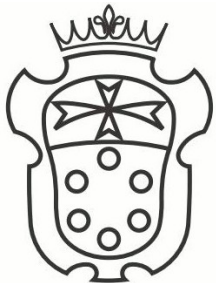
Intuitive visualization of surface properties of biomolecules

Raluca Mihaela Andrei

PhD Thesis in Molecular Biology 2012

Supervisor: Monica Zoppè



Scuola Normale Superiore di Pisa

To my brother,
Ciprian

The greater the obstacle,
the more glory in overcoming it.
(Molière)

CONTENTS

LIST OF ABBREVIATIONS

3D – three-dimensional

aa – amino acid

APBS – Adaptive Poisson-Boltzmann Solver

API – Application Programming Interface

ATP – adenosine tri-phosphate

CG – computer graphics

CPK – Corey-Pauling-Koltun

CPU – central processing unit

EP – electrostatic potential

GE – game engine

GFP – green fluorescence protein

GPU – graphics processing unit

GUI – Graphic User Interface

MD – Molecular Dynamics

MEL – Maya Embedded Language

MLP – molecular lipophilic potential

NMA – Normal Mode Analisys

OS – oligosaccharide chains

RGB – red, green, blue

RMSD – root mean square deviation

ABSTRACT

In living cells, proteins are in continuous motion and interaction with the surrounding medium and/or other proteins and ligands. These interactions are mediated by protein features such as Electrostatic Potential (EP) and hydropathy expressed as Molecular Lipophilic Potential (MLP). The availability of protein structures enables the study of their surfaces and surface characteristics, based on atomic contribution. Traditionally, these properties are calculated by phisico-chemical programs and visualized as range of colours that vary according to the tool used and imposes the necessity of a legend to decrypt it. The use of colour to encode both characteristics makes the simultaneous visualization almost impossible. This is why most of the times EP and MLP are presented in two different images. In this thesis, we describe a novel and intuitive code for the simultaneous visualization of these properties.

For our purpose we use Blender, an open-source, free, cross-platform 3D application used for modelling, animation, gaming and rendering. On the basis of Blender, we developed BioBlender, a package dedicated to biological work: elaboration of proteins motion with the simultaneous visualization of their chemical and physical features.

Blender's Game Engine, equipped with specific physico-chemical rules is used to elaborate the motion of proteins, interpolating between different conformations (NMR collections or different X-rays of the same protein). We obtain a physically plausible sequence of intermediate conformations which are the basis for the subsequent visual elaboration.

A new visual code is introduced for MLP visualization: a range of optical features that goes from dull-rough surfaces for the most hydrophilic areas to shiny-smooth surfaces for the most lipophilic ones. This kind of representation permits a photorealistic rendering of the smooth spatial distribution of the values of MLP on the surface of the protein.

EP is represented as animated line particles that flow along field lines, from positive to negative, proportional to the total charge of the protein.

Our system permits EP and MLP simultaneous visualization of molecules and, in the case of moving proteins, the continuous perception of these features, calculated for each intermediate conformation. Moreover, this representation contributes to gain insight into the molecules function by drawing viewer's attention to the most active regions of the protein.

INTRODUCTION

1 General aspects of visualization

1.1 Visual perception

We are used to perceive information from the surrounding world through the five senses: sight, hearing, smell, touch and taste. The information we receive by sight, apparently without any effort, is the result of an elaboration process involving the eyes and the brain.

Visual perception is the sense which allows the brain to intercept and interpret visible light, creating the ability to see. Amongst the senses, Plato [Plato] considers sight the most noble; while in the other senses, the process implies two parts (a sensor and the sensed, like an ear and a sound), the sight involves three parts – the viewer, the seen and the light. *"Of all the senses, trust only the sense of sight"*, said Aristotle [Aristotle], ranking sight the first of the five senses. The primacy of the visual is emphasized also by the popular phrase *"seeing is believing"*.

Several different processes are involved in visual perception. Physiological processes are the reactions of the cones and rods cells to different light waves, which convert photons into a signal that is delivered to the brain throughout the optic nerve. Neuro-psychological processes allow the brain to interpret the stimuli received.

Figure 1. Examples of incomplete drawings. Our brain is capable to process the information received from eyes and fill the gaps so that we can recognize a triangle with a sphere on each tip, an S shape, a spiky sphere, a snake and a panda. This technique is also used for logos, for example IBM and CNR-IFC.

Seeing can be described as the process of decoding the information

present in the acquired image. The Gestalt school of psychology [Koffka1935] believed that our perception is the result of the relation between stimuli, rather than the sum of the existing stimuli. Humans are able to form a complete mental image from incomplete drawings because our brains fill the gaps (Figure 1); thus, vision is not necessarily what we see but how our brain interprets the world around us. Therefore, it is through our own experiences that we shape how we perceive this world.

The fact that we humans are very good at extracting information through visual observation is well synthesized in the old adage *"a picture is worth a thousand words"*. Psychological studies showed that humans process visual information very effectively.

Visual perception is a complex process that cannot be treated in details here. As a short summary it includes [Sutaria1984]:

- **colour perception and colour constancy** – the ability to distinguish different colours and to recognise different shades of colour in different light intensities;
- **shape perception and shape constancy** – the ability to distinguish shapes and to recognise a shape regardless of size, colour or the angle from which it is viewed;
- **spatial relations** – interpreting the position of one object relative to others;
- **visual analysis and synthesis** – the ability to differentiate between parts and the whole object (e.g. letters that make up words);
- **visual closure** – the ability to complete an incomplete image (see Figure 1);
- **visual conceptualizing** – the ability to make pictures in mind based on observations, experiences and data;
- **visual discrimination** – the ability to interpret differences between objects observed (e.g. b versus d);
- **visual figure-ground distinction** – the ability to focus on important characteristics amidst many (e.g. selecting a blue pencil among many or focusing on a particular word among others);
- **visual memory** – the ability to store and recall information perceived with the eyes (e.g. remembering where an object is situated);
- **visual pattern-following** – the ability to recognise and repeat a visual

pattern;

- **visual sequence** – interpreting images in a realistic order (e.g. arranging pictures of events in the sequence in which they are presented).

1.2 Visualization

Visualization is the process of creating graphical representations such as diagrams, images, animations, maps, *etc.* from data. It reflects also creative ways of representing data visually. Even excluding highly codified visual elements (letters, words, numbers), there is no limit to what kind of information can be translated into an image.

Visualization is a human activity that arose thousands of years ago with cave paintings, in the attempt of people to transmit their ideas to the others. Since childhood, we use several types of graphical representations to describe things. At kinder garden we draw houses, animals, trees, flowers, people and we discover that images are a good way to communicate what we see. Then, at school, teachers use drawings, schemes, maps to help us understand more quickly concepts from physics, biology, chemistry, history *etc.*. We all normally draw an approximate map to show someone the indications to a specific location in town. All these are forms of visualization.

1.2.1 Symbols

Symbols surround us in our everyday life, from street signs to computer icons and marks in scientific disciplines (Figure 2). They are an important issue in our lives and are introduced to ease the communication of concepts and the description of phenomena. We all agree on their significance and use them to indicate precise things. Symbols are good instruments for communication, convey unequivocal concepts and they do not raise doubts when seen.

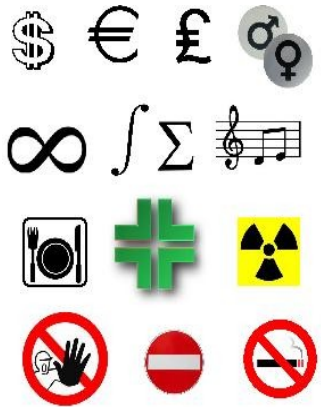


Figure 2. Examples of symbols. Some symbols are commonly used, others are discipline-specific (mathematics-physics, music).

1.3 Scientific visualization

In 1987, a special issue of *Computer Graphics* on 'Visualization in Scientific Computing' published the formal definition of scientific visualization [McCormick1987]: *"Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights"*. The primary goals of Scientific Visualization is to provide insight into scientific data and to allow scientists an easier way to improve and strengthen their understanding and share their data. The transformation of numerical data into a visual representation organizes them in a way that permits the brain to understand relationships within large amount of data.

The type of data usually visualized are numbers, abstract theoretical quantities or relationships, or reflect a gradation or a change in some quantity with respect to others. These datasets are often converted into contours and isosurfaces, glyphs, colour maps and image information. The main task of these figures is to convey as much information as possible about the dataset when they are observed by viewers, facilitating the recognition of patterns and/or the detection of exceptions.

Visualization represents a way to explore (to search for new things), to analyse (to verify existing hypotheses) and to present (to communicate results). Maybe the most important characteristic of visualization is its ability to go beyond the visual.

Traditional areas of Scientific Visualization are: engineering, medical imaging, biology, chemistry, physics, astrophysics, meteorology *etc.*.

The necessity of computer-aided scientific visualization emerged as a

result of rapid advances in computing and electrical engineering technologies, especially high-performance computing during the mid-1980's. Since that time, scientists and engineers have been flooded with increasing amount of data from experimental equipments and computer simulators.

Complementary to the traditional hypothesis-test method of inquiry, data visualization brings together data from different fields, allowing understanding and processing of enormous amount of information quickly, as it is all represented in a simple image or animation. In this way new questions may arise and more discoveries can be made.

In the last years, the interest in biological processes visualization and divulgation increased enormously. David Goodsell, in the preface of “The machinery of life” wrote that his illustrations are meant to “allow us to look at the molecular structure of cells, if not directly, then in an artistic rendition”. In his books, the molecular drawings are in scale, which permits a comparison of various illustrations to understand the dimensions [Goodsell1998]. Recently, Nature Methods [Nature Methods2010] and Science Magazine [Science2011] dedicated special issues to visualization of biological data. Computer-based visualization is widely used in biology to help understand and communicate data, to generate ideas and to gain insight into biological processes especially with the advent of 'omics' studies. These journal issues present a collection of reviews that examine the methods used to visualize genomes, alignments and phylogenetics, macromolecular structures, system biology data and image-based data.

The huge amount of programs available is an indication that biologists still find that their exact requirements are not met by current tools and often prefer to create their own.

Part of the job of being a scientist is to explain the work to others; these might be colleagues, a wider scientific community or the general public. When a scientist is writing for other scientists he or she typically uses very specific scientific terminology, but when they explain complex cellular mechanisms to the general public they need to use metaphors related to human experience to better engage the imagination, and therefore the perception of the audience.

A more direct and very effective way to explain things is by drawing models. Beyond this basic means of visualizing an idea or an observation, 3D physical or virtual models are nowadays the most significant communication

method in Molecular Biology. The development of Computer Graphics made explanations of molecular processes even easier by means of animations.

1.3.1 Scientific visualization steps

The basic elements of scientific visualization follow a series of procedural steps, whose boundary is not always clear cut, but which can be roughly classified as:

- **data acquisition** — scientific data are obtained through an iterative process which consist of observational processes (satellite, medical, microscopy imaging, genomics, proteomics), experiments and simulations;
- **data processing** — raw data require adequate transformation processes to organize, extract and enhance information;
- **computer graphics** — scientific data is converted into a displayable form through two or three-dimensional geometrical modelling, rendering and animation processes;
- **observation and interaction** — visible interaction with the data leads the user to gain better understanding of the information.

Among the four elements described above, the choices pertaining to the modelling and rendering processes may have the most significant impact on how visualized images are perceived by viewers. In particular, good selection of rendering methods and parameters can produce striking photo-realistic images.

Designers of scientific visualization systems have to consider the influence of the human visual system on how such visual appearances are perceived. The selection of appropriate geometries and visual features for rendering holds the key to generating good information visualization. Colours and parameters of rendering and animations are important in effectively conveying information.

2 Proteins

2.1 Protein architecture

Proteins are polymers of long sequences of 20 different amino acids. Each aa consists of a chiral carbon atom (called α carbon) bonded to an amino group (NH_3^+), a carboxyl group (COO^-), a hydrogen atom and a distinctive side chain (residue), see Figure 3.

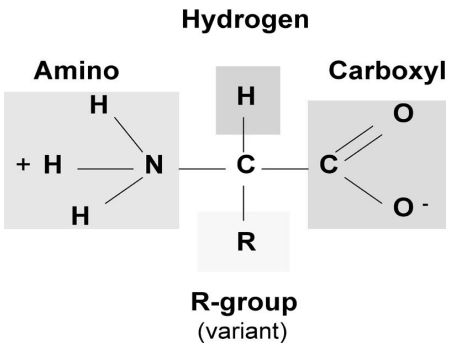


Figure 3. Amino acid general structure. The structure of an aa consists of a fixed structure (the same for all aa) formed of an alpha carbon, an amino group, a carboxyl group, a hydrogen, and a variable side chain (residue).

The specific chemical properties of each aa side chain determine the role of the amino acid in protein structure and function. Amino acids can be grouped in four categories according to the properties of the side chains:

- **non-polar** aa with ***hydrophobic*** side chains that tend to be located in the interior of the protein, where contact with water is minimal;
- **polar** aa with ***hydrophilic*** side chains that tend to be located on the surface of proteins and form hydrogen bonds with water;
- **basic** aa with ***positively charged*** and ***hydrophilic*** side chains located on the surface of proteins;
- **acidic** aa with ***negatively charged*** and ***hydrophilic*** side chains usually located on the outside of proteins.

Amino acids are chained together by peptide bond between the carboxyl group of one aa and the amino group of the successive one (Figure 4). They form polypeptide chains of variable length, up to thousands of aa, with two distinct ends: N terminus (the one terminating in an amino group) and C terminus (the one terminating in an carboxyl group).

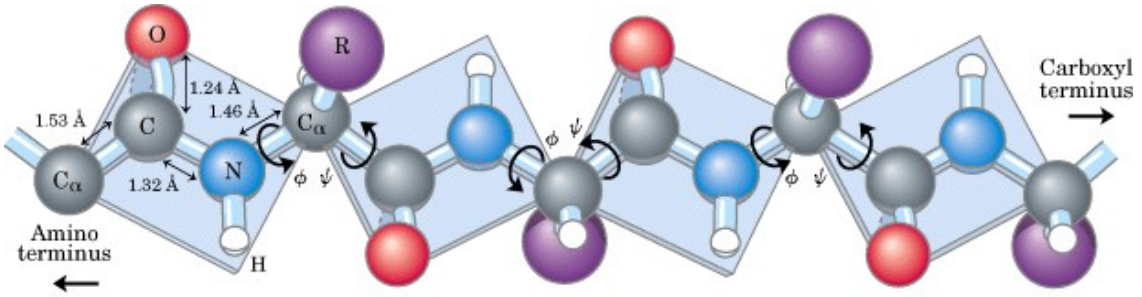


Figure 4. Polypeptide chain structure. Proteins are formed by a sequence of aa connected by the planar peptide bond (represented here as a rigid plane). The bond length and the direction of rotation of the bonds (ϕ and ψ rotational angles) are depicted.

Four distinct aspects define protein's structure (Figure 5) as described by K.U. Linderstrøm-Lang in 1952 [Linderstrøm-Lang1952]:

- **primary structure** is the sequence of aa in the polypeptide [Sanger1951, Sanger1953], also called chain, from N terminus to C terminus;
- **secondary structure** is the local, regular arrangement of aa in the polypeptide, stabilized by hydrogen bonds between CO and NH groups of the main chain; the most common are the α-helix and the β-sheet [Pauling1951a, Pauling1951b, Pauling1951c]. An α-helix is a coiled conformation with the CO group of one aa forming a hydrogen bond with the NH group of the aa located four residues downstream along the linear polypeptide chain. A β-sheet is formed by two or more segments of the polypeptide chain (that can be also distant in sequence) lying side by side and held together by hydrogen bonds. It can be composed of several strands, oriented either parallel or antiparallel to each other;
- **tertiary structure** represents the three-dimensional folding of the polypeptide chain held by interactions between side chains from different regions of the primary structure [Kendrew1958]. Usually, α-helices and β-sheets are connected by loops and fold into compact globular structures called domains, with hydrophobic aa localized in the interior and hydrophilic aa facing the surface of the protein. The tertiary structure is also determined by the interactions between polar and charged aa side chains, forming hydrogen and ionic bonds. The structure of the proteins in the secretory pathway are further stabilized by S-S bonds between cysteines;

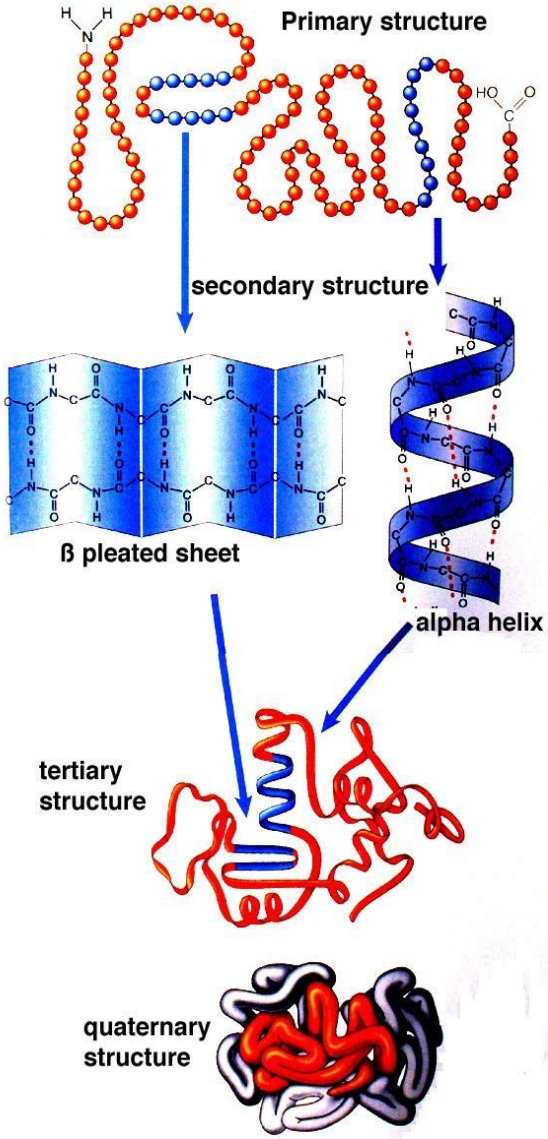


Figure 5. Aspects of protein structures. *Primary structure* is the sequence of aa. In blue are coloured the aa that form an α-helix and a β-sheet (*secondary structure*). *Tertiary structure* represents the 3D organization of the protein in space. The blue elements of secondary structure show that distant aa in the linear sequence may reside close in the 3D space. *Quaternary structure* defines the assembly of different polypeptides in multi-chains proteins, in red one domain.

- **quaternary structure** consists of interactions between different polypeptides in proteins composed of more than one chain [Svedberg1927].

Additional levels of structural classification can be identified:

- **super-secondary structures** – proteins show patterns of interaction between helices and sheets close together in the linear sequence, including α-helix hairpin, β-hairpin and β-α-β unit;
- **domains** – many proteins contain compact, independent units, called domains. They are difficult to define precisely and at present the subdivision of structures into domains varies according to the classification used, which can be structural or functional;

- **modular proteins** – they are multi-domain proteins which often contain many copies of closely related domains: for example, fibronectin contains 29 domains including multiple repeats of 3 types of domains called F1, F2 and F3.

Analysis of the structures of proteins revealed similarities between proteins, an important issue of structural biology. A classification of proteins structures can be consulted on SCOP (Structural Classifications of Proteins) [Murzin1995] and CATH (Class, Architecture, Topology, Homologous superfamily) [Orengo1997] databases. Although based on different criteria, these two databases classify proteins structures into 4 categories: all-α (structures essentially formed by α-helices), all-β (structures essentially formed by β-sheets), α/β (structures with α-helices and β-strands) and $\alpha+\beta$ (structures in which α-helices and β-strands are largely segregated).

2.2 Why it is important to see proteins

Proteins are essential components of cells and are involved in every aspect of biological activity. They have structural functions (actin, tubulin), mechanical function (myosin), act as transporters of small molecules (myoglobin, haemoglobin) or cellular vesicles (kinesin, dynein), transmit information within (kinases) and between cells (protein hormones), provide defence against infections (antibodies), act as enzymes, receptors, are involved in cell adhesion, cell cycle, DNA transcription and replication, RNA synthesis and the production of more proteins.

Molecular genetics research revealed that many diseases stem from specific protein defects [Alberts1998, Eisenberg2000]. Vast research activity is devoted to protein structures and functions, such as in the field of molecular biology or drug design. Seeing the proteins at work is crucial to understand their behaviour, the mechanisms they are involved in, to be able to create drugs and interfere where necessary. By being able to visualize proteins and other macromolecules in the cellular environment, we could get essential insights in the mechanisms of life.

To accomplish this, an important aspect is the information of the three-dimensional structure of proteins which allows understanding of protein fold and provides an insight into the way the proteins act *in vivo*. Understanding how particular amino acids residues are involved in protein function, especially when

combined with knowledge of the 3D structure of proteins, helps to grasp how particular sequences in proteins are involved in biological functions.

2.3 Historical overview of protein visualization

The solution of the 3D structure of myoglobin in 1958 by Kendrew [Kendrew1958] marked the beginning of the new era of structural biology. Since then, a wealth of protein structures has been solved by X-ray crystallography, NMR spectroscopy and cryo-electron microscopy and today the Protein Data Bank (PDB) [Berman2000, Protein Data Bank] counts over 67.000 proteins structures. These data provide much detailed information that help biochemists understand macromolecular functions. Protein structures are obtained and stored as atomic coordinates, which are impossible to interpret by the human brain, therefore they must be presented visually. After solving a protein's structure, its visual representation became the second preoccupation of scientists. Some representation methods are described below.

2.3.1 Physical representations

In 1958, the group of John Kendrew solved the myoglobin three-dimensional structure at 6 Å resolution [Kendrew1958]. Myoglobin is a heme-iron containing protein that carries and stores oxygen in muscle cells. Two years later the same laboratory obtained a map of the protein at 2 Å resolution [Kendrew1960]. For solving the structure of a protein, Kendrew was awarded the Nobel Prize for chemistry in 1962. The first model of a protein was a plasticine model of myoglobin (known also as the 'sausage model') made by Kendrew at Cavendish laboratory in Cambridge (UK). Its cylindrical shape, showing the track of the main chain and supported by wooden rods protruding from a pegboard base can be admired at the Science Museum in London (Figure 6).

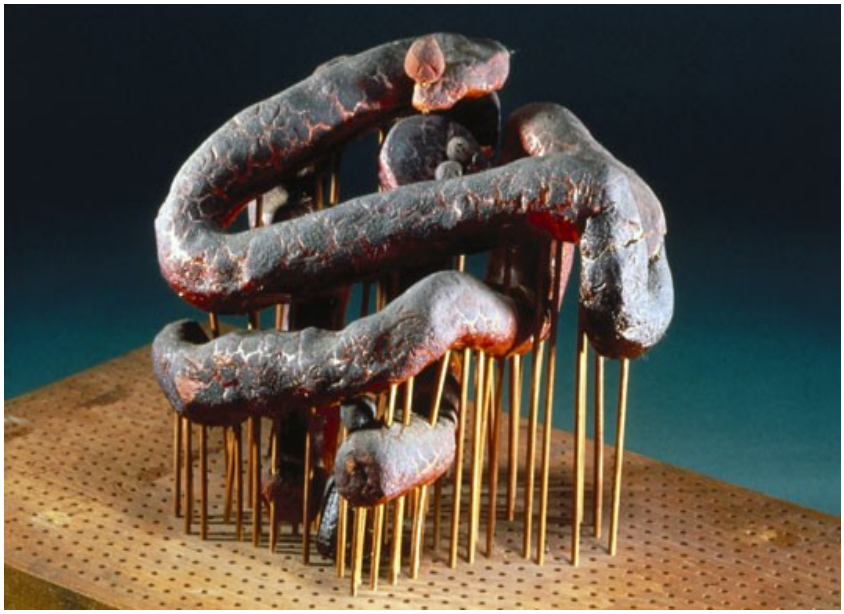


Figure 6. Myoglobin first physical model. The plasticine model of the main chain of myoglobin, made by Kendrew, is supported on wooden rods.

2.3.1.1 All atoms representation

Kendrew model

In 1969, Kendrew built the first atomic model of a protein (Figure 7 left); the structure, 1.8 x 2.5 m, looked like a 'forest of rods'.

For the interpretation of the 2 Å map of myoglobin with 1260 atoms (hydrogens excluded), Kendrew invented a new modelling technique inspired from the toy construction kit Meccano. About 2500 steel rods, 1.8 m high, were positioned on the pegboard; these were decorated with coloured clips to indicate the electron density, so that atoms could be positioned. After the insertion of atoms of the main chain, the densities related to the side chains could be seen at appropriate intervals. The scale of the model was chosen 5 cm/1 Å to allow human hand to reach and fix the clips. The positions of the atoms were continually adjusted as the structure was refined.

Figure 7. First brass models. (left) Kendrew building the all-atom brass representation of myoglobin. The rods help building and stabilizing the complex brass model. **(right)** The brass model of DNA created by Watson and Crick.

Most of the objects, that can be now admired at the Science Museum in London, originally were part of the laboratory where they were handled, discussed, measured, tested against data and corrected and refined.

A similar procedure was employed for the double-helical model of DNA, built of brass models and specially-cut metal bases (Figure 7 right) created by James Watson and Francis Crick from the same Cambridge laboratory.

The Richards Box (Fred's Folly)

In 1968, in a sabbatical year in David Phillips's lab for structural biology at Oxford University, just after solving the structure of ribonuclease at Yale University, Fred Richards and co-workers introduced an optical comparator [Richards1968] that facilitated the building of Kendrew-style brass models (Figure 8). Electron densities resulting from crystallographic solutions were printed by computers on paper, and electron density contour lines were traced by connecting numbers of similar values on the paper. These contour lines were then traced onto transparent plates (1 x 1 m). The plates were mounted vertically, equally spaced, creating a sliced three-dimensional electron density map. Half-silvered (semi-transparent) mirrors were arranged between the map and the hand-made wire model to superimpose the electron density map upon the brass model and adjust the brass atomic pieces so that they fit the density. As larger molecules were solved, the scale was reduced from 5 cm/Å as used initially to 2.5 cm/Å, and then to 1.0 cm/Å.

The Richard's box remained an indispensable tool used by crystallography labs around the world for about 10 years, when it was finally replaced by

computer graphics systems. The earliest computer systems were referred to as "electronic Richard's Boxes" and nowadays programs still use the same basic superposition of electron density maps and atomic models for structure building.

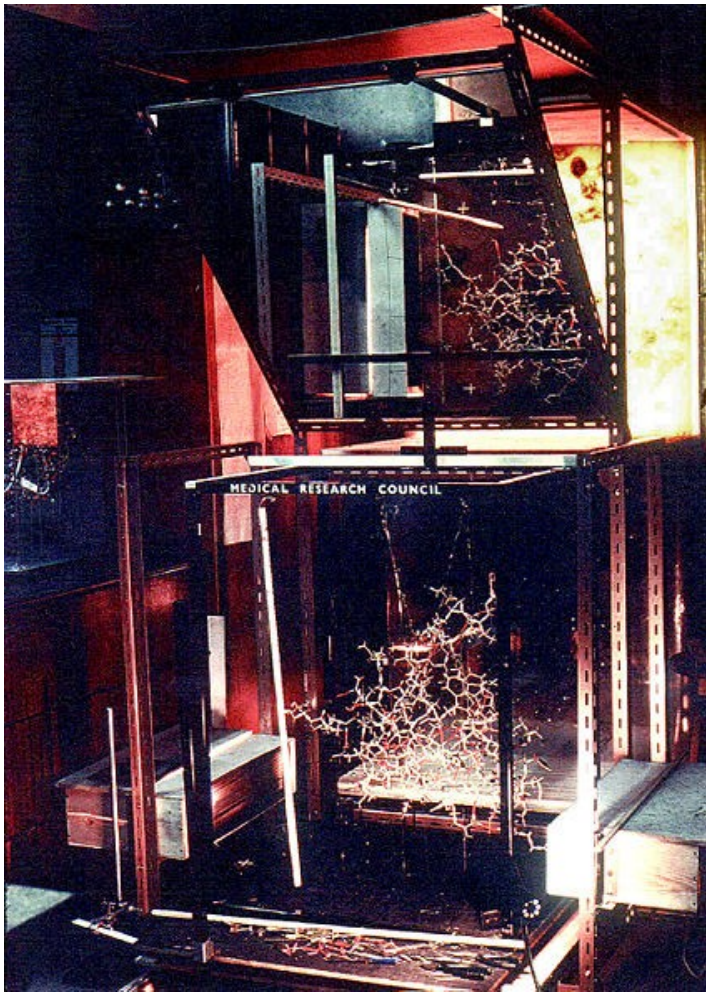


Figure 8. Richard's Box device. Behind the half-silvered mirror at the top is a transparent plastic sheet with a trace of a section of the electron density map. Gazing directly into the mirror, one can see the electron density superimposed with the physical model (at bottom). Light boxes at the bottom sides illuminate the section of the physical model represented according to the electron density map. Multiple plastic sheets containing different sections of the electron density map are stored in a rack at the top right (out of view).

2.3.1.2 Backbone trace models

The physical models containing all atoms were large and cumbersome. Backbone trace models were used to simplify the polypeptide chain as a series of virtual bonds connecting the α–carbons. The protein fold is specified by the bend angles for each α–carbon and the torsion angles along virtual bonds. Two examples of this category were in use in the 1970's: Byron Rubin's wire-bender model and Blackwell Molecular Models.

Byron Rubin's wire-bender model

The wire-bender model consists in a 3 mm-diameter steel rod bent according to the pattern of α-carbon chain of a protein. The bending was accomplished by means of a machine (Byron's Bender [Rubin1972, Rubin1985])

and a list of torsion and bend angles. The shortcoming is variability of the scale, since the α–carbon to α–carbon length is not fixed. For the first time, Byron Rubin introduces the ribbon representation of proteins: helices for α-helices and arrows for β-strands, a very common representation of proteins structures in nowadays software. The small backbone wire models from Byron's Bender were the most manipulable and portable models available at the time (Figure 9).

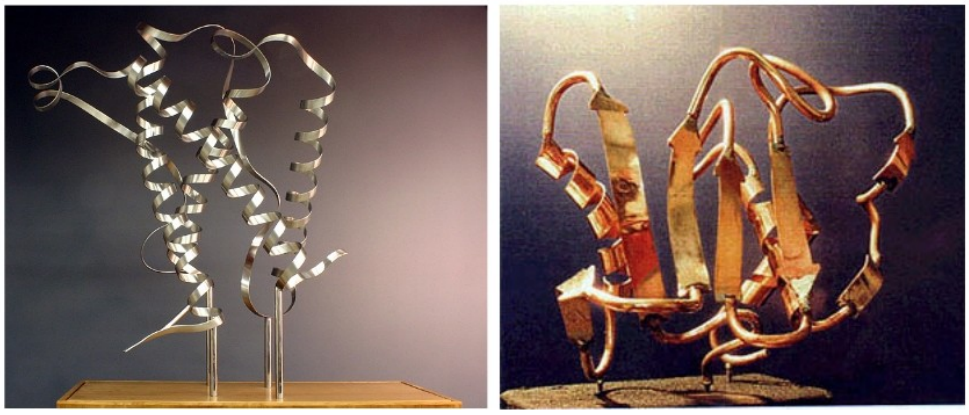


Figure 9. Byron Rubin brass ribbon models. (left) rubredoxin and **(right)** human neutrophil collagenase (on permanent exhibition at the Smithsonian Institution, Washington DC USA, 28 cm)

An example illustrating the importance of models occurred at a scientific meeting in the mid 1970's. David Davies brought a Bender model of an immunoglobulin Fab fragment, and Jane and David Richardson brought a Bender model of superoxide dismutase [Beem1977]. While comparing these physical models at the meeting, they realized that both proteins use a similar fold, despite having only about 9% sequence identity. This incident was the first recognition of the occurrence of what is now recognized as the immunoglobulin superfamily domain in unrelated proteins. The insight was published in a paper entitled *"Similarity of three-dimensional structure between the immunoglobulin domain and the copper, zinc superoxide dismutase subunit"* [Richardson1976].

Blackwell molecular models

Blackwell Molecular Models [Fletterick1982, Fletterick1985] (Figure 10) are more complicated and provide more chemical information. It is a ball-and-stick model system representing α–carbon positions and the peptide link between adjacent alpha carbons.

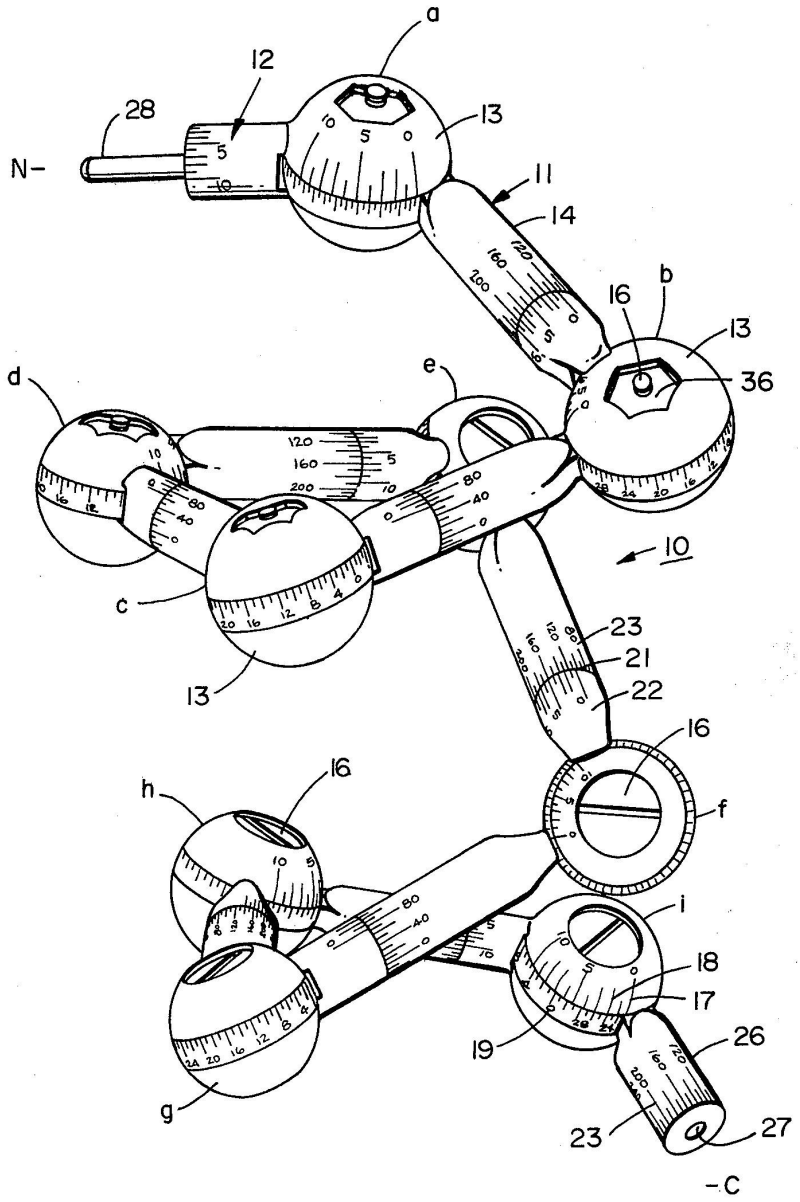


Figure 10. Blackwell representation. The 9 aa of an α –helix are identified by letters from a to i. Each element from the kit is identified by a number, to ease building of macromolecules. The spheres and rods are provided with verniers and scales to set atomic diameters and the bend, torsion and dihedral angles.

They are made of plastic, in twenty different colours to code for amino acid type, which push-fit together at 1 cm/Å scale and the α-carbon to α-carbon bond length is 3.8 cm. The scales and indices observed on the balls and rods serve to mark the angle of bend and the angle of torsion of consecutive amino acids in the molecule. This system allows the attachment of specific side chains. Support rods are added after the model is built, to fix some of the distances between α-carbons and to stabilize the finished model.

Since then, various groups built protein models, a significant collection can be seen at http://3dmoleculardesigns.com/.

2.3.2 Computer representations

Physical and CG representations co-evolved in the early times of structural biology, both used to understand protein structure.

1960's – 1970's

In 1966, Cyrus Levinthal and his colleagues at Massachusetts Institute of Technology developed the first system [Levinthal1966] to display what we call today a “wireframe” representation of molecules structures on a monochrome oscilloscope screen (the terminal display was named Kludge, and it was developed at Project MAC (**M**athematics **a**nd **C**omputation) [Stotz1963]), shown in Figure 11.

Figure 11. First device for wireframe representation. (left) The screen with the globe that control the direction and speed of image rotation. **(middle)** Overview of the molecular modelling system, with the Kludge in the lower left corner. Notice the space-filling models in front of the screen. **(right)** Details of the structure of myoglobin, showing the heme group with two segments of the polypeptide chain surrounding it.

The three-dimensional effect was achieved by rotating constantly the structure on the screen. The rate of rotation was controlled by a globe-shaped device on which the user rested his/her hand (an ancestor of today's trackball). At that time, the full potential of such a set-up was not completely settled. The conclusion of Cyrus Levinthal's description of the system in Scientific American, indicated that there was no doubt that it was paving the way for the future: *“It is too early to evaluate the usefulness of the man-computer combination in solving real problems of molecular biology. It does seems likely, however, that only with this combination can the investigator use his 'chemical insight' in an effective way. We already know that we can use the computer to build and display models of large molecules and that this procedure can be very useful in helping us to understand how such molecules function. But it may still be a few years before we have learned just how useful it is for the investigator to be able to interact*

with the computer while the molecular model is being constructed."

Levinthal's forecast slowly developed in the following decades. In 1965, Carroll K. Johnson, from Oak Ridge National Laboratory, released **ORTEP** (Oak Ridge Thermal-Ellipsoid Plot Program) [Johnson1965], a program written in FORTRAN, to produce stereoscopic drawings of molecular and crystal structures with a pen-plotter (Figure 12). The stereoscopic view aid the visualization of complex macromolecules. It rapidly became a favourite tool of crystallographers to produce illustrations of structures for conference presentations and publications. This program is still in use, and its latest version, ORTEP III, was released in 2000. The current version retains all features and functionality of the original, with the exception of the output which is now displayed on the screen, instead of being printed on paper.

These programs could not directly interpret the crystallographic data; the electron density maps were converted into Kendrew "sticks" models and the coordinates, measured by hand were introduced into the computer to build the virtual model. The next important step molecular visualization was the development of software tools able to interpret crystallographic data.

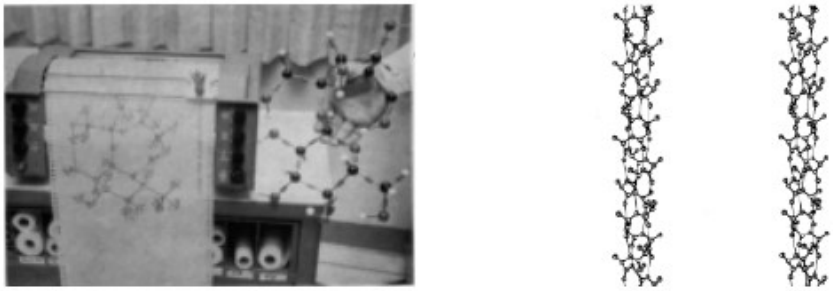


Figure 12. ORTEP system. (left) ORTEP pen-plotter machine. The physical model of the same molecule drawn also by ORTEP can be seen in the image. **(right)** Example of stereoscopic drawing designed with ORTEP.

In the mid 1970's, superoxide dismutase [Beem1977] was the first protein solved crystallographically and visualized entirely with computers by David and Jane Richardson and colleagues. They used a density-fitting computer system called "GRIP" at the University of North Carolina [Tainer1982].

In the late 1970's, more and more crystallographers made the transition to building models for newly solved protein crystals with computers ("electronic Richards' boxes") rather than with physical Kendrew-style models. One of the major advantages was that the computer kept track of the atomic coordinates,

contrary to the Kendrew model where atomic coordinates had to be measured manually, atom by atom. Four systems were widely used: (1) MMS-X, developed at the Computer Systems Laboratory at Washington Univesity in St. Louis [Barry1974, Miller1981], (2) GRIP-75 at the Computer Science Department of the University of North Carolina at Chapel Hill [Tsernoglou1977, Brooks1977, Wright1981, Lipscomb1981, Pique1991], (3) BILDER developed at the Medical Research Council Laboratory for Molecular Biology in Cambridge, England [Diamond1981a, Diamond1981b], and (4) FRODO [Jones1978, Jones1981, Jones1985].

1980's

In 1980, TAMS (Teaching Aids for Macromolecular Structure) project [Feldmann1980] produced the first stereo views of macromolecules (Human Immunoglobulin G). The stereo projection was obtained by using two conventional slide projectors equipped with orthogonal polarizing filters, each viewer wearing polarized glasses. This technique is still in use today and to make easier the understanding of the three-dimensionality of the molecules on printed documents, the images are displayed side-by-side (Figure 13).

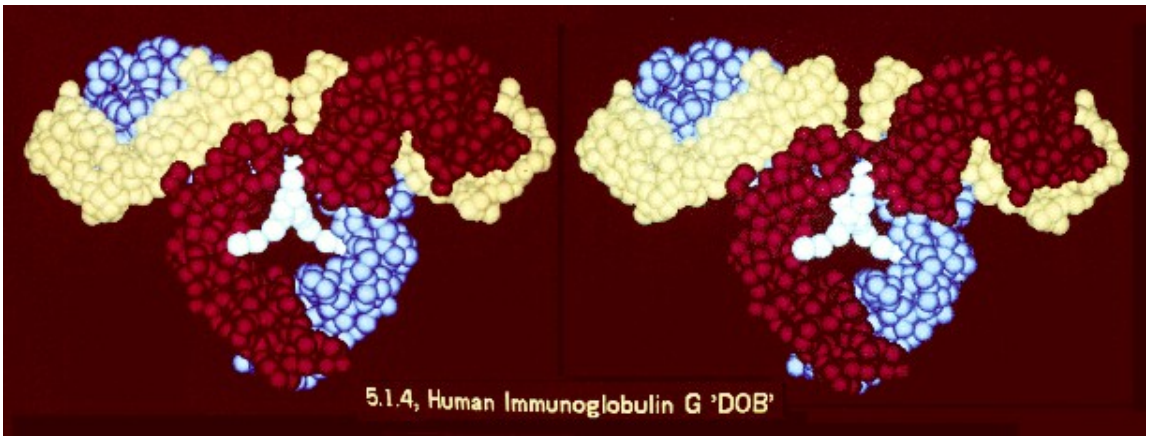


Figure 13. Stereo slide pair from the TAMS project. The example shown here is a crossed eyes stereo of human IgG 'DOB', alpha carbons only; carbohydrate white, light chain yellow, heavy chains red and blue.

During the 1980's, the most popular computer system for crystallographers was manufactured by Evans & Sutherland. These computers displayed the electron density map and enabled an amino acid sequence to be fitted manually into the map. The colour display showed a wireframe rendering of the amino acid chain, and could be rotated in real time. These systems used scalable vector graphics. Rapid rotation was accomplished with three hardware

matrix multipliers (one for each dimension, X, Y, and Z). The software package most often used on E&S computers was FRODO (now evolved to Turbo-FRODO).

During the 1980's, David and Jane Richardson pioneered computer graphics representations of molecular structure with a series of programs developed at Duke University. In the late 1980's, this led to a program called CHAOS written in Evans and Sutherland PS300 function-net language [Richardson1989].

In 1981 Fermi and Perutz published the 'Atlas of Molecular Structures' with stereodiagrams of myoglobin and haemoglobin. To facilitate the viewing, the diagrams were printed in red and green and a foldable pair of red-green glasses was delivered with the book. This technique to provide images with stereoscopic 3D effect is called anaglyph.

The anaglyph images consist of two colour layers (red and green or cyan) superimposed with an offset with respect to each other and the 3D effect is perceived by wearing glasses with coloured lens: red for the left eye and green or cyan for the right eye.

1990's

With the gradual progress of Computer Graphics tools and techniques, the development of software for protein visualization increased. For example, Molscript program [Kraulis1991], by Per Kraulis and released in 1991, was the successor of ORTEP, developed in 1965. Molscript is one of the main programs currently used for plotting protein structures.

A team led by TA Jones, from Uppsala, Sweden wrote the program O [Jones1991], in 1991. The program is designed for scientists with a need to model, build and display macromolecules. O is mainly aimed at the field of protein crystallography, bringing into use several tools, which ease the building of models into electron density, allowing this to be done faster and more correctly.

In 1992, the Richardsons described the **kinemage** (from **kine**tic i**mage**), interactive animated images of molecules, and their supporting programs MAGE and PREKIN [Richardson1992]. By virtue of its implementation on the Macintosh, this was the first program which brought molecular visualization to a large number of scientists, educators, and students. The programs were described in the lead article in the first issue of the journal Protein Science (early 1992), and the program itself was provided on a diskette which accompanied that issue. This

article also included instructions for using the program PREKIN together with MAGE for authoring new kinemages. In the subsequent five years, over a thousand kinemages accompanied articles in *Protein Science,* the majority being authored or edited by Jane Richardson.

Software tools for general use

The first tools for visualization of macromolecular structures were used only by specialists. The increasing number of macromolecular structures solved contributed to the development of software tools for general use, easier to handle.

In 1992, Roger Sayle developed RasMol (***Ras****ter* – the array of pixels on a computer screen – ***Mol****ecules*), an open-source, command-line and stand-alone program for macromolecular visualization [Sayle1992, Sayle1995]. The software included a ray-tracing algorithm that could shade solid objects as they rotated in space. The users explore the molecules structures via a scripting language which permits selection of proteins chains, different colouring and representations. RasMol is one of the most successful software due to an excellent compromise between rendering speed and image quality which permits even large molecules to be rotated in real-time. RasMol was the first program to run on personal computers. Before it, visualization software ran on graphics workstations only.

Since then a wealth of visualization tools have been created and now everybody can use them with minimum effort. Tools like Chimera [Pettersen2004, Chimera], MOLMOL [Koradi1996, MOLMOL], PMV [Sanner1999], PyMOL [DeLano, PyMOL software], Swiss-PDBViewer [Guex1997, Swiss-PdbViewer], VMD [Humphrey1996, VMD], Yasara [Krieger2002] are widely used. A full list of free molecular visualization programs can be consulted on The World Index of BioMolecular Visualization Resources web page [MolVisIndex].

Besides these stand-alone tools, web-based plug-ins were developed such as Chime MDL [Chime] and Jmol [Jmol viewer, Jmol website], integrated on Protein Explorer [Martz2002, Protein Explorer] and Proteopedia web-sites [Hodis2008, Proteopedia] respectively, and FirstGlance, PDBe, RSCB PDB.

Additionally to protein structure visualization, these programs provide also other important analysis tools for proteins characterization: alignment, side chains building, energy minimization, calculation of surface properties, such as electrostatic potential and hydropathy.

Commonly available graphic cards support now the use of OpenGL

Shading Language (included in software such as VMD and PyMOL), which is used to write small programs, called shaders (described below in Rendering section), that produce sophisticated visual effects. QuteMol [Tarini2006] (Figure 14 left) goes a step further and uses GLSL to produce illustrative rendering effects, computing ambient occlusion to better display three-dimensionality of the molecules. ProteinShader [Weber2009] (Figure 14 right) is a Java-OpenGL molecular visualization tool that introduces pen-and-ink style rendering for ribbons to enhance the three-dimensionality and uses shaders to map text labels onto the surfaces of ribbons and tubes, shown in colours.

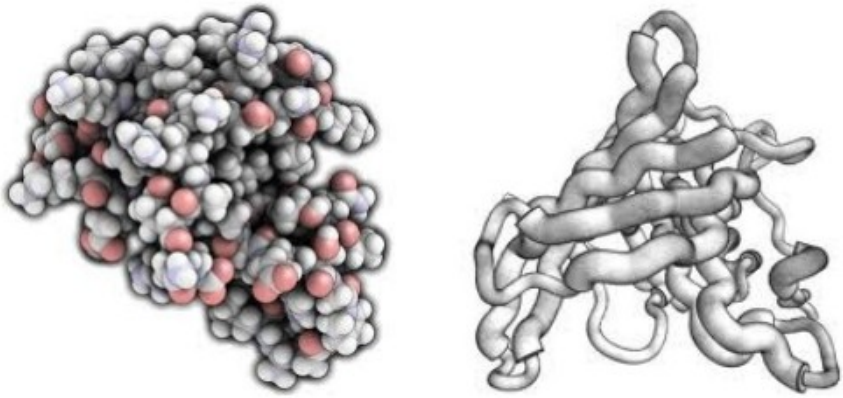


Figure 14. Examples of protein representations. (left) space-filling representation using QuteMol and **(right)** ribbon representation using ProteinShader.

Animations

Using 2D images to show the structure of biomolecules, large part of the 3D information is lost. If images are worth a thousand of words, then animations are worth much more and scientists became aware of the power of molecular movies to communicate their results. Although animations are costly in terms of user effort and computing time (hundreds or thousands of individual images are needed for a rendered movie) they are useful to highlight important features by offering a 'guided tour' of structures and macromolecules interactions.

In 1978, Harrison's group at Harvard solved the first atomic resolution structure of a spherical virus [Harrison1978]. A movie was the way they chose to divulgate the assembly of the virus; they shot with a 16 mm film with a Bolex camera from a computer screen, frame-by-frame. Thus, the movie showing the virus assembly became self-explanatory.

The remarkable progress in computer power and software tools permitted the development of programs to include movie generation, for example Chimera

or eMovie [Hodis2007] (a plug-in for PyMOL) enabling scientists to create informative molecular animations about their research. The commands include rotations, zooms, fading, colouring and different representations.

2.4 Experimental visualization techniques

Humans curiosity to understand biology contributed to the development of sophisticated techniques and methods for biological investigation. The easiest way to understand the shape of things is to look at them, with the help of light. Visible light is part of the electromagnetic spectrum (Figure 15), which extends over a wide range of frequencies (or wavelengths) and includes also radiowaves, microwaves, infrared, ultraviolet, X-rays and γ-rays.

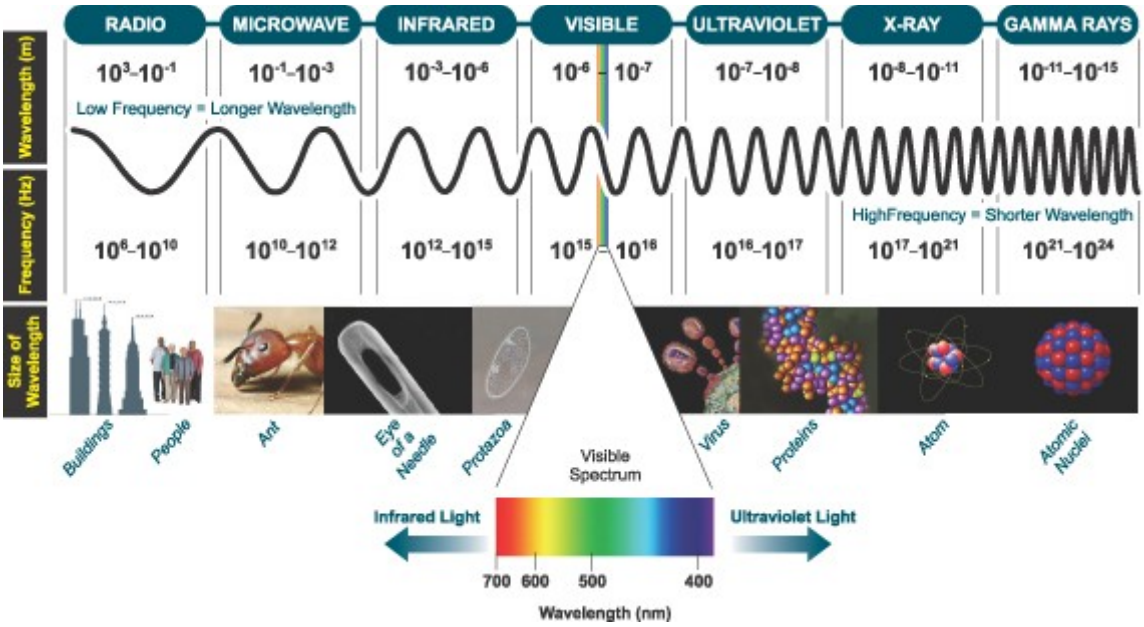


Figure 15. Electromagnetic spectrum size scale. Electromagnetic spectrum frequencies and wavelengths and a comparison between the wavelength and the size of macro and micro scale objects.

When the objects under investigation are too small to be seen by the naked eye, proper instruments are needed. By exploiting different regions of the electromagnetic spectrum and the interaction of atoms with radiation, important information can be acquired about the structure and dynamics of proteins. A wide range of techniques are available for studying proteins.

