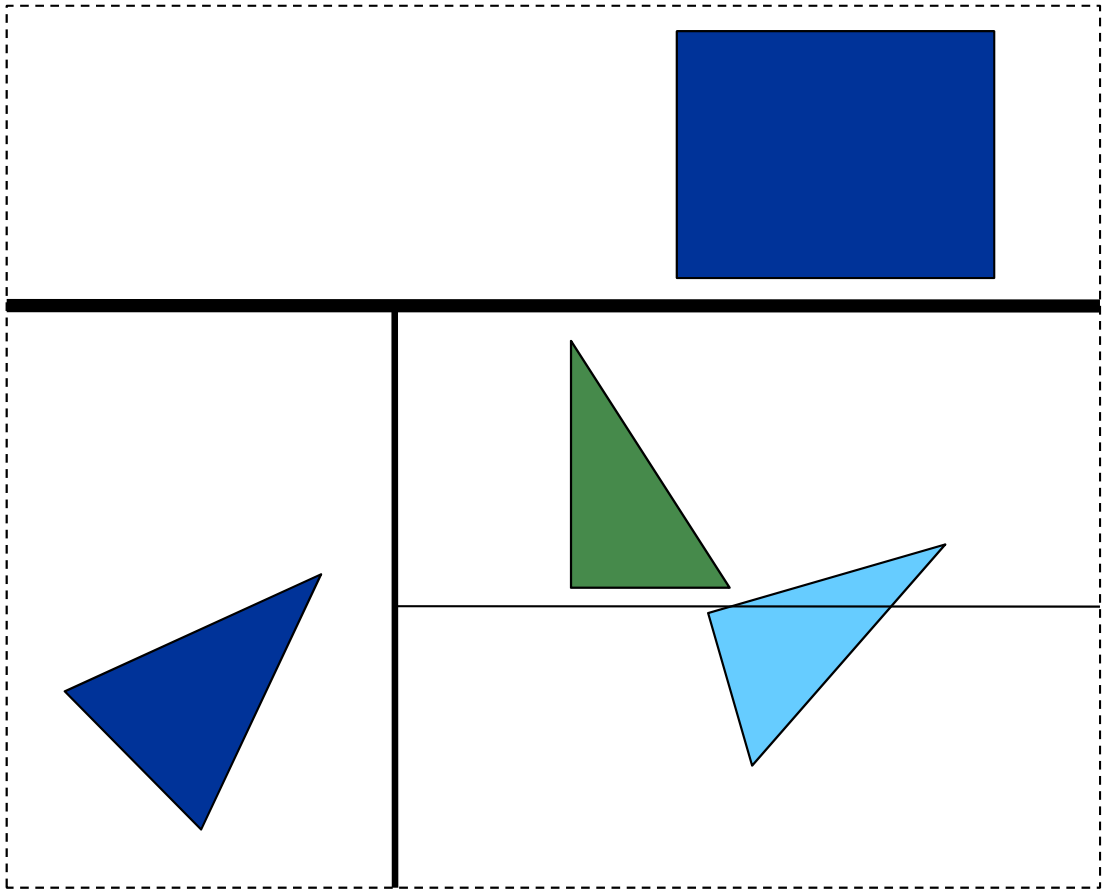
KAIST

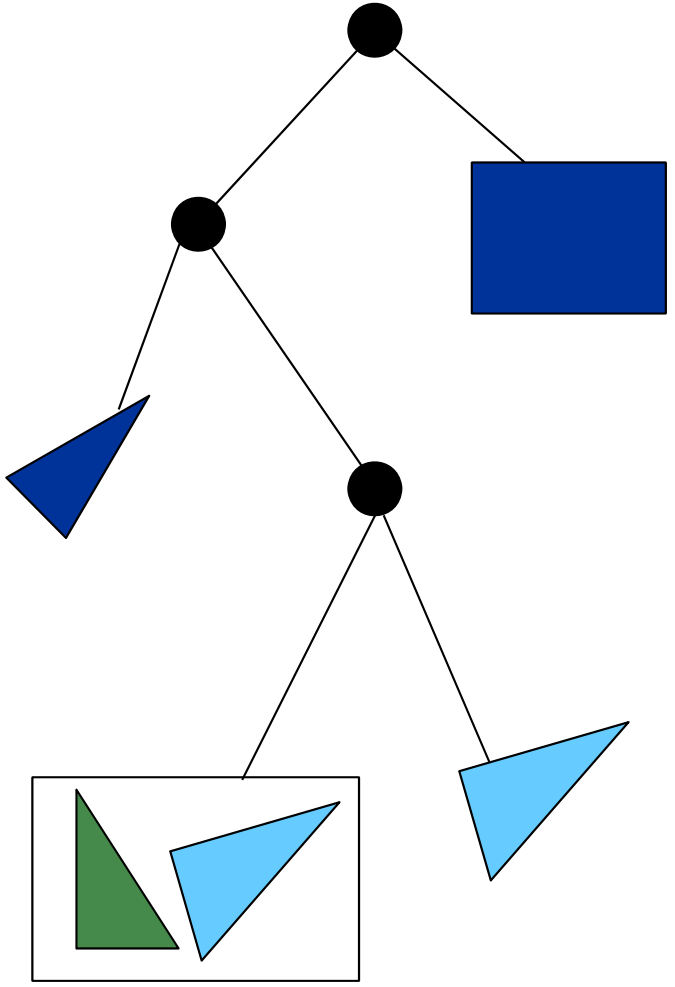
Kd-tree: Example



What about triangles overlapping the split?

