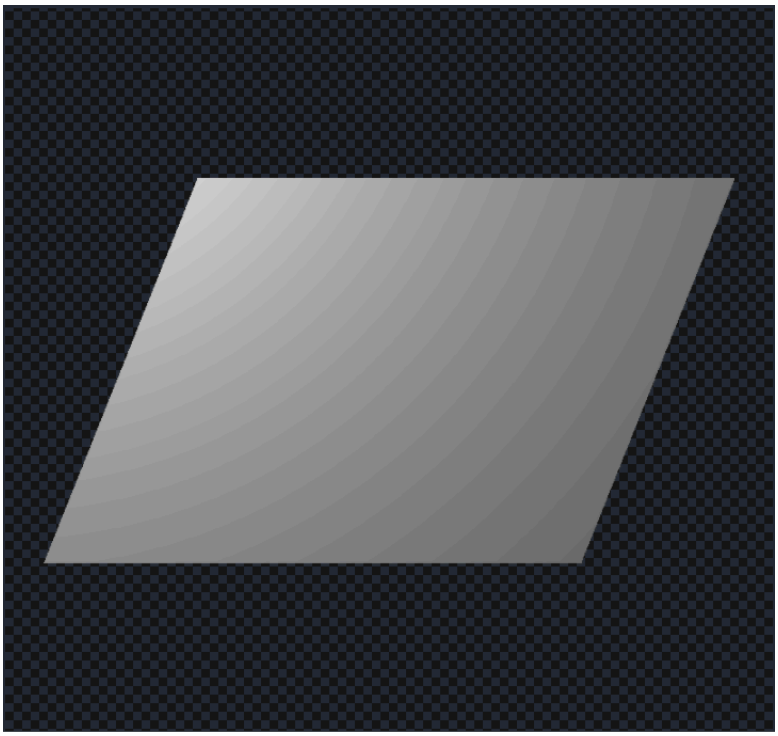
# *Assignment 2 Report*

## Ex.1

I set the parallelogram located at plane z=0. Then I created a function "ray_parallelogram_intersection" to check if a ray intersects with an arbitrary parallelogram. The input is ray origin e, ray direction d, parallelogram origin a and two edges u and v. I use carmer's rule to compute the intersection, and judge if it's valid accroding to its value.

```
bool ray_parallelogram_intersection(Vector3d &e, Vector3d &d, Vector3d &a, Vector3d &u, Vector3d &v) {
    Matrix3d M;
    M << -u(0), -v(0), d(0),
         -u(1), -v(1), d(1),
         -u(2), -v(2), d(2);
    Vector3d res = a - e;
    t = M(2,1)*(M(0,0)*res(1)-res(0)*M(1,0))+M(1,1)*(res(0)*M(2,0)-M(0,0)*res(2))+M(0,1)*(M(1,0)*res(2)-res(1)*M(2,0));
    t = -t/M.determinant();
    if (t < 0) return false;
    gama = M(2,2)*(M(0,0)*res(1)-res(0)*M(1,0))+M(1,2)*(res(0)*M(2,0)-M(0,0)*res(2))+M(0,2)*(M(1,0)*res(2)-res(1)*M(2,0));
    gama /= M.determinant();
    if (gama < 0 || gama > 1) return false;
    beta = res(0)*(M(1,1)*M(2,2)-M(1,2)*M(2,1))+res(1)*(M(0,2)*M(2,1)-M(0,1)*M(2,2))+res(2)*(M(0,1)*M(1,2)-M(1,1)*M(0,2));
    beta /= M.determinant();
    if (beta < 0 || beta > 1) return false;
    return true;
}
```

