Two Major Rendering Methods: Rasterization and Ray Tracing

Sung-Eui Yoon
(윤성의)

Course URL:
http://jupiter.kaist.ac.kr/~sungeui/SGA/

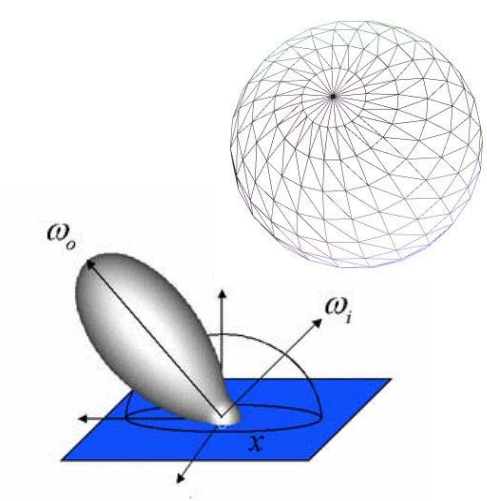
KAIST

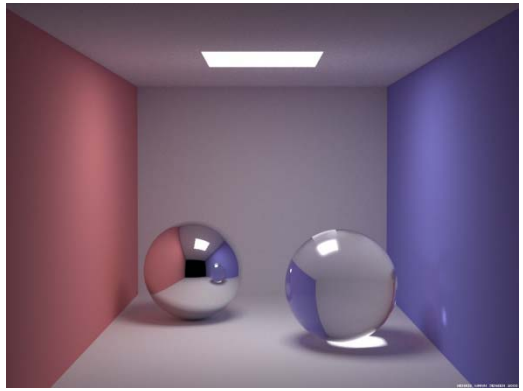
At the Previous Class

- **The overview of the course**
  - **Main theme: designing scalable graphics/geometric algorithms**
  - **Course policy: student presentations and a report at a chosen topic**

What is Rendering?

- **A process that draws an image from model descriptions**





Modelling

Simulation & Rendering

Image

Computer vision inverts the process

KAIST

Physically-based Rendering

- **Generate photo-realistic images based on reality, more precisely, physics**

hof.povray.org

KAIST

Two Major Rendering Techniques

- **Ray tracing**
  - **Main engine for global illumination**
  - **Used for high quality image generations for movies, architectures, etc.**

- **Rasterization**
  - **Simplified rendering method based on local illumination**
  - **Used for interactive rendering for games**
  - **Accelerated with hardware due to the simplicity of the method**

Local illumination

- Compute the colors of points on surfaces by considering only local properties
  - Position of the point
  - Surface properties
  - Properties of any light sources that affect it
- No other objects in the scene are considered neither as light blockers nor as reflectors
- Used in OpenGL and DirectX

