The human eye – an evolutionary look

By Ido Weisberg - 33905209

Introduction

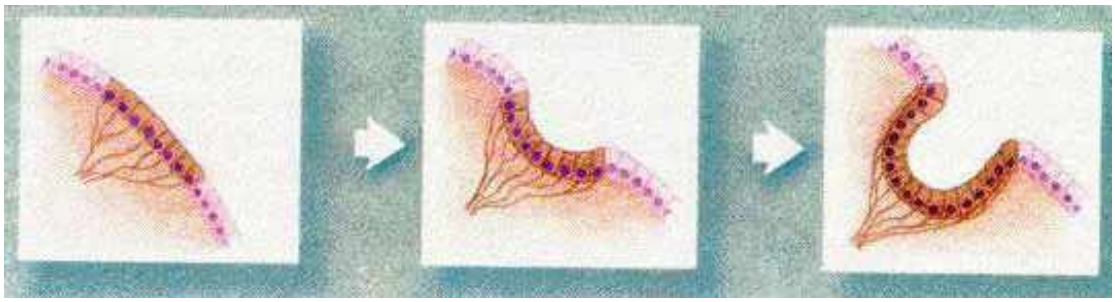
The goal of this project was predicting the number of generations required for an eye to evolve.
In my work I tried a different approach than the one described in "A pessimistic estimate of the time required for an eye to evolve" by Nilson and Pelger. What they did was look at one eye and apply on it a small change every generation, until a final eye has evolved. What I did was use a genetic algorithm on a population of individual eyes, until I got the desired eye. A sort of Darwin principle: the strongest eye survives.

Some basic theory

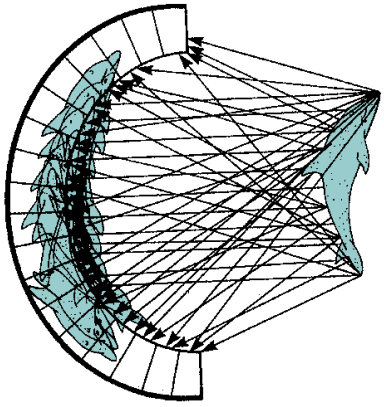
There are two parameters (which I dealt with) that make one eye fitter than another: the first is the shape of the eye and the second is its lens. The shape of the initial eye is a flat patch of light-sensitive cells sandwiched between a transparent protective layer and a layer of dark pigment. The eye naturally wants to achieve better direction resolution. This is achieved by two successive operations:

- Forming a depression in the patch
- Closing the “hole” formed

The first change can be seen in the next pictures:

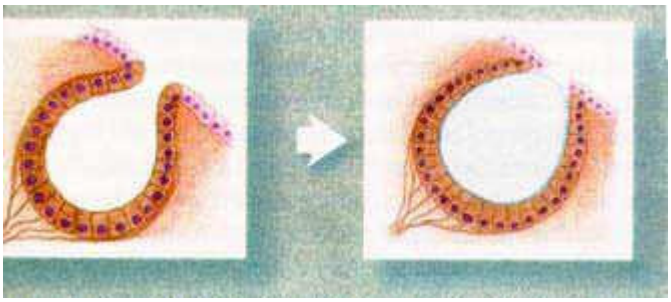


This deepening reduces the angle at which each cell receives light. Now we know where the light is coming from, but we still have a problem:

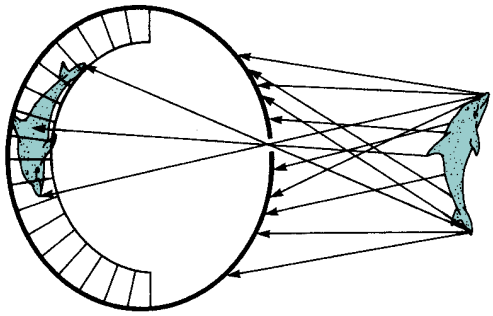


The problem comes from the fact that each point in the object (here - the dolphin) sends light to many different cells on the retina. We know that the object is on our right side, but we're not exactly sure where.

