

Protein Music

By Ross King, Colin Angus and Andreas Karwath

Introduction

Displaying molecular structures by the use of computer graphics is a common task for analysing protein structures. Such techniques are well established, e.g.: RasMol (<http://www.ch.ic.ac.uk/rasmol.html>).

Lesser known and used are techniques using other than the human visual sense, like the senses of touch or hearing. After vision, hearing conveys the most amount of information in a human. The use of sound to convey information is known as “audification”. ProteinMusic was a development of research into protein sequence audification. The original program (King, R.D. & Angus, C.G. (1996) PM: Protein Music. CABIOS. 12, 251-252.) was written by Ross King and Colin Angus in the programming language C. The program ran on an Apple Mac computer with a MIDI connection to a synthesizer, and enabled the direct creation of music (in real time) from input protein encoding DNA sequences. The problem with this set-up was that it was not easily transferable to other computers. Nevertheless, one song based on the PM protein/music mapping, S2 Translation, was a commercial success on a CD released by the Shamen see: <http://www.nemeton.com/axis-mutatis/s2.html>.

ProteinMusic is a complete and extended re-write of the original program in Java, allowing it to run on more or less all computers, ranging from Windows based PCs, over Apple Macs to big SUN workstations. The program provides the ability to read three different DNA sequence formats, to convert a DNA sequence into a MIDI file using its unique mapping, to play the MIDI music, and finally to save it for later use for other programs. Furthermore, the program provides a graphical user interface for all these tasks, as well as having the ability of changing tempo and timbre of the generated music.

The approach used for this program was to use the new advanced features of the Java programming language. In mid 2000, SUN released the new JDK1.3, including a MIDI library, featuring everything needed to create and play MIDI style music. This enabled the development of a DNA to music conversion program to run on almost all commercially available computers.

The mapping

The key idea behind the conversion from genomic data to music is to find a good mapping between chemical sequence and musical sequence. This is achieved in PM by taking advantage of different characteristics of the 20 amino acid residues. The program uses the nucleotide mapping for the top line, and the properties of the coded amino acids as bass line. Both lines are played simultaneously. For full details of the algorithm see King & Angus (1996).

Top line mapping (from nucleotides)

Nucleotide	Musical Note
A	A4
T	E3
G	C3
C	G3

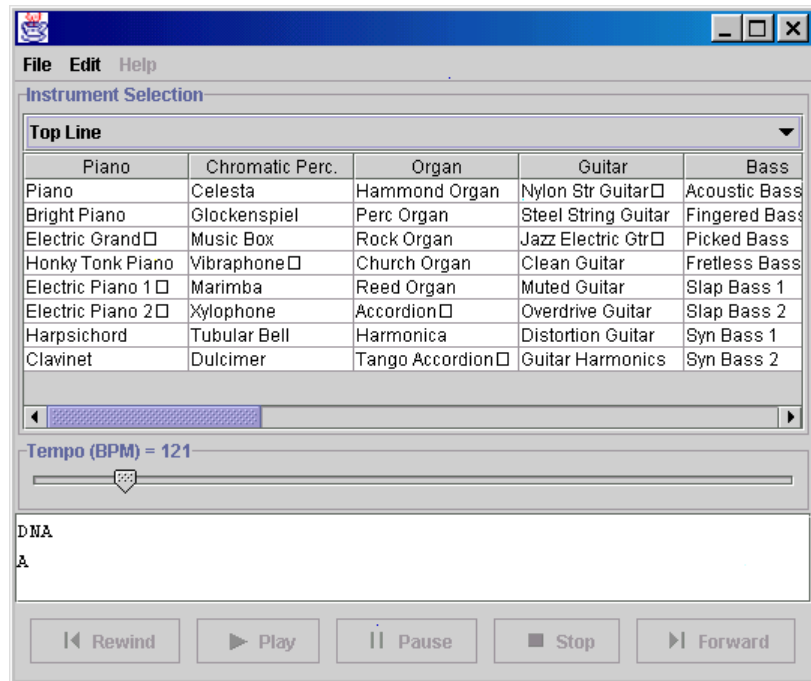
Bass line mapping (from the amino acid characteristics)

Characteristic	Musical Note
Polar	A2
hydrophobic	C1
charged	F1
positive	E1
aliphatic	G1
aromatic	D1

Amino acids have different sizes; these are normally divided into three categories: tiny, normal and large. So an extra mapping for the property *size* is introduced; if an amino acid is tiny, all notes from the two mappings are changed to one higher octave. To give an example: is an amino acid polar and tiny, the mapping for this amino acid will be A3, instead of A2. If the amino acid is normal or large, no octave change is undertaken.

