Journal of Computer-Aided Molecular Design, **13:** 625–643, 1999.
KLUWER/ESCOM
© 1999 *Kluwer Academic Publishers. Printed in the Netherlands.*

Visualisation and integration of G protein-coupled receptor related information help the modelling: Description and applications of the Viseur program

F. Campagne[a,*], R. Jestin[b], J.L. Reversat[b], J.-M. Bernassau[b] & B. Maigret[a]
[a]*Laboratoire de Chimie théorique, Centre Charles Hermite, Unité Mixte de Recherche 7565, CNRS-Université Henri Poincaré Nancy I, BP 239, F-54506 Vandoeuvre-les-Nancy, France*
[b]*Sanofi Recherche, Service de Chimie Structurale, 371, Rue du Pr. Blayac, F-34184 Montpellier cedex, France*

Received 17 February 1997; Accepted 3 March 1998

Key words: alignment, computer program, GPCR, HTML, integral membrane proteins, mutagenesis, tool, VRML

Summary

G Protein-Coupled Receptors (GPCRs) constitute a superfamily of receptors that forms an important therapeutic target. The number of known GPCR sequences and related information increases rapidly. For these reasons, we are developing the Viseur program to integrate the available information related to GPCRs. The Viseur program allows one to interactively visualise and/or modify the sequences, transmembrane areas, alignments, models and results of mutagenesis experiments in an integrated environment. This integration increases the ease of modelling GPCRs: visualisation and manipulation improvements enable easier databank interrogation and interpretation. Unique program features include: (i) automatic construction of 'Snake-like' diagrams or hyperlinked GPCR molecular models to HTML or VRML and (ii) automatic access to a mutagenesis data server through the Internet. The novel algorithms or methods involved are presented, followed by the overall complementary features of the program. Finally, we present two applications of the program: (i) an automatic construction of GPCR snake-like diagrams for the GPCRDB WWW server, and (ii) a preparation of the modelling of the 5HT receptor subtypes. The interest of the direct access to mutagenesis results through an alignment and a molecular model are discussed. The Viseur program, which runs on SGI workstations, is freely available and can be used for preparing the modelling of integral membrane proteins or as an alignment editor tool.

Introduction

G Protein-Coupled Receptors (GPCRs) are membrane proteins involved in one of the major cellular signaling systems: from HIV infection and blood pressure regulation to taste, smell and colour recognition. These receptors transmit into the cell the signal brought from the extracellular medium by a hormone, molecule, or photon. Since GPCRs are involved in most biological regulation systems, an increasing activity is devoted to this receptor super-family and GPCRs are an interesting target for the discovery of new drugs of pharmaceutical importance. Structurally speaking, all GPCRs share an extracellular N-terminal part, a trans-membrane seven-helix bundle and a C-terminal intracellular part. Such properties suggest a structural homology between these receptors that, again, increases their interest.

The most frequent data obtained from experiments on these membrane proteins are

- receptor sequences derived from gene cloning,
- agonist and/or antagonist ligand binding and/or transduction properties obtained from bioassays, and
- mutagenesis information obtained from site-directed experiments.

*To whom correspondence should be addressed at: Department of Physiology and Biophysics, Box 1218, Mount Sinai School of Medicine, One Gustave L. Levy Place, New York, NY 10029-6574, U.S.A. E-mail: campagne@inka.mssm.edu

The last two sets of data show how the receptor expression or the binding or activation by a ligand are modified when one changes the receptor sequence. These data are becoming more and more useful as their number increases rapidly. A review of GPCR mutations is given in Reference 1. Important sources for GPCR mutations are provided in References 2 and 3.

Due to the lack of precise three-dimensional experimental data for this receptor family (the only GPCR for which preliminary structural information is available is rhodopsin [4]), GPCR models are usually obtained by homology modelling, using the only available three-dimensional structure, of an acceptable resolution, currently available for a protein showing a seven-helix bundle. The structure of bacteriorhodopsin (which is not a GPCR, but could be a far common ancestor of the family members) is known at the atomic level of detail with a 3.5 Å resolution [5] and is usually used as a three-dimensional template for modelling. In addition to the three-dimensional structure of the template, homology modelling also requires the establishment of a sequence alignment between the sequence template and the protein to be built. Such an alignment between bacteriorhodopsin and any GPCR is not straightforward because the sequence homology between them is poor (less than 20%). This means that common sequence homology-based alignment methods cannot be employed. The alignment methods used are, therefore, very specific to the GPCR family [6]. Once an alignment is obtained between the receptor of interest and the template, a three-dimensional structure for the transmembrane part of the studied receptor can be built. Extra- and intra-cellular loops connecting the transmembrane helices can be added to the incomplete structure at this step or at a latter stage in the modelling process by a special procedure, as their treatment cannot rely on alignment techniques. Next, the helices can be rotated or moved along their axes to maximise the agreement between the structure and the experimental results known at that time.

To understand why one can, and should, adapt the model structure in this way, one has to consider the following points.

1. The template resolution is weak.
2. We know that the template is only indicative: bacteriorhodopsin is not a GPCR and its electron density projection map differs significantly from that of rhodopsin, so that the transmembrane helix axes are only roughly indicative of the relative position of these fragments inside the membrane bilayer.
3. The proline residues known to break the helical arrangement of residues are distributed among the transmembrane segments in different distributions for bacteriorhodopsin and GPCRs [7]. According to these elements, some models of rhodopsin were constructed in order to be used as templates for the modelling of GPCRs, according to various experimental data. The latter arguments to helices rotating around their axes apply also to this model.
4. The molecular modelling methods presently applicable to such molecular systems, namely minimisation, molecular dynamics or simulated annealing, can only approach the energy of the isolated receptor without any explicit consideration of the effects of the surrounding environment (i.e. phospholipids, water and ions are not included). However, some experimental studies support the hypothesis that interactions between the helices are more important to the structure of the bundle than interactions between helices and the surrounding environment. This is in agreement with the observed stability of modelled bundles using molecular dynamics in vacuum, in which the helices remain packed during the whole simulation, and with experiments in which truncated receptor moieties fold together. But, even according to this hypothesis, the simulation time for these systems, which consume a reasonable amount of computer time (one or two weeks), is currently limited to only one or two nanoseconds. This precludes the bundle from undergoing important rearrangements during the simulations. The helices will remain more or less in their original relative orientations. In the case of molecular dynamics, the system size limits the simulation to a very small period of simulated time, far away from the time scale necessary to obtain important rearrangements of the receptor. It seems reasonable, therefore, to bias the conformational search according to the current experimental knowledge in order to approach, as closely as possible, the interesting conformational sub-space where the agreement with experiment is satisfactory.

This introduction to GPCR modelling is far from complete, since some attempts have already been proposed to obtain 3D GPCR models from automatic procedures. Such possibilities to retrieve 3D GPCR models are available on the WWW from the EMBL [3] or Swiss Model [8] sites. The associated procedures oblige one to work blind, as no control on the process itself is possible by the end-user. More sophisticated procedures, associating molecular dynamics refinements of the raw 3D GPCR structure obtained from homology modelling have been proposed [9]. Most of the above approaches, however, do not take directly and explicitly into account, in the simulation process, the many available experimental data about the receptors and their ligands.