2.4.1 Microscopy data

The basic technique used by biologists is optical microscopy. With a

resolution of 0.2 μm, cells and large subcellular organelles such as nuclei, chloroplasts and mitochondria can be observed. Usually, the cells are fixed in order to stabilize and preserve their structure (brightfield and darkfield microscopy) and then stained with dyes to increase the contrast between the different components (brightfield microscopy).

It is also possible the visualization of living cells, without staining (phase-contrast microscopy) or by labelling the molecules of interest using fluorescent markers (the most used is Green Fluorescent Protein – GFP) to study their intracellular distribution (fluorescence microscopy). A variety of methods have been developed to study the displacement (fluorescence recovery after bleaching – FRAP) and interactions (fluorescence resonance energy transfer – FRET) of GFP-labelled proteins in living cells.

2D images are very informative, but cells are three-dimensional entities. The introduction of laser and various complex arrangements of mirrors contributed to obtain 3D high-resolution images of the samples (confocal, multi-photon excitation, super-resolution fluorescence, stimulated emission depletion – STED, photo-activated localization – PALM, stochastic optical reconstruction microscopy – STORM) [Cooper2007].

To visualize components of the cell smaller than 100 nm, light microscope is not suitable due to its limited resolution. Using electrons with wavelength of 0.004 nm, electron microscopes can achieve a greater resolution than light microscope, down to 1-2 nm. Two distinct phenomena are involved in various EM techniques: electrons transmission (transmission electron microscopy – TEM) through the specimen and electrons scattering (scanning electron microscopy – SEM). The observation of samples in their native environment is possible at cryogenic temperatures (-195 °C) by means of cryo-electron microscopy and cryo-electron tomography, which creates 3D reconstructions from 2D images.

These are the classical microscopy techniques, included in textbooks. The high speed of technical development and newest scientific discoveries contribute to the continuous development of new microscopy techniques. Almost every week, a new and sophisticated technique is published.

2.4.2 Atomic data

To investigate molecular structures at atomic level (Ångström – 10^{-10} m), the most common methods are X-ray crystallography [Drenth1994] and nuclear

magnetic resonance (NMR) spectroscopy [Wüthrich1995, Kai1997]. The interpretation of the results obtained by these techniques is not unambiguous and entails assumptions and approximations depending upon knowledge of the protein from other sources, including biochemistry.

X-ray crystallography [Drenth1994] is a technique that exploits the fact that X-rays are diffracted by the crystals. X-rays (wavelength 0.5-1.5 Å) are scattered by the electron clouds of the atoms of the studied molecule. For good results, high-quality crystals are needed; the best crystals are pure, perfectly symmetrical, containing a vast number of three-dimensional repeating arrays of precisely ordered, identical molecules. The effect of ordering the molecules in crystals is to enhance the intensity of the scattered signal. The crystals are usually very small (< 1 mm) and can be of different shape, from perfect cubes to long needles, obtained by growing concentrated solutions of the molecule in different conditions, such as temperature, pH and concentration of salts and protein. The crystal is rotated while it is bombarded with X-rays, providing electron diffraction maps. From the intensity of each diffraction spot in the 2D maps for different angles, the value of electron density can be calculated in every point of 3D coordinates (xyz), using Fourier transform. Next, all values of electron densities are integrated and 3D maps are obtained. A model is then progressively built into the experimental electron densities, refined against the data until an accurate molecular model is obtained and the position of each atom in the crystal can be retrieved.

Although very important residues have been (and will be) obtained by X-ray crystallography, the technique has some limitations: for example hydrogen position is very difficult to retrieve, since they have only one electron. Some proteins are refractory to crystallization (membrane proteins or very dynamic loops of proteins, considered disordered parts). The accuracy of the structure determination is validated by the resolution, a measure of how much data was collected. More data acquired, more details are present in the electron-density map features. The resolution is expressed in Å, lower value meaning higher resolution. Values lower than 1.5 Å indicate less ambiguity in positioning the atoms. An index of the structural quality is the R-factor, a measure of how well the model reproduces the experimental data. It is expressed as a percentage of disagreement between the observed diffraction pattern and the calculated model. R-factors less than 20% indicate well determined structures.

Nucleic magnetic resonance spectroscopy [Wüthrich1995, Kai1997] is a method used to obtain information about the structure and the dynamics of proteins by exploiting the magnetic properties of certain atomic nuclei (^{2}H, ^{15}N or ^{13}C). NMR is performed on aqueous samples of highly purified proteins labelled with ^{13}C and ^{15}N and introduced in a magnetic field. The results are chemical shift spectra relative to a reference signal. From these chemical shifts, a set of distances between atomic nuclei that define both bonded and non-bonded close contacts in the molecule are retrieved. Using this information, the atoms positions are calculated, usually more conformations being solved, which may reflect structural dynamics. In general, more internuclear distances measured indicate a higher accuracy of the models. NMR spectroscopists report overall RMSD between the atoms in secondary structure elements in all coordinates sets in the ensemble of structures as a measure of structure quality. An RMSD value of 0.7 Å indicates high-quality structures. In contrast to X-ray crystallography, NMR has been limited to relatively small proteins, usually smaller than 35 kDa. New development allows for NMR study of large proteins as shown in reviews [Mittermaier2006, Tzakos2006, Foster2007].

Electron microscopy arose as a technique usually used to solve 3D structures of large ensembles, such as viruses, nucleic pores, ribosomes; with a few Å resolution, it bridges the gap between the atomic information of single molecules and the micron size information of cellular data (between the molecular and cellular structure) [Studer2008, Zhou2011].

Interdisciplinarity

Biology is a discipline that encompasses many orders of magnitude in size and time. Even considering only the cell level, there is a difference of 6 orders of magnitude between the size of the cell and the dimension of an atom. In order to gain a deep understanding of biology it is necessary to combine data provided by different research fields like biochemistry, genetics, biophysics, molecular and cellular biology, structural biology, immunology, visualization techniques, *etc.*. Interdisciplinarity is key to build a rich image of cellular environments and processes. Cells are delimited by a lipid bilayer, the plasma membrane. The interior of the cell is a crowded ambient, with the cellular components (nucleus, endoplasmic reticulum, Golgi apparatus, proteins and small molecules) immersed in water solution. Here, the inhabitants are involved in a vast range of

processes (from responding to outside signals that induce muscle contraction or production of antibodies to synthesis of new proteins or cellular division). All these processes are part of the cellular activity that can be understood only merging knowledge from various research fields.

2.4.3 Databases

Atomic databases

Protein structures are deposited on the **Protein Data Bank** [Berman2000, Protein Data Bank], a freely accessible repository of experimentally determined three-dimensional structures of macromolecules. The majority, 85%, were solved using X-ray crystallography. These data are stored in .pdb files (identified by a 4-digits number), which contain several information: macromolecule's name and its organism of origin, experimental procedure used to obtain the structure and the detailed parameters, the authors and the related references, the primary and the secondary structure, specification of number of NMR models where the case, various structure refinement methods and their parameters, details about geometry and stereochemistry (covalent bonds length and angles, torsion angles, planar groups, cis-trans geometry *etc.*), list of heterogeneous atoms in the entry, connectivity between residues and other molecules (oligosaccharide chains, ligands, *etc.*). The largest part is dedicated to define the position (3D coordinates) of all atoms. Each line includes the atom identity, the corresponding amino acid, the chain identifier, the atomic x, y, z coordinates, occupancy parameters and thermal parameters (also called B-factor, an indication of the relative mobility of an atom). The file can also report the connectivity annotation (covalent bonds, disulfide bonds). The PDB file format can be read by humans. Each line in a .pdb file is self-identifying and consists of 80 columns.

A crystallographic PDB entry stores atomic coordinates of the crystal asymmetric unit (ASU), rather than of the biological assembly (also referred to as the biological unit) which is the functional form of a biomolecule. An ASU may contain one biological assembly, a portion of a biological assembly or multiple biological assemblies. Starting from ASU, several symmetry operations consisting in translations, rotations or their combinations may be needed to obtain the biological assembly. PDB entries contain two records that define the biomolecule: REMARK 300, which provides its subunit composition related to ASU and REMARK 350, which cites the matrices needed to build it from the

ASU. For example, if a homodimeric protein crystallizes with a monomer in the ASU, REMARK 300 mentions one chain and REMARK 350 two matrices. But if there is a dimer in the ASU, REMARK 300 cites two chains, and REMARK 350, only the identity matrix.

Raw data databases

The Protein Data Bank stores the final results of a molecular structure study: the 3D coordinates of each atom. The raw data, such as NMR spectra and electron microscopy density maps are stored in more specific data bases (NMRShiftDB [NMRShiftDB], nmrdb [nmrdb], EMDataBank [EMDataBank], Electron Microscopy Databank [Electron Microscopy Databank]).

Visualization databases

For human interpretation of structural data, visualization instruments are needed. **Proteopedia** is a powerful web tool to communicate 3D information about macromolecules structures. It is a wiki system that facilitates sharing among the scientific community and has an additional educational component. Proteopedia contains a page for every entry of Protein Data Bank and the Jmol window incorporated allows the user to explore the structure of interest by rotating, zooming, changing the representation (atoms, ribbon, surface) and colouring it (by atom, by amino acid, by secondary structure) *etc.*.

Also the Protein Data Bank and its mirrors offer means of visualization and other ways to explore structural data.

2.5 Protein structure and properties representation

Despite the fact that pdb files are human readable, atomic coordinates sets are impossible to interpret by humans, therefore protein structures are presented visually. Several standard representations exist for proteins: *balls-and-sticks* for atoms and bonds, to visualize covalent bonds, *space-filling* diagram with the atoms visualized as spheres with van der Waals radii to visualize the space occupied, *ribbons* as helices or tubes for α-helices and strands or arrows for β-strands to capture the protein secondary structure and *surfaces* to visualize the interaction with the medium: solvent accessible surface (Lee-Richards [Lee1971]) and solvent-excluded surface (Connolly [Connolly1983]). The information about the atom identity is often delivered by a standard colour-code introduced by Corey, Pauling and Koltun in 1953 (known also as the CPK model):

red for oxygen, blue for nitrogen, grey for carbon, white for hydrogen, *etc.* [Corey1953, Koltun1965]. Not all visualization tools respect this code, adopting new ones (e.g. PyMOL uses green for carbon). When studying proteins, scientists choose the most appropriate of these representations to visualize their protein according to the interest of their study. When standard codes are used, it becomes easy to communicate biological information through images, transmitting to the viewers the author's message.

2.5.1 Protein surfaces

In structural biology, the surface of a protein can be defined in three ways: van der Waals, solvent accessible and solvent excluded (molecular), as shown in Figure 16.

The van der Waals surface, also called space-filling molecular model is obtained by the union of all atoms drawn as rigid spheres with van der Waals radii. It approximates the space occupied by the macromolecule.

The solvent accessible surface is the part of a molecule exposed to the solvent, is calculated by the rolling-ball algorithm (usually the water molecule with the radius of 1.4 Å) and is traced by the centre of the sphere as it rolls over the van der Waals surface.

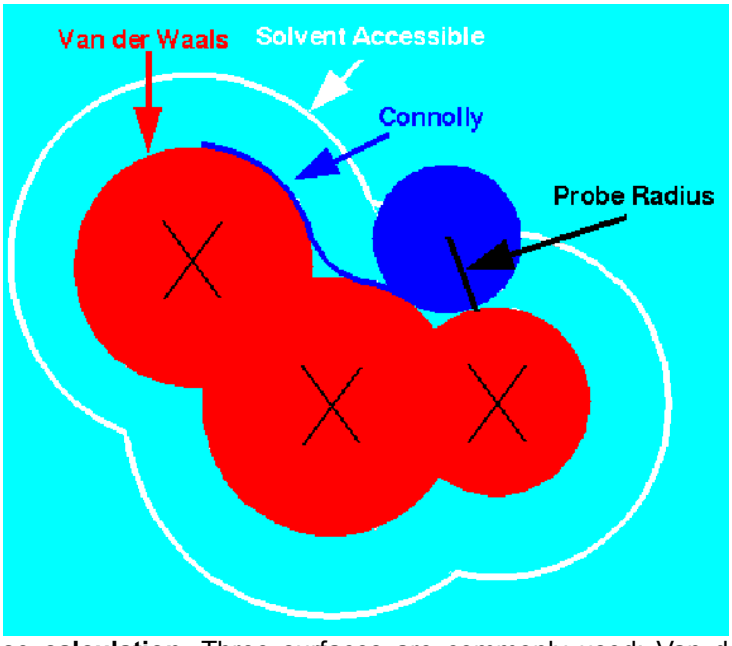


Figure 16. Surface calculation. Three surfaces are commonly used: Van der Waals, solvent accessible and solvent excluded (Connolly).

The molecular surface is considered as a bulk in the solvent and it is

traced by the contact points of the sphere rolling over the van der Waals surface. The rolling-ball algorithm was developed by Frederic Richards and independently implemented three-dimensionally by Michael Connolly in 1983 [Connolly1983] and Tim Richmond in 1984 [Richmond1984].

Several programs are available to calculate the solvent excluded surface, including MSMS (Michel Sanner's Molecular Surface or Maximal Speed Molecular Surface) [Sanner1996], NACCESS [NACCESS], Surface Racer [Tsodikov2002], ASC [Eisenhaber1993, Eisenhaber1995] and Molecular Surface Package [Connolly1993].

Since the surface of a protein is determined by the position of its atoms at any given moment, atomic motion implies that the surface can change in time. This concept applies both to the fast vibrational movements (bond vibrations, rotameric flipping and rotation of chi angles in long side chains) and to the slower conformational changes related to protein activities and intrinsic flexibility (such as disordered loops, or domain motion). For this reason, our system is set to calculate the surface of moving proteins on a frame by frame basis, showing in a visual manner the curvature properties of the molecule as they change in time.

Surface curvature is an important feature for the characterization of the shape of proteins and it is at the basis of shape complementarity; more concave surfaces are more favourable to binding of small molecules and ligands. In practice, the application of ambient occlusion during the rendering of images reveals the curvature feature of the surface.

Two surface properties, electrostatic potential and hydropathy, are usually visualized on the molecular surface.

2.5.2 Protein surface properties

In living cells, proteins are in continuous interaction with other macromolecules and the surrounding medium. In these interactions, two volumetric properties, electrostatic potential and hydropathy have a significant role. These physico-chemical properties are deployed locally along the surface, and understanding surface properties is important to understand proteins functions and interactions with other molecules. Protein-protein interactions are mediated by their molecular surfaces and are governed by the properties of the atoms that line the surfaces, such as hydropathy and electrostatic potential.

2.5.2.1 Hydropathy

Hydropathy is an important physico-chemical property of the protein surface, relevant to its propensity to establish molecular interactions. Hydrophilicity is the propensity of a surface to establish hydrogen bonds with water or polar solvents. On the opposite, hydrophobicity is the propensity to repeal water and preferentially associate with other hydrophobic surfaces. It can be estimated at the various levels: for example, in proteins it is considered that aliphatic and aromatic amino acids are non-polar and hydrophobic; O- and N-containing aa side chains are polar and hydrophilic. Hydropathy is usually expressed as numerical value calculated according to one of several formula, whereas hydrophobicity/hydrophilicity is a more practical measure of how strongly the molecule repeals water.

Walter Kauzmann introduced the term ‘hydrophobic bonding’ to describe interactions driven by exclusion of water [Kauzmann1959]. Hydrophobic interactions are important non-covalent forces that are responsible for different phenomena such as structure stabilization of proteins [Privalov1988, Hendsch1994, Wimley1996, Southall2002], folding of proteins [Dill1990], protein-protein interaction [Mueller2002] and maintenance of the lipid bilayer organization of membranes [Tanford1973]. The importance of hydrophobicity in protein stability, proposed theoretically by Kauzmann, was confirmed by the solution of first protein structure showing that hydrophobic residues are indeed preferentially buried in the protein interior [Kendrew1958].

Calculation of hydrophobicity is important in identifying various protein features such as membrane spanning regions and buried residues, which are highly hydrophobic or antigenic sites, exposed on the surface and which are hydrophilic domains. Usually, these calculations are shown as a plot along the protein sequence, making it easy to identify the location of potential protein features. The hydrophobicity is calculated by sliding a fixed size window (covering an odd number of aa) over the protein sequence. At the central position of the window, the average hydrophobicity of the entire windows is plotted (Figure 17).

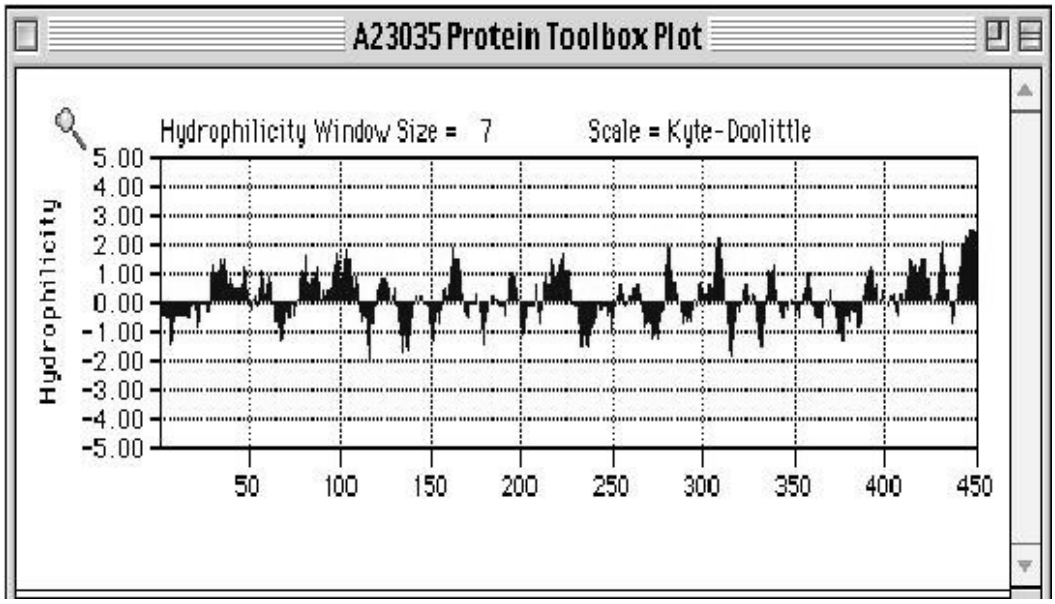


Figure 17. Hydrophobicity/hydrophilicity plot. The plot along the aa sequence, using Kyte-Doolittle scale.

Kyte-Doolittle scale [Kyte1982] is the most widely used for detecting hydrophobic regions in proteins. Additionally, several hydrophobicity scales have been published for various protein studies: Engelman [Engelman1986] for prediction of transmembrane regions, Hopp-Woods [Hopp1983] for identification of potential antigenic sites, Cornette [Cornette1987] for prediction of alpha-helices, Rose [Rose1985] and Janin [Janin1979] for determination of buried amino acid residues of globular proteins. All these scales assign hydrophobicity values to each aa (Figure 18).

aa	aa	Kyte-Doolittle	Hopp-Woods	Cornette	Eisenberg	Rose	Janin	Engelman (GES)
A	Alanine	1.80	-0.50	0.20	0.62	0.74	0.30	1.60
C	Cysteine	2.50	-1.00	4.10	0.29	0.91	0.90	2.00
D	Aspartic acid	-3.50	3.00	-3.10	-0.90	0.62	-0.60	-9.20
E	Glutamic acid	-3.50	3.00	-1.80	-0.74	0.62	-0.70	-8.20
F	Phenylalanine	2.80	-2.50	4.40	1.19	0.88	0.50	3.70
G	Glycine	-0.40	0.00	0.00	0.48	0.72	0.30	1.00
H	Histidine	-3.20	-0.50	0.50	-0.40	0.78	-0.10	-3.00
I	Isoleucine	4.50	-1.80	4.80	1.38	0.88	0.70	3.10
K	Lysine	-3.90	3.00	-3.10	-1.50	0.52	-1.80	-8.80
L	Leucine	3.80	-1.80	5.70	1.06	0.85	0.50	2.80
M	Methionine	1.90	-1.30	4.20	0.64	0.85	0.40	3.40
N	Asparagine	-3.50	0.20	-0.50	-0.78	0.63	-0.50	-4.80
P	Proline	-1.60	0.00	-2.20	0.12	0.64	-0.30	-0.20
Q	Glutamine	-3.50	0.20	-2.80	-0.85	0.62	-0.70	-4.10
R	Arginine	-4.50	3.00	1.40	-2.53	0.64	-1.40	-12.3
S	Serine	-0.80	0.30	-0.50	-0.18	0.66	-0.10	0.60
T	Threonine	-0.70	-0.40	-1.90	-0.05	0.70	-0.20	1.20
V	Valine	4.20	-1.50	4.70	1.08	0.86	0.60	2.60
W	Tryptophan	-0.90	-3.40	1.00	0.81	0.85	0.30	1.90
Y	Tyrosine	-1.30	-2.30	3.20	0.26	0.76	-0.40	-0.70

Figure 18. Examples of hydrophobicity scales.

The differences between the systems shown above, with respect to the

values of hydrophobicity is due to the different methods used for constructing the scales of specific amino acids. For example, Janin and Rose scales were both constructed by examining proteins with known 3-D structures and defining hydrophobic character as the tendency of a residue to be found inside a protein, rather than on its surface. Kyte-Doolittle, instead, is derived from the physico-chemical properties of aa side chains.

In the field of drug design, the propensity of a molecule to interact with the solvent is usually referred to as lipophilicity (the molecule propensity to interact with fats). It is a major determinant of pharmacokinetic and pharmacodynamic properties of drug molecules [Leo1971, Dearden1985, Kubinyi1979]. The quantitative descriptor of lipophilicity is the partition coefficient P [van de Waterbeemd1987, Leo1993], which represents the ratio of concentrations of a compound in the 2 phases of a system of 2 immiscible solvents at equilibrium (eq. 1 And Figure 19). P is a measure of differential solubility of a compound between these 2 solvents (usually water and octanol), a measure of how hydrophobic or hydrophilic a chemical substance is and it is expressed as logP.

$$P = \frac{[X]_{oct}}{[X]_{w}} \tag{1}$$

where $[X]_{oct}$ and $[X]_{W}$ are the molar concentrations of the compound in octanol and water, respectively.

It is mostly used in its logarithmic form, log P. Partition coefficient P values are in range 10^{-3} – 10^{7}, therefore, logP values are in range [-3, 7].

logP is mainly used to calculate hydrophobicity of small molecules. The methods for calculating logP can be divided in 2 major approaches: substructure and whole-molecule.

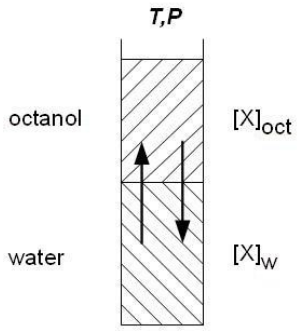


Figure 19. Partition coefficient. Partition of a compound in a two-phase system of octanol and water.

Substructure approaches consider the molecules as the sum of its basic groups (*fragmental methods*) or atoms (*atom contribution methods*). The final

logP is obtained summing the substructure contributions.

a) Fragmental methods [Rekker1977, Rekker1979, Hansch1979] evaluate molecules by fragments and apply correction factors in order to compensate for intramolecular interactions. Fragmental methods work according to the general formula given in Eq. 2

$$\log P = \sum_{i=1}^{n} a_i \cdot f_i + \sum_{j=1}^{m} b_j \cdot F_j \qquad (2)$$

f = fragmental constant; *a* = number of fragments; *F* = correction factor; b_j = frequency of F_j

The first term considers the contribution of fragment constants, f_i, and the incidence of this fragment, a_i, in the query structure; the second term considers the contribution of the correction factor, F_j, and its frequency, b_j. Defining fragments larger than single atoms guarantees that significant electronic interactions are comprised within one fragment; this is a prime advantage of using fragments. Arbitrary fragmentations and missing fragments that prevent calculation frequently hinder the use of this method.

b) Atom-based methods split molecules into single atoms and commonly do not apply correction rules. They work by summing the products of the contribution of an atom type *i* times the frequency of its presence in the studied molecule (see Eq. 3).

$$\log P = \sum n_i \cdot a_i \qquad (3)$$

n_i = number of atoms of type *i;* a_i = contribution of an atom of type *i*

Since the partition coefficient is not a simple additive property, the constitutive feature is covered by classifying huge numbers of atom types according to structural environment. An advantage of atom-based methods is that ambiguities are avoided; a disadvantage is the failure to deal with long-range interactions. Another disadvantage is that a huge number of atom types is needed to describe a set of molecules. Several atom-based approaches are available: Broto [Broto1984], Ghose-Crippen [Ghose1986], refined Ghose-Crippen [Ghose1998].

LogP values reflect only the overall lipophilicity of a molecule and it is usually used for small molecules. This one dimensional parameter contains limited information and becomes insufficient when topological features of molecules are analysed in the context of intermolecular interactions with

receptors.

Whole molecule approaches inspect the entire molecule. The most used approach to quantify hydropathy is *molecular lipophilic potential (MLP)*, which defines the influence of all lipophilic fragmental or atomic contributions of a molecule on its environment and offers a quantitative 3D description of lipophilicity. MLP is a major tool to assess the dependence of lipophilicity on conformation. The MLP describes how lipophilicity is distributed over the different parts of a molecule; it represents the spacial distribution of the ability to form hydrophobic interactions, and is a property that pertains directly to the surface, since its effect decreases very rapidly with the distance between the interacting parts.

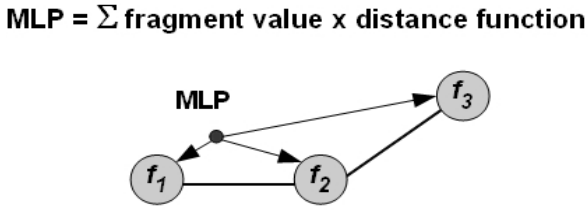


Figure 20. MLP definition.

At a given point in space, the MLP value represents the results of the intermolecular interactions between all fragments and the solvent system at that point. Two components are necessary to calculate MLP: a fragment scheme and a distance function, as shown in Figure 20.

Other whole molecule approaches include *molecular properties* such as charge densities [Klopman1981], surface area, volume, shape and dipole moment [Bodor1989]; molecular weight, heat of formation, SASA and LUMO energy [Makino1998] and electrostatic potential [Sasaki1991] (a quantum chemical approach) are used to predict log P.

Program	Author (Year)	Fragment Scheme	Distance Function
---	Audry (1986)	Rekker	hyperbolic
---	Fauchere (1988)	Rekker	exponential
---	Cohen (1990)	Crippen	hyperbolic
HINT	Kellog (1991)	Leo	exponential
MOLFESD	Brickman (1993)	Crippen	Fermi
CLIP	Testa (1994)	Broto	exponential

Figure 21. MLP formulae

However, no universal equation has been proposed for hydropathy calculation, the most common way being the use of an exponential [Fauchère1988, Gaillard1994, Testa1996], a hyperbolic [Audry1986, Furet1988] or a smoothed step-function, also referred to as the Fermi-like potential [Heiden1993], as shown in Figure 21.

Hydropathy is always visualized on the surface of the protein, using ranges of colours.

2.5.2.2 Electrostatic potential

Whereas many biologically relevant protein–protein interactions derive their affinity from the burial of hydrophobic surface, also polar interactions, hydrogen bonds and electrostatics have been shown to play a key role in determining specificity and, in some cases, the thermodynamics and kinetics of macromolecular association [Honig1995, Davis1990, Davis1991]. Protein-protein associations are mediated both by the hydrophobic effect and by electrostatic interactions [Cherfiels1991]. In some associations the hydrophobic effect may be the dominant driving force, while in others electrostatic interactions may play a very important role particularly when the binding interface is very hydrophilic [Xu1997b].

Electrostatics plays an essential role in all processes involving proteins, DNA or RNA, but its evaluation requires elaborate calculations that are highly sensitive to the solvent model [Sheinerman2000].

The electrostatic potential at a point is defined as the amount of work per unit charge required to move a charge from infinity to a given point.

Unlike MLP, for the calculation of the electrostatic potential (EP) two methods are available:

Coulomb formula (eq. 4) is the simplest method, implemented in Swiss-

PDBViewer software, which assumes the molecule is in vacuum, the calculation includes point charges and the mobile ions are not considered; nevertheless, it gives a good overview of a macromolecule in an environment.

$$V(r)=k\frac{q_2}{r} \tag{4}$$

Poisson-Boltzmann equation is a more complex and accurate way to calculate EP taking in consideration the contribution of the neighbour atoms and simulates the solvent and salts through a continuum model [Sharp1990, Gilson1987, Zhou1994].

Programs such as APBS [Holst2000, Baker2001] and DelPhi [Rocchia2002] are specialized in solving this equation.

EP poses great challenges for visualization since the electric field varies in magnitude and direction in the space surrounding the molecule.

While for EP calculation there is general agreement on the 2 methods described, for its visualization several solutions have been proposed. The most common way to visualize EP is using isosurfaces (Figure 22 left) enclosing all regions with values higher than a given threshold or directly mapped on the molecular surface (Figure 22 middle). The colours used are usually red indicating the negative potential and blue the positive potential.

Less common, EP can be visualized by field lines. VMD software calculates EP through APBS plug-in and displays it as field lines coloured according to the EP values (Figure 22 right). A combination of representations can also be used: EP values mapped on surface and field lines (Figure 22 right).

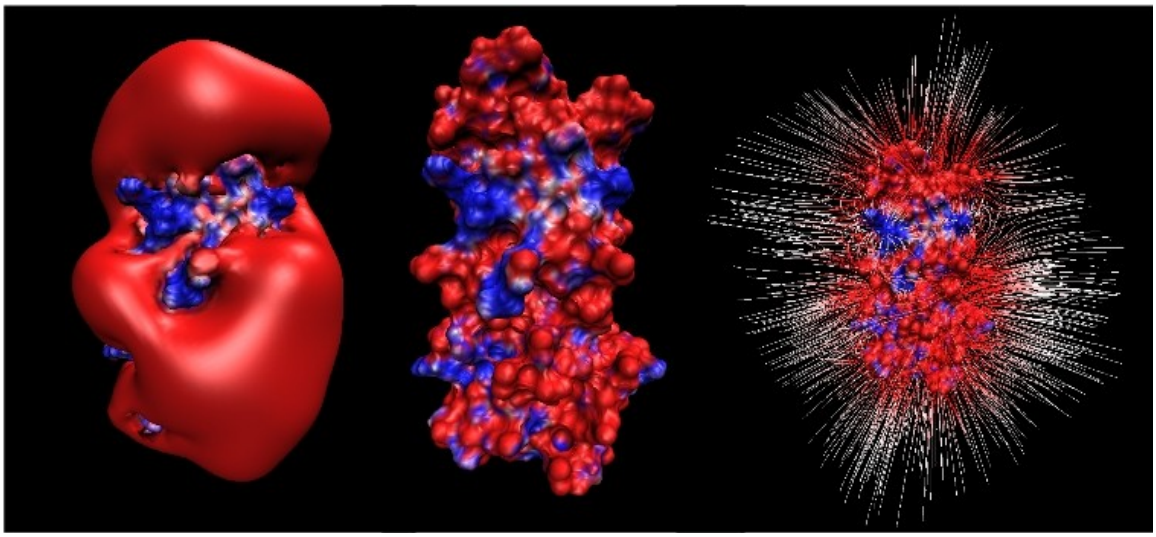


Figure 22. EP representations. (left) isosurfaces; **(middle)** EP values mapped directly on the molecular surface; **(right)** EP values mapped on surface and field lines; the colour range goes from red (negative) to blue (positive). All images are created using VMD.

An alternative to field lines, EP can be represented as a grid of little cones with the tip oriented towards the negative end of the field lines, as implemented in MOLCAD (Figure 23).

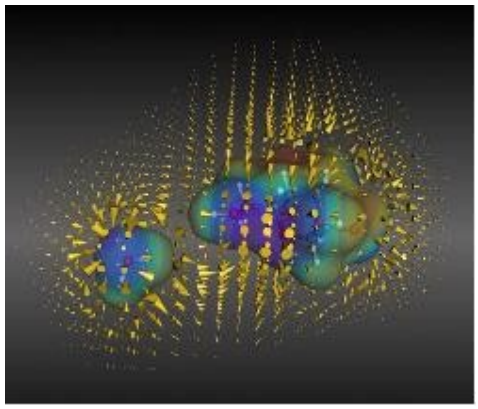


Figure 23. EP representation in MOLCAD. The grid of EP values are visualized as cones oriented towards the negative end of field lines.

2.6 Open issues in protein visualization

Since the common approach of the software tools is to use colours for different kinds of representations such as: atoms identity, secondary structure, electrostatic potential, hydropathy, binding sites, the message is not immediate captured.

Moreover, every visualization tool provides its own colour-code making information delivery confusing and imposing the necessity of a legend. Some of the ranges of colours available in VMD to visualize EP are shown in Figure 24.

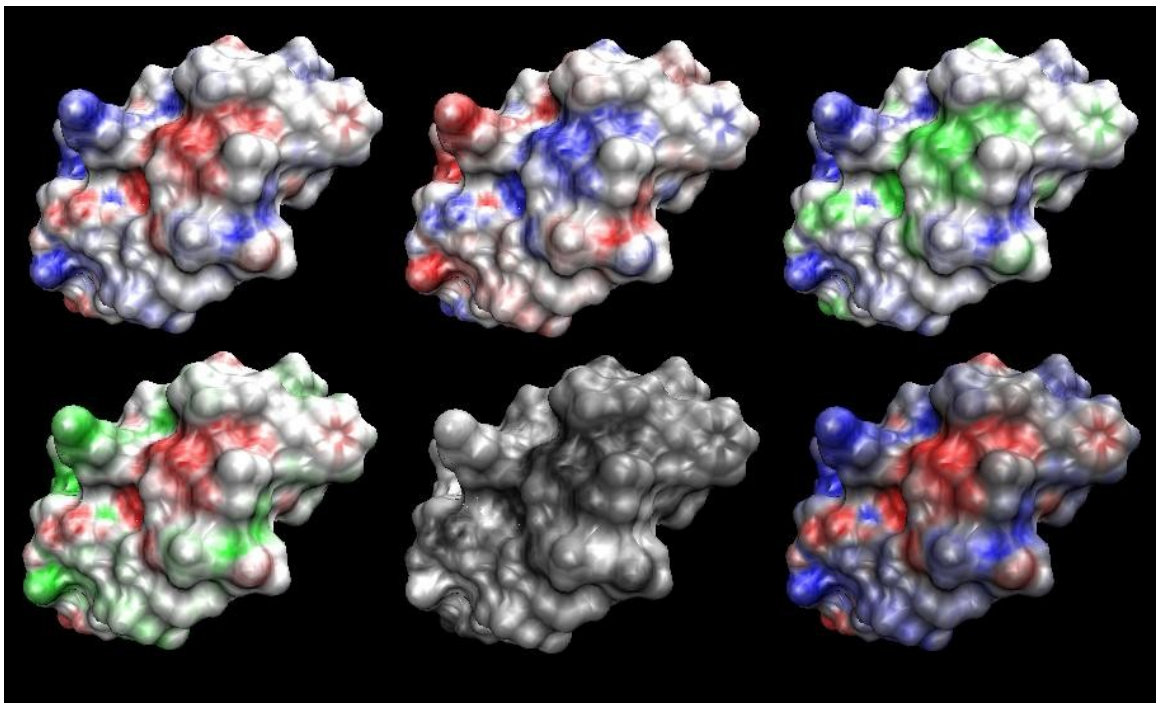


Figure 24. Colour codes for EP visualization. Various ranges of colours available in VMD are shown here.

Using colours to transmit information (hydropathy and EP in Figure 25), the simultaneous visualization of surface properties implies colours overlapping, which would result in a mixture difficult to interpret.

Due to this fact the surface features are always represented separately. A universal “metaphor” for surface properties should be established and in this thesis we are presenting two visual codes for EP and MLP representations which permit their simultaneous visualization.

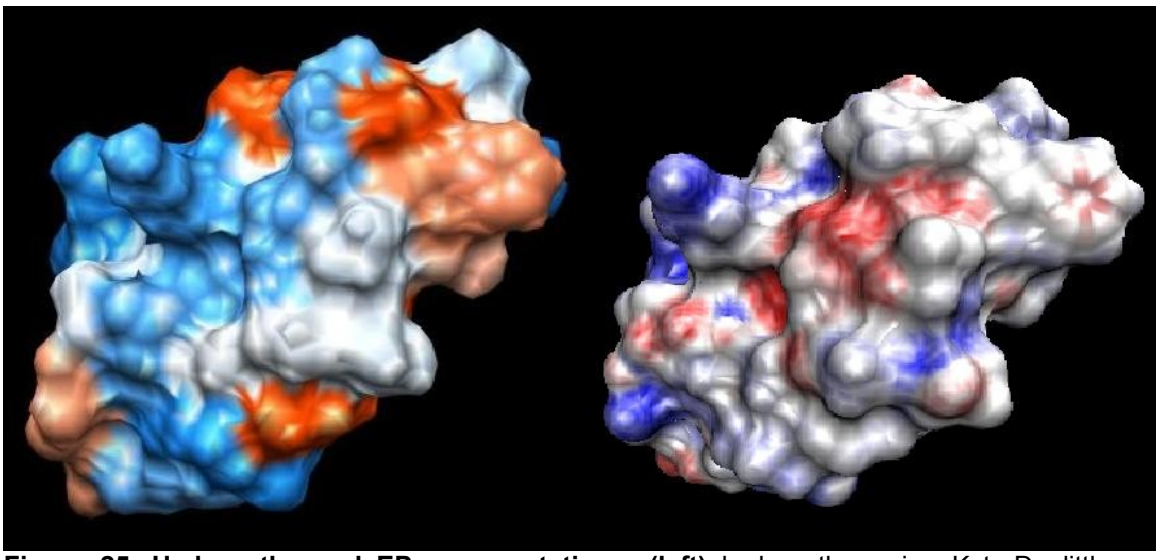


Figure 25. Hydropathy and EP representations. (left) hydropathy, using Kyte-Doolittle aa values, visualized in Chimera (orange for hydrophobic, blue for hydrophilic); **(right)** EP representation using VMD (red for negative, blue for positive).

3 3D animation and rendering

3.1 General aspects

Putting together available information about cellular and molecular studies, a very rich image about cellular environment and molecular processes can be created, taking advantage of Computer Graphics (CG) techniques. CG refers to any image or sequence of images generated using a computer and can be divided into two fields: two-dimensional and three-dimensional. 2D CG (Gimp, Photoshop) is related to the generation of digital images from two-dimensional models. In 3D CG, objects are built in a 3D space and not simply drawn on a plane (canvas) and the depth concept is added (the third dimension). Computer Graphics are a means of creativity, of expression of ideas and in essence they are similar to the art of painting, photography and cinema.

Since the late '90s, the development of CG techniques has advanced at spectacular pace. Among the most widely used tools, is the art and science of 3D animation. This technique consists in the creation and animation of 3D objects (complete with surfaces, skeletons, and simulated physical properties) in a virtual world, which can be 'filmed' using virtual cameras and lights. Several programs are available for this, including the commercial packages Maya, 3D Studio Max and Softimage XSI (all from Autodesk [Autodesk]), Cinema 4D (MAXON Computer GmbH [Maxon]) and the open-source Blender [Blender].

Not surprising, all of these have been used for the study and representation of biological molecules and processes. Some examples are collected and visible on www.molecularmovies.com, www.molshots.com or on www.scivis.ifc.cnr.it. The films range from the simple representations of the mechanical functioning of a single protein, to complex events involving many subjects such as DNA replication and RNA processing, to views of major cellular processes, such as apoptosis, *etc.*. These latter ones are important scientific efforts and add to their educational value the bonus of rising interest in the general public to approach biology.

For the biology community, some 3D animation tools have developed special features and interfaces especially created for molecular visualization. Examples of such tools are Molecular Maya (mMaya) based on Maya software or Embedded Python Molecular Viewer (ePMV [Johnson2011]), a plugin for Maya, C4D and Blender.

Traditionally the process of creating a 3D animation film consists of a number of steps roughly grouped in modelling, animation, rendering, special effects and compositing.

Modelling

Objects are created in the virtual world by modelling them in the 3D scene starting from 3D primitives (such as cubes, spheres, cones, pyramids, torus, *etc.*) or importing them from other programs. The virtual space is defined by 3 axes (X, Y and Z) that represent width, hight and depth, respectively. The intersection of these 3 axes is called 'origin' (0,0,0 coordinates). In Computer Graphics, objects are defined by vertices, joint together by edges and forming faces with defined normals. Modelling complex forms is achieved by manipulation of object components; the user can simply move the vertices, edges or faces to another location, can extrude them out (duplicate the components and then move them) or can scale them. Some special transformation techniques include extrude, revolve, loft, boolean operations, as well as sculpting. The user's creativity and imagination, combined with the precision of these methods contribute to obtain pieces of art, from abstract to organic models. Another method to build objects is by scripting, interfering with the CG system through APIs (Application Programming Interface), a set of functions that the user can use to communicate with the software via scripting.

Animation

Animation consists in a variation in time of position, colour, dimension, *etc.*. The object animation can be achieved in various ways: by direct rotations and/or translations of the object, by mesh deformation obtained by moving its components (vertices, edges, faces), via skeleton (inverse or forward kinematics), by moving object along a path or by using the Game Engine (GE), typically deployed in video games. Additionally, physics-based animations can be achieved by simulated forces such as gravity, magnetic, vortex, wind *etc.*. A new technique used is motion capture, which consists in registering the movements of an actor and transferring them in sequence to the 3D character. Particles animations are used to create special effects simulating snow, fire, explosions, rain. An animation is built by setting *key frames*, procedure that assigns values to an object's attribute (translate, rotate, scale, colour) at a specific time. Cameras and lights are also objects in the 3D virtual space, and as such, can be keyframe animated (including lenses and light intensity).

The software interpolates then between them creating an animation curve for each keyed attribute. The points of the curves indicates the values of an attribute at a particular time. The animation can be controlled by modifying the animation curves. A *time line* holding key frames permits the playback of the motion frame-by-frame or at film rate of 24, 25 or 30 frames/second. As well as modelling, animation can be achieved by scripting, using APIs.

Rendering

Once the animation is defined, the scene is 'dressed': objects are given specific surface properties and appearance (textures), a background is introduced and lights and cameras are created to proceed with the 'filming' (rendering of all frames). Rendering is the final process of calculation of lights and shadows, of the position of the materials and colours of the objects, of the movement of animated objects, etc. to generate a sequence of 2D images that display the content of the virtual scene. Rendering of a virtual environment is similar to taking a photo or filming a scene in real life. The rendering time depends on the complexity of the scene, the number of lights, the quality and the dimensions of the output.

For photo-realistic renderings, materials (shaders) and textures are applied to the objects. A shader is a group of controllable attributes that affect the material properties. In CG programs, shaders can be matte or reflective,

characteristics that can be combined to obtain particular effects. Materials give consistence (substance) to an object by adding colour, shininess or tactile feeling. Textures (procedural or bitmap) help materials become more natural, affecting how something may feel (smooth or rough) or how it looks (colours and patterns). Bitmap textures are images, while procedural textures use mathematical formulas to generate surface appearances. Textures are mapped onto the surface of the object like wrapping paper. The attribution of the texture to the object's surface is more complicated than this, as the texture should not be deformed or wrinkled. The procedure requires a technique called UV mapping [Catmull1974]. First, the object is unwrapped to generate a texture parametrization. UV unwrapping is a procedure that consists in flattening a 3D object (e.g. the world globe) on a 2D plane (e.g. the world map), so that each vertex of the 3D mesh is assigned a correspondent 2D texture coordinate (Figure 26). The image texture is also called UV map, where U and V are the texture axes.

Figure 26. UV unwrapping. It consists in flattering a 3D object onto a 2D plane.

In the same way as the 3D space is defined by x,y,z coordinates, the unwrapped objects have U and V coordinates in a 2D system, describing the width and the hight, respectively. The UV coordinates help positioning the texture on the object. The U and V values are comprised between 0 and 1.

When producing totally artificial images, if we want to obtain the impression of realism, we have to introduce a large set of effects, such as light sources, that can be point, spot, directional or area lights which permit us to see the scene and can modify the scene's appearance, casting shadows consistent with the illumination, assigning optical properties to materials, fixing camera properties, employing ray tracing (a rendering method where the light rays are calculated from the surface back to the light source) or radiosity (a rendering

method which considers the light reflected by the various surfaces before reaching the eye) and so on.

Final rendering of all frames can be very time- and memory-intensive; it may take days or weeks to complete, so renderings are most of the times entrusted to a render farm (a bank of computers reserved for renderings).

After rendering, the 2D images are played at a rate of 24, 25 or 30 frames/second (24 is the minimum speed the human eye needs to see to successfully create the illusion of movement or a continuous variation).

Special effects

Rendering software may simulate a series of visual effects such as depth of field, motion blur, rain, smoke, fire, fog, dust, caustics (light interaction with uneven light-refracting surfaces), ambient occlusion (the attenuation of light in the less exposed regions), subsurface scattering (light reflecting inside the volumes of solid objects such as human skin), *etc.*.

Compositing

Very often the scenes created in Computer Graphics are rendered as different layers to gain control on different parts of the scene, such as background and different elements of the foreground. The process of grouping the layers to form the final image is called composition.

3.2 Computer Graphics software

As mentioned above, some of the 3D CG packages available are: the commercial Maya, 3D Studio Max and Softimage XSI (all from Autodesk), Cinema 4D (MAXON Computer GmbH) and the free, open-source Blender. These programs share the tools to perform all the steps described above for movie creation, divided mainly into modelling, animation, rendering. They display a GUI (Graphical User Interface) formed of various panels (windows), each with a distinct function: the 3D viewport, a list with the objects in the scene, the objects properties, for animation, panels for the creation of complex materials and textures, for scripting, *etc.*. They also provide a console window in which a feedback of the user actions is kept and errors appear.

4 Molecular motion

In chapter 3 **3D animation and rendering** the structures of the proteins were considered as low energy, unique, native states. Most proteins exert their functions through some extent of motion, which implies conformational changes (modifications in the structure). The simplest mechanism of conformational change in proteins is a hinge motion, in which two parts of the protein move rigidly with respect to each other (e.g. myosin motion along the actin filaments [Holmes1998]). Sometimes the domains are packed closely at the interfaces and the conformational changes are induced by shift of the domains. Such shift are made possible by small rotations of the side chains. Many proteins (enzymes) change conformation in response to the binding of a ligand or a cofactor. Usually, the active site is located in a cleft between two domains and the binding of the ligand induces the closure of the cleft over it. Some proteins (prion, amyloid) undergo very large conformational changes, others (serpins, gp41 of HIV/SIV) are involved in movements known as the stressed → relaxed transitions. These changes include different folding topology. High-level conformational changes of complexes of proteins include GroEL-GroES chaperonin complex that catalyses the protein folding [Xu1997a, Xu1998] and ATP synthase [Abrahams1994, Noji1997, Stock1999] which pumps protons out of the mitochondrium.

Despite recent advances in highly performant methods, it is very difficult to obtain direct information on conformational changes of molecules. However, several techniques shed light on the variability of conformations of single polypeptide chains, such as X-ray crystallography, NMR spectroscopy, molecular dynamics simulations, indirect techniques like FRET, and the many disparate microscopy techniques. All these methods contribute new and important information that can advance our interpretation of biology in action at atomic level.

Molecular dynamics (MD) technique is a computer simulation of physical movements of molecules allowing insights into molecular motion at atomic scale. Using force-fields (such as AMBER [Case2005], CHARMM [Brooks1983], GROMOS [Christen2005], GROMACS [Lindahl2001], *etc.*) containing sets of parameters for each atom, MD simulate molecules behaviours in a solvent. Simulations span from nanoseconds to microseconds and require large computational power, often lasting several CPU-days to CPU-years. These

trajectories can be then analysed with visualization tools (such as Chimera, PyMOL, VegaZZ [Pedretti2002], VMD, *etc.*).

Unlike molecular dynamics, **NMR spectroscopy** provides experimental information on molecules in solution, and thus free to move. The data are presented as a collection of conformations in no specific order, that can be considered as static 'images' of proteins caught during motion. An NMR collection contains usually 15-50 models.

X-ray crystallography provides different states of a protein captured in different conditions (with or without a ligand or in association with other proteins). Analysis of these structures can reveal conformational changes.

4.1 Morphing in Blender Game Engine

Our group, Scientific Visualization Unit, uses Computer Graphics tools to recreate the biological ambient using the available knowledge from experimental techniques, some of which were described above. Our collection of movies shows different biological environments, crossing 7 orders of magnitude from millimetre 10^{-3} m (capillary) to micron 10^{-6} m (cell) and to Ångstrom 10^{-10} m (atom). These movies are meant to make visible the invisible world with its crowdedness, movements, interactions, and can be used as scientific divulgation instruments for the large public (schools, museums) and scientists. Besides this, our research work is focused on two main aspects: elaboration of proteins' motions and visual representation of surface properties of molecules.

Starting from NMR data and assuming that if a protein exists in more than one conformation it should be able to transit between the different states, our group developed a system to elaborate proteins' motions using Blender. Initially, RMSD is performed between all models using SwissPDBViewer (SPDBV); the two conformations with the highest RMSD are selected and imported in the scene of Blender as two different positions of the same 'object' and set at different time (higher the time interval between the positions, slower the motion). The GE, equipped with physico-chemical mimicking rules (e.g. the bond length is fixed, only rotation around the bond axis is allowed), interpolates between the start and the end conformations. For our aims, the essential features of the engine are the collision detection, the control of rotation with the rigid body constraints, and the capability of baking (recording) the movements of objects as calculated during 'game playing'.

As the test material for our procedure, we used the 25 models of Calmodulin (CaM) stored in 1cfc.pdb file [Kuboniwa1995]. CaM is a well studied protein composed of 148 amino acids, comprising 1166 atoms (2262 including hydrogens), well conserved along the evolutionary scale [Baba1984]. CaM is formed of two globular domains (heads) connected by a flexible linker. Each domain is also mobile: it consists of four alpha helices, organized in two EF hand-Calcium binding motifs, which undergo a major transition upon Ca^{++} binding, but are also quite flexible in solution in the absence of this ion.

The procedure to obtain sequential ordering and motion of a protein follows the steps shown in Figure 27.

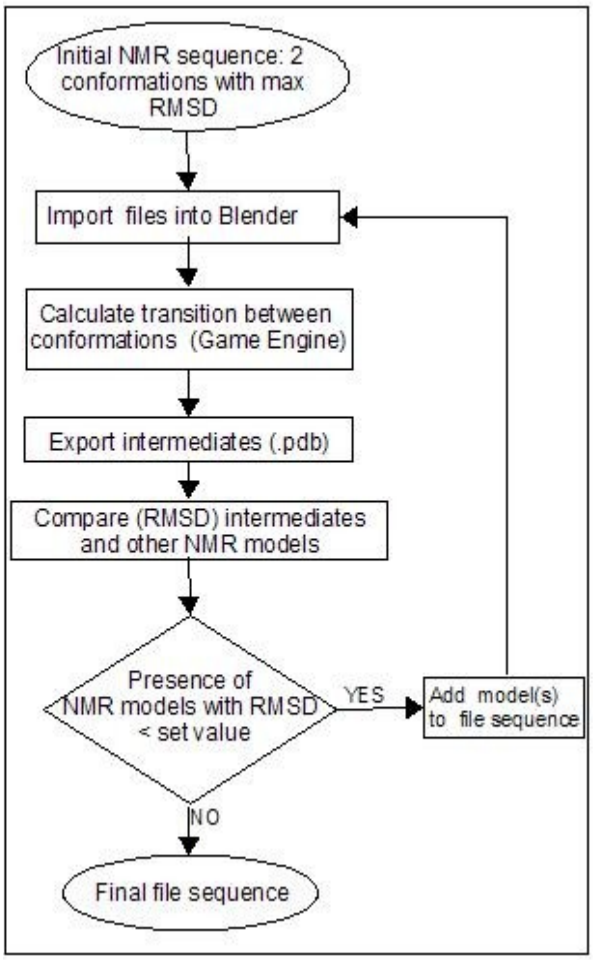


Figure 27. The workflow for morphing in Blender. It includes selection of 2 conformations with max RMSD, calculation of the transition using Blender GE, comparison of the intermediate conformations with the other NMR model and if the condition is satisfied, add a new model to the sequence, reiterating the entire process, until the final sequence is obtained.

The molecule is built in the 3D environment of Blender by creating a sphere for each atom (with its covalent radius or with a collision radius equivalent to the specific Van der Waals radius), and links (corresponding to chemical bonds, built using an amino acid library inserted in BioBlender) which are set as rigid body joints, allowing rotation around their own axis.