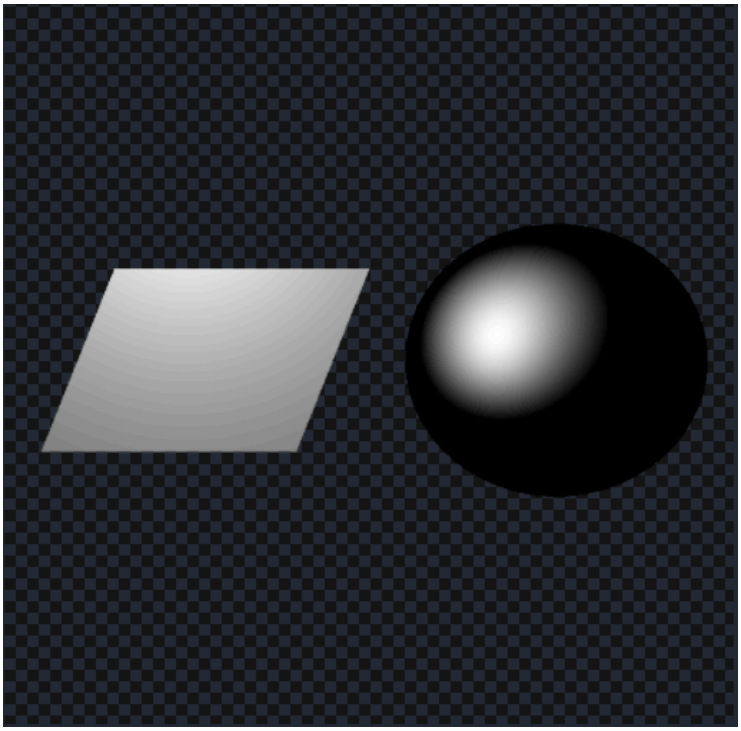
Using the function above, I got the result of parallelogram in orthographic rays as follow.



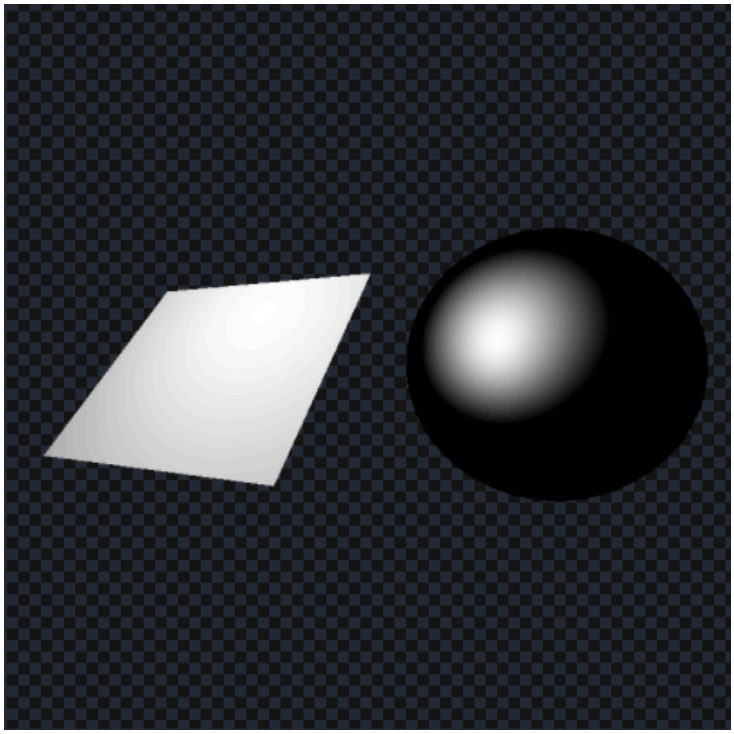
I  also created a function to check if a ray intersects with an arbitrary sphere. The input is ray origin e, ray direction d, sphere center and sphere radius.

```
bool ray_sphere_intersection(Vector3d &e, Vector3d &d, Vector3d &c, double r) {
    double A = d.transpose() * d;
    double B = 2 * d.transpose() * (e - c);
    double C = (e - c).transpose() * (e - c) - r * r;
    double delta = B * B - 4 * A * C;
    if (delta < 0) return false;
    t = -B - sqrt(delta);
    return true;
}
```

In the perspective raytrace, if the parallelogram plane is parallel with the screen, then no matter how we move the perspective the shape won't change (only size will change). However for sphere, if we change the perspective, the shape we get will change, too. As we can see following, the parallelogram is still a parallelogram, but the projection of sphere change from circle to ellipse.

