Evolution of Human Faces

January 19, 2004

Project Members:

Rudolf Dittrich: rudid@gmx.at
Martin Höggerl: martini1606@yahoo.com
Rene Leikermoser: rene.leikermoser@gmx.at
Thomas Widhalm: twidhalm@cosy.sbg.ac.at

Academic Supervisor:

Helmut MAYER
helmut@cosy.sbg.ac.at

Correspondence to:

Universität Salzburg
Institut für Computerwissenschaften
Jakob-Haringer-Straße 2
A-5020 Salzburg
Austria
Contents

1 Introduction 1

2 Description Of The Problem 3
  2.1 General description 3
  2.2 The technical part 4
  2.3 The graphical part 6

3 Program Visualization And Dependencies 10
  3.1 Visualization 10
  3.2 JEvolution 12

4 Experiments 13
  4.1 Introduction 13
  4.2 Experiment 1 14
  4.3 Experiment 2 18
  4.4 Experiment 3 21

5 Progress Of Work 27
  5.1 Monday, 14.4 27
  5.2 Tuesday, 15.4 27
  5.3 Wednesday, 23.4 27
  5.4 Wednesday, 30.4 28
  5.5 Wednesday, 7.5 28
  5.6 Wednesday, 14.5 28
  5.7 Wednesday, 4.6 28
  5.8 Wednesday, 11.6 29
  5.9 Wednesday, 18.6 29

6 Milestones 30

7 Links 33
Abstract

The use of Evolutionary algorithms is complicated and certainly not suitable for every appliance. Evolutionary algorithms are useful, when the search-space for a problem is highly dimensional. As a result, it cannot be searched with conventional searching strategies. Our project’s main task is to demonstrate the effect of Evolutionary algorithms. Instead of just showing abstract numbers or tables, we use something, we are all used to. Faces. The first chapters give a general introduction to the use of Evolutionary algorithms and a short technical overview about the project. After that, three interesting experiments are described and finally an insight into the progress of work during the development is provided.
Chapter 1

Introduction

Evolutionary algorithms (EAs) are search methods that take their ideas from natural selection and survival of the fittest in the real world. EAs differ from more traditional optimization techniques in that they involve a search from a "population" of solutions, not from a single point. Each iteration of an EA involves a competitive selection that weeds out poor solutions. What does face-evolution mean? Face - evolution describes the process of evolving a certain group of faces as a population of individuals. Each face has its own genome that is expressed once per generation to make a subjective adjudgement possible. Of course a non-graphical Evolution would be possible - but a user-independent fitness-function would have to be implemented.

The members of the population are all born and evaluated by the user, the only difference is, whether they die with successors or without. Only the fittest individuals pass their genes on.

Initialize the population
Evaluate initial population
Repeat
  ..Perform competitive selection
  ..Apply genetic operators to generate new solutions
  ..Evaluate solutions in the population
Until some convergence criteria is satisfied
Schematic workflow of EAs.

After a certain number of generations, the individuals should look like the user wants them to be. We speak of convergence when that phenomenon gets noticeable.

The aim of our project is to create a program that is able to, with the help of a person sitting in front of the computer, evaluate faces and, using evolution (mutation and selection), create new faces, hopefully looking better than the older generation.

Therefore we used the framework JEvolution written by Dr. Helmut Mayer, with that it was easy to create, evaluate, mutate and select a certain bitstring, that has defined our face. This bitstring was divided into different parts, where each part described something in the face, e.g. the nose, the mouth or the eyes etc.

The graphical user interface was written with the java swing-framework.
Once finished, the program may be used for psychological studies, like finding a trustworthy face. But finding everything else as long as it is a certain expression of a face, is also possible!

We divided the team into the programmers, namely Rudolf Dittrich and Rene Leikermoser, and the designers, Martin Höggerl and Thomas Widhalm. The programmers took care of the evolution of the genotype, whereas the designers took care of the expression of it, the phenotype of a face. As shown below, the GUI consists of a collection of faces, their appropriate evaluation-bar that is used for rating a face and an options area. Above all faces, within that options area, two sliders represent mutationrate and crossoverate.

