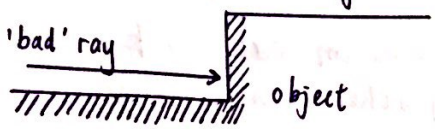
CS 348 Assignment 2

[1] an example of bad ray



If the ray is parallel to one side of the object, and it's distance to the object is really small (close to 0). In this situation, it would take a long time for the ray to intersect with object, since each time the delta of d would be small.
but not reached hitting condition

[2] ① Simple putting the individual distance estimators in a k-d tree would be more effient if the object is all over the scene and could be successful spt separated after each k-d tree spilt. In this case, time complexity of k-d tree would be $O(\log N)$ (N is the number of object) at least. while the individual distance estimator would be $O(N)$

② Hoverer, if the objects are all crowded in a small area of the scene and k-d tree could not sucessfully split, then implementing individual distance estimator would be more efficient

[3] If we set pError to zero, take 'sphereDE' image as an example! The image would consisted of discrete points as shown in the following image.

This is because we use everything is in float type. It's hard to guarantee the hit point is exactly the point we calculated. It might be very very close, but not exactly the same due to the floating number precision. So many points might not satisfy the conditions and got filtered out from the scene. So we could only see some discrete points.

[4] Troubles I met when implementing assignment 2.

① Having trouble creating correct subclass.
I had a had when implementing step 4 which seems an easy task.
I kept running into linker problems and many other minor issues.
Here is the several tips that I learned from debugging.

① [const = 0]: If declaring some function as const = 0, then the class would become abstract class. An abstract could only be used ~~for~~ for declaring & it's subclass. It should not be used to ~~dele~~ declare an object.
② Make sure you implement all pure functions (inherited or not)
③ If you declare functions with const, you should remain the same format when you actually implement the function.
④ Always remember to run cmake in build directory of pbrt.

② The second problem I come across is concerning the perspective transition. I didn't do the transformation of ray from ~~wind~~ world to object coordinate. space. I realized it when I was implementing Mandelbulb. I found out the rendered image is not placed at the right angle. So I added the transformation.

③ A shadow in 'InfiniteSphereGridDE'. The image I rendered didn't have correct shadow until I got the hint from piazza. It turn out when you implement IntersectP, ray.tmax would notonger be infinite it would become 0.9999 instead, so it's easize easy to get out of the while loop before any effective insect intersection. So we could not see shadow in the rendered image.
I also got the hint from Piazza, so I divided the distance returned from Evaluate function by the length of ray.d (direction) and finally got the right rendering.