TINGE

Task 3

Programming Assignment and Reading Task

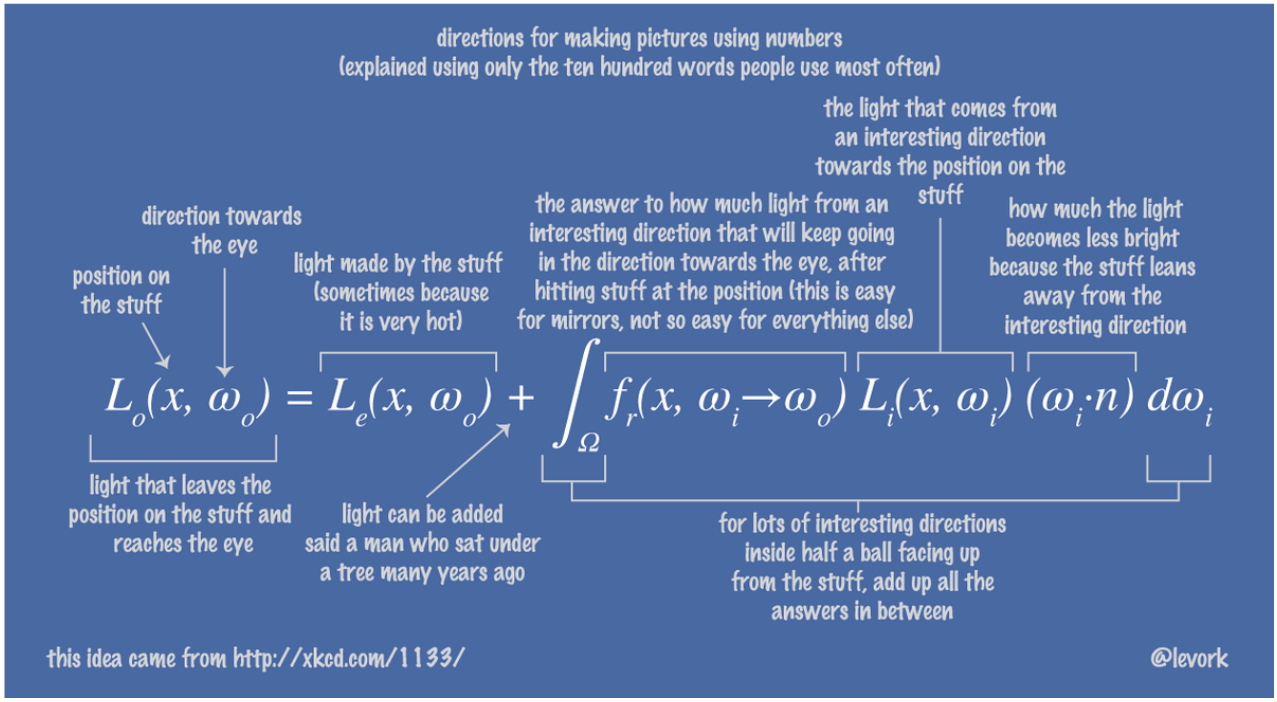
9 March 2025

Congratulations to the team for being able to create our own render! Pat yourself on the back for the amount of work you have done to achieve this. But, we are still far from being done with the project. Here are some more reading tasks and programs that you need to implement before the next meeting.

§1 The Rendering Equation

Recall the Rendering equation that you have seen during the Tinge mini Task that you were assigned. We will be working on this for the remaining part of the tenure.

For our first render, we had just equated the color of the pixel to be equal to the normal of the object that the ray had collided with. Now, we need to be able to actually find the color. This isnt as straight forward as you think: It depends on the material properties as well as the direction of the ray and such. This is the rendering equation. Take a look at this, the image itself is quite informative:



Here are other sources that I want you to refer to along with this:

1. Wikipedia page.
2. A Reddit page under 3Blue1Brown.
3. A good document by Technische Universität Wien giving the intuition behind the equation.

Now that you would have understood how the rendering equation works, we now need to implement this in code. The equation involes integration, so we need to look into Numerical Methods of how we can implement integration: Hence, we look at *Monte Carlo Integration*.

Read about Monte Carlo Integration from here:

1. Wikipedia page.
2. GeeksForGeeks showing python implementation, but it's easy enough to understand this for c++.

§2 A New Class: Random

If you're done reading about Monte Carlo Integration, then you know that we need to choose random points and operate on them. For doing this conveniently, you need to write a class `Random` which has the following static functions:

1. Init: Initialises the seed for the random operation.
2. GenerateUniformFloat: Returns a float which is uniformly distributed in the range of $[0, 1]$.
3. GenerateUniformPointDisc: Given a $2D$ unit disc centered at origin, returns a uniformly distributed point on it.
4. GenerateUniformPointSphere: Given a $3D$ unit sphere centered at origin, returns a point uniformly distributed on the surface of it.
5. GenerateUniformPointHemisphere: Given a $3D$ unit hemisphere centered at origin, returns a point uniformly distributed on the surface of it.

For doing this, you can use the std random lib in c++. Read about *Mersenne Twister* and use this.

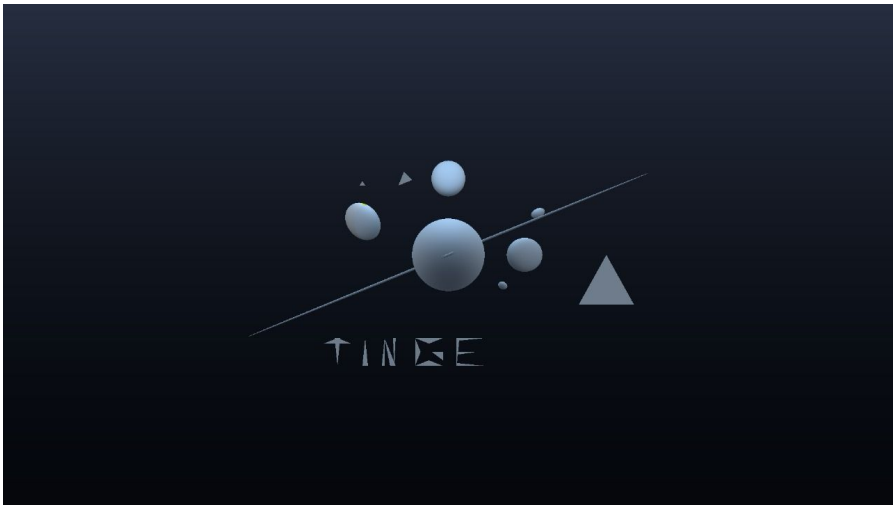
§3 Additional Task

A code is called good code when it's well documented: So far our code isn't. Instead of doing it all at the end, you now need to work on documentation of each part of code as and when they're written. You can use external tools for this: `doxygen` is what we ask you to use.

Read about how you can document using doxygen. We only typically use the tags `@class`, `@struct`, `@brief`, `@param` and `@return`. Any extra tags that you can add is up to you.

Try documenting all the parts of the code that you have written before the next Project meeting.

If you have any query about this assignment or anything about these topics in general, feel free to text us. Also let us know when you are done implementing this, you will get some brownie points for doing so 😎. Till then, enjoy this amazing render that Nuthan has made:



Good luck on your code! Here's KK signing off~