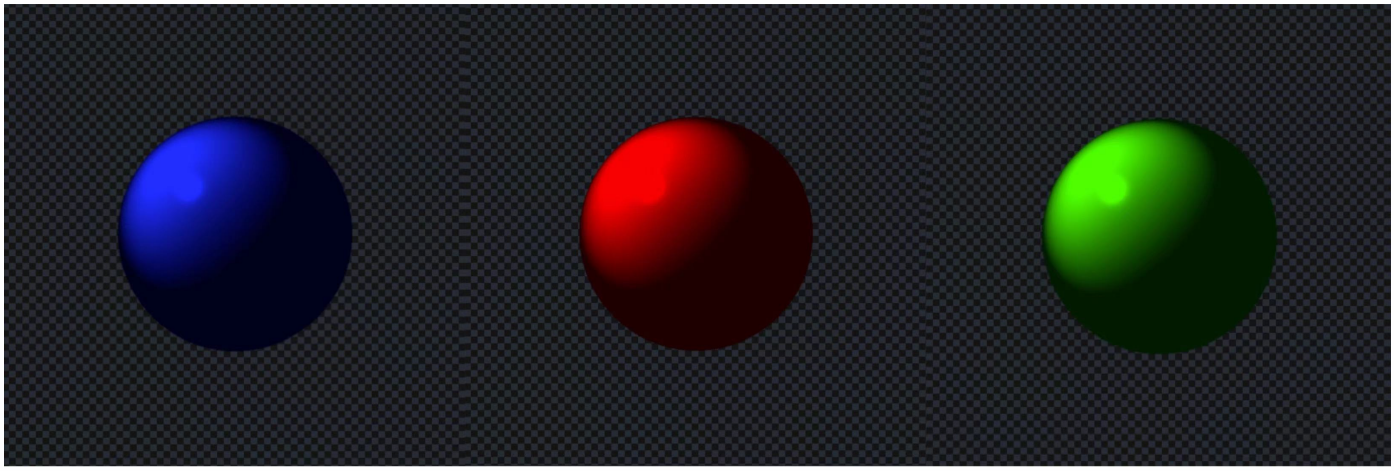


Be careful, that doesn't mean for plane, the projection won't change shape in perspective raytrace. If the plane is not parallel with the screen, then the projection will change too, as follow.