An attempt to introduce such a control from experimental data into the modelling process was described recently [10] but, in the proposed algorithm, the user has little possibilities to influence the different steps of the process. Furthermore, we have to mention that GPCR modelling is not a sequential process. It may happen that looking at the receptor alignment and the mutant information, one might notice inconsistencies that will have to be fixed. This confrontation of information could happen at different stages during the modelling process: (i) during the alignment procedure while looking at a snake-like diagram of the sequences, based on assumptions on the nature of side chains related to their environment, and (ii) when building the model by checking its consistency with mutational information.

It appears, therefore, that GPCR modelling is a complex task which implies taking care of large amounts of heterogeneous data that are highly interrelated. For this reason, we are developing the Viseur program which is designed to store, manage and visualise the GPCR-related data and to produce a raw 3D model. It enables one to build and use an in-house GPCR data bank containing available information and personal notes on a subset of GPCRs (we prefer to use the expression 'data bank', as opposed to 'database', to avoid confusion because the latter have a very precise meaning in the computer field).

In the first part of this paper, we describe the novel algorithms and methods implemented in the Viseur program. The second part describes the overall program features, the use of which is illustrated in the third part by the creation of GPCRDB snake-like diagrams, a preparation to the modelling of the serotoninergic receptors (5HT) and a reflection about the interest of browsing mutant data through an alignment.

Methods

Automatic snake-like diagram construction

A snake-like diagram is a schematic representation of a transmembrane protein sequence. The data needed to construct this representation are: the protein sequence, the number of TM segments and their transmembrane limits. These limits will be notated as TMnb for the position of the first residue in the transmembrane region n and TMne for the last residue of the same TM. According to this notation, the sequence could be divided into subsequences, each one associated with one of the `EXTREMITY`, `TM` or `LOOP` graphical representations. The extra integer number, which follows this representation mode, distinguishes between the direction of the drawing: -1 means that the sequence is drawn from top to bottom (at least for the first part in loops), while 1 means the opposite direction. The subsequence division based on the TM limits is given in Table 1.

Table 1. Limits and directions of drawing of the sub-sequence decomposition of a GPCR for the snake-like algorithm

Sub-sequence	Begins at	Ends at	Representation
N-terminal	1	TM1b$-$1	EXTREMITY -1
TM1	TM1b	TM1e	TM 1
IL1	TM1e+1	TM2b$-$1	LOOP 1
TM2	TM2b	TM2e	TM -1
EL1	TM2e+1	TM3b$-$1	LOOP -1
TM3	TM3b	TM3e	TM 1
IL2	TM3e+1	TM4e+1	LOOP 1
TM4	TM4b	TM4e	TM -1
...	...	...	...
IL(n/2+1)	TM(n$-$1)e+1	TMnb -1	LOOP $(-1)^{\frac{n}{2}+1}$
TMn	TMnb	TMne	TM $(-1)^{n+1}$
EL(n/2)	TMne+1	TM(n+1)b$-$1	LOOP $(-1)^{\frac{n}{2}}$
...	...	...	...
C-terminal	TM(nmax)e+1	Seq length	EXTREMITY 1

The algorithm uses the following functions:

- **draw_residue(position,residue)**: draw a residue at a given position with the given colour.
- **residue_radius**: is the radius of the residue. When the residues are drawn as circles, this is the *radius* of the circle. If the residues are drawn as squares, this is the diagonal of the square.
- **sequence_length(sequence)**: gives the length of the parameter sequence.
- **get_TM_number(protein)**: the number of transmembrane segments to present.
- **TM_SPACE** is the horizontal space separating two transmembrane segments.
- **reverse(sequence)** denotes the sequence read in the reverse order.

- **TM_start(loop) (TM_end(loop))** gives the number of the TM segment before (respectively, after) the loop.
- **get_extremity_size(subsequence)** is the size of the drawing of the extremity.
- **shorter(subsequence)** is a subsequence truncated by one residue at each end.
- **direction(transmembrane_segment)** is 1 or -1 according to the direction of the TM (1 means top to bottom, see Table 1).

In addition, the following notations are used: if *p* denotes a position, then *p.x* (*p.y*) is the *x* co-ordinate of the position (respectively, *y*). If *tm* denotes a transmembrane segment, then *tm.top_left* is the position where the first residue of the TM subsequence was drawn. Using the same convention, *tm.bottom_right*, *tm.top_right* and *tm.top_left* are defined as the positions where the residue at the bottom right, top right and top left of the TM drawing were drawn, respectively.

The algorithm details are presented in Appendix D. The most difficult technical problem while drawing such pictures is to determine, a priori, the dimensions of the drawing. This is needed for a large number of output media, from the computer screen to the PostScript description for a page. Our implementation computes the maximum dimensions of the drawing using the same algorithm as for drawing, but using a *draw_residue* function that draws nothing but updates the drawing limits, according to the position parameter.

Automatic access to a mutagenesis data server through the Internet

The Viseur program was modified to take advantage of the GPCR mutant retrieval system at Trömso University [2]. The idea was that it would be more useful to visualise and access the mutant data using the dynamic visualisation tools provided by the Viseur program than from a WWW static interface. We intend to describe here the method we use to retrieve the mutant information through the Internet. The following features needed to be implemented.

1. A protocol describing precisely the interactions between the client program (Viseur) and the server (TinyGRAP).
2. A client module, that follows the protocol, to access the data through the Internet.
3. A server module to serve the data according to the protocol. The first and second features are described in this section. The third one was realised by Ø. Edvardsen et al. at Tromsö, adding features to the GRAP data bank (constructed on the WWW CGI protocol [11]).

Interaction protocol

This protocol is the result of discussions that led to an agreement between the GRAP team and our group. The protocol says that mutant data will be delivered on request, from the GRAP WWW server, to a client connecting to a special URL, through an HTML document. This URL may change in the future so we will not give it explicitly here, but just mention that it enables the encoding of the receptor for which the client module requests mutant data. The key is the SWISS-PROT accession number that perfectly identifies a receptor for Viseur and GRAP. The key is sent as a parameter through a GET CGI-BIN query. The HTML file generated by the GRAP server contains a list of items that describe the mutations, written in a strict format to simplify the parsing. For each sequence for which mutant data is requested, the server sends a list of mutants identified by the residue position, the residue type in the wild type and the residue type in the mutant. Because multiple mutations are difficult to interpret and to visualise, they are not sent by the server.

Client module

The client module is responsible for connecting to the GRAP server, sending the query, getting the reply from the server, and storing the result on the client machine for later use by the Viseur program. According to the WWW protocol choice described in ‘Interaction protocol’, and because we wanted the client module to be independent and portable to a wide set of architectures, we decided to implement it using the Java language and associated libraries. These libraries propose an easy way to connect to a WWW server and to read a document from an URL. Finally, to facilitate the re-use of the data obtained from the server by the Viseur program, the client module (FetchTinyGRAP) translates the data to Viseur script files.

Integration of mutant data to the Viseur program

Two aspects are important here: (i) we have to let the user know in some way that mutant data is available for one residue and (ii) we have to provide a way for the user to access the data as easily as possible, directly from the Viseur program.

We achieved point (i) by colouring residues on request, as follows: if mutant data exists for the residue, we colour that residue with a colour named 'mutation'. Point (ii) was obtained via an asynchronous communication with the WWW browser: the Viseur program instructs the browser to update its Viseur reserved view to the URL that describes the mutant data at Tromsö when the user requests information on one residue for which mutant data exists. We used Netscape as the browser for this implementation, as it is available by default on SGI systems, but all browsers that provide commands to redirect a running instance of themselves to a URL are usable. Redirecting the user to the original WWW server data, rather than providing a copy locally, presents the advantage that the user always accesses the real, up-to-date information.

