SEEING COSMIC RAYS

Program description

In our simulating program (attached) we analyse mensurability of Cherenkov radiation by humans. To be capable of doing the analysis we are doing, the most challenging thing was to represent the surface of the retina . After taking into account all of the problems, we chose BST (Binary Search Tree) as a structure well optimised and best suited for mathematical analysis of the flat surface (Cartesian Coordinate System). All calculations are nessesery because, we need to know how many photons reach the "visable" area of our eye. In fact one unit of light is insufficient to eliciting an eye response. After compering quantity of light's units (photons) with theirs properties (such energy, wavelength, fosusing ect.) we are able to predic which of them will be visable. What in fact is BST? Referring to Wikipedia: "BST is a particular type of containers: data structures that store "items" (such as numbers, names etc.) in memory. They allow fast lookup, addition and removal of items, and can be used to implement either dynamic sets of items, or lookup tables that allow finding an item by its key (e.g., finding the phone number of a person by name)."

How it really works?
If we take two variables which will represent point in Cartesian Coordinate System, we are able to describe the entire surface. Then we register points where photons reach the retina and count number of that events. If the quantity is sufficient to notice, we will return co-ordinates of the point or just find that event occured.

A few more words about our implementatnion
Our tree is built from items called "nodes". Nodes are correlated hierarchically with each other. That kind of connections is called "edge'. Depends on importance we distinguish parent node (more important) and child node (less important). In our case co-ordinates are nodes and adjacent points is our edge (or even fact that they are next to each other). Structure operating this way is easy to use and control. The last but not at least important is fact that it is several times faster than most other solutions.

Retina Simulator

1. **Introduction**
   During our research we found it necesary to simulate the behavior of human eye's retina in order to state whether photons created in vitreous body are able to cause significant reaction in brain. That's why we decided to create C++ program (available in attachment).

2. **Main concept**
   The program works as follows: Eye's retina is represented by the common part of sphere with radius R and center in point O(a,b,c), where R is rational non-negative number and a half-space determined with inequality $Ax + By + Cz + D > 0$ there A, B , C, D are rational coefficients and x, y, z are coordinates of a point in Euclidean space.

$$\begin{cases} R = \sqrt{(x-a)^2 + (y-b)^2 + (z-c)^2} \\ Ax + By + Cz + D > 0 \end{cases}$$

Each photon is represented with its wavelength as it is required to describe photon's wave nature and with ray structure containing: direction unit vector describing the direction of photon propagation and origin vector which represents the point considered to be the source of the photon, relatively to Euclidean space center.

3. **Finding the point of intersection**

   With such an implementation we are able to trace the route of photon from the starting point A, which allows us calculate a point of intersection of photon and retina. This task is done by a following algorithm:

   (a) We need to find coefficient t, which satisfies set of two equations:
   $$\begin{cases} (x-a)^2+(y-b)^2+(z-c)^2=R^2 \\ x_v - x_u = x \\ \mathrm{y}_v + t y_u = y \\ z_v - z_u = z \\ Ax+By+Cz+D>0 \end{cases} \qquad (*)$$

   (b) After substitution, we receive a quadratic equation:
   $\alpha * t^2 + \beta * t + \gamma$
   where
   $\alpha = x_u^2 + y_u^2 + z_u^2 = 1$
   $\beta = 2(x_u(x_v - a) + y_u(y_v - c) + z_u(z_v - c))$
   $\gamma = (x_v - x_u)^2 + (y_v - y_u)^2 + (z_v - z_u)^2$

   (c) Now we can find a discriminant of the equation, which determines number of solutions:
   $\Delta = \beta^2 - 4\gamma$ and if
   $\Delta > 0$ - two solutions
   $\Delta = 0 - onesolution$
   $\Delta < 0$ - no solutions

   (d) If discriminant is positive, we are able to calculate solutions identified with:
   $t_1 = \frac{-\beta-\sqrt{\Delta}}{2}$
   $t_2 = \frac{-\beta+\sqrt{\Delta}}{2}$

   (e) Once we have our solutions, we have to check whether they are located on retina's surface - this part is done by checking the last inequation in set (*).

   In case photon has intersected retina, the program projects the point of intersection on plan, with distance between particular points conserved.

4. Calculating probability of collision with photoreceptor Basing on medical studies we can define an avarage density of photoreceptors on particular piece of retina, however, according to data from research. With satisfing accuracy we can interpolate polynomial function describing density in relation of distance **r** from center of the eyeball . Now we can find

number of photoreceptors **N** in given surface, calculating:

$$N = \int_{r_1}^{r_2} f(r) dr$$

Knowing the average photoreceptor size S we are able to calculate probability of collision with photoreceptor, determined with:

$$P(r_1, r_2) = \pi \frac{r_2^2 - r_1^2}{N * S}$$