Kd-tree: Example





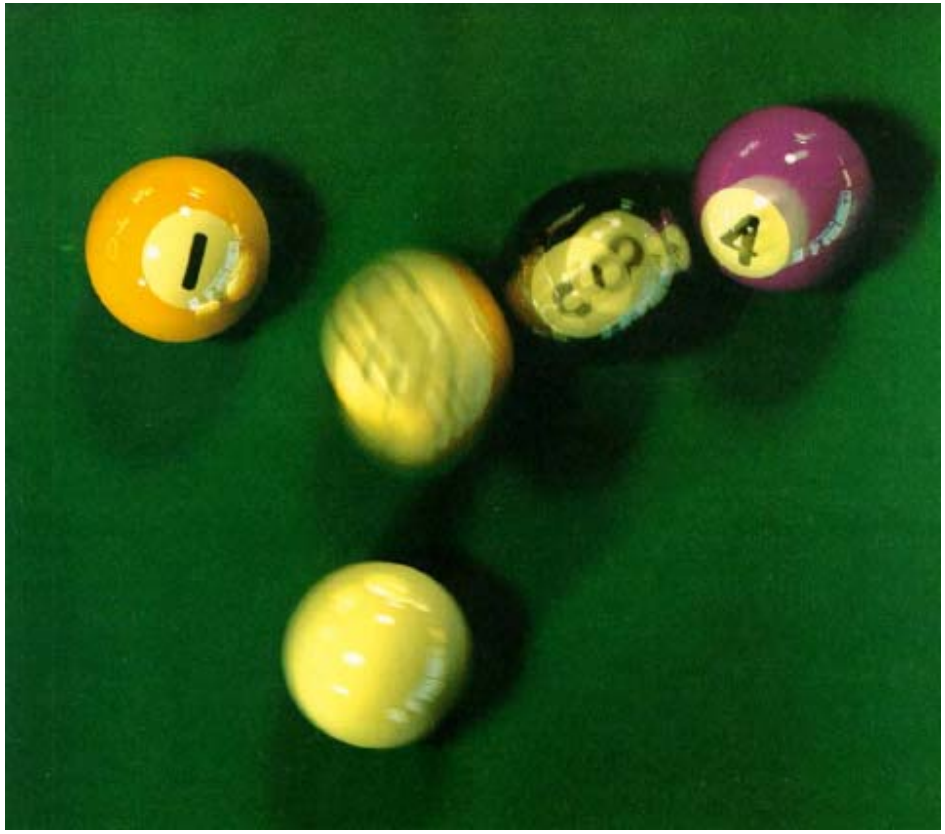
KAIST

Other Optimizations

- **Shadow cache**
- **Adaptive depth control**
- **Lazy geometry loading/creation**

Distributed Ray Tracing [Cook et al. 84]

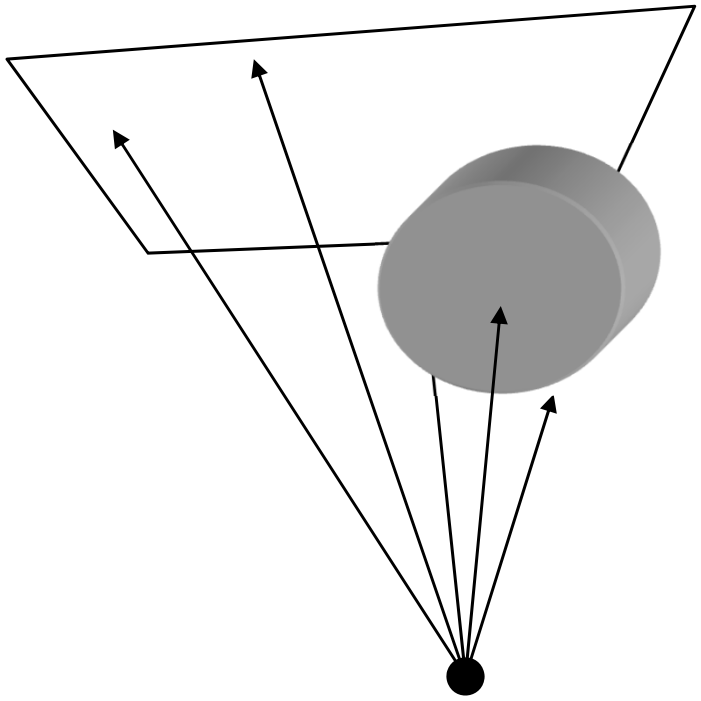
- **Cook et al. realized that ray-tracing, when combined with randomized sampling, i.e., "jittering", could be adapted to address a wide range of rendering problems:**