To solve this problem the "hole" is constricted, and a pin-hole eye is formed:



And so the object becomes much clearer:

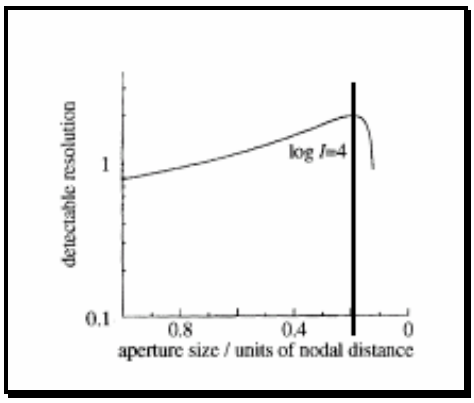


Experiments show that when the pit depth equals its width, deepening is no longer helpful, and closure of this pit is more important. This means that first we have the depression and after that the closing of the hole. These operations form our pinhole eye.

But there is a limit to the size of the opening. As this opening gets smaller the optical image becomes well resolved but also much dimmer, which leads to noises. This noise comes from the capture of photons. When a photon is captures it makes noise, and when the image is dim, the relative noise of this photon is large (compared to the object we are supposed to see), and this leads to a blur image. The theory of Snyder tells us that the maximum detectable special frequency is:

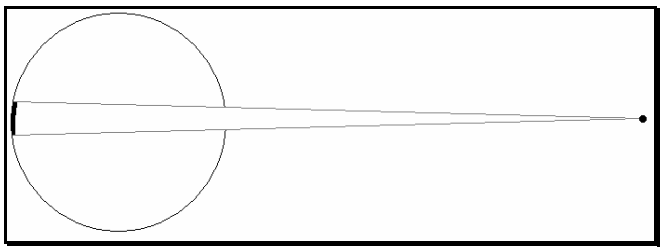
$$v_{\max} = 0.375\frac{P}{A}\sqrt{\ln(0.736A^2\sqrt{I})}$$

Where A is the diameter of the aperture (the opening) and P is the pit's depth. I is the light intensity, and therefore we can make it a constant. I'll assume *ln I* is 4, as suggested by Nilson and Pelger. The following graph shows us the resolution as a function of the ratio between the aperture and the depth of the pit:

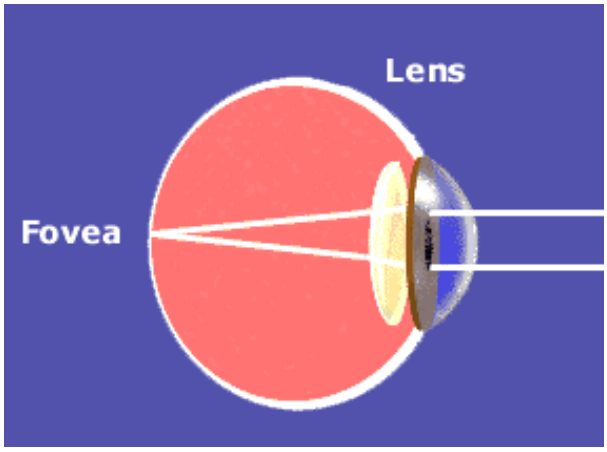


The maximum for this value will be obtained when the ratio between A and P is 0.19. This means that for a given eye size we have an optimum aperture. We cannot achieve a better picture by changing the shape of the eye and so we start developing the lens.

In a lensless eye each spot is blurred and has the size of the aperture:



A lens situated in the aperture will converge the light into one place, making the object sharper, without decreasing the brightness of the image, which would have happened if we tried to achieve sharpness by further closure of the aperture.



We would like to increase the reflective index. This can be done by reducing the focal length. It will make the blur spot become smaller. The optimal solution will be when the focal length (f) reaches the distance to the retina (P).

A short explanation about genetic algorithms

A genetic algorithm is an iterative procedure that consists of a constant-size population of individuals.
Every evolutionary step, which is called generation, the individuals are evaluated according to some pre-defined quality criterion called fitness.
Forming a new generation consists of three major stages:

- Selection – individuals are selected with a probability proportional to their relative fitness. This means that better fit individuals stand a better chance of reproducing, and the unfit will most likely disappear.
- Reproduction – two selected individuals change parameters by some probability.
- Mutation – there is a small chance for each individual to mutate. That means change a parameter by chance.

After these three stages a new generation is formed.

How I approached the problem

The human eye is a complex structure. As the real evolution of the eye involves the modification of many different quantitative characters it is almost impossible to coordinate them. And so my objection was to simplify the problem into the physical formation of the eye structure, and the construction of its lens. That is, finding how many generations it will take to evolve the eye's optical geometry.

What I needed was a unique function, the "fitness" function, which would help me in the selection stage. This function should be the quality of vision, because that is the reason for an eye to evolve in a certain way. An eye's fitness is evaluated by the following function:

$$\text{fitness}=\frac{1}{|5.263-(p/a)|}+\frac{1}{|p\text{-lens}|}$$

it can be seen that the fitness value is higher as the "p/a" ration is closer to 5.263, which is 1/1.9 as we've seen before, and as the focal distance reaches the distance to the retina, p.

