University of Victoria

Department of Computer Science

CSC 305: B01

Assignment #3

Ray Tracing Advanced

Date of Report Submission: Jan. 30th 2017

Submitted to:

Nicolas

Submitted by: April 9th

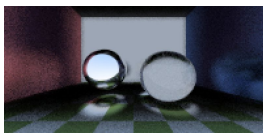
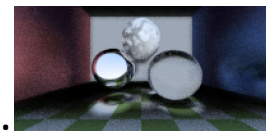
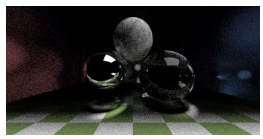
Ray Lin Honghu V00804639

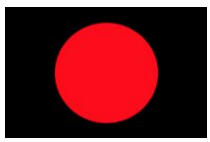
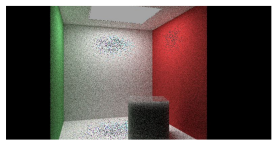
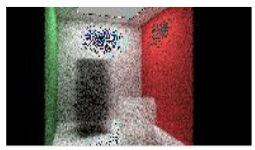
Objective

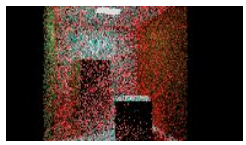
To familiarize C++ programming language. To understand the concept of ray tracing and rendering. We will learn how to use our knowledge from class to create features of images. This entails analyzing and building the basis of materials and meshes. Specifically, we learn how to make ray tracing stuffs such as motion blur, BVH, perlin noises, and etc.

Discussion

List of Figures:

1.  **2.**  **3.**  **4.** 

5.  **6.**  **7.**  **8.** 

9.  **10.**  **11.** 

In this Ray Tracing Assignment, the bonus I have implemented are: a standard vector class with simple operations / rendering a sphere / rendering a plane / rendering multiple spheres using concept of listing / positional camera/ Motion blur/ BVH/ solid texture/ perlin noises/ image mapping/ volumes and some other advanced requirements which will all be discussed in later report in detail with figures provided above.

If we want to have more objects, instead of creating many similar instances, we could have an abstract data type called hittable and hitable_list which featurize the information when we have several instances. Each instances such as sphere and planes can inherit from the class hittable so that the derived objects would have a hittable feature to themselves. Values like normals and valid hit intervals are recorded in structure hit_record, to indicate the closest hit object and the range which is valid to count as hitted. The camera class is used for locating and changing the perspective and angles to look at things. we can average the pixels to achieve the goal of antialiasing. For each ray (u,v)

For a diffuse material lights can be reflected randomly and absorbed more when hitting a dark surface. The ray bounces, let's say a point reached, the ray would be rejected at that point, and re-attempted if it is not in the range of the sphere. So I have ranndom_in_unit_sphere and drand48()

to construct when a vector hits a target. As it can be seen from figure 2, there are shadow underneath different material of spheres. It proves the specular Component is related to ambient, angle, and coefficient of diffuse illumination. The material is modelled by a class which has a function called scatter. The structure hit_record now also owns a pointer to a value type of material. The material class tells how vectors behave with different make-ups. Lambertian, dielectrics and metal are inherited from material class. Because a point rotated around its normal vector would not change the way it reflects light, yet the point will modify the way it reflects light in case the angle is different from initial normal vector. This is because the area is illuminated by a smaller fraction of the incident ray. In physics we know that when a light hits dielectrics, it would split into reflected and refracted ray, given Snell's formula n sin (theta) = n0 sin(theta0).

Different material shines differently, for example small specular power has low reflect value, high specular power has high reflect value. In dielectric not only we need to take consideration of normal, reflected and attenuation, also we need to get polynomial approximation by Schlink if we really want to get a nice proximation for glass material.

For getting a light object we need to create a class called diffused light which is a child class of material. For a ray p = vt + u, we need to know at what time it will hit the axis we want. given by formula x = ax + t*bx, and y = ay + t*by.so from inherited class xy_rect we store the coordinates from which it hits the objects and output light.see figure 6.

The basic idea of motion blur is to generate rays at random times so in the ray file we have time valuable in ray class. The key idea of a bounding volume is to find a volume that fully encloses all the objects. solid texture is created by replacing the vector color with a texture pointer the finish picture is like figure 4. Perlin noise was implemented in assignment 1. It is also doable to read in an image and use a 2d texture coordinate to index into the image.see figure 5. Another feature that is adding subsurface scattering, see figure 8.

Conclusion

In conclusion, we reinforced our knowledge on C++ language and object oriented programming, and how to inherit class from one to the other. Physics and mathematical modelling are well understood and used from the first assignment. Especially Linear algebra really helped me to construct some objects. class and structs also two dimensional array making my life easier to implement some methods.

References

- UVic 305 Lecture Sides, UVic Lab materials
- Ray tracing in one weekend
- Ray tracing next week