KAIST

Soft Shadows

- Take many samples from area light source and take their average
  - Computes fractional visibility leading to penumbra

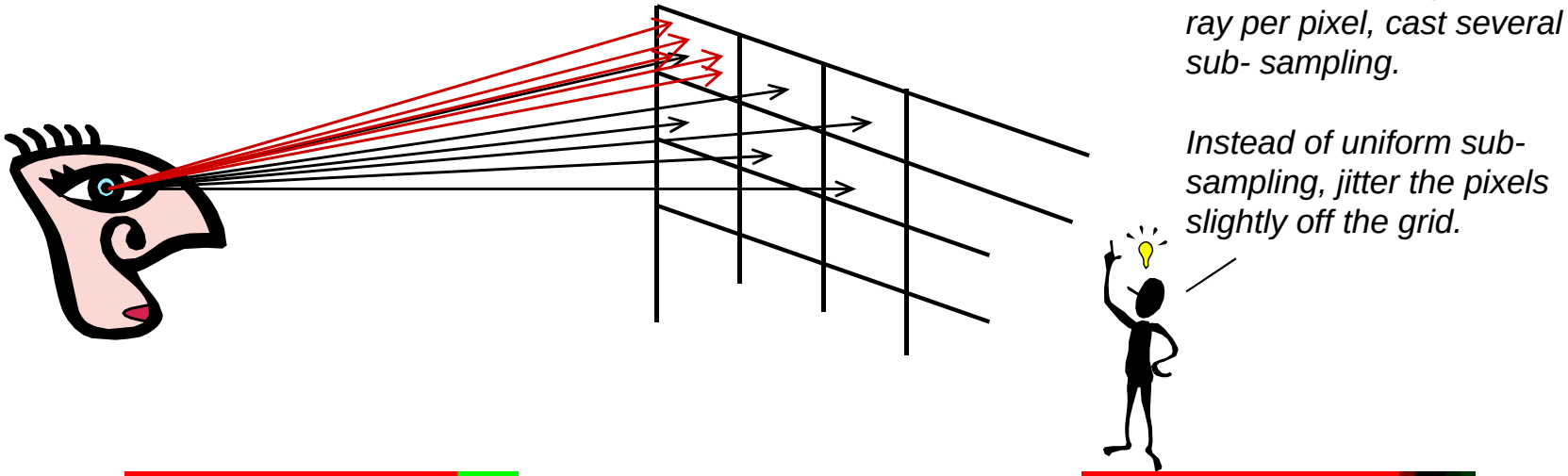


Antialiasing

- **The need to sample is problematic because sampling leads to aliasing**
- **Solution 1: super-sampling**
  - **Increases sampling rate, but does not completely eliminate aliasing**
  - **Difficult to completely eliminate aliasing without prefiltering because the world is not band-limited**

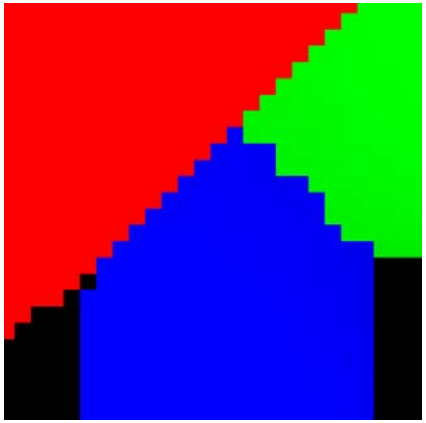
Antialiasing

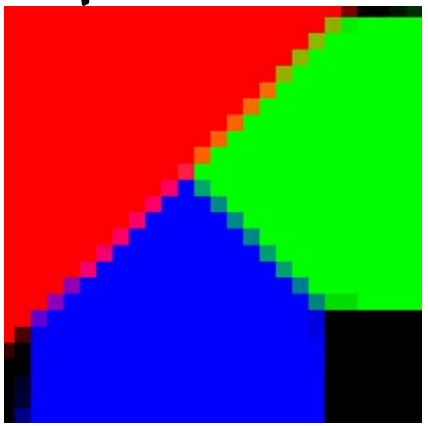
- **Solution 2: distribute the samples randomly**
  - **Converts the aliasing energy to noise which is less objectionable to the eye**



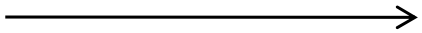
Instead of casting one ray per pixel, cast several sub- sampling.