![The GUI](image)

Every change has to be confirmed by pressing the 'reset' Button to the right of the two sliders. The next generation is created and drawn by pressing the 'next Generation' Button.
Chapter 2

Description Of The Problem

2.1 General description

The use of Evolutionary algorithms is complicated and certainly not suitable for every appliance. Our project’s main task is to demonstrate the effect of Evolutionary algorithms. Instead of just showing abstract numbers or tables, we use something, we are all used to. Faces.

One field of application of Evolutionary Algorithms is a high-dimensional search space\[NaCmpLct, page 3\]. In our case, the search space consists of approximately $2^{32}$ possible results, we refer to as individuals.

That is, because the genome of an individual is composed of 232 Bits. Even though not every Bit Pattern is valid, the number of different patterns still is very high. Thus a linear search for a certain face is neither efficient nor desirable. The great advantage of the employment of Evolutionary Algorithms in finding a favoured face lies in the fact, that the probability of getting at least one face similar to the desired one increases with every generation. The key point for that increase is the fitness function. In Evolutionary Algorithms, the fitness function determines the chance for an individual or solution to pass its genes and therefore characteristics on. This is called forced selection or breeding. In our case, the fitness function is replaced by an external rating, done by a user. Since similar parts of the desired face are included in the fittest faces, a convergence can be experienced. New generations of size n are either created by mutating the n fittest individuals or by mating (combining) the n fittest individuals two times.

The idea of using faces for the visualization of Evolutionary Algorithms is not that new. A slightly different approach to that problem has been made by another project some years ago. Instead of using comic faces as we did, scanned parts of real faces were, relatively unprecise, arranged on a shape of a face. Using scanned parts implies great problems with scaling and placing the parts correctly and, in succession, naturally. The mathematical representation of a (comic-) face is much more easy and efficient. A face is composed of circles, ellipses, squares rectangles and lines. These geometric objects are easy to compute, store and display. All can be described by four parameters at maximum. Although that technique may look quite simple, the faces can absolutely express various moods.
CHAPTER 2. DESCRIPTION OF THE PROBLEM

2.2 The technical part

The FaceGallery class is responsible for displaying the information and getting the marks from the surface. It creates the whole window and sets the Menu and the sliders. After that it initializes the Breed_Chambers. Since every individual has its own Breed_Chamber, an array of them is filled. When the 'next Generation' Button is pressed, every Breed_Chamber returns one JPanel, which has the drawn face on it. These JPanels are arranged in a grid-layouted big JPanel. A FacePhenotype is created, that saves the current Gallery, and is given to the Evolution-class. A FacePhenotype calculates its own fitness and can be seen as the expression of the genome.

```java
public void calcFitness(){
  note = gallery.getNote();
  this.fitness = (double)(1.0/note);
}
```

As seen here, the fitness function is just the inverse of the achieved mark. $\frac{1}{1}$ is greater than $\frac{1}{2}$ is greater than $\frac{1}{3}$ etc..

The Evolution-class creates a BitChromosome and sets its length. After that JEvolution is performing the evolutionary calculation.

```java
jE.addChromosome(chromX);
jE.setMaximalGenerations(maxGenerations);
jE.setPopulationSize(populationSize);
facePheno = fPh;
jE.setPhenotype(facePheno);
jE = JEvlution, chromX = chromosome
.
.
jE.doEvolve(1);
starts one step of Evolution.
```
The main work, as to perform the mating of the genes, the crossover of genes and spontaneous mutation of genes is done by the JEvolution Framework provided by Mr. Mayer. Therefore we only had to initialize the chromosomes, give them to JEvolution, let JEvolution do 1 evolutionary step and then read back the results from the calculation.

Nevertheless we had to provide a specific fitness function to tell JEvolution which genes are "better" than others. We decided to focus on a school-based rating scheme, where 1 stands for "best" and 5 for "worst".

\[ \text{FacesPanel} = \text{new JPanel();} \]
\[ \text{FacesPanel.setLayout(new GridLayout(FacePanelRows,FacePanelColumns));} \]
Layout for the main window.

Despite many troubles, which were partly the same as the problems our predecessors had, like the broken on-screen layout of the faces in the main window, we also had some troubles on how to convert the bitstrings into parameters for the face-generating class called the "BreedChamber".

\[ \text{int}[\ell] \text{ geneWidth} = \{5,8,6,6,6,...,1,1,4\}; \]
This array specifies the width in bits a gene has.
All in all, there are 43 different numbers that identify a face.