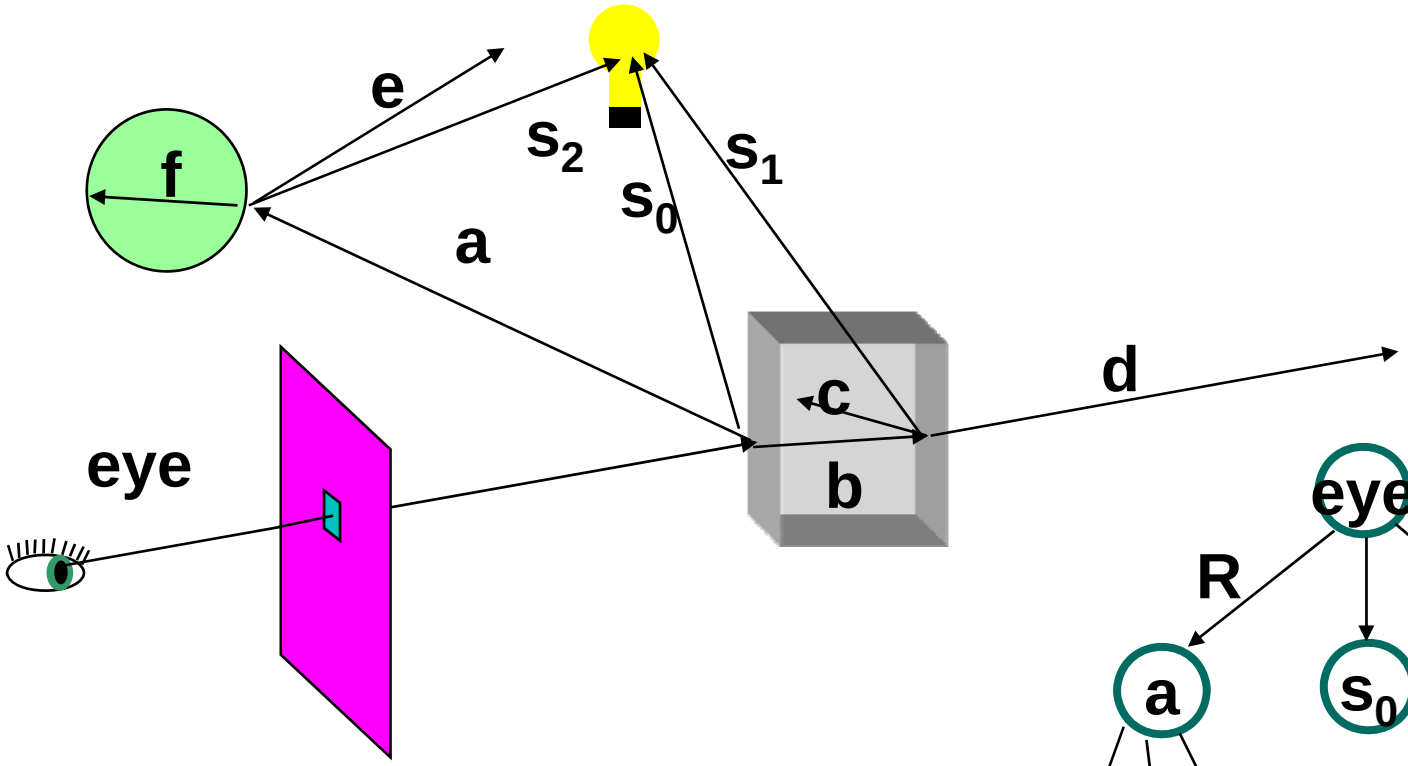
Global Illumination

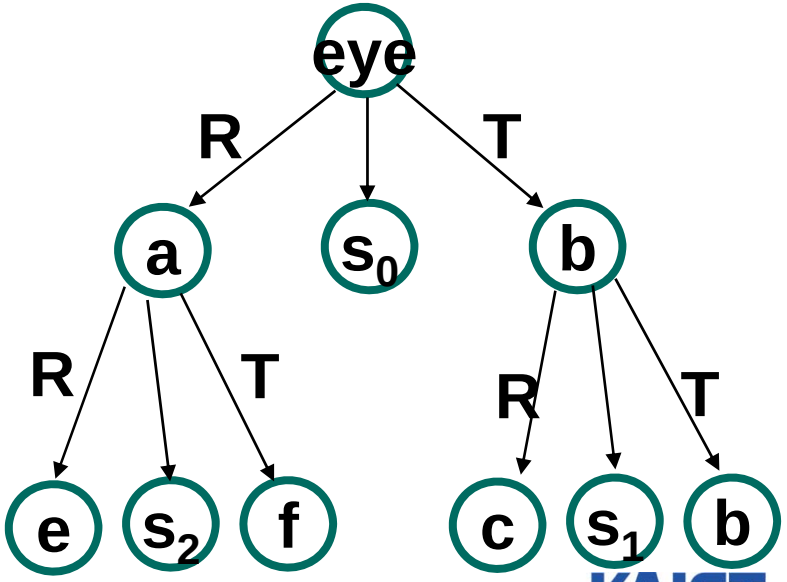
- **In the real world, light takes indirect paths**
  - **Light reflects off of other materials (possibly multiple objects)**
  - **Light is blocked by other objects**
  - **Light can be scattered**
  - **Light can bend**
- **Harder to model**
  - **At each point we must consider not only every light source, but and other point that might have reflected light toward it**

Basic Ray Tracing

- **Basic Idea [Whitted 80]**
  - **Cast a ray into the scene from eye through pixels**
  - **Computer a first-intersection and recursively construct reflected/shadow/refracted rays**
- **Isn't this backwards?**
  - **Light goes from the light sources to the eye**
  - **Why go the other way?**
- **Ray tracing minus secondary rays usually called "ray casting"**

Ray Tree

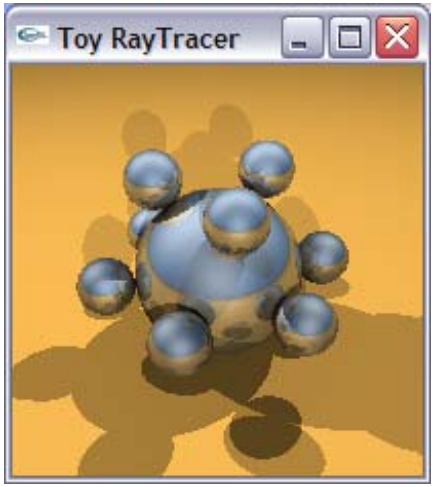




KAIST

Acceleration Methods

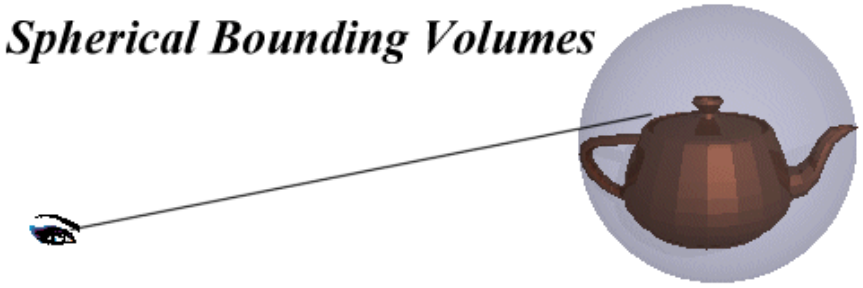
- Render time for a ray tracer depends on the number of ray intersection tests per pixel
  - Roughly dependent on the number of primitives in the scene times the number of pixels
- Early efforts focused on accelerating the ray-object intersection tests
- More advanced methods required to make ray tracing practical
  - Bounding volume hierarchy
  - Spatial partitioning hierarchy

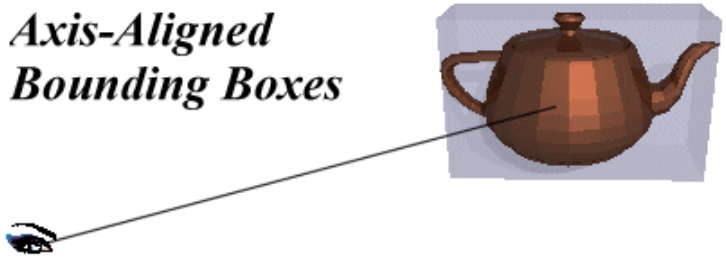


KAIST

Bounding Volumes

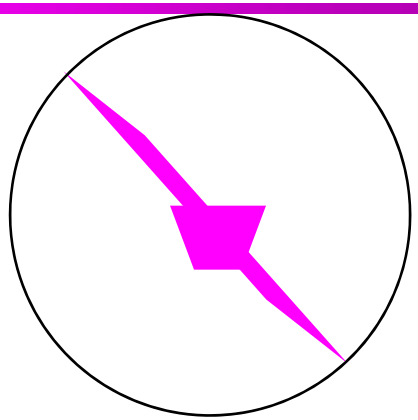
- **Enclose complex objects within a simple-to-intersect object**
  - **If the ray does not intersect the simple object then its contents can be ignored**
  - **The likelihood that it will strike the object depends on how tightly the volume surrounds the object.**
- **Spheres are simple, but not tight**
- **Axis-aligned bounding boxes often better**