Instead of uniform sub-sampling, jitter the pixels slightly off the grid.

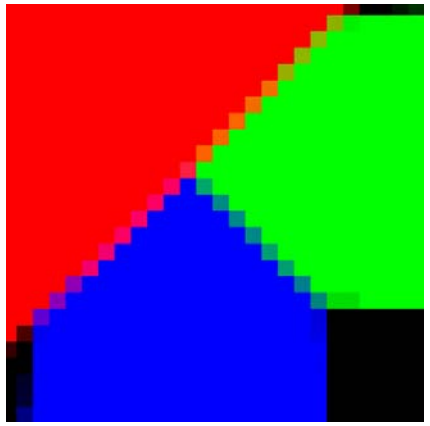


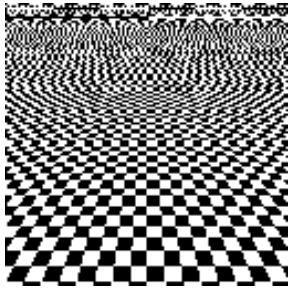


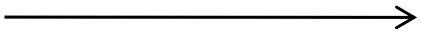
Jittering Results for Antialiasing

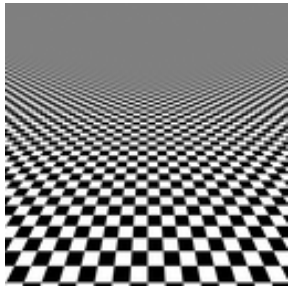


2x2
sub-sampling



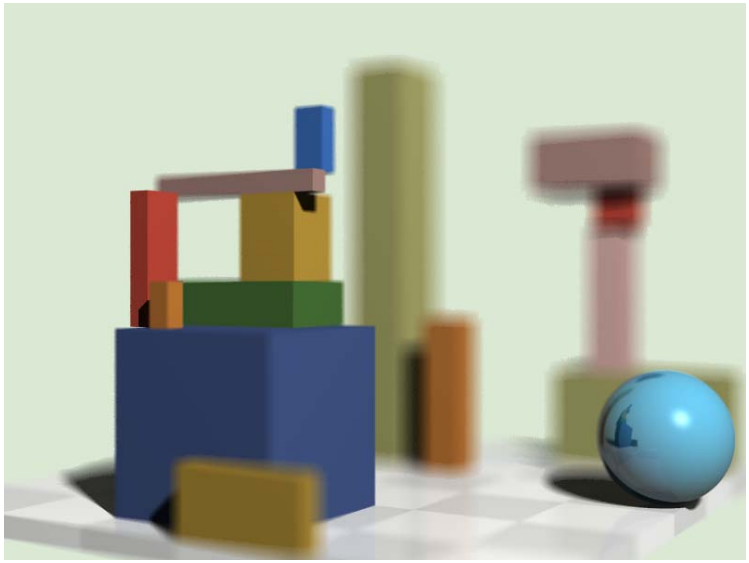




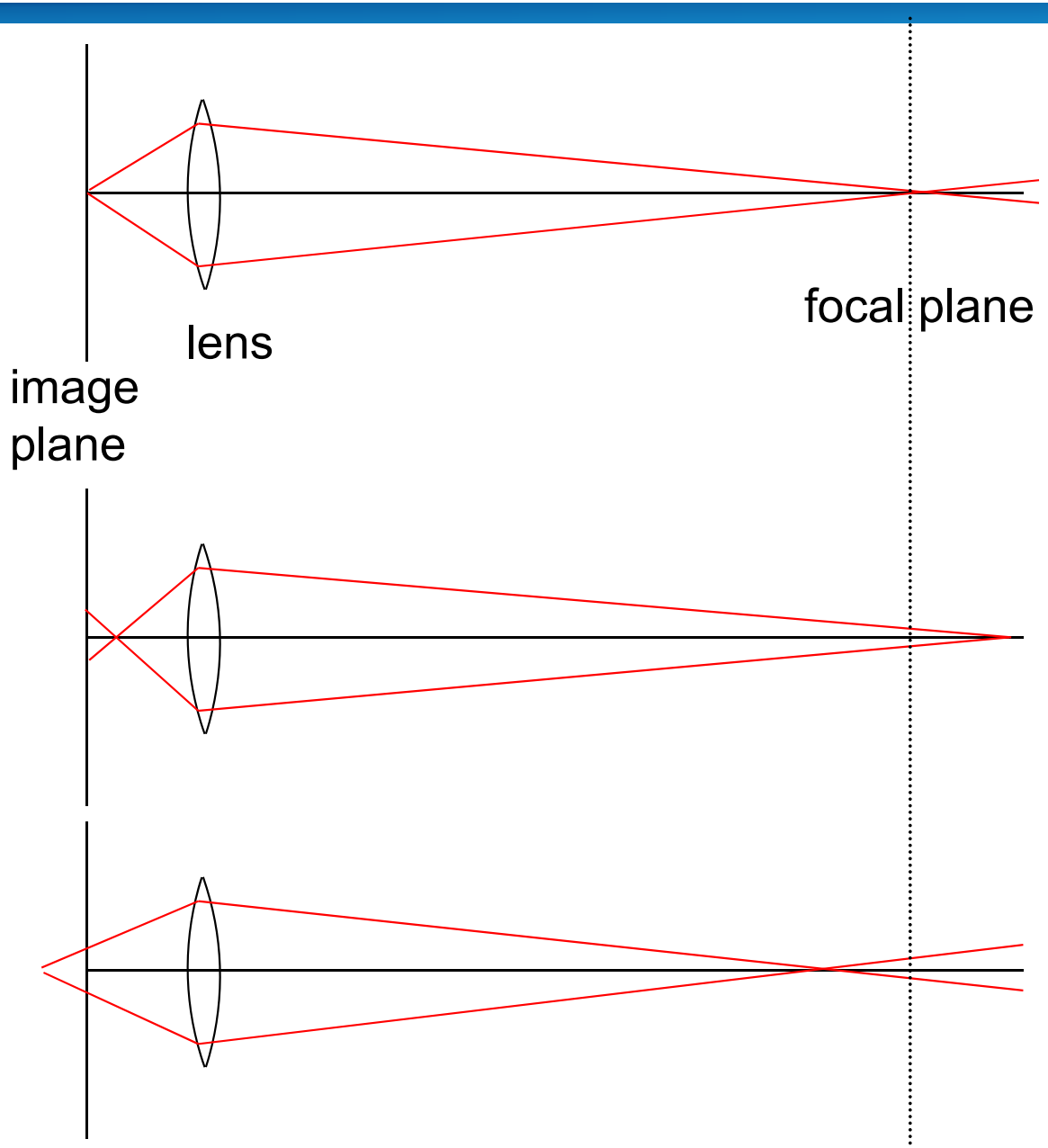


Depth-of-Field

- Rays don't have to all originate from a single point.
- Real cameras collects rays over an aperture
  - Can be modeled as a disk
  - Final image is blurred away from the focal plane
  - Gives rise to depth-of-field effects



