Rastislav Kučera

General representation of light in 3D

1 Light Source

How to store light source information in computer ?

The project is based on Rendering equation :

$$L_o(\mathbf{x},\,\omega_o,\,\lambda,\,t) \,=\, L_e(\mathbf{x},\,\omega_o,\,\lambda,\,t) \,+\, \int_\Omega f_r(\mathbf{x},\,\omega_i,\,\omega_o,\,\lambda,\,t)\, L_i(\mathbf{x},\,\omega_i,\,\lambda,\,t)\,(\omega_i \,\cdot\, \mathbf{n})\;\mathrm{d}\,\omega_i$$

From it, we determined that all light sources should have method *getNextBeam()* from which you can get information „from where to where is beam going“, some information about its color and which light source emitted it. This is how this project tries to model different types of lights.

Method *getNextBeam()* in LS[1]es implemented in this project works this way:
1) generates coordinates [x,y,z] – beams origin
2) generate direction
3) generate wave length – from which we will deduce color

We generate wave length and no RGB values, because this way, we can easily produce different optical effect just by adding know physical principles into renderer. Down side of this is that it introduces need for SPD[2]s and little bit of Color science. ~~This is why every Light Source has besides *getNextBeam()* , Constructor taking SPD and *get/setSPD* methods.~~

Method *getNumberOfBeams()* is needed for color (power on camera) deduction.

1- LS : Light source
2 – SPD: Spectral Power Distribution

2 Spectral Power Distribution

Spectral Power Distribution or SPD for short, represents how much energy is illuminated/absorbed/... on each wavelength.

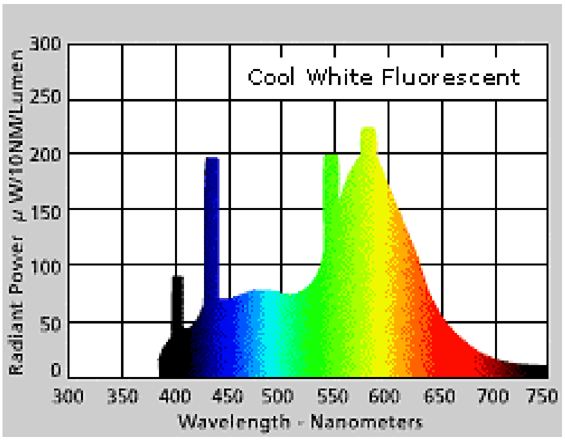

Illustration 1: SPD of flourescent lamp, source: http://www.lrc.rpi.edu/education/learning/terminology/spectralpowerdistribution.asp

In this project, SPDs are used for color deduction and wavelength generations. All SPDs integrals should be equal to 1.

In our *interface SpectralPowerDistribution* , we have method *getNextLamnbda()*, which should use SPD for „weighted random distribution“ of lambdas (wavelengths) - wavelength generations.

Then it has *getValue(double lnb)*, which should return SPDs power on wavelength lmb – used for color creation with power obtained form LS.

3 Color

Human eyes have 3 cones (S,M,L) which perceive color. They are sensitive to different, overlapping parts of visible light spectrum, and are more sensitive to different wavelengths in their corresponding part of light spectrum. This means that color does not depend only on which wavelengths are hitting your eye, but also on how many fotons with same/different wave length are there.

This is why there are things like CIE standard observer, and why we have *interface Color* with *SPDtoRGB(SPD a)* method. It takes SPD and outputs RGB color using ColorScience - it calcualte XYZ from SPD, using cone sensitivity defined in specific implementation of *Color*, and then it converts XYZ to RGB.

4 Math

~~Most of math functions for 3D are from THE INTERNET, and I haven't bothered to find time to understand them.~~

5 Non uniform distributions

How to turn uniform random distribution to nonuniform random distribution ?

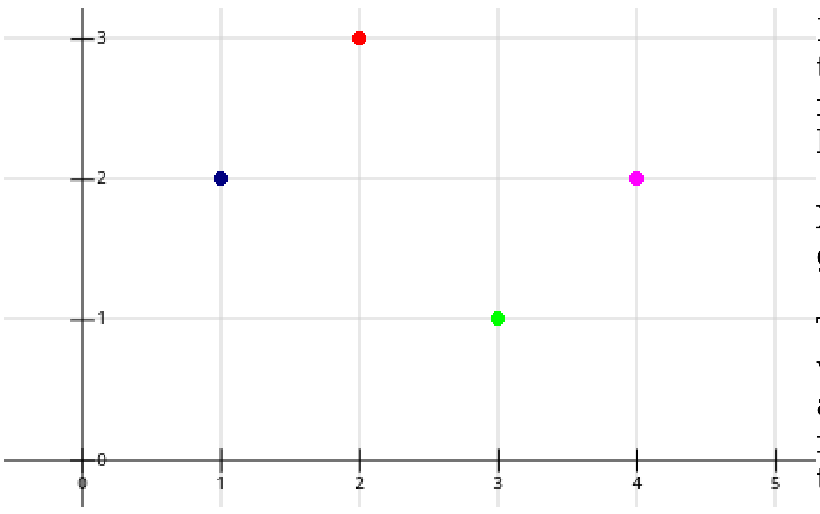

Illustration 2: some samples

In many cases, we want to create weighted random distribution based on some samples.

X is what we want to get, Y is its weight.

That means that we want same amount of *1* as *4*, but only half as many *3* as *1* (or *4*) and three times more *2* than *3*.

Easy solution is to compute the ~~integral~~ sum of functional values form 1 to 4 of this sample, which is 8, and then produce random number η form [0,1) , multiply it with found ~~integral~~ sum and then sum functional values until our sum is > η*8. Our random number from this weighted distribution is the last X used in sum.

Generalization:

We have samples from set {a,…,b} with values of f(x), x ∈ {a,…,b} .

1) Calculate ***I*** = $\sum_{n=a}^{b} f(n)$

2) Produce ***i* = *η* * *I*, *η*** is random number form [0,1)

3) Find greatest ***r*** for $\sum_{n=a}^{r} f(n) < i$ where ***r*** is our result

Problem with this solution is that it creates "Choppy" distribution,and if we want to fill gaps between two adjacent elements ***k,l*** ∈ {a,…,b} so that their f(x) wold be ,for example, linearly changing, we have to add more samples between k,l , what add more iteration. This is applicable also to continuous functions using integrals instead of sums, but such integral is usually too hard to solve analytically for ***r***.

6 Renderer

Right-handed system, x – is horizontal (bottom of monitor), y is vertical, z is depth (exiting from monitor towards viewer is positive value, entering into monitor is negative value)

It will most likely be bidirectional path tracer, with SPD and Power as color representation which enables easy use of known light interaction rules and thus should result in easily created photo-like images.

Renderer will be composed of following elements:

Scene

Contains SceneObjects, Cameras (CameraManager) and lights (LightSourceManager), manages light/camera path tracing.

SceneObject

Something that can interact with beams (light/camera). Provides ray intersection function, has relevant material info for collisions

Camera

Can create image from detected beam data, Can create camera beams, Is aware of “master light source” - LightSourceManager

Scene should contain CameraManager, CameraManager contains all other sensors (Cameras)

Light source

Can create light beams with some color data (wavelength, power), knows how many beams created.

Scene should contain 1 LightSourceManager, which contains all other LS

Materials

Have material function which takes beam and returns new one