

UNIVERSITY OF OSLO
Department of Informatics

An Educational
Genome Browser

A Project in Interactive
Learning

Master's thesis

Jørgen Hvamb Sveli

May 9, 2008



Preface

This thesis is part of the Master's Degree in Informatics at the Department of Informatics at the University of Oslo. The work with this thesis started in the winter of 2006/2007 and was completed during the spring of 2008. I would like to thank my supervisors, Professors Ole Christian Lingjærde at the Biomedicine group and Arne Maus at the group for Object-Oriented, Modeling and Language at the Department of Informatics at the University of Oslo, for their patience, good advice, and encouragement during this period. I would also like to thank Kristine Braathen Hein for contributing her ideas for how genome browser could be used in education.

Finally, I would like to thank my family and my My Anh for, at times, much needed personal support.

If you chase two rabbits, you will lose them both. - Native American Saying

And the users exclaimed with a laugh and a taunt: "It's just what we asked for but not what we want." - Anonymous

Never trust a computer you can't throw out a window. - Steve Wozniak

Contents

1	Introduction	1
1.1	An Educational Genome Viewer	1
1.2	The Ideal Genome Viewer	2
1.3	Structure of the Document	3
2	Basic Molecular Biology	5
2.1	An Outline of DNA	5
2.1.1	Initial Comments	5
2.1.2	DNA and Heredity	5
2.1.3	The Units of Heredity	7
2.1.4	DNA Structure	8
2.1.5	The Genetic Code	9
2.1.6	Expression	10
2.1.7	Genetic Variation	11
2.2	Biological Databases	12
2.2.1	Sequence data	13
2.2.2	Sequence Annotation	14
2.2.3	Genomic Meta-Information	16
2.2.4	Details of the Human Genome	16
2.2.5	Summary	17
2.2.6	Genome Browsers	17
3	Introducing Genome Browsers	19
3.1	Introduction	19
3.2	Genome Browsers	19
3.2.1	Motivation	19
3.2.2	Users	20
3.2.3	Abilities and Uses	20
3.2.4	List of Genome Browsers	21
3.3	Presenting a Selection of Browsers	22
3.4	The Ensembl Genome Browser	22
3.4.1	Using the Browser	22
3.4.2	Interaction	24

3.4.3	Features Visible	24
3.4.4	Level of Detail	26
3.5	The NCBI Map Viewer	26
3.5.1	Using the Browser	26
3.5.2	Interaction	27
3.5.3	Features Visible	27
3.5.4	Level of Detail	27
3.6	The X:Map Genome Browser	27
3.6.1	Using X:Map	29
3.6.2	Interaction	29
3.6.3	Features Visible	30
3.6.4	Level of Detail	30
3.7	Summary and Comments	32
3.7.1	Functionality	32
3.7.2	Interaction	32
3.7.3	Level of Detail	33
3.7.4	Types of Features	33
4	Analysis of Requirements and Domain	35
4.1	General Requirements	35
4.2	Functionality	36
4.3	Non-functional Requirements	37
4.3.1	Software Framework	37
4.4	Domain Model	38
4.4.1	First Draft	38
4.4.2	Discussion	40
4.4.3	Domain Model, Rev. 1	40
4.5	Arriving at a Class Model	41
4.5.1	Introduction	41
4.5.2	Diagrams	42
5	Designing an Educational Browser	45
5.1	Basic functionality	45
5.1.1	Select a Genome	45
5.1.2	Browse Chromosomes	46
5.1.3	Explore Chromosome	46
5.1.4	Level of Detail	48
5.1.5	Gene Information	49
5.2	Design Choices	50
5.2.1	Overview of Chromosomes	50
5.2.2	Exploring Chromosomes	51
5.2.3	Genes	51
5.2.4	Exons	51
5.2.5	SNPs	52

5.2.6	Sketches	52
6	Construction	55
6.1	Deciding on a Programming Language	55
6.1.1	C++	55
6.1.2	Python	56
6.1.3	Java	57
6.1.4	Conclusion	58
6.2	Graphical Framework	59
6.2.1	Demands	59
6.2.2	Java 2D	59
6.2.3	Jogl - OpenGL Bindings for Java	60
6.2.4	Other Alternatives	61
6.2.5	Technical Comparison, Java 2D and JOGL	62
6.2.6	Comments	63
6.2.7	Choosing JOGL over Java 2D	63
6.3	Investigation of Data Size	64
6.3.1	Data to be Retrieved	64
6.3.2	Chromosome Data	64
6.3.3	Attribute Data	65
6.3.4	Feature Data	65
6.3.5	Summary	65
6.4	Evaluation of Data Sources	66
6.5	NCBI Entrez	66
6.6	Biomart	70
6.7	Selecting Data Provider	71
6.7.1	Conclusion	72
6.8	Integrating Sequence Data	72
6.8.1	Regarding Data	72
6.8.2	Parsing Approaches	72
6.8.3	Sequential Read, Entire File	73
6.8.4	Random Access File	74
6.8.5	Memory Mapped File IO	74
6.8.6	Technical Comparison	75
6.8.7	Conclusion	76
6.9	Implementing Presentation of SNPs	76
6.9.1	Source of SNP Information	76
6.9.2	Drawing	77
7	Introducing Sigve	79
7.1	Initial View	79
7.2	Two-stranded View	79
7.3	Two-stranded View, Sequence Visible	81
7.4	Phenoportal	81

7.5	Known Bugs	84
8	Conclusions & Further Work	87
8.1	Conclusions	87
8.2	Complaints	88
8.3	Missing Features and Changes	88
8.4	Switching Graphical Framework?	89
8.5	Program Extension, through an API	89
8.6	Focus on Usability	90
8.6.1	Perfecting the GUI	90
8.6.2	Usability Testing	91
8.7	Visual Appearance	91
8.7.1	Location Probes	91
8.7.2	Visual Indication of Zoom	91
A	Signe, Instruction Manual	95
A.1	Installation	95
A.2	Starting the Program	96
A.3	Finding the First Gene	97
A.4	Using the Phenoportal	97
A.5	Known Issues	98
B	Discrete Topics	99
B.1	Extensible Markup Language - XML	99
B.2	Virtual Memory	100
C	Acronyms and Expressions	101

List of Tables

2.1	Ensembl: Homo Sapiens Genome Statistics	17
3.1	List of genome browsers	21
3.2	Ensembl feature data sources	25
3.3	NCBI Map Viewer: Data Columns	29
6.1	Benchmark results	63
6.2	Data stored for a chromosome	64
6.3	Data stored for a cytoband	65
6.4	Data stored for an attribute	65
6.5	Data stored for a feature	65
6.6	Average time cost of a read and standard deviation (SD)	76

List of Figures

2.1	Chromosomes In a Microscope	6
2.2	From DNA to Protein	6
2.3	A gene and its constituents	7
2.4	Schematic view of a chromosome (a), Chemical structure of DNA (b).	8
2.5	The splice process	11
2.6	Example of the FASTA file format	14
2.7	Example of the GenBank file format	15
3.1	Ensembl Overview of chromosome 1 (a), Ensembl Contig view (b).	23
3.2	Ensembl: Detailed view	25
3.3	NCBI: Genome view	26
3.4	NCBI: Master view	28
3.5	X:Map Starting view	31
3.6	X:Map Details view	31
4.1	Domain Model, 1st draft	39
4.2	Domain Model, Revision 1.	41
4.3	Class Diagram: Attribute Hierarchy	43
4.4	Class Diagram: Comprising Classes	44
5.1	User Interface Sketch 1	53
5.2	User Interface Sketch 2	53
6.1	Algorithm used for testing RandomAccessFile and Memory Mapped File IO	75
7.1	Sigve: Initial View / Overview	80
7.2	Sigve: Two-stranded View	81
7.3	Sigve: Examples of different levels of zoom	82
7.4	Sigve: Sequence View	83
7.5	Sigve: Phenoportal	84
A.1	1st time launch: Confirm download-dialog	96

LIST OF FIGURES

ix

A.2	Download and Decompression Dialogs	96
B.1	Recipe for bread encoded in XML	99

Chapter 1

Introduction

1.1 An Educational Genome Viewer

Traditionally, genome browsers have been targeted at researchers in biology and medicine. The goal of this project is to explore a novel utilisation of such software. Can a genome browser be used for educational purposes in educational contexts where non-experts constitute the target-group? The following scenario elaborates further:

Knut and Kari are students of biology in upper secondary education. They have had some lectures on the genetic material, DNA ¹. They have heard that a copy of the entire genetic code resides in the nucleus of each cell. The teacher has explained that our DNA comprises 23 or 24 distinct chromosomes which in turn comprises genes. They know that genes determine the way they look, how tall a person might grow, among other things. On the most basic level they have learnt that strands of four different molecules, called bases, form up the DNA-molecule. Knut and Kari find this subject compelling and they now thirst for more information, and they begin to discuss how it all connects. Knut wonders if genes are separate molecules, like he has heard chromosomes are. Kari comments that she has seen drawings of chromosomes but not genes.

Being an eager pair of students, Knut and Kari decide to research the topic further. Kari finds an article on DNA from an online encyclopedia. The article has an image of part of the DNA-spiral and the four different bases are depicted as ball-and-stick models like the ones they remember from basic chemistry in high school. They follow hyper links and read more about chromosomes and genes. On the bottom of an article they find a project which among other things displays chromosomes graphically on the web, Ensembl. Excitedly, they follow the link. They choose to view the X chromosome of the human species. A complicated screen loads. They try to discern some mean-

¹DNA - Deoxyribonucleic Acid. See chapter 2 for a detailed introduction

ing from what they are looking at, but there are too many unfamiliar elements.

They do some searches and find something interesting, "Simple Genome Viewer". It is a simple application, started from the web page with a click. They are presented with a selection of different species, including Homo Sapiens. Again they select the X chromosome. The screen now shows two lines running horizontally across the screen, and there is a number of boxes on the lines. Knut hovers the mouse pointer over one of the boxes and a tool-tip tells him that this is a gene. There is a graphic that tells Knut and Kari that they are looking only at a small portion of chromosome X. There are some buttons to move along the two strands and Knut finds out he can click and drag to move the view also. Kari wants to try now, she spotted the possibility to zoom in and out. She finds a gene and zooms in on it. Another set of boxes appear across the gene she is zooming in on. They are perplexed. Again a tool-tip explains, this is an exon. They access the help function to find out more about what an exon is. Knut takes over and zooms out, he notes that there is a lot of open space between the genes, and even inside the genes there are stretches that seemingly are unimportant. Then Kari points out that there is a gene on one of the strands and a different gene on the other strand right across from it. "There are even overlapping genes on the same strand", Knut remarks. They click on the two genes to read their full description. Verily, the genes have distinct functions. Exhilarated, they continue to explore the genomes. They can't wait to show their class this tool.

The preceding scenario is meant to epitomise the need for an educational genome browser. Several tools that visualise genomes can be found online today, however they are unsuitable for use in secondary education due to the following facts: They present a high number of details, they assume the user possesses a high level of theoretical knowledge and there is little or no focus on usability and ease of interaction.

In this scenario, the browser encountered is called "Simple Genome Viewer", which has been the working title of the educational browser created in this project. An abbreviation of this is "Sigve", a Norwegian, male first name. Note that the objective is not a simpler genome browser. The adjective reflects a strive for a conceptually simpler browser, and a browser that is easier to use. "Simple" is not intended to insinuate plainness or a lack of finesse.

1.2 The Ideal Genome Viewer

As previously indicated, students in upper secondary education are the target group for our genome viewer. The first year of upper secondary education, all students learn, in a subject on Natural Sciences, about the central dogma

of molecular biology. In the second and third year, students can choose to specialise in biology. The adolescents in this stage of their lives are thinking seriously about later studies. The ideal genome viewer is one that can spark these adolescent researchers' interest for the scientific subjects, possibly resulting in an increase in students in the scientific subjects.

The ideal educational genome browser has a lower level of detail than most existing solutions, but retains the core concepts. It can be used by the teacher to underline essential ideas in the learning material. It can be used by the students individually to solve tasks involving exploration of different areas of the genome. It makes information on the roles of the individual genes available, so that students can search for genes highly related to for example cancer. Offering this functionality, the ideal educational genome browser ultimately sparks the students' interest in the natural sciences.

1.3 Structure of the Document

This is a short presentation of the intentions and contents of the remaining chapters of the thesis. The sequential appearance of certain chapters may cause associations with the infamous waterfall model of software development. The actual development taken place did not follow this kind of model. A more iterative, and occasionally impulsive, style of development took place. The waterfall model, however, is fitted in presenting the project, as is the task of this thesis.

Chapter 2 Basic Molecular Biology

This chapter introduces the reader to some of the biological theory associated with the use of genome browsers. The structure of DNA, important mechanisms involving DNA, heredity, and genetic variation, as well as biological databases are presented. Later discussions on the creation of an educational genome browser relies on the reader's acquaintance with this material.

Chapter 3 Introducing Genome Browsers

This chapter explains the purpose with- and usage of genome browsers in more detail. An inspection of some of the many existing genome browsers follows next. A small selection of genome browsers is presented. Each browser is evaluated with regard to its suitability as an instrument of learning. For each browser, comments are made on some features and how appropriate they are in an educational browser.

Chapter 4 Analysis of Requirements and Domain

Chapter 4 starts by defining a set of user requirements and goals for the educational genome browser. Some of these have been found following

correspondence with a teacher such as the one described in the preceding scenario. The chapter then presents an analysis of the problem area. An attempt to find a suitable domain model is also made.

Chapter 5 Designing an Educational Browser

Design issues- and decisions are presented in this chapter. Here, lessons learned from looking at existing genome browsers (chapter 3) are used to solve the requirements laid out in chapter 4.

Chapter 6 Construction

Challenges, important aspects, and decisions made in the construction of the genome browser are presented. Firstly, some programming languages, relevant for various reasons, are introduced, and a selection is made. Java, Python and C++ are evaluated. Next, potential graphics libraries are presented, evaluated and tested. Potential data sources are evaluated following a calculation of the size of the data required. Finally, different programmatic techniques for handling underlying sequence data are presented and tested.

Chapter 7 Introducing Sigve

The end result is presented in this chapter. The functionality is described in detail with screenshots. The chapter can be used as a supplement to the instruction manual in the appendix.

Chapter 8 Conclusions & Further Work

This chapter lists features and functionality that were not finished. It also points out potentials for improvement in the browser. Some suggestions for strategy changes, for example in the choice of graphics library, are also made.

Chapter 2

Basic Molecular Biology

2.1 An Outline of DNA

This section hopes to equip the reader with some familiarity regarding DNA, its structure and purpose. This insight is necessary ballast for effectively reading this thesis. Readers with basic knowledge of DNA, e.g. from upper secondary education or senior high-school, will refresh their knowledge as well and most likely learn a few new details.

2.1.1 Initial Comments

The reader may be aware that all organisms have cells with DNA, however not all cells are organised similarly. The cell of organisms are either prokaryotic or eukaryotic. Cells of eukaryotes have membrane-enclosed nuclei, holding DNA organised into chromosomes. Chromosomes are visible in a powerful microscope, see Figure 2.1. Prokaryotic cells have no nucleus, their DNA is organised into smaller molecules, plasmids. In addition, eukaryotes have membranes and cytoskeletons. The remainder of the chapter focuses implicitly on eukaryotes: It is written with the genome of *Homo Sapiens* in mind. Fundamental differences exist between eukaryotic and prokaryotic genomes, which will not be focused upon herein.

The reader might also be aware that DNA is used to make proteins. This is known as *expression*, and progresses in steps as seen in Figure 2.2. This flow of information is the general course of events known as the *Central Dogma of Molecular Biology*. subsequent sections describes this in greater detail.

2.1.2 DNA and Heredity

Every organism has genetic material, which specifies the biological information of that organism. This material is found in our cells as DNA. Our DNA originates from our parents. And although DNA is transferred to offspring through a complex process, involving recombination and possibly mutations,

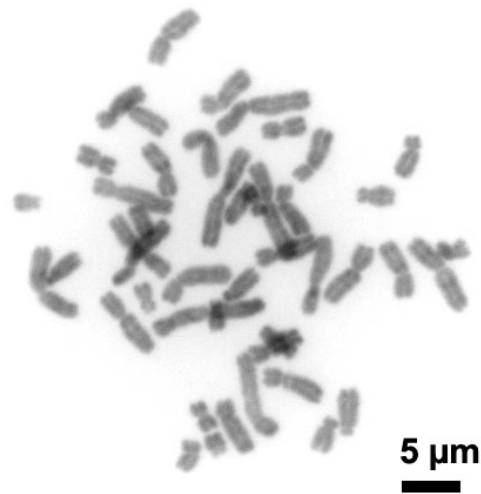


Figure 2.1: Stained chromosomes in condensed (compact) state of a female human lymphocyte. (<http://commons.wikimedia.org/>)

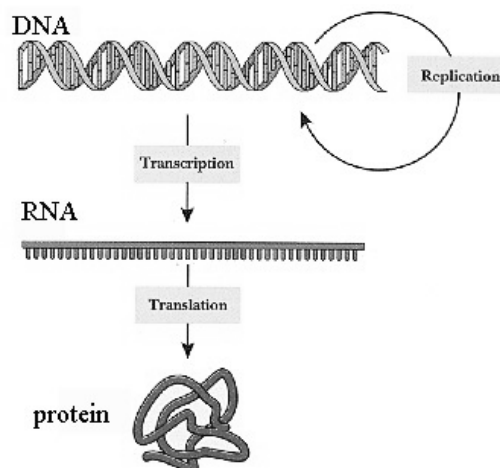


Figure 2.2: From DNA to protein. Part of a DNA molecule is transcribed/copied resulting in an RNA (can be thought of as single stranded DNA) molecule. RNA is translated into a protein. Part of the Central Dogma of Molecular Biology. (<http://commons.wikimedia.org/>)

our physical traits resemble those of our parents. Traditionally, a section of DNA is classified as either coding or non-coding. Coding DNA encodes recipes for proteins, while non-coding DNA has been thought of as junk DNA. Recent research have attributed important functions to these areas. A valid

simplification was that coding DNA determined our physical traits, called phenotypes. Some phenotypes are better understood than others, such as eye colour. However, other phenotypes are determined by a varying number of distinct sections of coding DNA, called genes. Figure 2.2 illustrates the path from DNA to protein.

2.1.3 The Units of Heredity

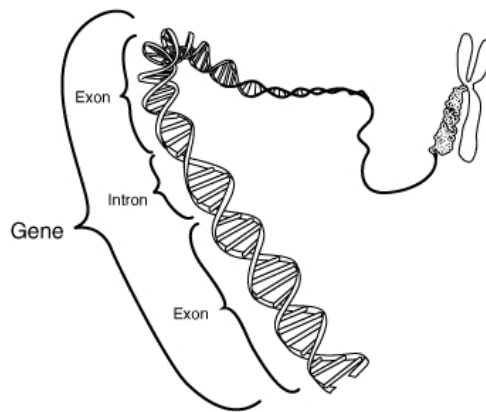


Figure 2.3: A gene with two exons and an intron, which are constituents of genes. (<http://commons.wikimedia.org/>)

Genes are the fundamental units of heredity, they are transferred between generations through DNA and determine the physiology of the organism. A gene consists of a coding region and a set of regulatory regions. The coding sequence is the actual recipe for this gene's product, typically a protein, while the regulatory regions participate in controlling the tendency of this gene to be transcribed. A gene can be thought of as a slot which is filled with some content. All members of a species have the same slots, but their content may be different, i.e. different alleles of the gene are present. In addition, the activeness of the gene varies according to individual-specific factors, organ, tissue type, cell type and even between cells having these characteristics in common. A gene's activeness may also change over time, controlled partially by external signals to the cell.

Whenever a gene is active (expressed), copies of the gene are made, taking the form of RNA. Some parts of the original gene, called introns, are left out, when translating the RNA molecule into a protein. The parts of the gene that are actually translated into a protein are called exons. Figure 2.3 shows an example of a gene on a stretch of DNA.

2.1.4 DNA Structure

In the nuclei of our cells, lies two (slightly different) copies of our hereditary material, DNA. It is packaged in a number of molecules known as chromosomes. The number varies from species to species. Female humans have 23 distinct chromosomes, while males have 24. Chromosomes can be stained by adding a dye (a mix of methylene blue and eosin). This creates bands of different shades of gray across the chromosome, known as Giemsa stains after an early malariologist, Gustav Giemsa. Biologists use the names of the bands to denote positions on the chromosome. In a strong microscope, one can see a human chromosome in its condensed (compact), state. This structure is schematically portrayed in figure 2.4(a). Several types of scaffolding proteins enable almost unfathomably long DNA molecules to be packaged up in this compact structure. On the lowest level, histones are responsible for structured coiling of the DNA strands into stable shapes.

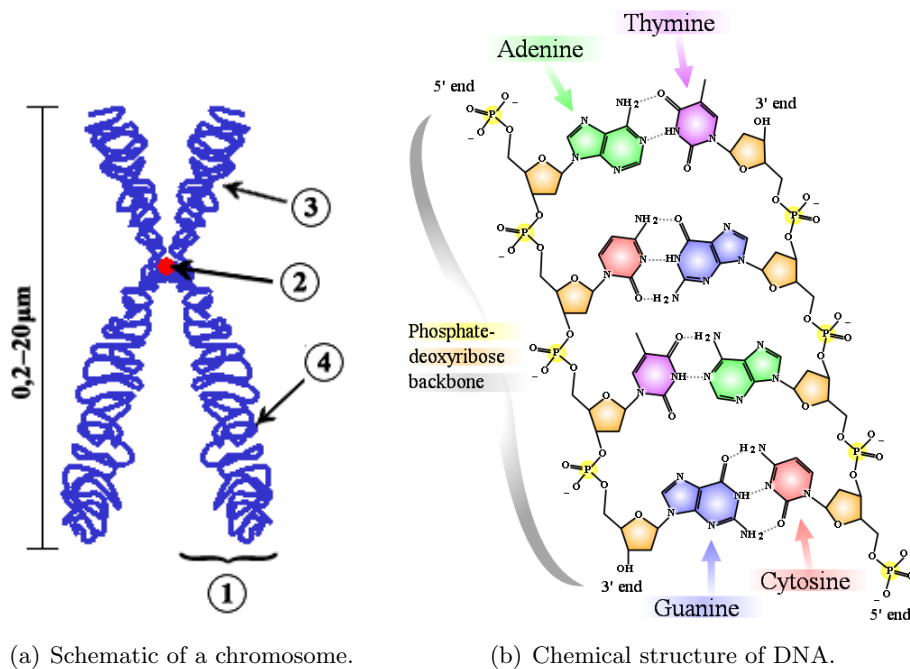


Figure 2.4: (a) Schematic portrayal of a chromosome in its condensed state. 1: One of the two identical chromatids of the chromosome. 2: The centromere, joining point of the chromatids. 3: Short arm. 4: Long arm.

(b) Chemical structure of 4 nucleotides.

<http://commons.wikimedia.org/>

The DNA molecule itself is a polymer, a repeating structure of smaller molecules forming a larger molecule. In DNA, the smaller molecules are nucleotides

consisting of a phosphate group, a sugar (deoxyribose), and one of 4 bases: Either adenine (A), cytosine (C), guanine (G) or thymine (T). Nucleotides form connections between deoxyribose and their phosphate, forming what is referred to as a phosphate-sugar backbone. The sequence of the bases along the phosphate-sugar backbone encodes the genetic information. The bases are further attached to bases on a similar phosphate-sugar backbone, through a one-to-one pairing of the bases. The pairing of bases is not random: adenine pairs with thymine and cytosine pairs with guanine. This forms a lattice structure with two complementary phosphate-sugar backbones, with attached bases. This structure can be seen in figure 2.4(b).

The directionality of the two strands is anti-parallel, meaning the direction in one strand is opposite to the other. So, where one strand starts, the other ends. What is considered as the starting ends are called the 5' ends, the endings are the called 3' ends. This naming reflect a chemical property in the ending nucleotides.

Whenever information from the DNA is needed outside the cell nucleus, for example in the ribosome when some part of it is to be translated into a protein, it is transcribed into RNA. RNA is quite similar to DNA, except for some structural details: RNA is single-stranded, it contains the sugar ribose instead of DNA's deoxyribose, and it uses the base uracil instead of thymine. The process of transcription is detailed further under section 2.1.6. RNA molecules are used for a range of purposes, reflected in the many existing naming prefixes. For example, a messenger RNA is termed mRNA.

2.1.5 The Genetic Code

A gene's nucleotide sequence encodes a sequence of amino acids. I.E. the result of *expression* of this nucleotide sequence is an amino acid sequence, a protein. There are 20 possible amino acids, so 3 bases are required to denote one particular amino acid. Hence, non-overlapping groups of 3 bases code for an amino acid. This gives 64 possible combinations, i.e. codons, to 20 amino acids, which implies some redundancy. This accommodates for a certain amount of aberration in the code. However the codon distribution is not entirely symmetric, and 4 codons are commonly used as indicators of start and stop of transcription. Three amino acids have six codons while two amino acids have only one codon. Tryptophan (IUPAC (International Union of Pure and Applied Chemistry) code: W) has only one codon.

The fact that the genetic code uses 3 bases long, non-overlapping codons means that a given sequence can be read in three different ways, six if we're considering the complementary sequence as well. The six different *reading frames* potentially encode quite different proteins. Note that the differentia-

tion between reading frames is merely a fact of nature, not a systematic rule: In any organism's DNA, genes will reside in all reading frames. There may also be overlapping genes in the same- or different reading frames. Understanding the concept of reading frames, it is easy to understand the higher impact of a deletion (or insertion) type mutation as opposed to a substitution type mutation: Insertions or deletions will affect all subsequent codons, while substitutions only affect the single codon.

Reading frames can be exemplified using the short sequence AATTGGTG. Sequences are measured in bases. A human gene could be 2 kilo bases (kb, thousand bases) long, however a shorter 8 base sequence provides a clearer illustration.

- ◆ Reading frame 1: **AAT TGG TG** - resulting in Asparagine (N), Tryptophan (W).
- ◆ Reading frame 2: A **ATT GGT G** - resulting in Isoleucine (I), Glycine (G).
- ◆ Reading frame 3: AA **TTG GTG** - resulting in Leucine (L), Valine (V).

A reading frame containing a start- and stop codon is called an open reading frame, ORF for short. DNA is typically full of ORFs, potential genes. However, only a fraction are actual genes. Looking for ORF is a strategy for finding genes, using special criteria which depend on what type of organism the DNA is from.

2.1.6 Expression

Expression is the process in which a section of DNA is copied to mRNA, transported outside the nucleus (to the ribosomes) and translated into a protein. That is, the copy is transported etc. not the DNA original, which never leaves the nucleus. The copy takes the shape of an mRNA-molecule, similar to DNA except the most important facts: that it is normally single stranded, employs the base uracil in place of thymine, and consists of a different pentose sugar than in DNA.

The resulting proteins have all sorts of functions depending on which gene has been expressed. For some genes the RNA molecule is the final step in the expression, RNA products typically have intra-cellular functions. These are important steps in the expression of a gene:

Transcription

RNA Polymerase is the central enzyme of DNA transcription. With the help of proteins called transcription factors, it binds to the DNA. It then reads

the strand of DNA that is complementary to the one to be copied, from 3' to 5', synthesising the RNA-molecule from 5' to 3'. By doing this, it creates an exact copy of the coding strand. The RNA molecule output of this process is known as pre-mRNA or precursor mRNA (messenger RNA).

Post-transcriptional Modification

This process prepares the pre-mRNA for translation into a protein. The most important modification made is the splicing of the RNA molecule, illustrated in figure 2.5. In some cases, the splice process reshuffles the exons, or leaves an exon out, giving rise to alternative splice variants. This heightens the efficiency of the genetic code, increasing the number of proteins possibly produced by each gene. Modification also adds a stabilising cap to the 5' end, and a signaling region to the 3' end.