Our initial stage begins with an eye, which is a flat patch of light-sensitive cells sandwiched between a transparent protective layer and a layer of dark pigment. This dark layer is in order to get light from one direction only (and not from behind).

The eye has 3 parameters. The first is constant: the length of the light-sensitive patch which is initialized to 100. The two other parameters are the ones that contribute to the fitness function. These 2 are the radius (which is the inverse of the curvature of the patch) and the lens focal length. The aperture and pit depth are calculated according to the radius. The radius is initialized to be 1000 – that means almost a straight line, and the focal length of the lens is 1000, which is far more than the final result, P, which we want to achieve. In fact, it is almost like there is no lens.

Each generation had 100 eyes. In each formation of a new generation 2 eyes are randomly chosen. This choice is with a probability proportional to their relative fitness. The reason we do this is we want stronger eyes to have a better chance of reproducing, but at the same time, leave a chance for the weaker eyes too. This is done by sorting the eyes according to their fitness, which means a fitter eye is stored in a higher cell in the eyes array than a weaker one. After that I pick an eye from the array getting the index with the following function: 99-(random^2)*100, where "random" returns a double value between 0 and 1(not included). This function makes the probability of a high number larger than that of a smaller one.

After 2 "parents" are randomly chosen they reproduce a child. This child has a 30% chance of being like his mother, 30% of being like his father, and 40% of being an overage of both.

After this, every child has a 0.1% chance to mutate. This means increasing or decreasing one of his parameters by 1%. This mutation is the actual reason for the

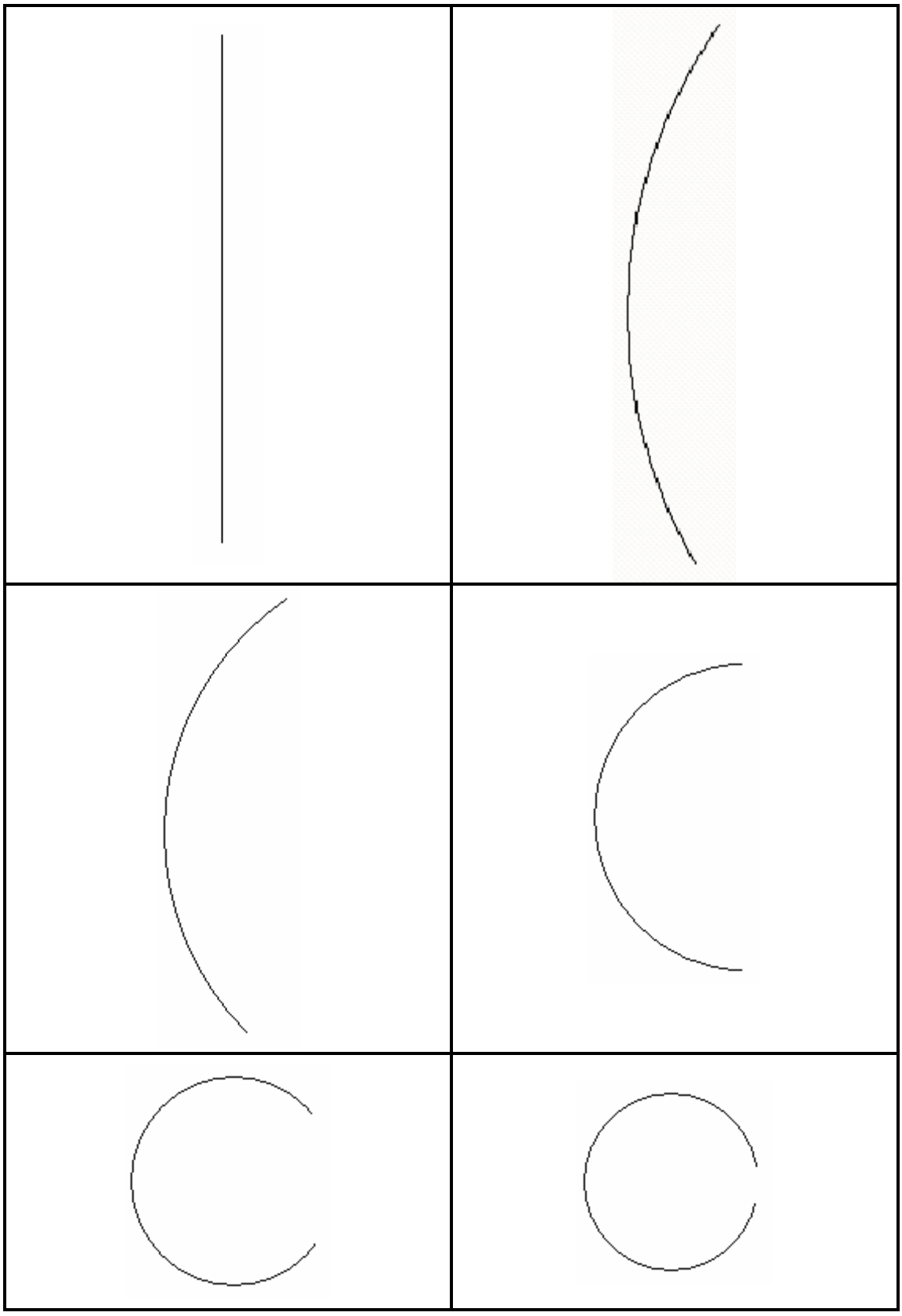
genetic change. If the mutation is good, that is - results in better fitness, than it stays. Otherwise, it is eliminated by the selection process.