With these settings, the Blender GE is played and the position of all atoms

at all intermediate frames are recorded. These are exported in a series of files in .pdb format and compared by RMSD with all the remaining models in the original NMR collection. Data are plotted in graphic form (Figure 28), and any model found similar (RMSD ≤ 2 Å) to one of the intermediates calculated by Blender, is considered a step in the route between the two distant models.

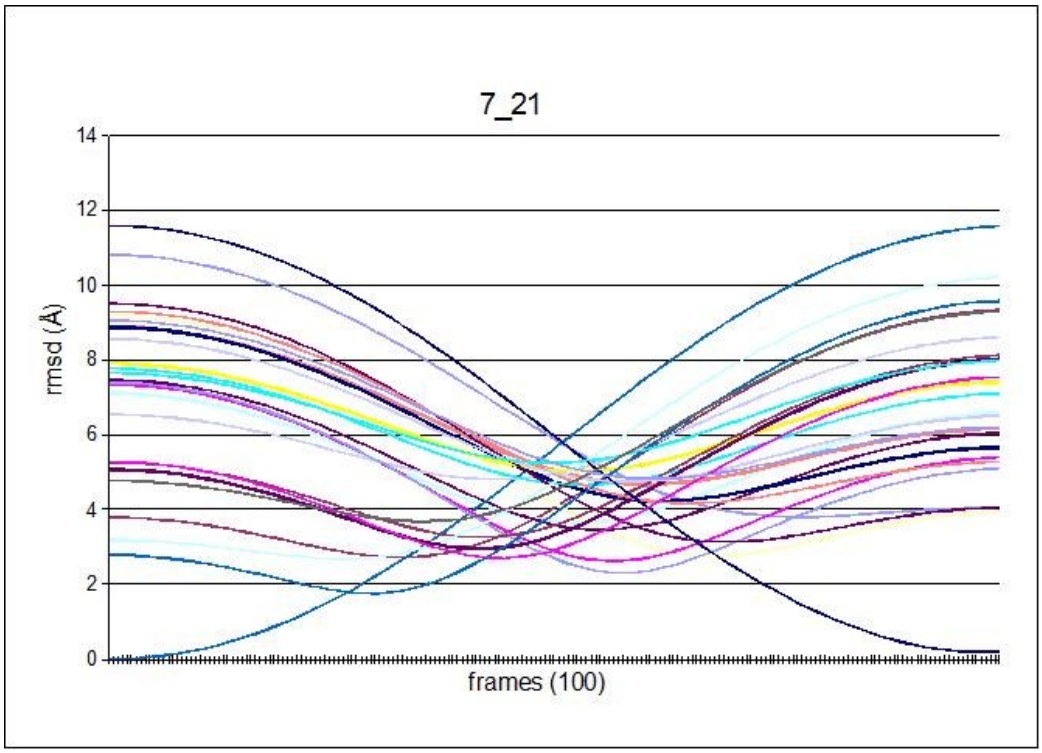


Figure 28. RMSD between intermediate conformations and original NMR models. The RMSD is performed between all the transition conformations calculated by Blender (frames 1-100) and the remaining models in 1cfc.pdb. The model with an RMSD ≤ 2 Å from any intermediate is introduced as a step between the starting conformations, 7 and 21.

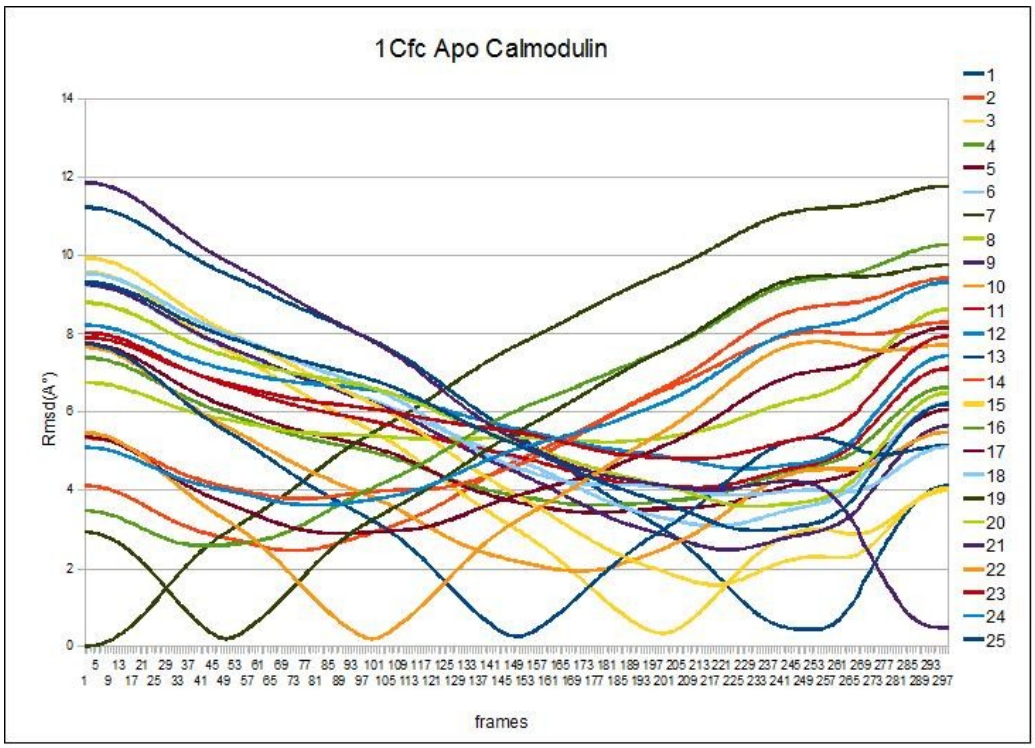


Figure 29. Sequence of models. RMSD between models of the transition and all models of 1cfc is performed; the result is a sequence of NMR conformations between 7 and 21.

The procedure is then repeated, this time introducing in Blender the start and the end conformations plus the ones found close to the path. By reiterating the process we are able to order some of the conformations (Figure 29).

Starting again and choosing other two different initial conformations, a map is built that allows all models of the NMR collection to be reached, starting from any other one. We plot the map in a 3D graph made with Jmol, in which all conformations are linked, see Figure 30.

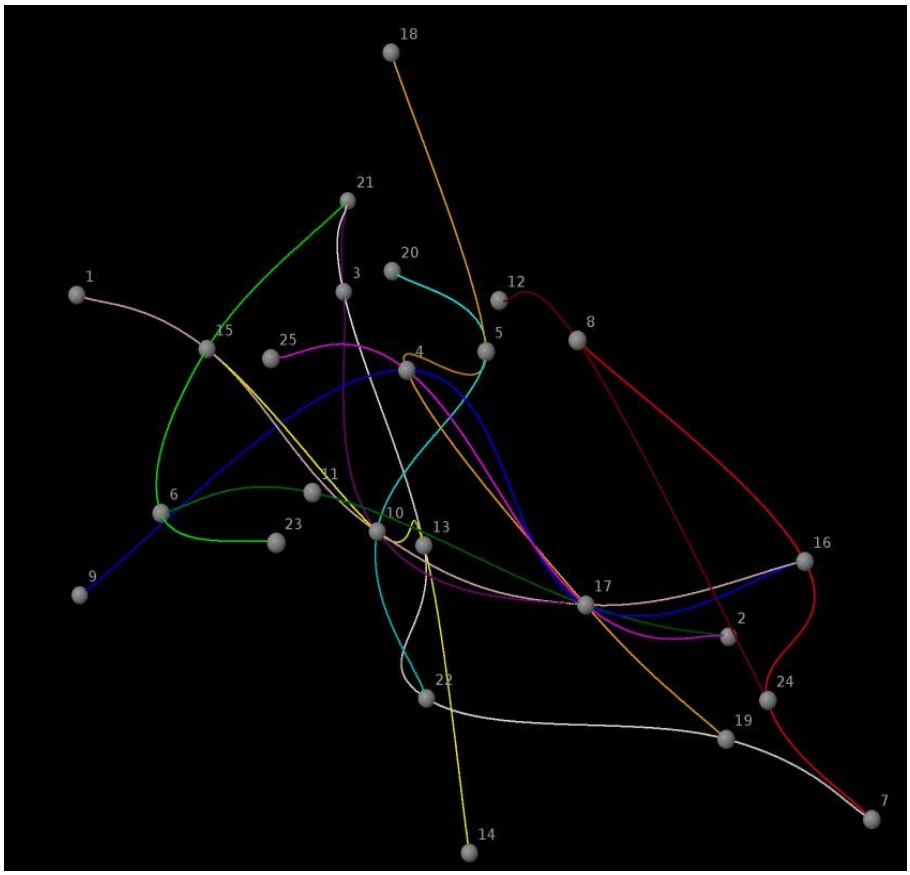


Figure 30. The navigation map of NMR file 1cfc. The image is a view of the map in Jmol. The numbers indicate the model in the NMR file and the colours various paths between the models.

Next, the question of the physical and chemical plausibility of the intermediate steps calculated by Blender GE is addressed: all the .pdb files of the sequence are exported and analysed with SPDBV, as shown in Figure 31.

Energy content of each conformation is evaluated by GROMOS 43B force field [Christen2005] included in SPDBV (script *spdbv_energy*) and plotted. A close examination of the contribution to total energy reveals that most peaks are due to minor geometrical distortions, mainly due to rotameric conversions, or to close proximity of atoms. Rotamers are manually adjusted, by inverting the names of the equivalent atoms involved, and distorted geometries are fixed through energy minimization, performed within SPDBV program using again the GROMOS force field.

Once validated, .pdb files are re-imported in Blender – one conformation every frame – so that it can be used to visually inspect the moving protein in a 3D environment.

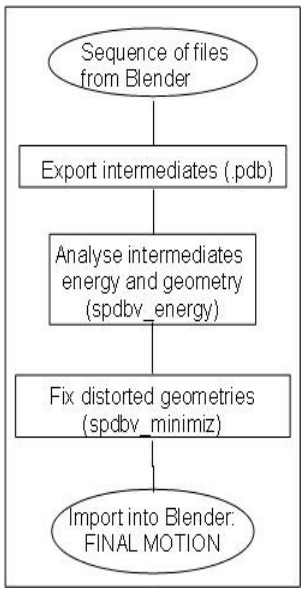


Figure 31. Intermediates evaluation. The intermediate conformations obtained with Blender are physico-chemical evaluated and re-imported in Blender for the final motion.

The sequence of files are the basis for the subsequent visual elaboration described in details in the **RESULTS** chapter of this thesis.

THE AIM OF MY THESIS

SciVis proposes a step forward in the direction of using bio-animation both as a divulgation and as a discovery tool. Our aim is to elaborate proteins' internal motion starting from structural data (as described in section 4.1 **Morphing in Blender Game Engine**), and to visualize surface properties of molecules in a more informative way. Both these tasks are done using Blender in conjunction with several scientific programs.

In the visualization study we propose a new visual code for the representation of the surface properties (electrostatic potential and molecular lipophilic potential) without involving colour. Visualizing them using features different from colour permits their simultaneous representation and their delivery in photo-realistic images leaving the utilization of colour for the description of other biochemical information. Using real-world features, the surface characteristics can be easily interpretable without the need of a legend. The main idea of the proposed visual mapping is to exploit perceptual associations between the values to be mapped and visual characterization of real-world objects. Ideally, by using already established perceptual association, the viewer will be able to understand the provided information more naturally.

3D creation software tools are appropriate to accomplish this purpose as they provide several possibilities to express surface properties: choice of material, transparency, incandescence/glow, ambient colour, diffuse, bump, displacement, specular shading, reflection, lights, *etc.*. Combining these attributes, it is possible to create realistic images, transmitting both visual information (such as surface properties and consistence) and tactile sensations.

For MLP mapping, two opposite surface characterizations able to convey a sense of affinity to oil or to water were selected. In our real-world experience, a very smooth and reflecting surface (like wax) is completely impervious to water but can be easily coated by oil. The opposite visual feedback is associated to grainy, crumbly, dull surfaces (like clay bricks or biscuits) which can be easily imagined being soaked in water. These considerations led to the showing of highly lipophilic areas as shiny, smooth material and of highly hydrophilic areas as dull and rough. These kind of visualizations that resembles our world are called in CG “photo-realistic representations”.

While the MLP value, due to its very short range effect, is only observable

on the surface itself, electrical phenomena are associated to the idea of an effect projected in the volume surrounding a charged object, and able to affect other objects (like in the high school textbook favourite amber rod attracting paper bits). Field lines are a common way to describe the effect of the electrical field. EP value is therefore represented by showing particles (drawn as short lines), moving along the path defined by field lines, from the positive to the negative end. The particles concentration is proportional to the total charge of the molecule, visualizing a high concentration of particles in areas where the electrical fields is strong.

We show MLP and EP as surface characteristics and animated particles, respectively, permitting their simultaneous visualization on moving molecules. This new method, by introducing the animated visualization, highlights the changes in surface features during proteins' conformational transitions.

TOOLS: PROGRAMS AND SCRIPTS

1 Programs

Maya 7 Unlimited/Autodesk – a commercial, cross platform software package for 3D animation. It includes two internal programming languages, MEL (Maya Embedded Language) similar to TCL and Perl, and since Maya 2008 also Python. Maya has a powerful particle system. Particles can be dealt with either as a unique object (*per object*), or on a *per particle* basis. Attributes such as position, velocity, acceleration, mass, colour, opacity *etc.* can be assigned to particles as per object, and in this case all particles have the same value of the attribute, or they can be assigned as per particle and in this case each particle has a different value.

Several internal renderers are available with Maya: Maya Software Renderer, mental ray for Maya Renderer, Maya Vector Renderer, Hardware Maya Renderer, each one with specific advantages.

RenderMan for Maya 1.0/Pixar [RenderMan for Maya] – a plug-in for Maya, an add-on module that extends Maya's capabilities customizing it for a specific rendering. It is specialized in high quality photorealistic renderings. Pixar's RenderMan is a high quality renderer, fast, efficient for handling complex images, it has additional features like deep shadows (offers high-quality shadows), special effects like motion blur (the image can be blurred if the object moves), ray-tracing effects such as global illumination (effects that create subtle soft shadows in a scene). This plug-in, among the internal renderers of Maya, has the capacity of rendering blobby particles (particles handled as *metaballs*, spheres that blend together giving the impression of a single surface that includes them, when they are close to each other) with colour as per particle attribute, blending colours as a consequence of blobby particles blending characteristic.

Blender 2.49 – a free, open-source, cross platform suite of tools for 3D creation. Mesh creation and assignment of colours to vertices, mesh unwrapping, UV texture mapping, Node Editor and particles are some of the features used for this project and are described in details in section 2 **Results in Blender**. Its functions can be accessed through the GUI or scripting, through its API's (Application Programming Interface), using Python scripting language. APIs are a

set of rules and specifications used to communicate with the software from scripting, without using the interface. The renderings are performed by its internal renderer which combines all the features needed to obtain photo-realistic images.

Blender 2.5 – a big change in the internal structure of Blender including the major advantage that now all Blender's functions can be reached through the API's. It incorporates a powerful Game Engine (GE), usually used for video games and special effects. GE instances the Bullet physics library [Bullet Physics Library], an open-source software for multi-threaded 3D collision detection, soft body and rigid body dynamics usually used for games and visual effects. Blender GE uses a system of graphical "logic bricks" (a combination of "sensors", "controllers" and "actuators") to control the movement and display of objects in the game.

PyMOL 1.2r3pre – an open-source, user-sponsored, Python-enhanced molecular graphics tool, used for visualization of .pdb files (proteins, nucleic acids, other macromolecules). Its internal renderer uses ray tracing to produce high-resolution images by casting shadows and smoothing the sharp edges. It calculates the electrostatic potential through APBS plug-in and generates the 3D mesh of the molecular surface for the molecule. The obtained geometry is exported in a VRML (Virtual Reality Markup Language) format, easily read by 3D software tools.

PDB2PQR-1.6.0 [Dolinsky2004, Dolinsky2007] – a Python software package that automates many of the common tasks of preparing structures for continuum electrostatics calculations, providing a platform-independent utility for converting protein files in PDB format to PQR format. It assigns partial atomic charge to every atom in the .pdb file according to different force fields (AMBER94 [Wang2000], CHARMM27 [MacKerell1998], PARSE [Sitkoff1994] or TYL06 [Tan2006]) and saves a .pqr file in which the occupancy and temperature columns are replaced by atomic charge and radius, respectively. It also adds missing hydrogens, calculates pKa values and generates an input (.in) for APBS calculations. The .in file stores the information on the 3D dimensions of the protein, the ionic concentration of solvent, biomolecular and solvent dielectric constants. Ionic concentration of 0.150 mol/l NaCl, biomolecular dielectric constant of 2 and solvent dielectric constant of 78.54 (water) were used for our calculation. It is available both as a web service and as a stand-alone program.

APBS-1.2.1 (Adaptive Poisson-Boltzmann Solver) [Holst2000, Baker2001] – a software for evaluating the electrostatic properties of nanoscale biomolecular systems, through solution of the Poisson-Boltzmann equation. APBS takes as inputs a .pqr and an .in files and calculates the electrostatic potential in every point of a grid in the protein space, which is output as a .dx file. The electrostatic potential can be then visualized with VMD or PyMOL programs.

OBJCreator – a custom software that maps the MLP values on the mesh and exports an .obj file. It takes as inputs the mesh (.wrl) created by PyMOL and the .dx file containing the values of MLP, created by pyMLP.py (see below). For every vertex of the mesh, the correspondent grid-cell is identified and the value of potential is calculated using trilinear interpolation. The resulting .obj file stores information about the position of each vertex of the mesh and its correspondent MLP value. The mapping process used by this program is the same used by scivis.exe, described next.

scivis.exe – a custom software written in C++ used to calculate the field lines and to export them in a ASCII file to be imported in Blender. This tool imports the 3D surface (.obj) and the Electrostatic Potential grid (.dx) calculated by the APBS program. The computation of the field lines is a multi-step process described in details in section 2.3.1 **EP calculation**. Briefly, EP values are mapped on the 3D surface by assigning the correspondent EP value, calculated using trilinear interpolation on the .dx grid data, to each vertex of the 3D mesh; a gradient grid is calculated in the volume containing the molecule; an automatic selection of areas with high values of EP is done and the corresponding field lines are computed for these areas using the gradient field.

When used as primary application, in addition to the described features, scivis.exe provides visual feedback for all its processing steps. It is possible to visualize the molecular surface, the EP grid, the gradient grid and the field lines. Different parameters and visualization modes are available in order to better understand the various data involved in the processing. For example, it is possible to restrict the visualization of the EP grid or of the gradient field only to a certain interval of values; it is also possible to change visualization colour ramps and to inspect the data in small parts of the computation volume. Most of the processing parameters can be changed in the interface and affect the processing in real-time. Thanks to the visual feedback, these options have been particularly useful when choosing the parameters for the processing of the EP values and

the generation of field lines.

For the static representation of the EP, pyramids (with the tip indicating the negative charge) are created along the field lines according to two parameters that can be set by the user: the size of the pyramids and the distance between them. These 3D objects are exported in an .obj file.

For the 3D interactive representation of EP, the field lines are exported as .json file; EP is visualized as comets flowing along field lines from positive to negative charges.

PRODRG2 [Schüttelkopf2004] – a free web server that provides MOL2-format files that contain bonding information useful for ligand parametrization. This file is required by PDB2PQR for adding charges to oligosaccharide chains. It is used along with the .pdb file of glycoproteins.

2 Scripts and scripting language

Python 2.6 – an interpreted, interactive, object-oriented, extensible programming language with a clear syntax. In this project, Python has been used in different stages, both as a scripting component of various software tools (like Blender and PyMOL) and as a stand-alone scripting language.

pyMLP.py [Broto1984, Laguerre1997] – a Python script written and kindly provided by Julien Lefeuvre (available from http://code.google.com/p/pymlp/source/browse/trunk/pyMLP.py); it contains a library of atomic lipophilic potential for every atom in proteins based on its chemical position (we added the values for sugars and nucleic acids) and it calculates the Molecular Lipophilic Potential (MLP) in every point of a grid in the protein space according to various formulae such as Fauchere [Fauchère1988], Dubost [Audry1986], Brasseur [Brasseur1991], *etc.* (we introduced Testa formula [Gaillard1994]). The grid step can be changed by the user to cope with the protein size and computer performances (in terms of memory occupancy and calculation time); the default is 1 Å.

MLP.py – a custom Python script used within the Blender integrated scripting engine to obtain image textures. It imports the .obj files into a Blender scene and converts the MLP values into vertex colours (levels of grey) assigning a colour to each vertex. The 3D surface is then unwrapped to obtain a UV texture parametrization. A texture map is built by baking the computed per-vertex colour values in the texture image.

texture.py – a Python script used within Blender, that builds a suitable material, including the texture images generated by MLP.py, and assigns it to the 3D meshes imported in the 3D scene.

import_curves.py – a Python script used in Blender to read the ASCII file where the field lines are saved; it creates NURBS curves in the Blender scene and associates a particle emitter to the positive end of every curve.

render.py – another Python script executed in the Blender integrated scripting engine during the final step of rendering. For each frame it selects the corresponding mesh from all those present in the scene, it assigns the appropriate image textures to it and renders the scene. It finally saves every rendered image as a .png or .exr file.

cycle.sh – a shell script for Linux that reads the .pdb files and executes external programs and scripts such as PyMOL, PDB2PQR, APBS, pyMLP.py and OBJCreator, saving the two .dx files with EP and MLP values and the .obj file with the MLP values mapped on the surface of the protein and stored in the V column.

RESULTS

1 Early attempts with Maya-Autodesk

In the early stages of our research, motion and visualization studies were done using small, simple molecules such as Triazine, Bitucarpin A and Alanine dipeptide. The first attempts to visualize macromolecules' properties were elaborated with Maya and referred to general properties (hydropathy and EP) applicable to all proteins and individual properties (fluorescence of GFP or high-energy bond content of ATP).

1.1 Atomic representation

The simplest representation of molecules is the atomic representation. Using custom scripts written in MEL by a former lab member, Yuri Porozov, the atoms identities and positions were imported from .pdb files into the virtual space of Maya as spheres with the radii proportional to the atoms covalent radii and coloured using the standard CPK code. This basic visualization of a molecule reveals only the atom identities and the covalent bonds between them, drawn as bones which behave like chemical bonds, have fixed length, and are constrained by codified rules.

An example is Triazine (2-chloro-4-methoxy-6-[(R)-1-phenylethylamino]-1,3,5-triazine), a small molecule used as a chiral solvating agent in NMR spectroscopy studies. It is composed of 31 atoms, in a relatively simple structure made of two rigid disks connected by a bridge. Triazine has been subject to dynamical simulation studies, [Alagona2007] that have revealed the energy landscape for all possible conformations that its two rotating bonds can assume. For this reason we chose it as the initial test molecule of our chemical Maya system for molecular motions (Figure 32) [Porozov2007].

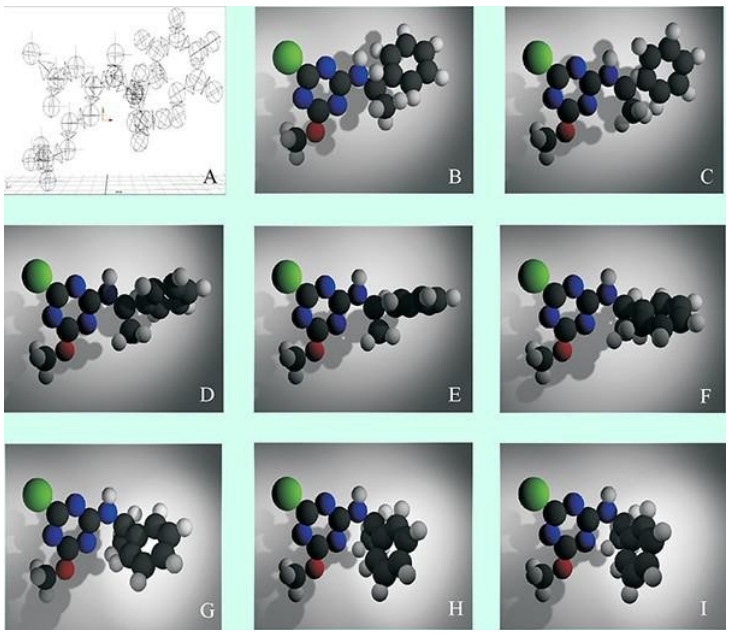


Figure 32. Triazine representation. Triazine motion is calculated by a skeleton enriched with physico-chemical rules **(A)** and some intermediate frames are presented here **(B-I)**. See the movie on http://vimeo.com/7391082

1.2 Surface and properties representation

To visualize molecules as surfaces, every atom was imported in Maya as blobby particle. Blobby particles are particles handled as *metaballs*, spheres that blend together giving the impression of a single surface that includes them, when they are close to each other (Figure 33). The visual result reminds of mercury beads which appear to meld into each other as they get closer together. As most biomolecules contain very large number of atoms, we use the Particle feature to create them in the 3D space of Maya. Particles are 'light entities' in terms of processing power in comparison with objects. In other words, large movements (such as bends on a hinge) and relative movements (of the object in space) can be imposed and calculated very fast.

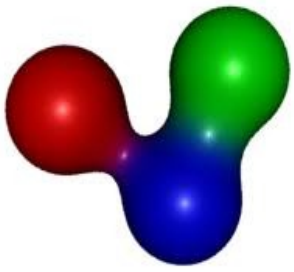


Figure 33. Metaballs. The spheres blend into each other at short distance.

Information about atoms identity is delivered using colours according to CPK code. Alternatively, we use a grey level code with values of grey proportional to the standard CPK. Atoms colours are a *per particle* attribute.

1.2.1 Hydropathy

The first attempts to display visually the surface properties were concentrated on hydropathy, considering the predominant molecule's characteristic: mostly hydrophilic or mostly hydrophobic. For these tests we chose some representative molecules: Bitucarpin A [Alagona2004], alanine dipeptide [Wang2004] and ATP as hydrophilic molecules and cholesterol as an amphipatic, mostly hydrophobic one.

Two real-world surface characteristics were chosen to represent hydropathy property: roughness for hydrophilicity and sliminess for hydrophobicity.

Bump and displacement are the two possible ways to achieve the impression of roughness in Computer Graphics. A *bump* texture is a feature applied to the surface at rendering, making the surface appear rough or bumpy, without altering the shape (i.e without moving the vertices of the mesh): it works as an optical effect obtained by modifying the trajectory of the reflected light. Bump mapping [Blinn1978] is a rendering technique generally used to represent very small scale geometry like scratches, roughness or graininess. By contrast, displacement mapping [Cook1984] does modify the geometry of an object in order to specify surface relief. For effective displacement, the object should have a very fine mesh composed of many polygons.

Bump and displacement texture maps are based on values from a grey scale image; bright areas appear to protrude from the surface while dark areas appear to sink into it. Some examples of textures are shown in Figure 34.

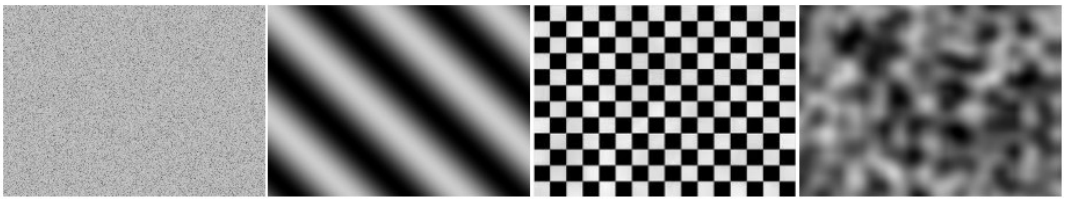


Figure 34. Examples of default textures in CG software. These textures can be used both for bump and displacement mapping.

The coarse-grained impression was obtained using a dull material. Three different visualizations for hydrophilicity were tested:

- rough (bumpy) surface and CPK code for atoms as for Bitucarpin A and ATP (Figure 35 a and b, respectively);
- rough (bumpy) surface and grey level code for atoms as for Alanine dipeptide molecule (Figure 35 c);

- rough (displaced) surface and grey level code for atoms as for ATP (Figure 35 d).

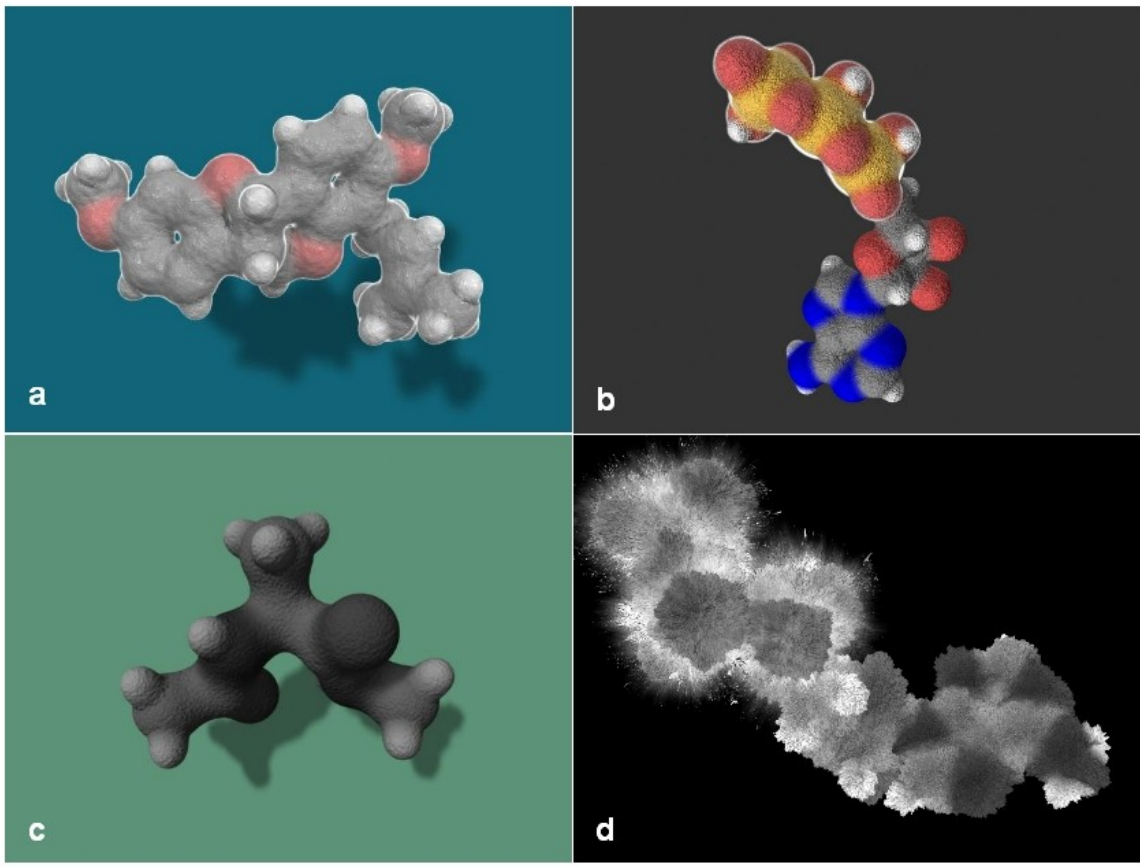


Figure 35. Hydrophilicity representation study. (a) Bitucarpin A, bump mapping and CPK code for atoms identity; **(b)** ATP, bump mapping and CPK code; **(c)** Alanine dipeptide, bump mapping and grey levels for atoms identity; **(d)** ATP, displacement mapping and grey levels for atoms.

Cholesterol is a strongly hydrophobic molecule (except the hydroxyl) and was represented as a slimy surface without considering the hydrophilic influence of oxygen. The sliminess is given by a shiny material. For the representation of atoms identity the CPK colour code and the grey level code were used.

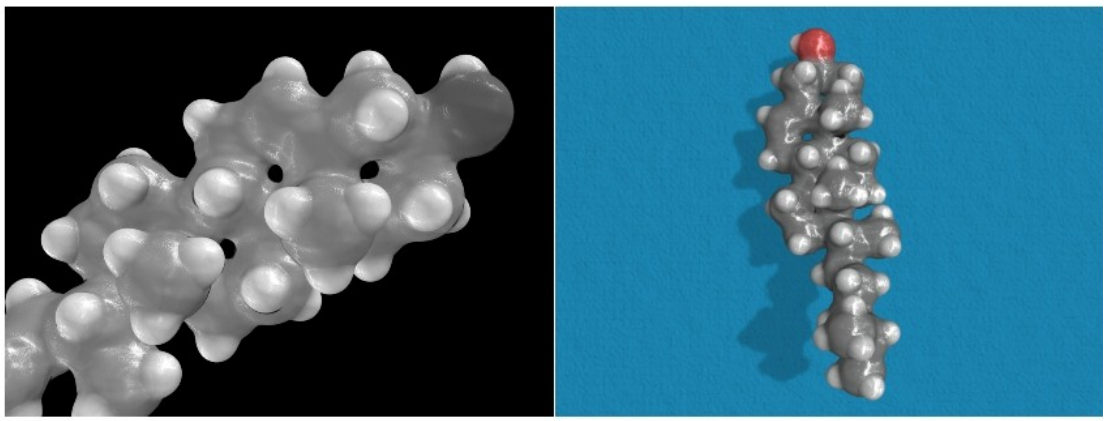


Figure 36. Hydrophobicity representation study. Cholesterol visualization as shiny material and atoms identity as **(left)** levels of grey and **(right)** CPK code.

1.2.2 Fluorescence

The representative model for the representation of fluorescence feature is GFP (Green Fluorescent Protein). It is a naturally fluorescent protein originally isolated from a jellyfish (*Aequorea victoria*) which emits green light when irradiated with UV light. The cromophore is the part of the protein responsible for the fluorescence emission and it is situated in the centre of the molecule. The GFP atoms (from 1gfl.pdb [Yang1996]) were rendered as blobbies to form a surface. Its fluorescence is visualized by using a dull material with green ambient colour and a green light emitting from the centre of the protein (where the cromophore is located), as shown in Figure 37.

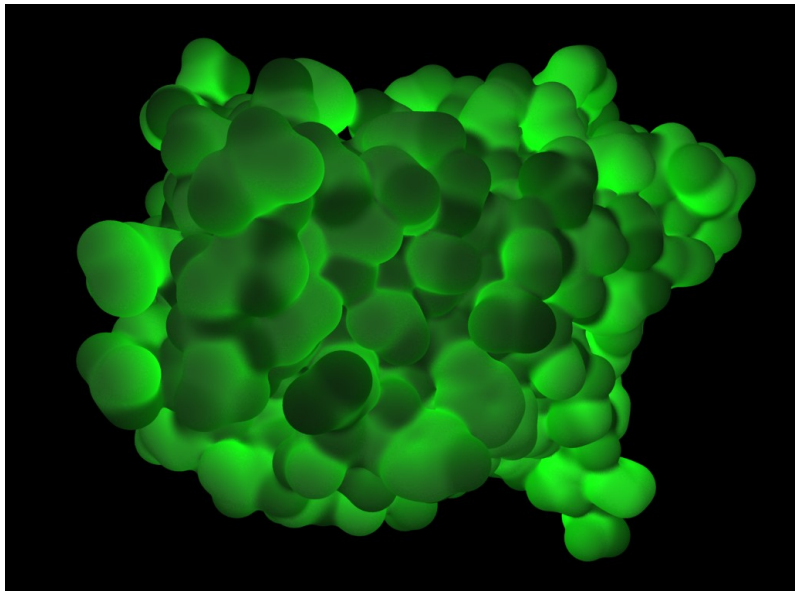


Figure 37. GFP representation.

1.2.3 Energy content

ATP (Adenosine Triphosphate from 1xsc.pdb [Swarbrick2005]) is the direct energy source for the majority of cellular functions which makes it suitable to study this characteristic. It works as a chemical battery, storing energy and releasing it when and where required.

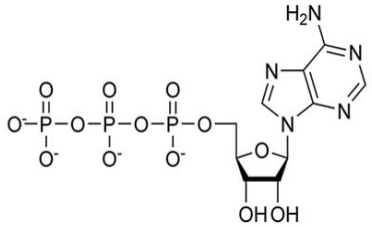


Figure 38. ATP chemical structure.

Chemical energy is stored in the ester bonds between phosphates, with the great amount of energy (7 kcal/mole) in the bond between the middle and the outermost phosphate groups. The terminal phosphate group in particular is frequently split off by hydrolysis, being transferred to other molecules or water and releasing energy required for synthetic reactions (Figure 38). These covalent bonds are known as “high-energy” bonds.

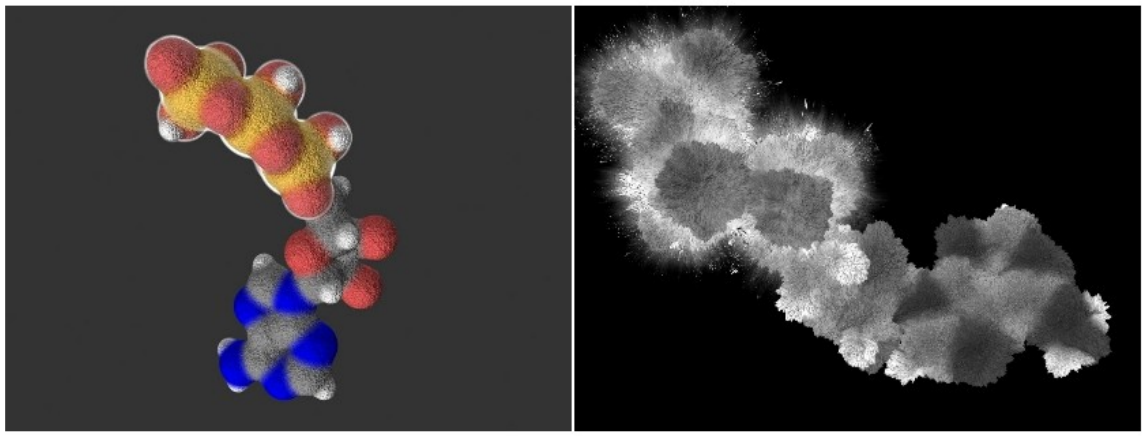


Figure 39. ATP representation study. (left) bump mapping for hydrophilicity and high value of incandescence for energy content, CPK for atoms identity; **(right)** displacement mapping for hydrophilicity and displacement combined with high value of incandescence for energy content, grey level code for atoms.

Besides the bond energy content characteristic, ATP is a hydrophilic molecule. For the visual representation of ATP, the two features were considered separately. The atoms identities and positions were imported into the virtual space of Maya as two different groups of particles superposed: the entire ATP was modelled as blobby particles and the three phosphate groups were modelled

as overlapping clouds (another type of particles). Two different shaders were created: one for the hydrophilicity and one for the “high-energy bond” characteristics. As described in chapter 3 **3D animation and rendering**, a shader (material) is a collection of attributes that define colour, shininess, transparency and other surface characteristics.

Two different representations for ATP were tested (as shown in Figure 39). The hydrophilicity, a property of the entire ATP, is represented as a coarse-grained surface, created with bump mapping (Figure 39 left) and displacement mapping (Figure 39 right); the shader gives the impression of a molecule that can easily interact with water. A second shader is applied to the three phosphates in order to display in a visible way the chemical energy stored in these covalent bonds. The high-energy content is represented by high value of incandescence (like an emitting light source), as shown in Figure 39 left or by displacement and high value of incandescence, as shown in Figure 39 right. The atoms identities are highlighted using the CPK code (Figure 39 left) or the grey level code (Figure 39 right). ATP was the first molecule with two of its characteristics represented simultaneously: hydrophilicity and high-energy bonds.

The representations described above give an overall view of the characteristics of some molecules. The particularity of our representations is that the molecules are always presented in motion: conformational changes (Triazine, Bitucarpin A, Alanine dipeptide) or vibrational motion (ATP, Cholesterol) calculated by interpolation between different conformations or simple rotations of the entire macromolecule to inspect better their structures, as in case of GFP. The animated representations, visible on our website or Vimeo, are more informative than the static images, providing insight into the three-dimensional structure and the flexibility of the molecule.

1.2.4 Glycoproteins

Glycoproteins are proteins with oligosaccharide chains covalently attached to them (at glycosylation sites). The challenge in visualization of glycoproteins is to transmit visually the different chemical natures of protein and sugars. gp120 is the first glycoprotein we used in this attempt. gp120 is derived from gp160, the envelope glycoprotein of HIV (Human Immunodeficiency Virus) and SIV (Simian Immunodeficiency Virus); gp160 is a homo-trimeric complex, in which each chain is cleaved, during transport to the surface of the infected cell, into two fragments

known as gp120 (protein on the surface) and gp41 (transmembrane protein) [Allan1985, Veronese1985, Center2002]. gp120 mediates the first contact between the virus and the target cell, by binding to the CD4 receptor [Dalgleish1984], and then to the specific co-receptor.

The crystal structure of unbound SIV gp120 (2BF1.pdb) [Chen2005] contains only the core of one monomer of the protein; flexible parts were removed to facilitate crystallization: the V1V2, the V3 and 220-228 loops and N and C termini. Of the oligosaccharide chains (OS), the most part was excluded, leaving max 5 OS monomers at each glycosylation site. Our aim was to build the entire gp120 monomer, including loops and integral OS (Figure 40). The V3 loop (1CE4.pdb) was positioned by overlapping the cysteines at the basis of the loop with the corresponding cysteines of the crystallized gp120. Modelling of the other loops was based on their aa sequence using ChemOffice and performing energy minimization. The 3D structure of OS were built according to their composition in gp120 of HIV [Geyer1988], using ChemOffice for proper bonding of monomers and for energy minimization.

For the visualization of the different chemical nature between the protein and the sugars surfaces, atoms were imported in Maya as blobbies and two different shaders were used: a dull material with a soft bump and a bright edge created with incandescence to represent protein atoms and a dull material with displacement to indicate sugars hydrophilicity. The atoms identity was revealed by softened CPK colour code.

As stated above, these representations are only some attempts to visualize molecules properties in CG without using the red-blue range of colours. Concluding we can say that the work was not entirely finished as the representation of electrostatic potential is missing and hydropathy is visualized only as an overall property of the molecule rather then as locally calculated.

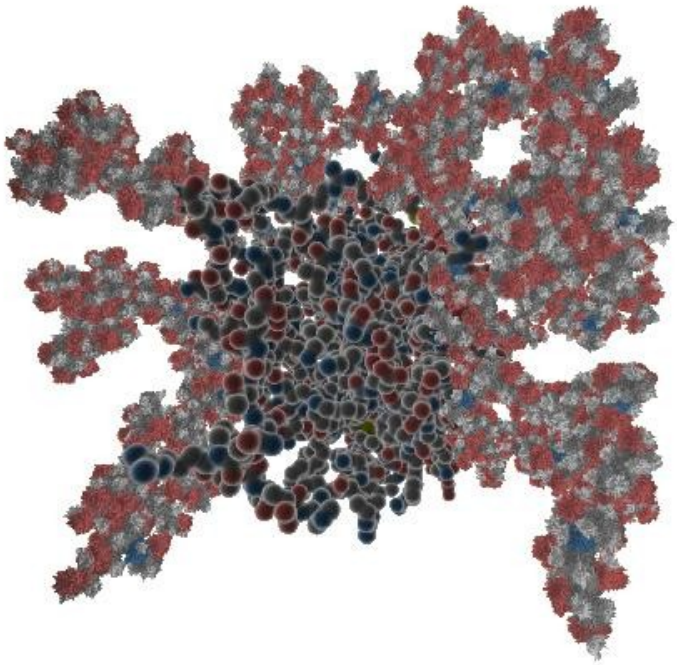


Figure 40. gp120 visualization. The flexible loops and the long branched sugars are missing for the crystal. We built the missing parts and displayed the monomer protein with a soft material and the sugars with displacement mapping. For an animated version, see http://vimeo.com/15800994.

Maya to Blender

In order to achieve our purpose, we faced some limitations, regarding the possibilities to customize some Maya features for our requirements. This was overcome by using Blender, an open-source software that permitted to find and create appropriate features. Being open-source, Blender is a more suitable tool for us to overcome the limits we had using Maya.

2 Results in Blender

The aim of our research is the visualization of the proteins motions with their surface properties. When showing proteins in motion as a rendered animation, every second of the resulting movie contains 24-30 images (we use 25 frames per second, which is one of the standard video speed). Because at every frame the atomic coordinates change, also the surface features (shape, EP and MLP) change accordingly, and must be recalculated. This implies a very large amount of calculations, but allows the elaboration of a sequence of images that is coherent from frame to frame, thus giving the impression of continuity.

In the production of animated molecular movies representing proteins in motion, the steps of object creation, surface calculation and data manipulation for both EP and MLP are elaborated independently using both scientific and CG programs to obtain the series of frames compositing the animation that include

this information.

2.1 Molecular surface representation

Solvent-excluded surface (Connolly) defines the space occupied by the molecule as an object, and is a convenient surface to display hydropathy and EP. Various software such as VMD, Chimera and PyMOL calculate Connolly surfaces and export them as ASCII files such as .obj and .wrl. The .obj file format, developed by Wavefront Technologies defines the geometry and it contains information about vertex coordinates, normals, UV texture coordinates and the faces that make each polygon of the mesh. An .obj file may access an external .mtl file that contains definitions of various material types. VRML (Virtual Reality Markup Language) file, with the extension .wrl, is a text file which contains the information about the vertices and the edges of a 3D object, along with surface material characteristics: colour, shininess, transparency, UV mapped textures, *etc.*, including also the view point coordinates for the initial view of the 3D scene.

Surfaces calculated by VMD and Chimera programs often contain some disjoint components (small, mostly internal surfaces disconnected from the outer surface of the molecule), as shown in Figure 41 that are difficult to handle; also, the .wrl format of VMD was incompatible with Blender (the modifications introduced by VMD in the formatting of the file were not compatible with the importer of Blender).

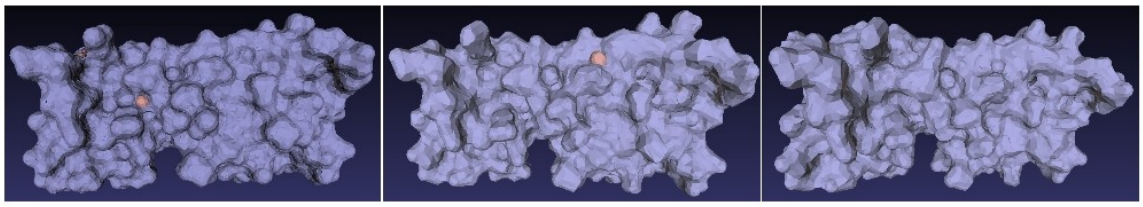


Figure 41. Examples of Connolly surfaces. The surfaces are calculated using VMD **(left)**, Chimera **(middle)** and PyMOL **(right)**. Notice the disjoint surfaces highlighted in orange on surfaces calculated using VMD and Chimera.

Disabling the creation of disjoint surfaces, continuous surfaces can be obtained in Chimera; however, these surfaces are not as regular as the surfaces obtained with PyMOL, as shown in Figure 42.

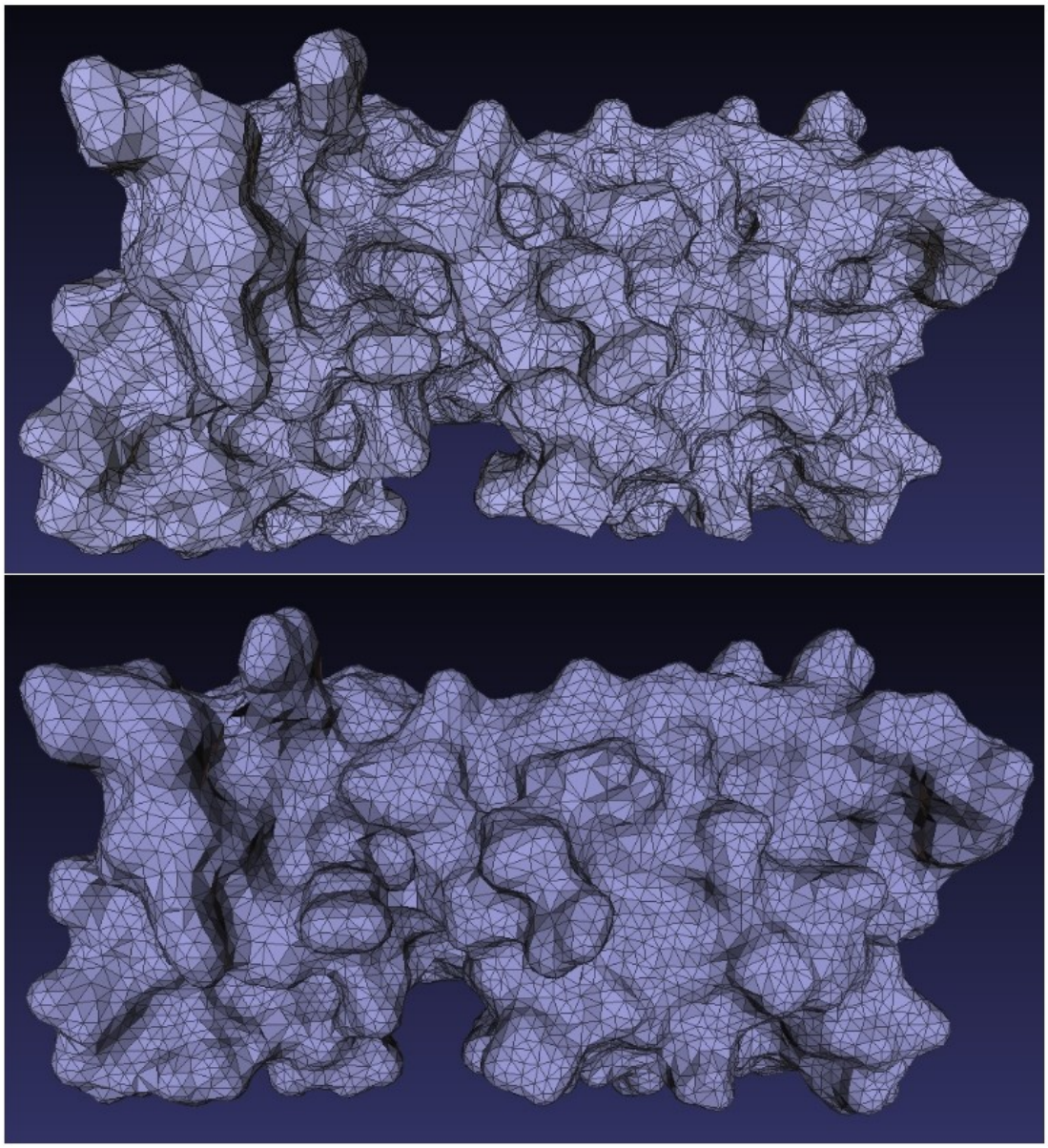


Figure 42. Irregular and regular surfaces. (top) irregular surface composed of different size triangles, calculated using Chimera; **(bottom)** regular surface with triangles of same area, calculated using PyMOL .

PyMOL was chosen to calculate molecular surfaces because the surfaces created by this software have a regular triangulation even at low polygon resolution and it is afflicted at low level by the problem of internal disjoint surfaces. In the 3D mesh used in the example reported in Figure 42 (bottom) and in other tests with wider range of dimensions (number of polygons between 4.5 and 50 thousands), all the triangles have similar areas. The surfaces are calculated starting from the .pdb files and the mesh is exported by PyMOL as a .wrl (described above).

2.2 Molecular Lipophilic Potential

2.2.1 MLP calculation

For the visualization of hydropathy we use the Molecular Lipophilic Potential calculation method (see chapter 2.5.2.1 **Hydropathy**), more appropriate for large molecules like proteins than the substructure approaches, mainly used for small molecules. The MLP calculation and rendering are done in several steps, indicated in the scheme in Figure 43.

Starting from a .pdb file, the Connolly surface is calculated using PyMOL and the MLP calculation is done using pyMLP.py (Figure 43, upper part). This Python script calculates the lipophilic potential in every point of a grid that contains the protein; the grid is obtained by sampling the space according to a parameter (set by the user) called “grid spacing”. The values are exported in a .dx file, in which the header contains information about the grid and the data storage (the grid origin, the grid spacing and the number of points on each axis). The script contains a library of atomic lipophilic potential values for every atom, based on its chemistry (only for proteins), and several formulas for MLP calculation, such as Fauchere, Dubost, Brasseur, Buckingham; however it does not support the Testa formula,

$$MLP(r) = \sum_i f_i \cdot e^{\frac{-|r - r_i|}{2}} \qquad (5)$$

where r is any position in the protein space, f_i is the atomic lipophilic potential for the atom i and r_i is the position of atom i.