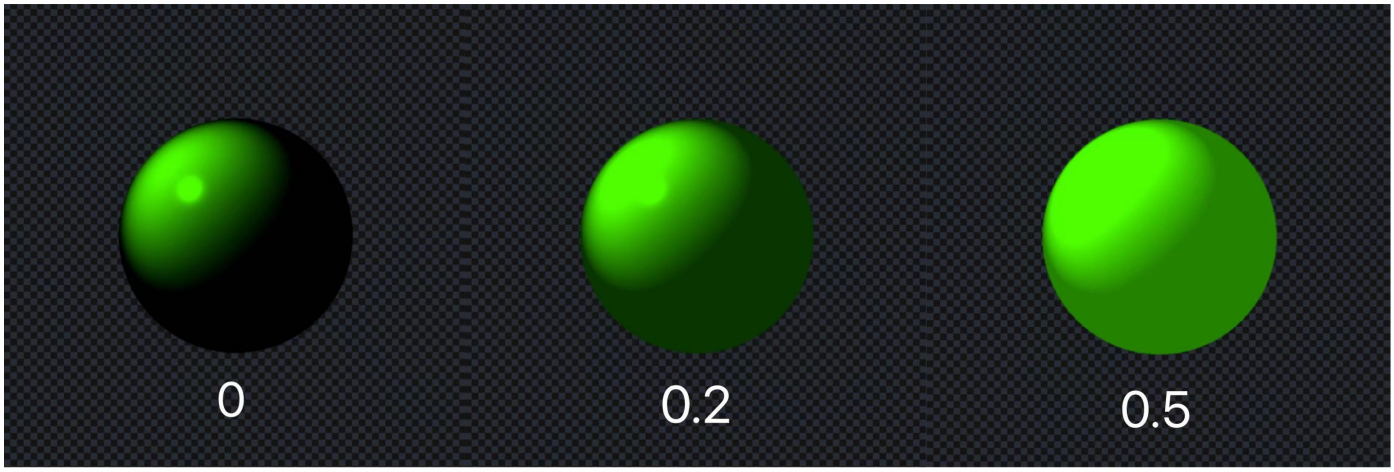


## Ex.2

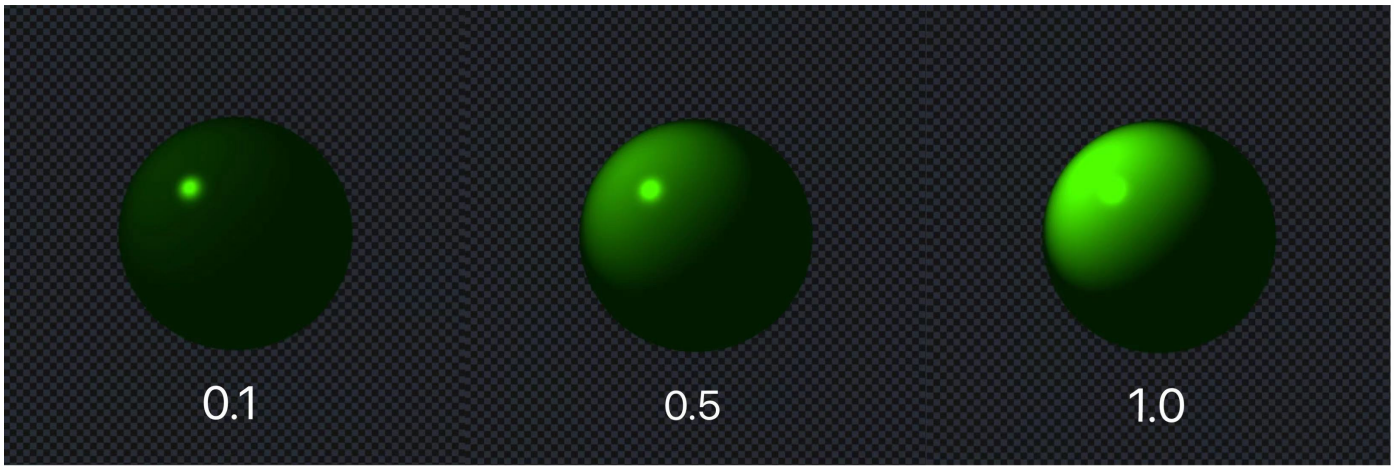
In the exercise 2, I make use of the "ray_sphere_intersection" function defined above to compute the perspective raytrace of an arbitrary sphere. We can change the RGB components to get different colors.



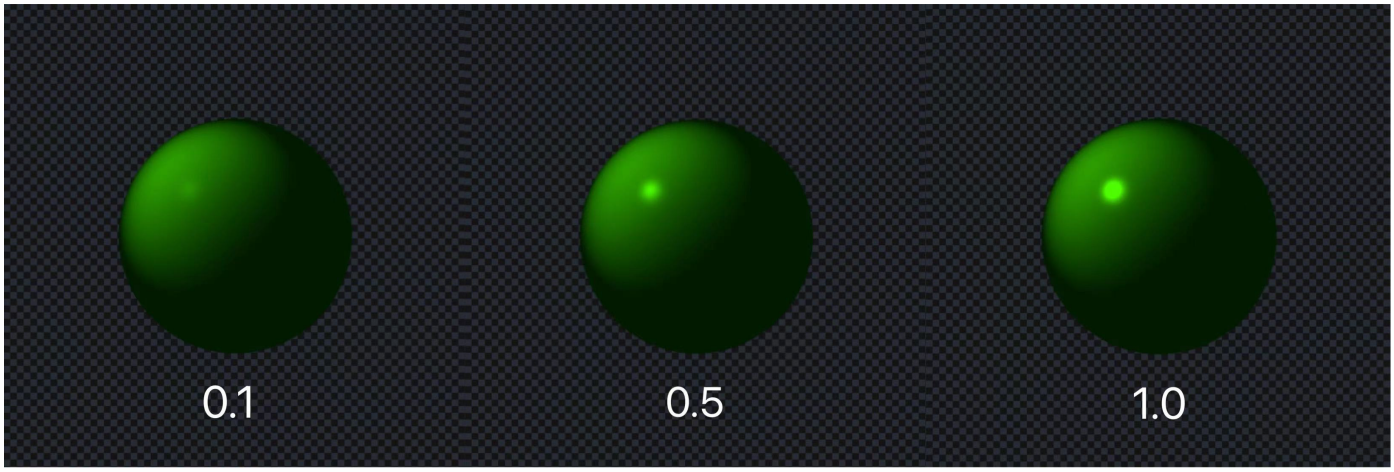
Ambient is an important parameter deside the light of the dark area(the shade under the light source). With higher Ambient parameter, the shade area become brighter.