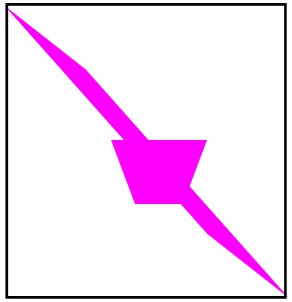




Bounding Volumes

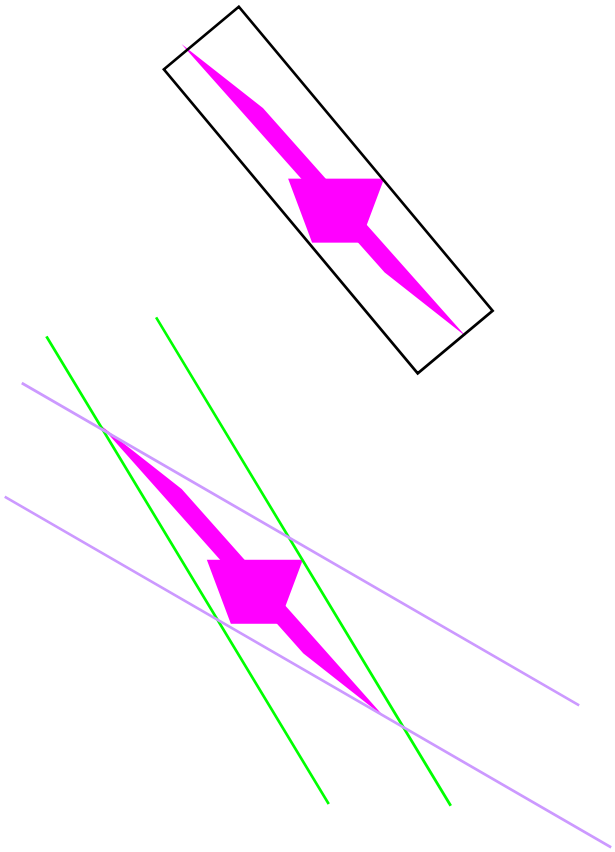
- **Sphere [Whitted80]**
  - Cheap to compute
  - Cheap test
  - Potentially very bad fit
- **Axis-aligned bounding box**
  - Very cheap to compute
  - Cheap test
  - Tighter than sphere





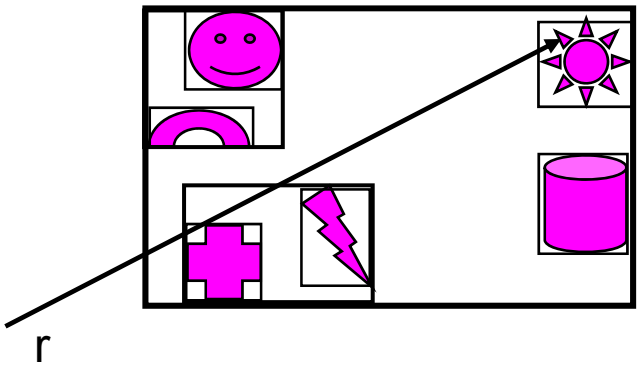
Bounding Volumes

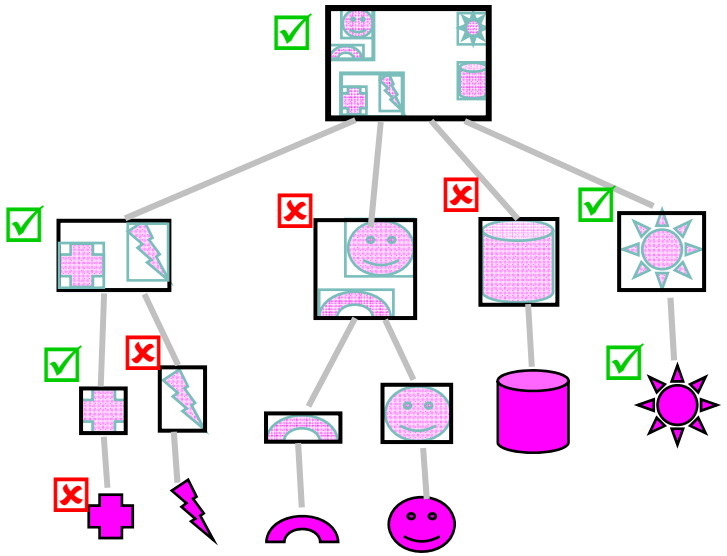
- **Oriented Bounding Box**
  - **Fairly cheap to compute**
  - **Fairly cheap test**
  - **Generally fairly tight**
- **Slabs / K-dops**
  - **More expensive to compute**
  - **Fairly cheap test**
  - **Can be tighter than OBB**



Bounding Volume Hierarchy

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



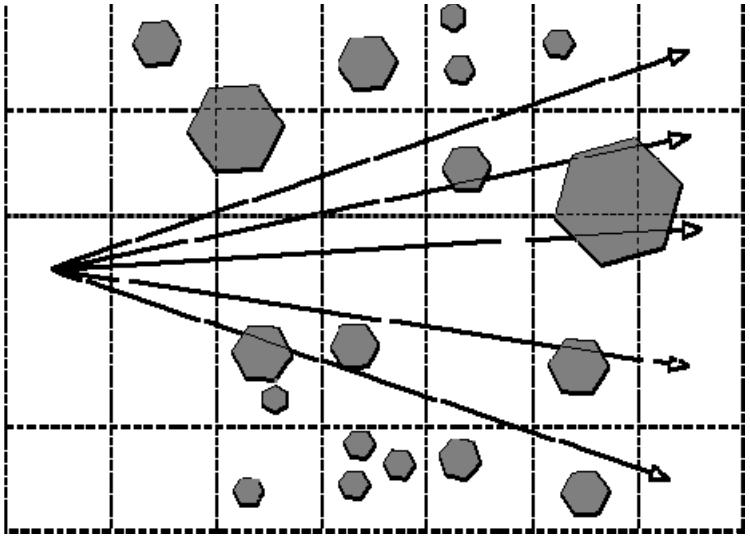


KAIST

Spatial Partitioning Hierarchy

Idea: Divide space to sub-regions

- **Place objects within a sub-region into a list**
- **Only traverse the lists of sub-regions that the ray passes through**
- **“Mail-boxing” used to avoid multiple test with objects in multiple regions**
- **Many types**
  - **Regular grid**
  - **Octree**
  - **BSP tree**
  - **kd-tree**