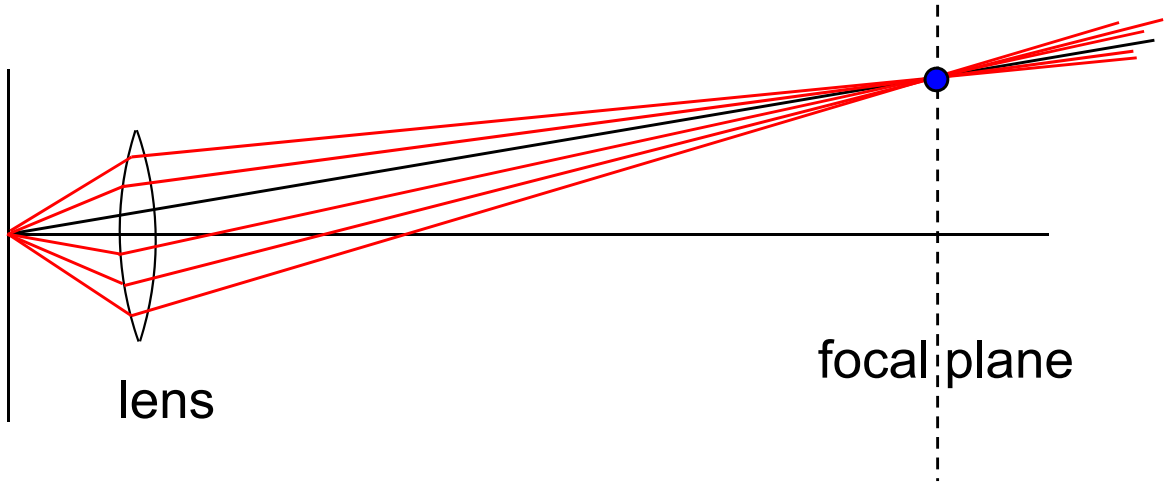
Depth of Field



KAIST

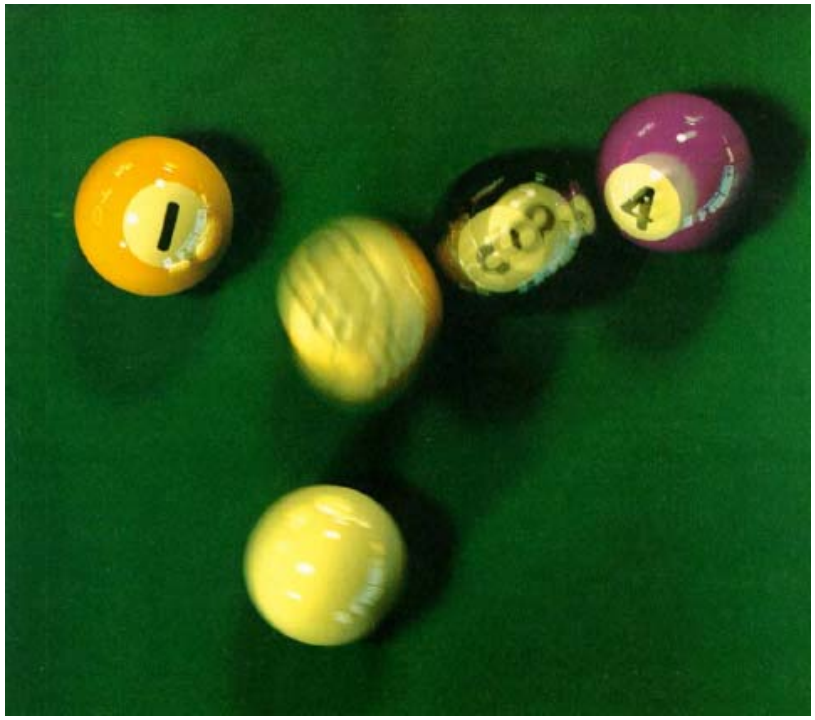
Depth of Field

- **Start with normal eye ray and find intersection with focal plane**