We finally decided to first determine how many bits the BreedChamber could handle, then generating the genes on this information and then pass the genestrings directly into the BreedChamber where the bitstrings were broken up and converted to the parameters for the different parts of the faces. The conversion is simply done by 'breaking up' the Bitarray into parts. The length of a part is determined by the array shown before. For example, the first 5 Bits describe the first parameter, in our case the additional width of a face. Since \(2^5\) is 32, the faces can differ within 64 pixels in width.

By doing this we provided flexibility for the both teams to either write the code for the GUI and the program or write the code for face-generation and then being able to join both parts without any difficulties.

Another problem that arose was the question how to save the bitstrings for the faces and add the corresponding fitness for each chromosome. We decided to store the chromosomes in a n*m-matrix where n is the number of faces to display and m is the chromosomelength. Another int-array with length n holds the marks for each face, so we were able to assign each face its own mark and provide this to the JEvolution package.
2.3 The graphical part

The Breed_Chamber class is responsible for turning a genome, namely an array of information into a face, a special form of JPanel. The most important method, apart from the draw-method is the breed-method. It is called with an array of ones and zeros. The breed-method calculates an array of 43 Integers out of the 232 Bit long array. It does that by using the elements of a geneWidth array, that contains the lengths of the parts of the BitArray, the features or parameters have. That is important, because the draw-method needs Integers, not Bits. In the draw-method, the elements of the array are checked and, if necessary, adapted. The correct values are then assigned to global variables that are used in the drawing process. The drawing process itself is done by creating a special JPanel by using the global variables.

The elements are drawn in the following order: hair, head, wrinkles, eyes, nose, extras like asian-type eyes, pupils, eyebrows, mouth and, at last, glasses. Some examples:

```java
    g.setColor(FACE_CL);
    g.fillOval(left = 40, down, FACE_BR, FACE_HO);
```

These two lines draw the head; FACE_CL (Color of the face), FACE_BR (width of the face) and FACE_HO (height of the face) are some of the global variables, we mentioned earlier.

```java
    g.setColor(Color.white);
    g.fillOval(XLEYE_POS, X_AXIS, EYE_BR, EYE_HO);
```

This draws the left eye. XLEYE_POS defines the deviation of the position of the eye horizontally towards a standard position. X_AXIS means, that the eye is located vertically at the x-axis. EYE_BR and EYE_HO are the width and the height of the eye.
1. A face can have several types of wrinkles. Wrinkles on the forehead, on the cheeks, surrounding the mouth, at the chin etc. Every type is either active (1) or inactive(0).

2. Nearly every face has hair on its top. In the example, we see the Afro-Style type. Hair, as well as face and nose, has its own color, in RGB-colors\(^1\).

3. The nose can vary between three base types, namely European, African and Asian. The thickness and the color (see point 2) is inherited separately.

4. The eyes differ in height and width. Apart from that, faces can have three different colors of the iris. The position of the pupil is at random.

5. Ears have the same color as the face itself, their size is variable.

6. The mouth is the most complex thing in the face. It is a part of a Sinus-Curve, that is determined by two parameters between 0 and \((2\pi) \times 10\).

7. Some faces have glasses, some not. There are three different types of glasses.

8. The face itself varies in height and width, the color is assigned the same way as to hair and nose.

\(^1\)Three numbers from 0 to 255, representing red, green and blue
CHAPTER 2. DESCRIPTION OF THE PROBLEM

Elements of a face (2)

1. This face has not much hair on its head. This type of hairstyle includes three rounded rectangles. One on top of the forehead, and two next to the ears (see point 4).

2. The ears are ovals, their size changes from individual to individual.

3. As mentioned before, the nose can vary between three base types. In fact, every face is assigned to one type. This, Asiatic type face is mainly characterized by a triangle, covering parts of the eyes and nose.

4. Faces may also have sideboards. These are just rounded rectangles, that are the same color as the hair.

Elements of a face (3)

1. As said before, sideboards are rounded rectangles and are sometimes integrated in other hairstyles.

2. The beard is obviously composed of several triangles. In Java, triangles are drawn as 3-edged polygons.
3. The nose is a collection of, as seen here, an oval, two triangles and one rectangle.

