# spheres	time without BVH	time with BVH	speedup
1000	30.060 s	0.618 s	48.6 x
10000	266.902 s	1.293 s	206.4 x
100,000	Too long (est 2350 s)	5.790 s	405.8 x

Programming language was C++. Ran on an i7-7700HQ laptop. One ray per pixel. To create a BVH, I took the list of hittable objects and recursively divided it based on the largest difference in centers. Each division separated the objects into roughly two sides. When detecting if a ray hits an object, I iterate through the tree. Some nodes require both subtrees to be looked at depending on the bounding box (which is the union of all bounding boxes below the node). We can see that the time does not scale linearly with # spheres, which is good (since linear means we test every sphere for every pixel).