This formula is an atom-based function using Broto fragment scheme and an exponential distance function, appropriate for protein calculations. Therefore, we modified pyMLP.py to include the Testa formula. Since, as described above, the library only contains aa, we also added a library of atomic lipophilic potential values for sugars and nucleic acids. The MLP accuracy depends on the grid spacing (the lower the grid spacing, the more accurate the calculation); the default is set at 1 Å, a dimension comparable to the mean size of the triangle edge of the 3D mesh, calculated by PyMOL. We selected this combination as a good compromise between MLP data, mesh triangulation, computer memory and calculation time. However, it can be changed by the user.

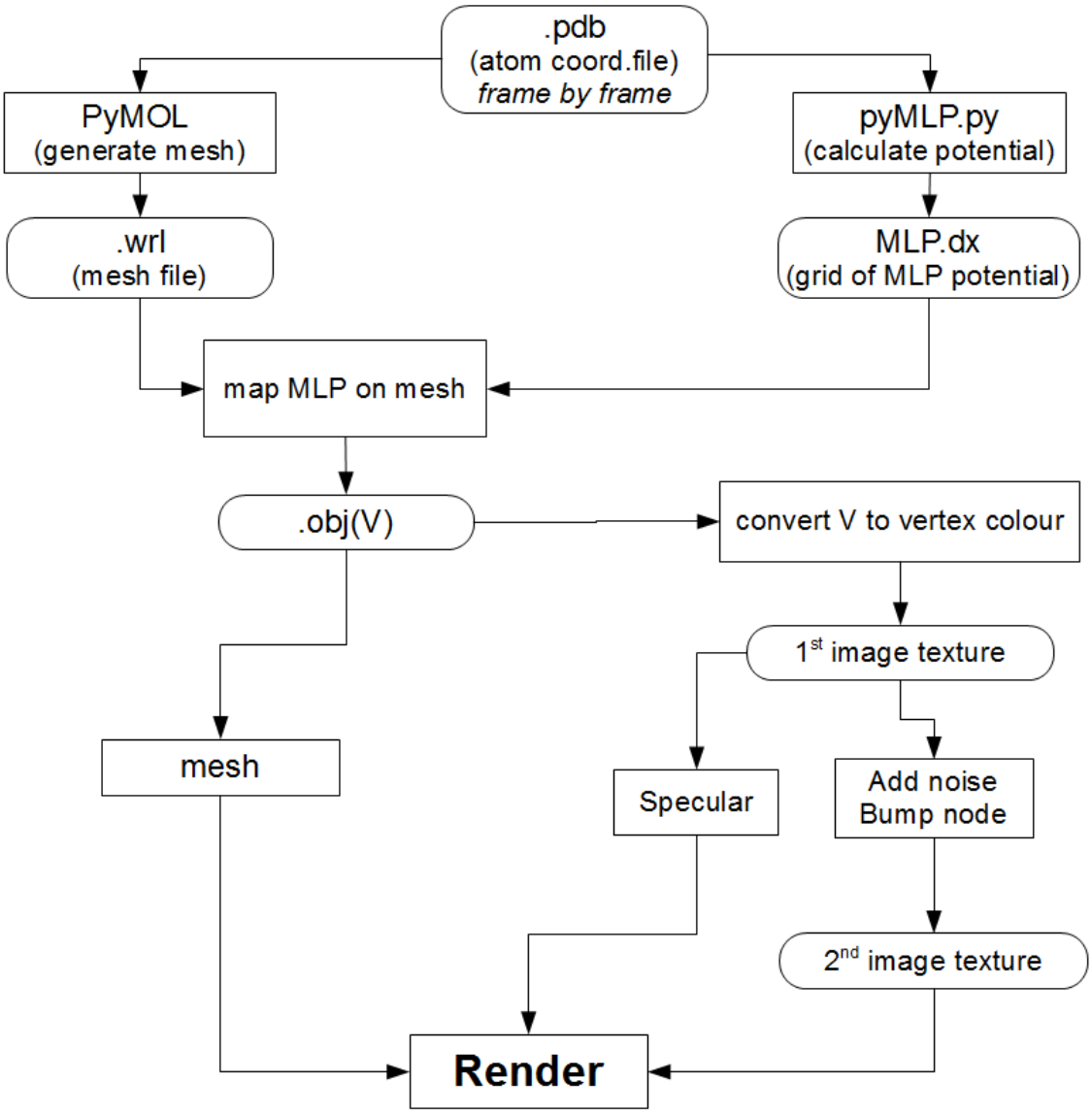


Figure 43. MLP calculation and representation workflow.

2.2.2 MLP rendering

Starting from data calculated on atomic basis, we propose a more detailed representation. Our visual code for hydropathy includes specific representations according to the values calculated: smooth and shiny surfaces for hydrophobic regions and rough and dull for hydrophilic ones. To obtain the roughness and shininess impressions, the bump mapping (described in section 1 **Early attempts with Maya-Autodesk**) and specular mapping (described below) are used, respectively. The bump and specular mappings require grey scale textures. Therefore, the MLP values are converted into a grey scale image texture.

Data elaboration for rendering is done in steps (Figure 43 , lower part):

1. *MLP values mapping on the mesh*. The MLP values (typically between -3 and 1 for soluble, membrane-embedded and cytoplasmic proteins) are mapped on the surface of the molecule by assigning values of MLP to the mesh. The algorithm (included in a custom program, OBJCreator) is simple: for every vertex of the mesh, the correspondent grid-cell, in the MLP grid, is identified and the value of potential is calculated using trilinear interpolation (Figure 44) and assigned to the vertex .

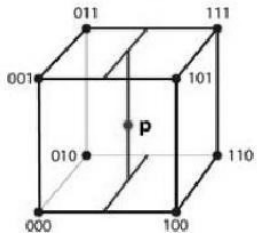


Figure 44. Three-linear interpolation algorithm. To find the coordinates of P, 3 projections are done on each of the 3 axis. In our case, P corresponds to a mesh vertex, while the cube vertices to the MLP grid. Therefore, starting from known MLP values of the cube vertices, the MLP value of each mesh vertex is easily calculated.

This process is very fast and the mesh vertex density is high enough to represent smoothly the potential spatial transition. The information about the MLP values corresponding to every vertex is stored in the V field of an .obj file as texture coordinates (U and V).

2. *MLP values conversion into vertex colours*. A classical dull material (the same material used in Maya) is assigned to the mesh and the MLP values (previously assigned to the vertices of the mesh) are converted into vertex colours.

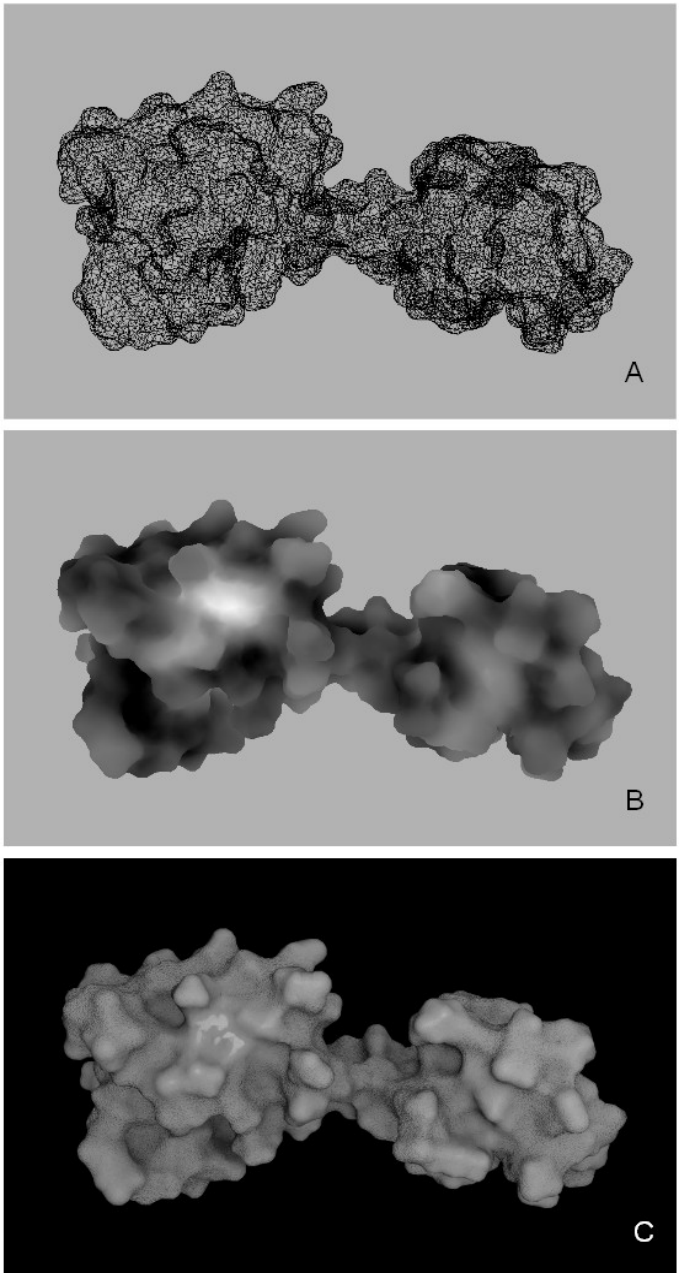


Figure 45. MLP representation. A wireframe visualization, **B** MLP visualization as levels of grey and **C** photo-realistic representation of MLP.

Each colour is defined by 3 RGB (Red, Green and Blue) values, in the range from 0 to 1 or from 0 to 255, depending on the software used. MLP values are converted into grey levels, obtained by setting the same value for each RGB channel. For example, in Blender, black is defined by (0.0, 0.0, 0.0) set of values while white is defined by (1.0, 1.0, 1.0). For the conversion, the range of the MLP values ([-3,1]) is normalized to the range of grey scale [0,1]. During normalization, the value 0 of MLP is set to correspond to the value 0.5 of the grey scale, resulting in visualization of neutral areas as middle grey. As an output of this step, protein's hydropathy is visualized in Blender as levels of grey: bright areas representing hydrophobic regions while dark areas representing

hydrophilic ones (Figure 45 B). The use of this default conversion scale provides a coherent representation for all proteins; however, at this step, to enhance MLP features for any particular protein under study, the user can modify the selected range of MLP values. The representation of MLP as levels of grey is the basis for the photo-realistic visualization. The code for the representation of hydropathy that we propose is a range of optical features that go from smooth-shiny surface for hydrophobic areas to rough-dull for hydrophilic ones, as discussed in chapter **THE AIM OF MY THESIS** and shown in Figure 45 C.

3. *Creation of the first image texture.* The photo-realistic representation is achieved by using appropriate textures. In our case we create custom textures starting from the vertex colours. The mesh is unwrapped (a technique described in section 3 **3D animation and rendering**) and the vertex colour (grey) values are saved ('baked') in an image texture (Figure 46 left). The steps 2 (MLP values conversion into vertex colours) and 3 (Creation of the first image texture) are achieved executing MLP.py script in Blender (Appendix, p. 125).

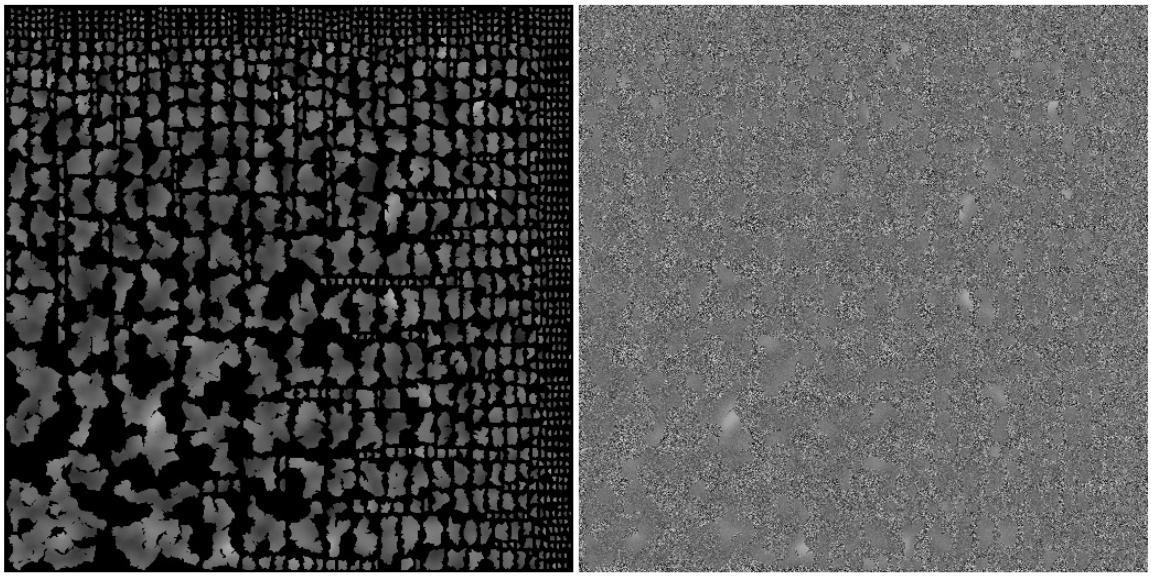


Figure 46. MLP image textures. (left) image texture obtained by baking the vertex colours (used for specular mapping) and **(right)** noisy image texture obtained as shown in Figure 47 (used for bump mapping).

4. Creation of the second image texture. In order to make the more hydrophilic areas rough, the procedure involves the addition of a noise pattern of amplitude proportional to the degree of grey of the texture. This is achieved using the Node Editor of Blender: a Gaussian noise (a noise that has a frequency distribution which follows the Gaussian curve) is added to the texture image (Figure 47), using a transparency ramp which leaves transparent the black regions and opaque the white ones. In this way, the combined image contains

strong noise over the black regions which is gradually reduced on grey regions until reaching a level without noise on white (Figure 46 right). In the rendering process this noisy image is converted into bump.

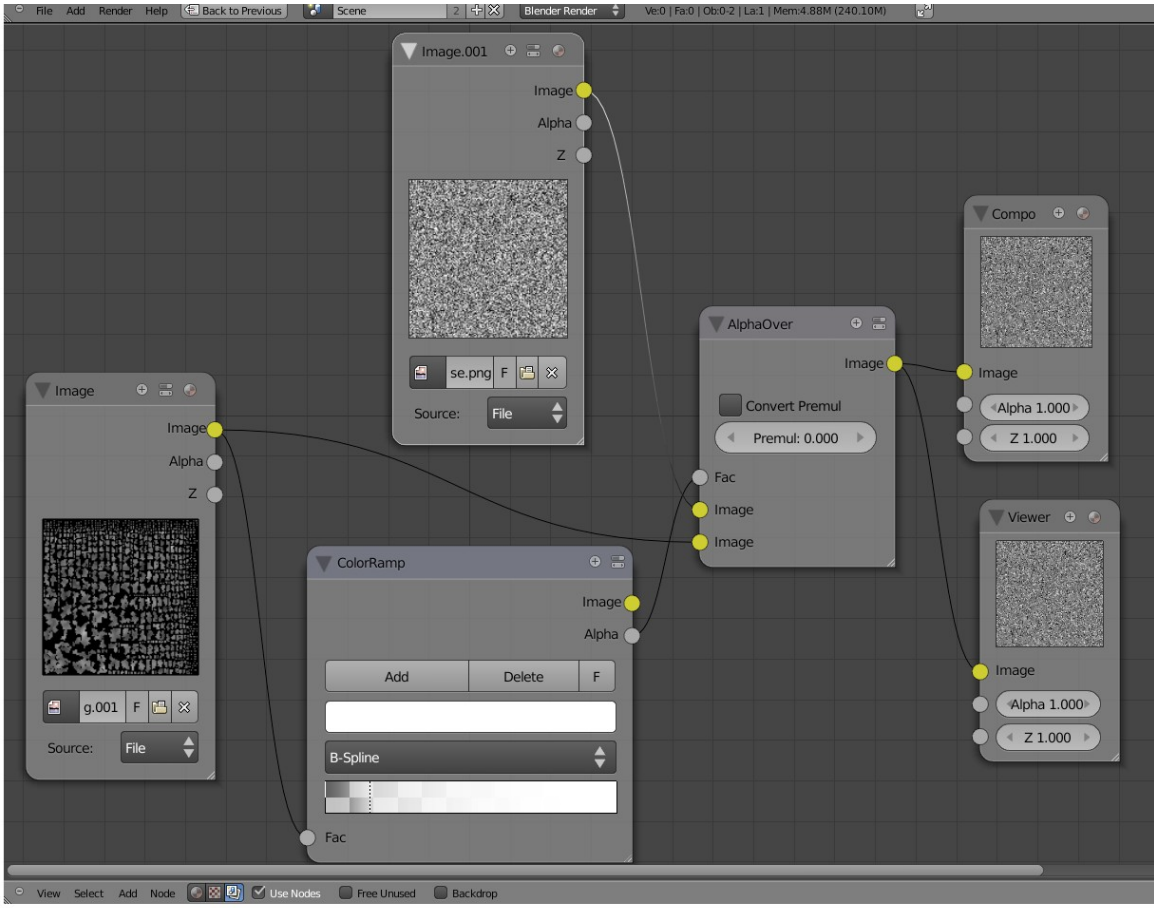


Figure 47. Node Editor of Blender. Noise is added to the image texture using the ColourRamp tool which leaves the black areas transparent and white areas opaque.

5. *Addition of specularity and roughness.* In the final rendering step, a default shiny material is assigned to the mesh. The Specular parameter is set to 0 to avoid additive shininess. The image obtained in the first step (grey scale image texture) is mapped on specularity: the black areas become dull, the white ones shiny. The second image is mapped on bump, where the bump intensity is proportional to the density of the noise. In the final image, hydrophobic areas are represented as reflective and smooth, while the more hydrophilic ones as duller and rougher (Figure 45 C). The mapping of the specular and bump textures is done by executing texture.py (Appendix, p. 127)

At some experimental step, the visual code included also the colour mapping of the grey scale image texture, obtaining dark-dull-rough surfaces for hydrophilicity and bright-shiny-smooth surfaces for hydrophobicity (Figure 48).

However, colour was omitted, as the non-exposed areas, usually darker in CG images due to less lightning, might be interpreted as hydrophilic regions.

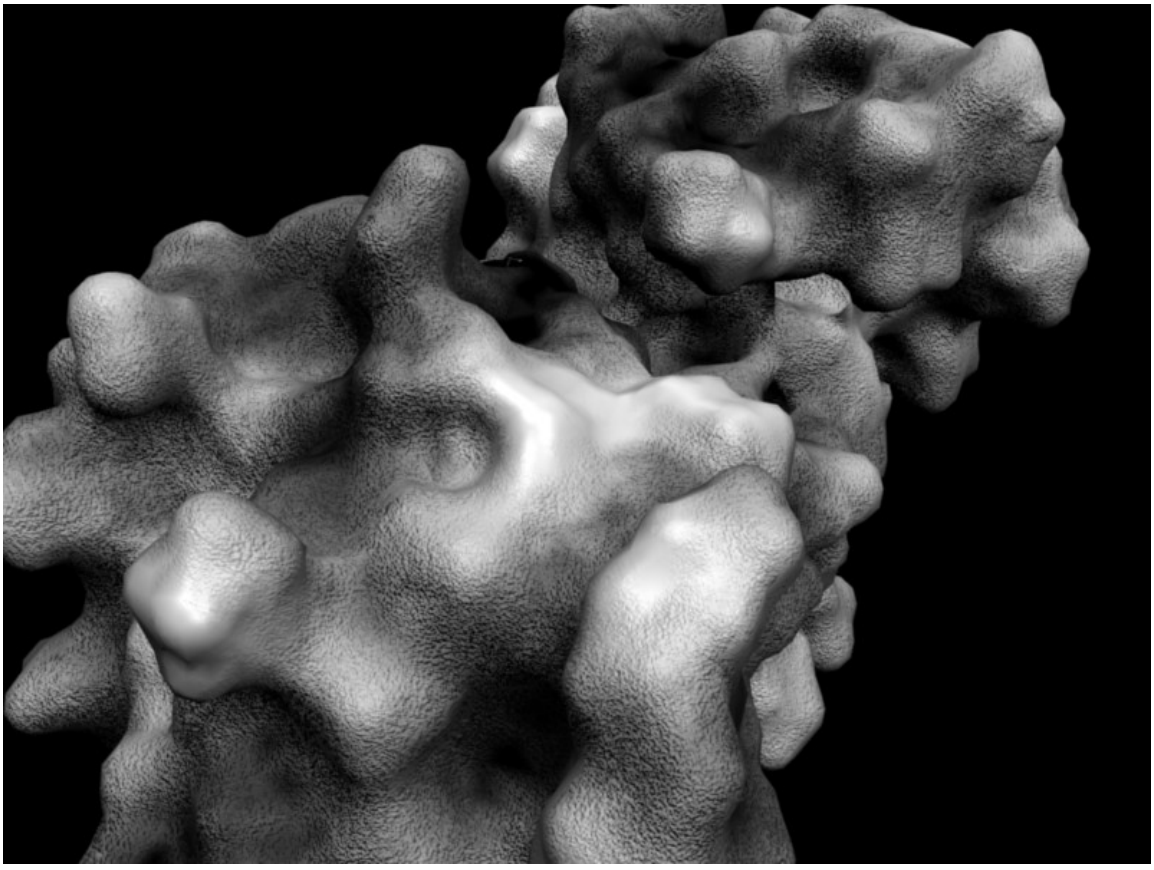


Figure 48. Alternative MLP representation. The range of features goes from bright-shiny-smooth surfaces for hydrophobicity to dark-dull-rough surfaces for hydrophilicity.

Compared to other algorithms that assign hydropathy to surfaces (eg. mapping Kyte-Doolittle values), this method has the advantage that in each point of the surface MLP is calculated as the contribution of neighbouring atoms, allowing the perception of a gradual changing on the surface of the protein.

2.3 Electrostatic potential

2.3.1 EP calculation

While the use of movies is mostly intended to show transition between conformations of a protein, it also allows the introduction of special effects of CG to convey other information. We have elaborated the following procedure using both Blender and external programs to calculate and display the EP associated with molecular (partial) charges (Figure 49, right side).

The .pdb file used for mesh creation and MLP calculation is submitted to PDB2PQR program which outputs 2 files: .pqr and .in. These files store information on the size and the charge of every atom (assigned using AMBER force field), and on the dimensions of the protein, the ionic concentration (0.15 mol/l NaCl), biomolecular and solvent dielectric constant (2 and 78.54 for water, respectively). Both .pqr and .in are input files for APBS program, that calculates the EP in every point of a grid that includes the protein and exports the values in a .dx file, analogous to the one seen above for MLP. The grid spacing is set by default to 1 Å and, similar to MLP, EP accuracy depends on grid spacing. The force field, the ion concentration and the grid spacing can be set by the user.

EP is redrawn as field lines calculated by a custom software, scivis.exe, that combines information from the mesh file (.obj) with EP values following different steps of computation:

1. Mapping EP on the surface mesh

2. Transformation of the grid of local values into a grid of gradients

3. Selection of most active surface areas by weighted Monte Carlo sampling

4. Drawing of field lines to be stored in a .txt file

The EP values are mapped on the surface of the protein by assigning a value of EP to every vertex of the mesh, with a process analogous to the one used for MLP, i.e. trilinear interpolation (see Figure 44).

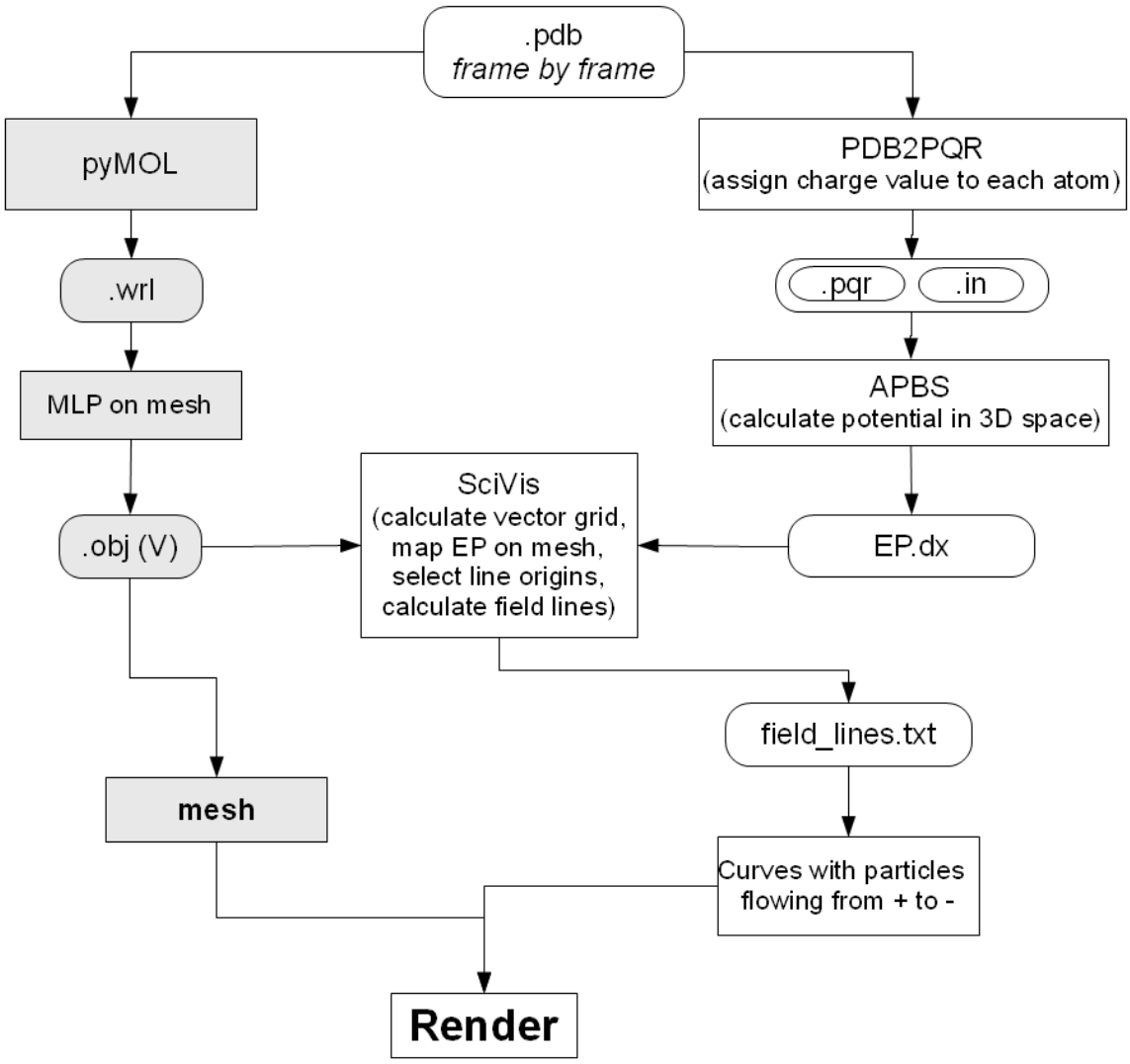


Figure 49. EP calculation and representation workflow.

A grid of gradient vectors is built starting from the scalar field of EP values: for each point, the gradient is calculated according to the values in neighbour points finding the direction and slope of EP change (Figure 50).

Faraday's criterion states that the density of the field lines is proportional to the electric field. One of the methods for the selection of seed points for field lines drawing is the evaluation of the gradient magnitude in every vertex of the EP grid, which is the method used by VMD where the control parameters are the gradient magnitude and the minimum and the maximum length of the lines. This implies drawing lines only where the gradient magnitude is the highest; however, the gradient magnitude is not an efficient parameter to compare two or more molecules from the electrostatic point of view because it indicates the rate of increase of the EP field, rather than its actual value. This is the reason why we start the drawing of field lines from points selected on the basis of absolute

values of EP. The selection of the seeds is done by Monte Carlo sampling, weighted with respect to the potential value of the surface in each area.

The gradient data are used to generate the field lines in the space surrounding the protein. From the infinite possible field lines, we are interested in generating a 'meaningful' subset comprising the lines associated with areas of the mesh with high value of EP, obtaining a distribution of lines that is proportional to the surface EP value: more lines rise in the more electrically active areas, and the total number of lines is proportional to the global level of potential of the molecule. Starting from the absolute value of EP, the field lines of various molecules can be compared.

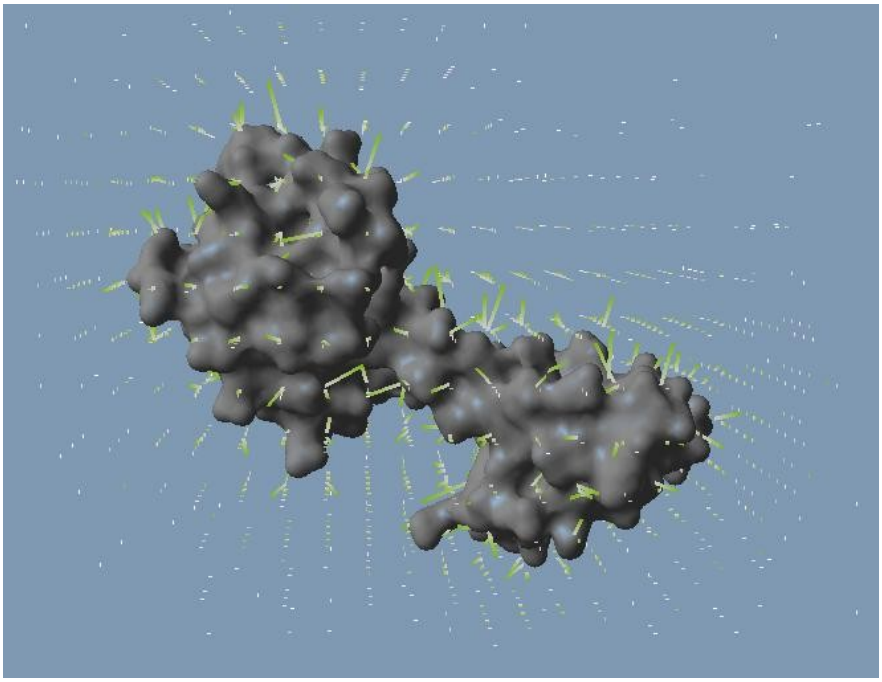


Figure 50. Gradient grid calculated by scivis.exe. The gradient is visualized as segments, with the length proportional to the gradient magnitude and coloured using a range that goes from green for positive EP values to white for negative ones.

For the selection of this subset, the user has two controls: the absolute EP value on the surface from which to create the field lines (lines are generated only in areas with an EP higher than a threshold – Minimum potential) and a parameter that represents the general line density (expressed as Number of lines x eV/Å^2). By modulating this parameter users can select the most appropriate value for a group of proteins, obtaining a concentration of field lines which is coherent across the various proteins.

Once the most electrically active locations (points) are selected, the lines

are calculated by following the gradient in both directions, iteratively moving with small steps according to the gradient (small-step integration). Line points are added until one of the following three conditions is met: 1. the limit of the calculated grid is reached, 2. the line intersects the mesh or 3. the field is too low (the gradient is approximately 0 or equal to the value set by the user). The lines are saved as sequences of points in an ASCII file (.txt).

Thanks to the random nature of the selection procedure, lines do change every time the procedure is run but the more electrically active areas (where more lines are present) are readily identifiable. This property proves to be particularly effective when represented in animation, since it gives the idea of fuzziness, useful for electricity representation, while conveying the information about EP distribution on the surface.

2.3.2 EP representation

Field lines are imported into Blender by a Python script (import_curves.py – Appendix, p. 129) as NURBS curves which are not rendered (they are invisible in the final image), but are used to guide a particle effect (Figure 51). Every curve starts at its most positive end which is associated with a particle emitter. The particles, drawn as short segments, flow along the curves from positive to negative, respecting the field lines convention in physics. In Blender, particles can have a series of attributes among which: the amount of particles, the first and the last frame of particle emission (particles can be emitted in one frame or continuously in a range of frames), their life time (the age at which each particle is switched off), the mesh component from which are emitted (vertices, edges and faces), the randomness (they can be emitted in the same time or at different random moments), some global effects can be applied such as acceleration, Brownian motion, *etc.*. Their specific appearance can be visualized as points, lines, crosses, axes, circles or specified objects.

In Blender 2.49 some functions could not be reached through API's and, among these, was the choice of the representation of particles as lines, points, *etc.*. For this reason, the win32lightcut090828.exe branch of Blender was used, a non-official version, in which the particles representation could be set from scripting. This branch is a Windows version that permits the import of field lines calculated by scivis.exe in the scene of Blender, the setting of the particles emitters at the positive end of curves and the rendering of particles as lines.

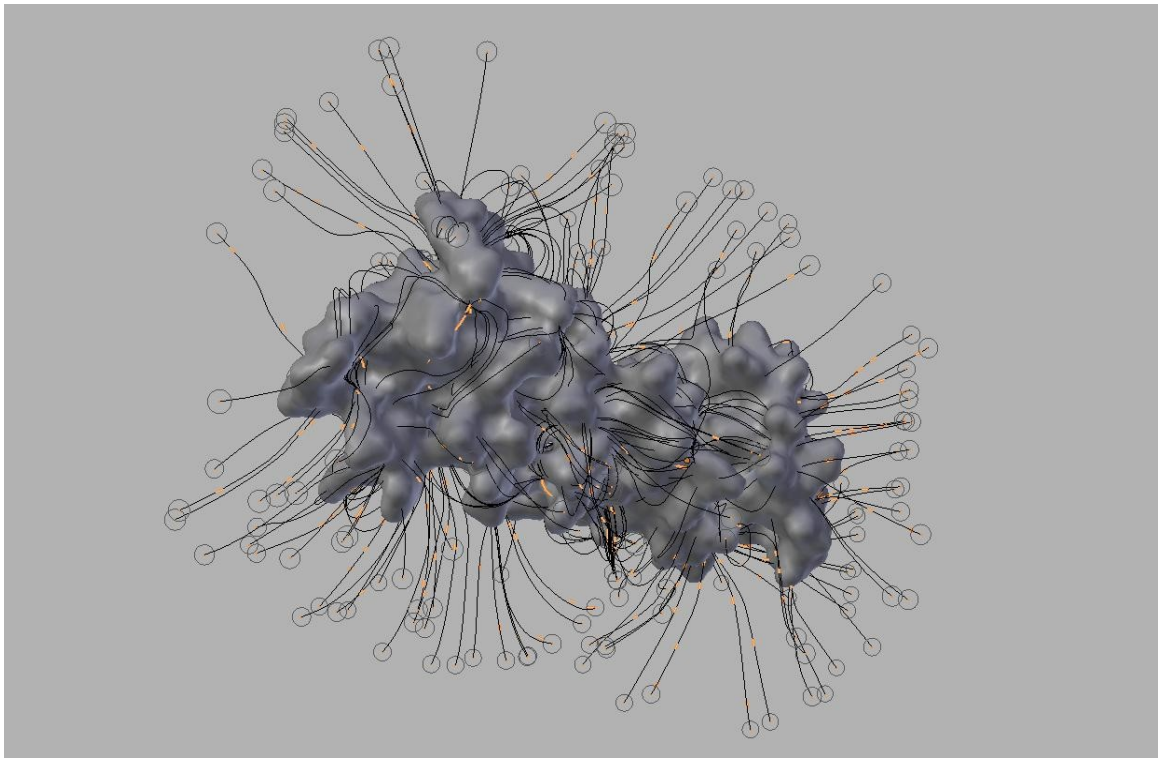


Figure 51. EP visualization in Blender. Field lines are visualized as curves, with a particles emitter at the positive end (depicted as circles); particles are drawn as small orange lines.

The particles are emitted randomly in time to avoid the simultaneous animation of particles along the lines and the amount of particles can be set by the user. The flowing of particles along the curves is achieved by converting each curve in a 'Curve Guide'; in this way the particles path is guided by the curve's shape.

For the visualization of a single conformation, the animated particles are emitted for 250 frames (10 sec) and have a lifetime of 20 frames. Representation of EP as moving particles on a trajectory, played in time, is interpreted easily and transmits the idea of polarity of the charged areas of a biomolecule.

Finally, for the simultaneous visualization of MLP and EP, the two representations must be combined: the textured mesh (representing the MLP), saved previously in a Blender scene, is appended (imported) to the scene containing the particle system. After setting the lights and the camera, the scene is ready for rendering (render.py – Appendix, p.131). The result is showed in Figure 52.

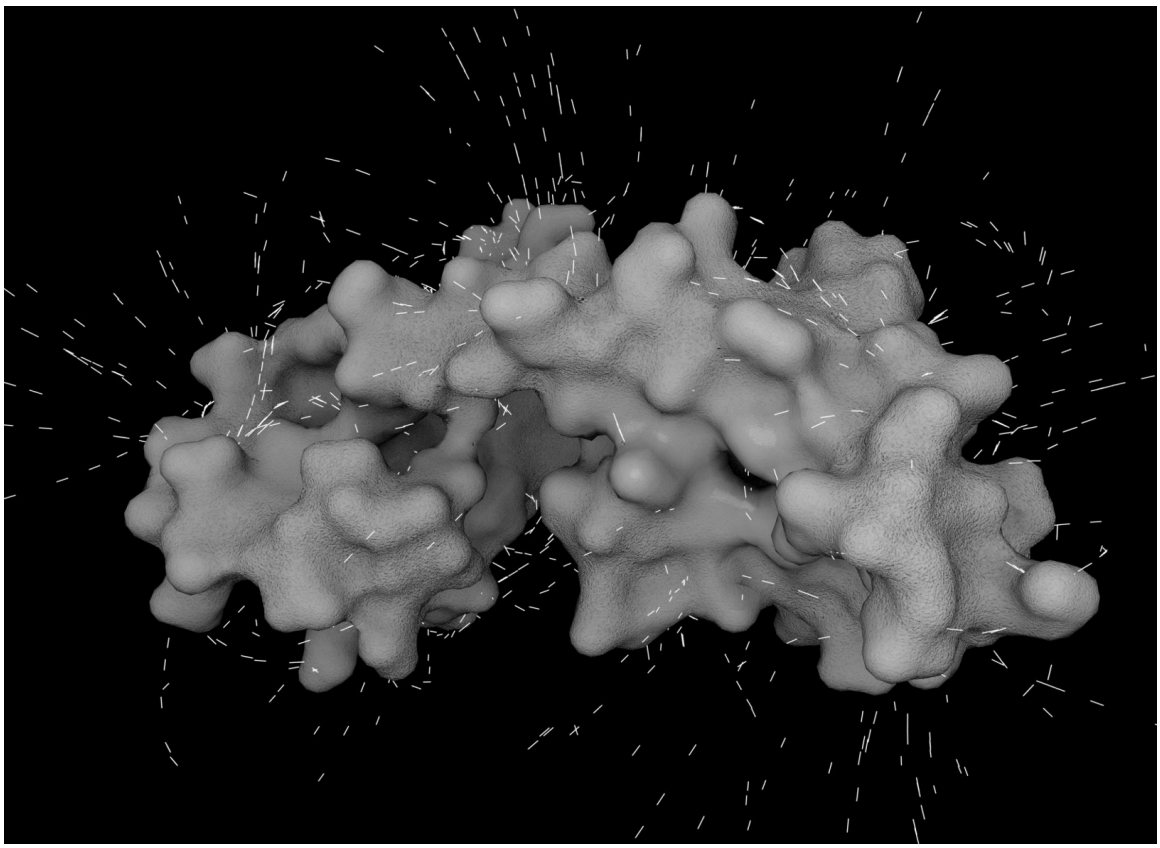


Figure 52. EP and MLP simultaneous representation.

2.4 Protein animation

Our method for the simultaneous visualization of MLP and EP is conceived to take into consideration the movement of proteins as described in section 4 **Molecular motion**. The conformational changes of proteins induce changes of the surface features, as they are calculated on atomic basis. To test our system, we do animations.

In the sections 2.2 **Molecular Lipophilic Potential** and 2.3 **Electrostatic potential** in Results, the MLP and EP representation was described for static proteins. The moving proteins are a challenge in terms of visualization of physico-chemical properties. The conformational change of proteins is due to modifications of atoms positions that imply changes in the shape and in the protein's surface characteristics. To visualize proteins in motion, the surface and the properties must be recalculated for every frame during the motion.

Therefore, for movies, the mesh and MLP are elaborated frame by frame; however, EP representation is obtained with an effect that takes several frames, and thanks to the continuity effect produced by the animated particles, it is not necessary to calculate EP for every frame. With these considerations, the method for the EP representation is slightly modified: particles are generated

every 5 frames (when showing proteins in motion, the emitter generates particles in one frame, in comparison to the static proteins where particles are generated for 250 frames) and have a life-time of 20 frames. This means that the system reaches steady state after the sixteenth frame as shown in the scheme in Figure 53; starting from the 16th frame, in every moment of the animation, there are 4 working emitters. Using this setting, the first 15 frames do not fully contribute to the EP visualization. To avoid the solution of discarding them, it is possible to create 3 different sets of field lines from the first .pdb file, associate 3 particle systems and set the start frame of particles generation at negative frames. Due to the random selection of the surface areas, the 3 sets of field lines are slightly different and the particles animation assures that the system is in steady state at the first frame.

The calculation of the particles' positions at each frame is a time consuming process and, in order to speed it up, particles are 'baked', which means that for each frame the particle properties and positions are registered to be reused afterwards, without the need of recalculating them. The registration is achieved by playing the animation of particles; the result consists in a series of files (called caches). Subsequently, when the animation is re-played or the rendering is performed, the information about the particles is read from the cache files.

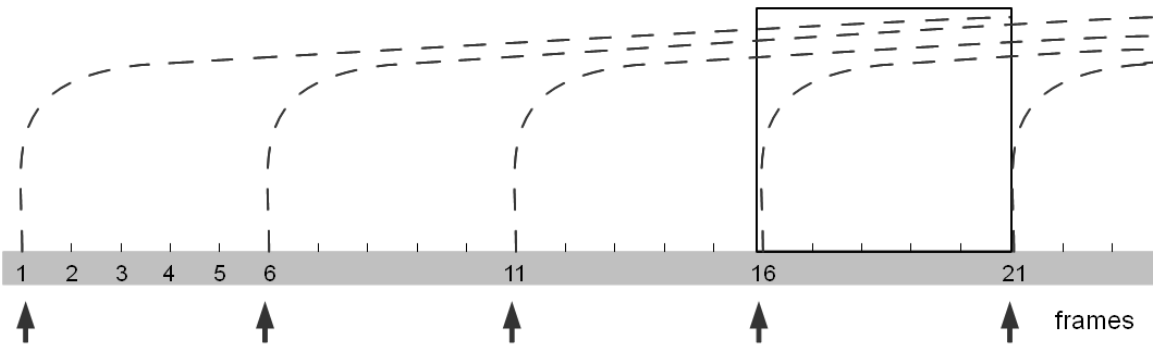


Figure 53. Particle flow. Particles are emitted every 5 frames and have a lifetime of 20 frames; after the 16^{tth} frame, the system is in steady-state.

In summary, for each frame (conformation) we visualize the molecular surface as mesh, MLP represented as texture and EP as curves and animated particles. The result is a sequence of frames showing the moving protein with its

surface properties represented together: MLP as a range of visual and tactile characteristics and EP as flow of particles that move from positive to negative along the invisible field lines (see PROTEIN EXPRESSION Study N.3).

2.5 Automation

As mentioned above, the visualization of moving proteins with their surface properties implies calculation of surface, MLP and EP for every frame of the animation. To achieve this, an automated system is created.

For simplicity, the method for visualization of proteins features is divided into 2 branches (as described in sections 2.2 and 2.3 in Results): MLP representation on one hand and EP representation on the other hand, which are combined in the end for the rendering process. The large amount of data that rise from calculations at every frame of animation make it impossible to import all the meshes and all the particles calculated due to computer limitations. Therefore, the animation is split into various scenes, including a set of meshes with their associate texture images and the relative particle systems. The major difficulty encountered when splitting the animation into sets of independent scenes is the continuity of particles flow. The particles continuity is maintained overlapping particle systems in consecutive scenes: the first 3 particle systems of a scene are the last 3 in the previous scene.

As calculations for each .pdb file imply a repetitive work, an automatic method was applied consisting in:

- calculation of molecular surface, EP and MLP, mapping of MLP on the surface and storage of MLP corresponding to each vertex in the V channel of an .obj file (for this step, cycle.sh, a shell script for Linux was written);
- import of the meshes in Blender, unwrapping, conversion of MLP values into colours and baking of image textures (MLP.py);
- addition of noise to these images (Node Editor of Blender);
- assigning of image textures to the unwrapped meshes (texture.py);
- calculation of the filed lines starting from the .obj file and the .dx of EP; the automation was done using a .bat script; this step is done in Windows;
- import of the field lines in Blender as NURBS curves and set the positive ends of each curve as a particles emitter (import_curves.py);

saving the cache of particles by playing the animation; due to the Blender branch OS limitation, the import of curves is done in Windows;

- rendering of each frame of the animation (render.py), is performed in Linux due to its stability and more efficient RAM usage, resulting in an increased rendering speed. Render.py is associated to each frame of the animation and it displays the corresponding mesh (from the list of all meshes present in the scene), image textures and performs the rendering, saving an image for every frame of the animation.

This workflow, consisting in systematic execution of a collection of Python scripts, managing of a set of Blender scenes, switching between OS, is tortuous, easily exposed to mistakes and difficult to follow. To easy the workflow, we created BioBlender, a Blender user-friendly interface that coordinates these steps, as described in section 2.7 **BioBlender**.

2.6 Movies

As mentioned in section 2.4 **Protein animation**, we do animations to test our visualization studies. The testing process resulted in creation of three movies: PROTEIN EXPRESSIONS Study N.1, PROTEIN EXPRESSIONS Study N.2 and PROTEIN EXPRESSIONS Study N.3 (all available on our website). During each test we improved the method for visualization and the automation steps. In the latest movie, the maximum computational limitations for Calmodulin scene is 145 meshes, each mesh with 22662 polygons, 2 image textures 1024x1024 and 32 particle systems with 150 curves and 200 particles each.

2.7 BioBlender

BioBlender [Andrei2010, BioBlender] is an extension of Blender 2.5 (see below), in which custom Python scripts have been implemented for building an interface especially for biologists. It is a tool dedicated to elaboration of proteins' motions and to visualization of surface properties of proteins. It outputs simultaneous visualization of EP and MLP on proteins in motion. The entire package is an ensemble of computer graphics software and physico-chemical programs and scripts, described in **TOOLS: PROGRAMS AND SCRIPTS** chapter. It includes all the steps previously described, which are now performed in an easy, but covert manner.

The BioBlender user interface is contained in the vertical Scene Property

Panel (one of the panels of Blender), as shown in Figure 54. The BioBlender interface allows the user to import and interactively view and manipulate the macromolecules. BioBlender for Windows, Linux and MacOS is available from www.bioblender.org. Because of its specialized nature, it requires the installation of PyMOL, Python 2.6, NumPy (all are provided in the Installer folder from the downloaded package) and ProDy (available on www.csb.pitt.edu/prody).

As the import of molecules with more than few hundred atoms is very slow, Blender 2.5 was modified (a patched version was created) including the generation of atoms by duplication with multiple copies at once. When a .pdb file is read, the system records the number of atoms of each type, and atoms are created in the Blender scene by multiple duplication of the reference atoms saved in a hidden Blender scene (atoms library); for example, if in a .pdb file there are 1000 carbon atoms, BioBlender makes 1000 copies of the sphere (included in the library.blend scene), modifies the radius to be equal to the carbon covalent radius, assigns the proper colour and displays them in the Blender 3D view at the corresponding coordinates.

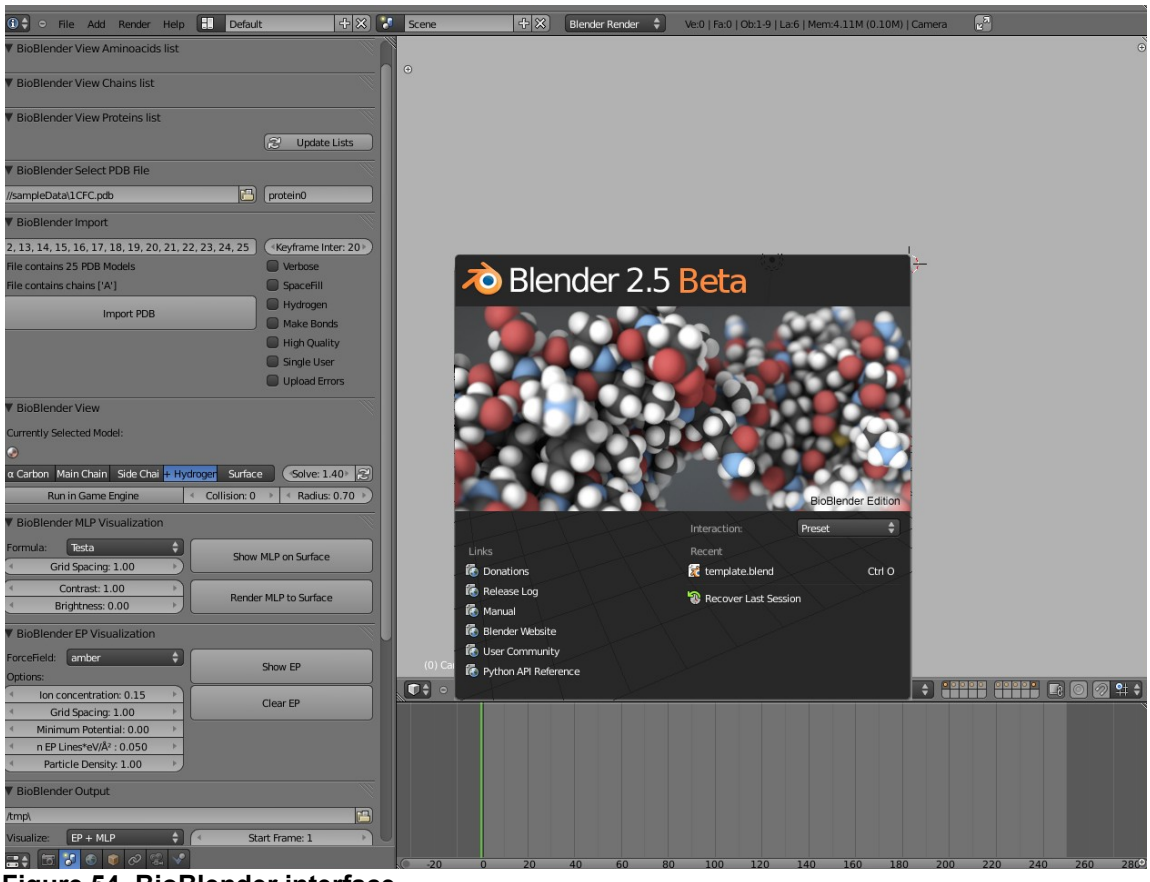


Figure 54. BioBlender interface.

BioBlender start-up scene not only has an optimized user-interface layout for biologists, but it also contains lights, camera and world settings that are ideal for visualizing and rendering molecules. This set-up ensures that researchers who are not familiar with the 3D software can still effectively use BioBlender. Each interface element (buttons, sliders, toggles) has help text associated with it. By placing the mouse over them a pop-up text describes the function. Errors and progresses are displayed in the console. Critical errors will appear in the main BioBlender as a pop-up under the mouse cursor. As the atoms size is of order of Ångström (Å), the scale used is 1 Blender Unit = 1 Å.

BioBlender interface includes a series of panels roughly divided into selection and import of .pdb file, visualization of the molecule, physico-chemical properties calculations and output.

1. Selection and import of a .pdb file

In *BioBlender Select PDB File* panel, the user can select a .pdb file by browsing locally for the file saved or by simply typing the 4-letter code of the protein of interest to be fetched from Protein Data Bank (1 in Figure 55). The name of the protein may be changed (2 in Figure 55 - by default it is called 'protein0'). Naming the proteins is a good habit that will help keeping the scene organized. Once a file is selected, the number of models (for NMR files) and the chains (for proteins with multiple chains) are detected and shown in *BioBlender Import* field (3 in Figure 55). In the case of NMR files, the models to be imported can be selected; these conformations will be then set in time at an interval determined by the *Keyframe Interval* slider (4 in Figure 55).

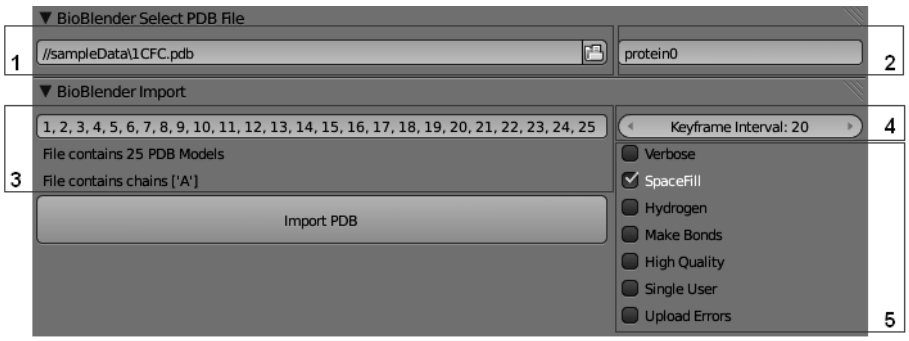


Figure 55. BioBlender Select PDB File and BioBlender Import panels.