4. Wrinkles consist simply of lines arranged in bows or straight lines.
Chapter 3

Program Visualization And Dependencies

3.1 Visualization

The whole Program is started with Gallery.java, which starts FaceGallery.java. Here all components are initialized and the GUI is set up. It holds an array consisting of different BreedChamber-objects. Each BreedChamber-Object holds a bitstring which represents the genetic code for one face. Faces are currently represented through 232-Bit genes. For details in constructing faces see Chapter 2.3. These bitstrings are also given to JEvolution, the backend of the program which does all the calculations on the different bitstrings. The JEvolution Program is not only initialized with the different bitstrings, but also with a specific mutation rate, crossover rate and tournament type, which stays the same over a whole population. After starting the program the user has to give marks for the different displayed faces. The rage is from 1, which represents the best, to 5, which is worst. Also a face can be give an 'E' (for 'elite'). After this is done, JEvolution calculates the next population of faces, based on the marks given by the user. Better ranked faces pass their genes more likely to other faces than the worse rated faces do. This process can be repeated as long as the user wants, to create different experiments.

UML-Diagram of the FaceEvolution program
Detailed view of the UML-Diagram, showing variables and methods.
3.2 JEvolution

JEvolution is a Java package for Evolutionary Algorithms. Much attention is payed to object-oriented design. The gain of clarity and manageability comes at the cost of speed. JEvolution is relatively easy to use.

It is important to know, that JEvolution demands a user defined Fitness function. According to it’s documentation\(^1\), JEvolution also supports other projects like netJEN (evolution of Artificial Neural Networks), fuzJEN (evolution of Fuzzy Controllers) or evAlloc (generic solver for allocation and scheduling problems). JEvolution is available in version 0.7.

Even though the whole Documentation is included in the downloadable JEvolution-version (see footnote), here the key-features of JEvolution.

1. Lean and compact Java package
2. Supports bitstring-, integer, permutation and real-encoding (native chromosomes)
3. Supports evolution of chromosome length (bitstring encoding) by means of non-homologous crossover (experimental)
4. User may provide custom chromosomes
5. Genotype may comprise an arbitrary number of different chromosomes
6. Chromosome shuffling
7. Records a "Star Pool" (containing all intermediate best individuals)
8. Flexible termination criteria (minimal/maximal number of generations, fitness threshold)
9. Handles maximization and minimization problems without fitness scaling
10. Supports tournament selection with arbitrary tournament size (native selection method)
11. User may provide custom selection method (implementing the Selection Interface)
12. No recomputation of already evaluated individuals (fitness repository)
13. Simple Java Interface for problem-specific code
14. Direct access to genotype for Lamarckian evolution or repair methods
15. Set up for distributed computation of fitness
16. Set up for thread interrupt
17. Reports simple statistics on evolutionary progress

\[^1\text{Included in: http://www.cosy.sbg.ac.at/~helmut/Stuff/jevolution.tar.gz}\]
Chapter 4

Experiments

4.1 Introduction

The following terms are used in the experiments setup:

In all experiments we used the so called "Tournament Selection" to determine the best faces in a population, which are passing their genes to other individuals in the population.

Tournament Selection is described as:
"Tournament Selection draws a number of Individuals randomly from the Population and conducts a tournament between the Individuals in the group where only the winner survives. Tournaments are conducted until the offspring Population is filled to its size. Before each new round of tournaments Individuals are shuffled within the Population so as to vary the participants in a specific tournament. In each round of tournaments each Individual takes part in exactly one tournament." From: JEvolution Documentation

The Mutation rate is defined as follows:
The mutation rate determines how many bits in a faces' genestring "flip" spontaneous, that means how many bits are switching from 0 to 1 or vice versa. This should simulate the natural mutation of human genes caused by radiation for instance. In the JEvolution framework which we used, the mutation rate can be adjusted between 0 and 1. 0 stands for no mutation whereas 1 is full mutation, that means the bitstring is flipping as a whole. A value of 0.005 means 0.5 percent of the bits in the genestring are switching their state. In a genestring consisting of 200 bits this would be exactly 1 bit.
This bitflipping can cause more or less effects on the characteristics of a face, depending on the position of the bit in the genestring. In further processing of the bitstring, the string is split up and converted into decimal values, so bitflipping has not such a big impact on the value of the result when the bit flipped was the least significant bit (LSB). But it has a very large effect on the final result if the bit flipped was the most significant bit (MSB).