VRML molecular models hyperlinked on residues

The Viseur program can create OpenInventor [12] files that can be translated into the VRML language. The conversion is done by the ivToVRML [13] tool. A Viseur script command may be used to insert OpenInventor or VRML objects, into the model. This way, ribbons generated from a PDB file as VRML by PDB2VRML [14] can be inserted in the GPCR Viseur Model usual view. This use of the OpenInventor standard to render pieces of a model with different programs and integrate the result in one view seems a first way of improving 3D molecular representation reuse. Another application would be the extension of hyperlinked VRML output to non-GPCR proteins.

Constructing a VRML model hyperlinked on residues requires that the VRML scene is constructed hierarchically with a WWWAnchor separator object as the top node of the drawing subgraph for each residue. This means that each VRML node that draws a part of a residue should be a son of the residue WWWAnchor node: this makes the picking work as expected. A URL is then added to the VRML anchor for each residue to link. From VRML2.0, it is possible to add parameters to the URL that redirects the URL to another frame. This feature is really useful to present the data: the 3D model and the information data are displayed both at the same time, enabling one to change the point of view while consulting the information. The Viseur implementation makes it easier to pick residues by adding cubes on the C_β. This atomic position was chosen as optimal for GPCR models because the direction between the helix axis and the C_β helps visualise the orientation of the side chains, but for general proteins the C_α position may be preferable, as it enables to highlight glycine residues as well.

From the application point of view, these VRML representations, that allow one to browse residue information in a 3D molecular representation, may allow the user to look at and evaluate a model without any specialised tool besides a VRML aware WWW browser. Whereas such browsers are becoming more and more common, regardless of the platform, one could expect a growing interest in this type of representation. We could mention the following limitation of such representations: the 3D representation of the protein is static (for instance, there is no way to change a few atoms to ball-and-stick after the representation has been constructed). This may be a sensible limitation for certain uses but the actual WWW technology does not allow one to build, from the client side, 3D models on the fly that will be rendered with correct performance. The current development of client side API for 3D may enable the development of macromolecular hyperlinked model browsers in the future.

Results

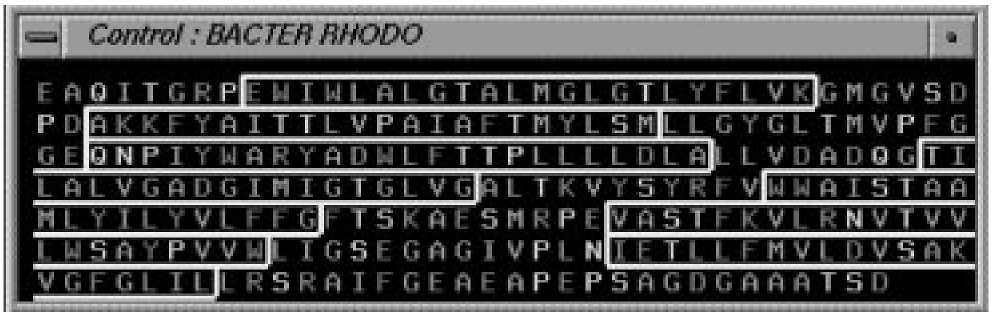
The novel features presented are implemented into the Viseur program, which proposes an integrated solution for storing, visualising and manipulating GPCR sequences, alignments or molecular models. This section will describe the features of the implementation at the time of writing.

Receptors

The program allows the user to read protein sequences. The following file formats are recognised.

- SWISS-PROT (original [15], GCG modification, and N.I.H. variant)
- SQIDE3 (CAS) Registry System [16]
- MSF (from an alignment)
- VISEUR (defined in the appendix)

The transmembrane domain limits are read from the file when it is SWISS-PROT but not SQIDE3 or MSF. The sequence view enables the user to view the receptor sequence and the associated information (clicking on one amino acid in this view with the right mouse button displays or updates the information view for this sequence). As illustrated in Figure 1, the transmembrane limits are displayed on the top of the sequence with rectangular areas. The user can move



```
EAQITGRPEWIWLALGTALMGLGTLYFLVKGMGVSD
PDAKKFYAITTLVPAIAFTMYLSMLLGYGLTMVPFG
GEQNPIYWARYADWLFTTPLLLLDLALLVDADQGTI
LALVGADGIMIGTGLVGALTKVYSYRFVWWAISTAA
MLYILYVLFFGFTSKAESMRPEVASTFKVLRNVTVV
LWSAYPVVWLIGSEGAGIVPLNIETLLFMVLDVSAK
VGFGLILLRSRAIFGEAEAPEPSAGDGAAATSD
```

Figure 1. Viseur's program sequence view for bacteriorhodopsin.

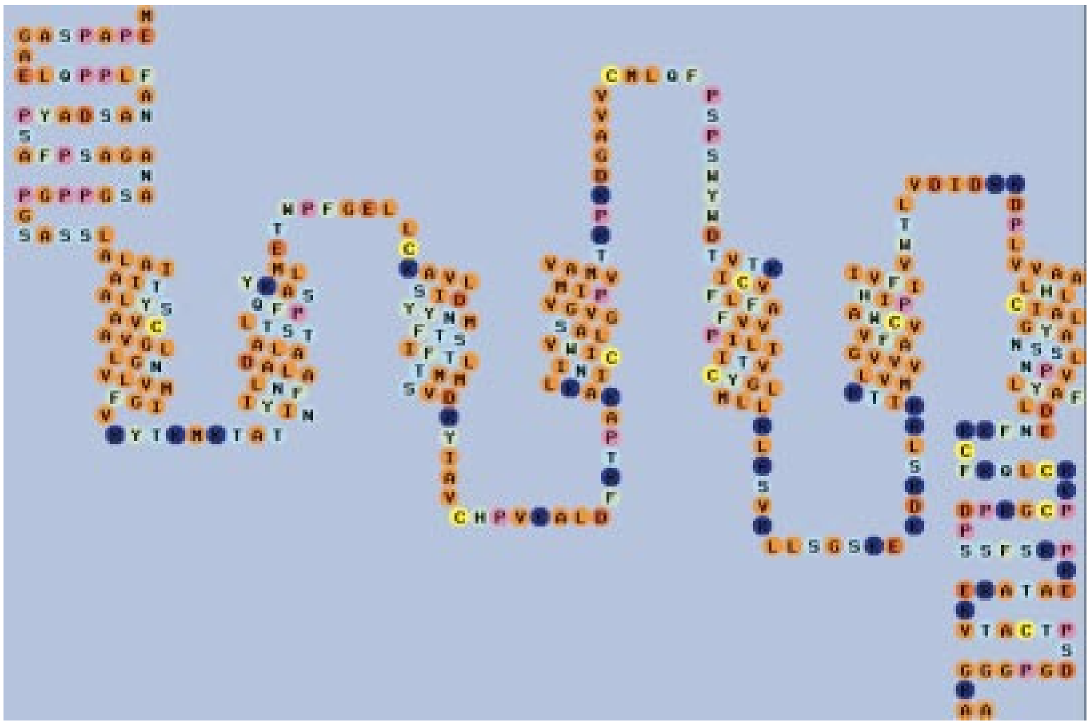


Figure 2. Viseur's snake-like view, HTML output for the OPRD_HUMAN receptor.

a TM area dragging the TM rectangle or resize a TM area by dragging the left or right side of the rectangle.