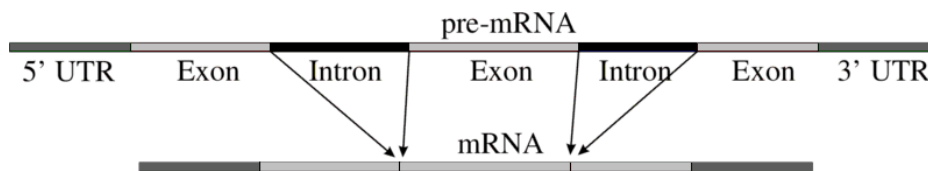


Figure 2.5: The splice process, removing introns. The leading and trailing untranslated regions (UTRs) are not translated into amino acids, but contain some regulatory information. (<http://commons.wikimedia.org/>)

Translation

The transcript has now become a mature mRNA. Translation takes place in the cytoplasm, outside the nucleus. An RNA complex, the ribosome reads the mRNA codon by codon and adds the corresponding amino acid to the growing polypeptide (protein) chain.

2.1.7 Genetic Variation

Mutations may arise from a number of causes and are a source of genetic variation. Different types of mutations exist, and can establish themselves in the population if it is carried over to the offspring of the individual with a mutation. A mutation that has established itself in a population becomes a polymorphism. The impact of different types of polymorphisms upon genetic variation is continuously researched.

Copy Number Variation

A copy number variation (CNV), is present if there is a difference in the number of copies of a certain gene or a genetic region. If a certain CNV establishes itself in a population, it is called a copy number polymorphism (CNP). If the norm is two copies of a gene and a group of individuals have four copies, that gene would potentially be doubly expressed in these individuals. Sebat et al. [2004] and Kehrer-Sawatzki [2007], among others, report that copy number polymorphisms contribute substantially to genetic variation.

Single Nucleotide Polymorphism

Single nucleotide polymorphisms (SNP) are differences in a single nucleotide that have become established in a population. A criteria used for considering a SNP as established is that it is present in 1% of the population. As an example, in a certain base the population norm is a T, whereas some individuals have C. A major source of SNP identifications, the International HapMap Project aims to provide a complete haplotype map of the human genome, describing all common patterns of human DNA sequences.

Even though SNPs are more numerous than CNVs, it is likely that CNVs are just as important a factor in genetic variation as SNPs. In dbSNP, a major SNP database, there are in excess of 6 million verified SNPs in the human genome (over 12 million in total). Compared to the number of known CNVs, just under 9000, the amount of SNPs are staggering. Estivill and Armengol [2007] state that CNVs account for over 15% of the variance in the assembled human genome. For SNPs this number can be calculated from the amount of SNPs and total amount of bases to be 0.004%. SNPs have been associated with more diseases than other types of polymorphisms, because it has been easier to detect. In addition, SNPs are a much easier mechanism to understand than most other kinds of changes. For this reason, for the sake of explaining genetic variation, using SNPs as examples thereof is a good idea.

SNP information is available from many sources. The database dbSNP has been mentioned. The web site SNPedia shares peer-reviewed scientific publications of information on phenotypes governed by SNPs.

2.2 Biological Databases

This is a small introduction to biological databases, their content and purpose, within the realm of biological data. Biological databases play a major role in molecular biology and bioinformatics. Much of the work done within these disciplines revolves around biological data and meta-data. Biological databases store a range of different kinds of information, used for a variety of purposes.

Generally, three types of biological data are stored in such databases. Primary data, e.g. sequence data; secondary data, e.g. sequence annotations; and tertiary data, which is used as a resource in whole- or multi genome comparison and analysis. Herein, I focus on biological databases containing variants of primary and secondary data.

2.2.1 Sequence data

An example biological database is the GenBank sequence database, which is hosted by the National Center for Biotechnology Information (NCBI). GenBank stores the complete human genome sequences as well as sequences from a wide range of other species. The size of the latest release of GenBank measured in base pairs is 79 billion (7.9×10^{10}). The human genome is one of about 700 complete genomes, a further 1300 are either incomplete or nearly complete [NCBI, 2008]. In NCBI's Taxonomy Browser, the lineage and taxonomic position of more than 265 000 organisms can be found, that have at least one sequence stored in GenBank. The exact amount at the time of writing was 265 071 organisms.

The Human Genome Project (HGP) which was finished in 2003, had among other goals the intention of sequencing the entire human genome. Presently at build 36, the nucleotide sequence of the human genome counts approximately 3.1 billion bases. The HGP enjoyed the benefits of successive advances in whole genome sequencing techniques, and in biosciences in general. The result of this was that the project progressed faster than initially planned and that milestones often were reached ahead of time and with the results holding a higher standard than was initially aimed for [Collins et al., 2003]. The post-HGP period has also been marked with great advances in whole genome sequencing. This is reflected by the fact that the GenBank database is growing at an exponential rate, doubling in size every 10 months. The human genome, together with sequences of other species available in GenBank, presents a substantial amount of data. Sequence databases are not limited to holding nucleotide sequences. In GenBank, protein sequences are also available. In fact sequence comparison using protein sequences yields more distant relatives, called distant homologues, and is therefore more common.

Data Formats

A few different plain-text file formats for genetic sequences are used. The FASTA format was designed as the input format to a software package for sequence alignment (known as FASTA or FASTP). The FASTA format encodes a sequence of base pairs or amino acids using single-letter codes. An example of an amino acid sequence can be seen in Figure 2.6. An initial header-line contains version info and more. The simplicity of the FASTA format means

```

>gi|47576196|ref|NM_001000520.1|Rattus norveg. olfactory receptor 1346
ATGGCCACACAAGTGCACAGAAACGGAAGTCTCTCAGCAGTGCCTTGCAGGGGTTTCGTTCTGGTAGGGT
TTGGGGGAAGTGCAGAGACCCAAGCTCTGCTCTTTGCTGTGTTCCCTAATCATGTATGTAGTACTGTCCT
GGGCAACCTCACCATGATTGTGGTCATCACTCTGGATGCCCGCCTGCACTCCCCATGTACTTCTTCCTC
AAGAACCTGTCCTTCGTTGACCTCTGTACTCTTCTGTTATTGTCCCAAAGCCATGGCCAACTTACTTT
CTTCCACTAAGGTCATCAGCTTTGCAGGATGTGCCACTCAGTTCTTCTTTTTCTCCCTTCTGGTTACTAC
TGAAAGCTTTCTATTGGCAGTCATGGCCTACGATCGCTTCATGGCCATCTGCAGTCCCCTGAGGTACCCT
GTGACCATGTGCCCTATGGCATGTGCCCGTCTGGTCTGGGTGCCTACTGTGGTGGCTGCCTGCACTCCA
TCATAGAGAGCAGCCTCACGTTCCGGCTGCCCTTCTGCAGCTCCAACCGTATCAACCACTTCTACTGTGA
TGTGCCCCCATTGCTCCAGCTGGCCTGTGCTGACACAACCTCTCAATGAGCTTGTGCATGTTTGGCATCTGT
GGACTCATCATCGTGTCTACCACTCTCGTGGTCTGCTCCTATGGCTACATCACAGTGACCATTCTCA
GGATGCGCTCTGGGTGAGCCGGCACAAGCTCTTCTACTTGTGGTTCACACATGACAGCTGTGTCCTT
GTTTTATGGAAGTGTGTTTGTGCATGTATGCTCAGCCAGGCGCTCTGACATCCATGGAGCAGGGGAAAGTG
GTCTCTGTCTTCTACACCCTGGTTATCCCATGCTGAACCCCTCATCTACAGCTGCGAAACAAGGATG
TGAAGGATGCCCTTAGGAGGCTGGGACAGAGGCACAGTCTTGTGAAGGAGGATGTGCAGTGA

```

Figure 2.6: Example of the FASTA file format, encoding a gene related to smell in rats.

that it is easily manipulated using text processing- or scripting tools.

The GenBank format features a more extensive preamble than that of FASTA. A GenBank record contains extensive version and species information, as well as a large list of references as evidence for the record. Dissimilarities in the representation of sequences between the two formats is evident in Figure 2.7.

Whose sequence is it?

The DNA sequence of *Homo Sapiens* found in Genbank isn't the sequence of a particular individual, although an unrelated release has been made of the sequence of a known individual, James Watson. The sequence available in Genbank is the consensus sequence of the samples sequenced and assembled during the HGP. Collecting samples for the HGP was done by local public advertisements around the areas of the participating laboratories. Samples were de-labeled before selecting 5 to 10 percent of the samples for sequencing, unbeknownst to the sample donors [NIH, 2008].

2.2.2 Sequence Annotation

Sequence annotation is an important companion to raw sequence data, in that it adds meaning to what in itself is a sequence of characters. Sequence annotation data typically describes where genes are situated, where transcription starts or describes areas that are a source of genetic variation. Ensembl is a database where annotation is created automatically, using statistical methods. Sequence annotation is traditionally (and occasionally still) created manually.

```

LOCUS      NM_001000520          972 bp    mRNA    linear    ROD 10-FEB-2008
DEFINITION Rattus norvegicus olfactory receptor 1346 (Olr1346), mRNA.
ACCESSION  NM_001000520 XM_237406
VERSION    NM_001000520.1  GI:47576196
KEYWORDS   .
SOURCE     Rattus norvegicus (Norway rat)
  ORGANISM Rattus norvegicus
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Glires; Rodentia;
            Sciurognathi; Muroidea; Muridae; Murinae; Rattus.

...

CDS        1..972
           /gene="Olr1346"
           /note="olfactory receptor gene Olr1346"
           /codon_start=1
           /product="olfactory receptor Olr1346"
           /protein_id="NP_001000520.1"
           /db_xref="GI:47576197"
           /db_xref="GeneID:316633"
           /db_xref="RGD:1333294"
           /translation="MATQVHRNGSLSAVSLQGFVLVGFGGSAETQALLFAVFLIMYVV
TVLGNLTMIVVITLDARLHSPMYFFLKNLSFVDLQYSSVIVPKAMANLLSSTKVISFA
GCATQFFFFSLLVTTESFLLAVMAYDRFMAICSPLRYPVTMCPMACARLVLGAYCGGC
LHSIIESSLTFRLPFCSSNR.INHFYCDVPLLQLACADTTLNELVMFGICGLIIVSTT
LVVLVSYGYITVTILRMRSGSGRHKLFSTCGSHMTAVSLFYGTVFVMYAQP GALTSME
QGKVVSVFYTLVIPMLNPLIYSLRNKDKDALRRLGQRHSLVKEDVQ"

ORIGIN
1 atggccacac aagtgcacag aaacggaagt ctctcagcag tgtccttgca ggggttcggt
61 ctggtagggt ttgggggaag tgcagagacc caagctctgc tctttgctgt gttcctaatac
121 atgtatgtag ttactgtcct gggcaacctc accatgattg tggatcatcac tctggatgcc
181 cgcctgcact cccccatgta cttcttcctc aagaacctgt ccttcgttga cctctgttac
241 tcttctgtta ttgtcccaa agccatggcc aacttacttt cttccactaa ggtcatcagc
301 tttgcaggat gtgccactca gttcttcttt ttctcccttc tggttactac tgaagcctt
361 ctattggcag tcatggccta cgtcgcctc atggccatct gcagtcccct gaggtaccct
421 gtgacctgtg gccctatggc atgtgcccgt ctggctcctgg gtgcctactg tgggtgctgc
481 ctgcactcca tcatagagag cagcctcacg ttccggctgc cttctgcag ctccaacctg
541 atcaaccact tctactgtga tgtgccccca ttgctccagc tggcctgtgc tgacacaact
601 ctcaatgagc ttgtcatgtt tggcatctgt ggactcatca tcgtgtctac cactctcgtg
661 gtccctggtc cctatggcta catcacagtg accattctca ggatgcgctc tgggtcaggc
721 cggcacaagc tcttctctac ttgtggttca cacatgacag ctgtgtcctt gttttatgga
781 actgtgtttg tcatgtatgc tcagccaggc gctctgacat ccatggagca ggggaaagtg
841 gtctctgtct tctacacctt ggttatcccc atgctgaacc ccctcatcta cagcctgcga
901 aacaaggatg tgaaggatgc ccttaggagg ctgggacaga ggcacagtct tgtgaaggag
961 gatgtgcagt ga

```

Figure 2.7: Example of the GenBank file format, encoding a gene related to smell in rats. Portions of the preamble is omitted here. Note that the translated amino acid sequence is included. Notice the start- and stop codons (ATG and TGA, respectively) appearing in the same reading frame.

The Vertebrate Genome Annotation (VEGA) database contains manually curated sequence annotation for a selected set of species, including *Homo Sapiens*. Therein, human annotation data is the product of the Havana group of Wellcome Trust Sanger Institute.

2.2.3 Genomic Meta-Information

Some projects output information that describes or categorises the primary types of genomic data. An example is Mendelian Inheritance in Man (MIM), available electronically through NCBI in the form of Online Mendelian Inheritance in Man (OMIM). It is a database cataloguing heritable, genetic diseases related to human genes. Searching with the name of a disease as keyword, yields records relating this disease to specific genes. A search for cancer will result in a long list of records relating different forms of cancer to different genes. One such record is named "Breast Cancer". This record summarises many years of research into the interplay of genetic variations leading to breast cancer.

Another meta-information project is the Gene Ontology (GO) project. This project consist of two distinct parts. The first is an ontology, that is a controlled vocabulary for describing genes and their products. The second part is the gene annotation database linking known genes with a set of GO-terms describing the gene or gene product's molecular function, its role in biological processes and localisation to cellular components.

Recall from our search in OMIM, the result "Breast Cancer". One gene mentioned in this record is BRCA1 or Breast Cancer 1. Inputting this in the search function on the GO web site yields a list of descriptions of this gene. Among the most interesting are these biological processes in which BRCA1 is involved:

- ◆ DNA damage response
- ◆ chromosome segregation
- ◆ positive regulation of DNA repair
- ◆ post-replication repair

This demonstrates the value of combining searches in several databases.

2.2.4 Details of the Human Genome

Table 2.1 lists some interesting updated details of the human genome. The number of novel-, pseudo-, and RNA genes gives a slight indication of how complex the system is. The length of the golden path is the length of the

Assembly:	NCBI 36, Oct 2005
Genebuild:	Ensembl, Dec 2006
Known protein-coding genes:	21,541
Novel protein-coding genes:	1,199
RNA genes:	4,421
Pseudogenes:	2,081
Genscan gene predictions:	69,073
Gene exons:	275,708
Gene transcripts:	48,400
SNPs:	13,099,397
Base Pairs:	3,253,037,807

Table 2.1: Ensembl: Homo Sapiens Genome Statistics. Notice relative amount of Genscan (An algorithm that attempts to find genes in unknown sequences, using statistical methods) gene predictions compared to amount of known genes.

longest continuous sequence that has been sequenced (actually put together from a set of shorter sequences, sequenced laboratorially). The golden path is commonly shorter than the total length, because the ends and centres of chromosomes can't be sequenced using current methods.

2.2.5 Summary

Biological databases form much of the basis for biology, biomedicine, bioinformatics as well as other scientific disciplines. To biologists researching in the fields of metabolism and evolution, biological databases are an invaluable source of information. The development of software for comparison and analysis of biological data is one aspect of bioinformatics. Indeed, bioinformatics arose as a discipline from the need to create large databases to store the rapidly growing amounts of biological data.

2.2.6 Genome Browsers

Genome browsers present genomic data from sources such as those mentioned in the preceding sections. The insurmountable data amounts defining a species is not suited for human comprehension textually. Genome browsers help users understand this data through visual presentation. The following chapter describes motivating factors in more detail, and then examines a few existing browsers.

Chapter 3

Introducing Genome Browsers

3.1 Introduction

In this chapter, the concept of genome browsers is introduced in more detail. How-, why-, and by whom such software is used is explained. The chapter then examines a small selection of what has become a myriad of existing solutions for visual interaction with genomes. I observe the main features of these solutions, discussing if they should be pursued in a new, simpler genome viewer. The main criteria for this is suitability for learning about the genomic structure in general.

3.2 Genome Browsers

3.2.1 Motivation

With many organisms having been completely sequenced, the need for effective presentation of these genomes has arisen. Traditionally, an organism's DNA was either coding or non-coding. Research has shown that the ontology is far more complex than this binary categorisation. While the complexity of this data is shown to be more and more complex, the human capability for understanding remains constant. The primary motivation for visualising genomes is therefore to help understanding more of the picture regarding a genome.

Using the human genome as an example, 3 billion (10^9) base pairs as plain text, the futility of trying to gain understanding becomes apparent. Furthermore, there are a high number of features and objects in an organism's genome that constitute the defining parts of that genome. Considering Homo Sapiens, the numerousness of different features are summarised in Table 2.1. Draw-

ing a comparison between genomes is even more hopeless. Genome browsers are attempts to ameliorate the situation. Genome browsers are tools that present genomic information visually for different purposes. Genome browsers rely heavily on biological databases. A selection of popular existing genome browsers is explored in following sections. On the most basic level, a genome browser displays entire genomes, utilising sequence data from one or more species. The usefulness of visualising a character sequence is limited, however displaying sequence annotation data on top of this quickly communicates more meaning. This combination gives researchers a visible map of the genome and the structure, product-coding status, regulation in different regions. Areas of interest can be explored in progressively greater detail, right down to the very sequence of amino acids or even bases.

3.2.2 Users

Traditionally there have been two general types of users of genome browsers. Experts in fields such as molecular biology, genetics or medicine and non-expert users who come across browsers by chance. Close to all existing browsers are created with the first group in mind.

3.2.3 Abilities and Uses

A basic genome browser displays annotation data and/or sequence data. A visual presentation of such information can be useful to a biologist or a medical doctor in researching a particular gene. The surroundings of the gene can be explored for related features, and the relative level of genetic variation in the area can be assessed.

Some genome browsers also have the ability to show sequences with annotation from several organisms, aligned. This is useful when investigating common features of a number of organisms.

A genome browser might also display genetic markers. Put shortly, genetic markers are "identifiable portions of a chromosome whose inheritance patterns can be followed", according to Xiong [2006]. They can be used to hunt for the genetic cause of a hereditary disease. It is known that the closer two features are situated on a chromosome, the more likely they are to be inherited together. Thus, finding a specific genetic marker, it is likely that the gene we are looking for, which hasn't yet been localised accurately, lies in the vicinity. Finding the genetic marker in individuals with the disease while other individuals miss it, then points to the gene being at least partly responsible for the disease.

A special use for genome browsers can be created by combining the "ba-

UCSC Genome Browser	http://genome.ucsc.edu/	University of California Santa Cruz has this browser. Detailed and scientific. Several organisms.
Ensembl	http://www.ensembl.org	A cooperation between the European Bioinformatics Institute and European Molecular Biology Laboratory. Detailed and complex browser. Many organisms available.
NCBI Entrez Map View	http://www.ncbi.nlm.nih.gov/mapview/	American national centre for biotechnology provides this browser. Comprehensive and detailed. Links to OMIM. Many organisms.
X:Map Genome Browser	http://xmap.picr.man.ac.uk/	Created by the bioinformatics group at a cancer-research group associated with the University of Manchester, UK. Visualises clone distribution in the human genome.
Argo Genome Browser	http://www.broad.mit.edu/annotation/argo/	Developed at Massachusetts Institute of Technology. Open source. Possibility of browsing several different genomes.
VISTA Genome Browser	http://pipeline.lbl.gov/cgi-bin/gateway2	The Lawrence Berkley Laboratory is responsible for the VISTA family of software tools which includes a genome browser. The browser is a dynamic java applet and many organisms are available. It can visualise alignments of multiple genomes.

Table 3.1: List of genome browsers

sic” data with other types of data. An example of such a browser is X:Map which is further discussed later in the chapter. This browser is useful for designing DNA microarray experiments. DNA microarray experiments probe the expression level of genes. A microarray is a small chip with thousands of probes, appearing as spots, representing the expression of a single gene. Starting with a list of genes, selecting probes is typically done by accessing the so-called clone database of a producer of microarrays. Affymetrix is such a producer. Using the afore-mentioned genome browser, the researcher designing this experiment can find clones for the neighbouring genes or features, that in his eyes are also interesting for the experiment.

3.2.4 List of Genome Browsers

A list of some genome browsers follows, with name, web address, and short description of each. Appearance in the list signifies notability, according to the author of this thesis.

3.3 Presenting a Selection of Browsers

The selection of genome browsers presented herein, contains two browsers that are the responsibility of two of the largest bioinformatics organisations. They are therefore assumed to be among the most advanced, feature-rich and popular browsers. They are rich in detail and functionality. The selection contains also a less known browser, selected since it stands out from most other browsers in its simplicity and interaction style. Selecting just a few browsers in this way will leave many out, some of which may be just as well known as the ones that made my selection. One such browser is the University of California, Santa Cruz genome browser. This is a popular and well known browser, but it does not stand out from the first two on my list, when it comes to functionality and appearance, and is thus left out.

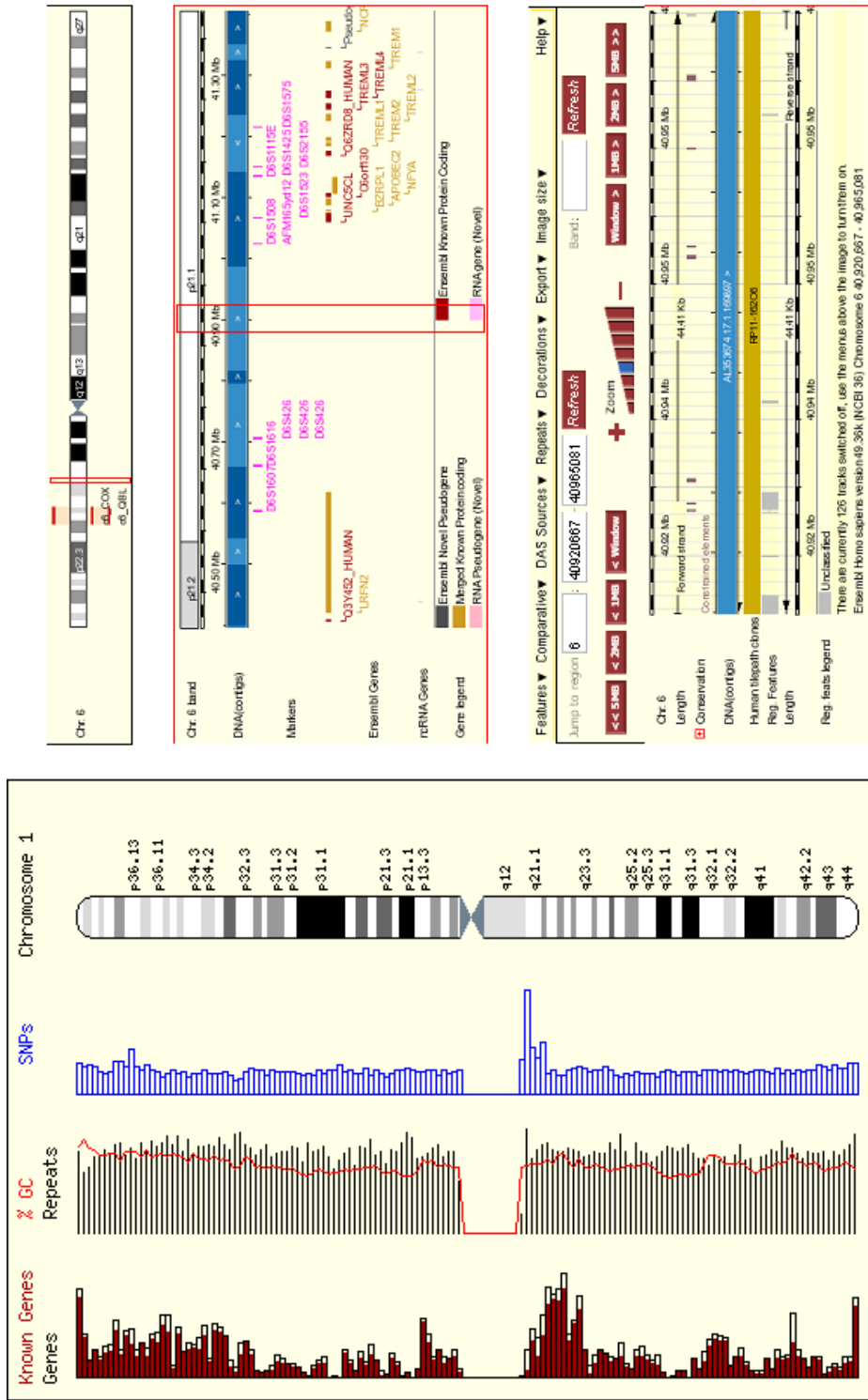
The first genome browser I look at is Ensembl, a European browser. It is quite complex, consisting of multiple views with different level of detail. The second browser is the NCBI Map Viewer. This browser has much of the same functionality as Ensembl, presented with a very different interface. The last browser presented is X:Map which was created by the Bioinformatics Group at the Paterson Institute for Cancer Research, which is connected to The University of Manchester, UK.

3.4 The Ensembl Genome Browser

Ensembl is a joint project by the European Molecular Biology Laboratory (EMBL), the European Bioinformatics Institute (EBI) and Wellcome Trust - Sanger Institute (WTSI), to enable automatic analysis of genomes and generation of genome annotation. Besides sequence material download opportunities, their web-based genome browser provides a means for visually exploring the genomes in their databases.

3.4.1 Using the Browser

Genome browsing is commenced by first selecting an organism from a list on the index page of the web site. This will load an overview of the different units that the selected genome consist of. In case of the human genome, one will see 25 different chromosomes (chromosome 1 - 22, X and Y, as well as chromosome MT representing the mitochondrial DNA). Clicking on any chromosome, will enter Map View, which gives some details of the chromosome and an overview of the selected chromosome, with 3 histograms showing the spatial distribution of genes, GC-repeats and SNPs along the chromosome. A screen shot of this overview is shown in Figure 3.1(a).



(a) Ensembl: Overview of chromosome 1 showing SNPs, repeat content and gene density along the chromosome.
 (b) Ensembl: Contig view: Chromosome overview, Overview and Detailed view.

Clicking this overview will open Contig View, centred to the clicked location. Contig View consists of up to 4 separate views: Chromosome view, Overview, Detailed view, and Basepair view. Chromosome view is an overview of the entire chromosome and the current position within it. Overview is simply a smaller, local overview. Detailed view shows a stretch of the current chromosome 100 000 bases or 100kb (kilobases, thousand bases) long. Basepair can show underlying amino acid- and nucleotide sequence, but is deactivated when not zoomed in close enough. An overview picture of Ensembl Contig view can be seen in Figure 3.1(b).

3.4.2 Interaction

Ensembl Genome Browser employs a click-and-wait like interaction style, the reason for which lies in its web-based nature. Data for displaying is fetched, analysed and displayed on the go.

The user has a plethora of tools for zooming in and out. Clicking and dragging creates a rectangle which the user can select either to zoom in on, center on or view in Base pair view. While clicking and dragging like this in either of the overviews, selecting too large of a section to display in Detailed view or Base pair view, results in nothing being displayed. Furthermore, right-clicking in any view, opens a context menu with options for zooming in- or out and centering. There are also input fields for entering a specific interval of bases to display. A host of buttons allow quick access to specific zoom levels, and functions for jumping set amounts of base pairs forward or backward, as can be seen in the screen shot in figure 3.2.

3.4.3 Features Visible

By default, Ensembl Genome Browser displays several different types of features and structures. Features in this context, includes genes, transcripts, markers and features in general. Structures include single nucleotide polymorphisms. Different types of RNA, genes and so on are displayed, if present. Furthermore, a distinction is made on the identifying source of the displayed features. Features are organised into tracks that can be switched on and off, being displayed or not. Each track displays a specific type of feature from a specific source. Table 3.2 presents a short summary of some sources.

Certain tracks are displayed (switched on) by default, but the majority are hidden. Still, the amount of detail presented in the default view is vast. In addition, more often than not, a feature identified by one source is identified by the other sources as well.

