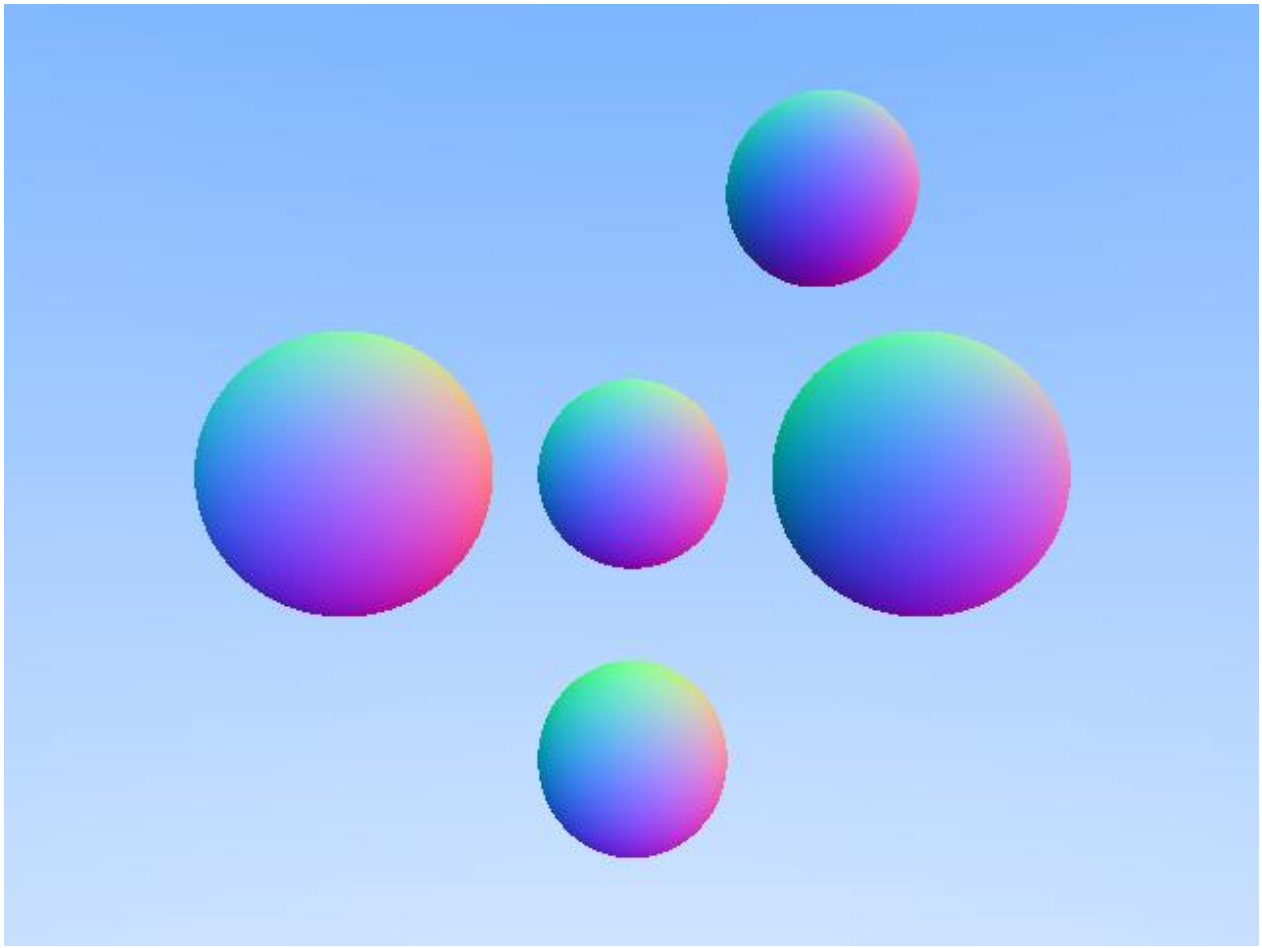
Efehan Güner 25151

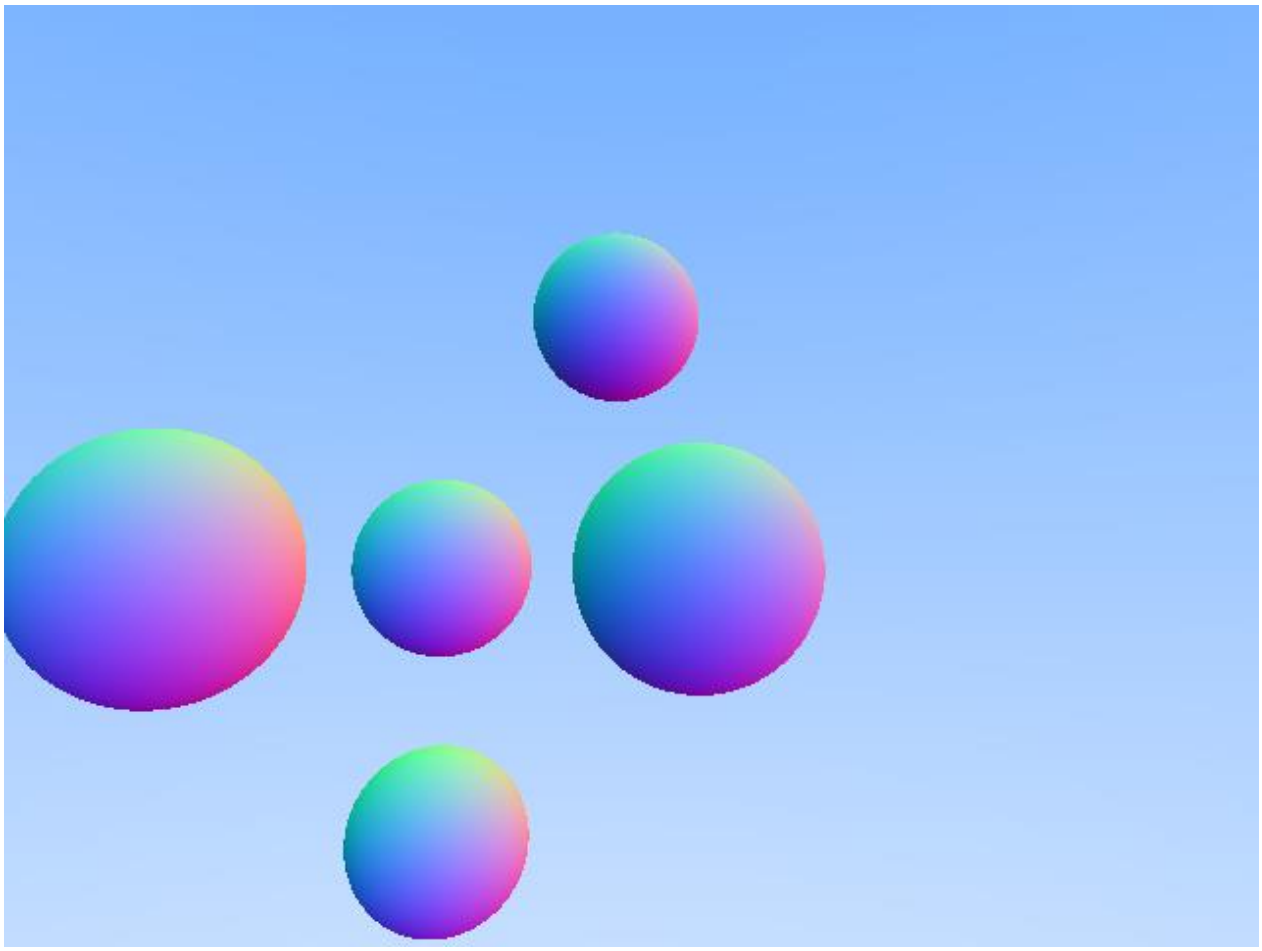
CS405 Ray Tracing Project

1)



Camera: Pos: (0, 0, 10),Look At Pos: (0, 0, 0)

Sphere Positions and Radius:
(0, -3, 0), 1
(0, 0, 0), 1
(2, 3, 0), 1
(-3, 0, 0), 1.5
(3, 0, 0), 1.5



Camera : Pos: (0, 0, 11), Look At Pos: (2, 1, 1)

2)