The 100 "children" form the new generation which starts this procedure all over again.

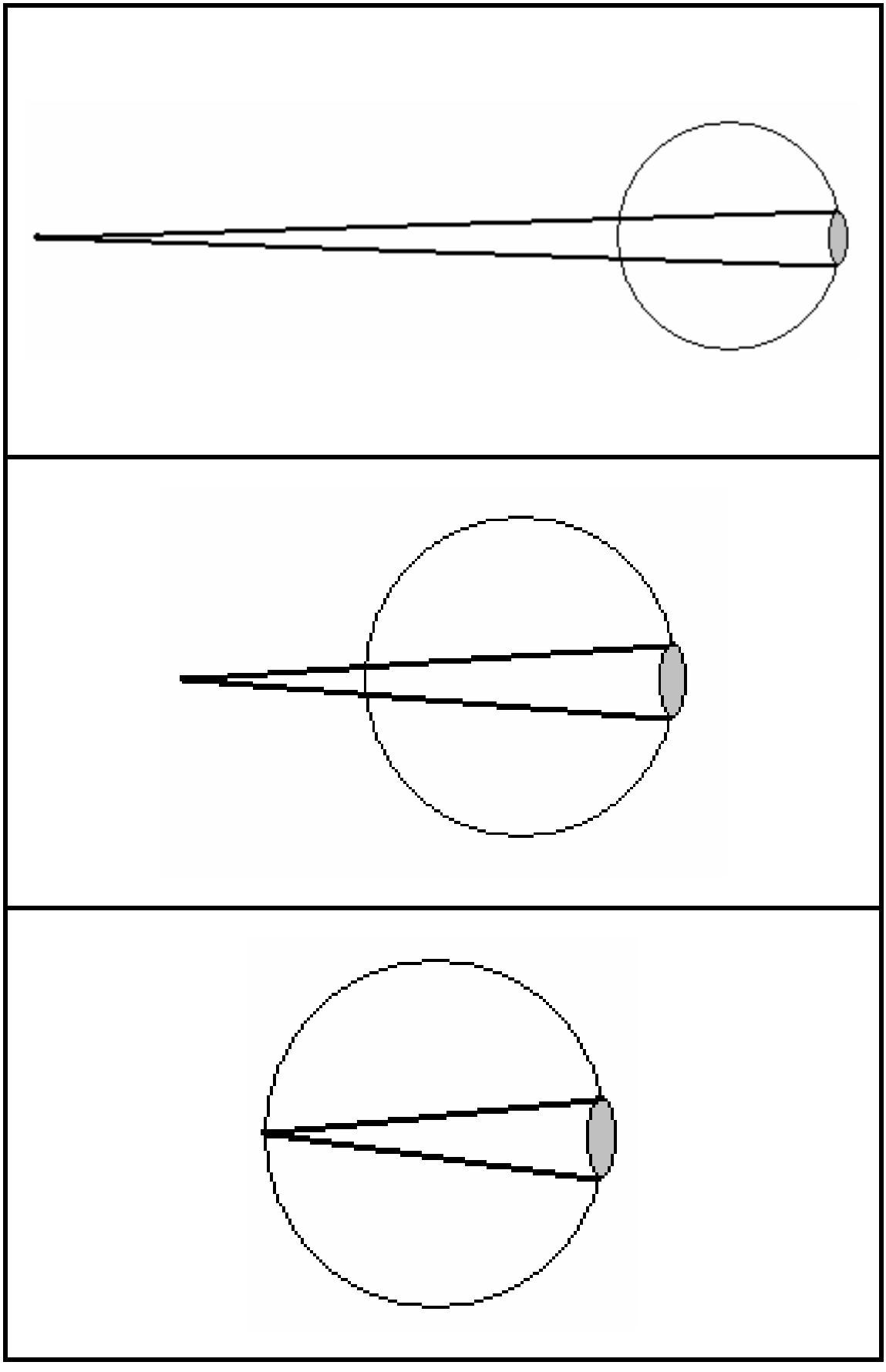
The algorithm is stopped once a fit eye has evolved. A fit eye is an eye with a fitness value which is above 100.