Information

An information view is updated when the user selects one amino acid from one view, whatever this is. The information view displays the following information.

- Receptor name,
- residue position,
- residue type,
- information files attached to this receptor (clicking on these buttons brings up an editor). Currently, these files contain the sequence itself when it is imported and a Viseur script that describes the mutations attached to the receptor,
- mutations: what was the residue mutated into? If there exists a mutation on the selected residue, then the Web browser is requested to display the URL associated with that mutation.

The information files are ASCII text files organised in directories. Directories are named according to the sequence the information belongs to. One file contains the original sequence file that was imported into Viseur so that no information is lost when sequences are imported. Information about amino acids is stored in files in that special directory. Such a way

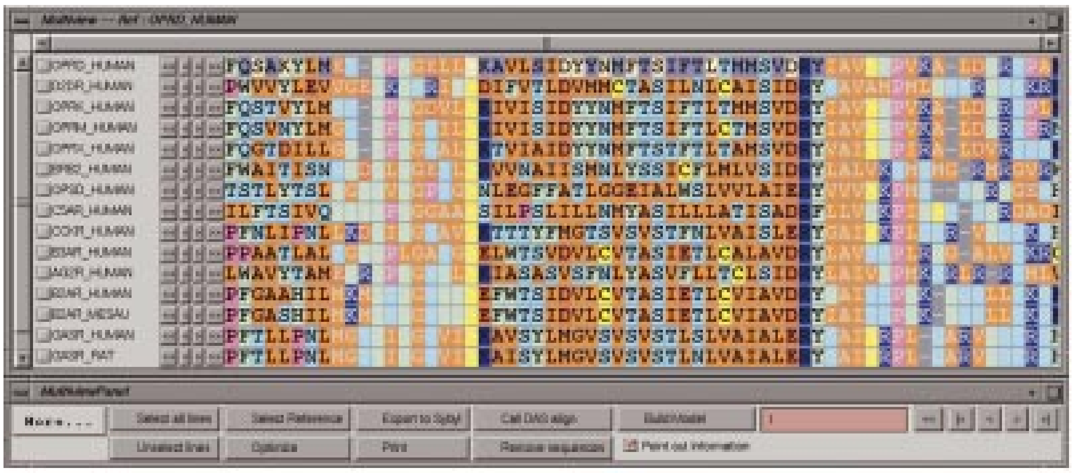


Figure 3. Viseur's alignment editor, with residues painted according to user defined convention for OPRD_HUMAN sequence.

to store data enables the use of different file formats. For example, one can think about using HTML files to describe the information attached to an amino acid.

Colouring

By default, the residues are coloured according to their types and to a user-defined association colour table. For instance, it is possible to colour residues according to their chemical properties, or according to given crystallographic conventions. It appeared from preliminary use of the program that being able to colour residues individually on the basis of their properties is a powerful feature. Viseur enables the user to colour individual amino acids manually or via the script language and to switch between a default colour visualisation mode and a user-defined colour visualisation mode (mode Point out information off/on). Colouring amino acids is available on every view: sequence, snake-like, alignment and model. The colouring can be saved or restored on request, leading to a set of coloured annotations: the user can use this feature to highlight similarities between sequences in an alignment, to point out data from mutagenesis databases, or at his convenience. We describe in Appendix C the file record format for the colouring of amino acids and the example of a script that can be useful to someone who wants to build such files with a program.

Snake-like view

The snake-like view, shown on the picture part of Figure 2, allows the user to visualise the receptor in a schematic way. The sequence is drawn according to the TM limits contained in the receptor object. A snake-like view is updated every time the helix positions are changed through the associated sequence view.

This view can be printed as Encapsulated PostScript or written as an HTML file for Web publishing. The HTML output is shown in Figure 2. The image is clickable. Using the script language one can easily build an HTML page from a SWISS-PROT file and add URL and different colour to the interesting amino acids in the sequence. This presentation is a particularly attractive and clear way of publishing GPCR mutational information on the WWW [17]. Such an application is presented in this article.

Alignments

The alignments can be read from MSF files or from Viseur alignment files. A Viseur alignment consists of a set of receptors and it describes the gaps that should be added inside the alignment to have the optimum number of matches between amino acids in all sequences. Viseur was designed to separate the receptor and the alignment concepts. An alignment should contain only the gap information and references to receptors, because the receptor information is redundant when also included in an alignment object. This is a common engineering practice that simplifies the management of data, but it is usually neglected by alignment software designers (e.g. the Multiple Sequence Files: MSF, a widely used file type). This information partition enables users to access and modify the information attached to a sequence from any alignment that contains the sequence, without breaking the information consistency.

The Viseur alignment module (Figure 3) was designed with the idea that no perfect automatic alignment program is yet available. Instead, alignments should be tuned manually to take into account all the available information. This module is a tool which makes it easier to align a lot of receptor sequences manually. The main features of the editor are:

- unlimited receptor sequences,
- receptor sequences ordering by block of sequences,
- fixed areas (column areas are constrained aligned),
- multiple gap insertion/removal,
- multiple sequences destination for gap insertion/removal,
- visualisation of transmembrane areas (full light level),
- high quality postscript output for publications,
- amino acid deletion/insertion,
- amino acid painting: to highlight information on the top of an alignment.

Moreover, the integration of the alignment module as a Viseur object enables the automatic update of amino acid colour and transmembrane area positions, direct access to the information module and construction of clickable molecular models for immediate inspection.

Models

The models are read or written in a PDB format [18]. The connectivity information is assumed to be implicitly present in PDB files: it can be derived from atom types. A model, in the Viseur program, is a GPCR 3D structure. It contains atomic positions and connectivity. It is constructed from an original model (experimental or modelled data) or from an original model and an alignment. We decided to include a molecular model visualisation module into the program to take advantage of comparisons between sequences, alignment derived properties and a 3D representation.

The program allows one to build a model from an alignment in two ways.

- The first way creates, given just the alignment of a receptor sequence with the template sequence, a Sybyl (Tripos Inc.) command file. This Sybyl SPL file can be applied (with Sybyl) to the template structure to get the new protein with the skeleton from the template and the side chains rotated to remove poor contacts.
- The second way creates a schematic model internally using the following procedure.

1. Read a template structure from a PDB file;
2. Clone the template structure;
3. Remove the loops (defined according to the template receptor transmembrane areas position);
4. For each aligned amino acid available in the template: replace the atoms of the side chain on the cloned structure by new side chain atoms. This results in a template skeleton structure carrying the side chains of the transmembrane regions of the target receptor;
5. The side chains are rotated to put the new C_β at the template C_β position. No more optimisation/changes are done;
6. The new structure is written in a PDB formatted file.

This simplistic model building method will create raw models. The main interest of such crude models is that they can be used advantageously to combine all the information available on a system or as a starting 3D arrangement for testing hypotheses and confronting with sequence oriented data.

Ribbons are drawn on the model view by means of the PDB2VRML tool. The 3D model visualisation displays different representations according to the state of the Point Out information mode. When the mode is off, this view adds cubes to any C_β, coloured with the default residue colour (described previously). When the mode is on, only those residues that were painted are highlighted by a C_β cube, coloured with the painting colour for each residue. This highlights information on the model.