Crossover rate:
The crossover rate determines how much percent of the bits from the individuals A and B are transferred into the genestring of the new individual C of the population.
4.2 Experiment 1

Task: Observation of unsupervised evolution.
In this experiment we wanted to see what happens to a population of 3 different faces over time when all faces was given the same marks "3".

Setup:
No. of Faces: 3
Mutation rate: 0.003
Crossover rate: 50
Marks: static (3 for all faces all the time)

Generation 1.

Generation 2.

In the first generation we have 3 different faces, while in the second generation, the light face with the glasses is being replaced by the green face with the red hair. Also we see that the left face has changed its hairstyle and the type of the mouth. This might have caused by crossover and mutation.

Generation 3.

As we take a look at the third generation we see that the rightmost face has not changed at all, while the leftmost face got the green facecolor, the round facetype, the wrinkles
and the pink nose from the two green faces in the second Generation. It kept its glasses
and the mouthtype. Also the middle face changed its color which it got from the leftmost
face in generation 2. It has a new hairstyle, probably due to mutation and its mouthtype
changed from a bad looking mouthtype into direction of a more neutral mouthtype like
the left face in generation 2 had.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig.png}
\caption{Generation 4.}
\end{figure}

Generation 4:
In generation 4 the middle face from generation 3 was entirely replaced by the rightmost
face of generation 3.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig.png}
\caption{Generation 5.}
\end{figure}

Generation 5:
In generation 5 something interesting happens: the leftmost face changes its hairstyle from
no hair to little hair, it also gets a beard and it loses its glasses. Also its mouthtype changes
to negative looking. The rightmost face gets a lighter haircolor due to mutation.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig.png}
\caption{Generation 6.}
\end{figure}

Generation 6:
Here the left face takes over the hairstyle from the middle face in generation 5, whereas
the right faces hairstyle gets an intermediate between its original hairstyle and the hairstyle
of the left face of generation 5. Also the haircolor is an intermediate between its light
haircolor from generation 5 and the haircolor of the left face from generation 5.
CHAPTER 4. EXPERIMENTS

Generation 7:
The left and the right face are exchanging their hairstyle, the left face gets a more neutral looking mouth and the middle face gets smaller, most likely due to mutation.

Generation 8:
The left face gets more hair, the mouthstyle changes back to the same as in generation 6. The right face gets as small as the middle face.

Generation 9:
The mouthtype of the middle face changes to a more neutral type. The left face gets smaller.

Generation 10:
The left face gets a slightly darker skin color than the middle and the right face.
Generation 11:
The mouthcolor of the left faces changes from black to red, the right face loses its hair.

Generation 12:
The left face takes over the hairstyle from the two other faces, it also gets a black mouth again. The middle face gets smaller and gets the red mouth from the left face. Also the right face gets the red mouth from the left face.

Generation 13:
The left face gets its hairstyle from the other two faces. Also its mouth gets red again and its nose gets smaller, whereas the middle face gets the hairstyle from the left face, has now a black mouth and its nose is getting wider. The right face’s haircolor is becoming lighter.

Generation 14.
Generation 14:
The left and the middle face exchange their hairstyles, mouthcolors and nosetypes again. The right face’s haircolor is getting darker again.

Generation 15:
The left face is giving its hairstyle, nosetype and mouthcolor to the other two faces, but gets the hairstyle, the nosetype and the red mouthcolor from the other two faces.

Generation 16:
The haircolor of the left face changes suddenly, most likely due to mutation, to a light brown color, the hairstyle is getting the same as the one of the other two faces. The middle face got the hairstyle from the left face, the mouthcolor changes to red.

Generation 17:
Now all faces have the same hairstyle, the same nosetype and the same mouthtype and -color.

4.3 Experiment 2
Task: Finding a specific type of face.
In this experiment we first wanted to get a specific type of face: male, little black hair,
green face, no beard.

Setup:
No. of Faces: 3
Mutation rate: 0.003
Crossover rate: 50
Marks: user defined

Generation 1:
The left face is exactly the face we are searching for, therefore it gets a '1'. The middle and the right face do not meet the requirements, so they both get a '5'.

Generation 2:
The left face comes close to our needs, so he gets a '3', the middle face is not the type of face we are searching for, so it gets a '5'. The right face is the best representation of our demands, so it gets a '1'.