The program

Here is a screen shot of ProteinMusic after the start:



The user interface is divided into two main sections, the menu bar and the play back canvas. The menu bar options are responsible for file input and output, while the play back canvas is used to alter tempo and timbre of the converted sequence as well controlling the play back of the music

The ProteinMusic menu bar

The program bar is divided into two categories: File and Edit. File handles the input and output of files while Edit handles the pasting of DNA sequences by hand.

File – Open Project

Opens a DNA file. The file has to be of one of the following types:

1. FASTA

FASTA is one of the most commonly used primary sequence file types. This might be due to its simplicity. It requires a header line starting with a greater sign (>), followed by a description. The description is usually ignored. Following this header line are multiple lines containing the pure DNA sequence using the four-letter alphabet (a, t, g, c). An example would be:

```
> some fasta sequence
acttctttgagctcaactacgaactccctaataatgcaattaaatgatgacaccaggctctac
agtaatgactttaactccggagaagctaacacttctgatgcatttaactggacagtcgac
```

...

2. GenBank

The GenBank format was introduced by the National Center for Biotechnology Information (NCBI). This format contains far more information about the sequence than FASTA. It supplies information of loci, genomes, number of base pairs, etc. The general structure is like the following: the first 12 characters of each line are reserved for the attribute names, while position 13 to 80 contains the actual data. Each entry is ended with a double forward slash (//). The sub-structure of the format varies, depending on the entries. For a full description, please see the file <ftp://ncbi.nlm.nih.gov/genbank/gbrel.txt>.

3. EMBL

The EMBL (European Molecular Biology Laboratory) format is similar to the GenBank format. A full description can be found here: http://www.ebi.ac.uk/embl/Documentation/User_manual/format.html.

When this menu item is activated a file dialog box appears, allowing the user to choose the desired file. By either double clicking on the file name, or single clicking on the file name followed by Open select the file, loads and converts the sequence into a MIDI song.

File – Close Project

Closes the current project and resets ProteinMusic. The same can be achieved by loading in a new sequence.

File – Save MIDI File

Saves the MIDI sequence for later use by other MIDI software like MIDI players. The file name stays the same; only the file extension is changed to *.mid*.

File – Save MIDI File As

Has the same function as File – Save MIDI File, but allows the user to choose the filename and the file extension.

File – Exit

Finishes the program and return to the command line.

Edit – Paste Sequence

Allows the user to paste or write a sequence without using the File – Open Project menu item. This can be useful for cut and paste cases from web sites. The sequence has to be in one of the allowed file formats (FASTA, GenBank, or EMBL). This option opens a dialog box for pasting or writing the sequence. The sequence is only set and converted when the sequence is confirmed by the Okay button and of valid type.

Edit – Set Primary Mapping (not implemented yet!)

This option will allow the user to change the initial DNA to musical note mapping for the top line.

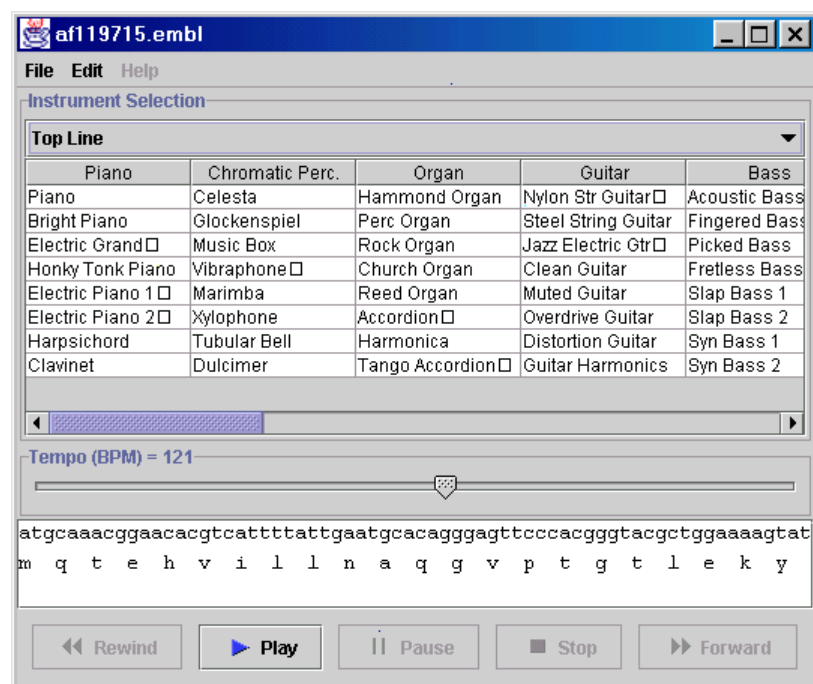
Edit – Set Secondary Mapping (not implemented yet!)