A list of options are available to be considered before importing the protein in the Blender scene (5 in Figure 55):

– *Verbose*: enable to display in the console extra information for debugging;

– *SpaceFill*: enable or disable to display the atoms with Van der Waals or covalent radii in the 3D scene, respectively;

– *Hydrogen*: enable to import Hydrogens if they are present in the .pdb file. This option makes importing much slower and it is important only for visualization. If the .pdb file does not contain Hydrogens (or if you chose not to import them), they will be added during the Electrostatic Potential calculation using external software;

– *Make Bonds*: enable it to have atoms connected by chemical bonds. Despite being time consuming (16 seconds for 1166 atoms) this operation is essential in motion calculation;

– *High quality*: displays high-quality atom and surface geometries; slow when enabled;

– *Single User*: enable to use shared mesh for atoms in Game Engine; slow when enabled;

– *Upload Errors*: enable to send us automatically and anonymously an email with the errors you generate. This makes us aware of the problems that arise and helps us fix them.

2. Visualization in the 3D viewport of Blender

Once imported, the protein is displayed in the 3D scene of Blender; if more than one model are selected, Blender interpolates linearly between conformations and displays the protein in motion. By default, and if the Hydrogen option was enabled, all atoms are visualized. BioBlender View enables different views: only alpha Carbons, main chain (N, CA, C), main chain and side chains (no hydrogens), all atoms and surface (Figure 56).

If the user selects the view as *Surface*, BioBlender computes Connolly surface of the protein by invoking PyMOL software. It uses the Solvent Radius (the radius of the solvent probe sphere, usually 1.4 Å the radius of the water molecule) to create the molecular surface.

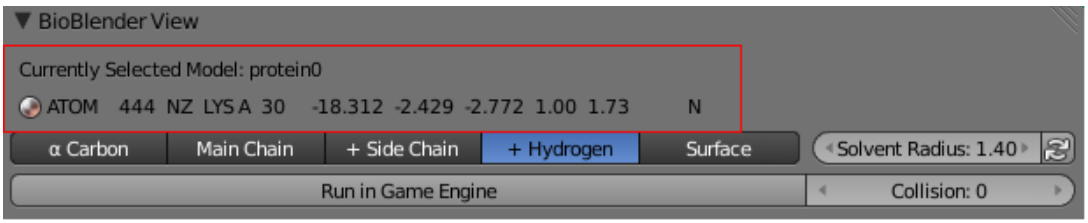


Figure 56. BioBlender View panel. Various kinds of visualization available in BioBlender; information about the atom selected in the 3D viewport is highlighted.

When atoms are displayed, by selecting one atom in the 3D display, the information of the selected atom is printed in the area outlined in Figure 56; in the 3D view the selection extends to all atoms of the corresponding amino acid.

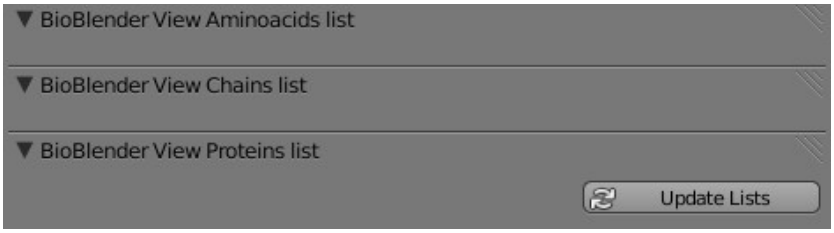


Figure 57. BioBlender lists of molecular components. The aa, proteins chains and various proteins present in the scene can be selected and highlighted in the 3D viewport.

Once the protein is imported, by pushing *Update Lists* button more information about the protein components are displayed (Figure 57); it is possible to select amino acids, chains and proteins (if more than one protein is imported) present in the scenes.

3. Calculation of protein motion using game engine

Run in Game Engine button enables the use of Game Engine to calculate the transition between different conformations. When in Game Engine mode, the protein visualization is changed: all atoms are white, ambient occlusion effect is applied (to give a better sense of depth) and the mouse controls the rotation of the protein, allowing to inspect it from all angles (Figure 58 top). Calculation of motion is done in 2 ways: simply linear interpolation (setting the *Collision* to 0) and linear interpolation considering collisions (*Collision* to 1); when the latter one is used, the motion can be recorded (*Collision* to 2). The positions of all atoms are recorded by setting a key frame on each frame of the animation. They are saved in the Timeline panel at different time (200 frames away from the last model imported) as shown in Figure 58 bottom; in this way both sets of transitions are available for comparison. These conformations can be exported

as described later in section 7.

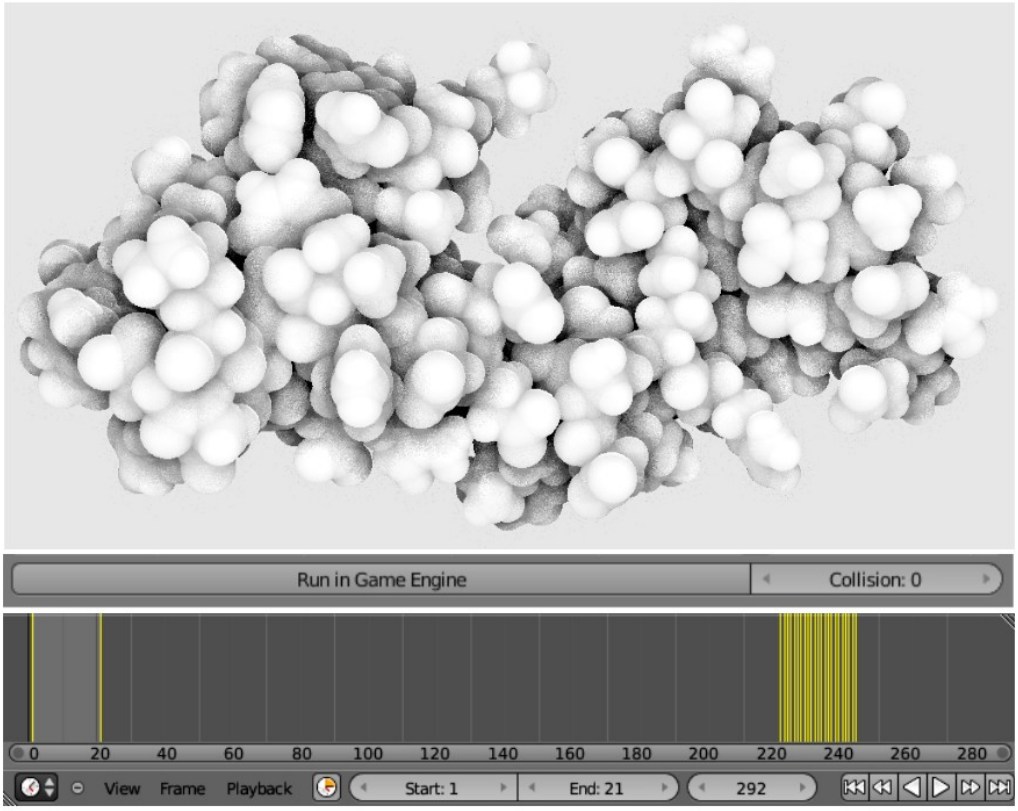


Figure 58.Game Engine in BioBlender. (top) Protein visualization using GE; **(middle)** GE parameters; **(bottom)** timeline panel: keyframes at frames 1 and 21 are the positions of the 2 conformations imported in the scene, and the keyframes from 221 to 242 are all the keyframes calculated using GE.

4. Molecular Lipophilic Potential visualization

BioBlender MLP Visualization panel collects all the scripts and parameters (1 and 2 in Figure 59 left) necessary for MLP representation. After selecting the *Formula* and the *Grid Spacing*, *Show MLP on Surface* button invokes PyMOL for the creation of the molecular surface, pyMLP.py for calculation of MLP in every point of a grid containing the protein and other scripts to map the MLP values on the surface, convert them into levels of grey and assign them to vertex colours (see section 2.2.2 **MLP rendering**). The protein is imported in the 3D space of Blender with MLP represented as levels of grey (light areas for hydrophobic regions and dark areas for hydrophilic ones), as shown in Figure 59 right.

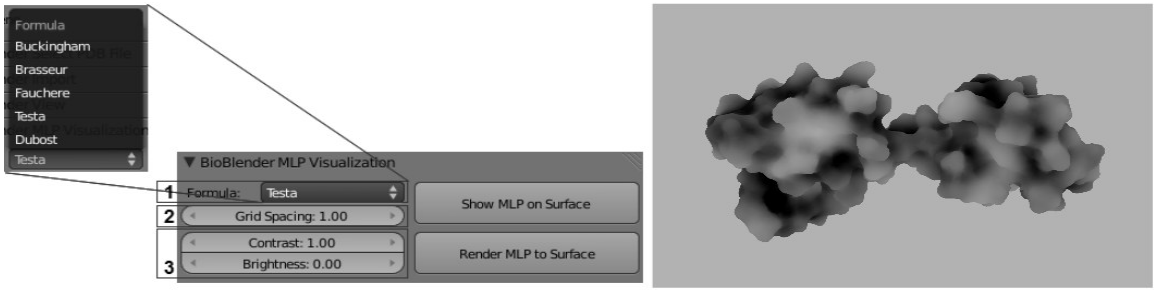


Figure 59. BioBlender MLP Visualization. (left) BioBlender MLP Visualization panel, Formula and Grid Spacing are the parameters used for MLP calculation, Contrast and Brightness are parameters used to enhance MLP visualization; **(right)** MLP visualized as levels of grey.

As stated in the description of the MLP calculation, the user can enhance the visualization of the protein of interest by modifying the range of absolute MLP values. This can be done in BioBlender modulating the *Contrast* and *Brightness* sliders (3 in Figure 59).

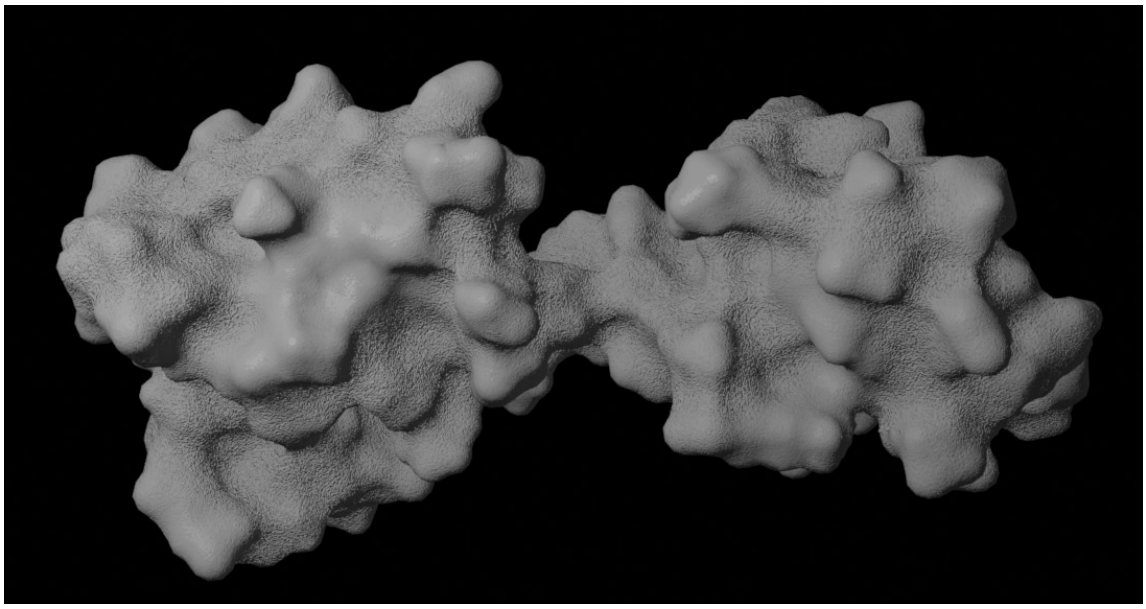


Figure 60. MLP photo-realistic representation.

Once the user is satisfied with the grey level visualization, the next step is to obtain the photo-realistic representation. The unwrapping of the surface, baking of vertex colours into image textures, addition of noise and assignment of textures to the material of the object are all automatically performed by clicking *Render MLP on Surface* button. The bright areas become shiny and smooth while the dark areas become rough and dull (Figure 60).

5. Electrostatic Potential visualization

In *BioBlender EP Visualization* panel, all the parameters for EP calculation and representation are included. The input parameters are *ForceField*, *Ion concentration*, *Grid Spacing*, *Minimum Potential*, *n EP lines*eV/Å²* and *Particle Density* (Figure 61). The latter parameter can be changed also after the import of curves into the Blender scene to modify the number of particles per curve. By default, for non-moving proteins, the particles are emitted for 250 frames and have a life-time of 20 frames.

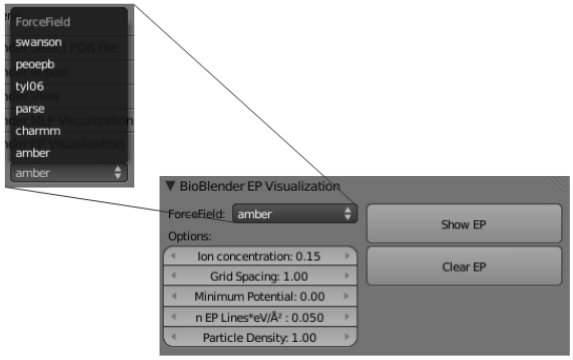


Figure 61. BioBlender EP Visualization panel.

6. Normal Mode Analysis Visualization

BioBlender NMA Visualization panel controls all the parameters needed by ProDy package [Bakan2011] for NMA calculation performed: *Mode* (from 1 to 20), *NMA steps* (number of conformations to be calculated in both directions along the given mode), *RMSD sampling* (RMSD between the given and the farthest conformation), *NMA cutoff* and *NMA Gamma*. By pressing *Calculate NMA trajectories (pdb)* button, (2 *NMA steps* + 1) conformations are calculated and selected for import into the Blender viewport. Press *Import PDB* button in the *BioBlender Import* panel to import them and visualize the movement in Blender.

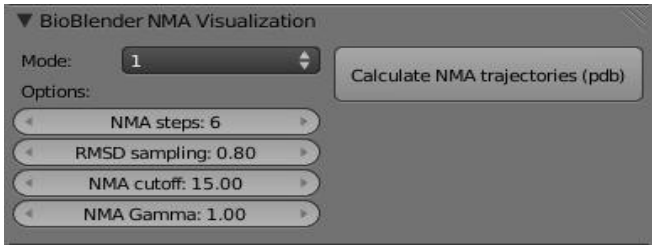


Figure 62. BioBlender NMA Visualization panel.

7. Output

The movie making parameters are enclosed in *BioBlender Output* panel. Various kinds of representations can be chosen from the Visualize curtain, such as: atom (only atoms), plain surface (only the surface), MLP (surface with MLP), plain+EP (surface and EP without MLP), EP and MLP (surface with MLP and EP).

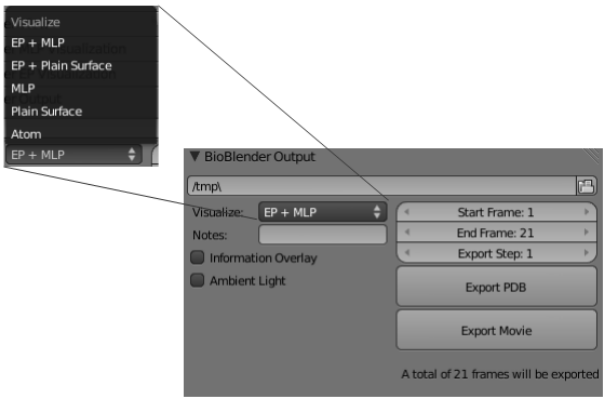


Figure 63. BioBlender Output panel.

This panel also allows user to export intermediate key frames calculated with Blender Game Engine in .pdb format, using *Export PDB* button. The new conformations can be evaluated with scientific programs, as described in section 4.1 **Morphing in Blender Game Engine**.

2.8 3D Interactive and still images

The main focus of our lab is the development of tools to visualize proteins' motions; however this is only possible where sufficient data are available from the literature. Furthermore, in many cases it may be important to focus on a single conformation, or a printed image not allowing animation is necessary, which requires a static representation.

2.8.1 3DNP

As initial attempt for 3D interactive visualization we used 3DNP (3D No Plugin), developed by Thorsten Schlüter [3DNP], a JavaScript that stores a series of images into memory and uses web browsers to simulate the 3D view by analysing the user's mouse movement and quickly swapping the images. 3DNP is accompanied by a Python script for Blender 2.49 which creates a sphere around the object of interest. The rendering is then performed by positioning the

camera, successively, in each vertex of the sphere.

As well as in still images (Figure 64 left), in 3D interactive mode, particles cannot transmit the polarity of the charged areas of molecules. Moreover, interactive exploration result in an effect in which the direction of the particles seems to depend on the direction in which the mouse is moved. To overcome this limitation, we propose the use of oriented objects, such as cones or pyramids. Pyramids are preferred as they are computationally lighter than cones, being formed by a lower number of polygons. Pyramids are built along the field lines in scivis software with the tip oriented towards the negative end of the line, maintaining the convention used in physics as shown in Figure 64 middle. Pyramids creation is based on 2 parameters: pyramids size and the distance between them. If the distance between the pyramids is kept constant, the resulting effect is of concentric spherical waves around the protein (Figure 64 middle). This little shortcoming is overcome by positioning the pyramids at random distances in the range [-40%,+40%] of the distance parameter set by the user (Figure 64 right). The 3D interactive representation described here allows its implementation in Proteopedia [Calmodulin motion on Proteopedia] or in 3D PDF, delivered as electronic document.

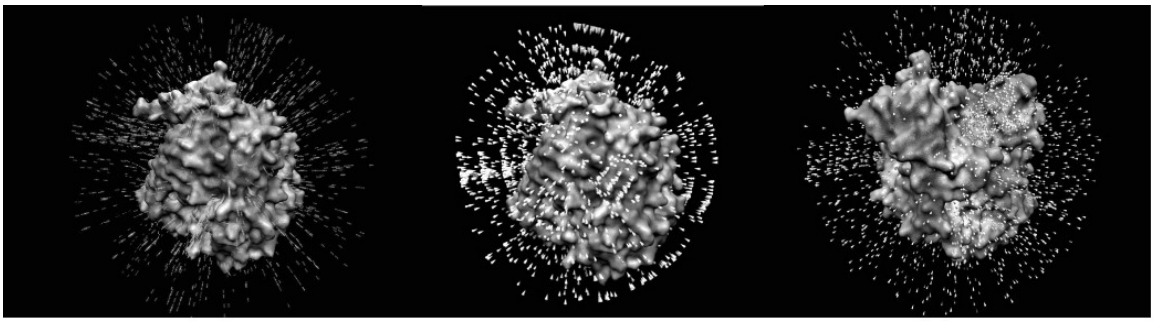


Figure 64. Still images and 3D Interactive visualization. AchE protein based on 1VOT.pdb file, a very charged protein, with negative charged areas alternating with positive charged ones. **(left)** EP visualized as particle as in BioBlender; **(middle)** EP visualized as cones equidistantly displayed, resulting in concentric circles; **(right)** EP displayed as random positioned cones along the field lines, resulting in a non-uniform distribution of the cones.

This 3D Interactive visualization permits the protein visual investigation from all points of view, but it requires computer space to store the rendered images necessary for a detailed representation. Because of this we took advantage of SpiderGL, which does not need a pre-rendered set of images, and adapted it for molecular visualization.

2.8.2 SpiderGL

For the 3D Interactive exploration of a protein physico-chemical properties, we use SpiderGL [Di Benedetto2010], a library for WebGL [Group2009]. WebGL (Web-based Graphics Library) is a library that extends the capability of the JavaScript programming language to allow it to generate interactive 3D graphics within compatible web browsers. It uses HTML5, a language for presenting content on World Wide Web, implemented in Mozilla Firefox 4 and Google Chrome and in development releases of Safari and Opera.

SpiderGL provides a set of data structures and algorithms to support the management of geometrical entities. To ease the creation of graphical applications, SpiderGL provides a series of classes and functions which cover the various aspects and levels of implementation of a CG program such as:

- basic structures which include linear algebra algorithms for 3D points;
- management of 2D data for the definition of 3D objects, textures and other components used in the rendering process;
- scene management through specific helpers to place the objects in the 3D scene, set the viewpoint, and simplifies the user interaction with the 3D elements;
- management of rendering by assignment of materials and textures to the objects;
- application, enhanced by the interactive visualization of this library. All the visualization code is embedded in the page, allowing the transparency of the data processed and the possibility of sharing knowledge, two important aspects in research and educational tools.

For the visualization of proteins with EP and MLP within SpiderGL, we developed a visualization method [Callieri2010] which uses data previously calculated with BioBlender. For the MLP representation we use the 2 image textures used for bump mapping and specular mapping (visual code described in section 2.2.2 **MLP rendering**) and the parametrized mesh (the unwrapped mesh, ready for texture mapping) exported from BioBlender in an .obj format. For the EP, calculated with BioBlender, a small modification is introduced: the field lines are calculated externally, using scivis.exe program, and exported as JSON (Java Script Object Notation) file, an easy to write format, natively supported by JavaScript interpreters.

To reproduce the EP representation as moving particles flowing along the

field lines, we create a shader that renders only small fragments of the imported lines according to a periodic function, animated using an offset. The shader gives the impression of small comets moving along the field lines, with the head oriented towards the negative end (Figure 65). This effect is much simpler to obtain and less CPU/GPU demanding than a real particle system, while still effective in conveying the electric characteristic of the protein.

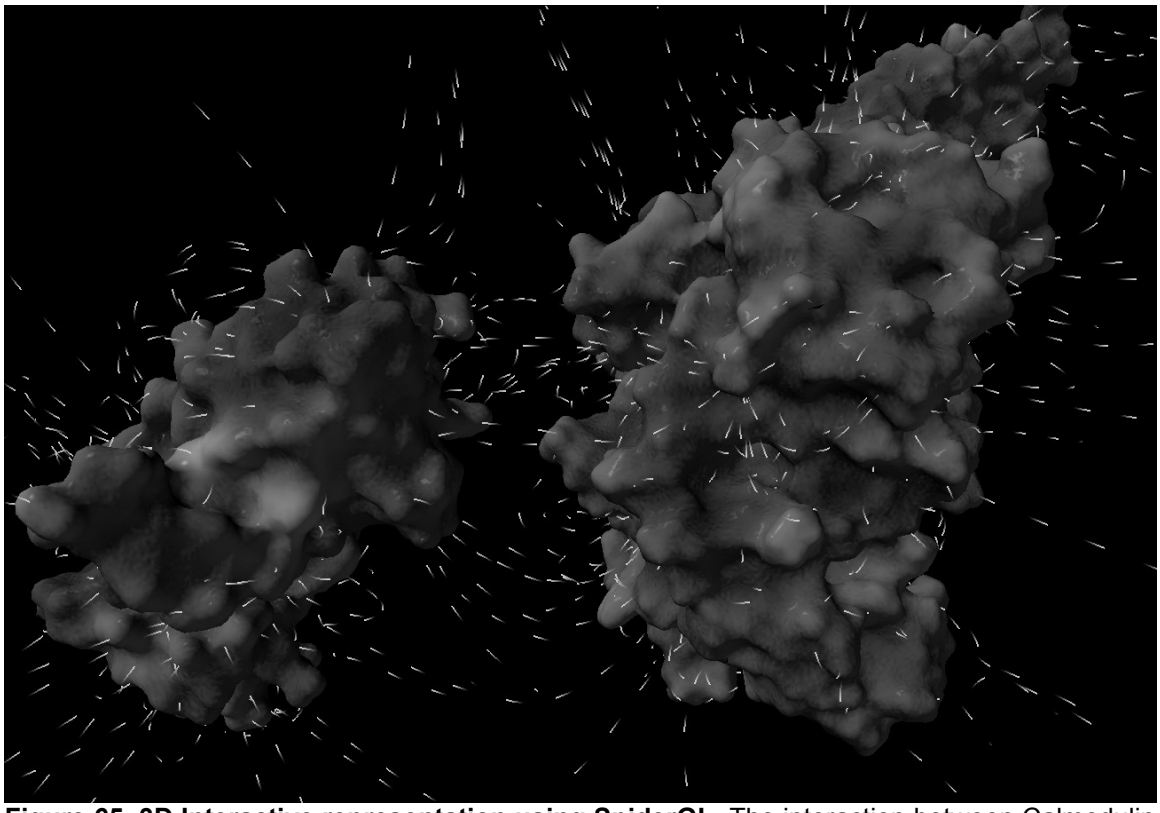


Figure 65. 3D Interactive representation using SpiderGL. The interaction between Calmodulin and MLCK head is mainly visible through EP.

When visualizing proteins as molecular surfaces with MLP and EP it might be useful to know the underling structures in particular regions of the surface. In SpiderGL, the dual representations, molecular and atomic (van der Waals), are available using transparency. The user can switch between the global transparency depending on the viewing angle (Figure 66 A) and the lens transparency which is focused on the area around the mouse pointer (Figure 66 B). The user can interact with the object using the left mouse button to rotate and the wheel to zoom. For the stereo 3D visualization, the user can toggle between anaglyph (Figure 66 C) and side-by-side representation (Figure 66 D).

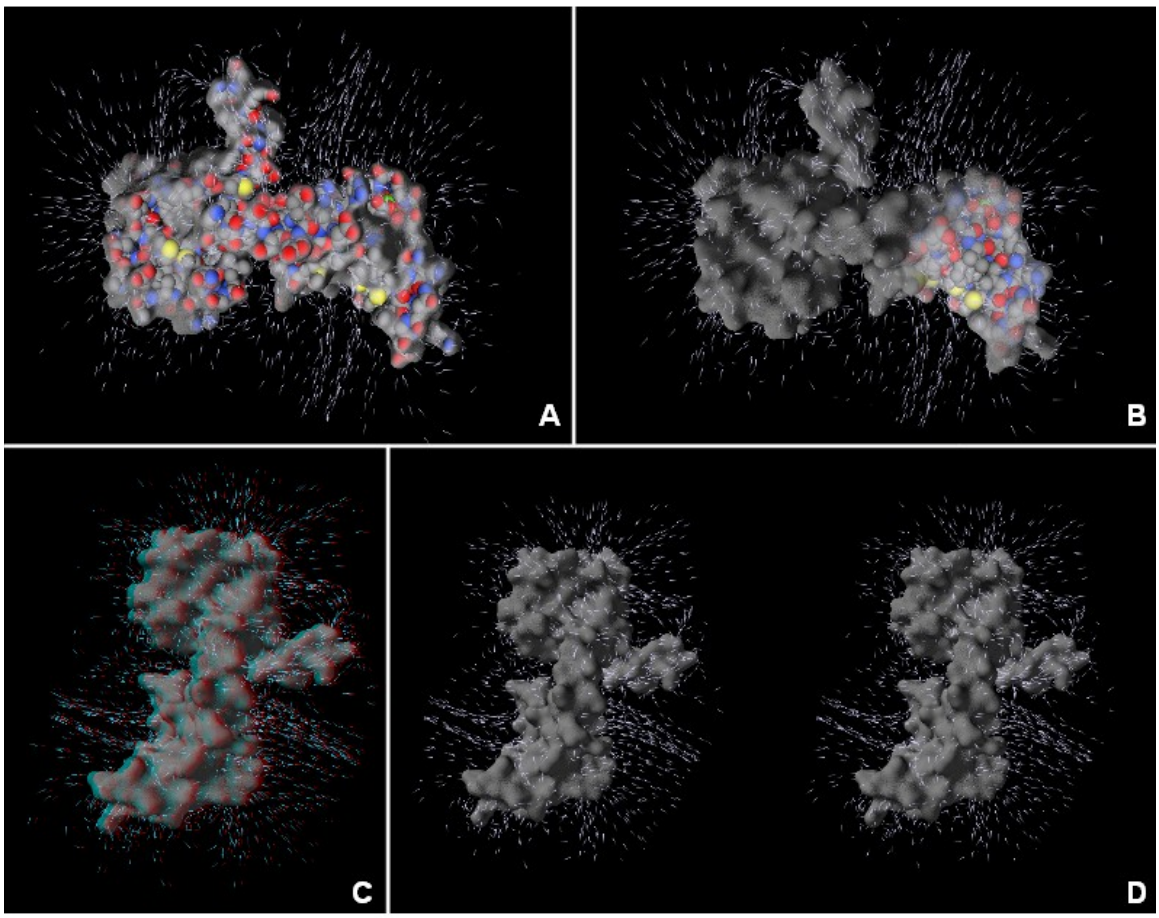


Figure 66. SpiderGL visualizations. (A) global transparency; **(B)** focused transparency around the mouse cursor; **(C)** anaglyph 3D stereo; **(D)** side-by-side 3D stereo.

The package of programs, complete with instructions, can be downloaded from http://www.scivis.ifc.cnr.it/index.php/3dinteractive/tutorial.html.

2.9 Ongoing project

Hydropathy on van der Waals surface

Besides this code introduced for the visualization of proteins' hydropathy, we also experimented a new representation which consists in mapping the atomic lipophilic values directly on the van der Waals surface and converting them into grey levels. This procedure is halfway between the visualization of hydropathy based on amino acid values of hydrophobicity (Kyte-Doolittle) and the MLP calculation; it is a rapid solution (the values are assigned once, without any further calculations), providing an overview of the hydropathy according to the spatial distribution of atoms.

This visualization method was tested in occasion of visual representation of MD data simulation. As mentioned in section 4 **Molecular motion**, MD is a technique used to calculate proteins conformational changes. The result of MD

analysis is a collection of a temporal sequence of conformations, called trajectories; by visualizing them we can actually 'see' the protein's movement. The most indicated protein representation to analyse its motion is the van der Waals surface. Instead of CPK code for atoms, we associate some physico-chemical information: hydropathy, visualized as atomic lipophilic potentials mapped on the atoms. The vibrational motions retrieved from MD analysis, translated in terms of CG animations as large movements of side chains during successive frames, might not permit a good perception of protein's motion if represented as surface and with hydropathy visualized as surface material features. The protein of our study is BPTI (bovine pancreatic trypsin inhibitor), a protein analysed by Shaw *et al.* [Shaw2010] in a 1 ms MD simulation which reveals various transitions between 5 stable conformations. The MD data, kindly provided by the authors, comprise the simulation between 0.3 and 0.9 ms in which 2 transitions are observed. Starting from these trajectories, we decided to analyse visually only one transition. The analysis of BPTI, with the atomic lipophilic potential representation, revealed new insights into the protein's major movement (transition between conformations). During the motion it is also possible to see how a hydrophobic area is being exposed.

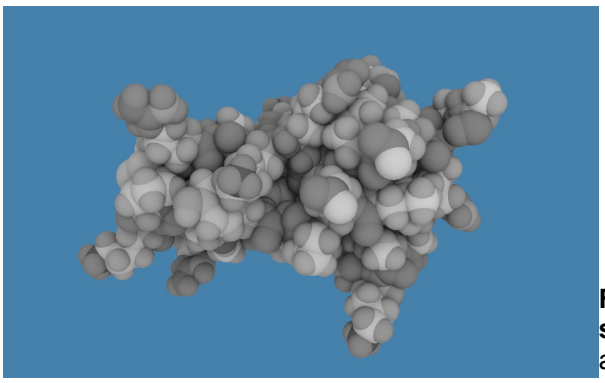


Figure 67. Hydropathy on van der Waals surface. Atoms are coloured according to the atomic lipophilic potentials.

DISCUSSION

The description of biological phenomena has always made use of graphical presentation, starting from the early botanical and zoological drawings, including famous anatomical folios, that greatly help viewers, professionals and not, to understand and learn about nature.

Since these early times, an artistic component has been included, often unnoticed by viewers, but greatly exploited by the scientists/artists. Even today, the clearest graphical descriptions of natural and artificial subjects are hand- or CG-drawn rather than photographic images. The 'artistic' dimension allows for a better interpretation of the subject, the choice of illumination, and the removal of irrelevant and disturbing effects.

The same attitude has motivated a number of scientists to use various graphical tricks when showing data related to structural features of macromolecules. Although most structural information contained in a .pdb file (a list of atoms and their 3D coordinates) is actually 'readable', biologists typically use graphical programs to explore protein structures; indeed the literature has an abundance of such programs, including some very popular. These programs can transfer the structural information from a linear list of atoms to a 3D display; positional information is interpreted with the aid of chemical information stored in libraries (of amino acids, nucleotides etc.), that introduce chemical bonds, electric (partial) charges, hydrophobicity scales and so on. In this way the user is enabled to observe features of the molecules of interest according to her/his needs.

Recent years have seen the development of 3D computer graphics techniques that have culminated in the success of the blockbuster movie Avatar, in which an entire world has been created in CG, including 'floating mountains' and forest with thousands of (CG built!) plants, animals, insects etc.

Similar techniques can be used to show the nanoscopic world of cells, populated with all sorts of environments, proteins, nucleic acids, membranes, small molecules and complexes. Indeed, there are several remarkable examples of efforts in this new discipline of Bio Animation, some of which have reached a large public. Besides the beauty and the educational value of these animations, we consider that the very process of creating such movies includes a heuristic importance both in the development of the graphical instruments and in the

studies implied in the elaboration of the subjects' (proteins) movements and interactions.

Our group is among those involved in the development of animated biology, and my role in this thesis has focused on one aspect of such effort, namely the elaboration, using Blender, of a code capable of showing two of the most critical features that determine the behaviour of macromolecules: their electrostatic and lipophilic potentials.

Choice of Blender

Among the professional packages developed for CG, one only has the double advantage of being open source and available free of charge: Blender.

Blender is the result of a world-wide, concerted effort to put tools of the highest standard for CG creations at the reach of any artist (or scientist) regardless of her/his capability of paying for such tools. The project is guided by the non-profit Blender Foundation, and animated by countless developers that voluntarily devote time and effort to constantly introduce the most up to date techniques into the package, equipping users with any instrument they need. Blender 2.5, the latest major release, introduced a new design of the user interface, new physics engine for smoke (volumetrics), particles and soft bodies, among others. An important, new feature is the possibility to achieve all Blender's functions from scripting, through APIs. This is a very important characteristic that enables the use of Blender, including modelling, animation, special effects, rendering without using the interface.

BioBlender

On the framework on Blender 2.5, we built BioBlender, which includes a section specifically built for biological work. Inside BioBlender, for the analysis of proteins structures, various types of visualization are available: alpha carbon, main chain, main chain and side chains, all atoms (including hydrogens) and molecular surface. The elaboration of proteins' motions and the simultaneous representation of surface physico-chemical properties of proteins in motion are the innovations that BioBlender introduces in macromolecular visualization.

Elaboration of protein motion

We use Blender's Game Engine to elaborate the movement of proteins, when more than one conformational state is known. Starting from data from NMR collections or X-ray of the same protein crystallized in different conditions, we

use Blender GE, equipped with special rules approximately simulating atomic behaviour, to interpolate between known conformations and obtain a physically plausible sequence of intermediate conformations. This sequence can be explored within Blender or can be output as a list of pseudo .pdb file (list of atoms and x,y,z coordinates) which are the basis for the visual elaboration.

It is important to notice that this procedure can be applied to any .pdb or (better) sequence of .pdb files representing a continuous series describing a conformational transition, obtained by Blender or by any other means, e.g. Molecular Dynamics simulation.

Visualization of moving proteins with their molecular surface features

The development of structural biology that made available tens of thousands of structures, not only improved our knowledge on structural features such as the richness of protein folds (secondary and tertiary structure), and of their association in groups (quaternary structure). It also increased knowledge associated with protein motion: in fact most proteins exert their function through some kind of motion. This is best understood by observing the movement in an animated film. The role of side chains, which are the determinants of such motions, is at present difficult to appreciate by using present visualization tools that either provide a fixed all-atom structure, or show dynamically only a limited number of atoms.

We have presented here a procedure that allows the direct observation of moving proteins focusing on their surface features, rather than their structure. In particular, we have focused on hydropathy and electrical fields as they appear, and change on and around the molecular surface.

These features can be calculated and visualized by a number of programs, which typically display them with a colour code. We reasoned that for these properties a more 'photorealistic' display would help viewers in the de-codification of their meaning, and elaborated the system here reported. The visualization of molecules physico-chemical properties using Computer Graphics was a huge challenge. Example of the use of these codes can be seen on Proteopedia page (http://proteopedia.org/wiki/index.php/Calmodulin_in_motion) for a single protein and in our movie Protein Expressions – Study N3 (http://www.scivis.ifc.cnr.it/index.php/videos) for a complex of proteins.

The representation of both features in a black and white display allows the

viewer to grasp their values, without distracting with arbitrary information which is not interpretable if not associated with a de-coding legend, making it easier to interpret.

For MLP elaboration we did not consider any of the available programs accurate enough to provide useful information: most molecular displaying packages simply attribute a fixed value of hydropathy to every atom of a given aminoacid, using the Kyte-Doolittle scale. This scale was elaborated almost 30 years ago with the aim of identifying structural features of proteins, namely the interior portions of globular proteins and membrane spanning segments in membrane associated proteins, but is not indicated for the evaluation of the distribution of hydropathy on the molecular surface. Indeed, some other program includes a more appropriate method of calculation, such as VASCo [Steinkellner2009] which employs the Brickman formula on an atom based library and a Fermi-type distance function. We have implemented a calculation that uses Broto atomic values library, and integrates atoms value in the space around the molecule by the Testa formula, which uses an atom-based fragment scheme and an exponential distance function.

The values thus obtained are plotted on the vertices of the molecular surface by simple trilinear interpolation. These values are then converted into vertex colours (grey level): brigh areas representing hydrophobic regions and dark areas hydrophilic ones. This procedure results in a very smooth distribution of MLP values which is then displayed with a scale of 'tactile' textures, ranging from dull-rough for hydrophilicity to shiny-smooth for hydrophobicity.

The advantage of such calculation and representation is mostly noticeable in animated movies showing the transition between different conformations of proteins, when patches of hydrophobic areas are gradually exposed, which will facilitate docking onto other macromolecules.

For EP, calculated using Poisson-Boltzmann equation, we developed a visual code based on animated particles (small lines) flowing along field lines from positive towards negative charges, proportional to the total charge of the protein; this is particularly useful for the observation of interacting molecules and for molecules whose field is changing when the conformation changes.

To elaborate EP we used several programs and integrated them in a flow whose final result is the continuous display of the EP and its development during

protein conformational transitions. Because this flow has to be repeated for every frame of the sequence, we put particular effort in the consistency of all steps.

Proteins and their surface properties can also be visualized in a 3D interactive way on web platform exploiting the new WebGL [Group2009] component of HTML5. Using this API, it is possible to display 3D content in a web page without the use of external plugins, by writing an appropriate visualization program using the OpenGL syntax. Using a javascript support library, SpiderGL, we built an interactive visualization scheme which accepts as input the meshes, curves representing the field lines and texture images for MLP calculated by BioBlender. The use of SpiderGL for biological visualization permits the 3D Interactive investigation of surface properties of still proteins. This kind of analysis is limited in movies, as the sequence is set by the author. Conversely, 3D Interactive visualization does not allow for proteins animations.

The protein that we have used as example is Calmodulin: after activation due to the binding of 4 Calcium ions, the protein undergoes a major conformational transition in which both its EP and its MLP change considerably: the Ca ions introduced in the 4 EF hands affect both the EP, by virtue of their own charge, and MLP by inducing the opening of each globular domain to expose two major hydrophobic patches which enable the protein to interact with its partners and push the calcium signal downstream in the biochemical pathway.

Since the calculations involved in the elaboration of both EP and MLP are computationally heavy and involve large data sets, we have developed BioBlender to automatically elaborate them, with limited human supervision.

CONCLUSIONS AND FUTURE PERSPECTIVES

In conclusion, we have developed a computational instrument, BioBlender, which is a combination of a Computer Graphics tool and various scientific programs. It allows the display molecular surfaces of moving (or still) proteins and other macromolecules, putting special emphasis on their electrical and lipophilic properties. We consider that this representation will allow better (or at least more immediate and intuitive) understanding of the dynamical forces governing intermolecular interactions and thus facilitate new insights and discoveries.

Our system permits the fast morphing of proteins, elaborating transition between conformations in a fast and reliable way, using Blender Game Engine. BioBlender includes also a novel, intuitive code for the visualization of physico-chemical characteristics, which contributes to gain insight into the function of molecules by drawing viewer's attention to the most active regions of the protein.

EP and MLP are shown simultaneously for each intermediate conformation of moving proteins avoiding the use of colour, which cannot be interpreted without a legend. Using real world tactile/sight feelings, the nanoscale world of proteins becomes more understandable, familiar to our everyday life, making it easier to introduce “un-seen” phenomena (concepts) such as hydropathy or charges, while leaving the utilization of colour space for the description of other biochemical information.

BioBlender is a new tool proposed for biological work and only a fraction of its potential was developed. Biological activities are characterized by many other features, and visualizing all of them is important for a better understanding of processes; additional concepts such as pH, oxidative/reduction potential, numerically calculated surface curvature, etc. can be introduced.

Biological processes at molecular and atomic level take place crossing different time scales from 10^{-12} s for side chain rotations to around 10^{-6} s for conformational transitions and up to seconds for protein folding. Time scales are useful concepts that can be introduced in BioBlender to distinguish between fast movements (vibrations) and slow ones (conformational changes of

biomolecules). It would be appropriate to develop a method to extract information regarding protein transitions, without being 'disturbed' by vibrations.

Our system calculates the mesh for proteins in motion in a frame-by-frame manner on the basis of atomic coordinates. This procedure is time consuming, and the process could be improved by developing a system that updates a mesh through its components, for example vertices could be moved, or even added and removed when necessary. This method would allow visualization of the conformational changes of protein surfaces directly in BioBlender workspace.

Interactivity is another interesting feature that can be added to BioBlender functionalities to offer the user the possibility to navigate through cellular environment and discover biological activities.

Finally, since we develop intuitive representations for biomolecules properties, it is important to perform perception tests with users of different background to evaluate them.

REFERENCES

Abrahams, J. P., Leslie, A. G., Lutter, R. & Walker, J.E. (1994) **Structure at 2.8 A resolution of F1-ATPase from bovine heart mitochondria** Nature 370:621-8

Alagona, G., Ghio, C. & Monti, S. (2004) **S. B3LYP/6-31G* conformational landscape in vacuo of some pterocarpan stereoisomers with biological activity.** Journal Cover:Phys. Chem. Chem. Phys., 2004, 6, 2849-2857 6:2849-2857

Alagona, G., Ghio, C. & Monti, S. (2007) **A Test Case for Time-Dependent Density Functional Theory Calculations of Electronic Circular Dichroism: 2-Chloro-4-Methoxy-6-[(R)-1-Phenylethylamino]-1,3,5-Triazine** Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta) 117:783-803

Alberts, B. (1998) **The cell as a collection of protein machines: preparing the next generation of molecular biologists** Cell 92:291-4

Allan, J. S., Coligan, J. E., Lee, T. H., McLane, M. F., Kanki, P. J., Groopman, J. E. & Essex, M. (1985) **A new HTLV-III/LAV encoded antigen detected by antibodies from AIDS patients** Science 230:810-3

Andrei, R. M., Pan, M. C. & Zoppè, M. (2010) **BioBlender: Blender for Biologists.** BlenderArt Magazine 31:27-32

Aristotle & Ross, W.D. (1924) **Aristotle's Metaphysics.**

Audry, E., Dubost, J. P., Colleter, J. C. & Dallet, P. (1986) **Une nouvelle approache des relations structure-activité: le potentiel de lipophilie moléculaire** Eur. J. Med. Chem 21:71-72

Autodesk [**www.autodesk.com**]

Baba, M. L., Goodman, M., Berger-Cohn, J., Demaille, J. G. & Matsuda, G. (1984) **The early adaptive evolution of calmodulin** Molecular Biology and Evolution 1:442-55

Bakan, A., Meireles, L. M. & Bahar, I. (2011) **ProDy: protein dynamics inferred from theory and experiments** Bioinformatics (Oxford, England) 27:1575-7

Baker, N. A., Sept, D., Joseph, S., Holst, M. J. & McCammon, J.A. (2001) **Electrostatics of nanosystems: application to microtubules and the ribosome** Proceedings of the National Academy of Sciences of the United States of America 98:10037-41

Barry, C. D., Bosshard, H. E., Ellis, R. A. & Marshall, G.R. (1974) **Evolving macromodular molecular modeling system** Federation Proceedings 33:2368-72

Beem, K. M., Richardson, D. C. & Rajagopalan, K.V. (1977) **Metal sites of copper-zinc superoxide dismutase** Biochemistry 16:1930-6

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P.E. (2000) **The Protein Data Bank** Nucleic Acids Research 28:235-42

BioBlender [**www.bioblender.org**]

Blender [**www.blender.org**]

Blinn, J. (1978) **Simulation of wrinkled surfaces.** Computer Graphics (Proceedings of SIGGRAPH 1978) 12(3):286-292

Bodor, N., Gabanyi, Z. & Wong, C.K. (1989) **A new method for the estimation of partition coefficient.** Journal of the American Chemical Society 111:289-294

Brasseur, R. (1991) **Differentiation of lipid-associating helices by use of three-dimensional molecular hydrophobicity potential calculations** The Journal of Biological Chemistry 266:16120-7

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S. & Karplus, M. (1983) **CHARMM: A program for macromolecular energy, minimization, and dynamics calculations** J. Comput. Chem. 4:187-217

Brooks, F. P. J. (1977) **The computer 'scientist' as toolsmith -- Studies in interactive computer graphics.** Information Processing 77:625-634

Broto, P., Moreau, G. & Vandycke, C. (1984) **Molecular structures: Perception, autocorrelation descriptor and sar studies. System of atomic contributions for the calculation of the n-octanol/water partition coefficients.** Eu. J. Med. Chem. 19:71-78

Broto, P., Moreau, G. & Vandycke, C. (1984) **Molecular structures: Perception, autocorrelation descriptor and sar studies. System of atomic contributions for the calculation of the n-octanol/water partition coefficients.** Eu. J. Med. Chem. 19.1:71-78

Bullet Physics Library [**www.bulletphysics.org**]

Callieri, M., Andrei, R., Di Benedetto, M., Zoppè, M. & Scopigno, R. (2010) **Visualization methods for molecular studies on the web platform.** Proceedings of the 15th international Conference on Web 3D Technology 117-126

Calmodulin motion on Proteopedia
[**http://proteopedia.org/wiki/index.php/Calmodulin_in_motion**]

Case, D. A., Cheatham, T. E. 3., Darden, T., Gohlke, H., Luo, R., Merz, K. M. J., Onufriev, A., Simmerling, C., Wang, B. & Woods, R.J. (2005) **The Amber biomolecular simulation programs** Journal of Computational Chemistry 26:1668-88

Catmull, E. (1974) **A subdivision algorithm for computer display of curved surfaces**.

Center, R. J., Leapman, R. D., Lebowitz, J., Arthur, L. O., Earl, P. L. & Moss, B. (2002) **Oligomeric structure of the human immunodeficiency virus type 1 envelope protein on the virion surface** Journal of Virology 76:7863-7

Chen, B., Vogan, E. M., Gong, H., Skehel, J. J., Wiley, D. C. & Harrison, S.C. (2005) **Structure of an unliganded simian immunodeficiency virus gp120 core** Nature 433:834-41

Cherfiels, J., Duquerroy, S. & Janin, J. (1991) **Protein-protein recognition analyzed by docking simulation.** Proteins: Struct. Funct. Genet. 11:271-280

Chime [**www.mdli.com**]

Christen, M., Hünenberger, P. H., Bakowies, D., Baron, R., Bürgi, R., Geerke, D. P., Heinz, T. N., Kastenholz, M. A., Kräutler, V., Oostenbrink, C., Peter, C., Trzesniak, D. & van Gunsteren, W.F. (2005) **The GROMOS software for biomolecular simulation: GROMOS05** Journal of Computational Chemistry 26:1719-51

Connolly, M. L. (1983) **Solvent-accessible surfaces of proteins and nucleic acids** Science 221:709-13

Connolly, M. L. (1993) **The molecular surface package** Journal of Molecular Graphics 11:139-41

Cook R L (1984) **Shade Trees.** Computer Graphics (Proceedings of SIGGRAPH) 18(3):223-231

Cooper, G. & Hausman, R. (2007) **The cell: a molecular aproach, forth edition** Sinauer Associates, U.S.A.

Corey, R. & Pauling, L. (1953) **Molecular Models of Amino Acids, Peptides, and Proteins.** Review of Scientific Instruments 24:621-627

Cornette, J. L., Cease, K. B., Margalit, H., Spouge, J. L., Berzofsky, J. A. & DeLisi, C. (1987) **Hydrophobicity scales and computational techniques for detecting amphipathic structures in proteins** Journal of Molecular Biology 195:659-85

Dalgleish, A. G., Beverley, P. C., Clapham, P. R., Crawford, D. H., Greaves, M. F. & Weiss, R.A. (1984) **The CD4 (T4) antigen is an essential component of the receptor for the AIDS retrovirus** Nature 312:763-7

Davis, M. E. & McCammon, J.A. (1990) **Electrostatics in biomolecular structure and dynamics.** Chemical Reviews 90:509-521

Davis, M. E., Madura, J. D., Sines, J., Luty, B. A., Allison, S. A. & McCammon, J.A. (1991) **Diffusion-controlled enzymatic reactions** Methods in Enzymology 202:473-97

Dearden, J. C. (1985) **Partitioning and lipophilicity in quantitative structure-activity relationships** Environmental Health Perspectives 61:203-28

DeLano, W. L. (2002) **The PyMOL molecular graphics system**

Di Benedetto, M., Ponchio, F., Ganovelli, F. & Scopigno, R. (2010) **SpiderGL: a JavaScript 3D graphics library for next-generation WWW.** Proceedings of the 15th International Conference on Web 3D Technology 165-174

Diamond, R. (1981) **Bilder: A computer graphics program for biopolymers and its application to the interpretation of the structure of tobacco mosaic virus protein discs at 2.8 resolution.** Biomolecular Structure, Conformation, Function and Evolution 1:567-588

Diamond, R. (1981) **Bilder: An interactive graphics program for biopolymers.** Computational Crystallography 318-325

Dill, K. A. (1990) **Dominant forces in protein folding** Biochemistry 29:7133-55

Dolinsky, T. J., Czodrowski, P., Li, H., Nielsen, J. E., Jensen, J. H., Klebe, G. & Baker, N.A. (2007) **PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations** Nucleic Acids Research 35:W522-5

Dolinsky, T. J., Nielsen, J. E., McCammon, J. A. & Baker, N.A. (2004) **PDB2PQR: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations** Nucleic Acids Research 32:W665-7

Drenth, J. (1994) **Principles of protein X-ray crystallography** Spinger-Verlag, New York & London

Eisenberg, D., Marcotte, E. M., Xenarios, I. & Yeates, T.O. (2000) **Protein function in the post-genomic era** Nature 405:823-6

Eisenhaber, F. & Argos, P. (1993) **Improved strategy in analytic surface calculation for molecular systems: handling of singularities and computational efficiency** Journal of Computational Chemistry 14:1272-1280

Eisenhaber, F., Lijnzaad, P., Argos, P., Sander, C. & Scharf, M. (1995) **The double cubic lattice method: Efficient approaches to numerical integration of surface area and volume and to dot surface contouring of molecular assemblies** Journal of Computational Chemistry 16:273-284

Electron Microscopy Databank [**http://www.cgl.ucsf.edu/Research/emdb/**]

EMDataBank [**http://www.emdatabank.org/**]

Engelman, D. M., Steitz, T. A. & Goldman, A. (1986) **Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins** Annual Review of Biophysics and Biophysical Chemistry 15:321-53

Fauchère, J. L., Quarendon, P. & Kaetterer, L. (1988) **Estimating and representing hydrophobicity potential.** J Mol Graphics 6:203-206

Feldmann, R. J. & Bing, D.H. (1980) **TAMS: Teaching aids for macromolecular structure. Teachers manual** Division of Computer Research and Technology (DCRT), NIH/PHS/DHEW

Fletterick, R. J. & Matela, R. (1982) **Color-coded alpha-carbon models of proteins.** Biopolymers 21:999-1003

Fletterick, R. J., Schroer, T. & Matela, R.J. (1985) **Molecular Structure.** Blackwell Scientific Publications, Oxford.