Figure No Super Sampling

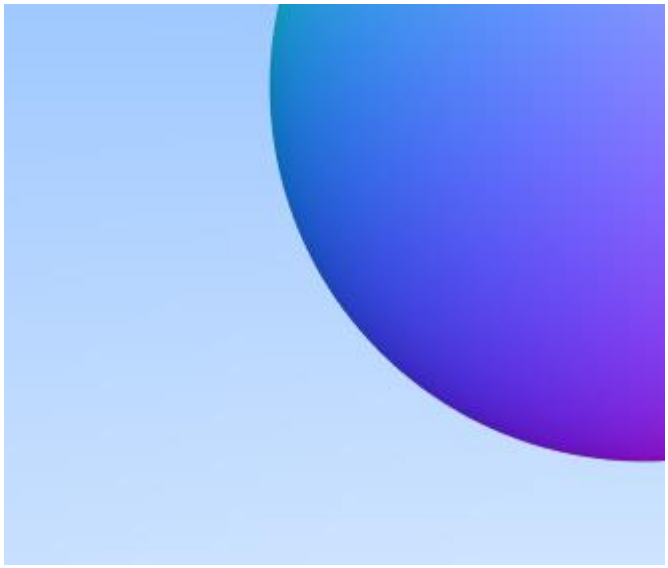


Figure Random Super Sampling