This option will allow the user to change the initial amino acid characteristics to musical note mapping (bass line).

Help (not implemented yet!)

The ProteinMusic play back canvas

The play back canvas is the main instrument of modifying and playing a converted DNA sequence. It consists of an instrument selection, a tempo changer, a sequence display and a play back control.



Play Back Canvas – Instrument selection

The user can select with what instrument the sequence should be played, by selecting an instrument from the table. This is done by simply clicking on the desired instrument name in the table. As there are 128 instruments to choose from, it is sometimes necessary to slide the table to the right or left, until the right instrument appears. The top line/ bass line menu gives the option to choose which of the to separate mappings the new selection should be applied to.

Play Back Canvas – Tempo Slider

The tempo slider gives the option to change the tempo of the play back. By selecting the slider, while holding the mouse button pressed, moving to the left decreases the speed, and moving to the right increases the speed of the play back. The tempo is measured in beats-per-minute (BMP). The tempo can only be changed when the sequence is not played.

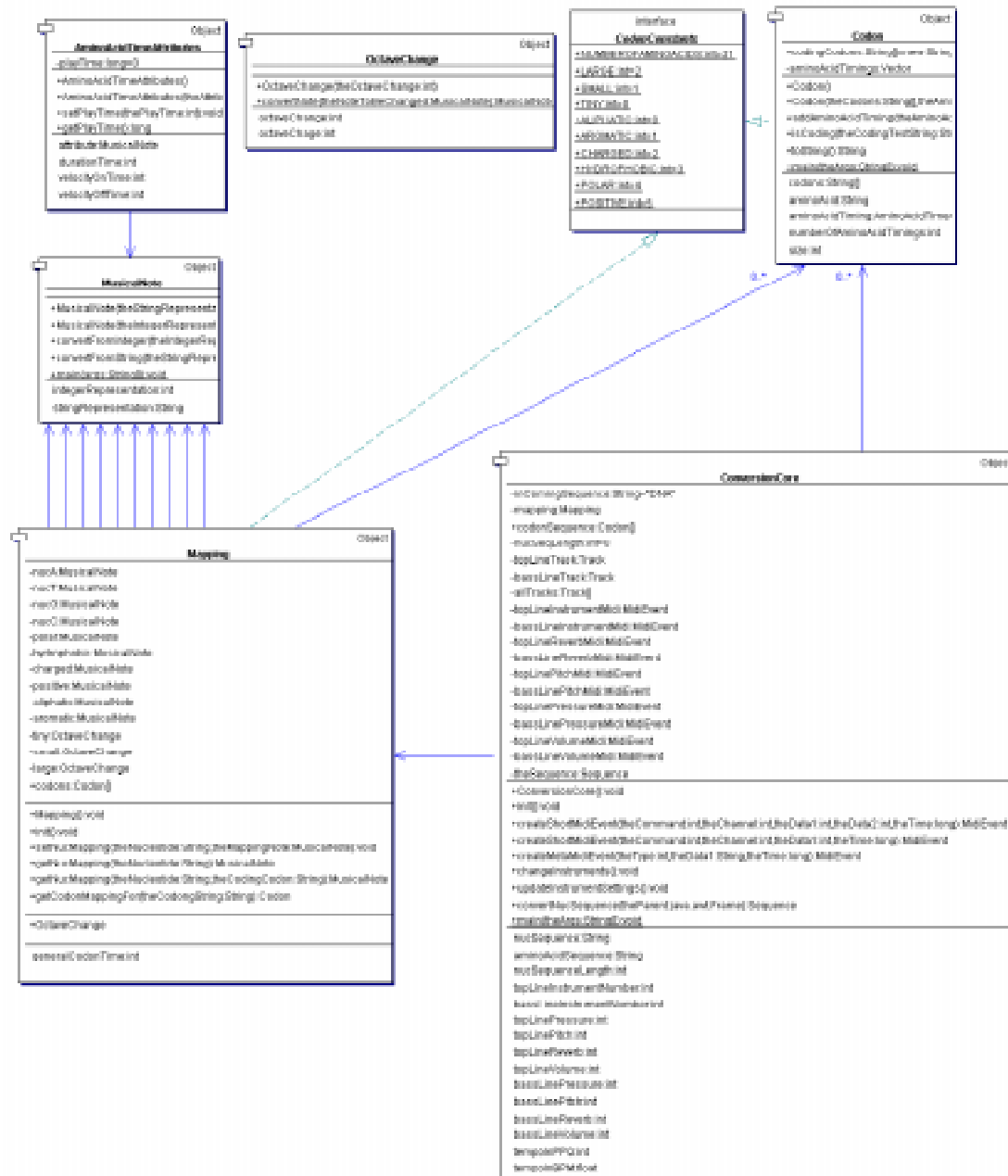
Play Back Canvas – Play Back Control

The play back control gives the option to start, pause, or stop the playing of the converted sequence. At the beginning, after loading in a new sequence it is only possible to start the music by pressing the PLAY button. After the music started, it is possible to stop the play back or pause it. If the STOP button was pressed, one can start the playing again by pressing the PLAY button, if the PAUSE button was pressed another press on the button will again activate the play back. During play back, it is possible to jump forward or backward in the sequence. The FORWARD or REWIND buttons will change the play back of the sequence either 9 nucleotides before or after the current position. If the sequence was at its end, pressing the FORWARD button will reset the play back to its initial position.

Play Back Canvas – Sequence Display

This is a scrolling window, displaying the actual position played on the left hand side of the window. The top line shows the original DNA sequence and the bottom line the amino acids the sequence DNA codes for.

The program consists of 13 separate classes, some of them, the core conversion classes will be discussed here. The following figure shows the UML diagram of those classes.



Codon.java