Foster, M. P., McElroy, C. A. & Amero, C.D. (2007) **Solution NMR of large molecules and assemblies** Biochemistry 46:331-40

Furet, P., Sele, A. & Cohen, N.C. (1988) **3D molecular lipophilicity potential profiles: a new tool in molecular modeling** J. Mol. Graphics 6:182-189

Gaillard, P., Carrupt, P. A., Testa, B. & Boudon, A. (1994) **Molecular lipophilicity potential, a tool in 3D QSAR: method and applications** Journal of Computer-Aided Molecular Design 8:83-96

Geyer, H., Holschbach, C., Hunsmann, G. & Schneider, J. (1988) **Carbohydrates of human immunodeficiency virus. Structures of oligosaccharides linked to the envelope glycoprotein 120** The Journal of Biological Chemistry 263:11760-7

Ghose, A. K. & Crippen, G.M. (1986) **Atomic Physicochemical Parameters for Three-**

Dimensional Structure-Directed Quantitative Structure-Activity Relationships I. Partition Coefficients as a Measure of Hydrophobicity Journal of Computational Chemistry 7(4):565-577

Ghose, A. K., Viswanadhan, V. N. & Wendoloski, J.J. (1998) **Prediction of Hydrophobic (Lipophilic) Properties of Small Organic Molecules Using Fragmental Methods: An Analysis of AlogP and ClogP Methods** Journal of Physical Chemistry A 102(21):3762-3772

Gilson, M. K., Sharp, K. A. & Honig, B. (1987) **Calculating the electrostatic potential of molecules in solution: method and error assessment.** J. Comp. Chem. 9:327–335

Goodsell, D. (1998) **The machinery of life** Springer-Verlag

Group, T. K. (2009) **WebGL - OpenGL ES 2.0 for the Web** :

Guex, N. & Peitsch, M.C. (1997) **SWISS-MODEL and the Swiss-PdbViewer: an environment for comparative protein modeling** Electrophoresis 18:2714-23

Hansch, C. & Leo, A.J. (1979) **Subsistent constants for correlation analysis in chemistry and biology.** New York: Wiley

Harrison, S. C., Olson, A. J., Schutt, C. E., Winkler, F. K. & Bricogne, G. (1978) **Tomato bushy stunt virus at 2.9 A resolution** Nature 276:368-73

Heiden, W., Moeckel, G. & Brickmann, J. (1993) **A new approach to analysis and display of local lipophilicity/hydrophilicity mapped on molecular surfaces** Journal of Computer-Aided Molecular Design 7:503-14

Hendsch, Z. S. & Tidor, B. (1994) **Do salt bridges stabilize proteins? A continuum electrostatic analysis** Protein Science : a Publication of the Protein Society 3:211-26

Hodis, E., Prilusky, J., Martz, E., Silman, I., Moult, J. & Sussman, J.L. (2008) **Proteopedia - a scientific 'wiki' bridging the rift between three-dimensional structure and function of biomacromolecules** Genome Biology 9:R121

Hodis, E., Schreiber, G., Rother, K. & Sussman, J.L. (2007) **eMovie: a storyboard-based tool for making molecular movies** Trends in Biochemical Sciences 32:199-204

Holmes, K. C. (1998) **A molecular model for muscle contraction** Acta Cryst. A 54:789-97

Holst, M. J., Baker, N. A. & Wang, F. (2000) **Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. Algorithms and examples.** J. Comput. Chem. 21:1319-1342

Honig, B. & Nicholls, A. (1995) **Classical electrostatics in biology and chemistry** Science 268:1144-9

Hopp, T. P. & Woods, K.R. (1983) **A computer program for predicting protein antigenic determinants** Molecular Immunology 20:483-9

Humphrey, W., Dalke, A. & Schulten, K. (1996) **VMD: visual molecular dynamics** Journal of Molecular Graphics 14:33-8, 27-8

Janin, J. (1979) **Surface and inside volumes in globular proteins** Nature 277:491-2

Jmol website [**www.jmol.org**]

Jmol: an open-source Java viewer for chemical structures in 3D. [**www.jmol.org**]

Johnson, C. K. (1965) **OR TEP: A FORTRAN Thermal-Ellipsoid Plot Program for Crystal Structure Illustrations** ONRL Report No 3794. Oak Ridge, Ten., Oak Ridge National Laboratory

Johnson, G. T., Autin, L., Goodsell, D. S., Sanner, M. F. & Olson, A.J. (2011) **ePMV embeds molecular modeling into professional animation software environments** Structure (London, England : 1993) 19:293-303

Jones, T. A. (1978) **A graphics model building and refinement system for macromolecules** J. Appl. Crystallogr. 11:268-272

Jones, T. A. (1981) **FRODO: A graphics fitting program for macromolecules** Computational Crystallography 303-317

Jones, T. A. (1985) **nteractive computer graphics: FRODO.** Methods in Enzymology 115:157-171

Jones, T. A., Zou, J. Y., Cowan, S. W. & Kjeldgaard, M. (1991) **Improved methods for building protein models in electron density maps and the location of errors in these models** Acta Crystallographica. Section a, Foundations of Crystallography 47 (Pt 2):110-9

Kai, L. E. (1997) **NMR methods for the study of protein structure and dynamics** Biochem. Cell. Biol. 75:1-15

KAUZMANN, W. (1959) **Some factors in the interpretation of protein denaturation** Advances in Protein Chemistry 14:1-63

KENDREW, J. C., BODO, G., DINTZIS, H. M., PARRISH, R. G., WYCKOFF, H. & PHILLIPS, D.C. (1958) **A three-dimensional model of the myoglobin molecule obtained by x-ray analysis** Nature 181:662-6

KENDREW, J. C., DICKERSON, R. E., STRANDBERG, B. E., HART, R. G., DAVIES, D. R., PHILLIPS, D. C. & SHORE, V.C. (1960) **Structure of myoglobin: A three-dimensional Fourier synthesis at 2 A. resolution** Nature 185:422-7

Klopman, G. & Iroff, L.D. (1981) **Calculation of partition coefficients by the charge density method.** Journal of Computational Chemistry 2:157-160

Koffka, K. (1935) **Principles of Gestalt Psychology** Harcourt (New York)

Koltun, W. L. (1965) **Precision space-filling atomic models.** Biopolymers 3:665-679

Koradi, R., Billeter, M. & Wüthrich, K. (1996) **MOLMOL: a program for display and analysis of macromolecular structures** Journal of Molecular Graphics 14:51-5, 29-32

Kraulis, P. J. (1991) **Molscript: A program to produce both detailed and schematic plots of proteins structures.** J. Appl. Cryst. 24:946-950

Krieger, E., Koraimann, G. & Vriend, G. (2002) **Increasing the precision of comparative models with YASARA NOVA--a self-parameterizing force field** Proteins 47:393-402

Kubinyi, H. (1979) **Lipophilicity and drug activity** Progress in Drug Research. Fortschritte der Arzneimittelforschung. Progres des Recherches Pharmaceutiques 23:97-198

Kuboniwa, H., Tjandra, N., Grzesiek, S., Ren, H., Klee, C. B. & Bax, A. (1995) **Solution structure of calcium-free calmodulin** Nature Structural Biology 2:768-76

Kyte, J. & Doolittle, R.F. (1982) **A simple method for displaying the hydropathic character of a protein** Journal of Molecular Biology 157:105-32

Laguerre, M., Saux, M., Dubost, J. P. & Carpy, A. (1997) **MLPP: A program for the calculation of molecular lipophilicity potential in proteins.** Pharm. Sci. 3.5-6:217-222

Lee, B. & Richards, F.M. (1971) **The interpretation of protein structures: estimation of static accessibility** Journal of Molecular Biology 55:379-400

Leo, A. J. (1993) **Calculating log Poct from structures** Chemical Reviews 93(4):1281-1306

Leo, A., Hansch, C. & Elkins, D. (1971) **Partition coefficients and their uses** Chem. Reu. 71:525

Levinthal, C. (1966) **Molecular model-building by computer** Scientific American 214:42-52

Lindahl, E., Hess, B. & van der Spoel, D. (2001) **GROMACS 3.0: A package for molecular simulation and trajectory analysis** J. Mol. Mod. 7:306-317

Linderstrøm-Lang, K. U. (1952) **Proteins and Enzymes** Stanford University Press

Lipscomb, J. S. (1981) **Reversed apparent movement and erratic motion with many refreshes per update.** Computer Graphics 14 (4):113-118

MacKerell, A. J., Bashford, D., Bellott, M., Dunbrack, R. J., Evanseck, J., Field, M., Fischer, S., Gao, J., Guo, H., Ha, S., Joseph-McCarthy, D., Kuchnir, L., Kuczera, K., Lau, F., Mattos, C., Michnick, S., Ngo, T., Nguyen, D., Prodhom, B., Reiher, W. I., Roux, B., Schlenkrich, M., Smith, J., Stote, R., Straub, J., Watanabe, M., Wiorkiewicz-Kuczer, A., Yin, D. & Karplus, M. (1998) **All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins.** J Phys Chem B 102:3586-616

Makino, M. (1998) **Dependence of GC-RRTs on the solvent-accessible surface area of dioxins and related compounds** Chemosphere 37:13-26

Martz, E. (2002) **Protein Explorer: easy yet powerful macromolecular visualization** Trends in Biochemical Sciences 27:107-9

Maxon [**www.maxon.net**]

McCormick, B. H., DeFanti, T. A. & Brown, M.D. (1987) Computer Graphics - Visualization in Scientific Computing 21.6:

Miller, J. R., Abdel-Meguid, S. S., Rossmann, M. G. & Anderson, D.C. (1981) **A computer graphics system for the building of macromolecular models into electronic density maps** Journal of Applied Crystallography 14:94-100

Mittermaier, A. & Kay, L.E. (2006) **New tools provide new insights in NMR studies of protein dynamics** Science 312:224-8

MOLMOL [**www. mol.biol.ethz.ch/groups/Wuthrich-groups/software**]

Mueller, T. D. & Feigon, J. (2002) **Solution structures of UBA domains reveal a conserved hydrophobic surface for protein-protein interactions** Journal of Molecular Biology 319:1243-55

Murzin, A. G., Brenner, S. E., Hubbard, T. & Chothia, C. (1995) **SCOP: a structural**

classification of proteins database for the investigation of sequences and structures Journal of Molecular Biology 247:536-40

Naccess V2.1.1 - Atomic Solvent Accessible Area Calculations [**www.bioinf.manchester.ac.uk/naccess/**]

nmrdb [**http://www.nmrdb.org/**]

NMRShiftDB [**http://www.ebi.ac.uk/nmrshiftdb/**]

Noji, H., Yasuda, R., Yoshida, M. & Kinosita, K.J. (1997) **Direct observation of the rotation of F1-ATPase** Nature 386:299-302

Nature Methods Journal, March 2010, Volume 7 No 3s ppS1-S68

Orengo, C. A., Michie, A. D., Jones, S., Jones, D. T., Swindells, M. B. & Thornton, J.M. (1997) **CATH--a hierarchic classification of protein domain structures** Structure (London, England : 1993) 5:1093-108

Pauling, L. & Corey, R.B. (1951) **Configurations of Polypeptide Chains With Favored Orientations Around Single Bonds: Two New Pleated Sheets** Proceedings of the National Academy of Sciences of the United States of America 37:729-40

PAULING, L. & COREY, R.B. (1951) **The pleated sheet, a new layer configuration of polypeptide chains** Proceedings of the National Academy of Sciences of the United States of America 37:251-6

PAULING, L. & COREY, R.B. (1951) **The structure of feather rachis keratin** Proceedings of the National Academy of Sciences of the United States of America 37:256-61

Pedretti, A., Villa, L. & Vistoli, G. (2002) **VEGA: a versatile program to convert, handle and visualize molecular structure on Windows-based PCs** Journal of Molecular Graphics & Modelling 21:47-9

Pettersen, E. F., Goddard, T. D., Huang, C. C., Couch, G. S., Greenblatt, D. M., Meng, E. C. & Ferrin, T.E. (2004) **UCSF Chimera--a visualization system for exploratory research and analysis** Journal of Computational Chemistry 25:1605-12

Pique, M. E., Macke, T. J. & Arvai, A.S. (1991) **Flex: A light-weight molecular display program.** J. Mol. Graphics 9:40-41

Plato & Jewitt, B. (1941) **Plato's The Republic.**

Porozov, Y., Andrei, R. & Zoppè, M. (2007) **Visualization of moving biomolecules: a new approach based on professional 3D animation software** Nettab 2007 Network Tools and Applications in Biology. :

Privalov, P. L. & Gill, S.J. (1988) **Stability of protein structure and hydrophobic interaction** Advances in Protein Chemistry 39:191-234

Protein Data Bank [**www.pdb.org**]

Protein Explorer [**http://proteinexplorer.org**]

Proteopedia [**http://proteopedia.org**]

Rekker, R. F. (1977) **The hydrophobic fragmental constant. Its derivation and application. A**

means of characterizing membrane systems. Amsterdam: Elsevier

Rekker, R. F. & de Kort, H.M. (1979) **The hydrophobic fragmental constant; an extension to a 1000 data point set.** European Journal of Medicinal Chemistry 14:479-488

RenderMan for Maya [**http://renderman.pixar.com/**]

Richards, F. M. (1968) **The matching of physical models to three-dimensional electron-density maps: a simple optical device** Journal of Molecular Biology 37:225-30

Richardson, D. C. & Richardson, J.S. (1992) **The kinemage: a tool for scientific communication** Protein Science : a Publication of the Protein Society 1:3-9

Richardson, J. S. & Richardson, D.C. (1989) **Principles and patterns of protein conformation.**. :pp. 1-98

Richardson, J. S., Richardson, D. C., Thomas, K. A., Silverton, E. W. & Davies, D.R. (1976) **Similarity of three-dimensional structure between the immunoglobulin domain and the copper, zinc superoxide dismutase subunit** Journal of Molecular Biology 102:221-35

Richmond, T. J. (1984) **Solvent accessible surface area and excluded volume in proteins. Analytical equations for overlapping spheres and implications for the hydrophobic effect** Journal of Molecular Biology 178:63-89

Rocchia, W., Sridharan, S., Nicholls, A., Alexov, E., Chiabrera, A. & Honig, B. (2002) **Rapid grid-based construction of the molecular surface and the use of induced surface charge to calculate reaction field energies: applications to the molecular systems and geometric objects** Journal of Computational Chemistry 23:128-37

Rose, G. D., Geselowitz, A. R., Lesser, G. J., Lee, R. H. & Zehfus, M.H. (1985) **Hydrophobicity of amino acid residues in globular proteins** Science 229:834-8

Rubin, B. (1985) **Macromolecule backbone models** Methods in Enzymology 115:391-7

Rubin, B. & Richardson, J.S. (1972) **The simple construction of protein alpha-carbon models** Biopolymers 11:2381-5

SANGER, F. & THOMPSON, E.O.P. (1953) **The amino-acid sequence in the glycyl chain of insulin. I. The identification of lower peptides from partial hydrolysates** The Biochemical Journal 53:353-66

SANGER, F. & TUPPY, H. (1951) **The amino-acid sequence in the phenylalanyl chain of insulin. I. The identification of lower peptides from partial hydrolysates** The Biochemical Journal 49:463-81

Sanner, M. F. (1999) **Python: a programming language for software integration and development** Journal of Molecular Graphics & Modelling 17:57-61

Sanner, M. F., Olson, A. J. & Spehner, J.C. (1996) **Reduced surface: an efficient way to compute molecular surfaces** Biopolymers 38:305-20

Sasaki, Y., Kubodera, H., Matuszaki, T. & Umeyama, H. (1991) **Prediction of octanol/water partition coefficients using parameters derived from molecular structures.** J Pharmacobio-Dyn 14:207-214

Sayle, R. & Bissell, A. (1992) **RasMol: A Program for Fast Realistic Rendering of Molecular Structures With Shadows.** Proceedings of the 10th Eurographics UK '92 Conference, University of Edinburgh, Scotland :

Sayle, R. A. & Milner-White, E.J. (1995) **RASMOL: biomolecular graphics for all** Trends in Biochemical Sciences 20:374

Schlüter, T. [**http://www.thoro.de/page/3dnp-introduction-en**]

Schüttelkopf, A. W. & van Aalten, D.M.F. (2004) **PRODRG: a tool for high-throughput crystallography of protein-ligand complexes** Acta Crystallographica. Section D, Biological Crystallography 60:1355-63

Science, 18 February 2011 vol 331, issue 6019, pages 807-974

Sharp, K. A. & Honig, B. (1990) **Electrostatic interactions in macromolecules: theory and applications** Annual Review of Biophysics and Biophysical Chemistry 19:301-32

Shaw, D. E., Maragakis, P., Lindorff-Larsen, K., Piana, S., Dror, R. O., Eastwood, M. P., Bank, J. A., Jumper, J. M., Salmon, J. K., Shan, Y. & Wriggers, W. (2010) **Atomic-level characterization of the structural dynamics of proteins** Science 330:341-6

Sheinerman, F. B., Norel, R. & Honig, B. (2000) **Electrostatic aspects of protein-protein interactions** Current Opinion in Structural Biology 10:153-9

Sitkoff, D., Sharp, K. & Honig, B. (1994) **Accurate Calculation of Hydration Free Energies Using Macroscopic Solvent Models.** The Journal of Physical Chemistry 98:1978-88

Southall, N. T., Dill, K. A. & Haymet, A.D.J. (2002) **A View of the Hydrophobic Effect** Journal of Physical Chemistry B 106(3):521-533

Steinkellner, G., Rader, R., Thallinger, G. G., Kratky, C. & Gruber, K. (2009) **VASCo: computation and visualization of annotated protein surface contacts** BMC Bioinformatics 10:32

Stock, D., Leslie, A. G. & Walker, J.E. (1999) **Molecular architecture of the rotary motor in ATP synthase** Science 286:1700-5

Stotz, R. (1963) **Man-machine console facilities for computer-aided design.** AFIPS Conference Proceedings 23:323-328

Studer, D., Humbel, B. M. & Chiquet, M. (2008) **Electron microscopy of high pressure frozen samples: bridging the gap between cellular ultrastructure and atomic resolution** Histochemistry and Cell Biology 130:877-89

Sutaria, S. D. (1984) **Specific Learning Disabilities: Nature and Needs** Charles C Thomas Pub Ltd

Svedberg, T. & Nichols, J.B. (1927) **The application of the oil turbine type of ultracentrifuge to the study of the stability region of carbon monoxide-hemoglobin** Journal of the American Chemical Society 49 (11):2920–2934

Swarbrick, J. D., Buyya, S., Gunawardana, D., Gayler, K. R., McLennan, A. G. & Gooley, P.R. (2005) **Structure and substrate-binding mechanism of human Ap4A hydrolase** The Journal

of Biological Chemistry 280:8471-81

Swiss-PdbViewer [**www.expasy.org/spdbv/**]

Tainer, J. A., Getzoff, E. D., Beem, K. M., Richardson, J. S. & Richardson, D.C. (1982) **Determination and analysis of the 2 A-structure of copper, zinc superoxide dismutase** Journal of Molecular Biology 160:181-217

Tan, C., Yang, L. & Luo, R. (2006) **How Well Does Poisson-Boltzmann Implicit Solvent Agree with Explicit Solvent? A Quantitative Analysis.** J Phys Chem B 110:18680-7

Tanford, C. (1973) **The Hydrophobic Effect: Formation of Micelles and Biological Membranes.** Wiley, New York

Tarini, M., Cignoni, P. & Montani, C. (2006) **Ambient occlusion and edge cueing to enhace real time molecular visualization.** IEEE Transaction on Visualization and computer Graphics 12

Testa, B., Carrupt, P. A., Gaillard, P., Billois, F. & Weber, P. (1996) **Lipophilicity in molecular modeling** Pharmaceutical Research 13:335-43

The PyMOL Molecular Graphics System, Schrödinger, LLC. [**www.pymol.org**]

Tsernoglou, D., Petsko, G. A., McQueen, J. E. J. & Hermans, J. (1977) **Molecular graphics: application to the structure determination of a snake venom neurotoxin** Science 197:1378-81

Tsodikov, O. V., Record, M. T. J. & Sergeev, Y.V. (2002) **Novel computer program for fast exact calculation of accessible and molecular surface areas and average surface curvature** Journal of Computational Chemistry 23:600-9

Tzakos, A. G., Grace, C. R. R., Lukavsky, P. J. & Riek, R. (2006) **NMR techniques for very large proteins and rnas in solution** Annual Review of Biophysics and Biomolecular Structure 35:319-42

UCSF Chimera [**www.cgl.ucsf.edu/chimera/**]

van de Waterbeemd, H. & Testa, B. (1987) **The parametrization of lipophilicity and other structural properties in drug design** Advances in Drug Research 16:87-227

Veronese, F. D., DeVico, A. L., Copeland, T. D., Oroszlan, S., Gallo, R. C. & Sarngadharan, M.G. (1985) **Characterization of gp41 as the transmembrane protein coded by the HTLV-III/LAV envelope gene** Science 229:1402-5

Visual Molecular Dynamics [**www.ks.uiuc.edu/Research/vmd/**]

Wang, J., Cieplak, P. & Kollman, P.A. (2000) **How well does a restrained electrostatic potential (RESP) model perform in calculating conformational energies of organic and biological molecules?** Journal of Computational Chemistry 21:1049–1074

Wang, Z. & Duan, Y. (2004) **Solvation effects on alanine dipeptide: A MP2/cc-pVTZ//MP2/6-31G** study of (Phi, Psi) energy maps and conformers in the gas phase, ether, and water** Journal of Computational Chemistry 25:1699-716

Weber, J. R. (2009) **ProteinShader: illustrative rendering of macromolecules** BMC Structural Biology 9:19

Wimley, W. C., Gawrisch, K., Creamer, T. P. & White, S.H. (1996) **Direct measurement of salt-bridge solvation energies using a peptide model system: implications for protein stability** Proceedings of the National Academy of Sciences of the United States of America 93:2985-90

World Index of Molecular Visualization Resources [**www.molvisindex.org**]

Wright, W. V. (1981) **GRIP -- An interactive computer graphics system for molecular studies.** Computational Crystallography :294-302

Wüthrich, K. (1995) **NMR - this other method for protein and nucleic acid structure determination** Acta Crystallographica. Section D, Biological Crystallography 51:249-70

Xu, D., Lin, S. L. & Nussinov, R. (1997) **Protein binding versus protein folding: the role of hydrophilic bridges in protein associations** Journal of Molecular Biology 265:68-84

Xu, Z. & Sigler, P.B. (1998) **GroEL/GroES: structure and function of a two-stroke folding machine** Journal of Structural Biology 124:129-41

Xu, Z., Horwich, A. L. & Sigler, P.B. (1997) **The crystal structure of the asymmetric GroEL-GroES-(ADP)7 chaperonin complex** Nature 388:741-50

Yang, F., Moss, L. G. & Phillips, G.N.J. (1996) **The molecular structure of green fluorescent protein** Nature Biotechnology 14:1246-51

Zhou, H. X. (1994) **Macromolecular electrostatic energy within the nonlinear Poisson-Boltzmann equation.** J. Chem. Phys. 100:3152–3162

Zhou, Z. H. (2011) **Atomic resolution cryo electron microscopy of macromolecular complexes** Adv Protein Chem Struct Biol 82:1-35

APPENDIX: SCRIPTS

MLP.py

```
# MLP.py imports the .obj files and converts the MLP values stored on V to colour vertex
# Raluca Andrei  <r.andrei@sns.it> from SciVis group www.scivis.ifc.cnr.it

import Blender
from Blender import Mathutils
from Blender.Draw import *
from math import *
from Blender import *
from Blender import Material
import time
import datetime
import sys
import bpy

sys.path.append('')    # PLACE THE PATH TO YOUR OBJ FILES HERE
import import_obj
from import_obj import subtry

def import_mesh():
       #import object
       file=".obj"       #PLACE YOUR FILES NAME HERE
       import_obj.subtry(file)
       sc = Scene.GetCurrent()
       sc.update
       Window.RedrawAll()
       ob = sc.objects.active
       me = ob.getData(mesh=1)
       me.name = "mesh"
       #create material
       mat = Material.New('Material')
       mat.mode |= Material.Modes.VCOL_PAINT
       mat.mode |= Material.Modes.SHADELESS
       mat1 = bpy.data.materials['Material']

       #assign material to object
       for me in bpy.data.meshes:
```

```
            me.materials = [mat1]
        #remove double vertices and smooth the mesh
        me.remDoubles(0.001)
        me.vertexColors=1
        faces=me.faces
        for f in faces:
            f.smooth=1
        #convert the MLP values from V and assign to vertex color
        for i in range(len(faces)):
            for e in range (len(faces[i].verts)):
                        val = faces[i].uv[e][1]
                        val1=int(ceil(((val+3.0)/6.0)*255))      #change  the
range of MLP if necessary!!!!!!
                                faces[i].col[e].r = val1
                                faces[i].col[e].g = val1
                                faces[i].col[e].b = val1
                        else:
                                val1=int(ceil(((val+1.0)/2.0)*255))
                                faces[i].col[e].r = val1
                                faces[i].col[e].g = val1
                                faces[i].col[e].b = val1
                        me.update()
        #create new UV layer and unwrap the mesh
        me.addUVLayer('UVBake')
        me.activeUVLayer = 'UVBake'
        me.renderUVLayer = 'UVBake'
        Blender.Run('uvcalc_smart_project.py')
        #import image used for baking the vertex colour of the mesh
        img = Blender.Image.Load("bake.png" )
        img.updateDisplay()
        img.makeCurrent()
        Window.RedrawAll()
        for f in me.faces:
            f.image = img
        context = sc.getRenderingContext()
        #bake the texture
        bakeMode = 'TEXTURE'
        context.bake()
        #save the current image; the UV/Image Editor should be open
        image = Image.GetCurrent()
        image.setFilename(".png")  # PLACE THE NAME OF YOUR IMAGE HERE
        image.save()
        #unlink the mesh from the object
        me.fakeUser=True
        sc.update()
```

```
	Window.RedrawAll()
	#delete the object
	sc.unlink(ob)
#save all the meshes in a .blend file
import_mesh()
Blender.Save(".blend",0) #PLACE THE NAME OF YOUR .BLEND FILE HERE
```

texture.py

```
# this script assigns the texture to the material of the mesh imported previously. It uses as textures the image obtained with MLP.py and and the image with the noise added
#Raluca Andrei  <r.andrei@sns.it> from SciVis group www.scivis.ifc.cnr.it


import bpy
import Blender
from Blender import *
import Blender, meshtools, os, struct, sys, string
import time
import datetime
sys.path.append('') # PLACE THE PATH TO YOUR OBJ FILES HERE


#import object
import import_obj
ob= Blender.Object.Get("") # PLACE THE NAME OF YOUR FIXED OBJECT TO WHICH YOU ASSIGN THE MESHES YOU IMPORTED PREVIOUSLY WITH MLP.py
mesheslist0 = Mesh.Get()
mesheslist = []
for m in mesheslist0:
	strname=str(m.name)
	strname=strname[0:4]
	if m.name!="None_out" and strname=="mesh":
		mesheslist.append(m)
meshindex=0
me=mesheslist[meshindex]
#create material without specularity
mat = Material.New('Material_New')
mat.setSpec(0.0)
mat_1 = bpy.data.materials['Material_New']
#assign material to object
me.materials = [mat_1]
#create the first texture Image
comptex=Texture.New('bump')
comptex.setType('Image')
img1 = Image.Load('.png')	#PLACE THE NAME OF YOUR NOISY IMAGE HERE
# link the image to the texture
```

```
comptex.image = img1
# get the material
mat2 = Material.Get('Material_New')
# set the material's first texture
mat2.setTexture(0, comptex)
#get the texture and use the material's UV to map the texture
mtex = mat2.getTextures()
mtex[0].texco=Texture.TexCo.UV
#map the texture to COL and NOR
mtex[0].mapto=Texture.MapTo.NOR
#set the values for col and nor
mtex[0].colfac=0.5
mtex[0].norfac=3.3
#create a second texture
spectex=Texture.New('specular')
spectex.setType('Image')
img2 = Image.Load('.png')    #PLACE HERE THE NAME OF YOUR IMAGE  USED TO MAP IT
ON THE SPECULAR CHANNEL
# link the image to the texture
spectex.image = img2
# get the material
mat2 = Material.Get('Material_New')
# set the material's first texture
mat2.setTexture(1, spectex)
#get the texture and use the material's UV to map the texture
mtex = mat2.getTextures()
mtex[1].texco=Texture.TexCo.UV
#map the texture to SPEC
mtex[1].mapto=Texture.MapTo.SPEC
#set the value for spec
mtex[1].varfac=0.47
#check no RGB
mtex[1].noRGB=True
#create a thrird texture
coltex=Texture.New('color')
coltex.setType('Image')
img3 = Image.Load('.png')   #PLACE HERE THE NAME OF YOUR IMAGE  USED TO MAP IT
ON THE SPECULAR CHANNEL
# link the image to the texture
coltex.image = img3
# get the material
mat2 = Material.Get('Material_New')
mat2.setTexture(2, coltex)
#get the texture and use the material's UV to map the texture
mtex = mat2.getTextures()
mtex[2].texco=Texture.TexCo.UV
```

```
#map the texture to SPEC
mtex[2].mapto=Texture.MapTo.COL
Window.RedrawAll()
```

import_curves.py

```
#this script reads the vertices of the curves from the .txt files and builds a mesh that contains all the starting vertices of all curves. This script must be run in win32lightcut090828.exe blender branch as only this one has API to draw the particles as lines

# Raluca Andrei <r.andrei@sns.it> from SciVis group www.scivis.ifc.cnr.it


import Blender
from Blender import *
import os
import bpy
from Blender.Scene import Render


def import_file():
	filename=".txt"	#PLACE HERE THE PATH TO THE LINES FILES
	f = open(filename,"r")
	ik=0	#for the iterations of curves
	ibt=0	#for the iterations of the points of the curve
	try:
		scn = Scene.GetCurrent()
		context = scn.getRenderingContext()
		context.currentFrame(1)
		me = bpy.data.meshes.new('')  #CREATE A MESH AND GIVE IT A NAME
		obj = scn.objects.new(me, '')  #CREATE AN OBJECT TO WHICH THE MESH IS ATTACHED
		mat = Material.New('')  #CREATE A MATERIAL AND GIVE IT A NAME
		mat.setMode('Halo')
		mat.setHaloSize(0.1)
		for line in f:
			file_line=line.split()
			if file_line[0]=="n":
				cu = Curve.New() #for every n found it creates a new curve
				ik=ik+1
				ob = scn.objects.new(cu)
				ob.name = "curve_" + str(ik) #the curves get an incremental name
				ibt=0 #the interation of the points of the curve goes down to 0 when the building of a new curve is started
			elif file_line[0]=="v":
				ibt=ibt+1
				bt1coor=float(file_line[1])	#get the x,y,z coordinates of every vertex
```

```
                        bt2coor=float(file_line[2])
                        bt3coor=float(file_line[3])
                        bt = BezTriple.New(bt1coor,bt2coor, bt3coor)
                        if ibt==1:
                              curb = cu.appendNurb(bt) #the first point is
appended to the curve
                              me.verts.extend(bt1coor,bt2coor,bt3coor)   #add
the vertices to the mesh
                        elif ibt>1:
                              curb.append(bt) #the other points are appended
to the curve
                        AUTO = BezTriple.HandleTypes.AUTO
                        for point in curb:
                              point.handleTypes = [AUTO, AUTO]
                              cu.update()
                              cu.setFlag(30)
                        ob.setPIType(5)  #set the curve as Curve Guide
                        ob.setPIUseMaxDist(1)
                        ob.setPIMaxDist(0.05)
            part_sys = Particle.New ('')  #CREATE A PARTICLE SYSTEM, IT SHOULD
HAVE THE SAME NAME AS THE OBJECT
            part_sys.particleDistribution=0 #emits from verts
            part_sys.randemission=1  #emits random
            part_sys.startFrame=0
            part_sys.endFrame=1
            part_sys.amount=200  #the number of particles
            part_sys.lifetime=12  #the particles' life time
            part_sys.drawAs = Particle.DRAWAS.LINE   #particles drawn as lines
            for me in bpy.data.meshes:
                  me.materials = [mat] #assign the material to the particle
system
      finally:
            f.close()
import_file()
Blender.Save("",0)  #PLACE HERE THE PATH OF THE .BLEND FILE YOU WANT TO SAVE
```

render.py

```
#this script picks the mesh for every frame, assigns the correct images
textures to the material and renders
#Raluca Andrei  <r.andrei@sns.it> from SciVis group www.scivis.ifc.cnr.it

import string
from string import *
import Blender
from Blender import *
```

```
from Blender.Scene import Render
import sys


scn = Scene.GetCurrent()
context = scn.getRenderingContext()
Render.EnableDispWin()
context.extensions = True
mesheslist0 = Mesh.Get()
mesheslist = []
for m in mesheslist0:
	strname=str(m.name)
	strname=strname[0:4]
	if m.name!="" and strname=="mesh":    #PLACE HERE THE NAME OF THE DEFAULT MESH OF THE OBJECT
		mesheslist.append(m)
for ik in range():     #PLACE HERE THE RANGE OF OBJ FILES SEPARATED BY COMMA!!!!!!  FOR LARGER MOLECULES SPLIT YOUR RANGE OF FILES IN MORE RANGES
	j=1+ik
	Blender.Set("curframe", j)
	fra = str(Blender.Get("curframe"))
	print str(fra) + "==" + str(j)
	addzeros = 4 - len(str(j))
	toprepend=""
	for i in range(addzeros):
		toprepend += "0"
	name = "mesh" + str(j)
	mat=Material.Get()
	mat1=mat[0]
	mat1.setMode()
	nomecompos =  "" + toprepend + str(j) + ".png"     #PLACE HERE THE NAMES OF THE IMAGES WITH NOISE TO MAP ON BUMP
	nomespec =""  + toprepend + str(j) + ".png"      #PLACE HERE THE NAMES OF THE IMAGES WITH NOISE TO MAP ON SPECULAR
	nomecol =""  + toprepend + str(j) + ".png"       #PLACE HERE THE NAMES OF THE IMAGES WITH NOISE TO MAP ON COLOR
	for obx in scn.objects:
		obname=obx.getName()
		if obname=="":     #PLACE HERE THE NAME OF THE OBJECT TO WHICH ALL MESHES ARE ASSIGNED
			fixobj=obx
	text1 = Blender.Texture.Get("bump")
	text2 = Blender.Texture.Get("specular")
	text3 = Blender.Texture.Get("color")
	Mesh.Get(name).activeUVLayer="UVBake"
	fixobj.link(Mesh.Get(name))
	mymesh=fixobj.getData(mesh=1)
	objcomp=Blender.Image.Load(nomecompos)
```

```
        objspec=Blender.Image.Load(nomespec)
        objcol=Blender.Image.Load(nomecol)
        text1.setImage(objcomp)
        text2.setImage(objspec)
        text3.setImage(objcol)
        scn.update()
        Blender.Window.RedrawAll()
        fixobj.makeDisplayList()
        context.sFrame = Blender.Get("curframe")
        context.eFrame = Blender.Get("curframe")
        context.renderAnim()
        scn.update()
        Blender.Window.RedrawAll()
        Render.CloseRenderWindow()
        objcomp.glFree()
        objspec.glFree()
        objcol.glFree()
#Blender.Quit()
```

MOVIES (LINKS)

Video gallery: http://www.scivis.ifc.cnr.it/index.php/videos.html

NANOPLANET - An expedition to the cell http://vimeo.com/37182826

PROTEIN EXPRESSIONS - Study N 3 http://vimeo.com/12363247 (soon available on Science Magazine website)

PROTEIN EXPRESSIONS - Study N 3D http://vimeo.com/10979476

PROTEIN EXPRESSIONS - Study N 2 http://vimeo.com/7219809

PROTEIN EXPRESSIONS - Study N 1 http://vimeo.com/7533123

TSH Receptor on Red Blood Cells (English version) http://vimeo.com/30072649

TSH Receptor on Red Blood Cells (Italian version) http://vimeo.com/30072371

BPTI http://vimeo.com/23641380

gp120 http://vimeo.com/15800994

Triazine http://vimeo.com/7391082

BitucarpinA http://vimeo.com/7390965

Calmodulin http://vimeo.com/7390939

Cholesterol http://vimeo.com/7390734

GFP http://vimeo.com/7390920

ATP http://vimeo.com/7390896

Alanine Dipeptide http://vimeo.com/7390816

NOMINATIONS AND AWARDS

PROTEIN EXPRESSIONS – Festival Videominuto, September 2009, Prato

PROTEIN EXPRESSIONS Study N 2 – Finalist in the Suzanne Award Film selection, Blender Conference 2009, Amsterdam

PROTEIN EXPRESSIONS Study N 3 – Science Visualization Challenge 2010

PROTEIN EXPRESSIONS Study N 3D – Scienza in piazza, Bologna, February, 2011

PROTEIN EXPRESSIONS Study N 3 – European Short Film Festival at MIT, Boston, April 2011

PROTEIN EXPRESSIONS Study N 3D – DogVille Viladecans Film Festival, April 2011

PROTEIN EXPRESSIONS Study N 3D – Dentro il microscopio: Ottica, immagini, tecnologie. Mostra didattico-divulgativa sulla microscopia, Pisa, May 2011

PROTEIN EXPRESSIONS Study N 3D – 8th Annual LA 3-D Movie Festival, Los Angeles, May 2011.

PROTEIN EXPRESSIONS Study N 3 – Vedere la Scienza Festival, Milan, May 2011

PROTEIN EXPRESSIONS Study N 3D – Oaxaca International Film Festival, July 2011

PROTEIN EXPRESSIONS Study N 3 – Imagine Science Film Festival, New York, October 2011

TSH receptor – View Festival, Turin, October 2011

PROTEIN EXPRESSIONS Study N 3D – Festival della Scienza, Genova, October-November 2011

NANOPLANET - An Expedition to the Cell – Biomolecular Discovery Dome at the Biophysical Society 56th Annual Meeting, San Diego, February 2012

PROTEIN EXPRESSIONS Study N 3D – “Spazio MeM” special mention of the jury at Melzo Film Festival, July 2010

Lipid Raft image – 1st place at Art&Science contest at the Biophysical Society 56th Annual Meeting in San Diego, February 2012

PUBLICATIONS

Porozov Y. **Andrei R.**, Zoppè M., Visualization of moving biomolecules: a new approach based on professional 3D animation software **Nettab 2007 Network Tools and Applications in Biology.** 12 - 15 June 2007, Computer Science Department, University of Pisa, Italy

Callieri, M., **Andrei R.**, Di Benedetto M., Zoppè M, Scopigno R. (2010) **Visualization methods for molecular studies on the web platform.** Proceedings of the 15th international Conference on Web 3D Technology 117-126

Andrei R., Pan M., Zoppè M. (2010) **BioBlender: Blender for Biologists.** BlenderArt Magazine 31:27-32

Andrei R., Callieri M., Zini M.F., Loni T., Maraziti G., Pan M., Zoppè M. **Intuitive visualization of surface properties of biomolecules.** BMC Bioinformatics 2012, **13**(Suppl 4):S16

Visualization of moving biomolecules: a new approach based on professional 3D animation software

Yuri Porozov[1,2], Raluca Andrei[1,2], Monica Zoppè[1,*].

[1]Scientific Visualization Unit, IFC – CNR, via Moruzzi 1, Pisa. Italy.
[2]Laboratorio di Biologia Molecolare, Scuola Normale Superiore, Pisa, Italy.
[*]Corresponding Author. mzoppe@ifc.cnr.it
www.scivis.ifc.cnr.it

ABSTRACT
We are setting up a system that enable us to visualize proteins and other biological molecules in a 3D virtual environment built according to scientific information and physico-chemical properties. This system will permit a novel view and understanding of the functioning of cells, of protein interactions and of dynamical relationships occurring in the small units of all living systems.

INTRODUCTION
The vast amount of knowledge accumulated on the structure of cells, the shapes and movements of its constituents, the interaction among participants and with the environment is at present in a form which is accessible only to experts of the fields. Moreover, this information is often difficult to interpret in terms of dynamic deployment of single events.
Our aim is to use available biological information to describe the inside working of a cell in 3D animated representation. To reach this aim, we are using Maya/Autodesk, one of the most powerful software developed by the industry of 3D animation and special effects (1). With data imported directly from the Protein Data Bank (PDB), we animate protein movements in virtual space, according to information and rules derived from physics, chemistry, biochemistry and other scientific sources.

Using our Maya script, atomic coordinates of proteins and other molecules are imported, together with chemical structures. If more than one conformation is present for a molecule, then these are imported and the program is run to interpolate intermediate position that transit the protein from one conformation to another.

The first examples we present are two small molecules (Triazine, and Bitucarpin) for which theoretical dynamic studies already revealed the energy landscape, which is used for validation of our system.
Most biomolecules however, and notably proteins, contain very large number of atoms, requiring different, more complex programs that can accomodate the large information content of such molecules.
We will present results obtained with our system and offer demonstration of how it can be applied to peptides (we used the V3 peptide of HIV-1 gp120) and the entire gp120 protein.

From the point of view of animation programs, animating means inferring intermediate steps between a start and an end position, i.e. moving the virtual object in space along time. The kind of movement can be governed by sets of rules defined as mathematical expressions, many of which are already present in Maya; other can be codified.
It is important to notice that all what our system does is to interpolate positions while avoiding prohibited ones, therefore the calculations are extremely rapid. In this respect, it is completely different from the most widely used Molecular Dynamics programs, which compute positions according to very complex energy calculations. On the other hand, it is possible that some interpolated movements are patently wrong, which introduce the need for human revision of every animation.

Another important aim of our project is the delivery physico-chemical information of molecules and of the environment such as pH, electronegativity, hydrophilicity and others that are of importance for the way biomolecules behave. This process, in Computer Graphics, is called Rendering, and we will also show some of the progress in this respect.

Small molecules

Animation of molecular structures implies that information relative to the identity of the atoms, their positions, their reciprocal relations, are first imported in the animation system. After this, different positions can be assigned to every single atom at different time-points and interpolation of atom's positions between time points can be calculated. To this aim, we have first used a few small molecules for which the energy landscape of different positions has been generated through Molecular Dynamics studies: Triazine (2), Bitucarpin (3) and Di-Ala dipeptide (4).

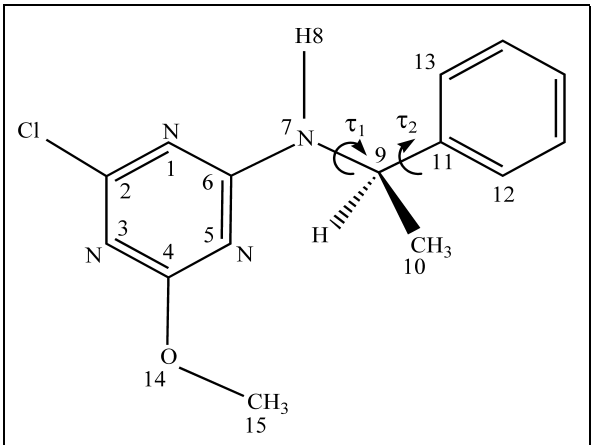


Fig. 1.

Triazine (2-chloro-4-methoxy-6-[(R)-1-phenyl-ethylamino]-1,3,5-triazine) is a small molecule composed of 31 atoms, with a relatively simple structure of two rigid disks connected by a C-N bridge (see Fig.1). The different conformational positions that Triazine can assume are basically variations of τ_1 and τ_2, i.e. rotations around the two chemical links that connect N7, C9 and C11.
Dynamical simulation studies by Alagona et al (2), have revealed the energy landscape for all possible conformations that Triazine can assume. For this reason we chose it as the initial test molecule of our bio-chemical Maya system.

The program we developed to import chemical data into Maya assigns every atom to a position. Atoms are linked through bones (see panel A in Fig. 2), which behave like chemical bonds, have fixed length, and are constrained by codified rules.
Four different conformers, three minimal energy positions and one intermediate (A-B-C and H, see Fig. 3) were imported, and assigned to time points in the animation (key-framed).
Coordinates for all atoms in some intermediate positions calculated by Maya along two possible pathways between C and H were retrieved and fed back into pdb-like files. Angles τ_1 and τ_2 were calculated and plotted entered into the energy map, allowing for physico-chemical evaluation of the path calculated by Maya.

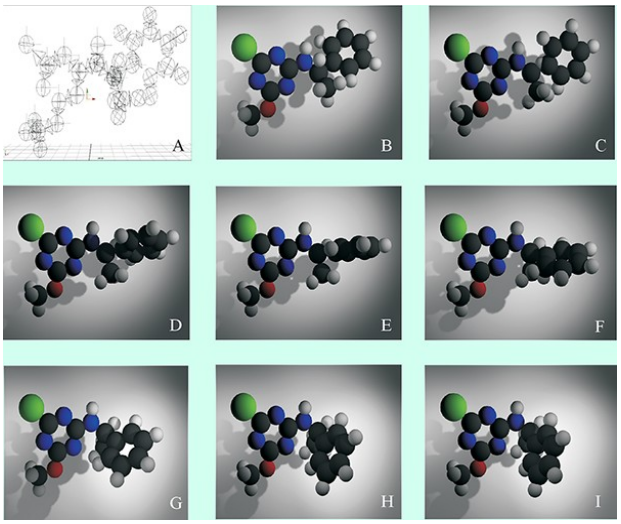


Fig. 2.

Fig. 3 reports the energy landscape, from ref. 2, with the paths calculated by Maya for transition between position C and position H, following the two possible trajectories.
Note that the path labelled with black dots, which includes an almost 180º rotation of the phenyl ring, spans the energy field more than once. This is because the energy is calculated on a chemical basis, where Carbons 12 and 13 are equivalent; however, in a topological view, each of them has its own identity, and the landscape is in fact twice as large.
These results show that Maya can calculate paths avoiding the energy peaks, i.e. describing

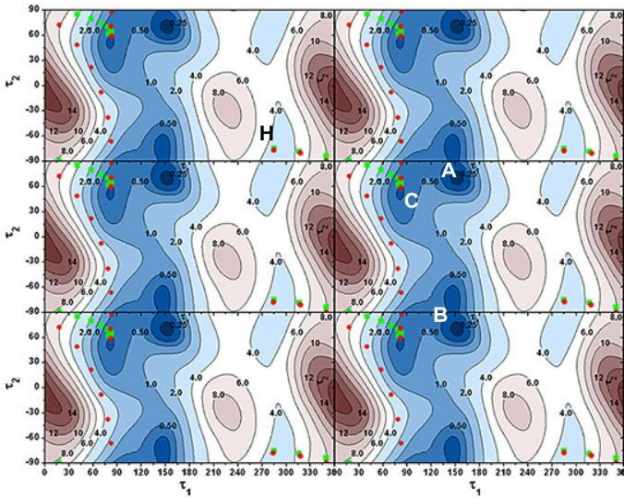


Fig. 3

a movement that is chemically acceptable, flowing naturally in the 'valleys' of the landscape.

Results for Bitucarpin, (a plant chemical for which the energy map has also been calculated) and for the Di-Ala dipeptide will be presented as demonstrations in real time at the meeting.

Peptides and proteins

Proteins can contain up to several thousands or tens of thousands atoms. For some of them, crystal structures have been determined in different conformations, allowing us to set two key-framed positions and to elaborate possible interpolations to transit from one conformer to the other. For other small proteins or peptides, NMR studies provide variable numbers of conformations that the peptide can assume in solution: we have taken advantage of this information to script an animation program that runs in Maya.

Fig. 4 shows the interface of our program. The user can upload the .pdb file to be used as source and set a number of features, including the kind of source (X-ray or NMR), the atoms to be represented (including or excluding hydrogens), the timing of animation and the atoms to be considered for the animation.

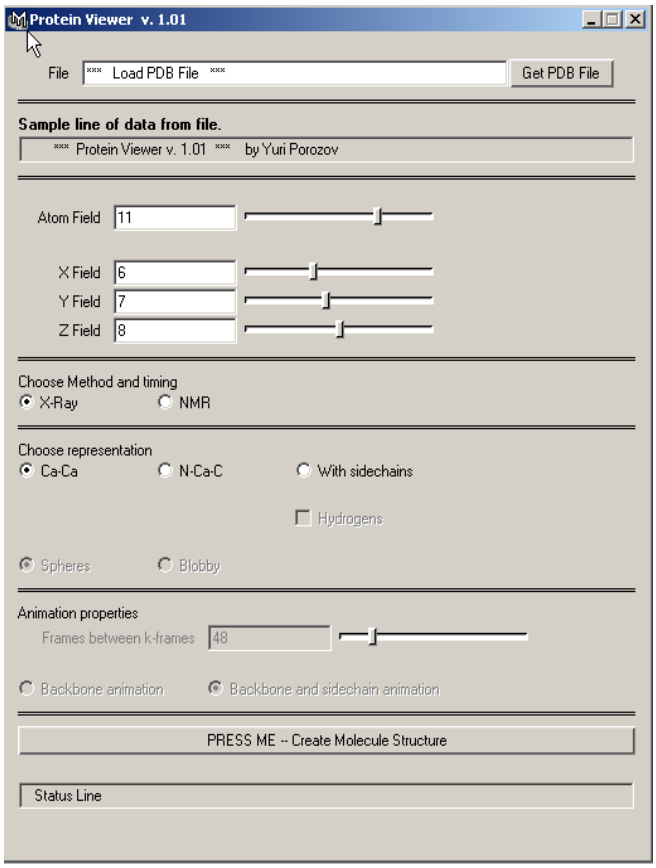


Fig. 4.

For proteins we have used the Particle feature to create them in the 3D space of Maya. Particles are 'light objects' in terms of processing power, and can be dealt with either as a single object that includes them all, or on a *per particle* basis. In other words, large movements (such as bends on a hinge) and relative movements (of the object in space) can be imposed and calculated very fast. The *per particle* attributes are used for rendering (where each kind of atom displays different), and for imaging detailed movements.

To test the system for NMR, we have used the 20 conformers of V3 (5), PDB entry 1CE4. All sructures, after being ordered using a statistical apporach, were imported in to Maya. Each conformation was assigned to a different time-point and animated. The resulting animations will be shown during the demonstration, and can be seen on our website www.scivis.ifc.cnr.it.

Gp120 structure has been solved in different conformations: either unbound (6) or bound (7) to CD4. Its interaction with the cellular receptor CD4 triggers a major movement of parts of the protein, in particular the V3 loop. V3 is implicated in the selection of co-receptor and in the subsequent step of co-receptor binding.

Rendering

Visual perception of the world is a very complex process that we perform automatically. When producing totally artificial images, to obtain the impression of realism, we have to introduce a large set of effects, such as light sources, casting shadows consistent with the illumination, assigning optical properties to materials, fixing the 'eye settings', i.e. the (virtual) camera properties and so on.
To assign texture to objects is a complicated task even for reproducing properties of 'real' objects, when all we have to do is to copy the 'visual feel'. The rendering process (i.e. assignment of visual properties to surfaces) in CG programs involves the setting of material (2d and 3d textures) color,

reflection, luminosity, lights, ambient light and camera movements. Proteins and other biological molecules are, in essence, chemical compounds with specific properties that are determined by the nature of their atoms and the way they are connected and organized in the 3D space.
These properties are defined, in physico-chemical terms, as potentials, typically expressed with complex equations and/or numerical values. One of the aims of our effort, is to convey the significance of these properties in a visual way. Chemical programs can calculate, for example, the electrostatic potential of a surface, or its hydrophobicity, and report it on the surface using a conventional code, typically a colour scale.
We report in Fig. 5 some images obtained while studying different ways to render applied to a form created with a random process or to a shape representing a branched complex sugar typically found on glycoproteins. Images were obtained using the RenderManForMaya plug-in from Pixar.

Fig. 5.

CONCLUSIONS

The initial work presented here is part of a large project that will bring to virtual (and visible) life the processes that occur in the real (but invisible) world of cells.
Because the understanding obtained through sight is much more direct than through word description or intellectual (mental) representation, this will permit researchers to get a more direct grasp of the phenomena under study. Providing a different vision, it should also enable the formulation of new questions, or an alternative way to formulate old, still unanswered ones.
Furthermore, the availability of a virtual cell might allow testing new hypothesis in the virtual cell before performing real experiments.

Also, a direct representation will greatly facilitate the teaching of cellular and molecular biology, at various levels, from secondary school to higher university, and it will also be available for museums, thus attracting new students to the fascinating field of biology.