Generation 3:
Generation 3:
Here we see that the face marked 1 has given its characteristics to the other green face. the blue face still does not cover our requirements to get anything better than a '5'. The other two faces both get a '1'.

Generation 4:
Although all faces are very close to our demands, the right face still has too much hair, so it gets a '3'. The other faces get a '1'.

Generation 5:
The right face has given it characteristics to the left face, so they both get a '3'. The middle face is the one we are looking for, it gets a '1'.

Generation 6:
Still the faces have the same characteristics as in generation 5. So all get the same marks as before.
Generation 7:
The both faces gave their characteristics to the middle face, they all look the same now. This comes really close to the type of face we were looking for.

We found convergence of the characteristics of the different faces in the 7th generation.

4.4 Experiment 3

Task: Observe an evolution with high mutationrate between identical faces.
In this experiment we start from 3 faces with the same characteristics.
We have chosen a high mutationrate and we give all the faces the same marks ('3'), so we are able to watch the mutations and how they are spread in the whole population.

Setup:
No. of Faces: 3
Mutationrate: 0.009
Crossoverrate: 50
Marks: static (3 for all faces all the time)
CHAPTER 4. EXPERIMENTS

Generation 2:
The left face changed its mouthtype. The characteristics of the other 2 faces remain the same.

Generation 3:
The mouthtype was exchanged between the left face and the face in the middle. The right face now has a beard. This is the first generation, in which every face has its own unique characteristics.

Generation 4:
The mouthtypes get back to the type of mouth we had at the beginning. The left face gets violet hair. It is also noticeable that the faces begin to differ in their skin color. The left face has the bluest skin, the face in the middle is the lightest one.
Generation 5:
Most characteristics of the faces nearly remain the same, only the right face now gets female. Notice the blue haircolor which seems to spread over the population.

Generation 6:
The female face disappears, also the blue haircolor seems to get replaced by the olive haircolor again. Notice the hairstyle of the right face: it has a slightly different style. Also the characteristics of the mouths are slightly changing.

Generation 7:
Now we have 3 faces, each with a different haircolor, the face in the middle also has a pink nose, also the style of the mouth is different in every face.
Generation 8:
The brown haircolor spreads over the population, also each face has now a different colored nose.

Generation 9:
The orange haircolor mixes with the brown haircolor.

Generation 10:
The orange haircolor mixes with the brown haircolor even stronger, the right face is female again.
CHAPTER 4. EXPERIMENTS

Generation 11.
The female face stays in the population, the haircolors get darker again.

Generation 12.
The female face disappears the second time, all faces now converge: they all have the same
haircolor, the same type of nose, the same color of the nose and the same shape of the
mouth. This is most likely caused by the fact, that the genes which were altered through
mutation did not have such a big effect on the characteristics of the face, than the genes
which were altered by crossover.

Generation 13.
The characteristics of the faces mainly stay the same. Again this can be explained through
the stronger effect of the genes exchanged by crossover over the weaker effect of the genes
that change through mutation.
CHAPTER 4. EXPERIMENTS

Generation 14.

Generation 14:
The left face changes its color of the face due to mutation.

Generation 15.

Generation 15:
The violet face from generation 14 is gone most likely due to the crossover of the genes from the other two faces.

Generation 16.

Generation 16:
Again mutation causes the faces to alter their characteristics. The left face gets darker, the face in the middle gets darker too, plus it is getting a little bit smaller.
Chapter 5

Progress Of Work

5.1 Monday, 14.4
First Meeting. Decision not to use the old project. First ideas about using comic faces instead. Faces could be separated into three or four horizontal parts, those parts could be combined. Idea of representing genome as one number or array of numbers. Ideas of morphing and warping coming up. Those could be used with fotos.

5.2 Tuesday, 15.4
First comic faces produced. Idea with that parts was not that good, too many problems with width. Too little influence on shape of face. Work with JEvolution harder than expected. Fitness function does not work at all. Connection not that easy. Interfaces to graphical part specified. Agreement on arrays of Bits as genomes. Idea of Elite-Faces comes up. Best faces should be saved into a special pool.