This class is a container class storing the details of which codons are coding for which amino acid, as well as keeping codon specific data like the timing information for the amino acid characteristics, or the size of the amino the codon is coding. It allows the usual setting and getting of the information and does require the coding codons in

form of a string array and the resulting amino acid as a string in the constructor. It also implements the CodonConstants interface.

CodonConstants.java

This interface stores the constants used for the Codon.java class. It describes the number of amino acids, the size of an amino acid as well as the other characteristics like aliphatic, aromatic, charged, hydrophobic, polar, and positive.

AminoAcidTimeAttributes.java

This class stores and handles information concerning the timing attributes for individual amino acid characteristics. The overall mapping of the amino acid characteristics is not always at the same time index. So although different amino acids might have the same or similar characteristics, the resulting musical notes can be played in a different order. Hence, this class provides a container for keeping the note to be played, the time index it has to be activated as well as duration, velocity-on, and velocity-off time. The mapping cannot be changed in this version of the program, but is designed to be changeable for future releases.

Mapping.java

This class is one of the core classes of the program. It implements the general mapping between the nucleotides and the codons (or amino acids) into musical notes. All the information like the mapping of nucleotides or amino acids to musical notes is hard coded into the class, as well as the different timing information for the same characteristics in different amino acids. Future releases ProteinMusic will have the same set up, but will be able to load and store this kind of mapping information as well as providing a graphical user interface for this kind of modification.

ConversionCore.java

This class is the main conversion class for the ProteinMusic program. It converts a DNA sequence into a MIDI music sequence. The class keeps track of all timbre and tempo settings. A ConversionCore object can be instantiated without any parameters. However, it is necessary to set a DNA sequence before converting it. If this is not done, the conversion method does return a null sequence. The MIDI sequence is organised into three separate MIDI tracks. The first track is the so-called top line track, containing the generated sequence from the nucleotides; the second track, the bass line track, containing the MIDI information for the notes generated from the amino acid characteristics. The third track contains the selected instrument information for the other two tracks.