Dynamic behaviour of objects

Dynamic behaviour is an important feature of the Viseur program. The user may change the attributes of objects and see the consequences immediately in each view related to that parameter. The preferences object was designed as an interface between the user and the parameters that affect Viseur's behaviour. If a parameter is changed by the user through the preferences object, it immediately affects the related Viseur behaviour. The preferences object gives a consistent interface to the Viseur static attributes (as opposed to functions): its use reduces the time spent programming the interface. This object also concentrates all the configurable attributes in one location, with a uniform interface, so that users who need to change the parameters of the program only need to look in one place. The main problems with this object are that (i) it is restricted with regard to the type of parameters it can handle and (ii) it cannot be used to apply multiparameter actions to the Viseur objects.

Applications

Hyperlinked snake-like HTML pages on the GPCRDB WWW server

The snake-like HTML output was used as an access point to mutational information on the Web for the GPCRDB server. This type of presentation is very attractive because it highlights immediately the nature of mutations: is the mutated amino acid located in the transmembrane part of the receptor or in the loops? Is it near the extracellular side or near the intracellular side? Is the amino acid in a highly lipophilic region or is it surrounded by hydrophilic residues? Following this quick inspection, the user can go into the details of a mutation by a click of the mouse on the amino acid drawing (classical hyperlink behaviour on images). The Viseur script file needed to build a snake-like HTML page from a Web server site is presented in appendix B.

Application to a modelling study of 5HT receptors

We present here a preparation of the modelling of receptors from the 5HT family. This family was chosen because the modelling of these receptors was already well studied by independent laboratories, and the results, if not the methodology, gave rise to a consensus. Our goal here is not to propose a new model of these receptors, but rather to highlight how the concepts introduced in the design of the program are useful for obtaining such a model.

Obtaining the data
Modelling of a 5HT receptor is most efficiently achieved considering the entire family. The following receptor sequences were, thus, retrieved from SWISS-PROT:

```
5HT1A_(HUMAN, RAT),
5HT1D_(CANFA, HUMAN, RABBIT, RAT)
5HT1B_(CRIGR, DIDMA, HUMAN, MOUSE, RABBIT,
       RAT),
5HT1E_HUMAN,
5HT1F_(HUMAN, MOUSE, RAT),
5HT2A_(CAVPO, CRIGR, HUMAN, MACMU, MOUSE,
       PIG, RAT),
5HT2B_(HUMAN, MOUSE, RAT),
5HT2C_(HUMAN, MOUSE, RAT),
5HT5A_(HUMAN, MOUSE, RAT),
5HT5B_(MOUSE, RAT),
5TH6_(HUMAN, RAT),
5HT7_(CAVPO, HUMAN, MOUSE, RAT),
5HT3_(HUMAN, MOUSE, RAT),
```

This constitutes a set of 43 sequences. The 5HT1 and 5HT2 sequences were loaded into the Viseur program and an automatic mutant search was run to retrieve mutagenesis information from the TinyGRAP data bank (algorithm described previously). This resulted in the integration of mutant information in the 5HT focused data bank maintained by the program during our modelling.

Aligning 5HT1 and 5HT2 receptor sequences

Aligning the sequences was easier than with classical alignment tools because the TM limits described in the SWISS-PROT files (*FT TRANSMEMBRANE*), even though indicated 'POTENTIAL', are a good starting point for finding the TM sub-sequences. Their quality could be controlled on a few sequences by using the snake-like view before continuing. The idea is to roughly align sequences on the basis of the highlighted sub-sequences (TM), reordering the sequences after each TM, if necessary, to group sequences with higher sequence homology. This way we noticed that the 5HT1A TM1 sub-sequences are really different from other TM1 sequences, although they are reasonably homologous for other TM sub-sequences. This may indicate that a recombination has occurred at some stage during evolution, and this information should be taken into account for the modelling because the importance of the functional role of TM1 in the whole family is now questionable: if the function was retained after a mutation as important as the replacement of one TM (perhaps followed by point mutations at a usual rate), a simple hypothesis would be that this TM contribution to the function was, and is, negligible. Another hypothesis is that the recombination resulted in a small change to the receptor pharmacology, resulting in the emergence of a new subtype: 5HT1A.

The alignment was refined, looking for conserved residues among the TM. At this stage, one may choose a colour residue association that groups chemical properties of residues: a colour for aromatic residues, another for lipophilic residues, etc... Viseur is able to paint residues, on a set of sequences, when they appear mutated (from the project data bank). This feature was

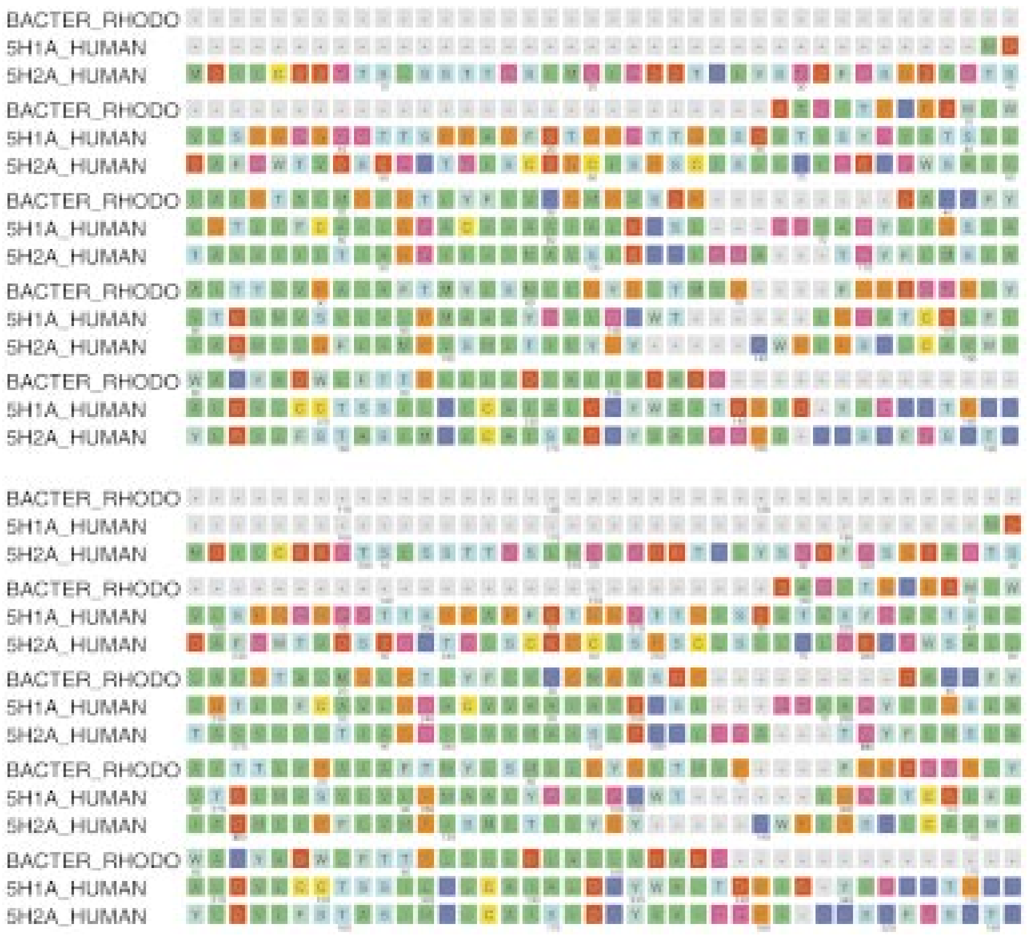


Figure 4. Sequence alignment of bR, 5HT1A_HUMAN, 5HT2A_HUMAN, (configurable PostScript output generated with the Viseur program).

used intensively here to browse the mutant information while refining the alignment.