5.3 Wednesday, 23.4
New version of JEvolution received. Problem with Fitness in JEvolution seems to be solved. Starting to write Paper. Introduction and structure ready. Features of a face drawn as circles, ellipses and rectangles. Parameter list created and implemented. Three different colors of skin: White, Black, Asian. Discussing additional colors and features for the faces. First ideas of distinguishing between men and women. Probably mating only possible between different sexes. Talking about possible problems like one sex outbalancing the other. Possible solution: Polygamy.
Deliberating upon introducing age to individuals and adopt fittest individuals into next generation. Should older individuals have an advantage or disadvantage of their age? Could older individuals have a higher or lower mutation-propability?
Deterioration of genome quality (higher propability of deletion by turning 1 into 0) possible.
CHAPTER 5. PROGRESS OF WORK

5.4 Wednesday, 30.4

Technical part seems to be stable, some problems in returning painted panels from the graphical class, named Breed_Chamber, to the main part of the program. Graphical part fully functional, relatively pleasing results.

5.5 Wednesday, 7.5


5.6 Wednesday, 14.5

Crossing-Over implemented. Rate specifies the amount in percent of genes a parental individual passes on to its successor. Chinese looking eyes implemented. Idea of morphing and warping postponed - not very suitable for comic faces.

5.7 Wednesday, 4.6

Change essential parts in the Breed_Chamber class. Several parameters changed from random to not random. The most important change is, that the color of the skin, as well as the color of hair and nose should be determined by three parameters each, numbers from 0 to 255. The RGB - mode. Those changes should allow even more possibilities.
5.8 Wednesday, 11.6

Introduction of wrinkles to the faces, the changes concerning colors have already been made. Suggestion to put converting of the bitstring to integers into the Breed.Chamber class. Concept of deletion and concept of survivors abandoned. Every generation consists of completely new individuals. Concept of elite-faces dropped. The waiving of strict seperating between men and women decided. Even same sexes may mate.

The graphical part produces faces like those by now.

5.9 Wednesday, 18.6

Graphical part finished, Parts joined. Continuing work on technical part. Again problems with fitness function.
Chapter 6

Milestones

1. Due date: 9.4.2003: Being able to describe, what the program should do
2. Due date: 22.4.2003: Having a first version of the graphical part
3. Due date: 30.4.2003: Joining better versions of the two parts
4. Due date: 9.5.2003: Finishing the work on the graphical part
5. Due date: 21.5.2003: Finishing the work on the technical part
6. Due date: 22.5.2003: Managing to create a test-version of our project.
7. Due date: 11.6.2003: Changing the Breed-Chamber towards more colorful faces.
9. Due date: 7.7.2003: Finishing the project.

ad 1: Reached.

ad 2: Reached. The first face was drawn, looked quite good. We should be able to reach our aims in time.

ad 3: Not reached, we might need another two days.

ad 4: The work is still not done, we have to adapt the graphical part to the needs of the programmers. It’s possible that we will not have time to realize our idea of using warping and morphing.

ad 5: We postponed the date for that one, since we saw that we wouldn’t be able to reach that milestone by 13th of May.
ad 6: The testversion is running, but it does not work properly.

ad 7: Done. We are now able to create any possible color for face, hair and nose.

ad 8: Thanks to Mr. Mayer, done. It was a problem with the fitness-function.

ad 9: Done. We are ready and proud of our work.

Having finished our project, we must say, that not every milestone was reached in time, as you can see in the listing above. Even though we were obviously not everytime close to reality with our plans, we tried hard and succeeded at last. Our milestones were mainly beacons to guide us towards the right direction. Some things were simply underestimated, and took us longer than we thought.
Bibliography

[JEDoc] Dr. Helmut Mayer: JEvolution Documentation

[NaCmpLct] Dr. Helmut Mayer: Natural Computation Lecture Script part 2
Chapter 7

Links

English Links:

http://tlc.discovery.com/convergence/humanface/timeline/timeline.html
An interesting document about what was considered beautiful in history.

http://brtom.org/face/face.html
This page shows how the face of a man changes over the years. This page made us think about introducing age as a new parameter. We did not do it in the end.

German Links:

http://www.bihler-online.de/mixed/pdf/ps2_script.pdf
A good paper that treats the aspects of judging people from the outside. In German only.

http://visor.unibe.ch/WS02/design/arbeiten/Schoenheit.pdf
This document tries to find out about the beauty of human faces. In German only.