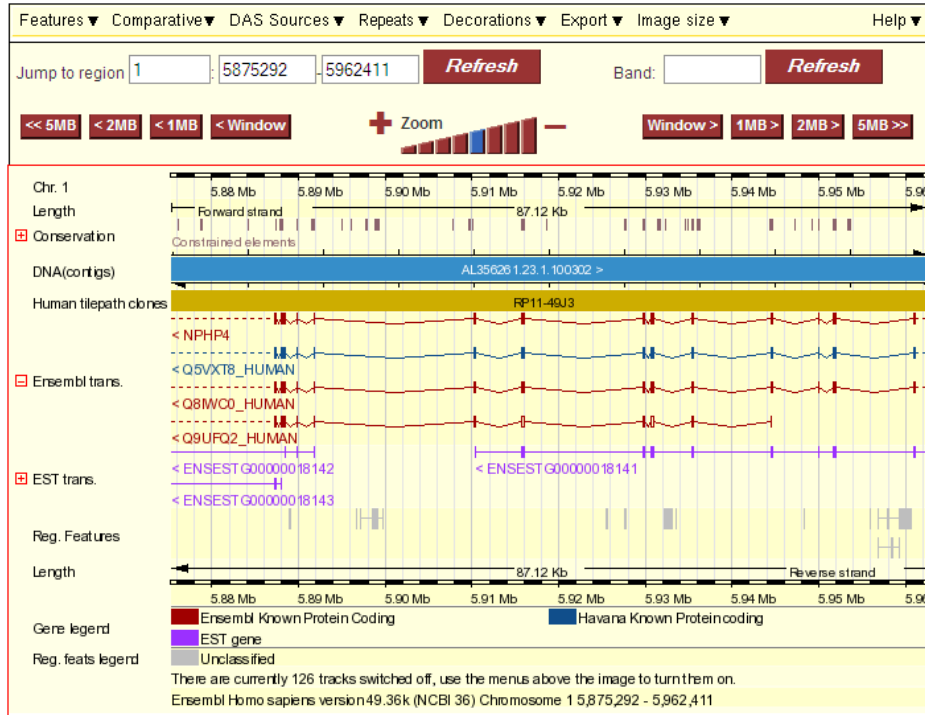


Figure 3.2: Ensembl: Detailed view

Ensembl	Features identified by Ensembl model predictions, using the gene-building procedure in the Ensembl analysis and annotation pipeline.
Vega Havana	Features manually identified by the Havana group at WTSI, obtained from the Vega database.
Vega External	Features manually identified by an external group, obtained from the Vega database.
SNAP	Features predicted ab initio by the Semi-HMM-based Nucleic Acid Parser.
Genscan	Features identified ab initio by the GENSCAN gene prediction program.

Table 3.2: Ensembl feature data sources

3.4.4 Level of Detail

As previously mentioned, Ensembl supports, through the use of separate windows, zooming from entire chromosomes down to the very sequence of amino acids and even bases.

3.5 The NCBI Map Viewer

The National Center for Biotechnology Information is a department of National Library of Medicine, which in turn is a branch of the United States National Institutes of Health. It was founded in 1988 and provides access to the genomic sequences in GenBank, medical publications in PubMed as well as other databases of biotechnically related information. It provides search functions to these databases through the Entrez search engine. The topic of discussion is its Map Viewer. The Map Viewer supports search and display of genomic information by chromosomal position. Regions of interest can be retrieved by text queries (e.g. gene or marker name) or by sequence alignment (BLAST - Basic Local Alignment Search Tool).

3.5.1 Using the Browser

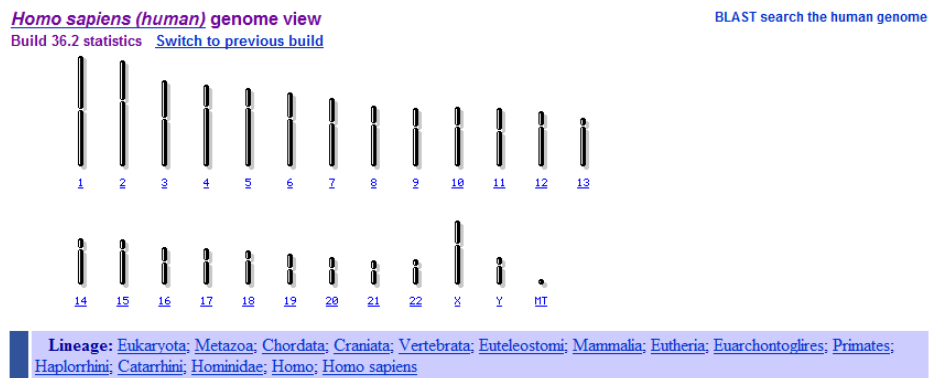


Figure 3.3: NCBI: Genome view

In much the same way as the Ensembl Genome Browser, NCBI Map Viewer opens with a selection of different species to choose from. In addition to a list of species, there is a tree representation, showing the species, or groups of species in a homology tree, a tree where the root is the common ancestor of the leaves.

Once a species is selected, its selection of chromosomes are shown, with names,

and icon size demonstrating relative lengths. This overview is depicted in figure 3.3. Clicking a chromosome opens the detailed view. The detailed view shows features in a part of the selected chromosome. The user can zoom in and out and an ideogram or mini map lets the user see the current position of the view. The underlying base sequence can be shown by clicking the desired point in the sequence.

3.5.2 Interaction

NCBI Map Viewer employs a click-and-wait style of interaction similar to that of Ensembl's browser.

Panning and zooming is done by clicking the chromosome which makes a small floating window to appear, letting the user select from either: recenter, zoom in x2, zoom in x4, zoom in x6, zoom in x8, or zoom out x2. Interaction can also be done with the other tracks visible. The user is then given additional options, e.g. to view the sequence for the clicked gene.

The Map Viewer focuses heavily on availability of information, judging from the amount of links to different databases provided in connection with features visible in the browser. The large amounts of text necessary to provide this, draws attention away from the visible features themselves.

3.5.3 Features Visible

The Map Viewer's detailed view is column-based. The view can be seen in figure 3.4. In Table 3.3, the content of each column is listed, from left to right.

3.5.4 Level of Detail

The detailed view can be zoomed in indefinitely. No additional detail is made visible by doing this. If users want to see the underlying base sequence, they must click the sequence maps and select show sequence, opening the sequence up in a new window. The sequence can easily be put into BLAST for a homology search.

3.6 The X:Map Genome Browser

X:Map is built on top of data from Ensembl and Affymetrix Chip Definition Files. Data from Ensembl is used to visualise features, while Affymetrix Chip Definition Files are used to indicate probeset matches along the DNA sequences. Its browsing interface is built using the Google Maps API (Application Programming Interface). Google Maps is an online map service much like map services found on Norwegian sites gulesider.no and finn.no. X:Map

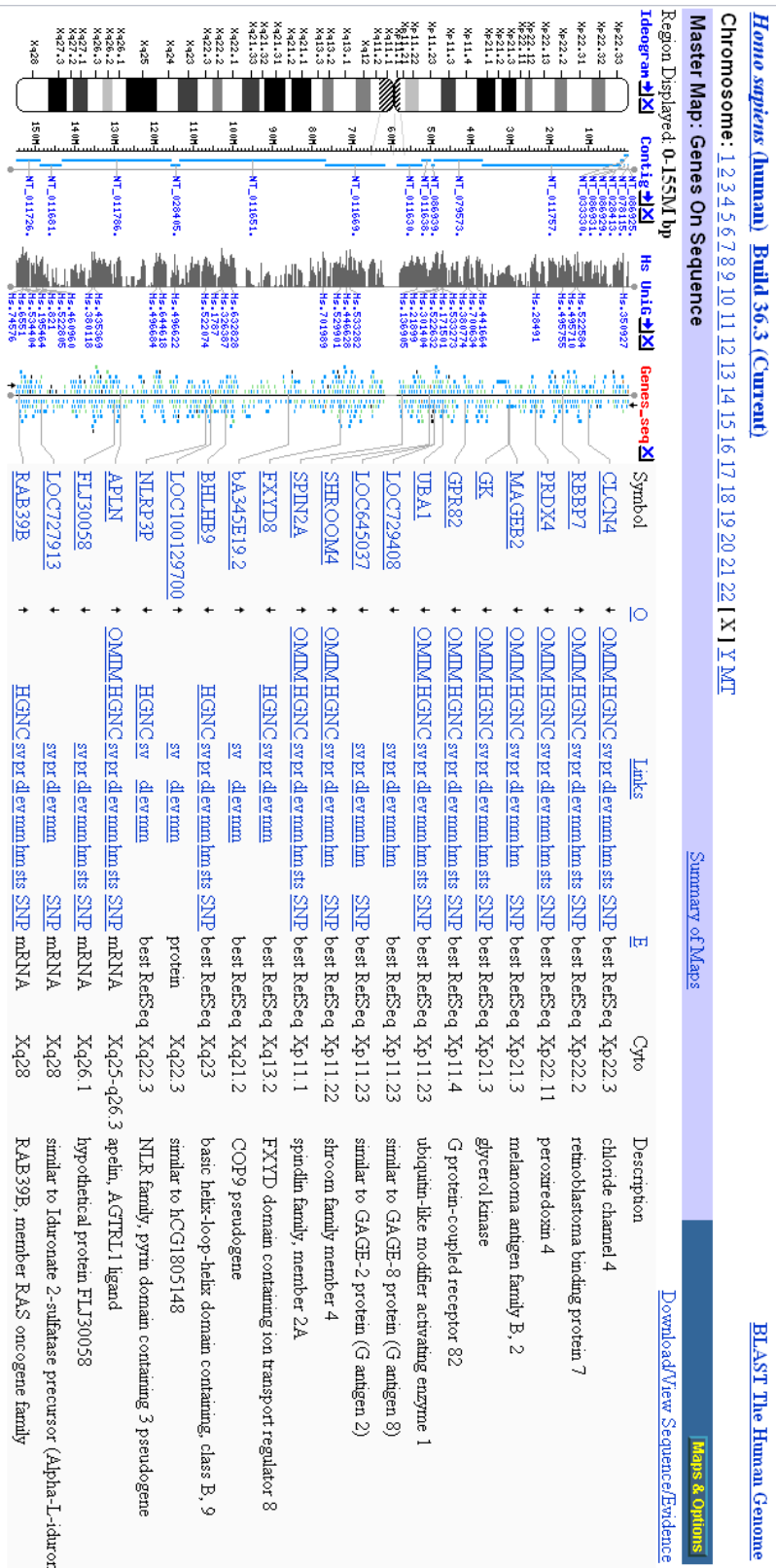


Figure 3.4: NCBI Master view. The entire chromosome X is shown vertically on the left-hand side. Each line describes a gene located approximately in that position on the chromosome.

Ideogram	Map of the section of the chromosome currently under scrutiny.
Contig	Map of the sections of DNA that has been separately sequenced in this range.
Gene Clusters (UniGene)	Map of the density of sequences that have been found to be expressed as e.g. mRNA.
Genes on Sequence	Map of genes in this portion of the chromosome. They are shown graphically as boxes on either side of a vertical line, being on the forward or reverse strand.
Selected Genes	20 of the genes located in the region in view, with names and IDs are listed. Lines are drawn pointing to each genes location in the graphic showing genes on sequence, in the preceding column.
Orientation	The orientation of the selected gene in this row is illustrated with an arrow.
Links	The neighbouring columns link the selected genes to different databases.
Evidence	The evidence locating the genes to this area.
Cyto	Which cytoband this gene is positioned in.
Description	The last column contains descriptive data on each selected gene.

Table 3.3: NCBI Map Viewer: Data columns and descriptions

is interesting in this context due to its simplicity. Because of this it falls closer to what this project tries to accomplish than any of the previously discussed browsers.

3.6.1 Using X:Map

When loading the web page, the browser loads a detailed view of the starting 30 000 base pairs of *Homo Sapiens* chromosome 1. A horizontal bar illustrates the DNA sequence, and annotated features is shown as boxes over or under this bar, depending on the strand on which the feature is located. In a hidable Tools-panel docked to the right of the window, drop down boxes indicate which species and chromosome are selected. The genome browser is immediately ready for interaction. Figur 3.5 shows the view to which X:Map opens.

3.6.2 Interaction

Clicking and dragging pans the view left or right, while Google Maps style buttons also pan the view as well as zoom in and out. Clicking the view places a vertical red line, called the cursor, at the clicked location. Any Features intersected by the red line, shows up in a list in the Tools-panel. In this list

each feature is identified by an icon describing its type, a 'G' icon identifies Genes and a 'P' icon identifies Probesets, and the identity of the Feature. An example of this identity is ENSG00000177693. This is Ensembl's id for this gene. Clicking a feature in the list activates a view below the browser itself. This view is split in two, with a tree representation of the selected feature on the left, and a list of details on the right. For ENSGG00000177693, this view is depicted in figure 3.6.

3.6.3 Features Visible

In X:Map, genes and their sub-features (transcripts, exons) as well as are visible by default. The feature tracks ESTs (Expressed Sequence Tags), Genscan genes and DNA repeats can be switched on.

3.6.4 Level of Detail

The browser is limited in the level of zoom. At the closest level, the browser shows approximately 20kb in a window of size 800 times 600 pixels. At the farthest level, this number is approximately 180kb. However at this level, the horizontal bar illustrating the DNA sequence which has marks at certain intervals, is barely visible, and the marks are far from legible. What is more, the boxes illustrating genes are uncommunicative, diffuse blobs.

The browser does however offer more detail than the display is able to show, but only for the features themselves. A tree structure represents the fact that a gene consists of one or more transcripts, which in turn consist of one or more exons. Clicking either of these opens a list of details regarding it in the right-hand part of the view. For genes, transcripts and exons, the following details are all shown:

- ◆ Name (identity)
- ◆ Chromosome
- ◆ Start Position
- ◆ End Position
- ◆ Strand

For genes and transcripts type, status and description are also shown. For exons the sequence of the exon and a list of transcripts where this exon appears is shown. This last item illustrates the many-to-many relationship between exons and transcripts.

Above the list of details, there are links to Ensembl Gene view, Gene Cards

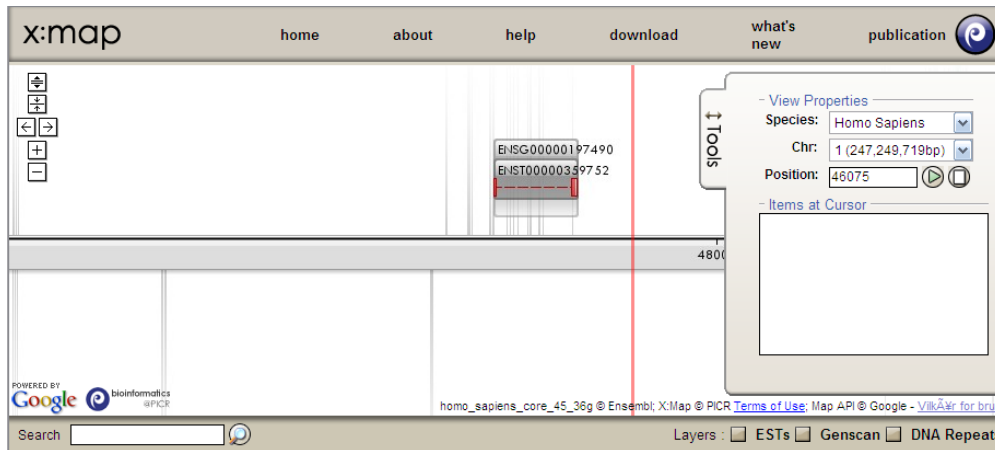


Figure 3.5: X:Map Starting view

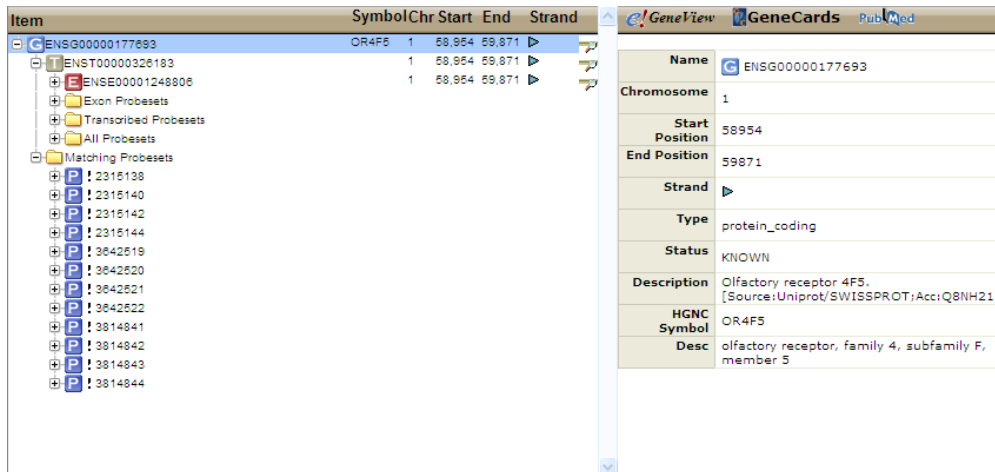


Figure 3.6: X:Map Details view

and PubMed, the last of which performs a search with the gene's name as established by HGNC (Human Genome Project (HUGO) Nomenclature Committee). The first two links opens web pages with details on this gene as provided by Ensembl and Gene Cards respectively.

3.7 Summary and Comments

In this section I will sum up the chapter by recounting which aspects of the described viewers I will move forward with.

3.7.1 Functionality

Of the functionality described in the study of Ensembl, the following will be pursued and elaborated on:

- ◆ Select organism : The possibility of browsing the genomes of different organism.
- ◆ Overview of chromosomes : Presentation of chromosomes of the organism.
- ◆ Chromosome meta-information : Gene density and possibly SNP density will be informative to the target group.
- ◆ Contig view : A view of the clicked area, showing attributes in the region.
- ◆ Base pair view : The possibility of seeing the underlying sequence of bases.

Several of these elements can be fused into one technical solution. In Ensembl, three windows is needed to show an overview, attributes and base pairs. In a simpler genome viewer, this three window solution should be scrapped. A solution where the latter two are fused together will present less of a challenge to users when learning to use the application. The highly restricted level of zooming allowed in X:Map should be avoided.

3.7.2 Interaction

A simpler genome viewer should be more acutely interactive, in the sense that the viewer should respond immediately to the user moving the view. This may allow users to more easily maintain oversight whilst navigating. The interaction style encountered in X:Map comes a long way in fulfilling this wish. The ability in X:Map to click and drag gives a higher sense of interacting with the genome, and should give a more intuitive experience compared to that of Ensembl and Entrez. Buttons for quickly moving around is suitable,

however too many buttons as in Ensembl's ContigView, will undoubtedly clutter the interface and disturb the user whose attention is focused on the data presented. A simple interface more like that of X:Map has a much lower threshold for learning to use, and will be more suited in a genome browser aiming at simplicity.

3.7.3 Level of Detail

In the first two genome browsers examined in this chapter, users can zoom in all the way from an overview of the chromosomes, to looking at the sequence of bases. I intend to match this level of detail. However, as previously stated, I intend to accomplish this in a different way than Ensembl and NCBI does.

3.7.4 Types of Features

The array of different types of features visualised in Ensembl Genome Browser will be meaningless to the majority of the target group. Distinguishing between the different sources of the annotation data is therefore not necessary. A single set of confirmed protein coding genes will be of greatest use in this case.

In X:Map, genes with all transcripts, and microarray clone targets, are the only two tracks visible by default. Microarray clones makes limited sense to put in a simpler genome viewer, mainly because the concept is unknown to the target group. Recall that ESTs, Genscan genes and DNA repeats can be switched on by checking their checkboxes. Of these, only DNA repeats is suited for inclusion in the genome browser. Genscan genes information is redundant when in addition to known genes. ESTs are significantly beyond the theoretical basis of the target group, and would therefore be disturbing to the simple picture they need.

As mentioned, DNA repeats might be interesting to include. However to simplify the picture, in the visualisation they could be shown as "genetic variation". Other features could be displayed in the same category, namely SNPs, LINEs and SINEs. LINEs and SINEs (respectively Short- and Long Interspersed Nucleotide Elements) are interesting especially in that they are used for genetic fingerprinting.

Chapter 4

Analysis of Requirements and Domain

We've now looked at a number of existing browsers, with respective qualities in different areas of gene browsing. It is now time to look at what qualities the new educational browser should have, and what challenges are offered by the domain. The requirements presented in the initial part of this chapter are conclusions based partly on the curriculum of the advanced biology course in Norwegian upper secondary education [UDIR, 2006], and partly on correspondence with a teacher in this course.

4.1 General Requirements

The genome browser should reflect the theoretical basis of the target group, at the same time adding to it. This means that some simplification should be made in certain aspects. The genome browser could still move beyond the theoretical basis of the target group to add to its understanding of certain topics.

Understanding of the DNA Structure in General

The genome browser should create a higher understanding of the DNA structure in general. There is a gap in understanding between the DNA at its most basic level, a sequence of four different bases, to a higher level, separate DNA-molecules known as chromosomes containing genes. The belief is that presenting chromosomes and how shorter and longer genes are scattered throughout, and allowing zooming in and out, will bridge this gap. Getting a visual impression of the amount of "junk" DNA versus expressed DNA, is also helpful in obtaining this kind of understanding.

Understanding of Genetic Variation

The browser should give the target group a higher understanding of genetic variation. Through topics in the curriculum, they are familiar with some sources of genetic variation, and their impact on biological diversity. The existence of alternative splice variants and SNPs presented as sources of genetic variation will broaden their understanding of the topic. Other sources of genetic variation, although perhaps just as important, are disregarded in order to simplify .

Understanding Genetic Diseases

The users will have the ability to search for genes that are associated with diseases. Looking at one such gene in the browser, the user may see that there are sources of genetic variation within this gene. This illustrates the association between genetic variation and phenotype variation, and may heighten the users understanding of genetic diseases.

Knowledge of some Biological Databases

Through use of the genome browser, one will become familiar with the existence of biological databases, and come into contact with information contained therein. An example of this could be information from OMIM regarding a specific gene, which will serve as a real-life example of a topic from the curriculum.

4.2 Functionality

Present Genomes

Having the possibility to inspect a gene in different forms across different species, can be an effective learning instrument. Studying insulin genes in a selection of species will give valuable insight into evolution and phylogeny. Ideally, the browser should provide possibility for browsing all available genomes. Looking at the same gene in different species, will give a deeper understanding of evolutionary processes. Observing differences in size and complexity between a specific gene in mouse and its homologue in other species, is undoubtedly more powerful than a textbook chapter explaining the same concepts.

Presents Chromosomes

Presenting the chromosomes allows comparison of chromosome lengths and gene counts. In addition the density of SNPs and genes across each chromosome can be shown using a histogram. Such a presentation of the chromosomes

illustrates the heterogeneity of genomes. One can take advantage of this in designing student exercises.

Browse Features

By visualising attributes as boxes of different size and colour on either of two strands, the following important information is encoded: position, length and type of attribute. This type of representation appears to be the standard, judging from what is seen in chapter 3. The two strands represent the unwound double helix of DNA. This is perhaps the most critical requirement of any genome browser.

Explain Intermediate Structures

An educational browser should offer a presentation of how the DNA double helix is packaged into chromosomes. This is abstracted away in virtually all existing genome browsers.

Provide Information in Norwegian

Textual information as a supplement to the visual presentation should be in the target group's native language, Norwegian. The information presented should hold a technical level suitable for the target group.

4.3 Non-functional Requirements

4.3.1 Software Framework

The genome browser should ideally be implemented as a relatively generic framework for displaying genomic data and linking external sources of information to that data.

Provide an API

By providing a simple API, user programmed objects can access the genomic information, the browser engine etc. and create specialised presentations, data dumps or other output. A teacher might write an object that jumps through the genome, stopping to present genes with a high degree of genetic variation (SNP content above some limit).

Ease of Extensibility

The browser should be designed with the possibility of adding new annotation tracks in mind. This is a feature of many existing browsers, and will allow the browser to be extended , so that more professional uses are possible.

4.4 Domain Model

The human genome, with its roughly 3 billion base pairs and therein 20- to 25 thousand genes, presents a challenge in terms of storage and object oriented representation. Moreover, while a DNA sequence might be structurally simple enough (alternating A,C,T and G), the function of some parts of a sequence, stored as sequence annotations, are not. The domain model evolving in this section is not intended to cover our full understanding of DNA, nor is it presented as the definitive model of the problem area. It is a pragmatic attempt to cover the concepts of the educational goals of the new genome browser.

4.4.1 First Draft

This is the first outline of the domain of DNA relevant to this project. Figure 4.1 represents a first draft of the domain model for the concepts of eukaryotic genomes. A short presentation of the different classes appearing in the diagram follows:

Species

Species represents a specific species, the Genome of which we are interested in browsing. Technically there are occasions where a species may have more than one Genome, there may be an older version or a different build. However such details are disregarded in this project.

Genome

A Genome represents the known genetic information regarding a species. A genome can be presented as a sequence, in which case it is not the genetic recipe for a particular individual, but rather a summary of all the individuals who have been studied to uncover the information present in that genome.

Chromosome

The molecules into which eukaryotic DNA is organised. Different species have anywhere from a handful to several hundred chromosomes [Khandelwal, 1990].

SNP

Represents an established variation in the genetic code of this species in one specific base pair. SNPs are highly associated with genotypic variation. They are used in genetic fingerprinting.

Gene

Represents a sequence of bases that is expressed through transcription of the sequence. Genes are also regarded as units of inheritance.

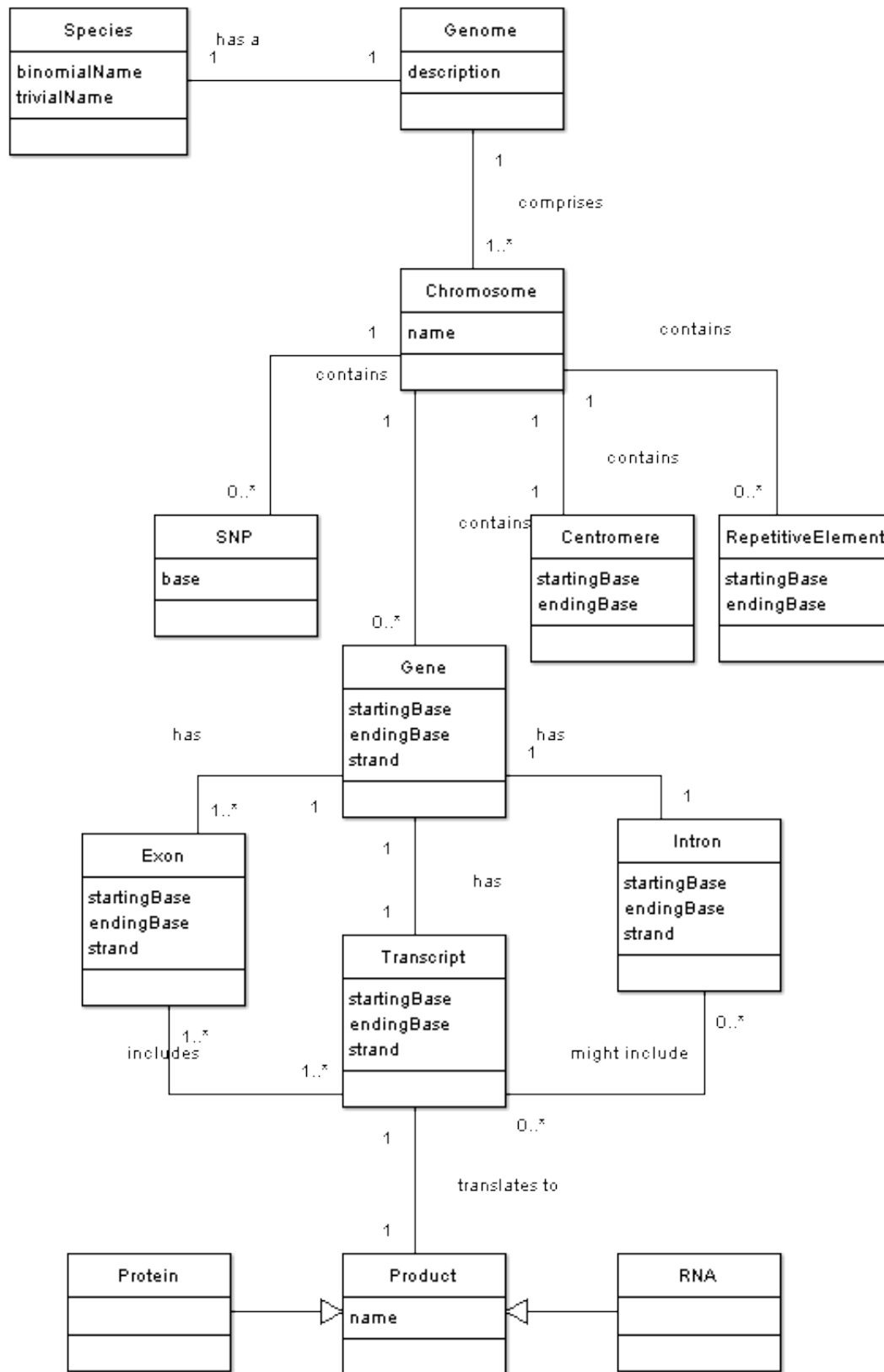


Figure 4.1: Domain Model, 1st draft

Repetitive Element, Centromere

Other features on the chromosome.