REFERENCES

1. www.alias.com
2. Alagona, G., Ghio, C., Monti, S. (2006). A Test Case for Time-Dependent Density Functional Theory Calculations of Electronic Circular Dichroism: 2-Chloro-4-Methoxy-6-[(R)-1-Phenylethylamino]-1,3,5-Triazine Theor. Chem. Acc. (in press). Published online: Jan. 5, 2007.
3. Alagona, G., Ghio, C. and Monti, S. B3LYP/6-31G* conformational landscape in vacuo of some pterocarpan stereoisomers with biological activity. Phys. Chem. Chem. Phys., 2004, 6, 2849-2857.
4. Wang, Z. X. and Y. Duan (2004). "Solvation effects on alanine dipeptide: A MP2/cc-pVTZ//MP2/6-31G** study of (Phi, Psi) energy maps and conformers in the gas phase, ether, and water." J Comput Chem **25**(14): 1699-716.
5. Vranken, W. F., F. Fant, et al. (2001). *Conformational model for the consensus V3 loop of the envelope protein gp120 of HIV-1 in a 20% trifluoroethanol/water solution.* Eur J Biochem **268**(9): 2620-8.
6. Chen, B., E. M. Vogan, et al. (2005). *Structure of an unliganded simian immunodeficiency virus gp120 core.* Nature 433(7028): 834-41.
7. Kwong, P. D., R. Wyatt, Robinson, J., Sweet, R. W., Sodroski, J., Hendrickson, W. A. (1998). *Structure of an HIV gp120 envelope glycoprotein in complex with the CD4 receptor and a neutralizing human antibody.* Nature **393**(6686): 648-59.

Visualization Methods for Molecular Studies on the Web Platform

Marco Callieri*
Visual Computing Lab
ISTI-CNR

Raluca Mihaela Andrei†
Scuola Normale Superiore, Pisa
Scientific Visualization Unit
IFC-CNR

Marco Di Benedetto‡
Visual Computing Lab
ISTI-CNR

Monica Zoppè§
Scientific Visualization Unit
IFC-CNR

Roberto Scopigno¶
Visual Computing Lab
ISTI-CNR

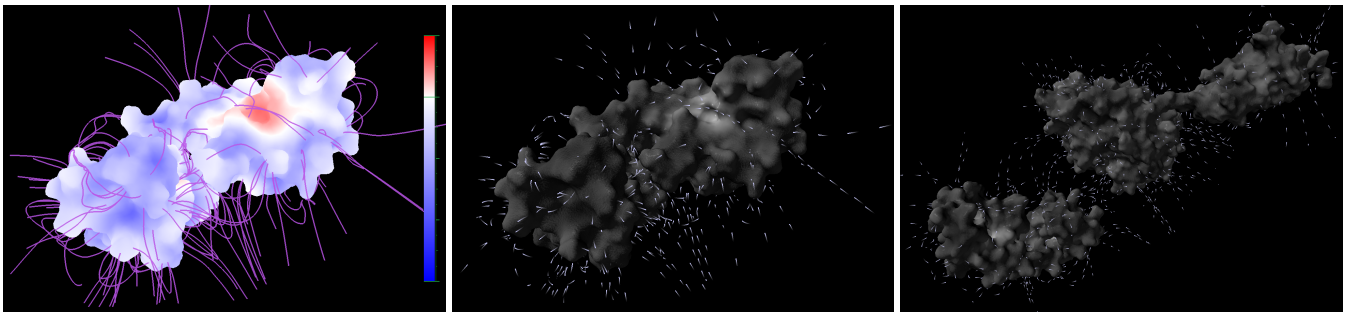


Figure 1: *Standard representation of molecular surface properties using color ramps and field lines (leftmost), the same properties drawn using complex shading techniques (center) and the electrical interaction of two proteins (rightmost), rendered on a Web Page by using SpiderGL and WebGL.*

Abstract

This work presents a technical solution for the creation of visualization schemes for biological data on the web platform. The proposed technology tries to overcome the standard approach of molecular/biochemical visualization tools, which generally provide a fixed set of visualization methods. This goal is reached by exploiting the capabilities of the WebGL API and the high level objects of the SpiderGL library, these features will give the users the possibility to implement an arbitrary visualization scheme, while keeping simple the implementation process. To better explain the philosophy and capabilities of this technology, we will describe the implementation of the web version of a specific visualization method, demonstrating how it can deal with both the requirements of scientific rigor in manipulating the data and the necessity to produce flexible and appealing rendering styles.

CR Categories: I.3.2 [Graphics Systems]: Distributed/network graphics—; J.3 [Life and Medical Sciences]: Biology and genetics —;

Keywords: Web platform,Molecular Biology, Molecular Surface Visualization, Protein Structure, physico-chemical properties, Interactive 3D, WebGL

*e-mail: callieri@isti.cnr.it
†e-mail: r.andrei@sns.it
‡e-mail: dibenedetto@isti.cnr.it
§e-mail: mzoppe@ifc.cnr.it
¶e-mail: scopigno@isti.cnr.it

1 Introduction

Interactive visualization of molecular structures and physico-chemical data is an important and interesting research field which span from the Computer Graphics world to the Biological and Molecular studies. The amount of complex structures that is available through public repositories and the level of detail of biochemical datasets which can be manipulated by physico-chemical tools has greatly increased in the last years, making it essential to employ dedicated visualization techniques to make an effective use of these data. While it may be easy to draw even large molecular datasets as a series of atoms (Van der Waals spheres) using simple rendering methods based on impostors and other tricks, the precise rendering of a high resolution molecular surface involves the management of more complex geometry. Furthermore, when it is necessary to represent interaction between different molecules or to introduce the rendering of further 3D elements and data layers (as in the examples of Figure 1), the required computational and rendering capabilities do increase significantly.

A previous work in the field of visualization of molecular structures, QuteMol [Tarini et al. 2006], has shown that, by using advanced shading techniques, it is much easier to convey the information regarding the geometry and structure of the molecule. We believe the same reasoning may also be applied to the visualization of other physico-chemical data: by using custom shading and rendering techniques the same improvement in clarity and expressiveness can be attained. This is however quite difficult at the moment,

Web3D 2010, Los Angeles, California, July 24 – 25, 2010.
© 2010 ACM 978-1-4503-0209-8/10/0007 $10.00

since it is rarely possible to finely control the rendering pipeline and shading process inside existing visualization tools, especially when working on on-line platforms. Here are some possible scenarios we are considering:

- a research group interested in proposing a new visualization method may want to publish a web page which shows an interactive version of such visualization;
- a public repository of biological data structures may want to let its users view the available data using a custom shading, designed to effectively show the characteristics of the provided information;
- an educational-oriented entity may want to present biological phenomena to a large public, possibly including a non-specialist audience, by using visualization method that are not just scientifically accurate, but also visually pleasant.

These three scenarios exemplify the need for advanced visualization methods, but also for the use of the web platform. Pervasive and easy to access, the web platform is becoming more and more important for sharing data, processing methods and visualization techniques. It is easy to foresee that the research community working on visualization for cellular and molecular biology will find in the web platform the ideal media for the purposes of research, education and science divulgation.

Up to now, the capabilities of web browsers to efficiently manipulate and display 3D content have been very limited. The task of putting online three-dimensional data has exploited the use of commercial or custom browsers plugins, and has been characterized by a series of problems, like low portability (each plugin/extension would work only on a subset of browsers and operating systems), scarce flexibility (in most cases the visualization plugins offered no way to configure the drawing pipeline or add special rendering modes) and poor performance (the different software layers of network, O.S., sandbox and plugin introduced lag and computational overhead, separating too much the rendering from the hardware layer). A suitable solution for this problem may reside in the upcoming standard of WebGL [Group 2009b], which is an API specification which defines the web-oriented analogous of the OpenGL API. The most interesting feature of this API is that it is implemented directly inside the browser, with direct control over the graphics hardware. This will help overcoming the compatibility problems and result in a much more efficient and performing platform. Deriving from the specifications of OpenGL-ES, WebGL provides a completely customizable rendering pipeline and the entire shading process is controlled through hardware-level GLSL shaders. This shader-based nature of WebGL is perfectly suited to cope with the need of creating a custom visualization scheme. Direct access to the hardware layer means not only better performances, but also the possibility to exploit the full repertoire of techniques and experience accumulated in years of computer graphics research.

Of course, WebGL alone is not enough to answer the needs of people interested in biochemical visualization (which are likely not experts in graphical programming); following the design philosophy of OpenGL, WebGL is a very low-level API which requires a good knowledge of computer graphics techniques and coding skills. It is therefore necessary, to ease the use of this technology, to introduce a library able to wrap the most low level function, while giving the user the ability to dive into implementation details, when needed. As the WebGL standard is taking shape, different wrapping libraries are appearing on the web [DeLillo 2009; Brunt 2010; Kay 2009]; one of these libraries, SpiderGL [Di Benedetto 2010], seems to provide the right balance between the ease of use of the higher level functions and the possibility to fully control the rendering pipeline.

We believe that the use of WebGL through the SpiderGL library will prove to be a very powerful platform to implement visualization methods on the web for the molecular biology and physico-chemical research community.

We review in Section 2 some of the previous work in the field of both molecular visualization and on-line publishing of 3D content. Then, in Section 3, we introduce the basic ideas of the proposed technology describing its main philosophy and by presenting the core library used. Finally, as an example of the use of this technology, we show in Section 4 how a specific visualization method has been adapted to the web platform.

2 Previous Work

2.1 Molecular visualization Off-Line and On-Line

The solution of the 3D structure of myoglobin in 1958 by Kendrew [Kendrew et al. 1958] marked the beginning of the new era of protein structural biology. Since then, a large number of protein structures have been solved and today the Protein Data Bank counts over 60.000 entries [Berman et al. 2003]. With the availability of all these data and the advance of computer graphics technologies, many research groups have developed tools for the manipulation and visualization of 3D structures such as VMD [Humphrey et al. 1996], SPDBViewer [Guex and Peitsch 1997], Chimera [Pettersen et al. 2004] and PyMOL [Delano 2002]. Beside working on the atomic structure, most programs can nowadays also calculate surface features such as electrostatic potential (using, for example, tools like APBS [Baker et al. 2001] or DelPhi [Rocchia et al. 2002]) and hydropathy [Kyte and Doolittle 1982].

In addition to the many standalone visualization tools, there are also web viewers especially designed for molecular structures, such as Jmol [jmo 2002] and MDL Chime, which represent a simple way to visualize molecules directly on browser. MDL Chime, used by the Protein Explorer website was gradually phased out in favor of Jmol, which is nowadays the most used plugin for molecular visualization, used by websites such as Proteopedia and RCSB PDB Protein Data Bank.

Following the advance of techniques for the generation of CG movies, in the last few years many different groups focused on the creation of animated movies depicting biological molecules and cellular processes. The movies range from the simple representations of the mechanical functioning of a single protein, to complex events involving many subjects. These works are important scientific efforts and add to their educational value the bonus of rising interest in the general public to approach biology. Some of these examples are collected on websites [McGill 2010; SCIVIS 2005].

2.2 3D Content on Web

The web platform has acquired through the years the ability to efficiently incorporate and deliver many different kinds of digital data such as still images, videos and sound. With respect to these additions, the management of 3D content through the web comes with a considerable delay. The reasons for this delay are likely to be found in the higher requirements of 3D graphics in terms of computational power, but also because the lack of a strong unifying standard behind the 3D content.

Several technologies have been developed over the years to achieve this integration. The Virtual Markup Modeling Language (VRML) [Raggett 1994] (then replaced by X3D [Don Brutzmann 2007]) was proposed as a text based format for specifying 3D scenes in terms of geometry and material properties and for the

definition of basic user interaction. The format itself was a standard, but the rendering in the web browser was relaying on specific plugins. The Java Applets are probably the most used method to add dynamic content, not necessarily 3D, in the web browsers. The philosophy of Java applets is that the URL to the applet and its data are put in the HTML page and then executed by the Java Virtual Machine, a third part component. The implementation of JVM on all the operating systems made Java applets ubiquitous and the introduction of binding to OpenGL such as JOGL [JOG] added control on the 3D graphics hardware. A similar idea lies behind the ActiveX [Microsoft Corporation 1996] technology, developed by Microsoft from 1996. Unlike Java Applets, ActiveX controls are not bytecode but dynamic linked Windows libraries which share the same memory space as the calling process (i.e. the browser), and so much faster to execute. These technologies enable the incorporation of 3D graphics in a web page but they all do it by handling a special element of the page itself with a third party component.

WebGL [Group 2009b] is an API specification produced by the Khronos group [Group 2009a] and, as the name suggests, defines the JavaScript analogous of the OpenGL API for C++. WebGL closely matches OpenGL|ES 2.0 and, extremely important, uses GLSL as the language for shader programs, which means that the shader core of existent applications can be reused for their JavaScript/WebGL version. Since WebGL is a specification, it is up to the web browsers developer to implement it. At the time of this writing, WebGL is supported in the nightly build versions of the most used web browsers (Firefox, Chrome, Safari), and a number of JavaScript libraries are being developed to provide higher level functionalities to create 3D graphics applications. For example WebGLU [DeLillo 2009], which is the WebGL correspondent of GLU [OpenGL ARB], provides wrappings for placing the camera in the scene or for creating simple geometric primitives, other libraries such as GLGE [Brunt 2010] or SceneJS [Kay 2009] uses WebGL for implementing a scene graph based rendering and animation engines.

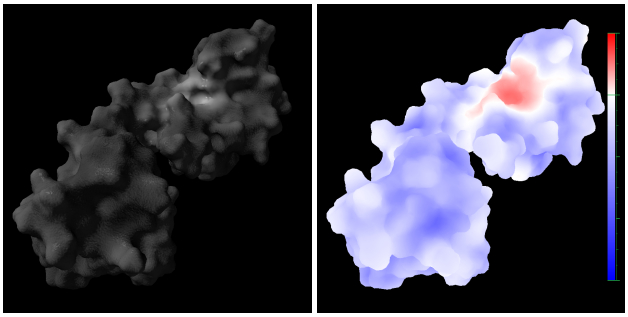


Figure 2: *Lipophilic Potential mapped on the surface of a Calcium-bound Calmodulin. On right, visualization using standard color ramp; on left, visualization using advanced shaders. The light color and the specularity clearly indicates a liphophilic patch on the right part of the molecule, while the dark, dull and rough surface indicates a more hydrophilic area.*

3 Building a custom web-based Visualization Scheme

As stated in the introduction, the aim of this work is to propose a technology for the implementation of advanced visualization schemes for molecular and biochemical data on a web platform. We are interested in a base technology that is able to cope with the needs of a completely customizable rendering while providing enough basic structures and higher level functions to be usable without a major programming effort.

Our idea is that the WebGL standard is able to provide the performances and fine-control over the rendering, since it directly uses the hardware layer for rendering, is built around the idea of a fully customizable rendering pipeline and gives access to the use of GLSL shaders, a really powerful instrument to achieve the desired visual output. While these features are absolutely necessary to reach our goal, they are not sufficient to provide a really usable development platform because the available functions are too low level to be effectively used (especially by a community of people with little or no experience in CG programming). By introducing a wrapping library as SpiderGL, it is possible to enrich this platform with a series of higher level functions that may be used as building blocks to implement the desired rendering method. As a final ingredient, we have also to consider what can be attained by a clever use of the JavaScript. Exploiting the ease of use and the expressive power of this language it is possible to read the source scientific data and do all the needed calculation.

3.1 SpiderGL

The core library used in this work is SpiderGL: a recent, ongoing project which aims to provide an easily usable but powerful wrapping to the lower-level WebGL functions. Most of the available JavaScript graphics libraries and browser plugins for 3D data management are based on the paradigm of *scene graph*. This choice is perfectly natural, in the sense that it mimics the idea of a three-dimensional scene composed of objects, rendered from a given point of view. However, this solution cannot fully answer the need of scientific visualization, where it is often needed to use very diverse data, and render them in a very controlled way. SpiderGL, on the other hand, does not follow this paradigm, but provides a set of data structures and algorithms to support the management of geometric and mathematical entities, in order to simplify the creation of arbitrary visualization prototypes. The idea of this library is to provide a complete wrapping layer to WebGL that, while hiding the details through higher level functions, allows full access to the native API.

To ease the creation of graphical applications, SpiderGL provides a series of classes and functions which cover the various aspects and levels of implementation of a CG program:
Basic structures: linear algebra algorithms for 3D points and vectors are very common tools for the CG developer; the geometry module of SpiderGL implements the essential mathematical objects such as vectors (2,3 and 4 components), quaternions and matrices, along with basic operations on them.
2D/3D Data: one of the fundamental parts of a graphics library is the management of data structures for the definition of 3D objects (meshes), textures and the other components used in the rendering process. While at low level, WebGL works directly on streams of vertex attributes and indices, SpiderGL, to provide a more structured object to manage, implements a mesh object, based on the usual paradigm of vertices+triangle connectivity. For a flexible but efficient use, SpiderGL supplies two different data structures: the first one, `SglMeshJS`, can be freely accessed and modified within the user script; the other, `SglMeshGL`, is generated from the first and used at GPU level for rendering. Management of textures is done through some specific functions which enable the creation of texture from images or raw data, texture sampler options and texture unit binding. A final set of classes is used to manage vertex shaders, fragment shaders and shader programs, with support from compilation feedback, binding and attribute management.
Scene management: while not introducing a scene-graph, SpiderGL provides some specific helpers to place entities in the

3D space, set the viewpoint and simplify the user interaction with the 3D elements. The matrix stack, legacy of the OpenGL library, is still extremely useful when populating a scene and it is implemented in the `SglTransformStack` object, which offers many different methods to manipulate and access these matrices. Other helpful classes include a camera object which implements the typical paradigm used in first-person shooter games (`SglFirstPersonCamera`) and a trackball manipulator (`SglTrackball`) for object inspection with pan, zoom, rotation and scaling operations.

Rendering: WebGL redering is a long series of function used to manage all the data streams, bind streams to attributes, select shaders and control the GL status. In SpiderGL, 3D mesh rendering is managed through the `SglMeshGLRenderer` helper class. This class takes care of all the setup steps required by WebGL and tries to simplify the stream mapping process by automatically match all the most used attributes. Additional helper classes give finer control over the mapping of data streams and deal with the management of shader parameters.

Application: interactivity is one of the focus of this library; for this reason, SpiderGL provides an event-based mechanism which is able to collect events from all the DOM and efficiently dispatch them to multiple listeners. Other application-level support structures like a log system are available through specific classes. Another extremely useful feature on the web platform is the asynchronous loading: many rendering algorithms requires the ability of asynchronous loading of data. Even if JavaScript does still not support multithreading, SpiderGL implements a simple mechanism based on the `XMLHttpRequest` object to queue data to be loaded and set a callback functions which will be invoked whenever the transfer of the requested data has completed.

3.2 Data Importing and Management

While using JavaScript it is not possible to read binary data, this may not be a major problem for the need of importing data from molecular databases or physico-chemical tools. Many biological-related file formats use ASCII coding, which make the parsing really straightforward. As an example, the importer for the PDB file format, which describe the structure of a molecule, is just a few lines long. Here it is possible to see part of the importing code and how the predefined JavaScript functions for tokenization help its parsing:

```
function AtomListFromPDB(atomlist, pdb_txt){
[.....]
var lines = pdb_txt.split("\n");
for (var lineIndex in lines) {
//atom line example
//ATOM 16 O ASP A 2 10.6 5.1 -6.1 0.0 0.0 O

   tokens = line.split(" ");
   if (tokens[0] == "ATOM") {  // atom line
      var atomtype = tokens[11];

      var position = [];
      position.push(parseFloat(tokens[6]));
      position.push(parseFloat(tokens[7]));
      position.push(parseFloat(tokens[8]));
[.....]
```

More recent biochemical tools may also export data in XML format, which is directly readable by JavaScript. Three-dimensional geometries are normally stored using one of the many standard file formats, and can generally be converted from one format to another; SpiderGL does at the moment support OBJ and COLLADA formats (other importers will follow), and different other formats may be parsed by using JavaScript. Less structured data may be imported also by making the data source export in the JSON format, which is quite easy to write and is natively supported by JavaScript interpreters. Since most physico-chemical tools have a scripting layer which can be used to specify custom data exporters, this is often a viable option.

Most of the more interesting visualization methods, however are not just based on loading existing data and displaying it in a controlled fashion, but also relies on some kind of data processing. JavaScript may be an effective ally also in this case, thanks to its ease of use, the great flexibility in data structure (dynamic typing, associative arrays, an advanced garbage collector), the presence of many built-in functions and its expressive power. And if it is true that probably JavaScript will never reach the computational efficiency of compiled C++ code, the newest interpreters and the introduction of just-in-time compilers have significantly reduced the gap. It is possible to say that, in this specific scenario, where most of the computational requirements have been moved from CPU to GPU, the difference between the two language is neglectable.

It is also interesting that, since all the computation is done at the JavaScript level and all the visualization code is embedded in the page, it is not possible to effectively hide the data or their processing. This impossibility of building a closed, protected system may be perceived as a serious limitation for industrial-related applications. However, in these context, this same limitation may turn out to be a very positive feature, since the transparency of the data processing (you may check that no hidden tweaking is done on the data) and the possibility of sharing knowledge (by letting others reuse your visualization code) are of capital importance in the fields of research and educational tools.

3.3 Implementation

Having all the necessary building blocks to load, manipulate and render the data, it is possible to build the desired visualization method. The setup of the scene and the definition of the rendering pipeline work similarly to a standard visualization application.

Looking at a webpage with dynamic SpiderGL content, it is possible to see that all of the page logic is defined in the scripting part of the HEAD section, while on the BODY section there is just the page structure and the interface elements that will be used for user interaction (like buttons, text areas and other controls). Among these elements, the most important is an html `canvas` object, that is the place where the WebGL layer does the on-screen rendering.

```
<canvas id="SGL_CANVAS" style="border: 1px solid gray" ↩
     width="900" height="600"></canvas>
```

This canvas is registered as the output area at the end of the scripting; a specific function connects the various events of the canvas to a script object.

```
var glMolViewer = new SpiderGLMolViewer();
sglRegisterCanvas("SGL_CANVAS", glMolViewer, 30.0);
```

The `glMolViewer` object is the main actor for the scene setup and rendering of our molecular visualization. The structure of this object employs the event handling subsystem provided by SpiderGL, which is inspired from the one used by the GLUT library [Kilgard]. Each event coming from the canvas triggers a specific function with a given name and parameters; SpiderGL exploits the JavaScript language feature to give the possibility to dynamically add or remove listeners and redirect events. In this simple example, the only listener is the main object itself.

```
SpiderGLMolViewer.prototype = {
```

```
  load : function(gl) { [...] },
  unload : function(gl) { [...] },

  update : function(gl, dt) { [...] },

  keyDown : function(gl, keyCode, keyString) {[...]},
  keyUp : function(gl, keyCode, keyString) {[...]},
  keyPress : function(gl, keyCode, keyString){[...]},
  mouseDown : function(gl, button, x, y) { [...] },
  mouseUp : function(gl, button, x, y) { [...] },
  mouseMove : function(gl, x, y) { [...] },
  mouseWheel: function(gl, wheelDelta, x, y) { [...] },
  click : function(gl, button, x, y) { [...] },
  dblClick : function(gl, button, x, y) { [...] },

  resize : function(gl, width, height) { [...] },

  draw : function(gl) { [...] },
};
```

Most of the initialization and data loading is done in the `load` function: it is here that the main properties of the rendering are chosen, the input data is loaded and the shaders are compiled.

```
load : function(gl) {
[...]
  this.xform = new SglTransformStack();
  this.camera  = new SglFirstPersonCamera();
  this.camera.lookAt(0.0, 0.0, 1.5, 0.0, 0.0, 0.0, ↩
      sglDegToRad(0.0));
  this.viewMatrix = this.camera.matrix;
  this.trackball  = new SglTrackball();
[...]
  this.prog = new SglProgram(gl, [sglNodeText("↩
      MY_VERTEX_SHADER")],[sglNodeText("↩
      MY_FRAGMENT_SHADER")]);
[...]
  var TextureOptions = {
  generateMipmap : true,
  minFilter : gl.LINEAR_MIPMAP_LINEAR,
  onload : this.ui.requestDraw
  };
  var ColorTexture = new SglTexture2D(gl, "↩
      molecule_color.png", textureOptions);
[...]
  this.meshJS = new SglMeshJS();
  this.meshJS.importOBJ("molecule.obj", true, function(↩
      m, url) { [...]
    this.meshGL_MOL = that.meshJS.toPackedMeshGL(gl, "↩
        triangles", 65000);
  [...] });
[...]
  var pdbtxt = sglLoadFile("mol.pdb");
  this.atomslist = [];
  var res = AtomListFromPDB(this.atomslist, pdbtxt);
[...]
  this.timeOffet = 0.0;  // particle animation offset
  this.stereoEnabled = false; // start with no stereo
  this.particlesEnabled = true; // start with particles
},
```

This monolithic way of managing data is fine for webpages devoted to the visualization of a single, compact dataset. More advanced examples may benefit from the asynchronous loading mechanism which allows efficient use of large datasets, streaming/progressive data or letting user dynamically load remote files.

The `draw` function contains the code for the actual rendering:

```
draw : function(gl) {
  gl.clearColor(0.0, 0.0, 0.0, 1.0);
  gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT | ↩
      gl.STENCIL_BUFFER_BIT);
  gl.viewport(0, 0, w, h);
  this.xform.projection.loadIdentity();
  this.xform.projection.perspective(sglDegToRad(45.0), ↩
      w/h, 0.1, 10.0);
[...]
  var uniforms = {
    u_mvp : this.xform.modelViewProjectionMatrix,
    u_normal_mat : this.xform.viewSpaceNormalMatrix,
    u_dotrasp : this.atomsEnabled,
    u_mousepos : [this.ui.mousePos.x, this.ui.mousePos.↩
        y]
  };
  var samplers = {
    s_texture_c : this.ColorTexture,
    s_texture_b : this.BumpTexture
  };
[...]
  gl.enable(gl.DEPTH_TEST);
  gl.enable(gl.CULL_FACE);
  sglRenderMeshGLPrimitives(this.meshGL_MOL, "triangles↩
      ", this.prog, null, uniforms, samplers);
[...]
},
```

This function may be called *continuously* or *on demand*: when registering the canvas with the `sglRegisterCanvas` function, if the last parameter is 0, then the canvas is only redrawn by explicit commands, otherwise, the parameters represent the desired frame rate. At each "tick" the SpiderGL will call the `update` function and then the `draw`. In both cases, the html rendering engine will then issue a page composition operation whenever it detects changes to the associated WebGL framebuffer.

GLSL shaders are included in the web page as script entities in the HEAD section:

```
<script id="MY_VERTEXSHADER" type="x-shader/x-vertex">
[...]
</script>
```

The resulting code is very schematic and organized in such a way that following the various setup and rendering steps is quite easy. This simple example is an optimal starting point for experimentation.

This development process is straightforward for someone with an experience in graphical programming, while may prove to be difficult for users with a different background, like biology, physics or chemistry. This kind of setup is for sure more difficult to master with respect to setup of other existing platforms, like Jmol which, true to their nature, provide much simpler (but restrictive) access to their scene graph, with specific functions to import data and a series of predefined rendering modes. However, the gain in terms of flexibility and expressive power vastly compensate the initial steeper learning curve. Moreover, the learning of this technology is made easier by the possibility of initially use the higher level structures and functions implemented by SpiderGL to easily setup a basic visualization scheme and then start playing with lower level functions to obtain more complex effects. It is also important to note that most of the available JavaScript utility/UI libraries on the net may be used in conjunction with SpiderGL, adding more ready-made components to assemble a powerful, interactive, webpage.

4 Visual Mapping of Molecular Properties

As an example of the strategy described in the previous section, we will describe how a specific visualization scheme may be implemented using the proposed technology in a very straightforward way.

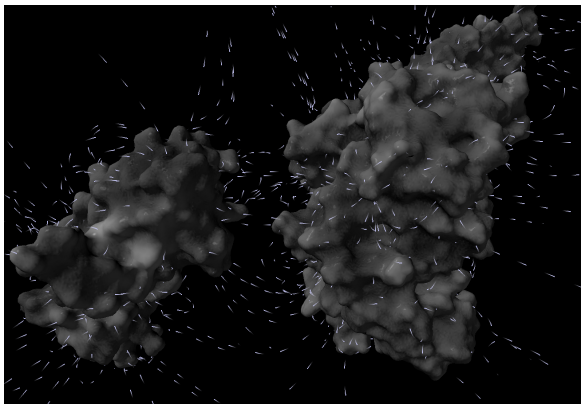


Figure 3: *Interaction between two molecules: the particle flow shows the electrical attraction between the Calmodulin and the MLCK head.*

The aim of this visualization method [Andrei et al. 2010], designed in the framework of the creation of a CG short movie, was to display two specific biochemical properties on the surface of molecules. The two surface properties were the Molecular Lipophilic Potential (MLP) and the Electrostatic Potential (EP). The ability of a molecular surface to establish bonds with water is called Hydrophilicity; its opposite, which is the ability to establish bonds with fat, is called Lipophilicity. The Electrostatic Potential is easier to understand: each atom in a molecule may have a charge, the various charges in the molecule produce an electric field in the surrounding of the molecule. The main idea of this visual mapping has been to exploit perceptual associations between the values to be mapped and visual characterization of real-world objects. Ideally, by using already established perceptual association, the viewer would be able to understand the provided information more naturally, without the use of explicit legends.

For the mapping of the MLP property, it was necessary to choose two opposite surface characterizations, able to convey a sense of affinity to water or to oil. In our real-world experience, a very smooth, hard surface (like porcelain) is completely impervious to water but can be easily coated by oil. The opposite visual feedback is associated to grainy, crumbly, dull surfaces (like clay bricks or biscuits) which can be easily imagined being soaked in water. These considerations led to the association of highly lipophilic areas as white, shiny, smooth material and of highly hydrophilic areas as dark, dull and rough. While the MLP value is obviously only observable on the surface itself, electrical phenomena are associated to the idea of an effect projected in the volume surrounding a charged object, and able to affect other objects (like the high school textbook-favorite amber rod attracting paper bits). Field lines are a common way to describe the effect of the electrical field. EP value is therefore represented by showing small particles, moving along the path defined by field lines, visualizing a higher concentration of particles in areas where the electrical fields is stronger.

A peculiar characteristic of this visual mapping is that it only uses shades of gray to represent the two molecular properties; this choice, which seems restrictive at first glance, is however capable to efficiently convey the two layers of information while leaving the utilization of color space for the description of other biochemical information. This visualization method is perfect to show the capabilities of the proposed strategy, since it involves data coming from BIO tools and rely on a controlled use of shading (bump mapping and specular map for MLP) and rendering effects like particles (moving along the field lines for EP). Moreover, the focus of this visual mapping is not only towards the scientific accuracy, but also towards the visual appeal of the representation.

4.1 From Scientific Data to Rendering

This visual mapping has been designed with the explicit purpose of being used in a CG movie [SCIVIS 2005], produced using the 3D modeling and rendering tool Blender. For this reason, most of the input data, coming from scientific tools, have been heavily processed in order to be converted in a format easily used inside Blender.

The geometry of the molecular surfaces of the depicted proteins has been generated using PyMOL starting from their atomic structure contained in their PDB files. The two properties have been calculated by using scientific tools, starting from atomic structure and reference tables for atomic electrical and lipophilic contributions. The lipophilic potential data, calculated using a dedicated python script (pyMLP) developed by a molecular scientist, is stored as a series of samples in the proximity of the molecule. These samples are then mapped on the molecular 3D surface using interpolation. This mapped value are used to generate color, specular and roughness texture map. The Electrostatic potential, calculated inside another physico-chemical tool (APBS), is basically a volumetric dataset which describes the electrostatic value computed in a regular grid surrounding the molecule. Using this data, it was easy to compute the potential gradient and use it to generate the field lines. The obtained lines were used in the movie rendering to animate a particle system.

In our example, we will start from the processed data, which is somehow in between biochemical data and standard computer graphics data, and then consider the kind of problem and possibilities introduced by the direct use of scientific data. In this conversion from the CG movie to the realtime web environment, it was possible not only to obtain the same look and feel of the rendered movie, as visible in Figure 4, but also to introduce additional elements which are only possible in an interactive context. Beside the usual interactivity which may be attained by the use of simple widgets like a trackball, the ability to configure the rendering pipeline make it possible to change rendering parameters on the fly, mix multiple rendering styles to visualize multiple data layers at the same time and add effects like the direct rendering in anaglyph-stereo. Again, the important point, more than the mentioned effects, is the possibility of overcoming the limits of the predefined rendering that characterize similar systems.

In the next sections we will detail how each component of the render has been implemented in order to obtain the same look and feel presented in the video. For each section we discuss possible alternatives for data source and rendering methods to show how it is possible to directly use biochemical data or create more complex visualizations.

4.2 Geometry

There are many different methods used in molecular biology to visualize the three-dimensionality of a molecule. There is a clear distinction between the representation of the molecular structure and of its surface. The molecular structure is generally displayed atom by atom (using a *Van der Waals spheres*, *sphere+stick* or *licorice* rendering) or as a series of structure elements (*ribbon, rod+arrow*). Conversely, the molecular surface [Connolly 1983], defined as the set of points which are "accessible" to a given solvent (typically water), is a more complex three-dimensional structure and it is generally displayed as a triangulated mesh, or as a series of nurbs patches.

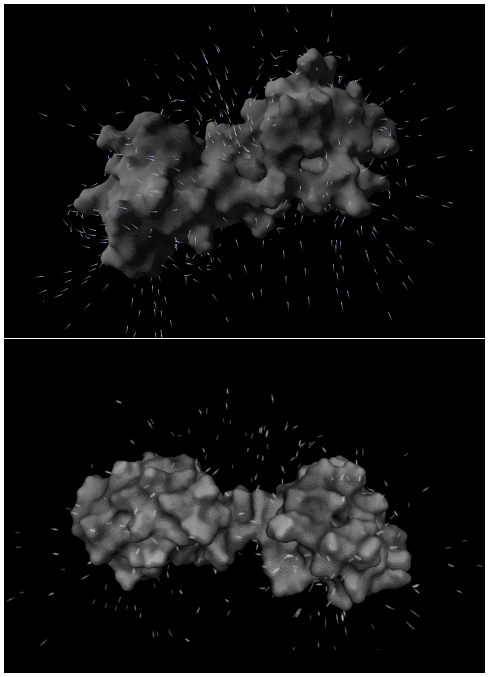


Figure 4: *Comparison of the molecular surface visualization rendered by Blender for the movie (bottom), and rendered using SpiderGL and WebGL (top).*

In this context we are more interested in the rendering of the molecular surface, since the two properties we visualize show their effect in proximity of this surface. Many biochemical tools (like PyMOL, used in this work) are able to compute the geometry of the molecular surface starting from the atomic description of the molecule itself. The result of this process is generally a triangulated mesh, which can be exported, depending on the tool, in different 3D file formats.

For the movie, the used file format was OBJ which is directly readable from SpiderGL: these models were also the starting point for the online visualization. We decided to import precalculated geometries in the scene as they were already available from the pipeline used to create the movie, but also because this is the most sensible option. In theory, it would be possible to compute the molecular surface on the fly starting from the atomic structure of the molecule, but this process would require a non-trivial amount of time and system memory.

As previously stated, to render the structure of the molecule using Van der Waals spheres, it is necessary to know the position, radius and color-coding of each atom. The standard way to represent a molecule structure in biochemical applications is through the use of a PDB file. A PDB file is just an ASCII file which contains (among other molecular-related info) a list of atoms with an associated position. Using JavaScript is quite easy to parse it (as shown in Section 3) and render with SpiderGL a series of colored spheres of appropriate size in the correct position. The atomic representation of the molecule shown in Figure 5 has been generated using this method. A more complex visualization of the molecular structure, like *ribbon*, may also be generated on the fly by starting from the parsed PDB file and a series of pre-defined 3D element templates. As we said before, since there are many molecular databases available online, the PDB file could also be retrieved directly from such a repository.

The availability of alternative representations of a protein structure, makes also possible their combination in a single scene, providing the user the ability to switch between the different representations. Again, this is quite common and nothing new but, since we can configure the rendering pipeline we can, for example, show the superimposition of the molecular surface and the Van der Waals representation by using transparency effects. As shown in Figure 5, it is possible to implement a "fresnel" transparency which depends on the viewing angle, or a more focused "x- ray vision" transparency area which follows the mouse pointer. These kinds of transparency effects are really simple to implement using GLSL shaders and let perceive both representations at the same time, to better understand the relationship between the surface properties and the underlying structure.

4.3 Lipophilic Potential

The visual mapping of lipophilic potential rely on a combination of color, surface roughness and specularity: these three effects are mapped on the molecular surface according to the local lipophilic potential value. For the rendered movie, the potential value has been used to generate the color/specular and bump texture maps inside Blender by baking on the textures a procedural material. To render these effects, we decided to use the same texture maps used in the rendering of the movie and to write a shader which uses simple shading techniques. *Bump Mapping* and *Specular Mapping* are standard shading techniques, but it is possible to apply a fine-control over their appearance by having the full control of the shader setup, which is not generally possible in commercial software for web publishing or in general purpose visualization tools. The result is pleasant and, as visible in the left side of Figure 2, the characterization of the surface is quite effective.

Again, the use of precomputed texture is the fastest way to produce this kind of effect. However, it is also possible to start from the initial data from which those textures have been generated. As in the case of the molecular structure file, the lipophilic information is contained in an ASCII file, which can be parsed using JavaScript and mapped onto the 3D surface as it was done when baking the texture. Once the values are mapped to the surface, a simple shader may be used to produce the same effect of the textures using a procedural approach. Basically, the color of the surface, the intensity of the roughness and the specularity only depends on the lipophilic value: there is nothing which cannot be done in the shader. Having the mapped lipophilic potential makes also possible a more classical rendering style which uses color ramps (right side of Figure 2). This second input method is more generic, since it uses directly the data generated by the biochemical tool, but may be slower (since no precalculation is done and the mapping has to be done at loading time) and less compact (since the lipophilic data may be larger than the textures).

4.4 Electrostatic Potential

Field lines are a widely used method to depict vector fields, especially for electrical and magnetical phenomena. However, the main problem with field lines is *how many* and *which* lines are needed: too few lines do not convey the necessary information and too many will obscure entirely the object of interest. The visualization method used for the movie tried to compute, from the infinite possible field lines, a "meaningful" subset of lines. The aim was to generate a distribution of lines proportional to the surface EP value: more lines would rise in the more electrically active areas, and the total number of lines would be proportional to the global level of potential of the molecule (in absolute value). This operation was done by using a Monte Carlo sampling, weighted with respect to the potential value of the surface in each area. The selected lines were then exported as a sequence of points, forming various poly-

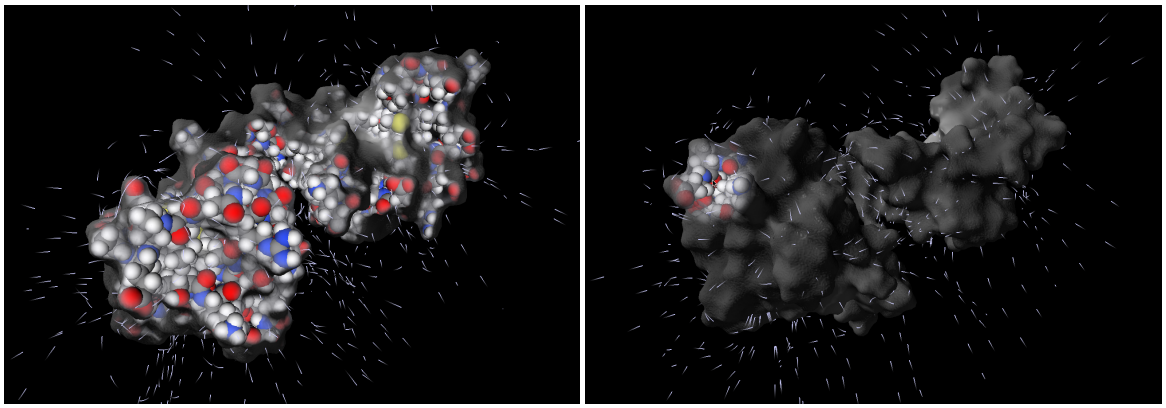


Figure 5: *Showing the superimposition of the Molecular Surface and the underlying Atomic Structure using a transparency based on view angle (left) and a localized transparency area which follows the mouse (right)*

lines. Instead of rendering these entities as solid lines (as visible in Figure 1) each curve was used to drive a particle system. By using moving particles, in fact, it is easier to perceive the flow direction of the field and thanks to their small size and movement, they do not hide the underlying molecular surface. The easiest way to load the lines inside the web implementation was to apply just a small change in the code for line calculation, in order to export the polylines JSON format. It was then possible to parse them using JavaScript. The loaded data may be rendered as a series of solid *line strips*, or used to produce a particle effect similar to the one used in the movie. In this case, since the particles flow on fixed lines, it is not really necessary to create a particle system, but it is possible to visualize the moving particles using a fragment shader which renders only small fragments of the imported polylines according to a periodic function, animated using an offset.

```
uniform float u_timeOffset;
varying float v_texcoord;
void main(void)
{
  const float part_density = 4.0;
  const vec3 part_color = vec3(0.8,0.8,1.0);
  float val = fract((v_texcoord+u_timeOffset)/part_density);
  if (val < 0.7)
    discard;
  else
  {
     val = smoothstep(0.9, 0.7, val);
     gl_FragColor = vec4(part_color * val, val);
  }
}
```

This effect is much more simple and less CPU/GPU demanding than a real particle system, while still effective in conveying the carachteristics of the electrical field surrounding the molecule, the areas of higher electrostatic potential and their polarity. The field particles are also useful to show the electrical interaction between different proteins: in Figure 3 it is shown a calcium-bound Calmodulin approaching an MLCK head, at that distance the two electrostatic field do start an interaction process which will eventually lead to the docking of Calmodulin, and this is shown by the particles flowing from one protein surface to the other. Also in this case it is possible to start directly from the raw physico-chemical data: the volume data of the Electrostatic Potential is saved in ASCII format and can be easily read using JavaScript. With these data, it is possible to compute the potential gradient field and extract the field lines according to the desired parameters. This option would give full control on the lines extraction and make it possible to control the selection parameters on the fly during rendering: given the importance of the line selection, as previously described, this feature may be useful to study the electrical field of the molecule. In any case, being able to load the entire volumetric information may open up new possibilities to visualize the electrostatic field around the molecule. Rendering methods such as ray-casting, interactive slicing and volume splatting are possible on this platform.

5 Future Development

The proposed technology is far from being complete: the WebGL standard is not yet completely finalized and also the SpiderGL wrapping is still an ongoing project. To provide a complete platform for the development of specialized visualization tools for the web platform, some more work will be needed to make this technology accessible to people with not much experience in computer graphics programming. This effort should ideally result in the creation of a reusable library of basic functions which will ease the creation of simple visualization schemes and, at the same time, serve as a code base for more complex results. In perspective, to give a useful instrument to the general public of molecular biology scientist, we will have to work in three different directions:

- a series of *importers* from different file formats which are common to the biology community: more readable formats means more diverse data to play with;
- *utility functions* to manipulate data: because visualization is always a matter of filtering data using standard mathematical/statistical approaches;
- a series of *standard shaders* to be used for rendering: a shader library would save the time required to write simple visualization techniques and give the base for experimentation in creating advanced custom shaders;

An active research problem in the biology community is the calculation of protein motion (i.e. the description of atomic trajectories while transiting from one conformation to another), this kind of online visualization technology would prove quite useful for the evaluation and sharing of new results with the research community. It is however still difficult to display animation of the molecular surface in a way that is biologically accurate but at the same time computationally effective. Animating a structure representation of a molecule (atomic sphere, balls+sticks, ribbon) may be easy enough, since it involves rigid roto-translation of rigid entities. On the contrary, the motion of molecules make the surface undergo major modification and radical change in topology (genus change, merging/dividing parts) thus making it impossible to use animation techniques like skeletal or keyframe. The use of techniques like

metaballs may produce surfaces in realtime, but with very low accuracy from the biological point of view. A better idea could be to exploit the GPU processing power to generate the animated geometry on the fly using, for example, ray-casting methods. An efficient storage and retrieval of such animations is another interesting problem, especially in the context of web-based applications, which present additional constrains of low resources and low bandwidth.

6 Conclusions

We have presented here a technology, based on the WebGL standard, which can be profitably used to build, on the web platform, interactive 3D visualization schemes for the scientific data produced by molecular and cellular biology research. By using the low-level features of WebGL, enriched by the utility functions and higher-level classes provided by the SpiderGL library, it is possible to build web-based visualization prototypes which are not only completely custom, but also use advanced shading and rendering techniques. We have discussed the possibilities offered by this technology, describing the available components and how they are used in the creation of an interactive visualization scheme. Moreover, we have shown how it was possible to use this technology to port a specific visualization method on the web platform, and how it was possible to enrich it with additional visual elements, made available by the use of this technology. This technology is still not complete, since the WebGL standard is not yet completely fixed, and the SpiderGL library still an ongoing project; nevertheless, this combination of libraries and working strategy is a promising instrument to deal with the needs of the molecular and cellular biology research community.

Acknowledgements

This work has been financed from Regione Toscana through the project "Studio Animazione 3D" to Monica Zoppè. This work sprouted from the collaboration between the VC Lab of ISTI-CNR and the SCIVIS Group of IFC-CNR, the authors want to thank all components of both groups for their support.

References

ANDREI, R. M., CALLIERI, M., ZINI, M. F., LONI, T., MARAZITI, G., AND ZOPPÈ, M. 2010. Intuitive visualization of surface properties of proteins. *BMC bioinformatics - in review*.

BAKER, N. A., SEPT, D., JOSEPH, S., HOLST, M. J., AND MCCAMMON, J. A. 2001. Electrostatics of nanosystems: application to microtubules and the ribosome. *Proceedings of the National Academy of Sciences of the USA*, 98, 10037–10041.

BERMAN, H., HENRICK, K., AND NAKAMURA, H. 2003. Announcing the worldwide protein data bank. *Nature Structural Biology*, 10, 980.

BRUNT, P., 2010. GLGE: WebGL for the lazy. http://www.glge.org/.

CONNOLLY, M. L. 1983. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 211, 709–713.

DELANO, W. L., 2002. The pymol molecular graphics system.

DELILLO, B., 2009. WebGLU: A utility library for working with WebGL . http://webglu.sourceforge.org/.

DI BENEDETTO, M., 2010. SpiderGL: 3D Graphics for Next-Generation WWW. http://spidergl.org/.

DON BRUTZMANN, L. D. 2007. *X3D: Extensible 3D Graphics for Web Authors*. Morgan Kaufmann.

GROUP, T. K., 2009. Khronos: Open Standards for Media Authoring and Acceleration . http://http://www.khronos.org.

GROUP, T. K., 2009. WebGL - OpenGL ES 2.0 for the Web. http://www.khronos.org/webgl/.

GUEX, N., AND PEITSCH, M. C. 1997. Swiss-model and the swiss-pdbviewer: an environment for comparative protein modeling. *Electrophoresis*, 18, 2714–2723.

HUMPHREY, W., DALKE, A., AND SCHULTEN, K. 1996. Vmd: visual molecular dynamics. *Journal of Molecular Graphics*, 14, 33–38.

2002. Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org/.

JOGL Java Binding for the OpenGL API. http://kenai.com/projects/jogl/pages/Home.

KAY, L., 2009. SceneJS . http://www.scenejs.com.

KENDREW, J. C., BODO, G., DINTZIS, H. M., PARRISH, R. G., WYCKOFF, H., AND PHILLIPS, D. C. 1958. A three-dimensional model of the myoglobin molecule obtained by x-ray analysis. *Nature*, 181, 662–666.

KILGARD, M. J. GLUT - The OpenGL Utility Toolkit . http://www.opengl.org/resources/libraries/glut/.

KYTE, J., AND DOOLITTLE, R. F. 1982. A simple method for displaying the hydropathic character of a protein. *Journal of Molecolar Biology*, 157, 105–132.

MCGILL, G., 2010. MolecularMovies.org: a Portal to Cell & molecular Animation. http://www.molecularmovies.com/.

MICROSOFT CORPORATION, 1996. Microsoft activex controls. http://msdn.microsoft.com/en-us/library/aa751968(VS.85).aspx.

OPENGL ARB. GLU OpenGL Utility Library . http://www.opengl.org/documentation/specs/glu/glu1_3.pdf.

PETTERSEN, E. F., GODDARD, T. D., HUANG, C. C., COUCH, G. S., GREENBLATT, D. M., MENG, E. C., AND FERRIN, T. E. 2004. Ucsf chimera–a visualization system for exploratory research and analysis. *Journal of Computational Chemistry*, 25, 1605–1612.

RAGGETT, D. 1994. Extending WWW to support platform independent virtual reality. *Proceedings of INET'94, the Annual Conference of the Internet Society*.

ROCCHIA, W., SRIDHARAN, S., NICHOLLS, A., ALEXOV, E., CHIABRERA, A., AND HONIG, B. 2002. Rapid grid-based construction of the molecular surface and the use of induced surface charge to calculate reaction field energies: applications to the molecular systems and geometric objects. *Journal of Computational Chemistry*, 23, 128–137.

SCIVIS, 2005. Scientific Visualization Unit, IFC CNR. http://www.scivis.ifc.cnr.it/index.php/videos.

TARINI, M., CIGNONI, P., AND MONTANI, C. 2006. Ambient occlusion and edge cueing to enhance real time molecular visualization. *IEEE Transaction on Visualization and Computer Graphics 12*, 6 (sep/oct).

By -
Raluca Mihaela Andrei,
Mike Pan
and Monica Zoppè

Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

Introduction

Biologists know that, if the information of life is stored and transmitted through nucleic acids (DNA and RNA), the processes that do the actual work are most of the times proteins. These are active in all aspects of life, and in the latest years we are starting to get a glimpse of how they work. Proteins are machines composed of amino acids, which are in turn small groups of atoms arranged in specific ways[1]. Scientists are obtaining more and more information on the 3D arrangement of such atoms, and are starting to understand their activity through motion.

On the basis of information obtained by experiments of nuclear magnetic resonance (NMR), 3D visualization tools provided by BioBlender allow biologists to build a reasonable sequence of movement for proteins. It also includes a dedicated visual code to represent important features of their surface (Electric and lipophilic potential) on the protein itself, using photo realistic rendering and special effects.

BioBlender is a software extension of Blender 2.5[2], an interface for biological visualization that allows the user to import and interactively view and manipulate proteins. It was developed and is maintained by the Scientific Visualization Unit of the CNR of Italy in Pisa, with the help and contribution of several members of the Blender community. Material, scenes, publications and other relevant information can be found at www.BioBlender.net and/or www.scivis.ifc.cnr.it.

BioBlender for Windows is available from www.bioblender.net (on Linux machines it can be used with Wine). Because of its specialized nature, it requires the installation of PyMOL[3.4] , Python 2.6 [5] and NumPy[6] , which are all provided in Installer folder from the downloaded package.

Using BioBlender to build an animation

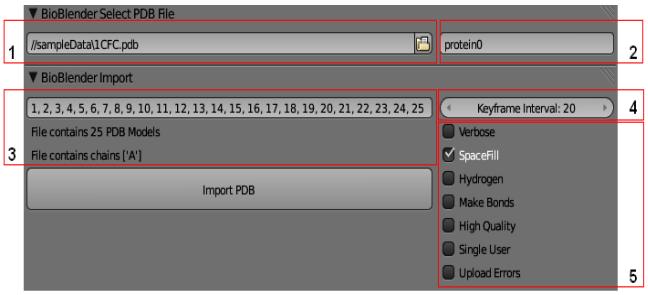
To start BioBlender, simply go to the Bin folder and launch blender.exe, then open the **template.blend** scene (stored in BioBlender folder).

Notice that the template file not only has an optimized user-interface layout for biologists, but the template scene also contains lights, camera and world settings that are ideal for visualizing molecules. This setup ensures that researchers who are not familiar with the 3D software can still effectively use BioBlender. Each interface element (buttons, sliders, toggles) has help text associated with it. By placing the mouse over them a pop-up text describes the function. Errors and progresses are displayed in the console. Critical errors will appear in the main BioBlender as a pop-up under the mouse. The atoms size is of order of Ångström (Å), therefore the scale used is 1 Blender Unit = 1 Å.

This tutorial assumes that you already have BioBlender downloaded on your computer, with the required programs installed.

1. Select and import a .pdb file

PDB files contain a description of one or multiple conformations (positions) of a single molecule. Different conformations of the same protein are listed in one NMR file and are called MODEL 1, MODEL 2 etc.



In the **BioBlender Select PDB** File panel:

- Select the .pdb file by browsing from your data (**1** in figure). The file included in sampleData folder contains the 25 models of Calmodulin [7]. Alternatively, simply type the 4-letter code for the .pdb file to be fetched from www.pdb.org [8] (make sure to pick an NMR file);
- Change the name of the protein (by default it is named "protein0") in the field on the right (**2** in figure). Naming the proteins is just a good habit that will help keeping the scene organized. Once a file is selected, the number of models and the chains are detected and shown in the BioBlender Import field (**3** in figure);
- Choose 2 models to import in the scene (by default all models are listed) typing their number separated by comma;
- In the **Keyframe Interval** slider (**4** in figure) set the number of frames between the protein conformations (Min 1, Max 200).