- **Choose jittered point on lens and trace line from lens point to focal point**



KAIST

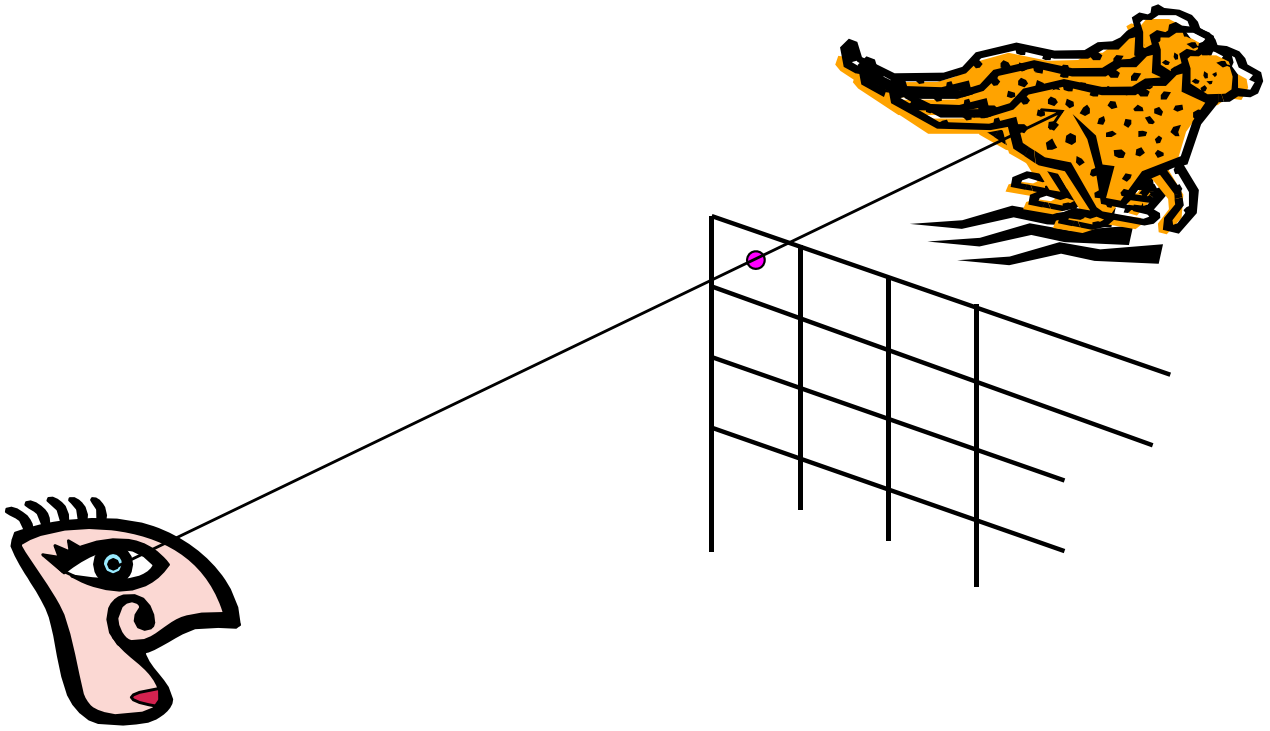
Motion Blur



- **Jitter samples through time**
  - **Simulate the finite interval that a shutter is open on a real camera**