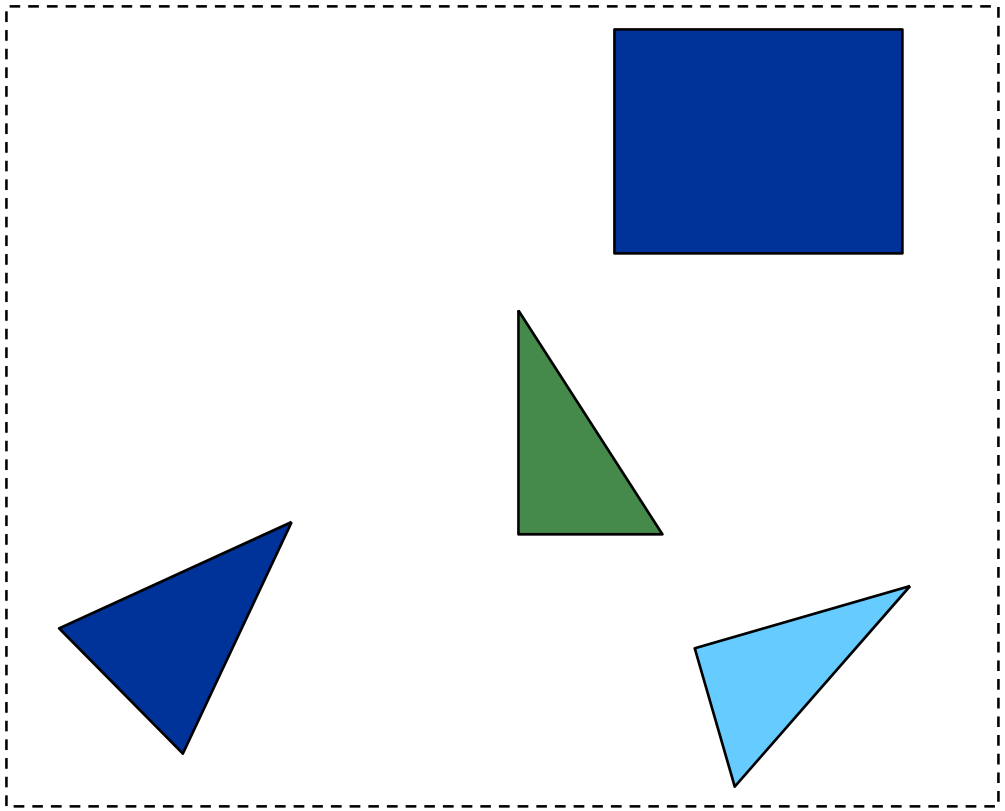


KAIST

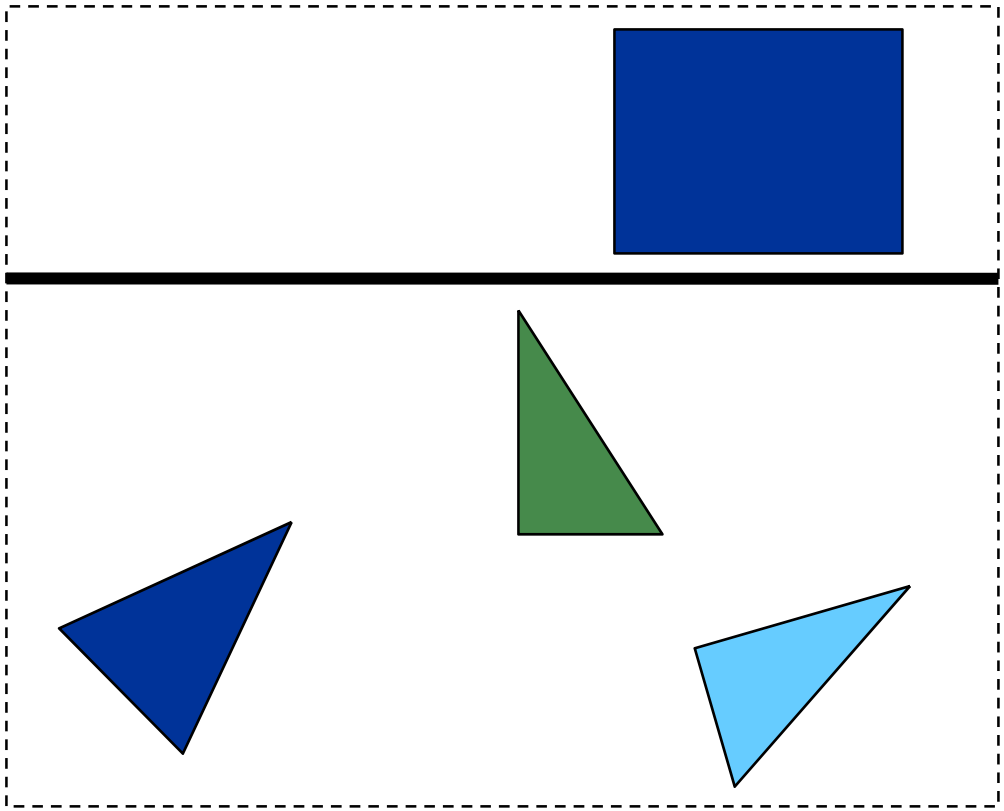
kd-trees: overview

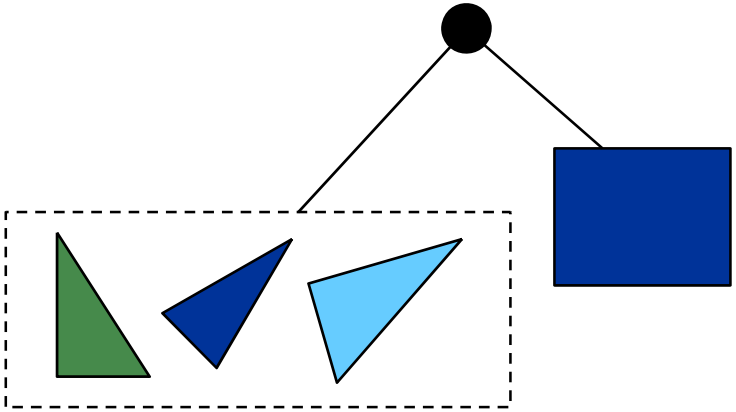
- **Binary spatial subdivision (special case of BSP tree)**
  - **Split planes aligned on main axes**
  - **Inner nodes: subdivision planes**
  - **Leaf nodes: triangle(s)**