The obtained alignment was compared to published alignments [1, 3, 19–22]. All were found to be in good agreement (for TM1, TM2, TM4, TM6, TM7), except for TM3, as, according to the authors, one or two helices turn differences appear for the alignment of bR with the 5HT sequences. The alignment we retained between bR, `5HT1A_HUMAN`, `5HT2A_HUMAN` was printed as PostScript with Viseur and is presented in Figure 4.

Evaluating raw molecular models

At this stage, a few raw models were constructed for `5HT1A_HUMAN` and `5HT2B_HUMAN`, to check the agreement between the alignment and mutant data on the 3D model. There are two ways to change the model to fit the mutant data. One could rotate the helices or shift the alignment by a few residues. According to the low homology between bR and 5HT sequences, the two methods can be applied equally here, unless some special method is used for an automatic alignment of the sequences based on calculated properties. On the contrary, if the alignment is straightforward, the rotation of the helices is the only rational degree of freedom in the model. These considerations lead to the following changes to the model.

- TM5: For the receptor `5HT1A_HUMAN`, Ser199 and Thr200 are found by mutagenesis studies to be important to the binding. Here the alignment was shifted by two residues.
- TM7: For the receptor `5HT2A_HUMAN`, N376 (TM7) seems to interact with D120 (TM2) [23], which implies that these two residues should be near in space. The TM7 was rotated to accommodate this constraint.

As rotating the helices is not yet possible from Viseur Model view (this feature is planned to be added in a later release of the program), this part of the modelling was done with Sybyl, using the Sybyl command file generated by Viseur to construct models from the bR PDB file. The modelling was pursued using molecular modelling methods.

Finally, Viseur was used to check the agreement of the resulting models with the mutant data. We may use it in the future (after updates to the mutation server) to check the agreement of refined models with new mutants.

Accessing mutagenesis data through the alignment, application to the IN/OUT concept

Alignments may be used to transfer a special type of information: structural information from one amino acid (taken from the templates) to one aligned amino acid of the receptor of interest. With the construction of the exact determination of the set of aligned residues, this is the basis of the homology modelling. Here, we suggest that in certain cases, where homology between proteins is high, the information transfer could be extended to mutational data. The cases where this transfer is not possible, because of contradictions, are important to notice because the homology hypotheses are certainly not fulfilled for the parts implied in the alignment. This criterion will become more and more useful as the mutational data volume increases.

If in an alignment, we define $(A)_{i,j}$ as the residue of sequence i at the position j in the alignment (j includes the gaps: `A`$_1$`B`$_2$`--C`$_5$), $(A)_{I,c}$ as the residue for the sequence of the receptor I being modelled at position c in one alignment, and $(A)_{R,c}$ the residue from receptor R at the same position c in the alignment, then, according to these notations, we say: if there is no mutational experimental result on $(A)_{I,c}$, but mutational experimental results exist for $(A)_{R,c}$, then one can derive new information on $(A)_{I,c}$. When only one residue in a column is present, for which we have information, the transfer is $\textit{Information}((A)_{I,c}) \leftarrow \textit{Information}((A)_{R,c})$. Because there is only one piece of information in the column, we cannot confront it with others, so we consider its quality to be inferior to the following cases. If an ensemble of residues exists in a column

$$Sc = \{(A)_{Rk,c}, (A)_{Rl,c} \ldots\}$$

and

$$card(Sc) > 1$$

and

$$\forall R\ \textit{Information}((A)_{R,c}) = \textit{consensus}$$

then it is nearly reasonable to say Information$((A)_{I,c})$ $\leftarrow$ *consensus.*

The last case is when a set of residues exists in a column

$$Sc = \{(A)_{Rk,c}, (A)_{Rl,c}, \ldots\}$$

and

$$card(Sc) > 1$$

and

$$\forall R\ \textit{Information}((A)R, c) = I_R$$

Assigning Information$((A)_{I,c})$ becomes a subjective task. It seems to us that this situation should be detected because it can highlight some regions of the alignment where the homology hypotheses are not well fulfilled. Browsing mutant data directly on the alignment is a way to facilitate the detection of such situations.

When mutational information is not available on the receptor of interest (or the information quantity is low) the above procedure can be used when confronting the IN/OUT problem. The problem is essentially (i) to know which amino acids are pointing towards the core of the receptor (IN amino acid) and which are pointing towards the lipids (OUT amino acid) and (ii) to correct the helix arrangement according to the information from (i).

The Viseur program can be useful during step (i). IN/OUT information can be derived from mutagenesis experiment results by the user with the tools provided by the program. If a receptor amino acid is mutated and the new receptor has quite the same activity as the original one, the amino acid can be considered anywhere on the model (OUT or IN but not interacting). If the activity changes (agonist binding change) then the receptor could be considered IN, though the possibility of indirect effects makes this assignment unsafe.

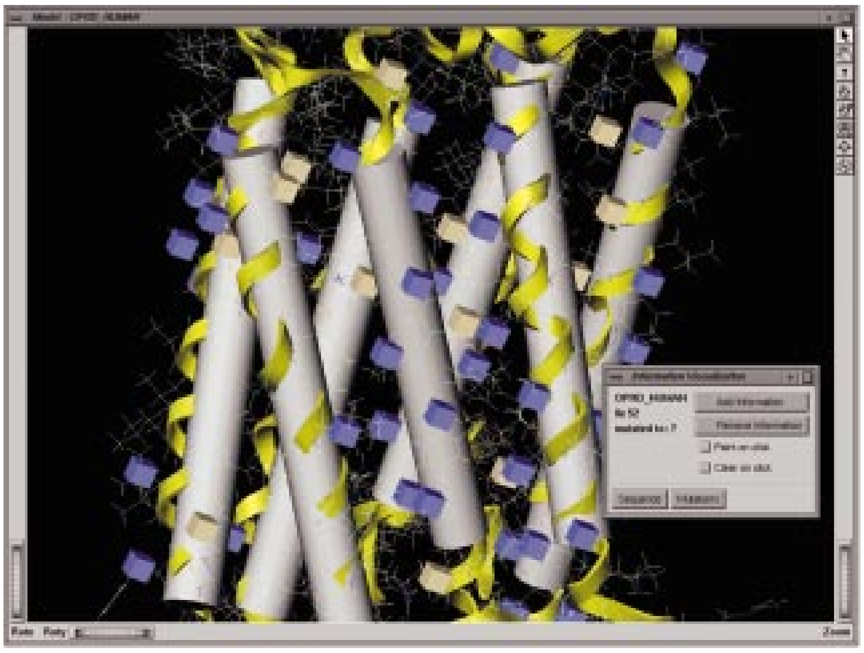


Figure 5. Viseur's Model View of the OPRD_HUMAN receptor, IN/OUT painted. White coloured residues are considered IN, blue residues are OUT or IN but without contact with the ligands.