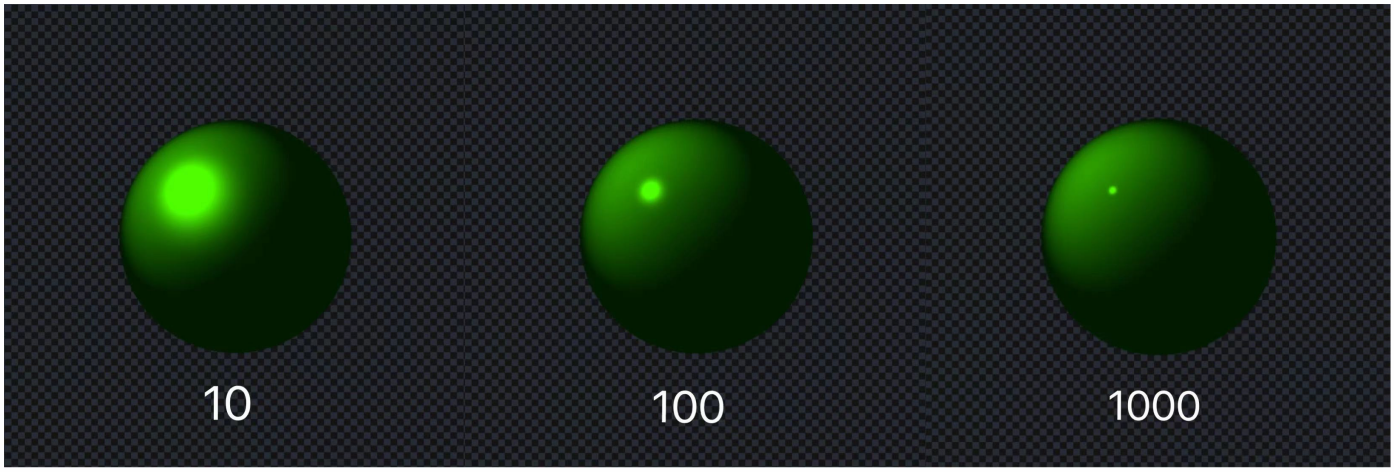

The diffuse coefficient decides the brightness of the area lighted by light source.



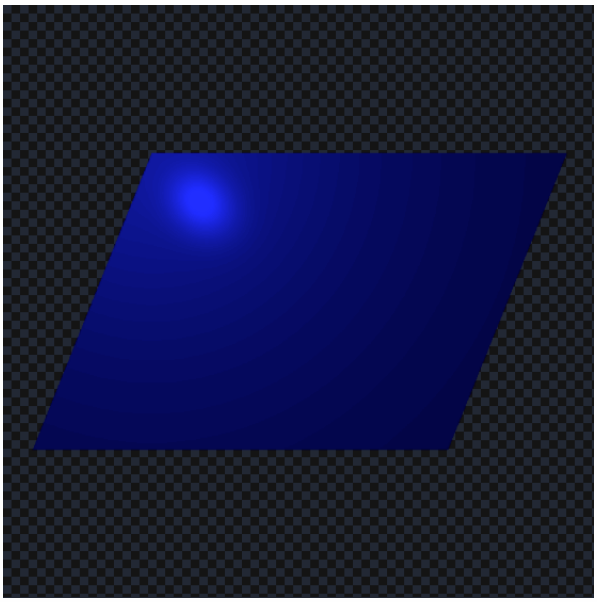

The specular coefficient decides the brightness of the highlight area.

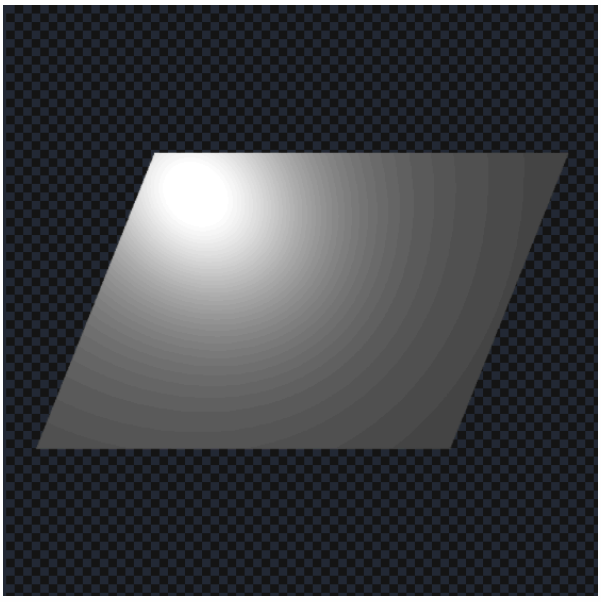



And the phong exponent decides the area size of highlight spot.




I also make a parallelogram version shading, the parameters' influence is just as same as sphere one so I won't paste too many pictures.





## Conclusion

In this assignment, I learned the basic ideas of how to implement different raytrace, including orthographic one and perspective one. I got the basic idea of shading, and the influences of different parameters.