Results

The different stages of the development of the eye's structure:



And the different stages of the lens focal length:



The following results depend on a lot of different choices made:

- What is the probability for mutation
- By how much to mutate each time
- What are the initial factors given
- What is the percentage of children that look exactly like their parents
- The different possible options for a selection function
- And – of course, the fitness function.

All these can have a great impact on the results. What is really interesting to learn is the way the evolution works on the eye, and not the actual number of generations. Anyway, here are the results and their average for different chances of mutation:

mutation	1	2	3	4	5	Average	Max error
0.001	18385	16369	17403	16693	17520	17274	6%
0.004	6123	6315	6008	5892	5863	6040	5%
0.007	4198	4219	4337	4312	4116	4236	3%

And some results for a different amount of change in each mutation:

Amount of change	1	2	3	4	5	Average	Max error
1%	18385	16369	17403	16693	17520	17274	6%
0.1%	172694	173488	171983	173700	173450	173063	~0%

Conclusions

The parameters I used in my genetic algorithm for calculating the number of generation were very simple. In this kind of calculation it is impossible to give attention to every part of the eye because of its complexity. This is a very rough estimation, of course, but it lets us see that it is possible, which is the main reason, in my opinion, to this project.
As can be seen from the results the number of generations it takes the eye to evolve varies a lot by the parameters introduced. There are many different options that can be taken, and it is difficult to know which one is better: in each generation, should the change be 1% or 0.1%???
What is interesting to see is that in every group where the parameters stayed the same, the results were very close to one another. This indicates that in the long run, even though this algorithm is random, the evolution goes in one clear direction. Unfortunately, what is calculated is just how the eye looks now. It would have been nice to know how the eye would continue to change, but we'll leave that to evolution…