IN/OUT reduced information is determined for each amino acid of the alignment for which some mutagenesis information is available, for instance colouring residues according to an (IN,OUT) convention. The procedure described above may be used, manually, to transfer that data to the receptor of interest (i.e. colouring the receptor residues IN or OUT). Considering the significant uncertainty about these derived data, it is advisable to consider an average over adjacent residues when tuning the helix arrangement.

The Viseur alignment editor can be used to display the IN/OUT information by a given colour on the alignment. The Model view can be of valuable help during step (ii). The IN/OUT amino acid can be visualised easily on the 3D Model. See Figure 5: this view shows the OPRD_HUMAN receptor model (from Reference 24) IN/OUT painted according to mutagenesis experimental results (OUT means that the residue should not interact with the ligand).

Conclusions and perspectives

In this article, we have described an algorithm to draw snake-like diagrams. This algorithm allows the presentation of GPCR sequence and mutant data in a schematic way, which GPCR scientists are used to. A method to present GPCR models on the WWW, hyperlinked per residue to information, is also presented and its interest discussed. We have described the Viseur program, which implements these algorithms, as well as some applications.

The Viseur program was originally developed for GPCR modelling but may be used for modelling any integral transmembrane proteins since the helix number is configurable. From a practical point of view, the program proposes interconnected visualisation modes that can be used to integrate a lot of the information that is available for a set of GPCRs, and so, to facilitate the modelling. But this program also allows one to test and evaluate the relevance of new working methods: access to mutant information through a sequence multi-alignment editor or a molecular model. It seems to us, according to our experience on GPCR systems, that these methods are of great help during the modelling process, and that their implementation for modelling different protein systems would be advisable.

Finally, the Viseur program visualisation features allow one, considering the HTML and VRML output features, to publish a GPCR annotated model on the Web. The latter features may improve information exchanges on GPCR between the experimental and the modelling scientific communities.

For up to date information about the Viseur program, including distribution access, and recent developments, please visit http:/www.lctn.u-nancy.fr/viseur/viseur.html.

Acknowledgements

The main part of this work was realised in collaboration with 'Sanofi Recherche', under grant number 061064300. The latest developments are realised thanks to the grant accorded to F.C. by the high performance computing center 'Centre Charles Hermite' and the 'Université Henri Poincaré, Nancy I'. F.C. would like to thank G. Vriend and G. Poda for interesting discussions and H. Vollhardt for the PDB2VRML tool. Last but not least, we would like to thank Christophe Chipot and Steven Spencer for carefully reading the manuscript.

References

1. Van Rhee, M. and Jacobson, K.A., Drug Dev. Res., 37 (1996) 1.
2. Kristiansen, K., Dahl, S.G. and Edvardsen, Ø. Proteins Struct. Funct. Genet., 26 (1996) 81. URL: http://www-grap.fagmed.uit.no/GRAP/homepage.html.
3. URL: http://www.sander.embl-heidelberg.de/7tm.
4. Schertler, G.F.X., Villa, A.C.M. and Henderson, R., Nature, 362 (1993) 770.
5. Griegorieff, N., Ceska, T.A., Downing, K.H., Baldwin, J.M. and Henderson, R., J. Mol. Biol., 259 (1996) 393.
6. Cserzö, M., Bernassau, J.-M. and Simon, I., J. Mol. Biol., 243 (1994) 388.
7. Weinstein, H. BIOSCI International Newsgroups for Molecular Biology, bionet.molbio.proteins.7tms_r, 11 April 97.
8. URL: http://expasy.hcuge.ch/cgi-bin/ProMod-GPCR.pl.
9. Joseph, M.-P., Maigret, B., Bonnafous, J.-C. and Scheraga, H.A., J. Protein Chem., 145 (1994) 381.
10. Herzyk, P. and Hubbard, R.E., Biophys J., 69 (1995) 2419.
11. CGI-BIN protocol, specifications 1.1, URL:http://hoohoo.ncsa.uiuc.edu/cgi.
12. Open Inventor Architecture Group, The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Addison-Wesley, Reading, MA, 1994.
13. URL:http://vrml.sgi.com/intro.html, URL:http://webspace.sgi.com/Tools.
14. Vollhardt, H. and Brickmann, J., In Hunter, L. and Klein, T.E. (Eds.) Proceedings of the Pacific Symposium on Biocomputing '96, World Scientific Publishing, Singapore, 1995, pp. 663–673.
15. Bairoch, A. and Apweiler, R., Nucleic Acids Res., 24 (1996) 21.
16. Chemical Abstracts Service, Columbus, OH, U.S.A.
17. Berners-Lee, T., Cailliau, R., Groff, J. and Pollerman, B., World Wide Web: the Information Universe, Electronic Networking: Research, Applications and Policy, Vol. 1 (1992).
18. PDB Contents Guide, URL:http://www.pdb.bnl.gov.
19. Hibert, M.F., Trumpp-Kallmeyer, S., Bruinvels, A. and Hoflack, J., Mol. Pharmacol., 40 (1991) 8.
20. Trumpp-Kallmeyer, S., Hoflack, J., Bruinvels, A. and Hibert, M., J. Med. Chem., 34 (1992) 3448.
21. Glennon, R.A. and Westkaemper, R.B., Drug News Perspect., 6 (1993) 390.
22. Glennon, R.A., Dukat, M., Westkaemper, R.B. and Ismaiel, A.M., J. Mol. Pharmacol., 49 (1996) 198.
23. Seafon, S.C., Chi, L., Ebersole, B.J., Rodic, V., Zhang, D., Ballesteros, J.A. and Weinstein, H. J. Biol. Chem., 270 (1995) 16683.
24. Poda, G. and Maigret, B., manuscript in preparation.

APPENDIX

APPENDIX A: VISEUR input/output file syntax and grammar

Quite often, when a new program is written, a new data format appears. The fact is regrettable but seems inevitable. The reason for this fact is that a new program usually adds new attributes to the data it transforms, which cannot be stored in a standard way using existing file formats. The situation is worse when the programmers ‘adapt’ an existing standard to accommodate the own needs of their programs without describing explicitly the ‘adaptations’. This way one can find at least three SWISS-PROT dialects, differing only by the format of one part of one line. A consequence is that a program designed to read SWISS-PROT should be aware of the three dialects to be able to read any SWISS-PROT file available at this time. This increases program development time and complexity, which is not really compatible with research development. When designing Viseur we wanted a file format with some concept oriented properties and a good potential to change without becoming incompatible with older releases of the program. We describe here the syntax and grammar used widely by Viseur files (receptor, alignment, colouring and script files). The files are a ‘list of thing’ like this (optional elements are surrounded by brackets):

```
identifier [ (tag1) [(tag2)] ... ] {

 [list of thing]
}
```

A more formal grammar description is given here: `list_of_commands`:

`command list_of_commands`$_{opt}$

`command:`
`identifier list_of_tags`$_{opt}$ `command_block`$_{opt}$

`list_of_tags: tag list_of_tags`$_{opt}$

```
command_block:
{ list_of_commands }
```

A command is first composed of an identifier `([^ ]+)` followed by any number of tags (anything enclosed by two matching parentheses), then optionally another list of commands enclosed by ‘{’ and ‘}’. Comments

```
#[^\n]*
```

can appear anywhere on a line, they begin at a ‘#’ (not inside a tag) and finish at the end of the line. This syntax enables concept/object oriented description of the data if used in the following way:

```
RECEPTOR {

  NAME (Vasoactive intestinal polypeptide receptor human)
  LEN (457)
  ...
}
```

The identifier should name the concept. For example, Viseur uses the RECEPTOR identifier to group attributes for a GPCR: the concepts enclosed in the brackets will refer to the GPCR. RECEPTOR defines a GPCR scope. A GPCR has a name: the NAME identifier inside the GPCR scope is the name of the GPCR. The tag of the NAME attribute gives the value of the NAME attribute for the GPCR.