Transcript

This represents a mature transcript, mRNA, ready to be translated. As explained in section 2.1, a transcript always includes one or more exons, and occasionally includes an intron. This is reflected in Transcript's associations with Exon and Intron respectively.

Product

The results of expression of a gene are known collectively as gene products or just products.

Protein

A type of product.

RNA

A type of product.

Exon

A class representing coding regions within genes.

Intron

A class representing non-coding regions within genes.

Comments

There is a one to one relationship between Species and Genome. In addition we are only interested in the Genome of a species. This makes inclusion of a class Species in the domain model redundant. Letting *binomialName* and *trivialName* describe a Genome instead, I can safely exclude Species without losing anything of significance.

The features Repetitive Element and Centromere lie outside the educational goals of this genome browser, and will therefore be overlooked.

4.4.2 Discussion

The concepts captured in the above section are simplified, yet cover more than the demands of the curriculum of advanced biology in Norwegian upper secondary education.

4.4.3 Domain Model, Rev. 1

In this revision of the domain model, following the discussion in the previous chapter, Species has been cut. Furthermore, The decision to represent protein coding genes exclusively, obsoletes the parts of the domain model pertaining

to product types. The revised domain model hence takes the form seen in figure 4.2

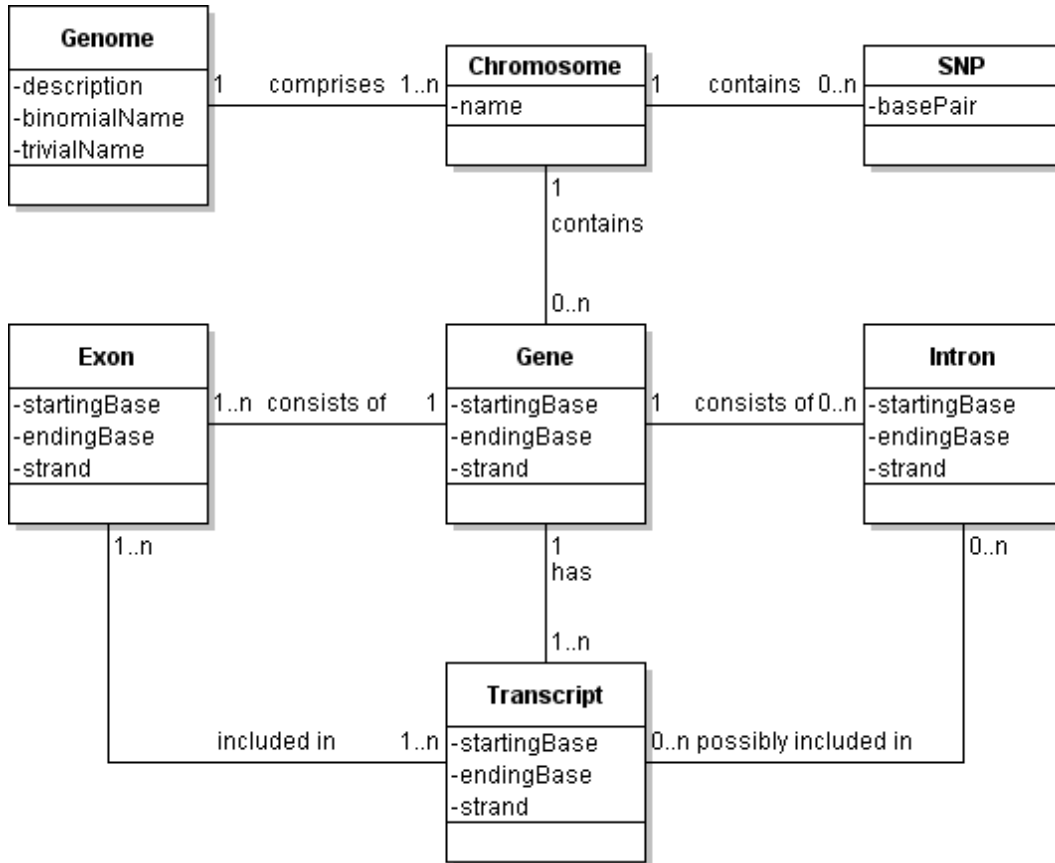


Figure 4.2: Domain Model, Revision 1.

4.5 Arriving at a Class Model

This sections presents the class model finally used. It does not presume to have found the most elegant class model, nor the most suitable. The class model finally used merely passed the criteria of being simple, extensible and easy to work with, and was thus used.

4.5.1 Introduction

The class model that was ultimately used, bore little comparison to the domain model. A simpler more general model was used, adapted from the domain model. The classes Gene, Exon, Intron and Transcript were removed in

favour of two generic classes: Feature and Structure. They are enumerated types, an integer signifies whether a Feature is a Gene or a different Feature. A Structure can for instance be an Exon.

The concepts Intron and Transcript were dropped. The existence of multiple transcripts for a gene was thought to unnecessarily complicate the picture given to users. The important theory was thought to be better presented using literature.

The result is a class model that is more generic while still being able to represent the necessary data. A more generic class model is favourable when potentially using multiple data sources with data structured differently. What is more, the model gives the possibility of adding other types of attributes at a later point in time. This class model is also conceptually easier to work with. Yet, this model is not as descriptive as the domain model, it will do little to explain the domain it represents. Instead data must be adapted to the class model. There is a neglectable danger that this will impose unintended restrictions.

Feature

Features is a loose classification of longer genomic attributes. Genes are part of this category. The category could also potentially include LINES and CNVs.

Structure

Structures is the counterpart of Features, a loose category of shorter genomic attributes. Exons, SNPs, and potentially SINES are assigned to this category.

4.5.2 Diagrams

The diagrams in this section presents the final class model used for the domain classes. The first diagram, figure 4.3, presented represents the class hierarchy involving the classes *Feature* and *Structure*, mentioned in the preceding section.

The second diagram, figure 4.4, presents the comprising classes *Chromosome* and *Genome*. The interfaces of all classes, i.e. the methods available are specified separately throughout the domain package. This is a deliberate attempt to keep the implementation of the classes separate from the specification of them. All references in class implementations should then go to interfaces of other classes.

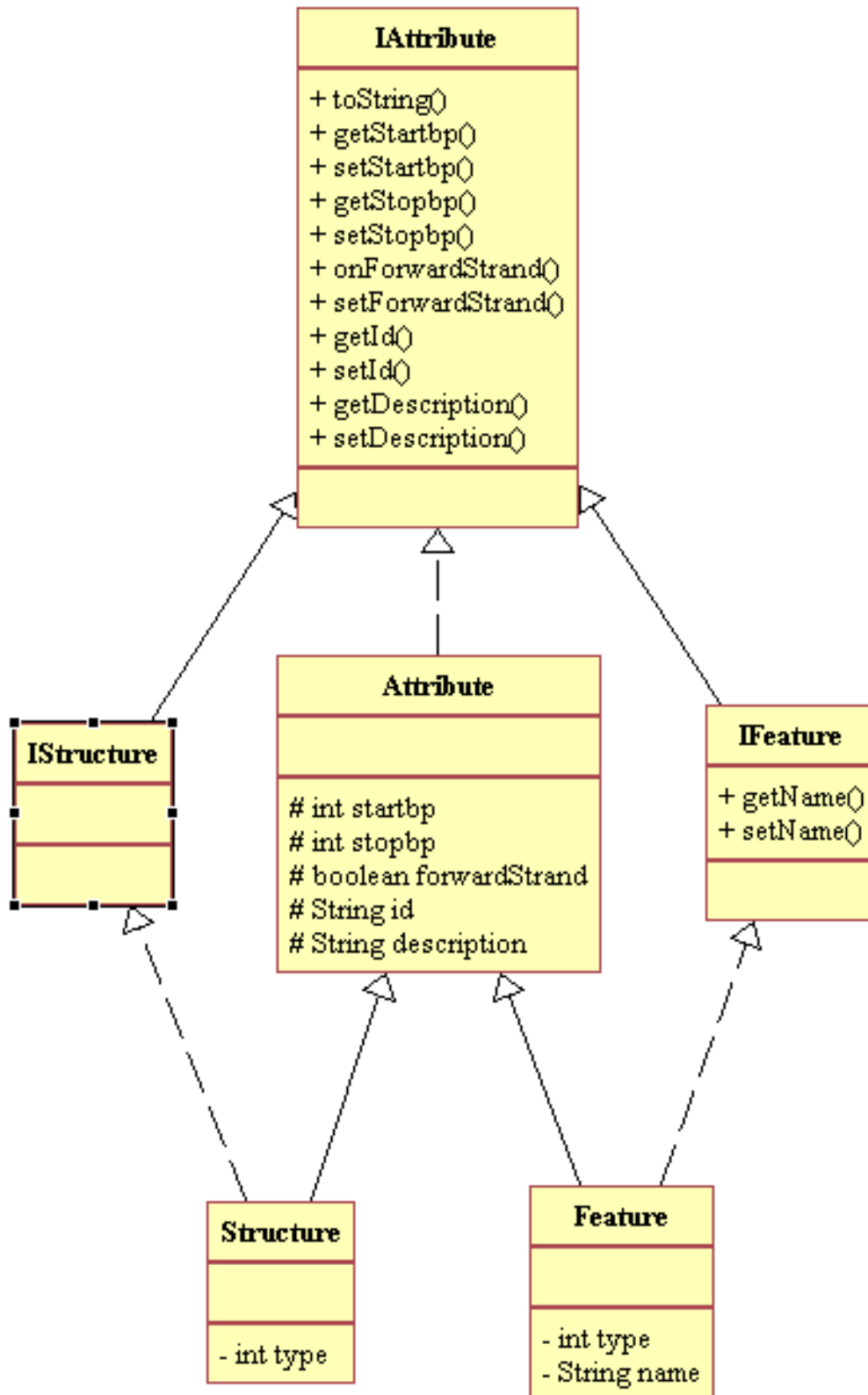


Figure 4.3: Class Diagram: Attribute Hierarchy.

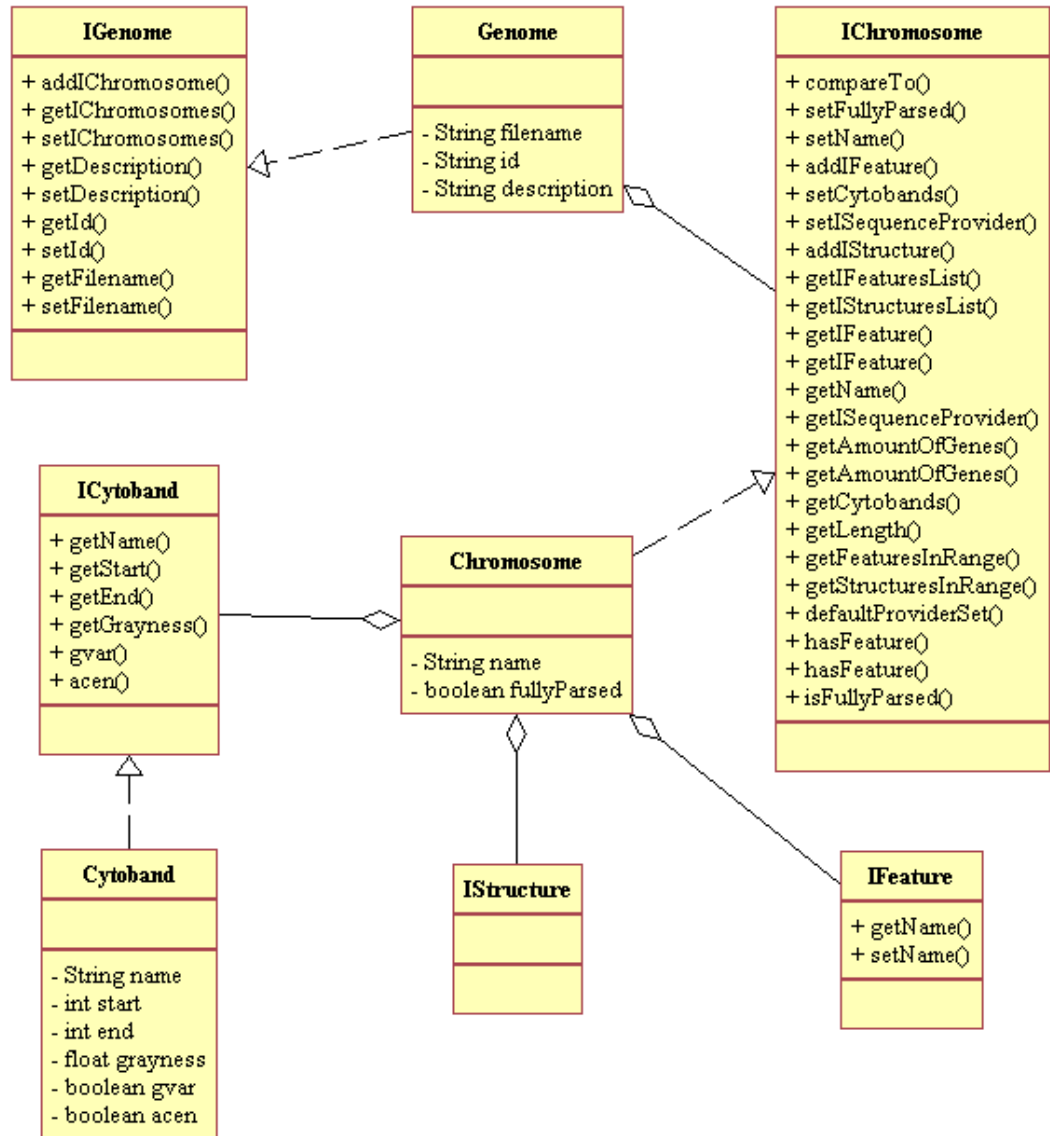


Figure 4.4: Class Diagram: Comprising Classes Genome and Chromosome, with respective interfaces.

Chapter 5

Designing an Educational Browser

With a set of requirements of an educational browser in place and having performed a pragmatic analysis of the data domain, we now move to deciding the actual functionality of the new educational browser. I start by discussing functionality to include, and then discuss the appearance of the functionality included.

5.1 Basic functionality

This section describes the desired functionality of the new simpler genome viewer. It was decided that a minimal version should first be completed, before adding advanced functionality. The rationale behind this is to prevent the finished product from being a set of half-finished features. Following this rationale, the functionalities are divided into two groups, primary and secondary priority.

5.1.1 Select a Genome

Selecting among different genomes might not be a crucial element of an educational browser. The human genome is the most critical and interesting, given the browser's intention. Another rationale behind focusing on the single genome, is that it allows more work to be put in to the remaining functionality. This aspect will set the browser apart from many existing browsers, which allow browsing the genomes of a range of organisms. However this is a functionality which will not be missed by the target group. There is adequate complexity and interesting learning opportunities in the human genome alone. Therefore, implementing the possibility to browser different genomes, receives low priority.

5.1.2 Browse Chromosomes

Seeing the chromosomes distinctly is the first conceptually concrete middle step on the way from a whole-genome view, down to a two-strand view. The DNA of (a eukaryotic) organism is divided into chromosomes, something which can be seen with the naked eye in a certain stage of cell division. Listing these separate bodies of DNA material provides a good abstraction on top of the DNA itself, besides reporting the number and sizes of the various chromosomes. The browser will therefore present the human genome firstly as a list of chromosomes. Each chromosome is presented visually along with a set of details pertaining to each chromosome. The set of details of each chromosome include: Chromosome name, length, cytobands, and number of known genes. A short comment on each detail follows:

Name

Distinguishing between chromosomes is done with the names of the chromosomes. The name of a human chromosome could be '10', 'X' or 'MT'. Chromosome names are universal, a piece of literature may mention chromosome 21, e.g. in connection with a disease, the same chromosome will be known as 21 in all other literature and databases.

Length

In presenting chromosomes it is important to illustrate their lengths compared to each other. The length is denoted in basepairs.

Cytobands

Looking at chromosomes under a microscope, it is necessary to add a dye in order for the chromosome to become visible. This staining results in the patterns of stripes normally seen in graphical representations of chromosomes. The chromosomes will be represented with this information.

Known Genes

Illustrating the complexity of the greater picture, it is interesting to note the content of known, protein-coding genes in each chromosome. One will notice that this number is not strictly dependent on chromosome length.

5.1.3 Explore Chromosome

Exploring a chromosome, browsing the interesting contents of a chromosome, is perhaps the primary functionality of genome browsers. Though it varies from browser to browser what features are interesting, all provide this functionality without known exceptions. In this project also, exploring chromosomes is necessary in some form in order to fulfill the requirement of conveying an understanding of the general structure of DNA.

All of the genome browsers investigated in chapter 3 implement this functionality using the abstraction of locating features in either of two strands. This abstraction appears to be a baseline feature in most genome browsers. A genome browser presenting the contents of a chromosome without regard to strands will neglect an important aspect of DNA structure. Imparting this knowledge to the target group being a central requirement, the new browser will provide this functionality in the following way: Having selected a chromosome to 'zoom in' on, users will see the distribution of features across the chromosome, on either of two strands.

Features Visible

What types of features being visualised, varied considerably in the browsers in chapter 3. Ensembl genome browser is representative for a range of browsers in that it shows features as 'tracks' or layers on top of the representation of the physical genome. More types can be added by switching on other tracks. Likewise, the view can be simplified by switching off tracks. In this project, avoiding the implicit demand that users declare which types of features to see will contribute to lowering the learning threshold for using the browser. Providing a small, fixed set of all-important features to visualise is an admissible simplification.

While exploring a chromosome, users will have the possibility to zoom in on features. The types of features that visualised by the application include: Genes, exons, SNPs (as genetic variation), nucleotide sequence. A discussion on each element follows:

Genes

Genes often constitute the main feature of genome browsers, and other features included are often linked to them. For example in the browser X:Map, gene chip markers are shown on top of genes, simplifying the design of an experiment for measuring a certain gene. Looking at genes themselves is often a motivation for using genome browsers. To the target group this certainly is true. The simplification of only including known, protein coding genes is justified because of the theoretical background of the target group.

Exons

The shorter sequences within genes that are translated into a product during expression of a gene. Showing exons as constituents of genes contributes to the targets group understanding of the DNA structure in general, and is an important detail in gene expression. Alternate splice variants are an important aspect in understanding genetic varia-

tion. Presenting exons is vital in conveying the full picture of one gene, multiple transcripts, using a subset of exons.

SNPs

A SNP is a type of mutation and polymorphism that is quite easy to understand. Their association with genetic variation is highly researched over the course of what in the field is a considerable amount of time. Many diseases and other phenotypes well known to 'normal' people have been shown to be associated with SNPs. SNP information is therefore a valuable inclusion in the browser. The browser will use SNPs to illustrate genetic variation. To illustrate the informational value of the distribution of SNPs, consider a map of a city at a police station. The concentrations of pins in different areas expose the bad neighbourhoods. A high occurrence of SNPs within or close to a gene indicates a gene that is highly associated with phenotype variability.

Nucleotide Sequence

The visualisation of underlying sequences is possible in the Ensembl Genome Browser. In NCBI Map Viewer, sequences are available for download or as a link to NCBI's sequence viewer, a separate web application. In this project the inclusion of underlying sequences is intended to better the users' understanding of the general structure of DNA. The helpfulness of providing sequences as raw data on the side will then be limited. Zooming in close enough, the user will eventually see the underlying sequence of bases. In this way one can for example "aim" at the starting point of a gene and zoom in to see the special starting sequence of that gene. This opportunity will hopefully contribute to bridging the gap in understanding between DNA at its most basic level and what the concept of a gene represents in other contexts.

Comments

As a consequence of the above discussions, the regions within genes not coding for proteins will not be visualised. These areas are known as introns (from intra-genic regions), and they are commonly excluded in existing genome browser, presumably to clean up the view. Introns do appear in a more detailed picture, however they can be abstracted away when the focus is on a general understanding.

5.1.4 Level of Detail

In some browsers there are no limitations on how much the strands can be magnified. NCBI Map Viewer is one such browser. In other browsers such as the Ensembl Genome Browser, zooming in can only be done to a certain limit. Such a limit should only be set at a point where zooming in further provides

no new understanding. In the Ensembl browser, zooming in beyond viewing one base in the entire frame is impossible. In a browser where the nucleotide sequence is available, this is a reasonable limit. No new understanding can be gained from zooming in on parts of bases.

The X:Map browser employs a more stringent policy on zooming. At the closest level, the in a standard-sized (800 pixels wide) browser window, about 20000 bases are visible. This doesn't allow much detail to be presented graphically. The browser presents other details using lists and tables instead. The new browser borrows many concepts of simplicity from the X:Map browser, such as the style of interaction. However, it should allow for the same level of zoom possible in the Ensembl browser, since both can present nucleotide sequences.

Another aspect is the hiding of certain feature types above or below certain levels of zoom. When exploring a chromosome in the new simpler genome browser, the initial view is of a relatively long stretch of DNA. At this level only genes are visible. The user will have the ability of zooming in and out. As the user zooms in, at a fitting level, the constituents of genes, exons, appear. A fitting level is one that ensures that the number of objects (genes, exons) visible at any time does not overload the user. With exons appearing when only a handful genes are visible, the risk of losing overview is

5.1.5 Gene Information

Providing gene information is done to various extents in existing browsers. The users of a simpler genome browser, must be provided some information as assistance in understanding what they see. Finding an optimum balance between too much and too little information will make the browser a more effective learning tool. Having found an interesting gene, the information that a user can obtain on it is listed here:

- ◆ Name or ID
- ◆ Length
- ◆ Description
- ◆ Gene ontology keywords

Discussion

The result is a very limited set of features that will be visualised. In fact, many features that are interesting elements in a genome have been left out. Genes that have been deactivated through the course of evolution, pseudo genes, are thought to play a role in regulating gene expression, and can be

switched on through mutation. Pseudo genes are but one example of other interesting features.

5.2 Design Choices

In this section, I present design choices and discussions on these, regarding visualisation in the browser.

5.2.1 Overview of Chromosomes

In some browsers, a graphical overview of a single or all chromosome is not available. A checkbox is used to select between them. The new browser should present chromosomes more intuitively, since it focuses on transmitting understanding. A chromosome with cytobands in correct colour, narrowed at the centromere and rounded ends, is a common representation in current genome browsers and in relevant course literature. By presenting an overview of chromosomes in this way, the user sees something recognisable, and the learning threshold in understanding the overview is lower.

Having selected a chromosome, most browsers change immediately to a presentation involving the two strands of the DNA molecule. By doing so they ignore the intermediate structures. Completely ignoring this, important structural details are obscured from the target group. The breakdown from chromosome to two-stranded DNA is a series of steps involving different scaffolding proteins and concepts which a traditional genome browser will be inept at explaining. The new, simpler browser can offer a video sequence or a set of images explaining this, along with links to relevant literature.

Also, when going from the chromosome overview to the two-stranded view, users might become lost. To prevent the user from losing track of the context, having a miniature map resembling the chromosome as it appeared in the overview will be helpful. The miniature map can also serve as a reference, showing where in the larger picture the view is currently concentrated.

Cytobands

Tagging the cytobands with their respective names might be defocusing to the users, therefore only the coloration will be added. There is a danger that more savant users will miss these important references, however, this insight lies outside that of the target group.

5.2.2 Exploring Chromosomes

As previously mentioned, in almost all cases the contents of a chromosome is visualised as objects in either of two parallel strands going across the screen. This the conceptually closest, easily implementable representation of the double strand. This browser will make use of this representation as well. Where some browser splits the screen in two and having one half represent the forward strand and the other- the reverse strand, this browser will use two black lines to represent the strands. Two lines is thought to lie closer to what the users expect: the two strands of DNA.

An important design choice in this genome browser is the application of only one view or window for presenting the DNA. This is in contrast to the Ensembl browser discussed in chapter 3, where as many as 5 views could be active at certain times showing different levels of detail. Using only one window will potentially prove challenging, however, many windows imposes tougher challenges on the users, interpreting what they see.

5.2.3 Genes

Rectangles are a commonly used representation of features. Rectangles of a certain colour corresponds to genes. By going for this representation, users known to genome browsers will be right at home. Also, this representation is used throughout the genre, and choosing a different representation dissociates this browser from existing browsers in an unfavourable way. Therefore, genes will be visualised as rectangles on either of two strands, indicating which strand of DNA it is located on. Gene boxes will be green, setting them apart from other features visualised as boxes of a different colour. Genes are the largest features of a chromosome visualised by this browser. Therefore genes are the first features that are visible. Smaller features, e.g. exons, are not shown until the user zooms in.

5.2.4 Exons

Following the rationale for using rectangles as representation of genes (and features in general), Exons will be represented as rectangles as well. When the user zooms in close enough, exons will appear. Like in NCBI and Ensembl, looked at in chapter 3, they will not be visible until zoomed close enough. This prevents the view from being swamped with too many objects. During experiments, I have found that when there are only a handful genes visible, it is suitable to let exons appear. This occurs when the relationship between pixels and bases is about 10^{-3} , meaning 10^3 bases are represented by each pixel. Exons will in the same way as genes, be visualised as boxes inside genes. Exon boxes will be blue.

5.2.5 SNPs

Using data from the Ensembl browser, SNP count for the shortest human chromosome, number 21, is 178'438, while the total length of the chromosome is 46944323 bases. For comparison, chromosome 21 has 284 known, protein coding genes. So, a SNP occurs on average every 263. base. All this means that at the same level of zoom where exons become visible, 3802 SNPs would be visible. This must be considered when deciding how to present the presence of SNPs visually. At this point, representing individual SNPs along the strands will only become effective on a much closer level. At all other levels of zoom, a diagram indicating the density of SNPs will likely be more intuitive. In the Ensembl browser, a histogram in the form of a bar chart is used to show the density of SNPs in different parts of the chromosome. A bar chart or alternatively a curve chart are two options.

5.2.6 Sketches

The following section presents sketches of the browser illustrating the design choices sketched out in the parent section. They are based in part on an early prototype and in part on sketches.

Sketch 1

The first sketch shows a portion of a chromosome, with two horizontal lines symbolising the DNA strands, and features of different type symbolised by coloured rectangles. The sketch also features a bar chart histogram for indicating SNP density, similar to the what is used in the Ensembl browser. The use of a bar chart in this setting is counterintuitive because the shapes in a bar chart coincidentally are somewhat similar to the boxes representing genes and exons.

Sketch 2

In this sketch (figure 5.2), a curve chart represents SNP density. This representation is more subtle than the one presented in the first sketch, and will perhaps for that reason, be more intuitive.

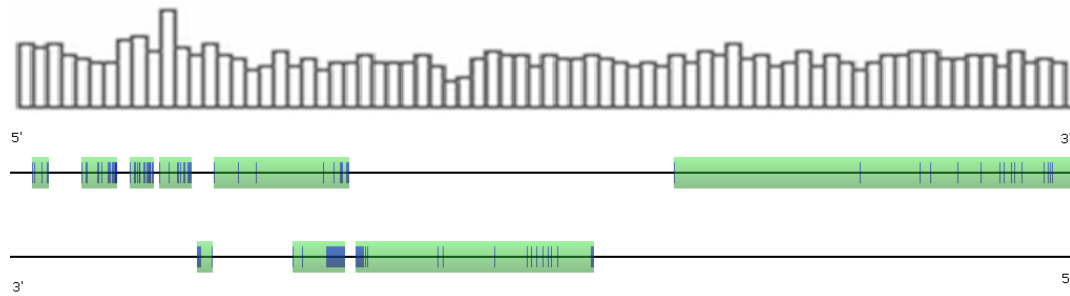


Figure 5.1: User Interface Sketch 1. *Top*: A bar chart histogram of SNP density. *Middle*: Forward strand with some genes and exons. *Bottom*: Reverse strand.

