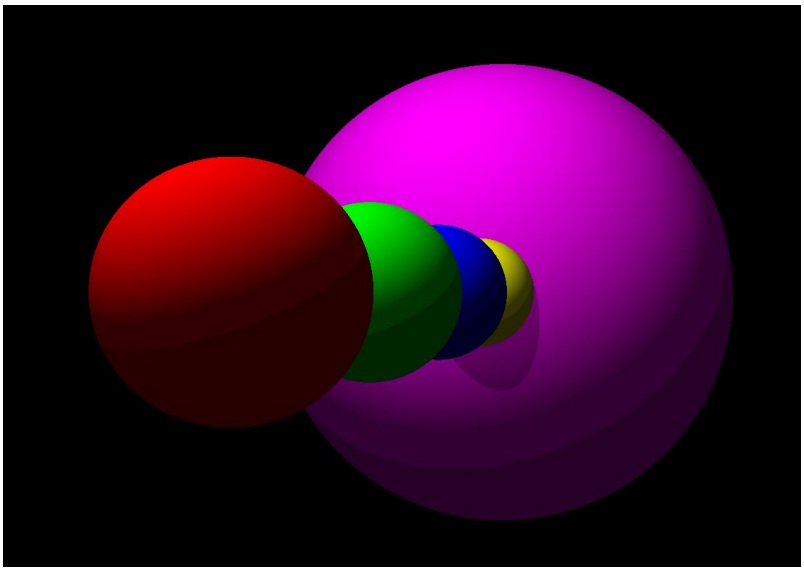
Ray Tracer Basics



When doing computer graphics one of the more interesting drawing techniques is called ray tracing. Compared to other computer graphic methods light simulation is relatively easy in a ray tracer, simply shoot a beam of light. When the beam is shot from the camera into the environment it will potentially hit things inside the world, which could be reflective, in shadow or just a solid color. Lucky for you other members on the team have already figured out all of the math for simulating the lighting effects and even shooting a ray. You however have a much more challenging task.

One caveat to a ray tracer is that the shot ray will keep going forever, hitting multiple objects on it's route, even worse this ray will proceed in both positive and negative directions. Your job is to design a function (or in this case program) to state the color of the first hit object as well as the time of the hit. A negative hit should not be recorded as a hit. To make things simpler the world has been shifted so that when a ray is shot it will always go down the same axis, and for even more simplicity there will only be spheres in this world.

HINT: Think of where the sphere hit matters

INPUT:

You will be given a list of spheres that fell on the ray when it was shot. Each sphere has a center location (on the ray) and a radius. The location L will range from -1,000,000 to 1,000,000. Radius r will range from 0 to 1,000,000. This will be followed by the color code for the sphere in Red Green Blue format. You will be responsible for multiple cases so the very first number you get will be an integer ranging from 0 to 100.

OUTPUT:

The ray tracer needs to know what the color was of the first hit sphere, and the time of the hit. Output the time rounded to the nearest 100th and then the color as the RGB value ea 255 255 255 = white. If there was no hit the program should state "NO HIT!" without the quotes. Please also list the current case number so your team can know what hit correlates to what ray.

Sample Input

```
1

1 1 255 0 0
2 1 0 255 0
3 1 0 0 255
4 1 255 255 0
5 1 255 0 255
0 0
```

Sample Output

```
CASE 1: T = 0.0 RGB = 255 0 0
```