APPENDIX B: snake-like HTML output script

We present here the Viseur script file needed to build a snake-like HTML page from a Web server side.

```
# viseur command file (2.42 compatible).
# Creates a snake-like html output in ./Export.
# Needs further configuration of the HTML attributes depending
# on your web http daemon and file organisation.

load-sequence (OPRD_HUMAN.sw) (s1) # load the sequence OPRD_HUMAN
                                   # with the attached identifier s1

viseur-color {                     # define a colour with a legend.
        RGB (255) (255) (255)
        text (info here)
}

view-painting (on)               # switch on AA painting display.

sequence (s1) {                  # put the scope on sequence s1
        uri (sequence.html)      # attach a URI/URL to it.
        paint-AA (2) (info here) # paint amino acid 2 with the
                                       # colour "info here".
        paint-AA (32) (info here)
        aa (2) {                 # put the scope on amino acid 2 (in s1).
                uri (2.html)     # attach a URI/URL to it.
        }
        aa (32) { uri (32.html)}
}
il-work-around-kludge

2dview (s1) (2d1) {     # create a snake-like view from a sequence (s1).
                        # give it an identifier (2d1)
#       show            # show Snake-like view (open the window).

        html {          # puts the scope on HTML attributes.
                        # please take a look at
                        # http://www.lctn.u-nancy.fr/
                        # viseur/using/documentation/CommandFiles.html
                        # for detailed info on these attributes.

                base-name (OPRD_HUMAN)
                image-map-tool (http://htbin/htimage)
                default-uri (def.html)

                server-type (W3C) # W3C or NCSA depending on your server.
        }
        export-html (./Export)  # put the snake-like html output
                                # in the directory ./Export.
                                # the directory is created if
                                # it didn't exist..
}
```

APPENDIX C: Colouring script example

```
for-sequence (OPRD_MOUSE) {

   paint-AA (95) (agonist binding change)
   paint-AA (107) (no effect)
}
for-sequence (OPRK_MOUSE) {

   paint-AA (105) (agonist binding change)
}
```

The for-sequence node puts the scope on the receptor sequence. The name of the receptor is given by the first tag. The `paint-AA` nodes in this scope will change the colour of the amino acid whose number is given by the first tag to the colour associated with the legend (second tag). The legend is a text associated with a colour. This way the colouring files are more readable and the legend could be used on screen or in a PostScript output. The colours are attached to legends using a colour set file. An in-depth (and up to date) description is available in the machine inline Viseur documentation.

APPENDIX D: Algorithm details

```
begin draw_snake_like(prot is a protein,
        seq is a sequence,
        start is a position)

        p = start;

        EE is a subsequence initialized to seq[1-(TM1b - 1)]
        IE is a subsequence initialized to seq [(TMne+1)-
                                        get_sequence_length(seq)]

        upper_left is a position = (RADIUS_AA, RADIUS_AA)
        bottom_right is a position =
                        get_extrimity_size(EE) +        upper_left
        bottom_EE is a position = (0, bottom_right.y)

        draw_extremity(TM1.upper-left,
                                upper_left, bottom_right,
                                -1,
                                reverse(EE))

for each TM tm:
        TMi is a subsequence initialized to sequence[TMib-TMie]
        pos.x= pos.x + TM_SPACE
        pos.y = bottom_EE.y
 direction = direction(tm)

        draw_tm(pos,TMi, direction)

endforeach
```

```
upper_left is a position =
                                (TM(get_TM_number(protein))).bottom_right
bottom_right is a position = get_extremity_size(IE)
start is a position =

        (TM(get_TM_number(protein))).bottom_right

draw_extremity( start,
                                upper_left, bottom_right,
                                1,
                                IE)

for each LOOP l
  tmb is an integer = TM(originTM(l))b
         tme is an integer = TM(endTM(l))e
  LOOPi is a subsequence = sequence[tmb-tme]

if tmb modulo 2 equal 1 then  # IL

        start is a position = originTM(l).bottom_left
 end is a position = endTM(l).bottom_right
        direction = -1

else # EL

        start is a position = originTM(l).top_left
 end is a position = endTM(l).top_right
        direction = 1
endif

  draw_loop(start, end, LOOPi, direction)
end draw_snake_like

begin draw_extremity (start is a position,
                                        upper_left_limit is a position,
                                        bottom_right_limit is a position,
                                        direction is an integer,
                                        subseq is a subsequence)

        start.y = start.y + 2*residue_radius * direction
 start.x = start.x + 2*residue_radius
# new position
# for drawing is just beside the start position, according
# to the direction

        horizontal_direction is an integer = -1 # to the left
 vertical is an integer = 0

while (exist (r is the next residue of subseq))

if vertical equal 0 then
```

```
        if horizontal_direction < 0 then
                if start.x > upper_left_limit.x then
                        start.x = start.x - 2 * residue_radius*direction
                else
                        vertical = 1
                        start.y = start.y + 2 * residue_radius*direction
                endif start.x
        else
                if start.x < bottom_right.x then
                        start.x = start.x + 2 * residue_radius*direction
                else
                        vertical = 1
                        start.y = start.y + 2 * residue_radius*direction
                endif start.x
        endif horizontal_direction
else
        start.y = start.y + 2 * residue_radius*direction
        vertical = 0
        horizontal_direction = horizontal_direction * -1
endif vertical

draw_residue(start, r)
endwhile
end draw_extremity

begin draw_loop(start is a position,
                        end is a position,
                        loop a subsequence,
                        direction is an integer)

start.y = start.y + 2*residue_radius * direction
end.y = end.y + 2*residue_radius * direction

len is an integer = sequence_length(loop)

if len < 0 end draw_loop

if len * 2*residue_radius <
                                (end.x - start.x + 2*residue_position) then
                # everything on one line

                start.x = start.x + residue_radius
                while (exist (r is the next residue of loop)
                        draw_residue(start, r)
                        start.x = start.x + residue_radius
else
                draw_residue(first(loop), start)
                draw_residue(last(loop), end)
                draw_loop(start,end,shorter(loop),direction)
endif len
end draw_loop
```

```
begin draw_tm(start is a position,
                                tm is a subsequence,
                                direction is an integer)

is_for is an integer = 1
while (exist (r is the next residue of tm))
        ... we will not describe this part: it simply draws the
subsequence corresponding to the TM in a rectangular area,
according to the direction indicated as a parameter.
endwhile
end draw_tm
```

APPENDIX E: Technical information

The first releases of the Viseur program were written with the C language, but new functionalities are now added with the C++ language. The interface was written for the X Window system and Motif toolkit. The program itself is available for SGI workstations only. A few parts of the code should be changed to allow portability but developing, maintaining and packaging releases for more than one architecture would consume too much of our resources. The Model view is written with the OpenInventor 2.1.1 API. The program was tested under IRIX 5.3, 6.1, 6.2, 6.3, 6.4.