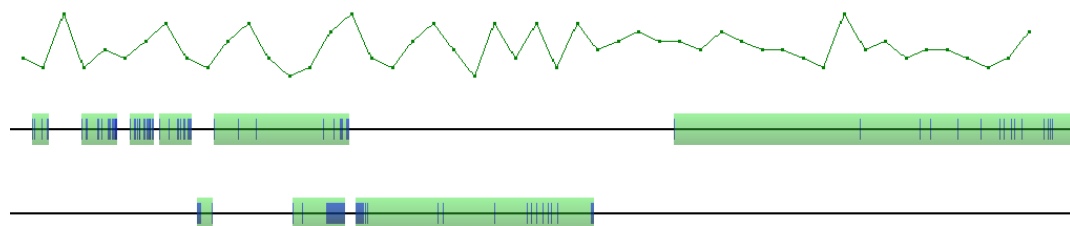


Figure 5.2: User Interface Sketch 2. *Second from the top*: A curve chart histogram of SNP density.

Chapter 6

Construction

This chapter presents aspects and discussions regarding the construction of the simpler genome viewer. Put shortly, the most technically interesting areas of the genome browser are the obtaining and handling of a large amount of data, the display of a high number of graphical objects on the screen. These aspects will be detailed further in this chapter.

6.1 Deciding on a Programming Language

This section discusses the choice of a language for implementing the new genome browser. While Java, for a number of reasons, was selected early on, a discussion of natural options is still in order. Two languages were taken into consideration:

6.1.1 C++

For an application with emphasis on graphics, it is natural to consider C++. Its creator Bjarne Stroustrup set out to improve C with features Simula that he had found useful for large scale software engineering. C had been designed with rapid execution in mind, and was a general purpose language. Traditionally one has had the idea that only a language that is sufficiently low level will perform well enough. In C++ one has direct access to system libraries, such as OpenGL which will be introduced later.

Virtually a dogma of programming languages is the fact that C++ is not suited as a first language for learning programming. An argument often raised is the type safety of pointers, which can be broken. Experienced programmers might use this effectively, however novice programmers experience this feature unknowingly. A common criticism of C++ is that it supports different programming paradigms and has a vast number of functional features, both high-level and low-level, supported in the language, this allows programmers

to solve a problem in many different ways. This is reflected upon the learning curve which reportedly is relatively steep. Mastery of the language is said to take several years.

The same multiplicity of possible approaches that for novice programmers presents a significant learning barrier, is a mark of quality to more experienced programmers. It allows rapid implementation of code for a certain problem, which may have a low degree of readability to less experienced programmers.

The standard library of C++ is a collection of classes and functions. It contains data structures, input- and output streams, common mathematical functions, and other useful functions. It also contains the C standard library. In addition to the standard library, there is an abundance of third party libraries readily available. There are no language features supporting creation of multi-threaded software. Third party libraries or system calls must be used, potentially creating portability issues.

There still exists a performance gap between C/C++ and Java, however it is probably less significant than most programmers realise. Optimised versions of a Java port of Quake 2, a 3D first person shooter initially implemented in C, have achieved 85 - 100% of the performance of the original version [Stöckel, 2006]. At a programming language comparison central (Computer Language Benchmark Game), C++ outperforms Java by 10- to 400 percent in a range of programs (algorithms), in terms of processing time.

The lack of portability of C++ is often pointed out as a weakness. C++ code is compiled to binary machine code executable on a particular platform. Hence, a program must be compiled to two different targets by a compiler in order for it to run on two different platforms, e.g. windows and a Linux distribution. A further aspect is the lack of standards compliance among the available c++ compilers. Even after the first ratification of the language standard in 1998, the implementation of certain features are left up to compiler creators to decide. A consequence of this is that object code produced by different compilers is incompatible. In spite of this, writing portable C++ code is possible, by following specific guidelines available from a variety of sources.

6.1.2 Python

Python is another option. It was created by Guido Van Rossum with emphasis on readability, simplicity and modularity. It is normally applied to application development, scripting, and education. Python is an interpreted language with major implementations of the interpreter implemented in C, Java, and .NET.

Python aims at being the number one language for learning computer programming. To achieve this, its syntax was designed to be highly readable. English text is used instead of punctuation, and program blocks are denoted by fixed-width indentations. Furthermore, Python supports language features found in functional-, object oriented-, and imperative programming. This ensures that a particular style of programming is not forced upon its users. Lastly, Python's creators argue that dynamic typing is favourable when learning computer programming. While being able to use a statically typed language is important in the long run, overlooking this complexity allows advances in more important areas early on.

Python sports a large standard library, often mentioned among its greatest strengths. It includes support for a selection of formats and protocols, making Python excel at internet oriented usage. The standard library, in addition to facilities typically offered by other standard libraries, includes a unit testing framework.

A feature of Python's design is extensibility. Additions to the standard library can be realised in either Python or C. In fact, any library in pre-compiled machine code is usable from Python. Python bindings for OpenGL are available. In addition, higher level libraries for developing applications embedding 3D are numerous. Several of these label themselves 'game-engines', reflecting their features and intended use.

An objection often raised, is the performance of Python. Being interpreted, Python will perform considerably poorer than compiled languages like C++ and half-way compiled languages like Java. At the Programming Language Benchmark Game, C++ is up to 277 times faster than Python (processing time). In Memory usage, though Python suffers less, and even outperforms Java. Poor performance is often traced to loops doing manipulation of numerical data. In such situations, the use of third party, pre-compiled data structures will remove much overhead.

6.1.3 Java

A structured, imperative, object oriented language, Java was developed by Sun Microsystems and released in 1995. Its syntax is largely based on C and C++, but its object model is somewhat simpler. Still, the object model of both languages have many similarities, being reimplementations of the object model of Simula, arguably the first object oriented programming language.

The main ideologies of the language were platform independence, network programming support and selecting the object oriented programming paradigm.

Platform independence is achieved through compiling Java programs half way, to Java byte code. This byte code is then run on a virtual machine, in a sense a program that understands Java byte code. This is a continuance of an idea employed by e.g. Pascal. With version 1.3, Java received a new virtual machine, Java HotSpot Virtual Machine, offering just-in-time compilation. This is a technique that reduces run-time by compiling heavily used parts of the byte code into native-language code.

Java became increasingly popular due to several factors. Among these is the fact that quite early, major web browsers started getting support for running Java Applets. This allowed rapid publishing of relatively advanced applications online. The supporting library grew steadily in size from the initial release, notable APIs added early on are Swing and Java2D, which is revisited later. It saw it's biggest expansion with the release of version 1.4 in 2002. Among the new features were: A facility for reading and writing images in different file formats (Image I/O), regular expressions, and an XML¹ parser. Later versions, 1.5 and 1.6 have seen even further extensions of the supporting library and the language itself.

Another source of Java's popularity is the fact that it is developed, starting with version 1.4, as a community process (known formally as Java Community Process or JCP). Proposals for new features take the form of formal documents called Java Specification Requests, which are voted on by the JCP Executive Committee. By mid 2007, Sun had released Java as open source software, spawning a myriad of projects adapting and developing new implementations of Java. A large community of developers means that help and documentation is abundant.

6.1.4 Conclusion

Comparing C++, a multi-paradigm language to a language such as Java which employs a narrower set of paradigms is unfair, since they are of different merit. The conclusion is therefore not that C++ is a lesser programming language, it is simply less suitable for use in this project, by a novice C++ programmer.

Python might have been better suited as a language to learn from scratch than C++. However, there are no weighty technical advantages of using Python. A personal suspicion that Python is less suited as a language for developing large applications further excludes Python as an option.

Choosing a programming language should ideally be done in the same way as a carpenter chooses a tool for a task. The best tool for the job is the only

¹XML - Extensible Markup Language. A standard language for marking up information, making it parsable in an implementation-independent way. Further information available in chapter B

criteria. Ultimately, ones own personal preference is a factor as well.

6.2 Graphical Framework

This section will detail the selection of a platform for the genome browser's graphical component. Java has built in two-dimensional graphics capability, Java 2D. There are also alternatives to Java's built-in graphics API (Application Programming Interface), Jogl is one such alternative. In discussing Java2D and Jogl, I also illustrate differences in complexity in drawing one simple shape to the screen.

6.2.1 Demands

The desired level of interactivity demands the drawing of a high number of graphical objects on the screen. The designed functionality of the genome browser allows the user to arbitrarily choose level of zoom. At one extreme, browser must display all the features on that chromosome, i.e. the entire chromosome is visible. Features that are longer than the length of the chromosome divided by the width of the browsers frame will then be displayed. This, coupled with the need for processing large amounts of data in the background, means special considerations should be taken.

6.2.2 Java 2D

The Java2D API was released with the first version of Java, version 1.0. It has since then been optimised and given new features. Most recently it has been made to interoperate with 3D OpenGL rendering. This allows Java 2D graphics to be mixed with rendering created by the alternative discussed in the next section.

The Java 2D API comprises convenience classes for displaying and manipulating a wide array of geometrical shapes, text and images. In theory, drawing a simple line in Java 2D entails the following steps: Creating a line segment; transforming it according to the current transform; stroking it to create a thin rectangle; telling the shape to compute the pixels being affected, generating the pixels using `java.awt.Color.BLACK`, and then compositing the results onto the screen, or to a different destination. Other destinations could be a printer, a memory location, or an object accepting Java 2D images for conversion into Vector Graphics files, a testament to the versatility of Java 2D.

Compared to OpenGL it provides a higher level approach to drawing.

Drawing a Rectangle in Java 2D

Drawing a rectangle in Java 2D entails the following steps:

- ◆ Override e.g. a JPanel's **paintComponent** method
- ◆ Create an instance of **Rectangle2D.Double**
- ◆ Use the **Graphics** instance (argument to method in step 1) to paint the instance.

An illustrative code snippet is shown below:

```
//Declare square as a class-
//attribute in a subclass of JPanel

private Rectangle2D.Double square =
    new Rectangle2D.Double(10, 10, 350, 350);

public void paintComponent(Graphics g) {
    //Clear the drawing area:
    clear(g);
    //Cast to Graphics2D, has more functionality:
    Graphics2D g2d = (Graphics2D)g;
    //Use graphics-object to draw the square
    g2d.draw(square);
}
```

Comments

The first pair of arguments to the constructor of `Rectangle2D.Double` are `x` and `y` coordinates in the panel, of the upper left corner of the rectangle. The second pair specifies width and height of the rectangle, respectively.

Java 2D was considered as the technical solution for the graphical engine, due to its versatility and simplicity. I conducted a small-scale test to see how it handled large amounts of objects moving on the screen.

6.2.3 Jogl - OpenGL Bindings for Java

OpenGL (Open Graphics Language) is a specification put forth in 1992 by Silicon Graphics Inc. Its intention was to standardise access to hardware, ameliorating the situation at the time where each graphics hardware vendor had custom interfaces and drivers. The advent of OpenGL led to hardware manufacturers supplying drivers which could be communicated with in the way OpenGL specified.

Jogl (Java Binding for the OpenGL API) is a project that provides Java applications access to the APIs in the OpenGL 2.0 specification, through Java Native Interface. It is currently being developed by Sun as a Java Community Process. Access to OpenGL APIs means that the rendering capabilities of the graphics hardware can be utilised from within Java. Using OpenGL standard also lets one take advantage of the documentation readily available for OpenGL.

6.2.4 Other Alternatives

There are other alternatives for hardware-near graphics programming in Java. Java3D is an API developed by Sun first released in a beta in 1997. It provides abstractions on top of either OpenGL or Direct3D, Microsoft's alternative to OpenGL. Java3D provides an object oriented approach to 3D programming, with drawing based upon a data structure called a scene graph. In a scene graph, the various elements to be rendered are represented as objects in a directed acyclic graph. This eases organisation of large scenes. Moving a rider and his horse is done just by moving the horse, the rider is located after the horse in the scene graph and inherits the horse's position.

The lightweight java gaming library (LWJGL) is another higher level alternative. Developed as an open source project, the API has the goal of making a number of multi-platform technologies for game programming available in Java. These technologies include OpenGL, Open Audio Language (OpenAL), interfaces to game controllers and joysticks, and imaging- and music libraries. The philosophy of LWJGL is to expose these technologies, either unavailable or poorly developed in Java, as a thin API Layer. In this respect, LWJGL lies somewhere between JOGL and Java3D.

The added workload of learning a larger and perhaps more complicated API could be defended by the simplicity offered. However there was a suspicion that the drawing to be done in this project was not of a nature that would benefit noticeably from this. A relatively simple way of presenting the various genomic features was opted for, as explained in chapter 5. Moreover, the authors personal interest in gaining experience with OpenGL played in. JOGL is thus the simplest API, providing the closest mapping of OpenGL and no abstraction to drawing.

Drawing a Rectangle in JOGL

Using JOGL, drawing anything requires roughly the following steps:

- ◆ Create a class implementing the interface **GLEventListener**
- ◆ Implement the 4 methods specified, among these **display**

- ◆ In **display**, get an instance of **GL**
- ◆ Make 4 calls to **glVertex** to create the 4 vertices of the rectangle.

An example of the display method is shown below:

```
public void display(GLAutoDrawable arg0) {
    GL gl = arg0.getGL();

    //Set the clearing colour to white
    gl.glClearColor( 1.0f, 1.0f, 1.0f, 1.0f );
    //Clear the necessary buffers
    gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);

    //Let the buffer know that we're starting to draw, as well as what.
    gl.glBegin(GL.GL_QUADS);
    //Set drawing colour to black.
    gl.glColor3f(.0, .0, .0);

    //Draw 4 vertices at the 4 corners of the rectangle.
    gl.glVertex2i(5, 5);
    gl.glVertex2i(10, 5);
    gl.glVertex2i(10, 10);
    gl.glVertex2i(5, 10);

    //We're done, the area within the
    //vertices are filled according to the
    //chosen color
    gl.glEnd();
}
```

6.2.5 Technical Comparison, Java 2D and Jogl

In order to compare Java 2D and Jogl, a simple benchmark application was created. A version using either platform was written. Both applications use the same algorithm. The algorithm proceeds textually as follows:

1. Set rectangle size equal to screen size.
2. Fill screen with as many rectangles as there is space.
3. If the time taken is more than 1 second, terminate.
4. If not, reduce size of rectangles.
5. Repeat from step 2.

The application terminates when it has drawn the maximum possible amount of rectangles on the screen within a second. The results, shown in Table 6.1, show that Jogl outperforms Java 2D more than 15 times.

	Amount of Rectangles
Java 2D	774
Jogl	15446
Java 2D w/ Hardware Acceleration	3777

Table 6.1: Benchmark results

Through optimisations and putting newly developed hardware capabilities to use, JOGL can render even faster. This is also true for Java 2D: Enabling a feature introduced in Java 2D in mid 2007 allows Java 2D to render with hardware acceleration. This boosts the performance of Java 2D considerably, however it is still outperformed by JOGL.

What is presented here is the result of a highly specialised benchmark of Java 2D and JOGL. An exhaustive test would test each API using different render methods and filtering techniques, comparing the changes in results across these tests. One would for example expect Java 2D to suffer less from enabling anti-aliasing, as its graphics pipeline is specialised for these sorts of filtering techniques. A truly exhaustive comparison of the two APIs would require other test designs as well. Still, this test mimics the usage for which the graphics APIs are needed, drawing a large number of rectangles. This taken into consideration, the test gives a good indication of the performance to be expected from the two APIs.

6.2.6 Comments

Drawing with JOGL is obviously done on a lower level, programmatically. As opposed to Java 2D where one is working with geometrical primitives, one is working solely with vertices (and transformations), and resolve the coordinates required to draw the shape.

6.2.7 Choosing JOGL over Java 2D

While JOGL lacks the versatility of Java 2D as mentioned above, JOGL provides direct access to the graphics hardware, freeing the CPU of work related to drawing. Calls to OpenGL functions return instantly, allowing the Java thread to continue, while the function called waits for execution in the OpenGL pipeline. This opens for higher performance compared to Java 2D. A consideration when using a non-standard library, is the impact it may have on platform independence, and how the application is to be distributed. JOGL requires two libraries, and a dynamic link library (DLL) in order to run. Distributing the software as a Java Web Start application (requires Java Runtime

Environment 1.4.2 or newer), allows these resources to be automatically downloaded.

6.3 Investigation of Data Size

One goal of this project is for the browser to display annotation data for a genome, initially focusing on the human genome. The browser should also attempt to present the underlying DNA sequences. This section presents an investigation into the total size of the data to be handled. I first summarise what data is needed, secondly, I investigate the size of each component. Equations are presented in the summary of this section.

6.3.1 Data to be Retrieved

In chapter 5 the features to be displayed for different parts of the human genome was presented. Here is a short summary of this.

- ◆ Chromosomes (*Genome data*)
- ◆ Genes (*Annotation data*)
- ◆ Exons (*Annotation data*)
- ◆ SNPs (*Annotation data*)
- ◆ Whole-genome sequence data (*Sequence data*)

An investigation into the total size of this data now follows.

6.3.2 Chromosome Data

Tables 6.2 and 6.3 sum up the data for chromosomes. A chromosome has an average of 40 cytobands. Summing up this information, equation 6.1 reveals the data size of chromosome information is as good as insignificant by present standards.

Field	Type	Size (Bytes)
Name	String	10
Length	Integer	4
Amount of Genes	Integer	4

Table 6.2: Data stored for a chromosome

Field	Type	Size (Bytes)
Cytoband name	String	12
Cytoband start	Integer	4
Cytoband end	Integer	4
Cytoband color/shape	String	16

Table 6.3: Data stored for a cytoband

Field	Type	Size (Bytes)
ID	String	30
Starting base	Integer	4
Ending base	Integer	4
Strand	Boolean	1 bit

Table 6.4: Data stored for an attribute

6.3.3 Attribute Data

The data in Table 6.4 is generic data stored for attributes of all kind. With the limitations introduced in section 5.1, approximately 450 000 attributes will be viewable. Equation 6.2 shows that although numerous, the size of attribute data is surmountable.

6.3.4 Feature Data

In addition, attributes classified as features, e.g. genes, are further specified by the data listed in Table 6.5. About 25000 features are also genes. Equation 6.3 shows that larger fields mean that feature data is the largest component. Still the size is not deterring.

6.3.5 Summary

The inclusion of SNP information into the browser has been omitted due to limited time. The following equations sum up the previous chapters: chromosomes:

$$25 * (10 + 4 + 4) * 40 * (12 + 4 + 4 + 16) = 648000 \quad (6.1)$$

attributes:

$$450000 * (30 + 4 + 4) = 17100000 \quad (6.2)$$

Field	Type	Size (Bytes)
Name	String	20
Description	String	1000
Type	Integer	4

Table 6.5: Data stored for a feature

features:

$$23000 * (20 + 1000 + 4) = 23552000 \quad (6.3)$$

The total size of this data, below 50 MB, is highly manageable. In addition to this data, is the sequence data to be displayed on high levels of zoom. A study into the handling of this is made in section 6.8.

6.4 Evaluation of Data Sources

An important decision in this project is the source of the data to be displayed. As seen in section 2.2, databases offering annotation data and sequence data, are abundant. The following sections presents two alternatives that have been considered.

6.5 NCBI Entrez

One of the major databases for sequence data is GenBank. It is hosted by NCBI which provides tools for developers for accessing GenBank and other databases. Entrez is NCBI's web interface towards GenBank, and other databases, for searching, displaying and downloading data. Entrez has a set of development tools for building specialised data pipelines, called eUtils. One such tool is used to search for records in specified databases, using a string, or other type of term. The result is a list of matching records appearing in order of relevance. Another tool can then be used to retrieve the most relevant one. Results take the form of XML documents. A short introduction to XML can be found in chapter B. Many different types of data are available through the eUtils, and advanced ways of relating data are possible using the different tools. Information in a particular database might have links to information in other databases, enabling the creation of powerful specialised data mining tools.

A procedure for retrieving some data using eUtils could progress in the following way, using the two most intuitive eUtils, ESearch and EFetch:

1. Construct a URL to the search tool (e.g. ESearch), with parameters (eg. `?db=nucleotide`).
2. Post the URL with HTTP-GET.
3. Interpret/parse the XML response, possibly obtaining a primary ID
4. Create a new URL, now to the EFetch tool, with the obtained primary ID as an obligatory parameter.
5. Post URL and parse the XML response.

Primary IDs

All records are denoted by a primary ID, which is returned as the results of searches, and expected when fetching records. Primary ID is a generic name for the ID of a record available through eUtils; For records stemming from the taxonomy database, the primary ID is the TAXID, while the primary ID of records in the nucleotide database is the GI number. This is important when pipelining searches across different databases.

Databases

The databases accessible through the eUtils aren't necessarily concrete databases. E.g. the first database, nucleotide, contains/provides access to nucleotide sequences stored in several different databases. There is an interactive map of the Entrez database model helping to understand this.

- ◆ nucleotide
Contains nucleotide sequences. Each record typically represents a separate gene.
- ◆ genome
Contains whole chromosomes, sequence contigs and genetic- and physical maps.
- ◆ omim
Contains records for genes, and their relatedness to diseases or other phenotypes, and records for diseases mentioning related genes.
- ◆ protein
Contains protein sequences for different genes.
- ◆ pubmed
Contains entries describing publications in the pubmed databases, with abstracts and links to downloadable full text copy.
- ◆ taxonomy
Taxonomy database records each organism represented with sequence data in Entrez.

eUtils tools

eUtils tools are accessed with a URL pointing to the tool script, with parameters. The possible parameters differ for the respective eUtils. Some parameters are required, while others are optional. These are the tools that make up the eUtils, as described by Sayers and Wheeler [2007].

- ◆ EInfo
Provides a list of all databases. If parameterised, it provides details for the database in question (field info, update dates etc.).
- ◆ ESearch
Searches a specific database, Responds with a list of resulting IDs. Searches PubMed by default.
- ◆ EGQuery
Tests how many hits are generated in each accessible database by a text query.
- ◆ ESummary
Takes a list of IDs, returns the summaries of these records.
- ◆ EPost
Takes a list of IDs, saves it, and returns a query used to retrieve it.
- ◆ EFetch
Returns the data records denoted by a list of IDs. The WebEnv of a search or other query can also be given.
- ◆ ELink
Can be used to relate records in a database to associated records in either the same database or in another Entrez database.

An example query

An example query now follows. In this example A search for the terms "horse" and "insulin" is made. Then one of the resulting records is fetched. For reference the base URL of all of the eUtils is given here:

```
http://eutils.ncbi.nlm.nih.gov/entrez/eutils/
```

The search for horse insulin is performed using the ESearch eUtil, by parameterising the HTTP request to ESearch, one can specify the search term as well as additional directives. The URL of the ESearch call:

```
esearch.fcgi?db=nucleotide&term=horse+AND+insulin&usehistory=y
```

Notice the parameters **db=nucleotide**, **term=horse+AND+insulin** and **usehistory=y**, specifying which database to search, the search term and opting to store this search on the server for easily retrieving the results. The last option results in the addition of a WebEnv code in the result. This can be seen as a type of browser cookie, used to remember this search.

The result is given as an xml file:


```

<?xml version="1.0"?>
<!DOCTYPE eSearchResult PUBLIC
"-//NLM//DTD_eSearchResult ,_11_May_2002//EN">
<eSearchResult>
  <Count>80</Count>
  <RetMax>20</RetMax>
  <RetStart>0</RetStart>
  <QueryKey>1</QueryKey>
  <WebEnv>0wohZKRxGEZ-oI6Sii
ZwjoS0DdBy4S9i-KZxrDwo2xCr
jDYOptpUiMBFrwbTox5ySiuq0R_
ogVz9@40510C507E9D2720_0195SID</WebEnv>
  <IdList>
    <Id>126722830</Id>
    <Id>169264560</Id>
    <Id>169264559</Id>
    <Id>167621432</Id>
    <Id>155969720</Id>
    <Id>47523915</Id>
    <Id>126352613</Id>
    <Id>154089545</Id>
    <Id>149758908</Id>
    <Id>149758890</Id>
    <Id>149758872</Id>
    <Id>149758870</Id>
    <Id>149757787</Id>
    <Id>149757777</Id>
    <Id>149756157</Id>
    <Id>149755414</Id>
    <Id>149745103</Id>
    <Id>149745051</Id>
    <Id>149741614</Id>
    <Id>149738505</Id>
  </IdList>
  <TranslationSet>
</TranslationSet>
  <QueryTranslation></QueryTranslation>
</eSearchResult>

```

Fetching these results can now be done by extracting the WebEnv and QueryKey fields, passing these as parameters to eFetch. The lengthy WebEnv is not repeated here:

```
efetch.fcgi?database=nucleotide&WebEnv=X&query_key=1
```

The result is a specially formatted file.

SOAP Interface

SOAP (initially: Simple Object Access Protocol, presently no intended abbreviation) is a protocol defining the passing of messages between web services. SOAP messages take the form of XML, a short introduction to which is found in appendix B. SOAP allows a client written in Java, running in a Unix environment to interact with a server written in C#, running on a Windows Server. Working on top of the same protocol used to transfer web content, SOAP messages has less chance of being stopped by firewalls and security measures than older approaches. SOAP was initially short for Simple Object Access Protocol. This was concluded to be misleading, following a later release and consequently dropped.

eUtils also has a web service interface, accessible through exchanging SOAP messages. Requests to the eUtils are then treated as Java (or e.g. C#-) objects, and the URL parameters previously introduced are set as object attributes. Results are returned with the query call, so any downtime or processing time will block the execution of the query. Results are also represented by objects.

Comments

Entrez eUtils is an impressive and versatile tool. However the two most important reasons for using such a tool are not prevalent in this project; firstly, the need for continuously retrieving the latest data, secondly the need for interactively fetching and linking data.

6.6 Biomart

Biomart is a data mining tool, created partly under the responsibility of EBI. Biomart itself is a system that can be downloaded and configured to provide e.g. a web interface for querying a single- or several databases. EBI has a biomart interface to Ensembl at www.biomart.org.

Querying Biomart

To create a query, one uses the web interface to select database, dataset(e.g. Homo Sapiens Genes), a list of attributes to include and lastly delimiting filters. The resulting dataset can be downloaded right away, or saved. Saving the query, as an XML document or Perl² script, allows for downloading the result set at a later time. Downloading the result set, one can specify format

²Perl, a versatile and now widely used scripting language.

(HTML³, Comma spaced values, tab spaced values etc.) and whether or not to compress the file.

Data Format, Considerations

Downloaded data sets in flat file formats containing chromosome, gene and exon data, do contain redundancy. All information requested are included in each line in the file. This means that if we have requested chromosome name, gene name, gene description and exon name, and a particular gene has 3 exons, three lines in the file will be identical except in the field exon name.

6.7 Selecting Data Provider

This section discusses the advantages and disadvantages of the two sources investigated, and leads up to the selection of one of them.

Data Quality

The only requirements made by this project concerning the quality of the data, is that it is correct and that it can easily be linked to the corresponding data in other databases. These requirements are fulfilled by both alternatives, though in slightly different ways. In eUtils, facilities for linking are embedded, though only with the supported databases. With Biomart, appropriate gene info can be extracted and used to search any database for meta-information.

Data Expiration

A flat file downloaded through Biomart, will eventually become outdated. The latest build of the entire human genome was released in September of 2006, and annotations are continuously updated. In this project, however, details regarding for example the exact position of genes are not critical. What is more, re-downloading the required information overcomes the problem. Querying Entrez, one can be sure of always downloading the latest data. This being said, a data retrieval model that is continuously downloading required information, will be highly bandwidth intensive and not feasible. This means that both alternatives involve downloading the information at longer intervals, e.g. at most once a month.

Versatility

Initial development will be more rapid if opting for a flat file. It is faster to use a simple parser for a flat file than to use a generic xml parser to convert

³HTML - Hypertext Markup Language. A language for marking up content, creating a web page.

data into domain objects. A tradeoff by using flat files is the need to rewrite the parser if new fields are added to the query.