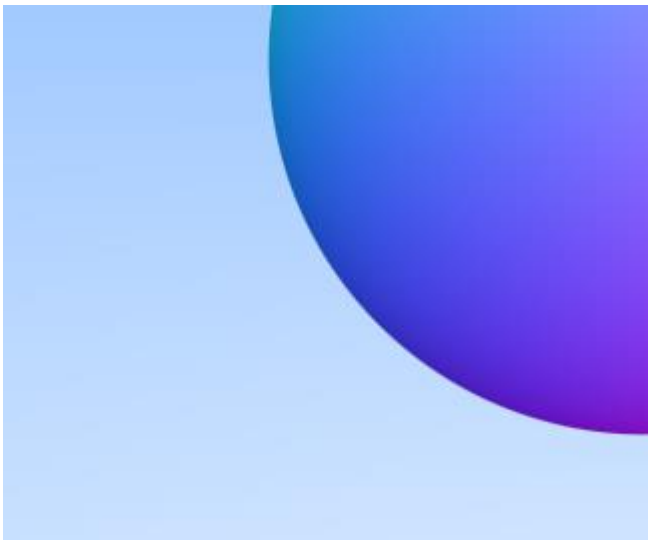
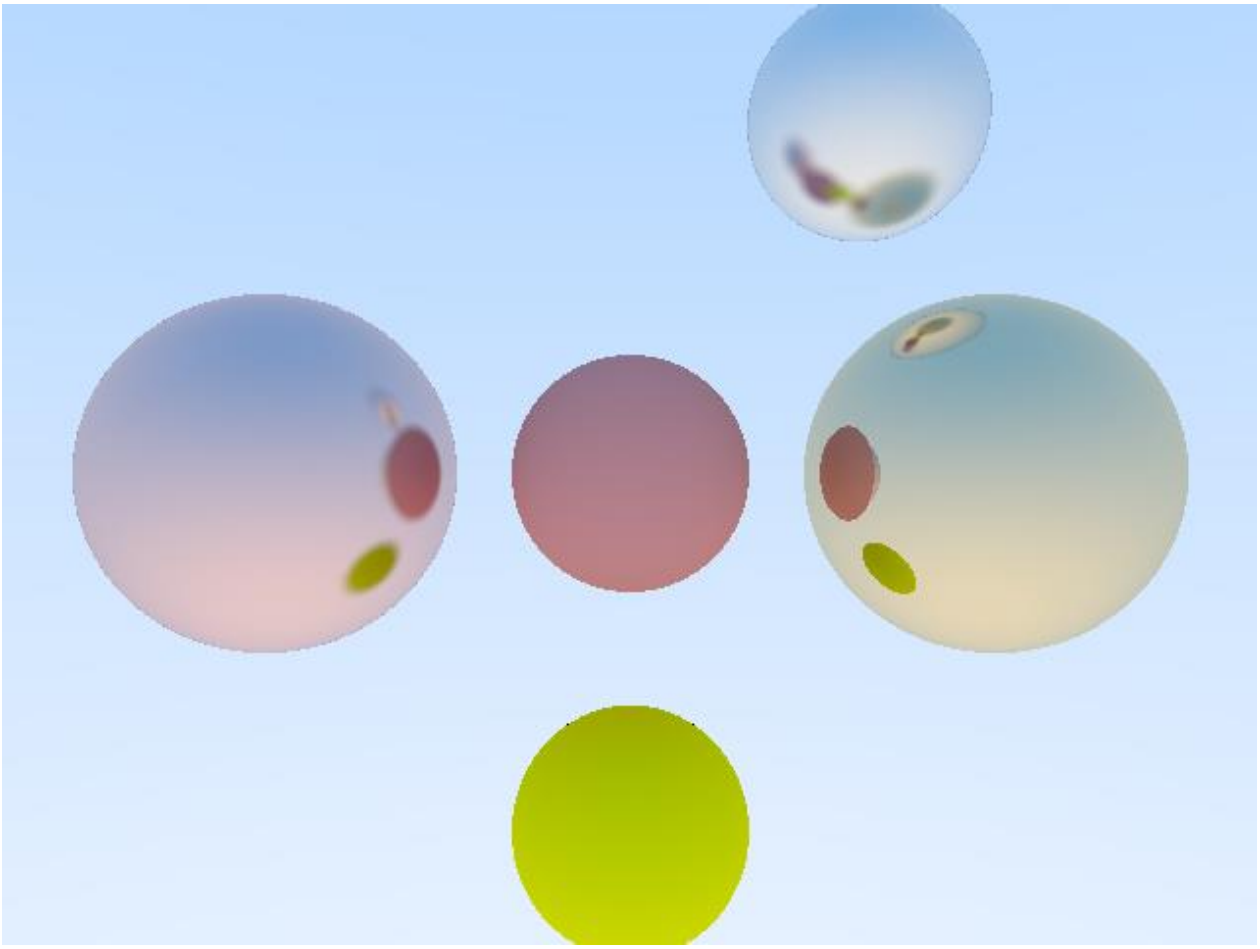


