

Spring 2007
22C:251 Advanced Computer Graphics (Image Synthesis)
Assignment 1

Due: Before class on Tuesday, February 6th

Goal: Improve your ray tracer by implementing a bounding volume hierarchy and a non-pinhole camera.

Problem 1: (30 points) Implement a bounding volume hierarchy for your ray tracer. For this exercise, you should select your split plane by dividing the longest axis so that each child node has half as many primitives as the parent.

- Single-person groups must implement one of the following, and multi-person groups must implement at least two of the following choices:
 1. A simple, naive bounding volume hierarchy (called *NaiveBVH*). This hierarchy is a standard tree structure with each node containing two pointers to its children. Memory for nodes can be allocated independently (need not be consecutive), and tree traversal can be implemented with a recursive, depth-first search.
 2. An array-based bounding volume hierarchy (called *ArrayBVH*), where nodes in the tree need not store pointers to children, but simply an array index. You may implement this array in one of two orderings: either it may be ordered such that a node i 's children are located at indices $2i + 1$ and $2i + 2$, or it may be ordered as a skip-tree (as discussed here: <http://www.cs.utah.edu/~bes/papers/fastRT/paper-node12.html>).
 3. An bounding volume hierarchy (called *BVH*), where the traversal step (that occurs inside *Intersect()*) is an iterative instead of a recursive process. The underlying tree representation can either be that of the *NaiveBVH* class or the *ArrayBVH* class. To implement this, you *may* need to implement your own stack inside *intersect*. In this case, the size of the stack should be set using a `#define` inside "Definitions.h". A good value for the stack size is 32, which allows the tree to store 2^{32} primitives.
- This should be implemented as a class that inherits from the *Group* base class.
- Because BVHs require a preprocessing step, you should implement a virtual *Preprocess()* method in the *Object* class and its derivatives. For most Object-types, this function does nothing. However *Group*'s should call *Preprocess()* for every object inside of the group, and your *BVH* classes should call *Preprocess()* for every object inside the BVH (in addition to building the tree).
- I will post a class *BBox* that defines basic operations on a bounding box.
- Because you will need to determine a bounding box for each object, you will need to declare a virtual method *ComputeBoundingBox()* for all *Object*-based classes, including all primitives.

Optimization note: To maintain the performance of your BVH throughout the rest of the semester, make sure you do not dynamically allocate memory while traversing the tree. Any dynamic memory allocation should occur only during the build process.

Problem 2: (20 points) Implement a thin-lens camera, as described on page 68 of *Realistic Ray Tracing*. In order to do this, you must write a new camera class *ThinLensCamera*. For good results, you will need to use multiple samples per pixel. To improve the modularity of the code, I suggest renaming the current *Camera* class to *PinholeCamera* and writing a base class *Camera* that both *PinholeCamera* and *ThinLensCamera* inherit from.

- Multi-person groups must also improve the OpenGL interface for their ray tracer to allow the following things:
 1. Allow the user to interactively change the number of sample rays per pixel.
 2. Allow the user to interactively change the focal length of the thin-lens camera.
 3. Allow the user to resize the OpenGL window. The rendering should resize to fill the entire window, while maintaining the same field-of-view in the y -axis.
 4. Allow the user to move around the scene via the mouse.
 5. (Optional) Allow the user to interactively swap between pinhole and thin lens cameras.

Note: Please create a web page to demonstrate the results of this (and future) assignments, and send me a link to this web page. Also tell me if you have problems with me linking this page from the class web page.