6.7.1 Conclusion

Biomart was the best option in this project, as it allowed the rapid build of a first prototype. Furthermore, the browser is not critically dependent on the latest genome and annotation data, hence the need for constantly fetching fresh data is absent.

6.8 Integrating Sequence Data

At high levels of zoom, the sequence, i.e. the DNA bases will eventually appear. The entire human DNA sequence, counting approximately 3 billion (10^9) bases, has a size of ca. 2800 MB in fasta format. This section explains difficulties, deliberations and decisions regarding this.

6.8.1 Regarding Data

Sequence data for the human genome available online stems as good as exclusively from a particular build of the genome. NCBI36 is the current build, available since the fall of 2006. Once again eUtils can be used, downloading from the Genbank database. Ensembl provides SQL⁴ and FTP⁵ interfaces to their data. The data stored by Ensembl is synchronised with that of Genbank. Sequence files in fasta format were downloaded for each chromosome. See section 2.2.1 for an example of the fasta format.

6.8.2 Parsing Approaches

When using the browser, the DNA sequence appears when zooming in close enough that one base fits inside each pixel. Each base can at that point be represented by a one pixel wide bar, coloured according to which base is present. The following user actions shall be possible:

1. Zoom in on an arbitrary part of the chromosome to see the sequence.
2. Move the view forward, and see the sequence.
3. Move the view backward, and see the sequence.

Each action is essentially an access of a part of the file of arbitrary length and position. Approaches for parsing the file and enabling this that have been considered are introduced in the proceeding sections.

⁴SQL - Structured Query Language. A widely used language standard for accessing the information in relational databases.

⁵FTP - File Transfer Protocol. A cross-platform internet protocol for effective file transfer.

6.8.3 Sequential Read, Entire File

The approach familiar to most novice java programmers, reading the file completely into memory, involves the following steps:

1. Open file for reading
2. Read entire file into a suitable data structure, e.g. StringBuffer.
3. On each request for a subsequence, simply index the data structure.

Storing the each sequence symbol in char type, as is essentially done in a String or StringBuffer, will have a file of 230MB take up 460MB of java heap space. The reason: a char is 16bits, while an ASCII encoded⁶ character is 8bits. Allowing an application to take up 500MB of memory is not a problem in most modern machines. However, such high memory demand will potentially restrict a number of users from running the browser. While there is no overview of the average amount of memory in computers in Norwegian schools, relying on 1024MB being available might be unrealistic. For this approach to be at all relevant, each symbol A, C, T or G must be encoded in the minimum amount of space, 2 bits. This is possible by encoding 4 symbols using a byte. The performance of this approach will be governed by the read operation in step 2 and is investigated in the following section.

Experiment

To test the performance of the sequential read approach, the arithmetic average of the time taken for 10 reads of chromosome 1 was calculated. Chromosome 1 at 230 MB, takes a just over a minute to read on The author's laptop (7200RPM Hard Drive, DDR2 800MHZ memory). Only storage in a StringBuffer is performed in the testing routine.

Encoding to 2 Bits

Compression of DNA sequences is a field of study within Bioinformatics separate from the topics of this thesis. A simple solution, encoding 4 symbols to 1 byte, is to construct a lookup table. Each byte in the possible range is resolved to a distinct sequence of 4 nucleotide symbols. The file is read four bytes at a time, the corresponding byte can be found by mapping using a hashing function.

⁶ASCII encoding is a standard for representing the English alphabet in digital computers, created by the American Standard Code for Information Interchange (ASCII).

6.8.4 Random Access File

A java `RandomAccessFile` (RAF) seems perfectly suited for this problem. Instances allow file content to be accessed randomly, much like a standard array. The index into the array is a file pointer, that can be set by calling a method. In this way parts of a file of arbitrary length and position can be quickly read. Setting up a RAF and reading a subsequence would progress thusly:

1. Open `RandomAccessFile` for reading.
2. Seek to start of subsequence.
3. Read the `n` bytes.

The performance of this approach appears to be dependent on the current position of the file pointer in relation to the last. A quick test implies that a current position lower than last position often takes more time than the opposite case. In addition skipping far ahead takes more time than skipping only a little. None of the cases, however require more than 40 milliseconds. Memory usage is also marginal. RAF thus is a relevant alternative.

6.8.5 Memory Mapped File IO

A file can be mapped into the memory. The memory mapped file (MMF) is then a part of virtual memory, introduced in chapter B. Conceptually it is then divided into pages. Page size varies from one operating system to the next. Disk access is only performed when page boundaries are crossed, i.e. a previously 'untouched' part of the file is accessed. This is called a page fault. Only pages that are being accessed reside in physical memory. Revisiting a page might cause disk access if too much time has passed since the last access of that page. These steps can be followed to read the subsequence from an MMF:

- ◆ Create a `RandomAccessFile`
- ◆ Get its `FileChannel` (*1 method call*)
- ◆ Map the file channel into memory (*1 method call*)
- ◆ Position, then get the required amount of bytes.

The setup of an MMF is more elaborate than that of the previously discussed approaches. The performance of this approach is governed by the time a page fault takes to resolve. A quick test showed that this takes about 20 milliseconds. However, moving back and forth in a concentrated area takes an immeasurable (with java timing facilities) amount of time.

```

Initialise ( file )
WHILE i<100:
  StartTimer ()
  position = RandomPosition ( file )
  subsequence = Read ( position , 1000 )
  Store ( subsequence )
  time = StopTimer ()
  AddToMean ( time )
ENDWHILE
CalculateMean ()
CalculateSD ()

```

Figure 6.1: Algorithm used for testing RandomAccessFile and Memory Mapped File IO

6.8.6 Technical Comparison

This section presents a technical comparison of the two approaches RAF and Memory Mapped File IO. The algorithm used to test the approaches comprises the following steps:

1. Initialise
2. Seek to a random location in the file.
3. Read and store 1000 characters.
4. Repeat 100 times from step 2.
5. Calculate arithmetic mean and standard deviation.

Standard deviation was calculated using the following equation:

$$\hat{\sigma} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2} \quad (6.4)$$

The algorithm used is given in pseudocode in figure 6.1 The test algorithm used is only partially analogous to the usage pattern that will be present in the genome browser. A more realistic pattern of use involves fewer separated areas in the chromosome, and reads back and forth around these areas. The algorithm was chosen so that it theoretically partially favours both approaches. Judging from initial experiments, the RAF would incur a more or less constant cost when moving forward in a file, and a higher cost when reversing. Furthermore the MMF would incur about the same cost as RAF with the addition of overhead caused by page faults. The MMF would incur almost no cost when the read did not cause a page fault, the data would then already

	RandomAccessFile	Mem. Mapped File
Average , Best Run	11.25	6.25
SD , Best Run	13.6	9.6
Average , Worst Run	15.47	13.28
SD , Worst Run	19.25	17.9

Table 6.6: Average time cost of a read and standard deviation (SD)

be in RAM.

The results, seen in Table 6.6, speak in favour of memory mapped files. Surprisingly, the standard deviation is higher for RAF than MMF. An investigation into this shows that the standard deviation for shifts forward in the file is approximately 20 times larger. The mean times for shifts in either direction only differ slightly.

6.8.7 Conclusion

Of all the approaches considered, using a random access- or memory mapped file is the most relevant approach. Of these two, memory mapped file IO performs marginally better. However, it is not beyond reproach. For some uses, memory mapped files might not perform better than the other approaches. Sequentially reading through a large file, there will be all page faults, no part of the file has previously been accessed. The usage pattern is such, though, that this drawback will be mostly avoided. A personal motivation for using memory mapped file IO existed also. The chance to gain experience using a relatively novel and technically interesting programming feature was welcomed. Consequently, memory mapped file IO was used for handling sequence data.

6.9 Implementing Presentation of SNPs

The visualisation of SNPs in the new, simpler browser was not completed. The informal, incremental style of development used, lead to this feature being prioritised after the visualisation of genes, exons and nucleotide sequence. This section presents some investigations and work that was done in preparation of the implementation.

6.9.1 Source of SNP Information

In addition to annotation data needed for gene information etc., Biomart offers SNP data for various genomes. More than 13 million SNPs are available in total, approximately 6 million are validated. Parsing a flat file with about 6 data columns per SNP would bring new challenges to this project.

6.9.2 Drawing

The discussion in chapter 5 led up to the decision that SNPs would only be shown individually when viewing the chromosome up close. With 10 thousand bases visible, 50 SNPs would be visible. Farther away, the presence of SNPs would be quantified using a graph indicating the density of SNPs. By using a 3rd party graphing API based on Java2D, the drawing of the graph could be automatised, and the result integrated in the browser window itself. A package developed at the University of Southern Queensland, Australia is capable of this.

A central technical decision is whether the class width of the histogram, represented by the graph, should be fixed or recalculated for different levels of zoom. The latter option is favourable, allowing a reasonable resolution at any level of zoom. Constant recalculation might hamper performance, so this would require thorough testing and optimisation.

Chapter 7

Introducing Sigve

This chapter presents the appearance and functionality of Sigve, the new, educational browser.

7.1 Initial View

Having started Sigve, users are faced with the view shown in Figure 7.1. Users get an overview of the different chromosomes of the human genome, names, the amount of genes in each chromosome, and cytoband information is visible. Cytoband names appear in tool-tips when hovering the mouse over them.

Possible Interactions

Users can click and drag on a chromosome, creating a 'lasso'. This action allows users to 'inspect' part of the chromosomes, and they get a feedback stating the length in basepairs of the inspected region and the amount of genes in it. The details are then displayed in the subpanel titled 'Inspeksjon' in the topmost part of Figure 7.1.

Single-clicking a chromosome, opens a two-stranded view of that chromosome. The browser centers the view on the part of the chromosome that was clicked. The two-stranded view opens at a fixed level of zoom, an example is shown in Figure 7.2.

7.2 Two-stranded View

Two-stranded view (Figure 7.2) shows features and structures on a particular chromosome. In the current version, features i.e. genes are visualised as green rectangles whereas structures, i.e. exons are visualised as slightly narrower rectangles. The lengths of rectangles indicate the spatial extent on the chromosome, of the feature or structure represented by that rectangle.

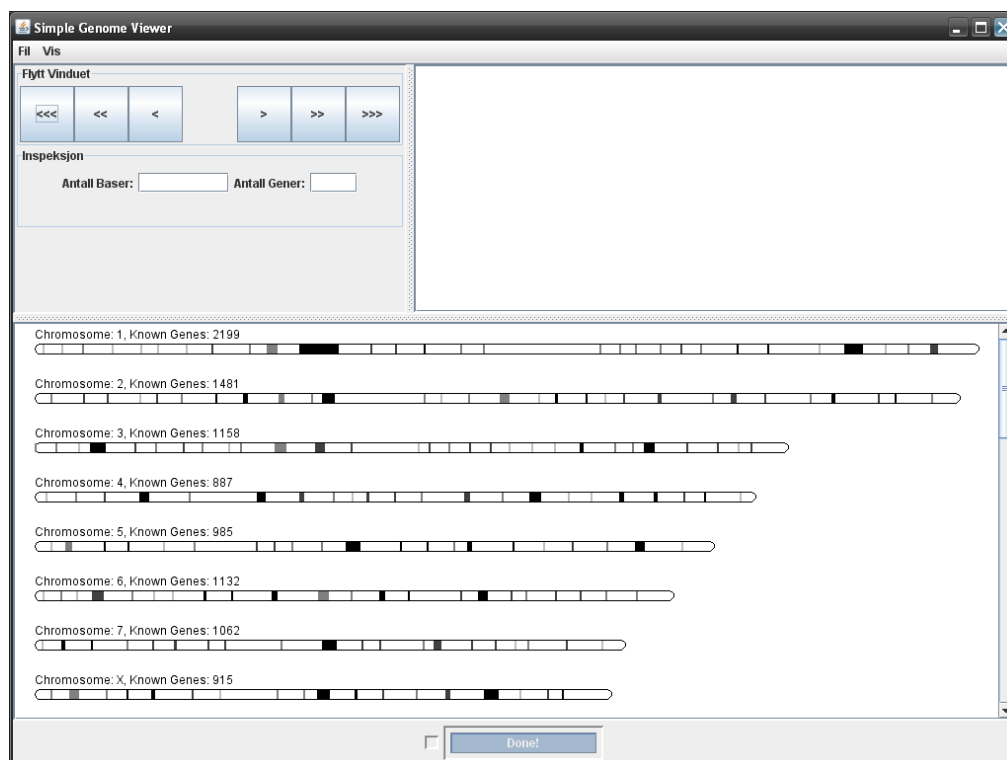


Figure 7.1: Sigve: Initial View / Overview

In the bottom of the view, there is a representation of a chromosome with a triangle indicating the current position on the selected chromosome. It also shows the size of the displayed view relative to the entire chromosome, however this is only intuitive at certain levels of zoom.

Possible Interactions

Zooming in and out can be done using arrow keys (up- and down) or using the mouse wheel. The subpanel titled 'Flytt Vinduet' can be used to pan the view various distances to the right or to the left. The window is repainted continuously when zooming or panning the view, yielding a higher sense of interacting with the genetic material. In Figure 7.3, the browser is displaying an area at three different levels of zoom.

Inspection can be made in the same way as in the initial view, by clicking and dragging. Lassoing the DNA in this way, can be done on either- or both strands. One will see the amount of basepairs and genes inside the lasso in the subpanel titled 'Inspeksjon'.

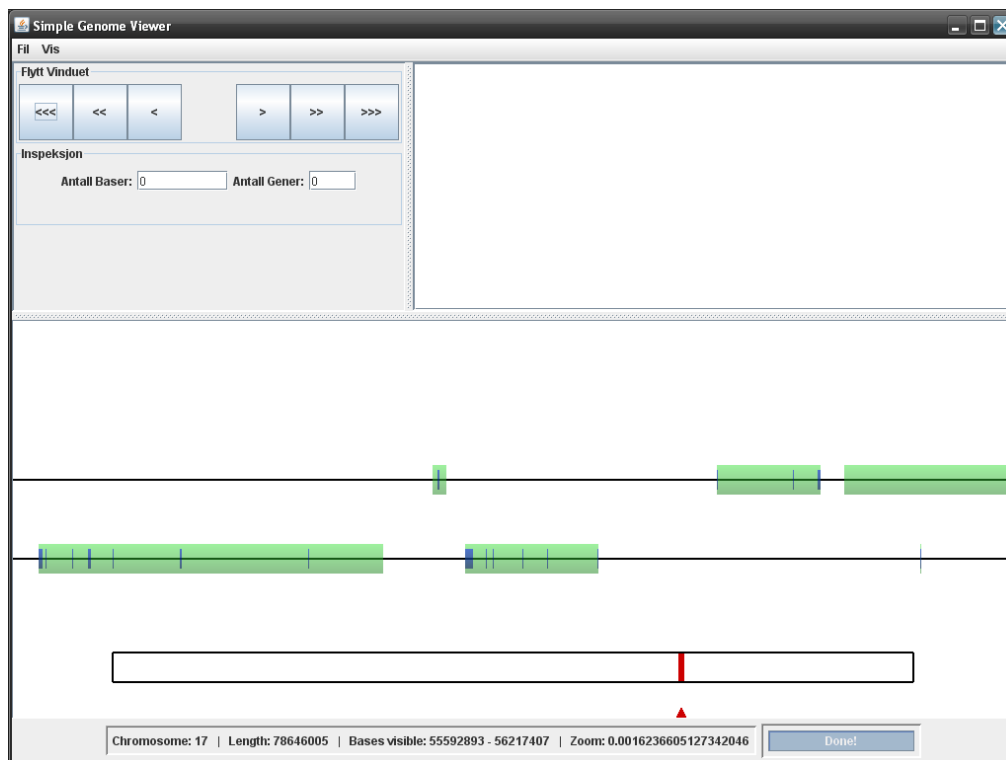


Figure 7.2: Sigve: Two-stranded View with genes and exons visible

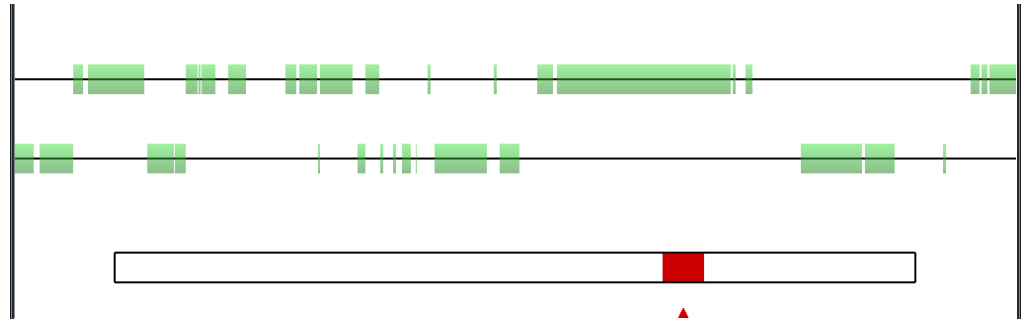
Users can click visible features and structures for more information on them. Clicking a gene will insert a green element in the list in the topmost part of the browser. This element will show gene name and description.

7.3 Two-stranded View, Sequence Visible

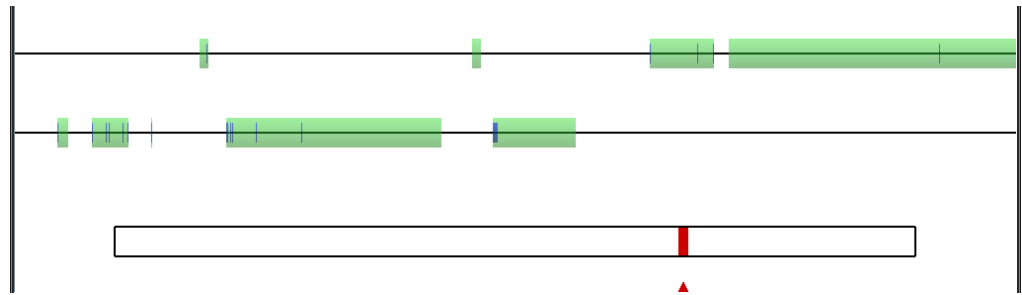
As the user zooms in close, the view will gradually transform into also displaying the nucleotide sequence. Figure 7.4 shows the leading part of the breast cancer-associated gene BRCA1. Present features or structures can still be seen as now larger rectangles. The same interactions are possible as for the two-stranded view.

7.4 Phenoportal

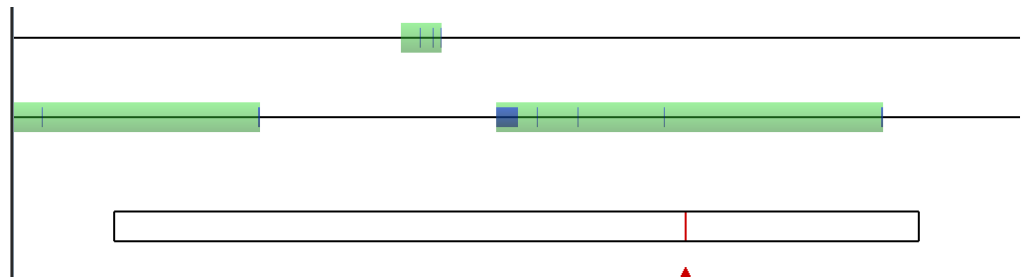
The Phenoportal is intended as an entry point into the browser for novice users. It provides a list of diseases and phenotypes and links them to features that have a proven relationship to that particular phenotype. It also provides



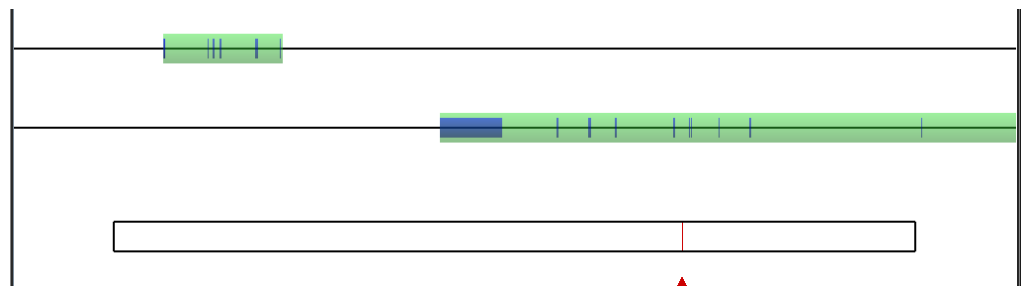
(a) Initial Zoom Level



(b) Close Zoom Level

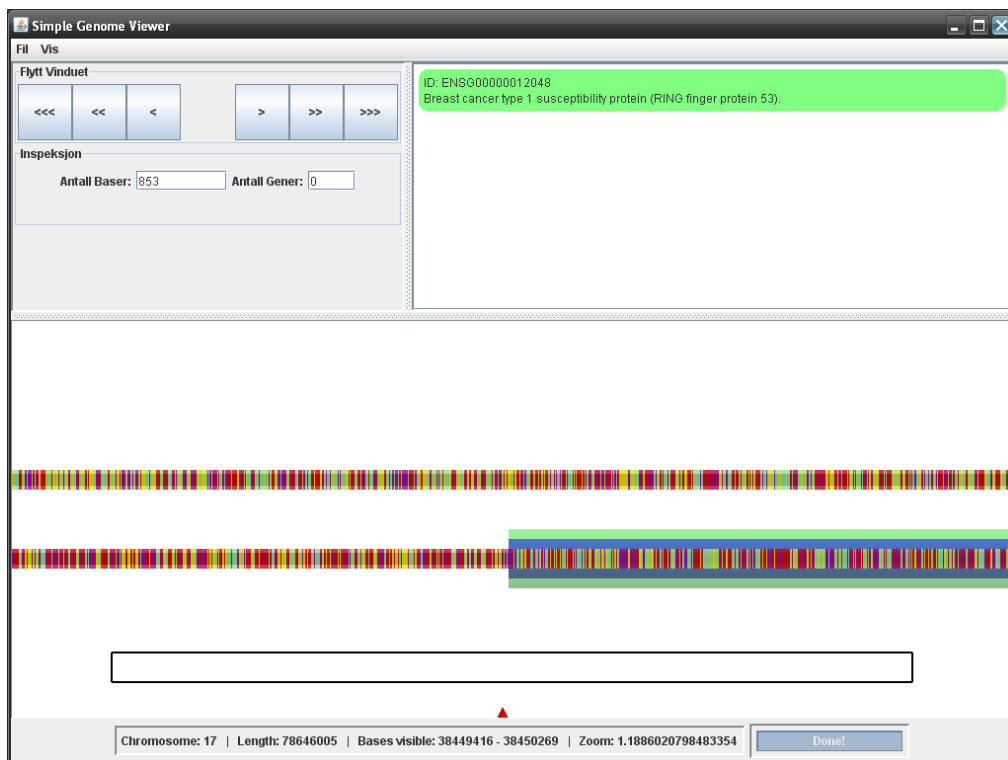


(c) Closer Zoom Level

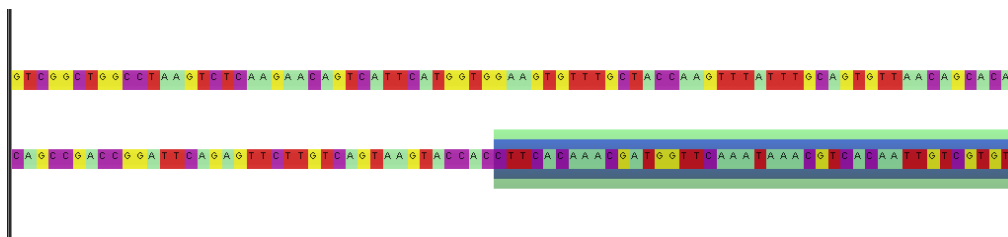


(d) Even Closer Zoom Level

Figure 7.3: Sigve: Examples of different levels of zoom



(a) Sequence viewed from afar



(b) Sequence symbols visible

Figure 7.4: Sigve: Two-stranded View with nucleotide sequence of BRCA1 visible

links to literature on both the phenotype and each related gene.

The Phenoportal is opened by clicking the menu bar item 'Vis' and then 'Fenoportal'. The user makes a selection in the list of phenotypes. This list can be seen in the upper left part of Figure 7.5. Situated immediately to the right, a description of the phenotype and a list of literature links will then be refreshed. Double-clicking a link will open the default web browser on that address. In the bottom part of the Phenoportal, the right-hand list is composed of genes with a proven relationship to the selected phenotype. Clicking

the overhead button, labeled 'Gå til/Se på', will bring the browser into focus and display the selected gene in two-stranded view. Back in the Phenoportal, links to literature describing each gene and its role in regulating the selected phenotype are provided in the next list to the right.

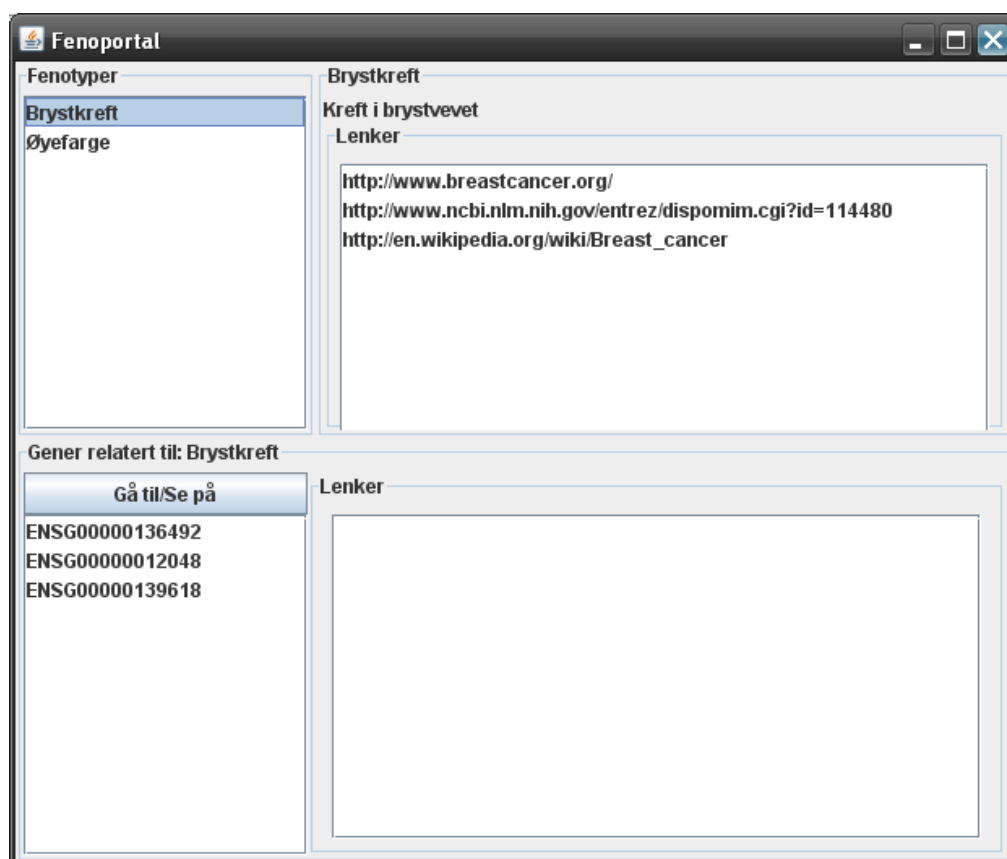


Figure 7.5: Sigve: The Phenoportal, listing literature links and genes related to breast cancer

7.5 Known Bugs

Inspection

There is some discrepancy in the reported amount of features and genes within the lasso. The amounts are off by some fixed percentage of the total amount of visible basepairs. The cause of this issue might be complex and some time could possibly be required in order to solve it.

Cytoband Tool-tip

When hovering the mouse over a chromosome's cytoband in the overview, a tool-tip appears with the name and stain level of the cytoband. The tool tip does not appear to change when moving the mouse to the next cytoband along the chromosome. This issue may be caused by the nature of the tool-tip component of the Java Swing GUI-toolkit (Graphical User Interface) or by errors in calculations in the underlying procedures. Using a different component to visualise the cytoband details could be the first step in isolating the cause of the issue.