kd-trees: example

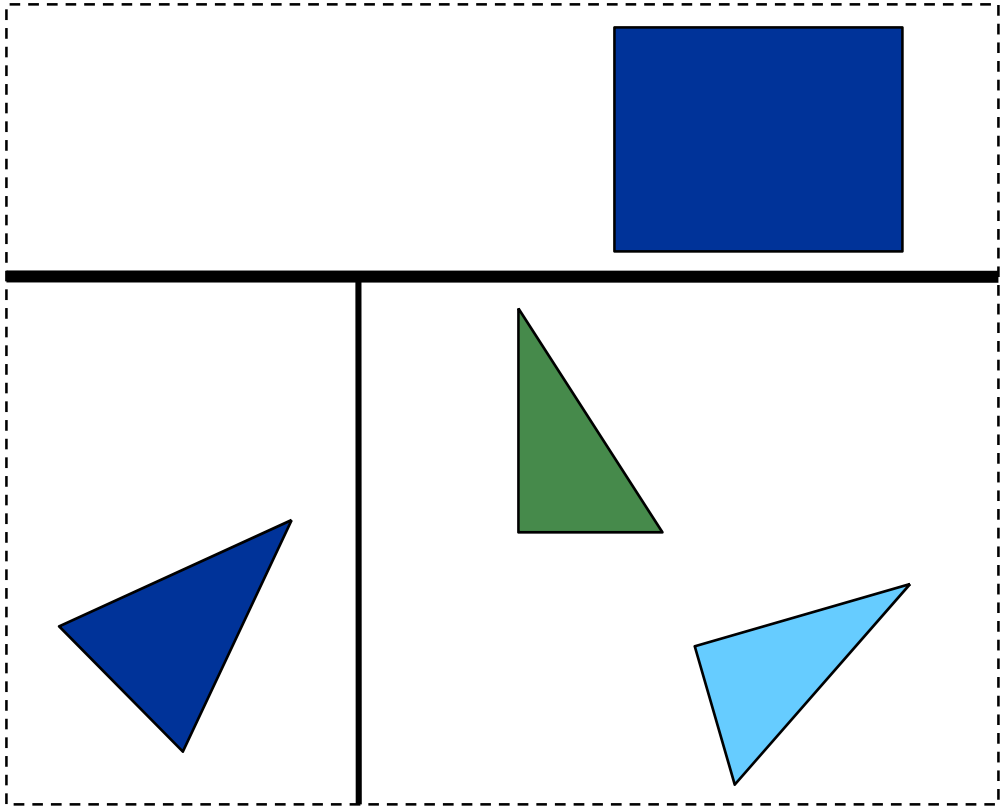


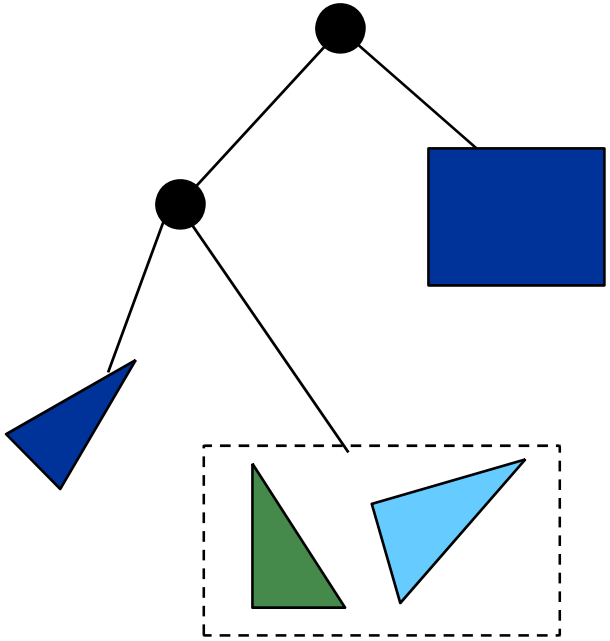
kd-trees: example



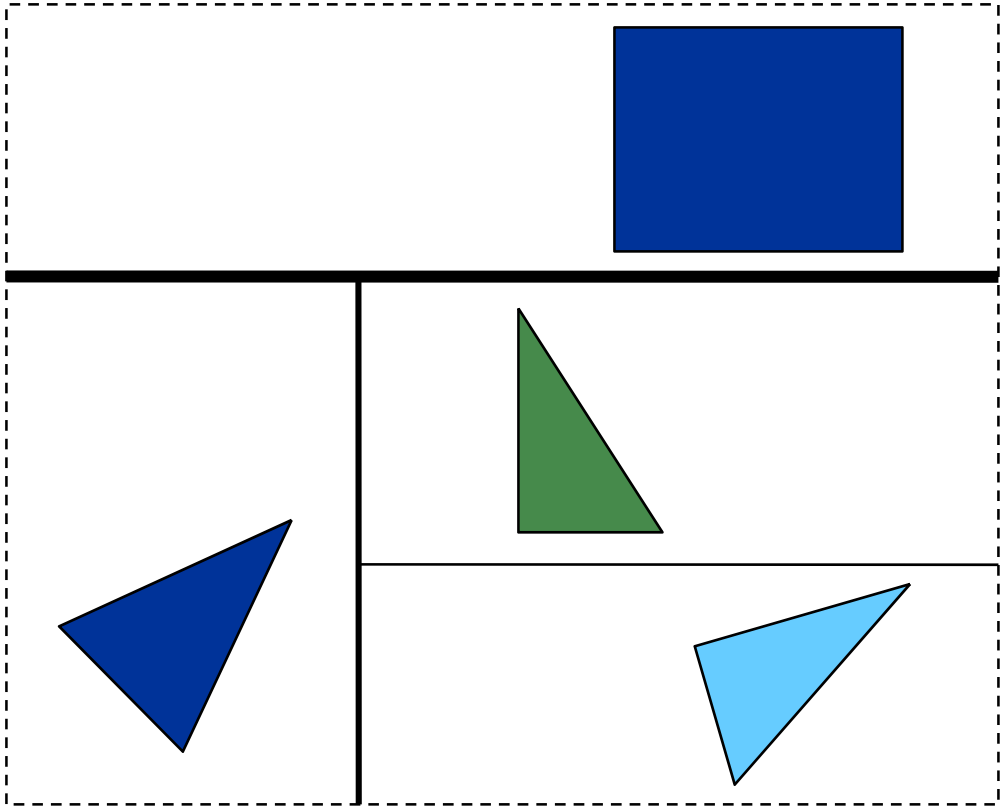


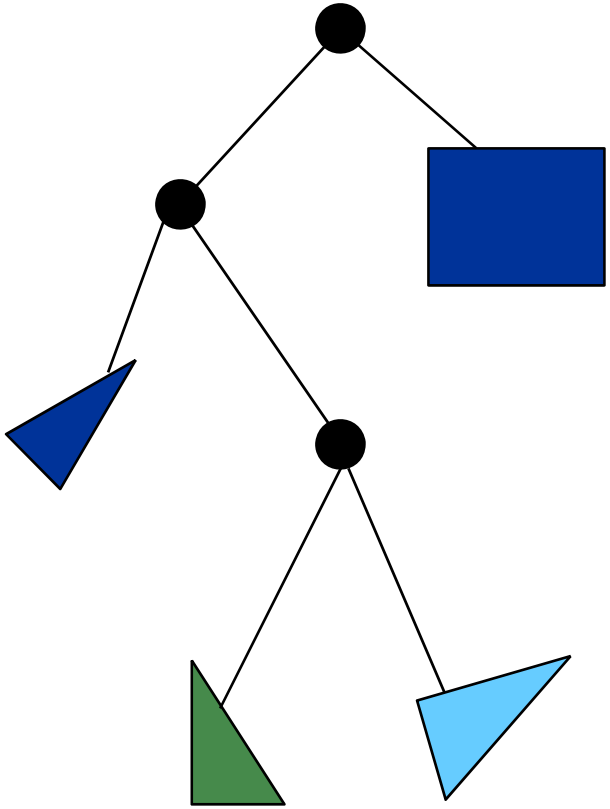
kd-trees: example



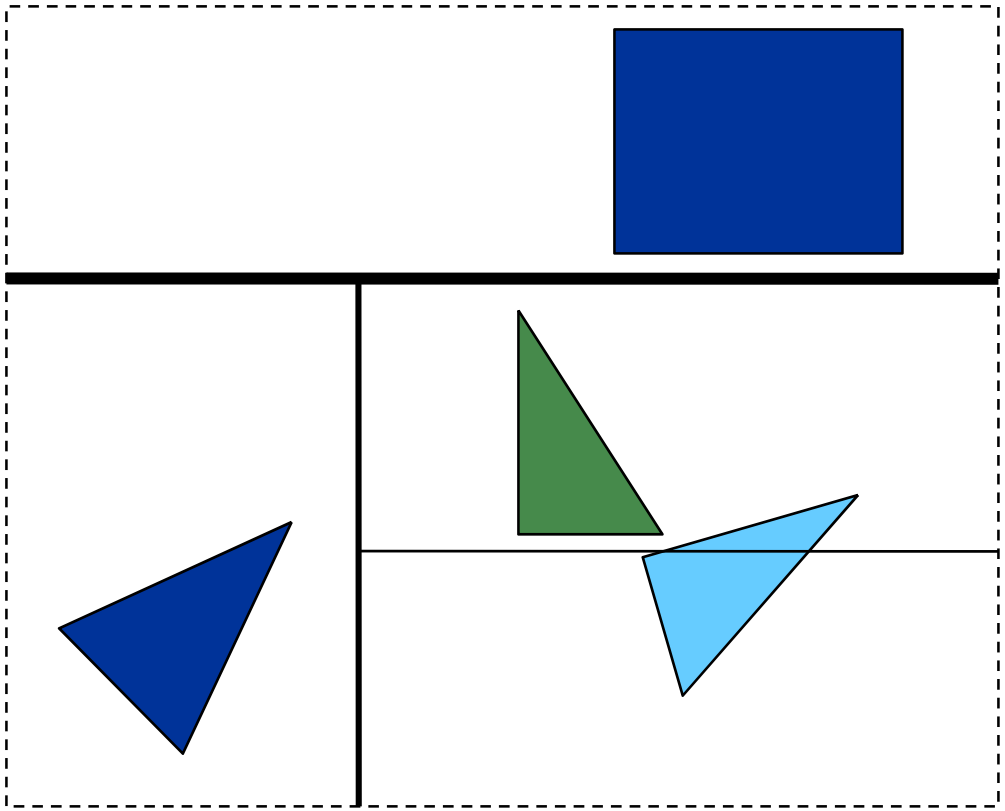


