PHYSICALLY BASED RENDERING

ARDEN RASMUSSEN

Abstract. The aim of this paper is to introduce the basic concepts of ray tracing, and lead the reader thought he entire process to construct a ray tracer from scratch. Once that has been achieved we will begin to introduce more involved components and develop a fully functioning path tracer that implements more complex features, such as BSSDF material shading, animated transformations, volumetric rendering, and other features. Any features that are our of the scope of this paper will be mentioned, and the structure will be such that additional features can easily be implemented.

1. Introduction

Physically based rendering is the method used for rendering high quality images based on reality, and through the use of simulations physical iterations of the light with the objects in the scene.

2. Mathematics

Before any real work can be done, it is important to develop a strong linear algebra and geometry library. This section skims over the details, as many tools can be used to implement a linear algebra library, but there are some necessities that our implementation depends upon, so we specify the implementation details that are required by the reset of the system.

For `C++` a library that implements almost all of the requirements is `glm`[1]. Although if one uses this, it is important to be careful with the different types of vectors, and template overloading can lead to unexpected issues.

2.1. Real Numbers.

The first component to implement is the real number type. The real number type (or floating point type) defines the maxim accuracy of the renderer. For example if the floating point type can only guarantee three digits of precision, then any computations will have relatively large errors. We will implement this type called `Float` like so.

```
#ifdef SPECULA_DOUBLE_PRECISION
  typedef double Float
#else
  typedef float Float
#endif
```

Date: March 4, 2020.

[1]https://github.com/g-truc/glm