Chapter 8

Conclusions & Further Work

8.1 Conclusions

In this project I've created a genome browser with a limited set of functions and a limited theoretical scope. The underlying idea has been that this angle of attack would produce a browser more suitable for use in education than existing ones. Throughout the project, focus has remained on incorporating the key functionality of a genome browser, and providing a good user experience. Usability and considerations regarding pedagogy have received much less focus. The idea all along has been that this project would result in a usable prototype, which in turn could be the starting point of later Master's theses.

Major technical challenges in this project have been the creation of a basic graphics engine from scratch, evolving parsing techniques to eliminate long waiting times, keeping memory consumption down when accessing sequence data files.

The resulting browser incorporates many of the functions expected of a genome browser. It visualises chromosomes, genes, exons, and nucleotide sequence. Information chromosomes, genes, exons is available. In addition Sigve has a portal, guiding users to interesting parts of the genome, the Phenoportal.

Sigve proves the feasibility of creating a genome browser from scratch. In doing this, one has more freedom of choice compared to opting to expand or convert an existing (open source) browser. This freedom of choice might hamper progress, offering many potential sidetracks and pitfalls. Devising a detailed and confined agenda is helpful in this regard. Studying existing browsers extensively proved very useful in this project.

I predict that some users of Sigve in the target group might benefit from

using the application. Sigve has the potential of being a tool with which students solve a set of exercises, possibly strengthening their comprehension. Having a relatively high learning threshold, some users might struggle more with the application itself and thus learn little from using it. Therefore, the statistical impact on theoretical comprehension in a test group might be so-so. It is however, the author's candid opinion that Sigve, with an amount of further work, has the potential of becoming an efficient learning tool.

8.2 Complaints

Perhaps the strongest complaint regarding this project, is the fact that I've attempted to create an educational genome browser with only marginal focus on pedagogy and only minimal involvement of people with a background in education. I haven't made use of advanced pedagogical theory in establishing requirements for an educational browser. Work has been based upon simple, a priori assumptions that an educational browser at least must be less comprehensive and more responsive than current browsers.

Furthermore, no focus group investigation or usability testing has been performed in order to establish the target group's need for-, special requirements for-, or benefit from an educational genome browser. All these issues should be considered in future projects intending to perfect Sigve for use in education.

Another complaint is the little amount time that has been available for ensuring a higher level of user-friendliness and usability. Throughout the project, I have been forced to prioritise the implementation of important functionality above these issues. In the ensuing sections, I discuss and propose functionalities and other changes that can better this situation.

8.3 Missing Features and Changes

The end product, presented in the previous chapter, does not meet all the requirements set for the ideal educational genome browser. There are a number of features missing, ranging from simple to comprehensive. There are various reasons why these features were not implemented. One of these is a limitation in development time. Another reason is that the preparatory work did not assure the feasibility of the feature to be implemented.

SNP Visualisation

The preparatory work presented in chapters 5 and 6 represents the progress made in accomplishing SNP visualisation. As indicated, large amounts of data will be involved when implementing this. Future developers in this project

must find an efficient way to solve this while maintaining the level of interaction and response presently offered.

More Available Attribute Information

Novice users will perhaps struggle to find information on visible features. In its present state, the browser responds with this information to mouse clicks on visible objects. This function is too hidden and unintuitive. Work should be put into amending this.

A More Modular Framework

The visualisation of genomic features is presently hard coded into the graphics module. This technical solution somewhat obstructs the desire for a more modular solution. However with the current graphical library used, only a couple of parameters need to be passed to an add-on module in order to accomplish this. In this way, the genomic features currently visualised will form the backdrop for visualisations made by add-on modules.

8.4 Switching Graphical Framework?

Creating a genome browser from scratch using OpenGL, a relatively low-level API, might be disadvantageous compared to starting out with a higher-level framework or a graphics engine. OpenGL does allow simple drawing using a few simple calls, however the programmatic complexity increases linearly with the complexity of the scene to be drawn. In this respect using a graphics engine could be more accommodating when the complexity of the graphics to be drawn increases. Such higher-level APIs have the added convenience of facilities for user interaction, e.g. clicking objects in the scene, and access to geometrical primitives. On the other hand, becoming proficient in using a larger API, requires more work.

8.5 Program Extension, through an API

Providing expert/computer savvy users with the means to develop custom presentations will contribute in making it an effective learning tool. Providing an interface to the graphics system and the genome data are the minimum requirements of such an API.

The idea of end-user programming was spawned by the revolution caused by Xerox, Apple and IBM putting a computer on every desktop. The idea was that the users would have more power and influence on their work situation with the ability to customise their business software and that this would cause

the same pervasive changes in society as literacy. The optimistic outlook, characteristic of these early ideas, have been replaced by more reasonable, reflected views. Still, APIs or SDKs (Software Development Kit) are being provided with modern software products. In a specific area of the software industry this is more of a standard than a supplemental feature. The computer gaming industry uses SDKs to ensure that their products have a longer life-span. With end-users continuously creating new content and features for the game, it maintains a large user-base longer. Using computer games as an example is not entirely comparable, since many gamers have computer science education. However there are many examples of gamers starting out with 'modding', as it is called, and ending up as programmers.

Python, one of the programming languages visited in the previous chapter, was created using ideas from a project aimed at stimulating end-user programming. It is also deemed to be a good educational language, in that it is simple and more textual than other languages. Without doubt, Python is a good choice as an API language. The technical issues of integrating Python modules in a Java application can be solved using a software package based on normal Python, which allows Python modules to be compiled to Java byte code, usable from normal Java applications.

8.6 Focus on Usability

This important aspect of application development has been left out of the scope of this project. The focus has been put on creating a prototype with as many working browsing capabilities as possible. In order to accomplish a browser such as the one envisioned in the scenario in chapter 1, much more work need to be put in to the usability of Sigve.

8.6.1 Perfecting the GUI

Of the different components of usability the most important in this project are perhaps the following: *Learnability*, *efficiency* and *satisfaction*. Promoting these in Sigve will undoubtedly make it a more efficient pedagogical tool.

The learnability of Sigve is hampered first and foremost by the fact that the users are given no introduction to the different views. Some guidance into the colour coding of the rectangles representing attributes is a bare minimum here.

The efficiency can be strengthened by giving users more tools helping them to find what they are looking for. A scenario is that students have been given assignments to find e.g. the gene related to Huntington's disease, and they are required to use other literature instead of the Phenoportal. From other (online) literature they might have found the name of the gene. Sigve presently

offers no search functionality for 'jumping' to a gene given the gene name. This is one feature that could increase Sigve's efficiency.

Satisfaction may or may not be adequate, as the sole developer it is hard to analyse this aspect from the target group's point of view. However, user satisfaction is crucial for the user to maintain motivation when solving such tasks as the one described above.

These three aspects are believed to be the key components of the usability of Sigve. Hence, setting a list of usability requirements with focus on these three should be fruitful.

8.6.2 Usability Testing

After a period of perfecting the GUI, measuring its qualities in regards to a set of usability requirements will reveal further possible areas for improvement. Useful results can be gained by studying actual users performing a set of tasks, measuring the time to complete tasks, measuring the users' level of stress afterwards, and so on.

8.7 Visual Appearance

A number of changes and additions should be made to optimise the browsing experience for the target group. These have received low priority in this project and have been omitted.

8.7.1 Location Probes

According to Card et al. [1999] location probes can augment the informative effect of visual structures. Location probes are view transformations that present detailed data or structure and are activated upon user interaction with special locations. The tool-tip is an example of a location probe. When the user hovers the cursor over a gene, detailed information on the gene, could be presented in a frame which becomes visible.

8.7.2 Visual Indication of Zoom

Except for the size of each particular feature and the mini-map, the user is given no graphical indication of how much the chromosome is enlarged. Not giving such an indication is thought to hamper the effectiveness of the visual presentation. Beyond a certain level of detail, the mini-map and using the size of features for reference becomes ineffective. There are a number of options for improving this.

In order to communicate more clearly the current level of zoom, marking the sequence at set intervals could prove useful. In a typical diagram, either axis is marked at intervals with the amount in the relevant denomination corresponding that point along the axis. Correspondingly, marking the sequence at a fitting interval of bases will help the user orientate, and interpret the level of zoom quicker. Simply placing such *sequence milestones* at fixed intervals will be problematic for disproportionate levels of zoom, compared to the interval used. Using a logarithmic scale allows this interval to vary, remaining informative across all levels of zoom. The challenge of differentiating the markers when different intervals are being used then arises. Using a graphical form to represent these *sequence milestones* that users are able to preattentively process will be of great help. Preattentive processing refers to the initial cognitive organisation of the visual field, that happen without focusing [Healy et al., 1996]. The length of the markers can be used for differentiation. For example, when the interval between markers is short, i.e. the user has zoomed in very far, the sequence markers are longer. As the user zooms out, and the interval lengthens, the sequence markers shrink.

Another coding that can be used to help indicate level of zoom is colour. Using colour to differentiate the sequence markers either exclusively or in addition to using different lengths, will help users quickly understand the picture as the view changes.

Bibliography

- Stuart K. Card, Jock D. Mackinlay, and Ben Schneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, Inc, 1999.
- Francis S. Collins, Michael Morgan, and Aristides Patrinos. The human genome project: Lessons from large-scale biology. *Science Magazine*, 300 (5617):286 – 290, 2003.
- Xavier Estivill and Lluís Armengol. Copy number variants and common disorders: Filling the gaps and exploring complexity in genome-wide association studies. *PLoS Genetics*, 3(10):e190, Oct 2007. doi: 10.1371/journal.pgen.0030190.
- Christopher G. Healy, Kellogg S. Booth, and James T. Enns. High-speed visual estimation using preattentive processing. *ACM TOCHI - Association for Computing Machinery Transactions on Human Computer Interaction*, 3(2):107–135, 1996.
- Hildegard Kehrer-Sawatzki. What a difference copy number variation makes. *BioEssays*, 29(4):311–313, 2007.
- S. Khandelwal. Chromosome evolution in the genus ophioglossum. *L. Botanical Journal of the Linnean Society*, 1(102):205 – 221, 1990.
- NCBI. Entrez genome project statistics. <http://www.ncbi.nlm.nih.gov/genomes/static/gpstat.html>, 2008.
- National Institute of Health NIH. The human genome project completion: Frequently asked questions. <http://www.genome.gov/11006943>, 2 2008.
- Eric Sayers and David Wheeler. Building customized data pipelines using the entrez programming utilities, eutils. *NCBI Short Courses*, 2007.
- Jonathan Sebat, B. Lakshmi, Jennifer Troge, Joan Alexander, Janet Young, Pär Lundin, Susanne Månér, Hillary Massa, Megan Walker, Maoyen Chi, Nicholas Navin, Robert Lucito, John Healy, James Hicks, Kenny Ye, Andrew Reiner, T. Conrad Gilliam, Barbara Trask, Nick Patterson, Anders

Zetterberg, and Michael Wigler. Large-scale copy number polymorphism in the human genome. *Science Magazine*, 305(5683):525–528, 06 2004.

René Stöckel. One step ahead - 3d engines with j2me. Technical report, Bytonic Software, 2006.

Norwegian Directorate for Education & Training UDIR. Curriculum of biology, August 2006.

Jin Xiong. *Essential Bioinformatics*. Cambridge University Press, 2006.

Appendix A

Sigve, Instruction Manual

A.1 Installation

Installation of Sigve is a simple process, thanks to Java Web Start. This section guides the user through the few steps required.

Prerequisites

Before commencing installation, the user must observe these aspects:

- ◆ Installation requires Java Web Start, a launcher built into the Java Runtime Environment as of version 1.4
- ◆ The Application requires version 1.6 of the Java Runtime Environment.
- ◆ Users with no version of Java installed, or an older version than 1.4 installed, are recommended to navigate to <http://www.java.com/> and download Java version 6, which is the recommended version.
- ◆ Users with a post-1.4 Java version installed will have the required version of Java automatically installed by the Java Web Start launcher.

Starting the Installation

Installation is commenced by pointing a web browser to:

- ◆ <http://heim.ifi.uio.no/jorgehsv/sigve/sigve.jnlp>

Alternatively, there is a limited sandbox-version (requiring only 20 MB to be downloaded) available here:

- ◆ <http://heim.ifi.uio.no/jorgehsv/sigve-sandbox/sigve.jnlp>

This will initialise the Java Web Start launching routine, downloading Java, if necessary, and then the application. After all dependencies and the application itself is downloaded, the application will be launched for the first time.

A.2 Starting the Program

1st Time Launch

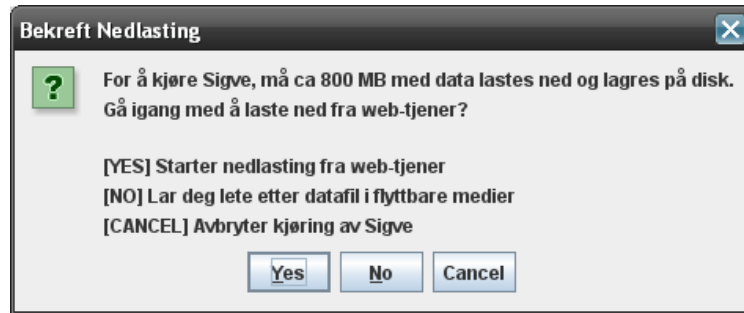


Figure A.1: A dialog asking users to confirm download of 800 MB of necessary data, or specify alternate data source

If Sigve is starting for the first time on a computer, some configuration is necessary. Sigve will look for necessary files. If these are not present, it will ask for the users permission to download necessary files, see Figure A.1. This can be a lengthy process for users with a slow internet connection. By choosing **No**, those users can specify an alternate download source, for instance removable media. By choosing **Yes**, the download of approximately 800 MB will commence. The download routine is relatively robust and reports progress, as can be seen in Figure A.2(a). When the download is complete, the data will be extracted, Figure A.2(b), and the application will start. **Cancel** will abort the launch.



(a) Download Progress Indicator

(b) Decompression Progress Indicator

Figure A.2: Download and Decompression Dialogs

Subsequent Launches

Subsequent application launches are done in either of the following ways:

- ◆ By clicking the desktop shortcut created by Java Web Start
- ◆ Finding "Sigve" under "Programs" on the Start Menu in Windows.
- ◆ By revisiting the URL mentioned in section A.1.
- ◆ By opening the Java Cache Viewer in the Java Control Panel ¹

The lattermost is only necessary in rare cases where Java Web Start fails to create shortcuts automatically.

When all necessary data is present, the application starts by displaying a splash screen while initial data is loaded. This normally takes less than 10 seconds.

A.3 Finding the First Gene

A successful launch leads users to the screen seen in Figure 7.1. Here, an overview of the chromosomes of Homo Sapiens is presented. By clicking a chromosome, browsing of that chromosome starts.

In the view that opens, genes are visible right away. Panning the view left and right, is done by right-clicking and dragging the mouse or using the buttons in the panel above the view.

Zooming in or out is done using the mouse wheel or using the arrow up- and arrow down keys. Information on a gene in the view can be shown by left-clicking it. This information appears as a list-item in the area to the above-right of the browser view.

Zooming in further, will expand the sequence and features. The black lines representing the strands will be replaced by the drawn nucleotide sequence eventually as the user zooms in.

Chapter 7 provides more details.

A.4 Using the Phenoportal

The Phenoportal shows a list of diseases or other phenotypes, and connects them to known regulating genes. There are links to literature on each pheno-

¹Detailed instructions available at: <http://java.sun.com/docs/books/tutorial/deployment/webstart/running.html>

type, and on the related genes. By clicking the button labeled 'Gå til/Se på', the browser will open the containing chromosome and center on the selected gene.

Chapter 7 provides more details.

A.5 Known Issues

Nvidia Graphics Hardware

Sigve has been tested on a limited amount of clients. A few of these are running Windows Vista. The combination of graphics hardware from the vendor Nvidia and the operating system Windows Vista have been noted to cause problems. In some cases unexplained crashes are experienced. In other cases hardware rendering fails and the application falls back on software rendering. In these cases, the application executes normally, but processor demand sky-rockets.

The cause of this is, according to unofficial sources, limited OpenGL support in Nvidia's device drivers for Windows Vista.

File Association

In some cases, installation of third party software or issues with the web browser, might have broken the Java Web Start file association with jnlp files. In such cases the application will not start simply by visiting the internet address given earlier. In these cases some work is required by the user to restore the correct file association. A guide to accomplishing this, as well as other helpful information on Java Web Start, has been written by Tobias Dezulian of the University of Tübingen. This guide is included as part of the appendix and will be a resource for solving the issue in question, whether on a Linux, Mac, or Windows platform.

Appendix B

Discrete Topics

B.1 Extensible Markup Language - XML

XML is a general purpose language used to structure and encode information. Information is surrounded by annotations, called tags. It is somewhat erroneously said to be a markup language, however it is a tool for creating markup languages. Comparing it to HTML clearly reveals this nuance. In HTML there is a set of tags available which are understood by the web browser. In XML one defines ones own tags.

An example XML document, a recipe for bread, is shown in figure B.1. A computer can understand and process the recipe for bread using an XML parser. Using the parser well, the computer will be able to understand any recipe. This is an important philosophy of XML.

```
<recipe name="bread" prep_time="5_mins" cook_time="3_hours">
  <title>Basic bread</title>
  <ingredient amount="3" unit="cups">Flour</ingredient>
  <ingredient amount="0.25" unit="ounce">Yeast</ingredient>
  <ingredient amount="1.5" unit="cups" state="warm">Water</ingredient>
  <ingredient amount="1" unit="teaspoon">Salt</ingredient>
  <instructions>
    <step>Mix all ingredients together.</step>
    <step>Knead thoroughly.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Knead again.</step>
    <step>Place in a bread baking tin.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Bake in the oven at 350(degrees)F for 30 minutes.</step>
  </instructions>
</recipe>
```

Figure B.1: Recipe for bread encoded in XML

B.2 Virtual Memory

Virtual Memory is a technique implemented by operating systems to make a larger amount of memory available to running programs than is physically available. In principle, programs, and their data, running on a computer must be in the main memory. Closed programs are stored on the hard drive. When more programs run simultaneously, the main memory might be exhausted. When the computer is out of memory, some program must be prematurely stopped. To counter this, the operating system requisitions available space on the hard drive. The least recently used program is moved into this portion of the now expanded memory. The mechanism responsible for this is the operating system's Virtual Memory Manager or VMM.

A central aspect of Virtual Memory is address translation. The computer's hardware components use various addresses to read data from RAM, hard drive etc. The addresses may be of different size and ranges may overlap. The Virtual Memory Manager makes a single, consecutive address space available to running programs. The block of data residing at a particular virtual address, may in fact reside on the hard drive. The VMM keeps track of this and translates the virtual address into the actual address.

Appendix C

Acronyms and Expressions

Expression	Explanation
NCBI	National Center for Biotechnology
EBI	European Bioinformatics Institute
EMBL	European Molecular Biology Laboratory
UCSC	University of California, Santa Cruz
VEGA	Vertebrate Genome Annotation, a database of manual sequence annotations
DNA	Deoxyribonucleic Acid. The molecular form of our genetic material.
RNA	Ribonucleic acid. Single stranded nucleic acid.
CNP	Copy Number Polymorphism. A population subgroup having a different number of copies of e.g. a gene.
CNV	Copy Number Variation. Also known as CNP.
LINE	Long Interspersed Nucleotide Element. An RNA reversely transcribed into the DNA (Retrotransposon)
SINE	Short Interspersed Nucleotide Element. A shorter retrotransposon.
API	Application Programming Interface. A set of tools for programmers to create or expand an application.
SDK	Software Development Kit. A more extensive set of tools than an API for creating / expanding software.
XML	Extensible Markup Language. A standard for structuring information, for easy transaction of information.
HTML	Hypertext Markup Language. A markup language for creating web pages.

ASCII	American Standard Code for Information Interchange. A specification for digital representation of the English alphabet.
SQL	Structured Query Language. A language for accessing relational databases.
FTP	File Transfer Protocol. An internet protocol allowing cross-platform file transfer.

Java Web Start Guide

by Tobias Dezulian

Java Web Start technology makes it possible to automatically download and start a Java application by clicking a link in a web browser. This works because the file ending “.jnlp” (MIME-type) is associated with the Java Web Start launcher program (“javaws”) which is part of your Java installation.

In some cases, this association is not in place. Please find below how you can manually set this association in your operating system. This should take less than 5 minutes.

Furthermore, this Java Web Start Guide tries to make the Java Web Start technology a bit more transparent.

Contents

Associating Java Web Start links	2
Windows	2
Mac	4
Linux	5
Launching the Java Control Panel	7
Windows	7
Mac	9
Linux	10
Launching the Java Console	11
Windows	11
Mac	12
Linux	13
About Java Web Start technology	15
Security issues	16

Associating Java Web Start links

In this section we'll have a look at

- how to detect whether the Java Web Start file association is correctly in place
- how to associate manually otherwise.

Windows

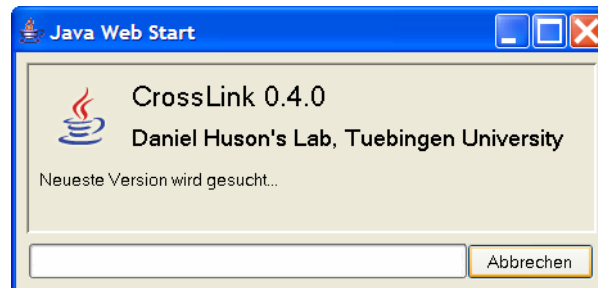


Figure 1 Download dialog – indicating that the file association is set correctly.

After clicking the Web Start link, this download dialog (above) should appear, indicating that the association is set correctly and the Java Web Start launcher is downloading the application.

In the firefox browser you can view these associations by opening the following window:

Tools -> Options

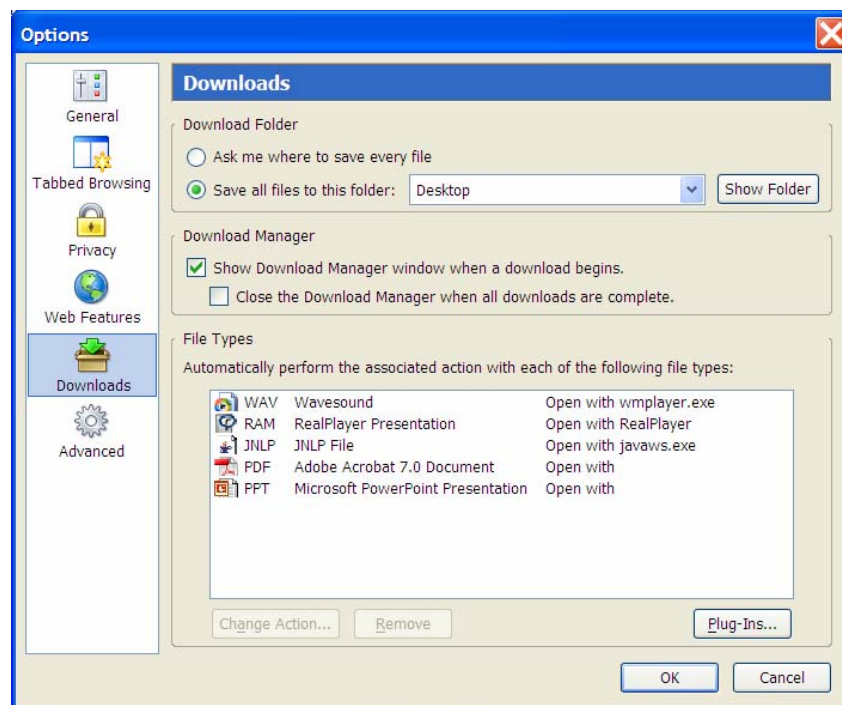


Figure 2 File associations in the firefox browser

If no association is set for the “.jnlp” file ending, your browser will ask what to do:

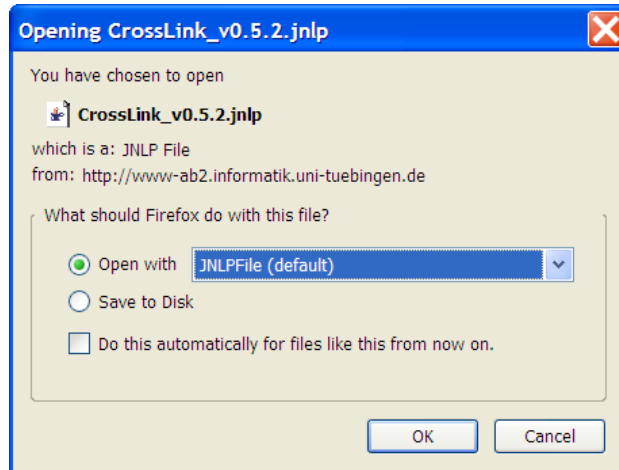


Figure 3 Browser asks what to do with this file type

In this case, associate the “.jnlp” file ending with your Java Web Start launcher program (“javaws”), which is located in the “bin” directory of your Java installation, e.g. at

C:\Programme\Java\jre1.5.0\bin\javaws.exe

You can use the windows „file search” to find the Java Web Start launcher program.

Mac

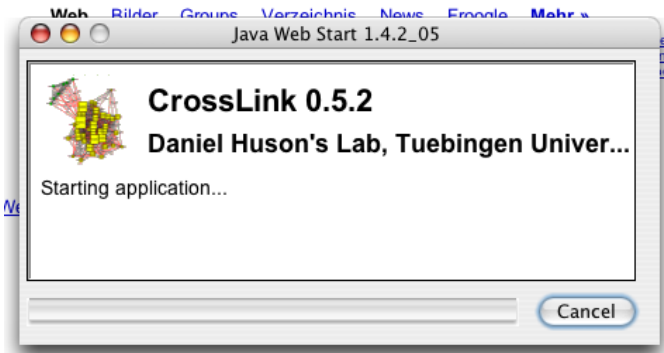


Figure 4 Download dialog – indicating that the file association is set correctly.

After clicking the Web Start link, this download dialog (above) should appear, indicating that the association is set correctly and the Java Web Start launcher is downloading the application.

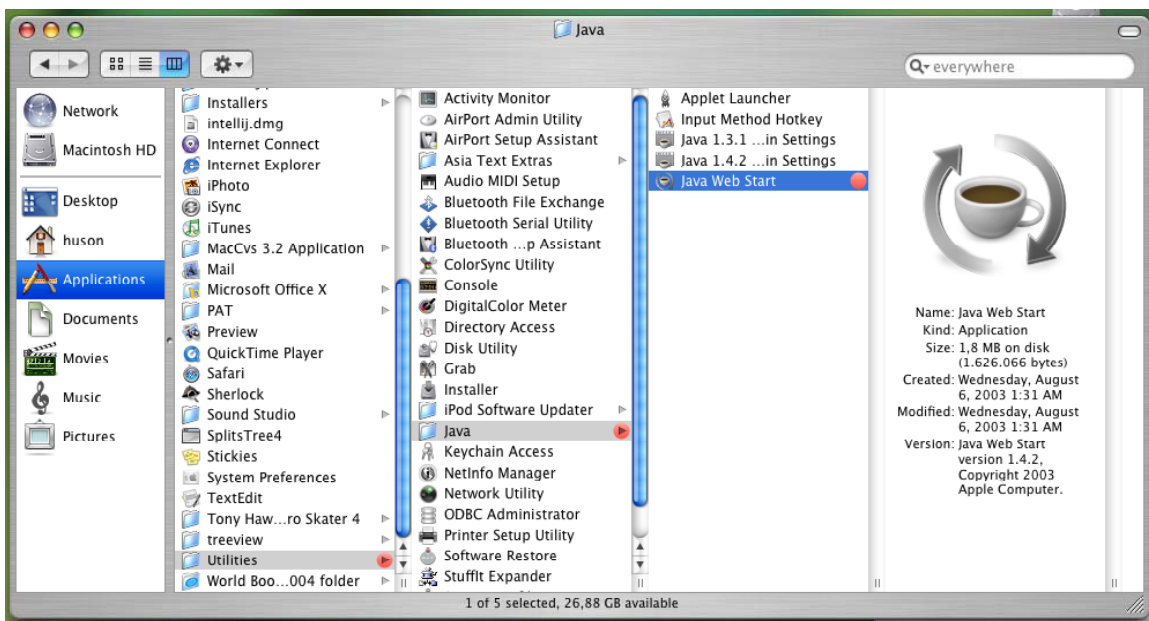


Figure 5 Location of the Java Web Start launcher program

If the file association is not in place and your browser can not handle the file correctly, you should associate the file ending (MIME-type) “.jnlp” with the Java Web Start launcher found at the location shown above.