The code

```
import java.lang.Comparable;
import java.lang.Math;

class Eye implements Comparable{
	private double radius=1000;
	private double lens=1000;
	private double fitness=0;

	public Eye(){}

//-------------------------------------------------------------------------------
// The compareTo function is the function needed in order to implement Comparable
// it compares the fitness of the eye to a given different eye.
//-------------------------------------------------------------------------------
	public int compareTo(Object object){
		Eye eye=(Eye)object;
		if (fitness>eye.getFitness())
			return 1;
		else if(fitness<eye.getFitness())
			return -1;
		else return 0;
	}
//-------------------------------------------------------------------------------
// this function mutates the eye.
// it does so by a chance of 0.001%
// if the radius is not right yet, then the radius is changed.
// otherwise, the lens is changed.
// at the end we update the eye's fitness
//-------------------------------------------------------------------------------
	public void mutate(){
		//first of all mutate by a chance of 0.001%:
		if(Math.random()<0.001){//if you have to mutate

			double random_number=Math.random();
			if(random_number<0.5){
				if (radius>17.1) // if the shape is not good enough
					radius=radius*0.99;
				else
					lens=lens*0.99;
			}
			else{
				if (radius>17.1)
					radius=radius*1.01;
				else
					lens=lens*1.01;
			}
		}//if you were supposed to mutate

		//now update your fitness
		updateFitness();
	}
//-------------------------------------------------------------------------------
// this function calculates the fitness of the eye according to the radius and lens
// it is invoked at the last stage of the creation of a new eye - after it is mutated
//-------------------------------------------------------------------------------
	public void updateFitness(){
		double alfa=Math.PI-(50.0/radius);
```

```
                double p=radius+radius*Math.cos(alfa);
                double a=2*radius*Math.sin(alfa);
                double how_close_radius=Math.abs(5.263-(p/a));
                double how_close_lens=Math.abs(p-lens);

                fitness=(1/how_close_radius)+(1/how_close_lens);
        }
//-------------------------------------------------------------------------------
// get methods
//-------------------------------------------------------------------------------
        public double getRadius(){
                        return radius;
        }
        public double getLens(){
                return lens;
        }
        public double getFitness(){
                return fitness;
        }

//-------------------------------------------------------------------------------
// set methods
//-------------------------------------------------------------------------------
        public void setRadius(double r){
                radius=r;
        }
        public void setLens(double l){
                lens=l;
        }
        public void setFitness(double f){
                fitness=f;
        }
        public void print(){
                System.out.println("radius: "+radius+" focal distance: "+lens+" fitness: "+fitness);
        }
        public boolean isFit(){//a method in order to stop the evolution
                if (fitness>100)
                        return true;
                else
                        return false;
        }
}

import java.lang.Math;
import java.util.Arrays;

class Generation{
        private Eye[] generation;
        private int generation_number=0;
        //-------------------------------------------------------------------------------
        // the constructor
        // constructs a new generation of 100 eyes
        //-------------------------------------------------------------------------------
        public Generation(){
                generation=new Eye[100];//initiallizing 100 new eyes
                for(int i=0;i<100;i++)
                        generation[i]=new Eye();
        }
        //-------------------------------------------------------------------------------
        // this is the main method that runs the algorithm.
```

```
// we sort the old generation by fitness.
// we then produce 100 new offsprings:
// in each iteration we select two parents, apply a mating function on them,
// and then by a small probability mutate their child
// this child is the choise of the new generation
//-------------------------------------------------------------------------------
public void run(){
	boolean stop=false;
	Eye final_eye=new Eye();
	while(stop==false){
		Eye[] new_generation=new Eye[100];
		Arrays.sort(generation);//after this the generation array is ordered
		Eye fittest=generation[99];
		stop=fittest.isFit();
		final_eye=fittest;
		fittest.print();

		for(int i=0;i<100;i++){
			Eye mother=selectEye();
			Eye father=selectEye();
			Eye son=reproduce(mother,father);
			son.mutate();
			new_generation[i]=son;
		}//while new generation hasn't been built yet
		generation=new_generation;
		generation_number++;
	}//while work is not done

	//the next lines are calculations for the final output
	double radius=final_eye.getRadius();
	double lens=final_eye.getLens();
	double alfa=Math.PI-(100.0/(2*radius));
	double p=radius+radius*Math.cos(alfa);
	double a=2*radius*Math.sin(alfa);
	System.out.println("the eye is ready after "+generation_number+" generations!!!");
	System.out.println("it's apperture is "+a+", it's depth is "+p+", and it's lens is "+lens);
}//run
//-------------------------------------------------------------------------------
// this method selects a parent for reproducing
// it gives more chances to fitter parents than to weaker ones, but still
// everyone has a chance
//-------------------------------------------------------------------------------
public Eye selectEye(){
	int selected_parent=99-(int)Math.floor(Math.pow(Math.random(),2)*100);//old
	//int selected_parent=(int)Math.floor(Math.pow(Math.random(),2)*100);
	return generation[selected_parent];
}
//-------------------------------------------------------------------------------
// this methods gets two parents and reproduced one offspring. the son has a 30%
// chance of ending up like his mother, 30% like his father, and 40% like the
// average of both his parents.
//-------------------------------------------------------------------------------
public Eye reproduce(Eye m,Eye f){
	Eye son=new Eye();
	double mother_radius=m.getRadius();
	double father_radius=f.getRadius();
	double mother_lens=m.getLens();
	double father_lens=f.getLens();
	double son_radius=(mother_radius+father_radius)/2;
	double son_lens=(mother_lens+father_lens)/2;
```

```
                if (Math.random()<0.4 ){
                        son.setRadius(son_radius);
                        son.setLens(son_lens);
                }
                else if (Math.random()<0.7){
                        son.setRadius(mother_radius);
                        son.setLens(mother_lens);
                }
                else{
                        son.setRadius(father_radius);
                        son.setLens(father_lens);
                }
                return son;
        }//reproduce
}//generation
```

Bibliography

- "a pessimistic estimate of the time required for an eye to evolve" – Nilson and Pelger
- "A Brief Introduction To Genetic Algorithms" – Sipper
- Mathworld.com
- Evolutionary Algorithms – Tomassini
- Lecture notes in "Introduction to Computational and Biological vision"
- "climbing mount improbable" – Dawkins