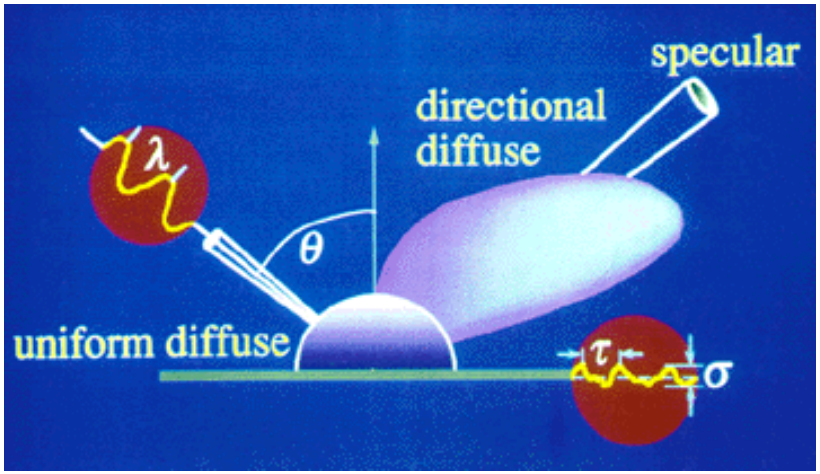
KAIST

Motion Blur



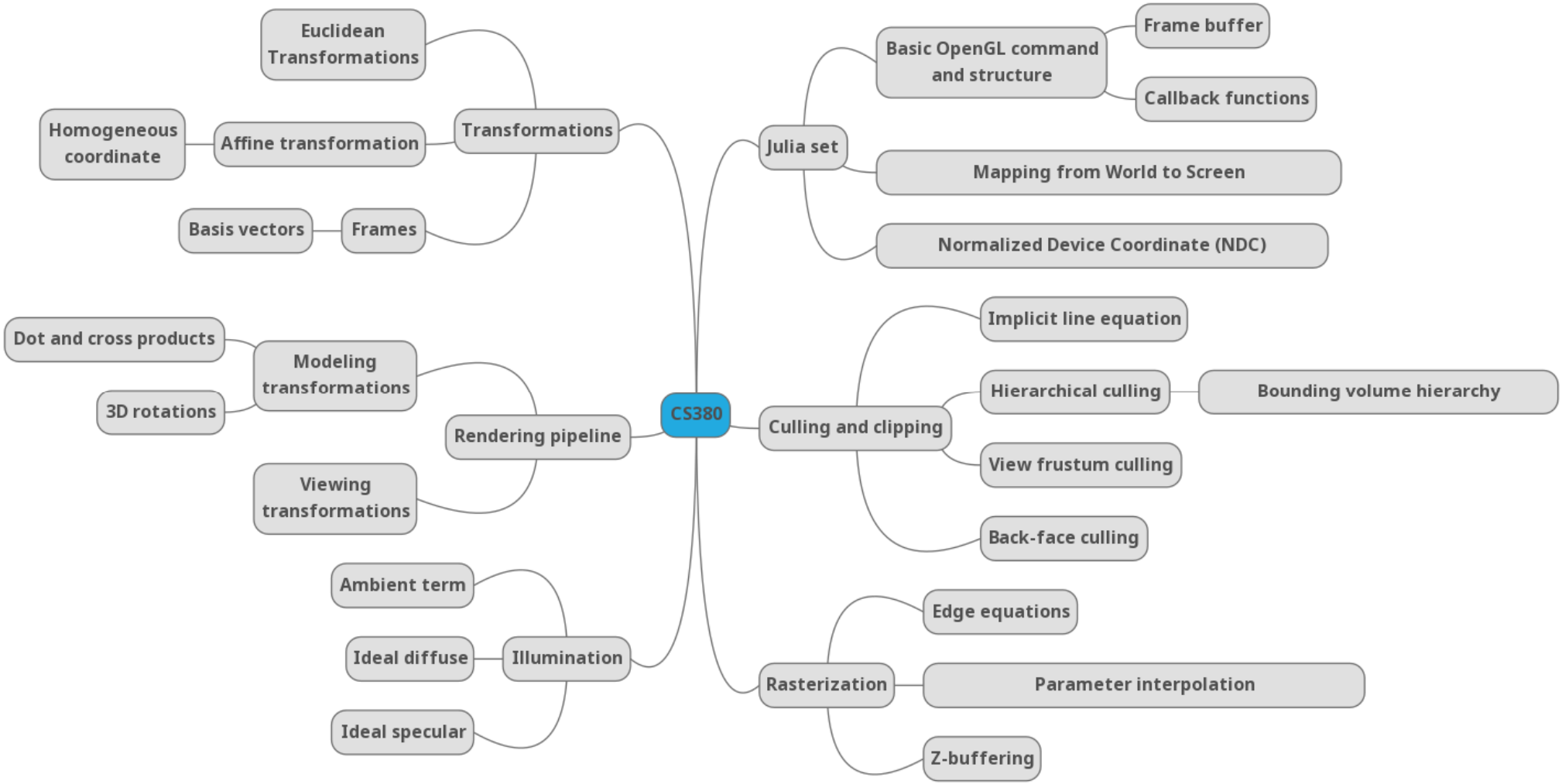
Complex Interreflection

- **Model true reflection behavior as described by a full BRDF**
- **Randomly sample rays over the hemisphere, weight them by their BRDF value, and average them together**
  - **This technique is called "Monte Carlo Ray Tracing"**