Selected methods of the ConversionCore class:

```
public void setNucSequence(String theNucSequence)
```

This method expects a String object as parameter containing the DNA sequence consisting of four nucleotide letters and which has to be dividable by 3. It sets the necessary attributes and creates a new Codon class with this information.

public String getNucSequence()

If a nucleotide sequence is set, this method returns the nucleotide sequence in form of a String object.

public String getAminoAcidSequence()

If a nucleotide sequence is set, this method returns the amino acid sequence, generated after the convertNucSequence method call, in form a String object.

public int getNucSequenceLength()

This method returns the length of the nucleotide sequence, this might be necessary for other classes like display objects.

public void setTopLineInstrumentNumber(int theInstrumentNumber)
public int getTopLineInstrumentNumber()

These two methods set and get the top line instrument numbers. A previous knowledge of what instruments are available is not strictly necessary, however helpful. The instrument number must lie between 0 and 127. The same method is available for the bass line instrument settings.

public void updateInstrumentSettings()

This method updates the instrument settings from before, in case an instrument change is performed while the sequence is already played.

private MidiEvent createShortMidiEvent(int theCommand, int theChannel, int theData1, int theData2, long theTime)
private MidiEvent createShortMidiEvent(int theCommand, int theChannel, int theData1, long theTime)

These two methods create a MIDI event containing a short MIDI message. Short MIDI messages are commands to the synthesizer, like switching a note on or off, different reverb settings and so on. As input, this method expects the MIDI command (*int*), the channel (*int*), the data necessary for the command with either one or two *ints*, and finally the time this command should be executed in the synthesizer. If the event cannot be created, an exception is caught and a null object returned.

private MidiEvent createMetaMidiEvent(int theType, String theData1, long theTime)

This method creates a MIDI event containing a MIDI Meta message. Meta MIDI messages are commands usually ignored by the synthesizer. They are used for general information such as lyrics. The methods expects the user defined command of type *meta* in form of an *int*, the data as String and the time this information should be stored in the MIDI file. If the event cannot be created, an exception is caught and a null object returned.

```
public void setTempoInBPM(float theTempoInBPM)  
public float getTempoInBPM()
```

These methods set and get respectively the tempo of the music in beats per minute.

```
public Sequence convertNucSequence()
```

This is the main conversion method for the nucleotide sequence. First, this method creates the necessary MIDI sequence containing three tracks: one track for the top line, one track for the bass line and another one for the instrument settings and timing information. It then sequentially traverses the DNA sequence and uses the *Mapping* object for converting the nucleotides and the amino acids (codons) into MIDI formatted music. It also adds *Meta* information, which can be used later by other objects implementing a *meta()* method, for real time displaying of the sequence.

MusicalNote.java

This class stores a musical note and supplies method for converting a musical note from int to String and vice versa. This class is used by the Mapping .java class. Strictly speaking, this would not be necessary. However, it is nicer to represent musical notes in String form, and when dealing with the MIDI sequence, use the integer representation. A musical note object can be instantiated using either the string or the integer representation.

Selected methods:

```
public int getIntegerRepresentation()  
public void setIntegerRepresentation(int theIntegerRepresentation)  
public String getStringRepresentation()  
public void setStringRepresentation(String theStringRepresentation)
```

All four methods do the obvious setting or getting of both different types of representations. The two setting methods use the following two methods respectively:

```
public String convertFromInteger(int theIntegerRepresentation)  
public int convertFromString(String theStringRepresentation)
```

Both method use simple switch statement to convert from and to the following, while using the mod function for the different octaves.

Integer	String
0	C
2	D
4	E
5	F
7	G
9	A
11	B

The method cannot convert half notes.

OctaveChange.java

The OctaveChange class is specifically set up to supply support for the amino acid characteristic of size. As described earlier, the original mapping for a codon changes if the amino acid it codes for is tiny. All notes generated for this codon will be mapped one octave higher as the usual mapping. This class supports this size dependent mapping. To convert a MusicalNote object the method *public MusicalNote convertNote(MusicalNote theNoteToBeChanged)* has to be called, returning a new MusicalNote object.

All other classes are a wrapping for these core classes and provide the graphical user interface, described at the beginning of the document. ProteinMusic is provided as compiled program including the source code, so users can change and extend the program to their will.

A. Karwath (January 2001)