kd-trees: example



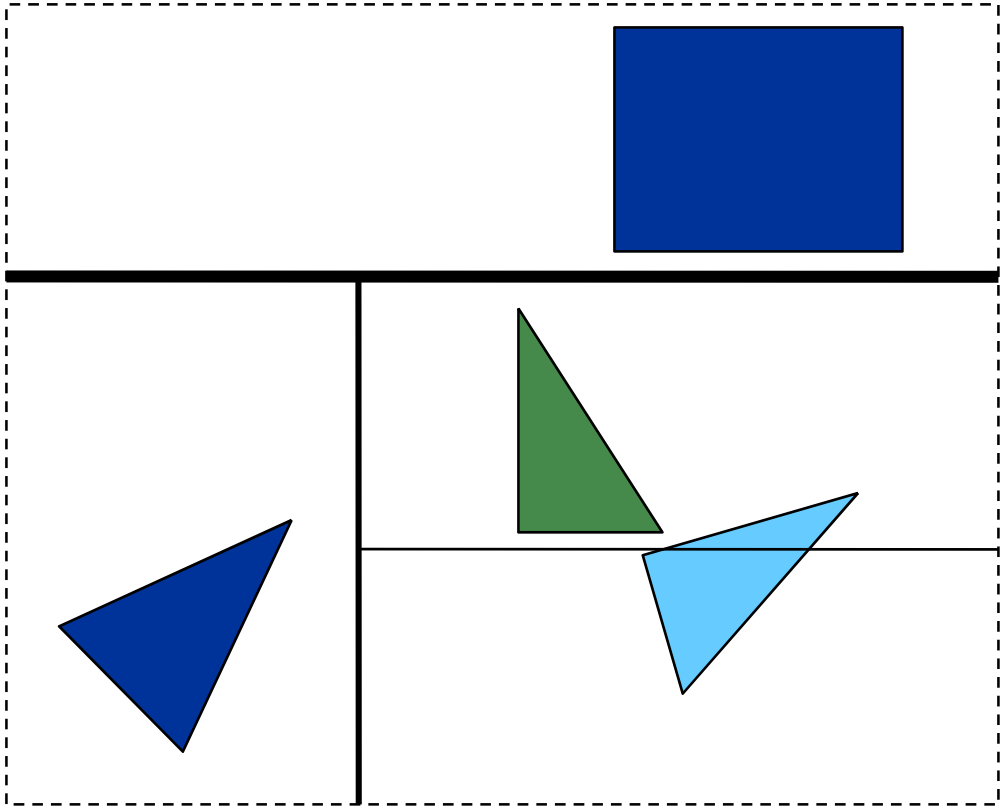


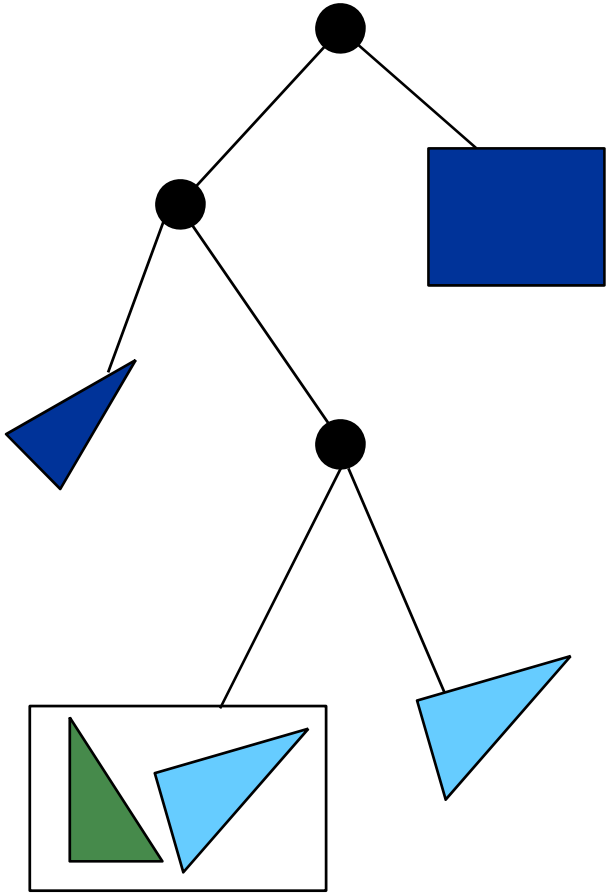
kd-trees: example



What about triangles overlapping the split?

kd-trees: example

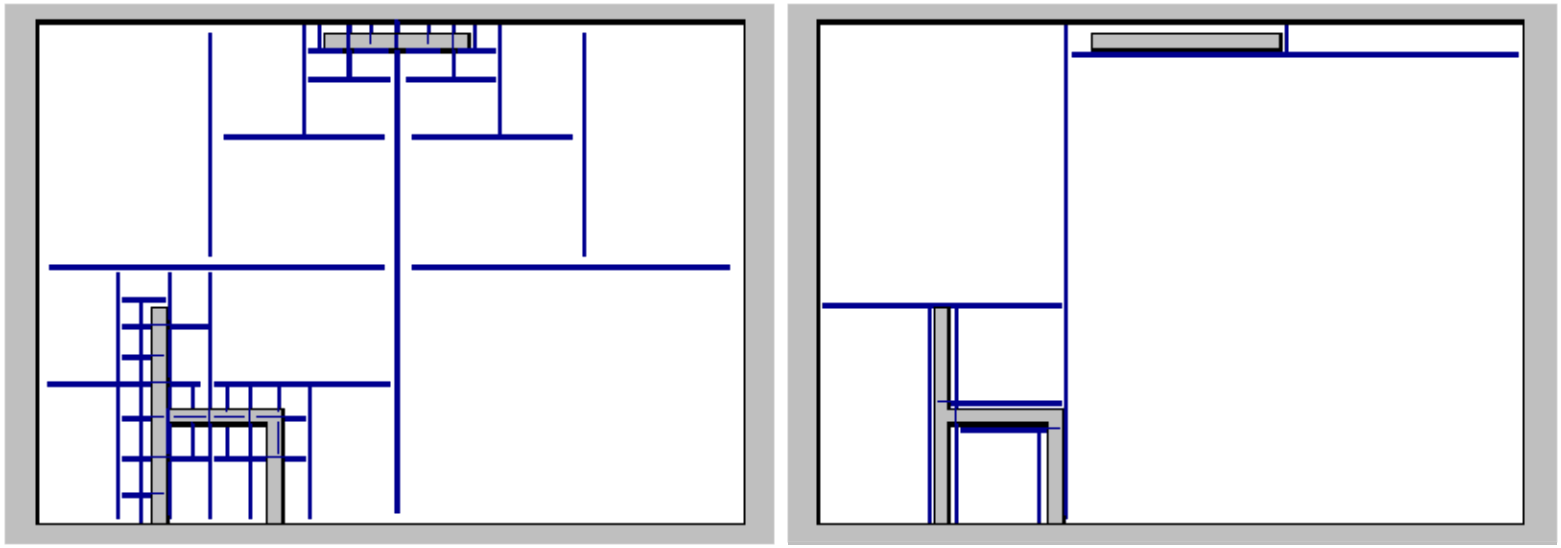




kd-tree: split planes

- How to select axis & split plane?
  - Largest dimension, subdivide in middle
  - More advanced: surface area heuristic [Goldsmith and Salmon 87] and other improvements [Reshetov05]
- Makes large difference
  - 50%-100% higher overall speedup [Wald04]

kd-tree: median vs. SAH



(from [Wald04])