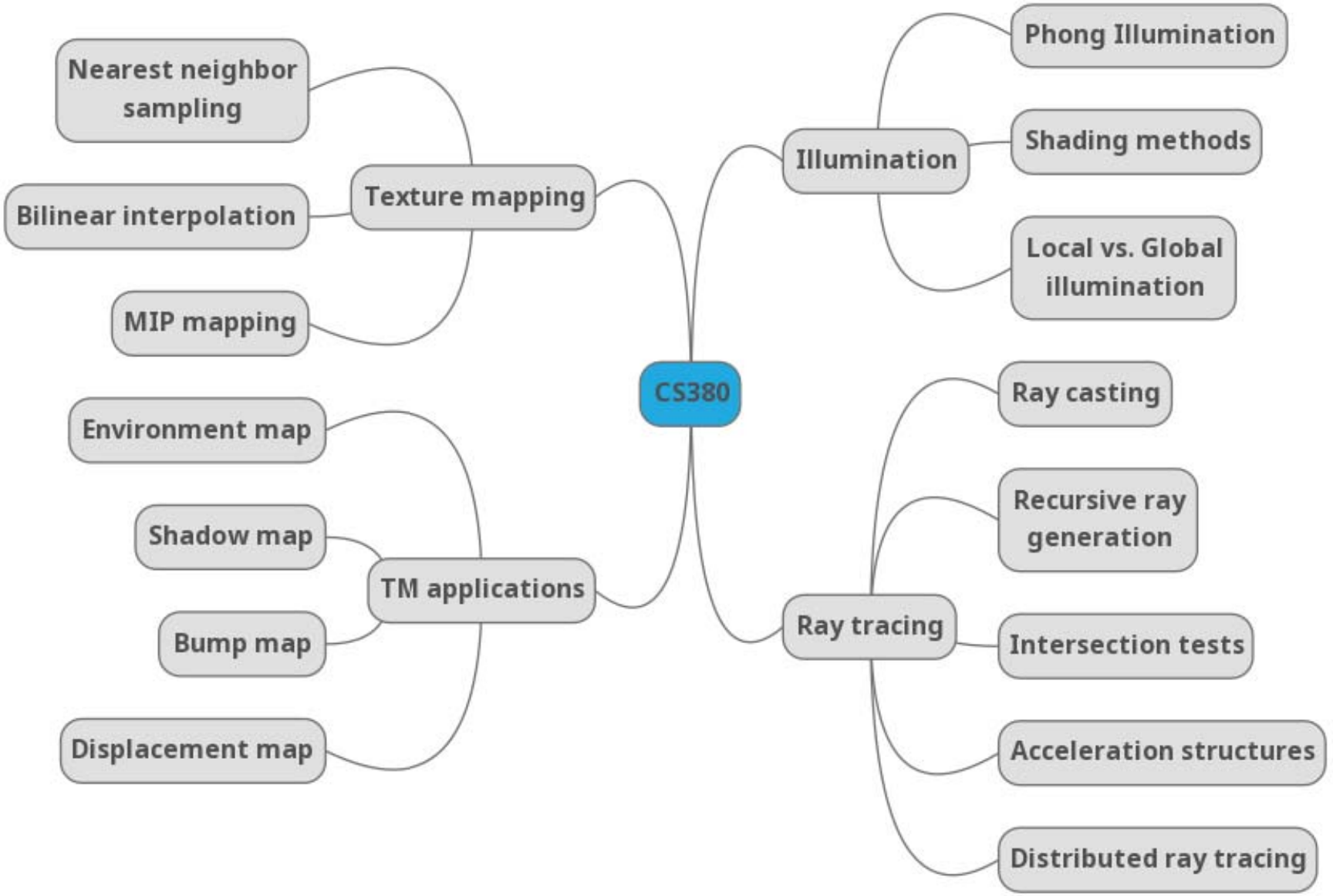
KAIST

Summary up to mid-term exam



KAIST

Summary after that



Related Courses

- **CS580: Advanced Computer Graphics**
  - **Focus on rendering techniques that generate photo-realistic images**
- **CS482: Interactive Computer Graphics**
  - **Interactive global illumination implemented by rasterization approaches**
  - **Techniques used in recent games**
  - **I'll teach it at Fall of 2021**