Figure Uniform-Grid Super Sampling

- Because of some sizing problems difference between two Super Sampling Methods are not easily noticeable, but Random Super sampling is smoother, so I will choose Random Super Sampling. Reason may be that, introducing randomness distributes variables better than adding just constant numbers in uniform-grid super sampling. Super sampling made this render much smoother in edges of the sphere, which is easily noticeable.
- I am sorry that in word document details were lost to compare the images better, please look for ppm render images. Also for formulas of these methods please refer to the ray coloring part near the end of the code.

4)



Sphere positions and Radius are same as task 1.

Material Properties:

Left Sphere: Metal with color rgb: (0.8, 0.6, 0.6) and fuzziness of 0.1

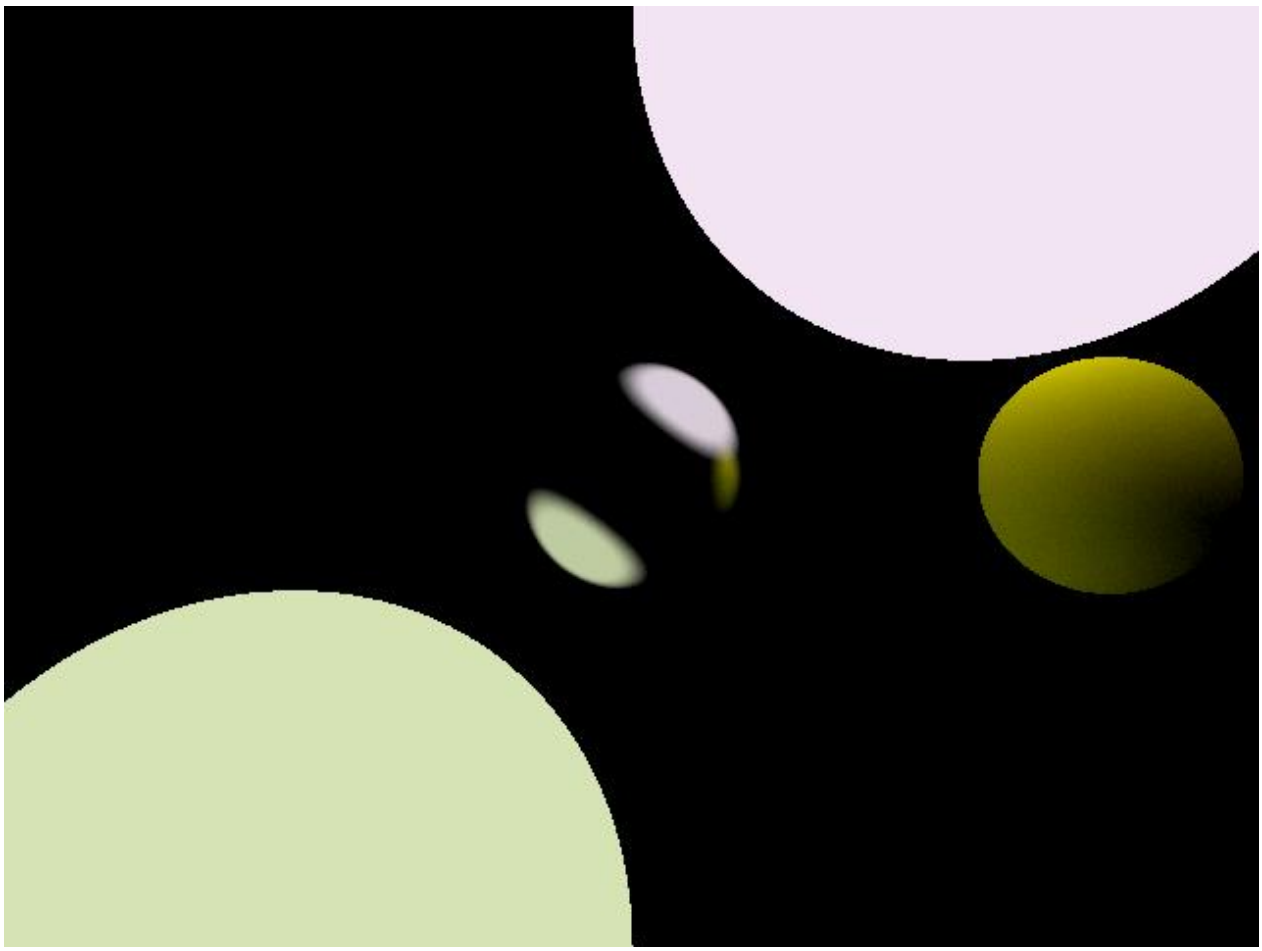
Center Sphere: Diffuse with color rgb: (0.7, 0.3, 0.3)

Lower Sphere: Diffuse with color rgb: (0.8, 0.8, 0.0)

Right Sphere: Metal with color rgb: (0.8, 0.7, 0.5) and fuzziness of 0

Upper Sphere: Metal with color rgb: (0.8, 0.8, 0.8) and fuzziness of 0.2

6)



Camera: Pos: (0, 0, 8),Look At Pos: (0, 0, 0)

Lights:

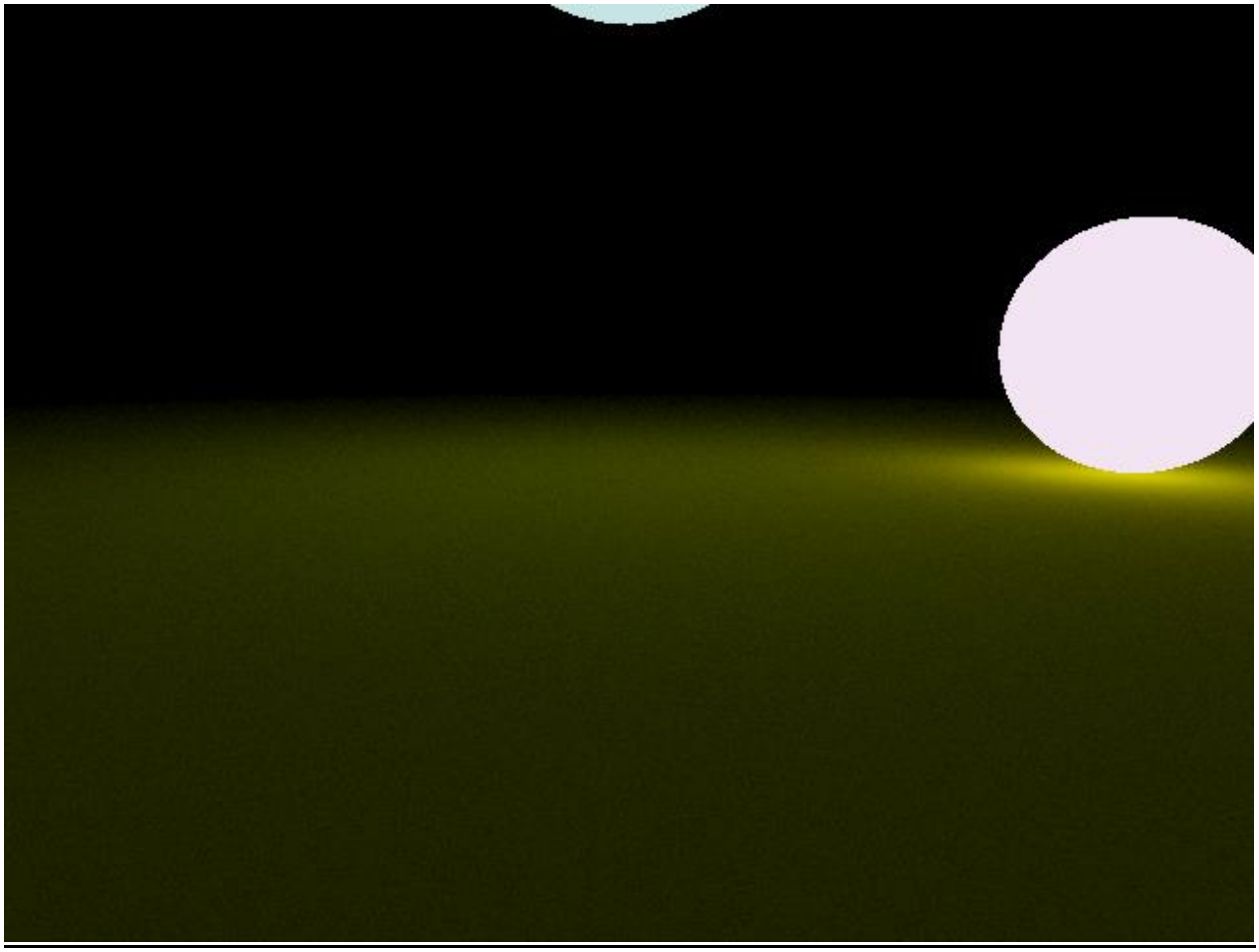
Pos: (-3, -4, 0),Radius:3, Color: (0.7,0.8,0.5)

Pos: (3, 4, 0),Radius:3, Color: (0.9, 0.8, 0.9)

Spheres:

Pos: (4, 0, 0),Radius:1,Diffuse with color rgb: (0.8, 0.8, 0.0)

Pos: (0, 0, 0),Radius:1,Metal with color rgb: (0.8, 0.8, 0.8)and fuzziness of 0.2



Camera: Pos: (0, 2, 15),Look At Pos: (0, 0, 0)

Lights:

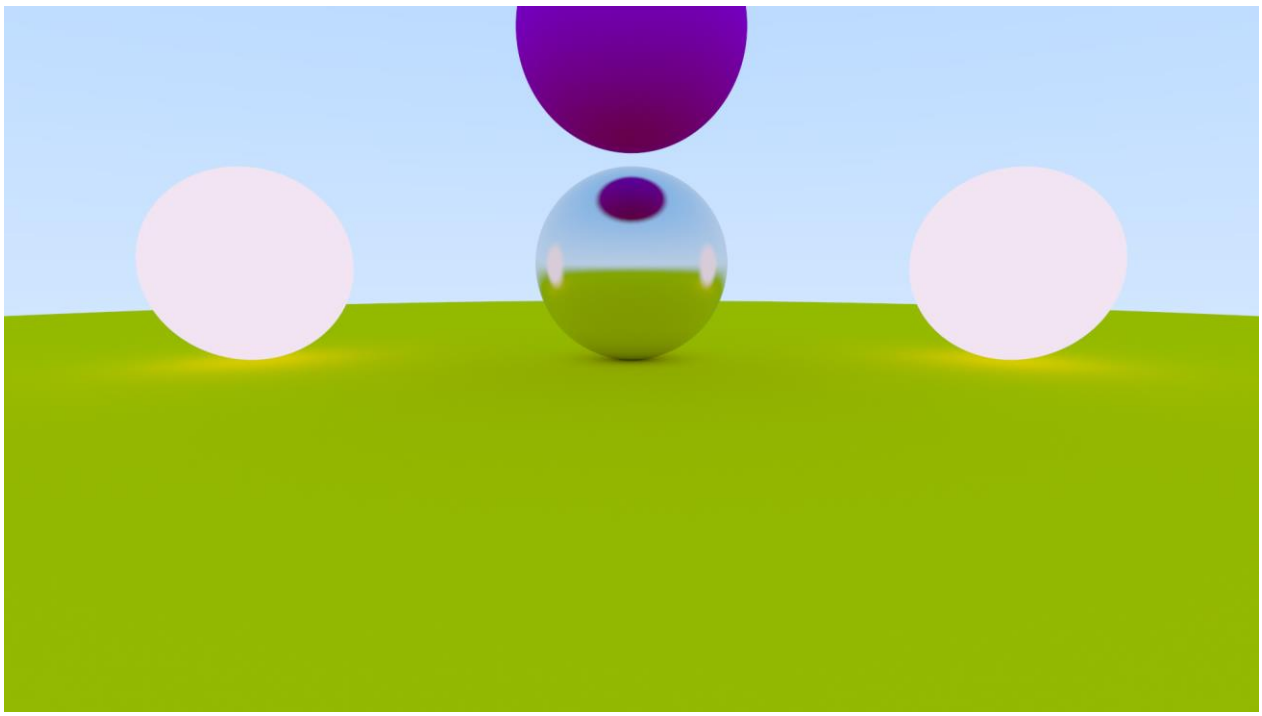
Pos: (0, 10, 0),Radius:3, Color: (0.6,0.8,0.8)

Pos: (8, 2, 0),Radius:2, Color: (0.9, 0.8, 0.9)

Spheres:

Pos: (0, -1000, 0),Radius:1000,Diffuse with color rgb: (0.8, 0.8, 0.0)

7)



This render shows an abstract scene of one purple sphere dropping from the sky, about to crash into metal sphere in the middle. A perfectly symmetrical scene is about to be completely altered. Light spheres on the left and right complement this symmetry. Sky is Clear and grass is great for playing outside, it is a shame metal sphere's day is about to get ruined. In this render you can see reflections, opacity, lighting and shading in just one scene. This is why I think it is important.

8)

a) Regardless of the traced objects' shape, in ray tracing, rays are sent everywhere, to each pixel. So, asymptotic time complexity is $O(b \cdot p)$, b being number of objects in the scene

(Number of triangles in this case)

and p being total number of pixels on the screen. We count the objects as well because rays may be reflected from these objects as well. So it only depends on number of pixels and number of objects. For accelerated ray tracing complexity is $O(\log b \cdot p)$, a bit faster but it is still a slow algorithm.

b) Image preprocessing prepares the image to be processed. Methods done in this step include; Color transformations, sub sampling, scaling, noise reduction filtering etc. Regarding image computing, actual processing of the image is done after preprocessing. For example, for ray tracing actual tracing of rays and their reflectivity, emission, shading, color calculations are done in this step.

c) Number of pixels is important because we send rays to every pixel. Number of objects in the scene is also important because these objects may emit rays or reflect rays that produces more rays, consequently requires more time to run. As an extra, computation of shading and texturing algorithms may also affect the overall performance of the ray tracing algorithm.

d) Time will be asymptotically dependent on resolution (number of pixels) times number of objets of the scene times number of triangles at the scene. If we use accelerated ray tracing full complexity in this case is 6000*6000*500*log(5000000). It will take very long time even for one frame, so there is a need for tremendous computing power.