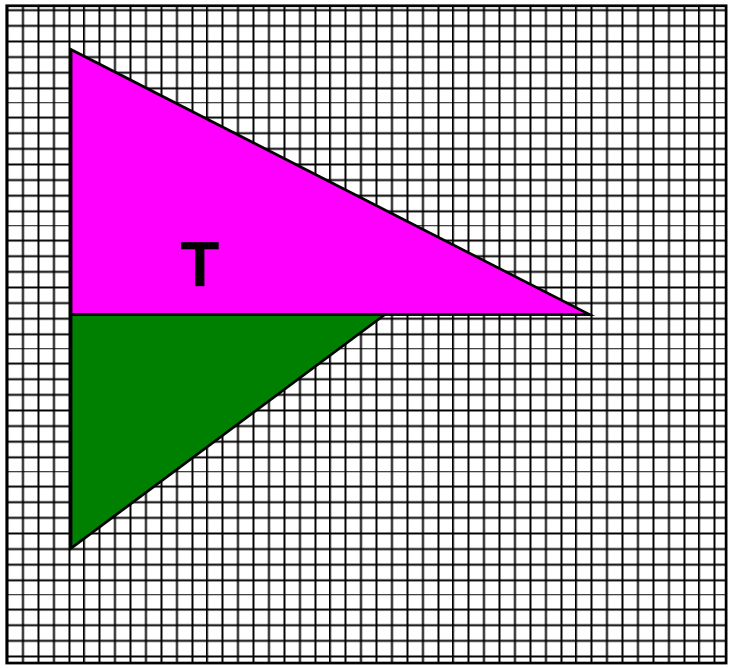
KAIST

Ray Tracing with the kd-tree

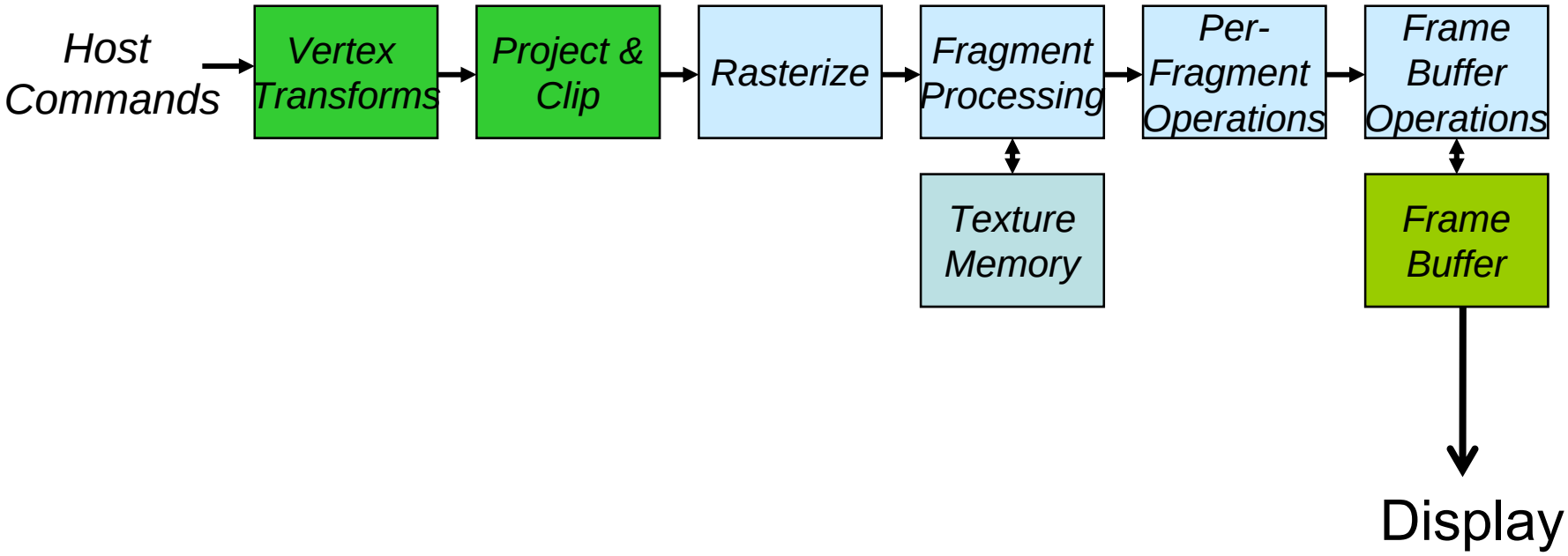
- Goal: find closest hit with scene
- Traverse tree front to back (starting from root)
- At each node:
  - If leaf: intersect with triangles
  - If inner: traverse deeper

Rasterization

- **Standard method for rendering**
- **Draw all triangles on a raster:**



Rasterization: Rendering Pipeline



KAIST

Rasterization

- **Advantage:**
  - **Use graphics hardware / GPUs (fast, growing even above Moore's Law)**
  - **1-2 orders of magnitude faster than ray tracing**
- **Disadvantages:**
  - **Local illumination**
  - **Performance ~ linear to # triangles: can we do better?**

Data Access Pattern

- Streaming or sequential accesses for rasterization
  - Object-driven
- Random access for ray tracing
  - Image-driven
  - Can we make ray tracing to have streaming data access pattern?

What are Issues?

- **Massive models**
  - **Hierarchy can take very large (memory and disk) space**
- **Many lights**
  - **Performance can linearly grow as the number of lights**
- **Dynamic models**
  - **Need to update hierarchy**
  - **Which one is better between BVH or SPH?**
- **Controllability**
  - **Can we trade the quality and performance?**

Homework

- **Start choosing a topic that is most interesting to you!**
  - **You can bring your own research to the course**
- **Reference**
  - **Course homepage**
  - **SIGGRAPH and other papers**
  - **Refer paper collections in the homepage**
  - **Google scholar**

Next Time..

- **Study culling techniques**
  - **E.g., visibility culling**