Linux

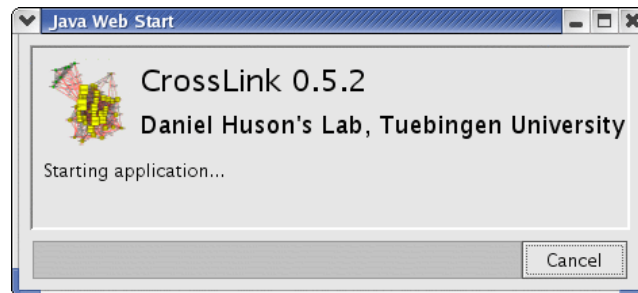


Figure 6 Download dialog – indicating that the file association is set correctly.

After clicking the Web Start link, this download dialog (above) should appear, indicating that the association is set correctly and the Java Web Start launcher is downloading the application.

If the association is not set correctly yet, the browser will ask what to do with the “.jnlp” file:

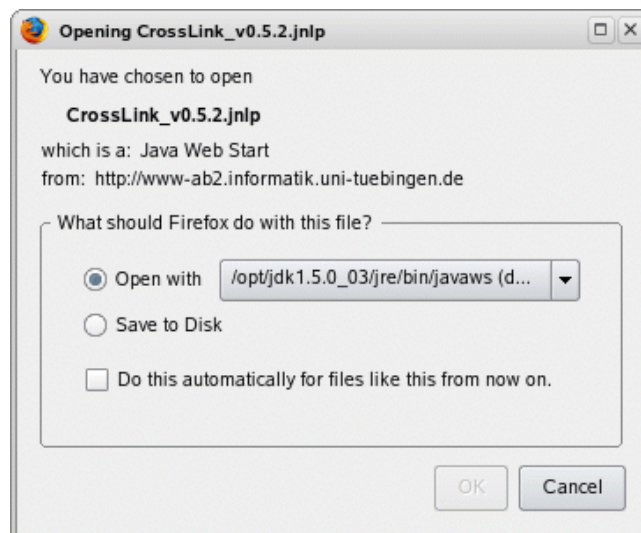
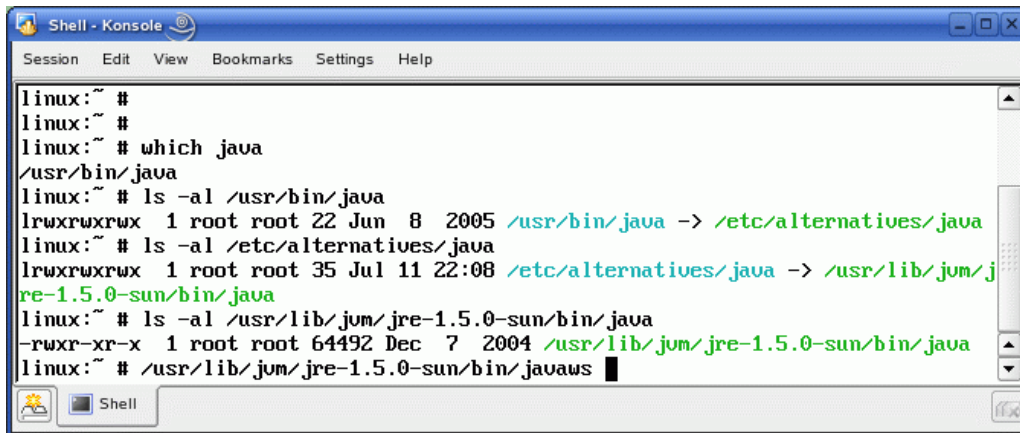


Figure 7 Browser asks for an association with the .jnlp file type

You will need to associate the “.jnlp” file ending with the Java Web Start launcher program (“javaws”) located in the “bin” directory of your java installation next to the “java” virtual machine.

You can find it using a disk search command (e.g. “locate”) or, if the java virtual machine (“java”) is in your path (check with the command “which”), by following the symbolic links, see below:



```
linux:~ #  
linux:~ #  
linux:~ # which java  
/usr/bin/java  
linux:~ # ls -al /usr/bin/java  
lrwxrwxrwx 1 root root 22 Jun 8 2005 /usr/bin/java -> /etc/alternatives/java  
linux:~ # ls -al /etc/alternatives/java  
lrwxrwxrwx 1 root root 35 Jul 11 22:08 /etc/alternatives/java -> /usr/lib/jvm/j  
re-1.5.0-sun/bin/java  
linux:~ # ls -al /usr/lib/jvm/jre-1.5.0-sun/bin/java  
-rwxr-xr-x 1 root root 64492 Dec 7 2004 /usr/lib/jvm/jre-1.5.0-sun/bin/java  
linux:~ # /usr/lib/jvm/jre-1.5.0-sun/bin/javaws
```

Figure 8 Following symbolic links to the „bin“ directory of the Java installation

Launching the Java Control Panel

The Java Control Panel is a graphical tool that allows you to set preferences for your Java programs – including the ones started via Java Web Start. The Java Control Panel is included in every Java installation.

In this section, we'll have a look at

- how to start the Java Control Panel in each operating system.

Windows

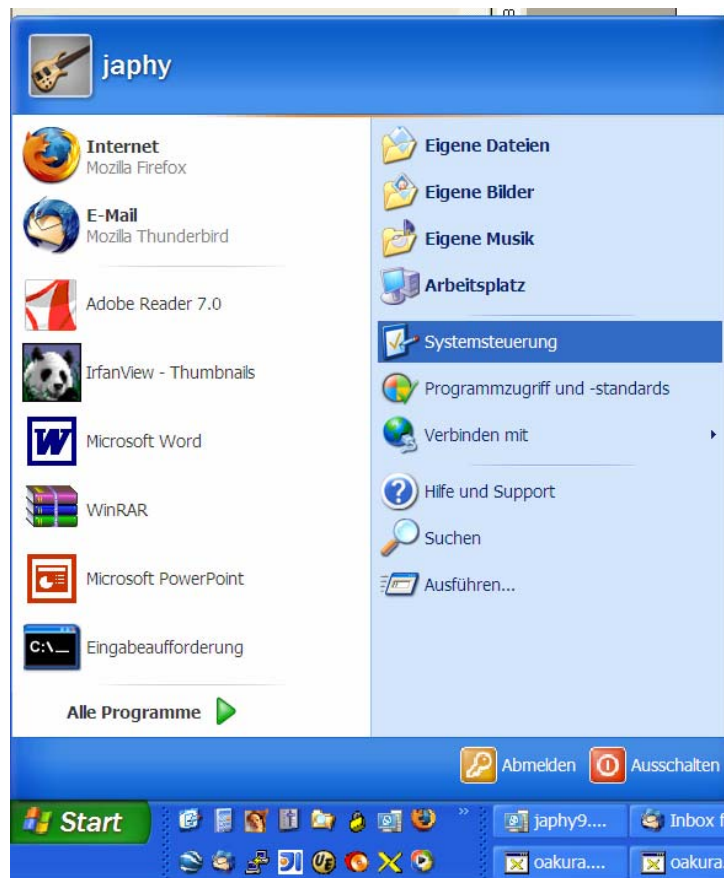


Figure 9 -- Step 1: Open the Windows control panel

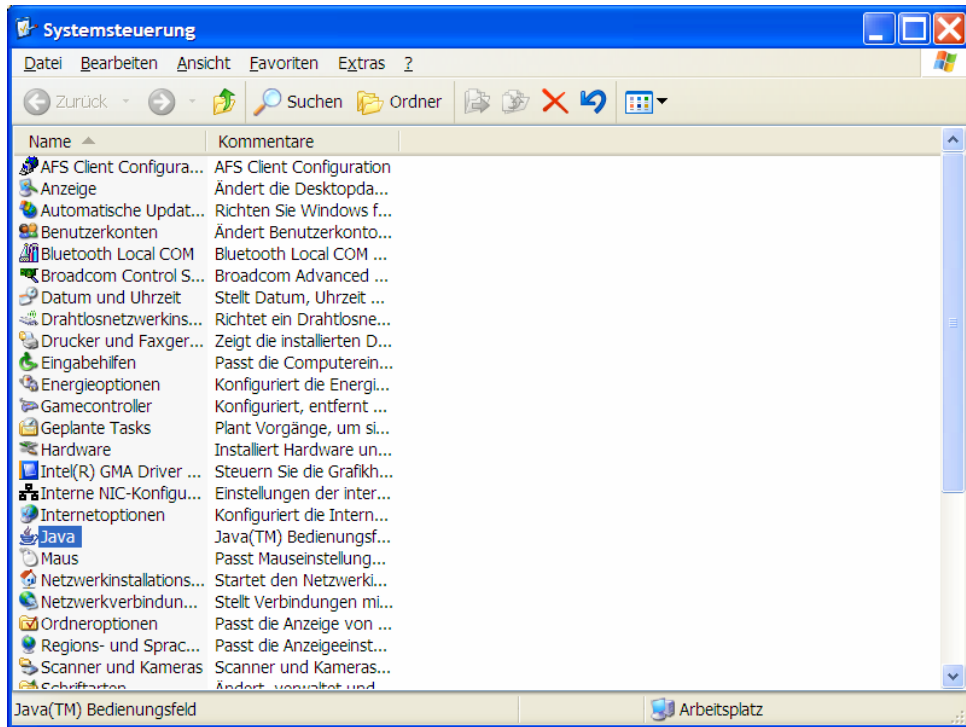


Figure 10 – Step 2: Click on the Java icon

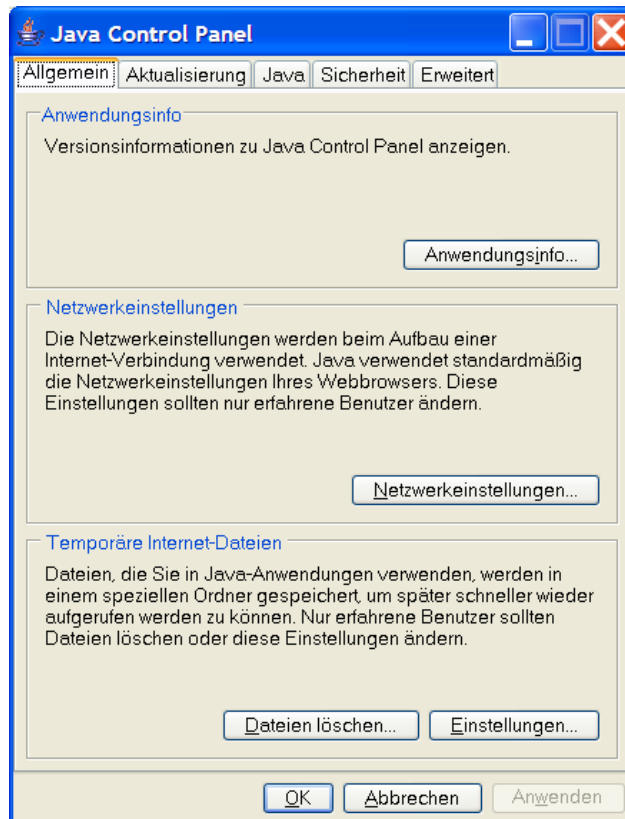


Figure 11 – Voilà: the Java Control Panel

Mac

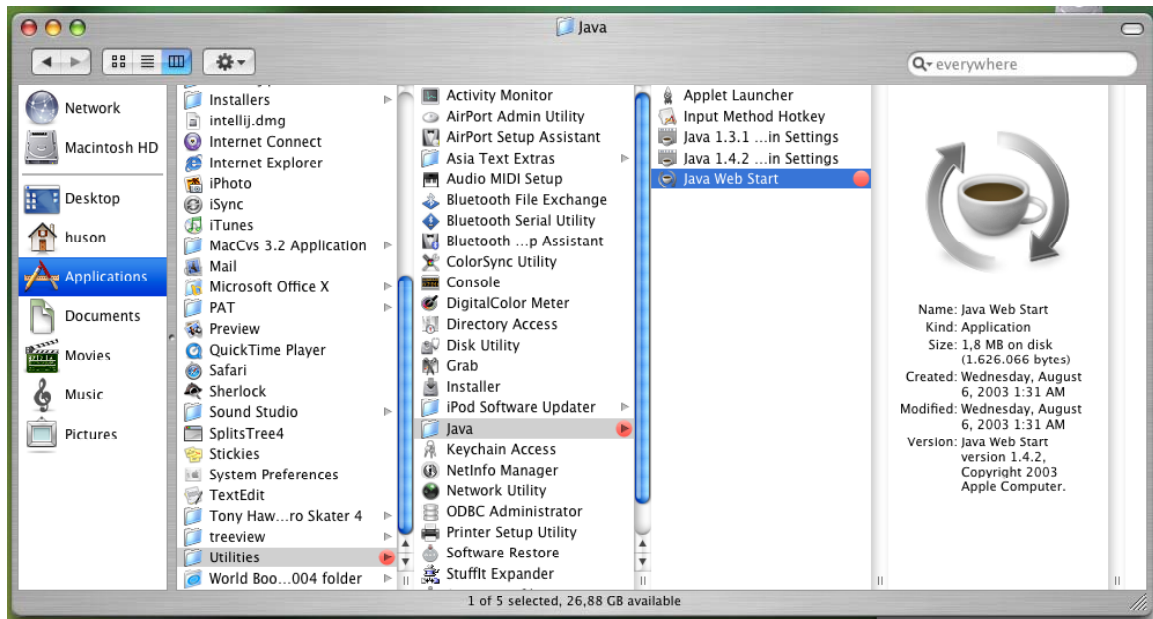


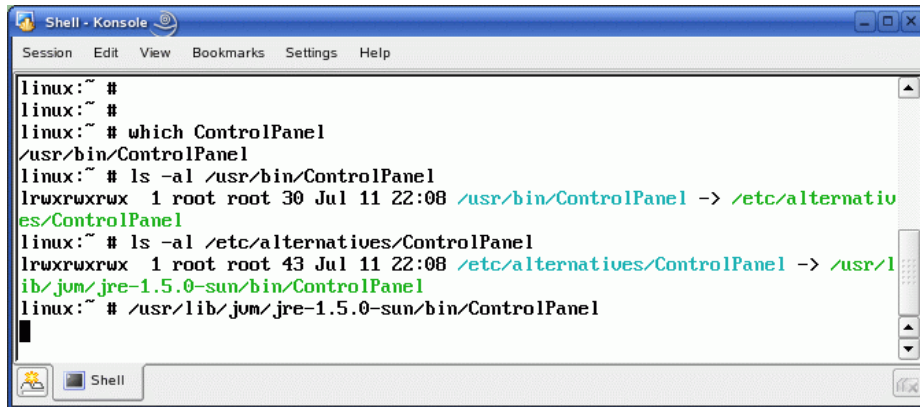
Figure 12 – Find the Java Control Panel here



Figure 13 – The Java Control Panel

Linux

You may need to search for the Java Control Panel in a similar manner as for the “javaws” application. Both should reside in the same directory.



```
linux:~ #  
linux:~ #  
linux:~ # which ControlPanel  
/usr/bin/ControlPanel  
linux:~ # ls -al /usr/bin/ControlPanel  
lrwxrwxrwx 1 root root 30 Jul 11 22:08 /usr/bin/ControlPanel -> /etc/alternativ  
es/ControlPanel  
linux:~ # ls -al /etc/alternatives/ControlPanel  
lrwxrwxrwx 1 root root 43 Jul 11 22:08 /etc/alternatives/ControlPanel -> /usr/l  
ib/jum/jre-1.5.0-sun/bin/ControlPanel  
linux:~ # /usr/lib/jum/jre-1.5.0-sun/bin/ControlPanel
```

Figure 14 – Searching for the Java Control Panel (“ControlPanel”) application

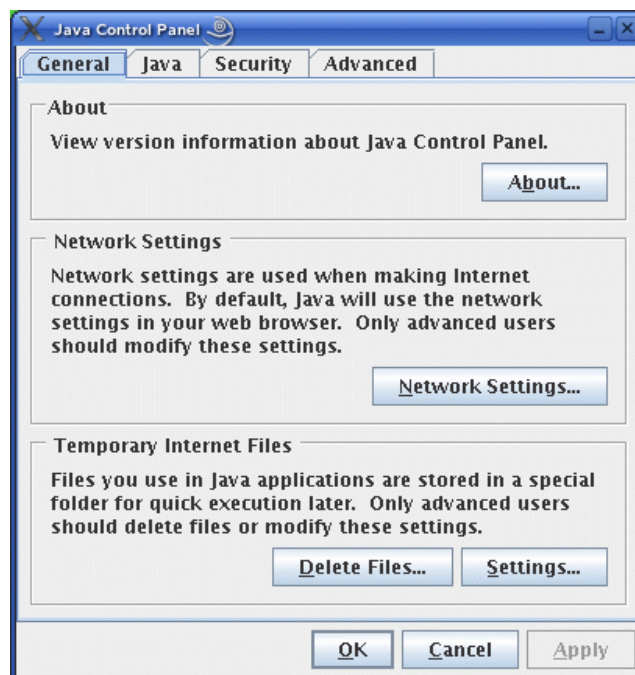


Figure 15 – The Java Control Panel

Launching the Java Console

The Java Console is a program that displays detailed output of a Java application that is normally not visible. In case of trouble with e.g. a Java Web Start application it may be useful to have a glimpse at the Java Console.

In this section, we'll have a look at

- ways to start the Java Console.

Windows

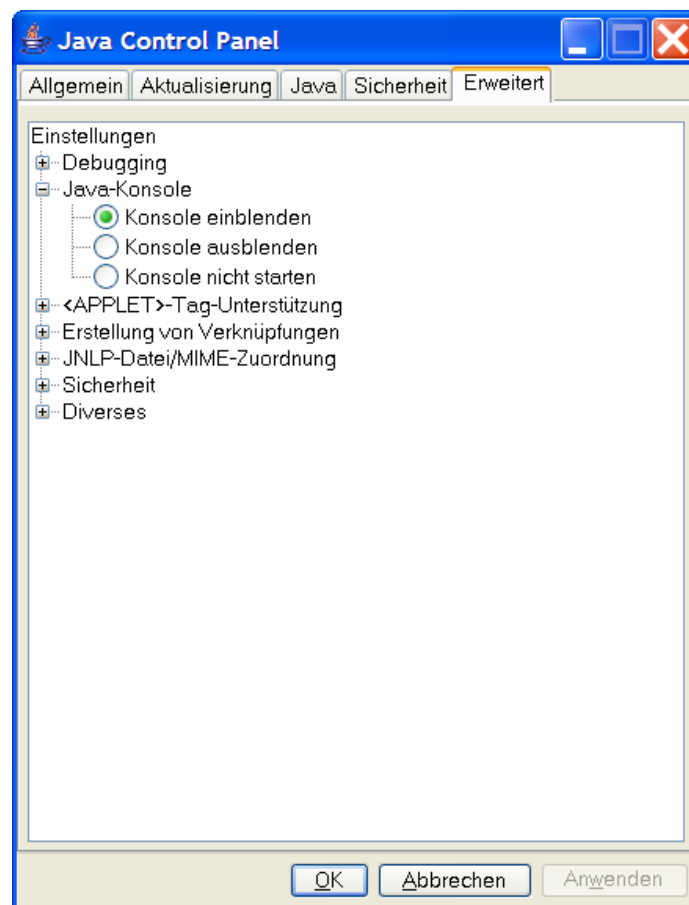


Figure 16 – In the Java Control Panel, set the Java Console on “always visible”

And it will appear when a Java Web Start application is started the next time:

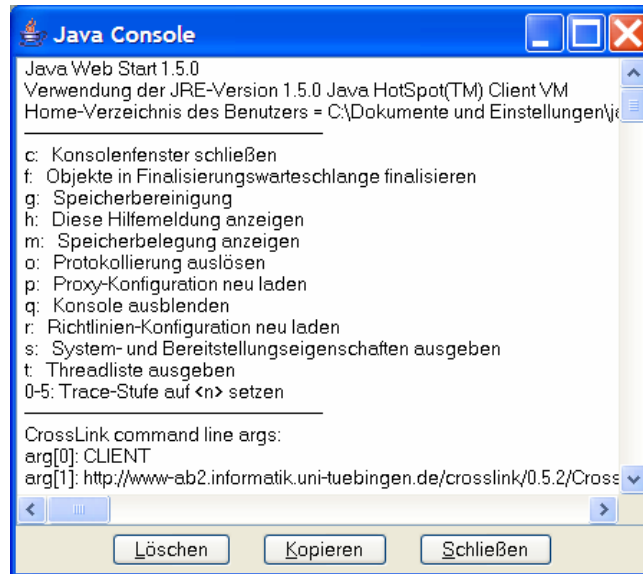


Figure 17 – The Java Console

Mac

You can enable display of the Java Console in the Java Control Panel settings:



Figure 18 – Enabling the Java Console

Linux

You can enable display of the Java Console in the Java Control Panel settings:

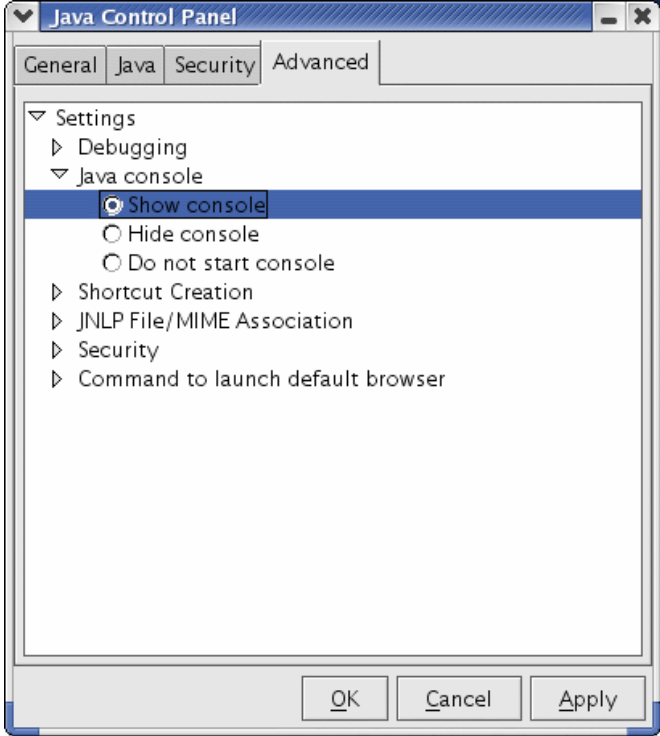


Figure 19 – Enabling the Java Console

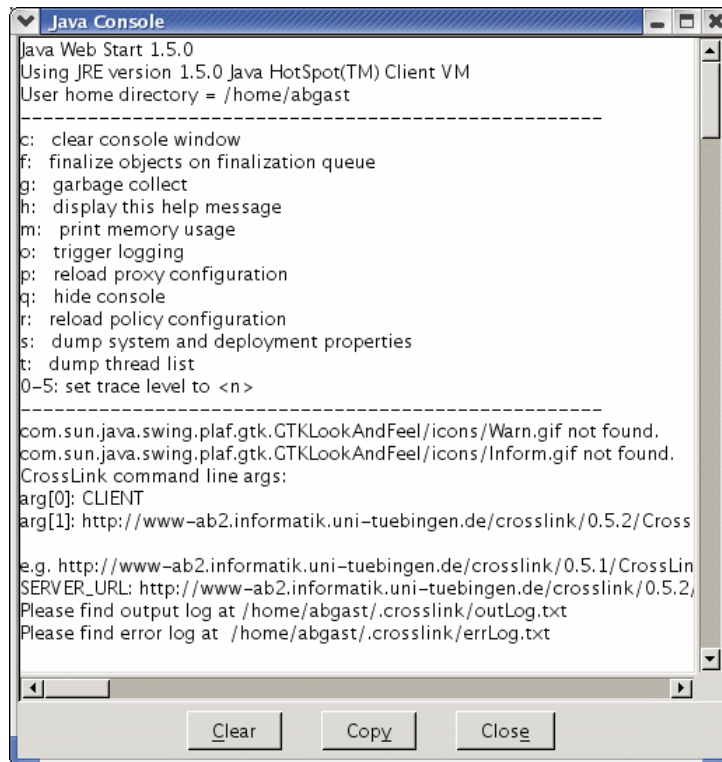


Figure 20 – The Java Console

About Java Web Start technology

In short, Java Web Start is an application distribution technology that allows semi-automatic on-demand download of Java applications to client computers using a local cache. The first time a Java Web Start application link is clicked, the application is downloaded and executed. Every next time, one of these actions are performed:

- if there is no internet connection, the application is started from the local cache
- if there is an internet connection, the version number of the application copy on the net and in the cache are compared and the most recent one is started

You can have a look into this local cache by opening the Java Web Start application. Click on the Java Web Start icon or type “javaws” on the command line (all operating systems).

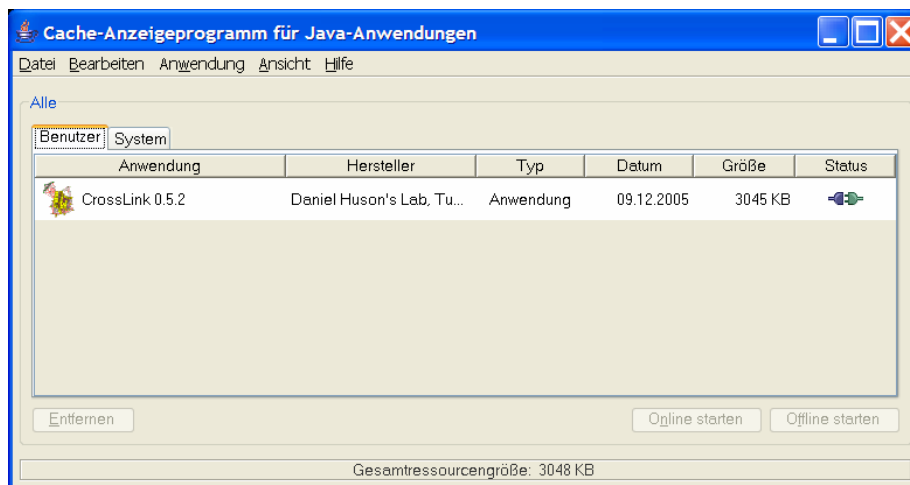


Figure 21 – Peeking into the local Java Web Start cache

Please find more information here:

All about Java Web Start technology

<http://java.sun.com/products/javawebstart/>

Java Web Start FAQ

<http://java.sun.com/products/javawebstart/faq.html>

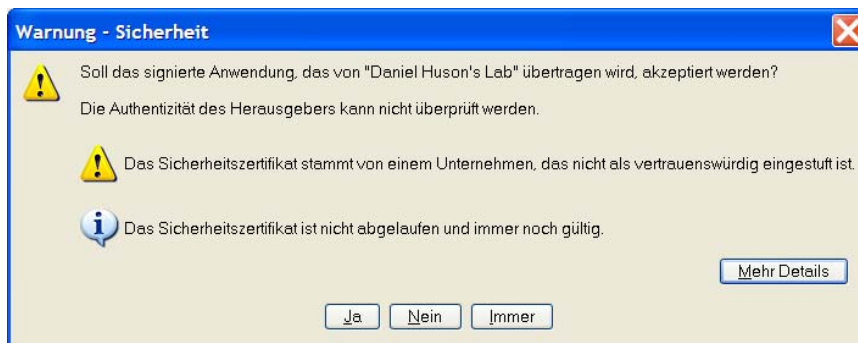
The unofficial Java Web Start FAQs (great!)

<http://webstartfaq.com/>

Security issues

Java Web Start will ask you whether you trust the downloaded application that is about to start. This is needed to allow the application to reach out of its “sandbox” and e.g. access files on your local hard disk. The potential for harm of such an application is identical to any executable file. Java Web Start assures that the application will not run if you do not trust it.

Windows



Mac



Linux