A list of options are available to be considered before importing the protein in the Blender scene (**5** in figure):

Verbose: enable to display in the console extra information for debugging;

SpaceFill: enable or disable to display the atoms with Van der Waals or covalent radii in the 3D scene, respectively;

Hydrogen: enable to import Hydrogens if they are present in the .pdb file. This option makes importing much slower and it is important only for visualization. If the .pdb file does not contain Hydrogens (or if you chose not to import them), they will be added during the Electrostatic Potential calculation using external software;

Make Bonds: enable it to have atoms connected by chemical bonds. Despite being time consuming this operation is very important in motion calculation;

High quality: displays high-quality atom and surface geometries; slow when enabled;

Single User: enable to use shared mesh for atoms in Game Engine; slow when enabled;

Upload Errors: enable to send us automatically and anonymously an email with the errors you generate. This makes us aware of the problems that arise and help us fix them.

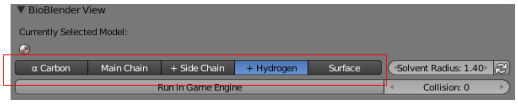
Finally, press **Import PDB** button to import the protein to the 3D scene of Blender. Blender displays the protein in motion (by linear interpolation between atoms in the conformations; Esc to stop the animation).

Raluca Mihaela Andrei[1,2], Mike Pan[1,*] and Monica Zoppè[1§]

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

2. Visualization in the 3D viewport

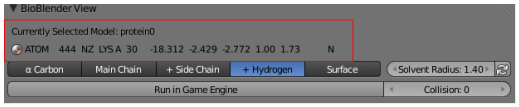
Once imported, the protein is displayed with all atoms, Hydrogens included (if the Hydrogens check-box was enabled). The first 4 buttons in the **BioBlender View** enable different views: only alpha Carbons, main chain (N, CA, C), main chain and side chains (no H), or all atoms.



If the **Surface** display mode is selected, BioBlender will compute the surface of the protein by invoking PyMOL software, an external application. It uses the **Solvent Radius** set by the user and returns the Connolly mesh [9], displayed on the BioBlender 3D view. The default radius (1.4 Å) is the standard probe sphere, equivalent to water molecules.

To check the appearance of surface calculated with different solvent radii, change the solvent radius value and press refresh button. The current surface is deleted and a new one is created.

When atoms are displayed, by selecting one atom in the 3D display, the protein information of the selected atom is printed in the area outlined below; in the 3D view the selection will extend to the other atoms of he corresponding aminoacid.



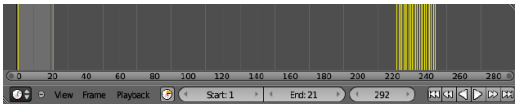
3. Protein motion using the physic engine

To calculate the transition of the protein between the 2 conformations the Blender Physics Engine is used. Press **Run in Game Engine** button to see the transition. Press Esc to leave GE and then 0 on Numerical Board to see from the camera point of view.

Hit **Run in Game Engine** button again for an interactively view. When inside the Game Engine, the mouse controls the rotation of the protein, allowing to inspect the protein from all angles. The also applies an ambient occlusion filter to the scene, giving the viewer a much better sense of depth.

Set the **Collision** mode to one of the following states: 0, 1 or 2. When set to 0 the transition between the conformations is done using linear interpolation; the atoms will simply move from one position to the other. When set to 1 the collisions between atoms are considered, resulting in a more physico-chemical accurate simulation[10].

When set to 2, the newly evaluated movement will be record to F-Curves. Go to the Timeline panel on Blender and see that the new conformations are recorded at different time (200 frames away from the last model imported) as shown in the figure below; in this way both sets of transitions are available for comparison. These conformations can be exported as described later in section 6.



Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

4. Molecular Lipophilic Potential Visualization

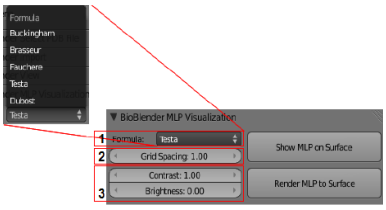
This visualization method is a novel way to see the MLP values of a protein onto the surface. Normally this is a relatively time consuming and tedious process involving running different programs from the command line, but BioBlender simplifies the entire process by allowing the user to do everything under one unified interface.

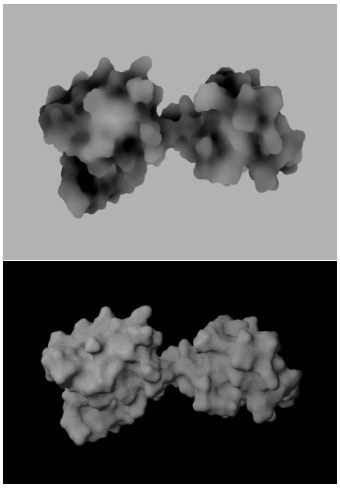
In **BioBlender MLP Visualization** section:

- Choose a **Formula** (**1** in figure; Testa formula [11] is set by default);
- Set the Grid Spacing (2 in figure; expressed in Å, lower is more accurate but slower) for MLP calculation;
- Press **Show MLP on Surface**. It may take some time as the MLP is calculated in every point of the grid in the protein space, then mapped on the surface of the protein and finally visualized as levels of grey (light areas for hydrophobic and dark areas for hydrophilic [12]).

A typical protein has varying degrees of lipophilicity distributed on its surface, as shown here for CaM.

Use **Contrast** and **Brightness** sliders to enhance the MLP representation of your protein. Once you are satisfied with the grey-levels visualization hit **Render MLP to Surface** button for the photorealistic render. This process is also time consuming and it always refers to last changes in the MLP grey-levels visualization. When the calculation is done (the button is released) press F12 on your keyboard.





Note:This is the MLP representation using our novel code: a range of visual features that goes from shiny-smooth surfaces for hydrophobic areas to dull-rough surfaces for hydrophilic ones. The levels of grey are baked as image texture that is mapped on specular of the material. A second image is created by adding noise to the first one and map it on bump. The light areas become shiny and smooth while the dark ones dull and rough as shown in the figure.

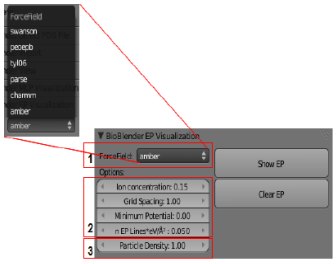
Press Esc to go back to the Blender scene.

5. Electrostatic Potential Visualization

EP is represented as a series of particles flowing along field lines calculated according to the potential field due to the charges on the protein surface. For this reason, it is necessary to perform a series of steps (as described in [12]), and to decide the physical parameters to be used in the calculation (2 in the figure).

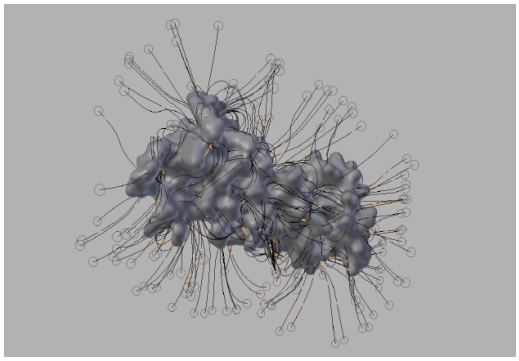
In **BioBlender EP Visualization section**:

- Choose a **ForceField** (**1** in figure; amber force field is set by default);
- Set the parameters for EP computation, using the options shown in the figure below:
- **Ion concentration** – 0.15 Molar is the default, physiological value;
- **Grid Spacing** – in Å, lower is more accurate but slower;
- **Minimum Potential** – the minimum value for which the field lines are calculated – the default value is 0 which implies calculation of all possible lines; increase it if you want to enhance the representation of EP;
- **n EP lines*eV/Å2** – the number of field lines calculated for eV/Å2.



Now press **Show EP** button. *The process is time consuming as **Show EP** button invokes a custom software that calculates the field lines and exports them in the BioBlender 3D scene as NURBS curves. The positive end of each curve becomes an emitter. The particles flow along the curves from positive to negative.*

Change the **Particle Density** (3 in figure) to modify the number of the particles visualized in the scene. **Clear EP** to delete the curves and the emitters.



6. Output

To see the protein movement with the surface properties you have to render a movie. Since the movement implies a change of the atomic coordinates, the surface properties must be recalculated frame by frame.

In the **BioBlender Output** panel set the output file path (by default it is set to tmp folder); choose the kind of representation you prefer to render from the **Visualize** curtain menu:

- **Atom** – render only atoms;
- **Plain Surface** – render only surface;
- **MLP** – render surface with MLP;
- **EP + Plain Surface** – render surface (no MLP) and EP;
- **EP + MLP** – render surface with MLP and EP;

set **Start Frame** – the first frame of the animation;

Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

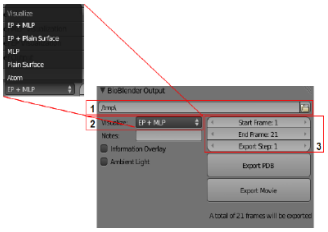
1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

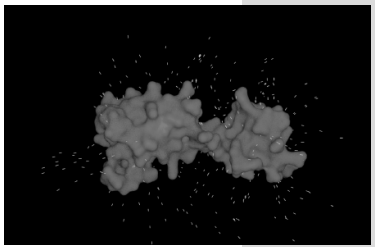
set **End Frame** – the last frame of the animation;

set **Export Step** – the number of frames to skip during export, mostly used for faster export of .pdb files; enable **Information Overlay** to print extra information on the final image; enable Ambient Light only for GE visualization; do not enable it for MLP representation as its effect is confusing for MLP visual code.



Hit **Export Movie** to render every frame of the animation. The output is a sequence of still images, this ensures that the rendering is resumed if the rendering process is disrupted. During section 3 Blender GE calculated and recorded intermediate conformations as keyframes. To save these coordinates as .pdb files for further analysis using external software, press **Export PDB**. A .pdb file is saved for each frame in the selected output.

To obtain the movie follow standard Blender procedures: open the Video Sequencer Editor: **Add -> Image**, select the sequence of images, go to **Properties** window and set the **Output** path and the **File Format** to **AVI JPEG** in the **Output** panel and **Start** and **End** frame in the **Dimensions** panel. Now press **Animation** button in the **Render** panel.



Now you have your protein moving with the surface properties visualized. An image of CaM with EP and MLP is shown in the image below ■

References

1 Zoppè, M; Porozov, Y; Andrei, R M; Cianchetta, S; Zini, M F; Loni, T; Caudai, C; Callieri, M (2008) Using Blender for molecular animation and scientific representation. Proceedings of the Blender Conference Blender

2 DeLano, WL, The PyMOL Molecular Graphics System, 2002

3 The PyMOL Molecular Graphics System, Version 1.2r3pre, Schrödinger, LLC

4 Python

5 NumPy

6 Kuboniwa H, Tjandra N, Grzesiek S, Ren H, Klee C B, Bax A (1995) Solution structure of calcium-free calmodulin. Nat Struct Biol 2: 768-76

7 Berman, H M; Westbrook, J; Feng, Z; Gilliland, G; Bhat, T N; Weissig, H; Shindyalov, I N; Bourne, P E (2000) The Protein Data Bank. Nucleic Acids Res 28: 235-42

8 Connolly, M L (1983) Solvent-accessible surfaces of proteins and nucleic acids. Science 221: 709-13

9 Zini, M F; Porozov, Y; Andrei, R M; Loni, T; Caudai, C; Zoppè, M (2010) Fast and Efficient All Atom Morphing of Proteins Using a Game Engine. (under review)

10 Testa, B; Carrupt, P A; Gaillard, P; Billois, F; Weber, P (1996) Lipophilicity in molecular modeling. Pharm Res 13: 335-43

11 Andrei R M, Callieri M, Zini M F, Loni T, Maraziti G, Pan M C, Zoppè, M (2010) A New Visual Code for Intuitive Representation of Surface Properties of Biomolecules. (under review) Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

12 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca,

13 Pisa, Italy

14 2Scuola Normale Superiore, Pisa, Italy

15 *Present address: University of British Columbia, Vancouver, Canada Corresponding author

Intuitive representation of surface properties of biomolecules using BioBlender

Raluca Mihaela Andrei[1,2], Marco Callieri[3], Maria Francesca Zini[1,†], Tiziana Loni[1], Giuseppe Maraziti[4], Mike Chen Pan[1,*] and Monica Zoppè[1,§]

[1]Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, 56124, Italy

[2]Scuola Normale Superiore, Pisa, 56125, Italy

[3]Visual Computing Lab, ISTI, CNR of Italy, Area della Ricerca, Pisa, 56124, Italy

[4]Big Bang Solutions, Navacchio (Pisa), 56023, Italy

[†] Present address: University of Pisa, Pisa, 56125, Italy

[*]Present address: Harvard Medical School, Boston, MA 02115. USA

[§]Corresponding author

Email addresses:

RMA: r.andrei@ifc.cnr.it

MC: callieri@isti.cnr.it

MFZ: myrtil@gmail.com

TL: tialo@tiscali.it

GM: giuseppe.maraziti@libero.it

MCP: mike.c.pan@gmail.com

MZ: mzoppe@ifc.cnr.it

Abstract

Background

In living cells, proteins are in continuous motion and interaction with the surrounding medium and/or other proteins and ligands. These interactions are mediated by protein features such as electrostatic and lipophilic potentials. The availability of protein structures enables the study of their surfaces and surface characteristics, based on atomic contribution. Traditionally, these properties are calculated by physico-chemical programs and visualized as range of colors that vary according to the tool used and imposes the necessity of a legend to decrypt it. The use of color to encode both characteristics makes the simultaneous visualization almost impossible, requiring these features to be visualized in different images. In this work, we describe a novel and intuitive code for the simultaneous visualization of these properties.

Methods

Recent advances in 3D animation and rendering software have not yet been exploited for the representation of biomolecules in an intuitive, animated form. For our purpose we use Blender, an open-source, free, cross-platform application used professionally for 3D work.

On the basis Blender, we developed BioBlender, dedicated to biological work: elaboration of protein motion with simultaneous visualization of their chemical and physical features.

Electrostatic and lipophilic potentials are calculated using physico-chemical software and scripts, organized and accessed through BioBlender interface.

Results

A new visual code is introduced for molecular lipophilic potential: a range of optical features going from smooth-shiny for hydrophobic regions to rough-dull for

hydrophilic ones. Electrostatic potential is represented as animated line particles that flow along field lines, proportional to the total charge of the protein.

Conclusions

Our system permits visualization of molecular features and, in the case of moving proteins, their continuous perception, calculated for each conformation during motion. Using real world tactile/sight feelings, the nanoscale world of proteins becomes more understandable, familiar to our everyday life, making it easier to introduce "un-seen" phenomena (concepts) such as hydropathy or charges. Moreover, this representation contributes to gain insight into molecular functions by drawing viewer's attention to the most active regions of the protein. The program, available for Windows, Linux and MacOS, can be downloaded freely from the dedicated website www.bioblender.org

Background

The fact that we humans are very good at extracting information through visual observation is well synthesized in the old adage "a picture is worth a thousand words". The solution of the 3D structure of myoglobin in 1958 by Kendrew [1] marked the beginning of the new era of structural biology. Since then, a wealth of protein structures has been solved and today the Protein Data Bank (PDB) counts over 67.000 protein structures [2, 3].

With the availability of all these data, and with the advance of computer graphics (CG) technologies, tools for the visualization of 3D structures were created such as VMD [4, 5], SPDBViewer [6, 7], Chimera [8, 9], PyMOL [10, 11] and others. Balls and sticks for atoms and bonds, ribbons for the secondary structures, and molecular surfaces are some of the possible representations of proteins. Most programs can also calculate surface features such as electrostatic potential (calculated with APBS [12] or

DelPhi [13]) and hydropathy (Kyte-Doolittle [14]). When present, these features are represented as field lines and/or as ranges of colors.

Since the late '90s, the development of CG techniques has advanced at spectacular pace. Among the most widely used tools, is the art and science of 3D animation. This technique consists in the creation and animation of 3D objects (complete with surfaces, skeletons, and simulated physical properties) in a virtual world, which can be 'filmed' using virtual cameras and lights. Several programs are available for this, including the commercial packages Maya/Autodesk, 3D Studio Max and Softimage XSI (all from Autodesk, [15]), Cinema 4D (from MAXON Computer GmbH [16]) and the open-source Blender [17].

Not surprising, all of these have been used for the study and representation of biological molecules and processes. Some examples are collected and visible on dedicated websites [18, 19]. The films range from the simple representation of the mechanical functioning of a single protein, to complex events involving many subjects such as DNA replication and RNA processing, to views of major cellular processes, such as apoptosis, *etc.*. These latter ones are important scientific efforts and add to their educational value the bonus of rising interest in the general public to approach biology.

For our purpose we use Blender, an open-source, free, cross-platform 3D application. Blender is a powerful instrument for 3D modeling, animation, gaming and rendering, that provides a complete workbench for producing still images, simple animations or very complex scenes with thousand of objects in motion, all textured, lighted and filmed for proper view.

Traditionally, the process of creating a 3D animation film consists of a number of steps roughly grouped in modeling, animation, rendering, special effects and compositing. Blender offers a platform to elaborate and integrate all of these steps. Objects are created in the virtual world by modeling them in the 3D scene starting from primitives or by importing them from other programs. A *time line* holding key frames (points in time in which objects have defined configuration set-ups) is used to animate the objects in the scene in various ways: by direct rotations/translations of the object, by mesh deformation obtained moving its components (vertices, edges, faces), via skeleton (inverse or forward kinematics) or by using the Game Engine (GE), typically deployed in video games. Additionally, physics-based animations can be achieved by simulated forces such as gravity, magnetic, vortex, wind *etc.* Objects are given a surface appearance by the use of material shaders and textures. These two elements define the behavior of the surface when illuminated, by specifying local information like color, reflectance (dull or shiny) and microstructure (roughness or smoothness).

Once the animation and texturing is defined, the scene is equipped with other assets such as a background, lights and cameras and the process concludes with the 'filming' (rendering of all frames which are assembled to generate a video).

In this article, we illustrate a step forward in the direction of using bio-animation both as a divulgation and as a discovery tool. Our aim is to visualize molecules in a directly informative way, also showing their motion obtained from structural data (Figure 1). This task is done using BioBlender [20], in which Blender is used to access several scientific programs. BioBlender is an engine built in Blender with an interface for biological visualization (Figure 2).

The use of Blender GE to elaborate the movement of proteins, starting from 2 or more conformations is described in Zini *et al*. (BioBlender: Fast and Efficient All Atom Morphing of Proteins Using Blender Game Engine, manuscript submitted). Briefly, starting from data from NMR collections or X-rays of the same protein crystallized in different conditions, we use Blender GE, equipped with special rules approximately simulating atomic behavior, to interpolate between known conformations and obtain a physically plausible sequence of intermediate conformations. This sequence is output as a list of pseudo .pdb file (list of atoms with their x,y,z coordinates) which are the basis for the visual elaboration described here.

As the result of this study, we propose a new visual code for the representation of two important surface properties: electrostatic potential (EP) and molecular lipophilic potential (MLP). Using features different from color permits their simultaneous delivery in photo-realistic images leaving the utilization of color space for the description of other biochemical information. Here we describe the details of this process.

Methods

Programs and scripts

BioBlender is an extension of Blender, in which custom Python scripts have been implemented for building the interface, importing the meshes and the curves, converting MLP values into vertex colors and managing various scientific programs (www.bioblender.org).

In the construction of BioBlender, we have made ample use of several existing programs, listed here.

Blender 2.5 – a free, open source, cross platform suite of tools for 3D creation [17].

PyMOL 1.2r3pre – a Python-enhanced molecular graphics tool [10], used for visualization of .pdb files. It calculates the electrostatic potential through APBS plug-in. This tool is also used to generate the 3D mesh of the molecular surface for the molecule. The obtained geometry is exported in .wrl format, easily read by 3D software tools.

PDB2PQR-1.6.0 – [21, 22] a software package that automates many of the common tasks of preparing structures for continuum electrostatics calculations, providing a platform-independent utility for converting protein files from PDB format to PQR format. It assigns partial atomic charge to every atom according to different force fields (AMBER 94, CHARMM 27 or PARSE) and saves a .pqr file in which the occupancy and temperature columns are replaced by atomic charge and radius, respectively. It also adds missing hydrogens, calculates pKa values and generates an input (.in) for APBS calculations. The .in file stores information on the 3D dimension of the protein, the ionic concentration of solvent, biomolecular and solvent dielectric constants. Ionic concentration of 0.150 mol/l NaCl, biomolecular dielectric constant of 2 and solvent dielectric constant of 78.54 (water) were used for our calculation.

APBS-1.2.1 (Adaptive Poisson-Boltzmann Solver) – [12] a software for evaluating the electrostatic properties of nanoscale biomolecular systems, through solution of the Poisson-Boltzmann equation. APBS takes as inputs a .pqr and an .in file and calculates the electrostatic potential in every point of a grid in the protein space, which is output as a .dx file.

scivis.exe – a custom software written in C++ used to calculate the field lines and to export them in a ASCII file to be imported in Blender. This tool imports the 3D surface (.obj) and the EP grid (.dx) calculated by APBS. The computation of the field lines is a multi-step process: EP values are mapped on the 3D surface, a gradient field

is calculated in the volume containing the molecule, an automatic selection of areas with high values of EP is done and the corresponding field lines are computed for these areas using the gradient field. When used as primary application, in addition to the described features, scivis.exe provides visual feedback for all its processing steps. It is possible to visualize the molecular surface, the EP grid, the gradient grid and the field lines.

Python 2.6 – an interpreted, interactive, object-oriented, extensible programming language [23]. In this project, Python has been used in different stages, both as a scripting component of various software tools (like Blender and PyMOL) and as a stand-alone scripting language.

pyMLP.py – a Python script written and kindly provided by Julien Lefeuvre (available from [24]); it contains a library of atomic lipophilic potential for all atoms present in proteins (we added the values for some mono-saccharides and nucleic acids) and it calculates the MLP in every point of a grid in the protein space according to various formulae such as Fauchere, Dubost, Brasseur, etc. (we introduced Testa formula). The grid step can be changed by the user to cope with the protein size and computer performances (in terms of memory occupancy and calculation time).

Results

We present here a software/method to produce the simultaneous visualization of EP and MLP on proteins. In the case of moving proteins, the program produces a rendered animation, in which every second of the resulting movie contains 25 images (24-30 frames per second is the standard video speed), and at every frame the shape, EP and MLP of the molecule are automatically recalculated.

In the elaboration of each frame representing proteins, still or in motion, the steps of object (mesh) creation, surface calculation and data manipulation for EP and MLP are elaborated independently using both scientific and CG programs to obtain the series of frames compositing the animation (Figures 3 and 4).

Protein surfaces

The molecular surface of proteins [25] is calculated in PyMOL starting from the .pdb file, as shown in Figure 3, upper left. For series of conformations (obtained with Game Engine or derived from molecular dynamics), the procedure is reiterated. PyMOL was chosen because the surfaces created by this software have a regular triangulation even at low polygon resolution and it is only marginally afflicted by the problem of internal disjoint surfaces. In the 3D mesh used in the example reported in Figure 5A and in other tests with wider range of dimensions (number of polygons between 4.5 and 50 thousands), all the triangles have similar areas. The mesh is exported by PyMOL as a .wrl, a file which contains information about the position of the vertices, edges, characteristics of the material of the polygon *etc.*.

MLP calculus

The MLP calculus (Figure 3 upper right) is done using pyMLP.py [26, 27]. This script calculates the lipophilic potential in every point of a grid in the space of the protein and exports the values in a .dx file. The script contains a library of atomic lipophilic potential values for every atom based on its chemistry, and several formulae for MLP calculation. However it does not support the Testa formula [28],

$$MLP(r) = \sum_i f_i e^{\frac{-|r-r_i|}{2}}$$

where r is a point in the protein space, f_i is the atomic lipophilic potential for atom i, and r_i is the position of atom i.

The Testa formula is an atom-based function that uses the Broto [26] fragment scheme and an exponential distance function, appropriate for protein calculations; therefore we modified pyMLP.py to include Testa formula. The MLP accuracy depends on the grid spacing (a in Figure 2); in BioBlender the default is set at 1Å, a dimension comparable to the mean size of the triangle edge of the 3D mesh; this parameter is a good compromise between MLP data, mesh triangulation, computer memory and time for calculation (see below).

pyMLP outputs a .dx file in which the header defines the grid origin, the grid step and the number of points on each axis.

MLP rendering

The code for the representation of hydropathy that we propose is a range of optical features that go from smooth-shiny surface (hydrophobic) to rough-dull (hydrophilic), as shown in Figure 5C.

Data elaboration for rendering is done in a series of steps (Figure 3, lower part):

1. *MLP values mapping on the mesh*. The MLP values (typically between -3 and 1 for soluble, membrane-embedded and cytoplasmic proteins) are mapped on the surface of the molecule by assigning values of MLP to the mesh. The algorithm (included in a custom program, OBJCreator) is simple: for every vertex of the mesh, the correspondent grid-cell, in the MLP grid, is identified and the value of potential is calculated using trilinear interpolation and assigned to the vertex.

This process is very fast (about 2 seconds on a personal computer for calmodulin) and the mesh vertex density is high enough to represent smoothly the potential spatial transition. The information about the MLP values corresponding to every vertex are stored in the V field of an .obj file as texture coordinates (U and V).

2. *MLP values conversion into vertex colors*. MLP values (previously assigned to the vertices of the mesh) are converted into vertex colors, assigning the same value for each RGB channel, to obtain levels of gray). For the conversion we normalize the range of the MLP values ([-3,1]) to the range of gray scale ([0,1]), and set value 0 of MLP to correspond to the value 0.5 of the gray scale. In this way the hydropathy of the protein is visualized in Blender as levels of gray: bright areas representing hydrophobicity and dark areas hydrophilicity (Figure 5B). The use of the default conversion scale provides a coherent representation for all proteins; however, at this step, to enhance MLP features for any particular protein under study, the user can modify contrast and brightness using sliders (b in Figure 2).

3. *Creation of the first image texture.* The mesh is unwrapped to generate a texture parametrization and the per-vertex color values are saved ('baked') in a texture image. UV unwrapping is a procedure that consists in flattening a 3D object (e.g. the world globe) on a 2D plane (e.g. the world map), so that each vertex of the 3D mesh is assigned a correspondent 2D texture coordinate [29]. The 2D image is also called image texture or UV map, where U and V are the texture axes.

4. *Creation of the second image texture*. In order to make the more hydrophilic areas rough the procedure involves the addition of a noise pattern of amplitude proportional to the degree of gray of the texture. This is achieved using the Node Editor of Blender, adding a Gaussian noise to the texture image, which produces an image with a strong noise over the black regions, gradually reduced on gray regions to reach a level of no noise on white. In the rendering process this noise is converted to bump, as explained below.

5. *Addition of specularity and roughness.* In the final rendering step, the image obtained in the first step (gray scale) is finally mapped on specularity from dull to

shiny, and the second image is mapped on bump. Bump mapping is a rendering technique generally used to represent very small scale geometry like scratches, roughness or graininess. This technique does not affect the geometry of the object: the perceived local geometry is only an optical effect obtained by light reflection modifications. In the final image hydrophobic areas are represented as reflective and smooth, while the more hydrophilic ones as duller and rougher (Figure 5C). By avoiding the use of color, as well as of gray scale, the differences in color are only due to the effect of light interacting with the surface, i.e. the darker areas are the least illuminated.

EP calculus

While the use of movies is mostly intended to show transition between conformations of a protein, it also allows the introduction of special effects of CG to convey other information. We have elaborated the following procedure using both BioBlender and external programs to display the EP associated with molecular (partial) charges (see Figure 4, right side). All programs are accessed through BioBlender interface, also used to set specific parameters.

The .pdb file used for mesh creation and MLP calculus is submitted to PDB2PQR program [21, 22] which outputs 2 files: .pqr and .in. These files store information on the size and charge of every atom, and on the dimensions of the protein, the ionic concentration, biomolecular and solvent dielectric constant, respectively. Both .pqr and .in are input files for APBS program [12], that calculates the electrostatic potential in every point of a grid in the space of the protein and exports the values in a .dx file, analogous to the one seen above for MLP. The force field, the ion concentration and the grid spacing can be set by the user (c in Figure 2).

EP is redrawn as field lines calculated by a custom software, scivis.exe, that combines information from the mesh file (.obj) with EP values (see below). This computation comprises different steps:

1. Mapping EP on the surface mesh

2. Transformation of the grid of EP values into a grid of gradients

3. Selection of more active surface areas by weighted Monte Carlo sampling

4. Drawing of field lines that are stored in a .txt file

The EP values are mapped on the surface of the protein by assigning a value of EP to every vertex of the mesh, as seen above for MLP, i.e. trilinear interpolation.

A grid of gradient vectors is built starting from the scalar field of EP values: for each point the gradient is calculated according to the values in neighbor points, finding the direction and slope of EP change.

The gradient data are used to generate the field lines in the space surrounding the protein. From the infinite possible lines, we are interested in generating a 'meaningful' subset comprising the lines associated with areas of the mesh with high value of EP, obtaining a distribution of lines that is proportional to the surface EP value: more lines will rise in the more electrically active areas, and the total number of lines will be proportional to the global level of potential of the molecule. This selection is done by Monte Carlo sampling weighted with respect to the potential value of the surface in each area.

For the selection of this subset, the user has two controls (d in Figure 2): the absolute EP value on the surface from which the creation of the field lines starts (lines are generated only in areas with an EP higher than a threshold – Minimum potential) and a parameter that represents the general line density (expressed as Number of lines x $eV/Å^2$). By modulating this parameter users can select the most appropriate value for

a group of proteins, obtaining a concentration of field lines which is coherent across the various proteins.

Once the 'interesting' locations (points) are selected, the lines are calculated by following the gradient in both directions, iteratively moving with small steps according to the gradient (small-step integration). Line points are added until one of the following three conditions is met: 1. the limit of the calculated grid is reached, 2. the line intersects the mesh or 3. the field is too low (the gradient is approximately 0 or equal to the value set by the user). The lines are saved as sequences of points in an ASCII file (.txt).

Thanks to the random nature of the selection procedure, lines do change every time the procedure is run but the more electrically active areas (where more lines are present) are readily identifiable. This property proves to be particularly effective when represented in animation, since it gives the idea of fuzziness, useful for electricity representation, while conveying the information about EP distribution on the surface.

In the case of Calmodulin, depicted in Figure 1, and even more evident in the WebGL animated representation, most lines are directed towards the surface, due to the fact that the protein is slightly acidic, with an isoelectric point of 4.09.

EP representation

Field lines are imported into Blender as NURBS curves which are not rendered (they are invisible in the final image), but instead are used to guide a particle effect. Every curve starts at its most positive end which is associated with a particle emitter. The particles, drawn as short segments, flow along the curves from positive to negative, respecting the field lines convention in physics. During animation, the particles are generated every 5 frames (0.2 sec) and have a life-time of 20 frames (0.8 sec). This means that the system is in steady state after the sixteenth frame (see the scheme in

Figure 6). Representation of EP as moving particles on a trajectory, played in time, is interpreted easily and transmits the idea of polarity of the charged areas of a biomolecule.

If the user is interested in visualization of only one conformation, the animated particles are displayed/played in loop (they are emitted for 250 frames and have a lifetime of 20 frames).

Moving proteins

In the visualization of proteins in motion, every frame is elaborated as a single .pdb file. Because at every frame the atomic coordinates change, also the surface features (shape itself, EP and MLP, calculated by integrating the atomic values) change accordingly, and must be recalculated. Due to extremely high-level modifications (topology changes, merging/separation of surface parts) it is not possible to use a single geometry and animate it through conventional tools. It is instead necessary to rebuild the surface geometry, importing a new set of mesh coordinates at each frame. This implies a very large amount of calculations, but allows the elaboration of a sequence of images that is coherent from frame to frame, thus giving the impression of continuity.

In summary, for each frame (conformation) we visualize MLP as textured mesh and EP as curves and animated particles. The result is a sequence of frames showing the moving protein with its properties, EP and MLP, represented together: MLP as a range of visual and tactile characteristics and EP as flow of particles that move from positive to negative along invisible field lines (as shown in the movie Protein Expressions - Study N. 3 [30]).

Discussion

The description of biological phenomena has always made use of graphical presentation, starting from the early botanical and zoological drawings, including famous anatomical folios, that greatly help viewers, professionals and not, to understand and learn about nature.

Since the early times, an artistic component has been included, often unnoticed by viewers, but greatly exploited by the scientists/artists. Even today, the clearest graphical descriptions of natural and artificial subjects are hand- or CG-drawn rather than photographic images. The 'artistic' dimension allows for a better interpretation of the subject, the choice of illumination, and the removal of irrelevant details and disturbing effects.

The same attitude has motivated a number of scientists to use various graphical tricks when showing data related to structural features of macromolecules. Although most structural information contained in a .pdb file (a list of atoms and their 3D coordinates) is actually 'readable', biologists typically use graphical programs to explore protein structures; indeed the literature has an abundance of such programs, including some very popular. These programs can transfer the structural information from a linear list of atoms to a 3D virtual space and display it on 2D surface; positional information is interpreted with the aid of chemical information stored in libraries (of amino acids, nucleotides and other molecules), that introduce chemical bonds, electric charges, hydrophobicity scales and so on. In this way the user is enabled to observe features of the molecules of interest according to her/his needs.

Recent years have seen the development of 3D computer graphics techniques that have culminated in the recent success of the blockbuster movie Avatar, in which an entire world has been created in CG, including 'floating mountains' and forest with thousands of (CG built!) plants, animals, insects *etc*.

Similar techniques can be used to show the nanoscopic world of cells, populated with all sorts of environments, proteins, nucleic acids, membranes, small molecules and complexes. Indeed, there are several remarkable examples of efforts in this new discipline of Bio-Animation, some of which have reached a large public. Beside the beauty and the educational value of these animations, we consider that the very process of creating such movies includes a heuristic importance both in the development of the graphical instruments and in the studies implied in the elaboration of the subjects' (proteins) movements and interactions. In fact, when a researcher is induced to take a different point of view, such as needed for the visual elaboration, s/he will be exposed to possible new insight, facilitating better understanding of the process under study. In this way a novel spatial reasoning can complement the classical biochemical reasoning typically employed in molecular research.
Our group is among those involved in the development of animated biology, and in this paper we report one aspect of such effort, namely the elaboration, using Blender, of a code capable of showing two of the most critical features that determine the behavior of macromolecules: their electrostatic and lipophilic potentials.

Choice of Blender

Among the professional packages developed for CG, one only has the double advantage of being open source and available free of charge: Blender.
Blender is the result of a world-wide, concerted effort to put tools of the highest standard for CG creations at the reach of any artist (or scientist) regardless of her/his capability of paying for such tools. The project is guided by the non-profit Blender Foundation, and animated by countless developers that voluntarily devote time and effort to constantly introduce the most up to date techniques into the package, equipping users with any instrument they need. Blender 2.5, the latest major release,

introduced a new design of the user interface, new physics engine for smoke (volumetric), particles and soft bodies, and, importantly, the possibility to achieve all Blender's functions from scripting, through APIs.

BioBlender

On the framework of Blender 2.5, we built BioBlender, which includes a section specifically built for biological work. Inside BioBlender, for the analysis of proteins structures, various types of visualization are available: alpha carbon, main chain, main chain and side chains, all atoms (including hydrogens) and molecular surface. The elaboration of proteins' motions and the simultaneous representation of surface physico-chemical properties of proteins in motion are the innovations that BioBlender introduces in macromolecular visualization.

Elaboration of protein motion

We use Blender Game Engine to elaborate the movement of proteins, when more than one conformational state is known. Starting from data from NMR collections or X-ray of the same protein crystallized in different conditions, we use Blender GE, equipped with special rules approximately simulating atomic behavior, to interpolate between known conformations and obtain a physically plausible sequence of intermediate conformations. This sequence can be explored within Blender or can be output as a list of pseudo .pdb file (list of atoms and x,y,z coordinates) which are the basis for the visual elaboration.

It is important to notice that this procedure can be applied to any .pdb or (better) sequence of .pdb files representing a continuous series describing a conformational transition, obtained by Blender or by any other means, e.g. Molecular Dynamics simulation.

Visualization of moving proteins, and of their molecular surface features

The development of structural biology that made available tens of thousands of structures, not only improved our knowledge on structural features such as the richness of protein folds (secondary and tertiary structure), and of their association in groups (quaternary structure). It also increased knowledge associated with protein motion: in fact most proteins exert their function through some kind of motion. This is best understood by observing the movement in an animated film. The role of side chains, which are the determinants of such motions, is at present difficult to appreciate by using present visualization tools that either provide a fixed all-atom structure, or show dynamically only a limited number of atoms.

We present here a procedure that allows the direct observation of moving proteins focusing on their surface features, rather than on their structure. In particular, we have focused on hydropathy and electrical fields as they appear on and around the molecular surface.

These features can be calculated and visualized by a number of programs, which typically display them with a color code. We reasoned that for these properties a more 'photo-realistic' display would help viewers in the de-codification of their meaning, and elaborated the system here reported. Example of the use of these codes can be seen for a single protein in the Proteopedia page [31] (see also Additional file 1) and for a complex in our movie Protein Expressions – Study N3 [30].

The main idea of the proposed visual mapping is to exploit perceptual associations between the values to be mapped and visual characterization of real-world objects. Ideally, by using already established perceptual association, the viewer will be able to understand the provided information more naturally, without the use of explicit legends. For MLP mapping, two opposite surface characterizations able to convey a sense of affinity to water or to oil were selected. In our real-world experience, a very

smooth, hard surface (like porcelain or wax) is completely impervious to water but can be easily coated by oil. The opposite visual feedback is associated to grainy, crumbly, dull surfaces (like clay bricks or biscuits) which can be easily imagined being soaked in water. These considerations led to the 'painting' of highly hydrophobic areas as shiny, smooth material and of highly hydrophilic areas as dull and rough.

While the MLP value is only observable on the surface itself, electrical phenomena are associated to the idea of an effect projected in the volume surrounding a charged object, and able to affect other objects (like the high school favorite amber rod attracting paper bits). Field lines are a common way to describe the effect of the electrical field. EP value is therefore represented by showing small particles, moving along the path defined by field lines, visualizing a high concentration of particles in areas where the electrical field is stronger.

The representation of both features in black and white allows the viewer to grasp their values, without distracting with arbitrary information which is not interpretable if not associated with a de-coding legend, making it easier to interpret.

For MLP elaboration we considered that none of the available programs are accurate enough to provide useful information: most molecular displaying packages simply attribute a fixed value of MLP to every atom of a given amino acid, using the Kyte-Doolittle scale [14]. This scale was elaborated almost 30 years ago with the aim of identifying structural features of proteins, namely the interior portions of globular proteins and membrane spanning segments in membrane associated proteins, but is not indicated for the evaluation of the distribution of MLP on the molecular surface. Indeed, some other programs include a more appropriate method of calculation, such as VASCo [32] which employs the Brickman [33] formula on an atom based library

and a Fermi-type distance function. We have implemented a calculation with the Testa formula, which uses an atom-based fragment scheme and an exponential function. The values thus obtained are plotted on the vertices of the molecular surface. This procedure results in a very smooth distribution of MLP values which is then displayed with a scale of 'tactile' textures, ranging from dull-rough to shiny-smooth. The advantage of such calculation and representation is mostly noticeable in animated movies showing the transition between different conformation of proteins, when patches of hydrophobic areas are gradually exposed on the surface of proteins which will facilitate docking onto other macromolecules.

For EP, we developed a visual code based on a flow of particles (small lines) flowing towards the negative pole: this is particularly useful for the observation of interacting molecules and for molecules whose field is changing when the conformation changes. To elaborate EP we made use of several programs and integrated them in a flow whose final result is the continuous display of the EP and its development during protein conformational transitions.

Time considerations

The entire process is very fast: a protein of 2262 atoms is imported in 7 sec, while MLP and EP computation with grid spacing 1 Å take 70 and 19 seconds, respectively, on a standard personal PC equipped with WindowsXP, Intel Core 2Duo CPU, 2.33 GHz, 3.25 GB RAM.

Our example is Calmodulin: after activation due to the binding of 4 Calcium ions, the protein undergoes a major conformational transition in which both its EP and its MLP change considerably: the Ca ions introduced in the 4 EF hand domains affect the EP by virtue of their own charge and the MLP by inducing the opening of each globular

domain to expose two major hydrophobic patches which enable the protein to interact with its partners and push the calcium signal downstream in the biochemical pathway.

Proteins and their surface properties can also be visualized in a 3D interactive way on web platform exploiting the new WebGL component of HTML5. Using this API, it is possible to display 3D content in a web page without the use of external plug-ins, by writing an appropriate visualization program using the OpenGL syntax. Using a javascript support library, SpiderGL [34], we built an interactive visualization scheme [35] which accepts as input the same data (meshes, field lines and the MLP texture) calculated by BioBlender.

Conclusions

In conclusion, we have developed a computational instrument that allows the display of molecular surfaces of moving (or still) proteins, putting special emphasis on their electrical and lipophilic properties. We consider that this representation allows better (or at least more immediate and intuitive) understanding of the dynamical forces governing intermolecular interactions and thus facilitate new insights and discoveries.

Abbreviations

EP – Electrostatic Potential

MLP – Molecular Lipophilic Potential

GE – Game Engine

CG – Computer Graphics

3D – three-dimensional

Competing interests

The authors declare no competing interests.

Authors' contributions

RMA performed research, wrote and tested software; MC, MFZ, GM contributed programming; MC, MCP contributed scivis.exe and BioBlender interface, respectively; TL, MZ contributed visual elaboration with Blender; MZ conceived research; RMA, MZ wrote paper.

Acknowledgements

We thank the PDB2PQR, APBS, PyMol teams, late Warren DeLano and the Blender users and developers community for kind answers to our many questions. Work funded by Regione Toscana grant 'Animazione 3D' to MZ.

References

1. Kendrew JC, Bodo G, Dintzis HM, Parrish RG, Wyckoff H, Phillips DC: **A three-dimensional model of the myoglobin molecule obtained by x-ray analysis.** *Nature* 1958, **181**:662-6

2. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE: **The Protein Data Bank.** *Nucleic Acids Res* 2000, **28**:235-42

3. **Protein Data Bank** [www.pdb.org]

4. Humphrey W, Dalke A, Schulten K: **VMD: visual molecular dynamics.** *J Mol Graph* 1996, **14**:33-8, 27-8

5. **Visual Molecular Dynamics** [http://www.ks.uiuc.edu/Research/vmd/]

6. Guex N, Peitsch MC: **SWISS-MODEL and the Swiss-PdbViewer: an environment for comparative protein modeling.** *Electrophoresis* 1997, **18**:2714-23

7. **Swiss-PdbViewer** [http://www.expasy.org/spdbv/]

8. Pettersen EF, Goddard TD, Huang CC, Couch GS, Greenblatt DM, Meng EC, Ferrin TE: **UCSF Chimera--a visualization system for exploratory research and analysis.** *J Comput Chem* 2004, **25**:1605-12

9. **UCSF Chimera** [http://www.cgl.ucsf.edu/chimera/]

10. DeLano WL: **The PyMOL Molecular Graphics System**, 2002 DeLano Scientific, San Carlos, CA, USA

11. **The PyMOL Molecular Graphics System**, Version 1.2r3pre, Schrödinger, LLC.

12. Baker NA, Sept D, Joseph S, Holst MJ, McCammon JA: **Electrostatics of nanosystems: application to microtubules and the ribosome.** *Proc Natl Acad Sci U S A* 2001, **98**:10037-41

13. Rocchia W, Sridharan S, Nicholls A, Alexov E, Chiabrera A, Honig B: **Rapid grid-based construction of the molecular surface and the use of induced surface charge to calculate reaction field energies: applications to the molecular systems and geometric objects.** *J Comput Chem* 2002, **23**:128-37

14. Kyte J, Doolittle RF: **A simple method for displaying the hydropathic character of a protein.** *J Mol Biol* 1982, **157**:105-32

15. **Autodesk** [www.autodesk.com]

16. **Maxon** [www.maxon.net]

17. **Blender** [www.blender.org]

18. **Molecular Movies** [www.molecularmovies.com]

19. **SciVis Unit** [www.scivis.ifc.cnr.it]

20. Andrei RM, Pan CM, Zoppè M: **BioBlender: Blender for Biologists.** *BlenderArt Magazine* 2010, **31**:27-32

21. Dolinsky TJ, Nielsen JE, McCammon JA, Baker NA: **PDB2PQR: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations.** *Nucleic Acids Res* 2004, **32**:W665-7

22. Dolinsky TJ, Czodrowski P, Li H, Nielsen JE, Jensen JH, Klebe G, Baker NA: **PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations.** *Nucleic Acids Res* 2007, **35**:W522-5

23. **Python Programming Language** [www.python.org]

24. **pyMLP.py** [http://code.google.com/p/pymlp/source/browse/trunk/pyMLP.py]

25. Connolly ML: **Solvent-accessible surfaces of proteins and nucleic acids.** *Science* 1983, **221**:709-13

26. Broto P, Moreau G, Vandycke C: **Molecular structures: Perception, autocorrelation descriptor and sar studies. System of atomic contributions for the calculation of the n-octanol/water partition coefficients.** *Eu. J. Med. Chem.* 1984, **19.1**:71-78

27. Laguerre M, Saux M, Dubost JP, Carpy A: **MLPP: A program for the calculation of molecular lipophilicity potential in proteins.** *Pharm. Sci.* 1997, **3.5-6**:217-222

28. Testa B, Carrupt PA, Gaillard P, Billois F, Weber P: **Lipophilicity in molecular modeling.** *Pharm Res* 1996, **13**:335-43

29. Catmull E: **A subdivision algorithm for computer display of curved surfaces.** PhD thesis 1974, University of Utah. (available at www.pixartouchbook.com/storage/catmull_thesis.pdf)

30. **SciVis videos** [http://www.scivis.ifc.cnr.it/index.php/videos]

31. **Calmodulin motion on Proteopedia** [http://proteopedia.org/wiki/index.php/Calmodulin#Calmodulin_in_Motion]

32. Steinkellner G, Rader R, Thallinger GG, Kratky C, Gruber K : **VASCo: computation and visualization of annotated protein surface contacts.** *BMC Bioinformatics* 2009, **10**:32

33. Heiden W, Moeckel G, Brickmann J: **A new approach to analysis and display of local lipophilicity/hydrophilicity mapped on molecular surfaces.** *J Comput Aided Mol Des* 1993, **7:**503-514.

34. Di Benedetto M, Ponchio F, Ganovelli F, Scopigno R: **SpiderGL: a JavaScript 3D graphics library for next-generation WWW .** *Proceedings of the 15th International Conference on Web 3D Technology: 22 - 24 July 2010; Los Angeles, California.* Edited by Spencer, SN. ACM: New York, NY, USA; 2010:165-174

35. Callieri M, Andrei R, Di Benedetto M, Zoppè M, Scopigno R: **Visualization methods for molecular studies on the web platform.** *Proceedings of the 15th International Conference on Web 3D Technology: 22 - 24 July 2010; Los Angeles, California.* Edited by Spencer, SN. ACM: New York, NY, USA; 2010:117-126

Figures

Figure 1 – Example of BioBlender representation.
The protein (Calmodulin) is shown with its chemical and physical features represented according to the proposed code, as described in the present article. The image is a single frame from an animated sequence, showing EP and MLP. For a 3D interactive example, please visit http://www.scivis.ifc.cnr.it/images/stories/3d_interactive/VIS_CaCaM/VIS_CaCaM.html

Figure 2 – BioBlender interface
The interface is structured in 9 panels: *amino acids list* – to select and highlight amino acids in the 3D viewport, *chains list* – to select different protein chains, *proteins list* – to select different proteins; *select .pdb file* – upload from user defined path, or access directly from PDB.org specifying the 4 letter code; *import* – at the import phase, it is possible to select various parameters, including covalent/Van der Waals radius,

include/exclude Hydrogens and others; *view* – visualization in 3D working space, activation of Game Engine; *MLP visualization* – Parameters for MLP; *EP visualization* – parameters for EP; *output* – export of .pdb files and rendered frames. a: choice of formula and grid spacing; b: contrast and brightness control; c and d: calculation and representation, respectively.

Figure 3 – Procedure for MLP calculus and representation
For each .pdb file, PyMOL and pyMLP.py calculate the surface and the MLP values, respectively; then, MLP (stored in a .dx file) is mapped on the surface and both are saved as an .obj file; MLP values are converted into vertex colours, and texture images are saved. These are finally mapped on the material of the mesh, and rendered as bump and specularity effects.

Figure 4 – Procedure for EP calculus and representation
Starting from the same .pdb file used for MLP calculation, PDB2PQR adds atomic charge to each atom, then APBS calculates the EP values and stores them in a .dx file; Scivis uses the information about the mesh (previously calculated for MLP – blue squares) and the .dx file to calculate the field lines; these are imported in Blender as curves along which travel particles, emitted from their positive end.

Figure 5 – MLP mapping on the surface of Calmodulin
Steps in the creation of an image of Calmodulin are shown. **A** Panel of the 3D scene of Blender with a wireframe view, showing the fine triangulation (average edge size 1Å) of the mesh. **B** MLP representation as levels of grey. **C** Final image at high resolution showing the variation of MLP distribution over the molecular surface.

Figure 6 – Particles generation and representation for moving proteins
Field lines are imported as curves every 5 frames (0.2 seconds). Particles have a life-time of 20 frames (0.8 seconds). After the sixteenth frame (0.6 seconds) the system is in ready-state (square).

Supplementary material

1. Additional file 1. Calmodulin in motion. The movie (in .avi format) shows several transitions of calmodulin in the Apo form (without Calcium) and the major conformational change induced by the binding of 4 Ca ions. The movie can also be seen online at

http://proteopedia.org/wiki/index.php/Calmodulin#Calmodulin_in_Motion

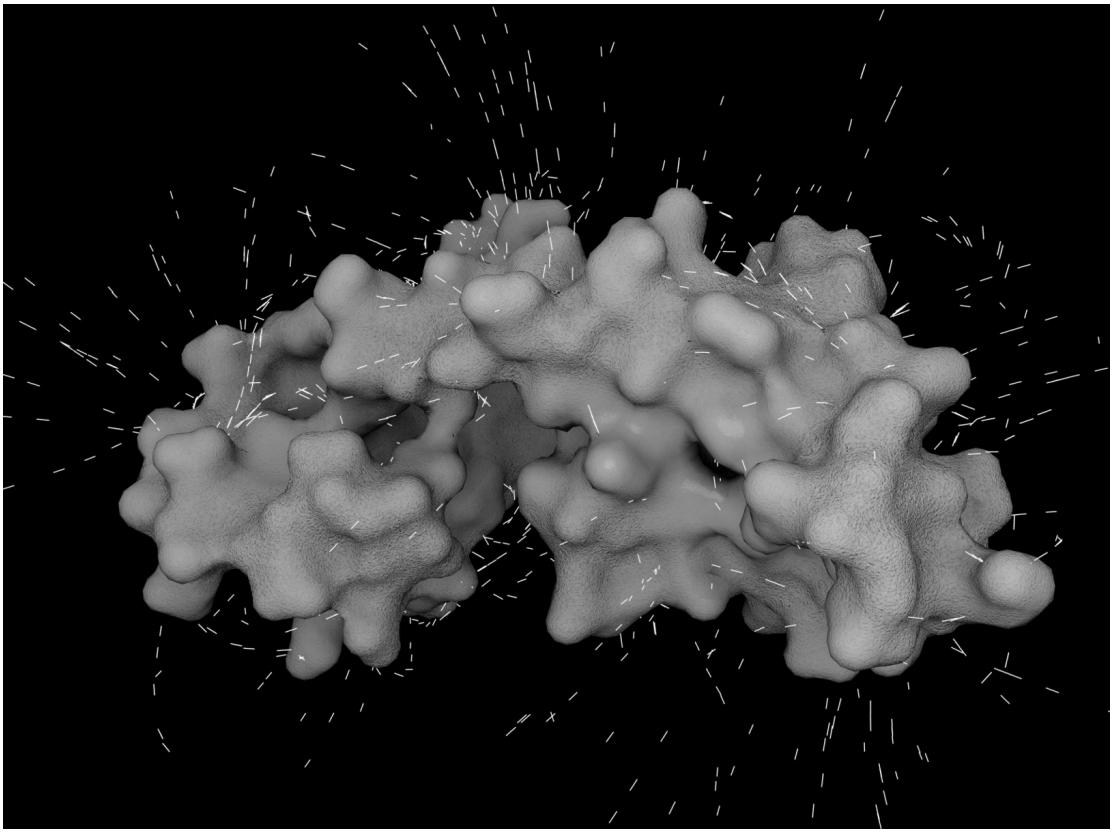


Figure 1.

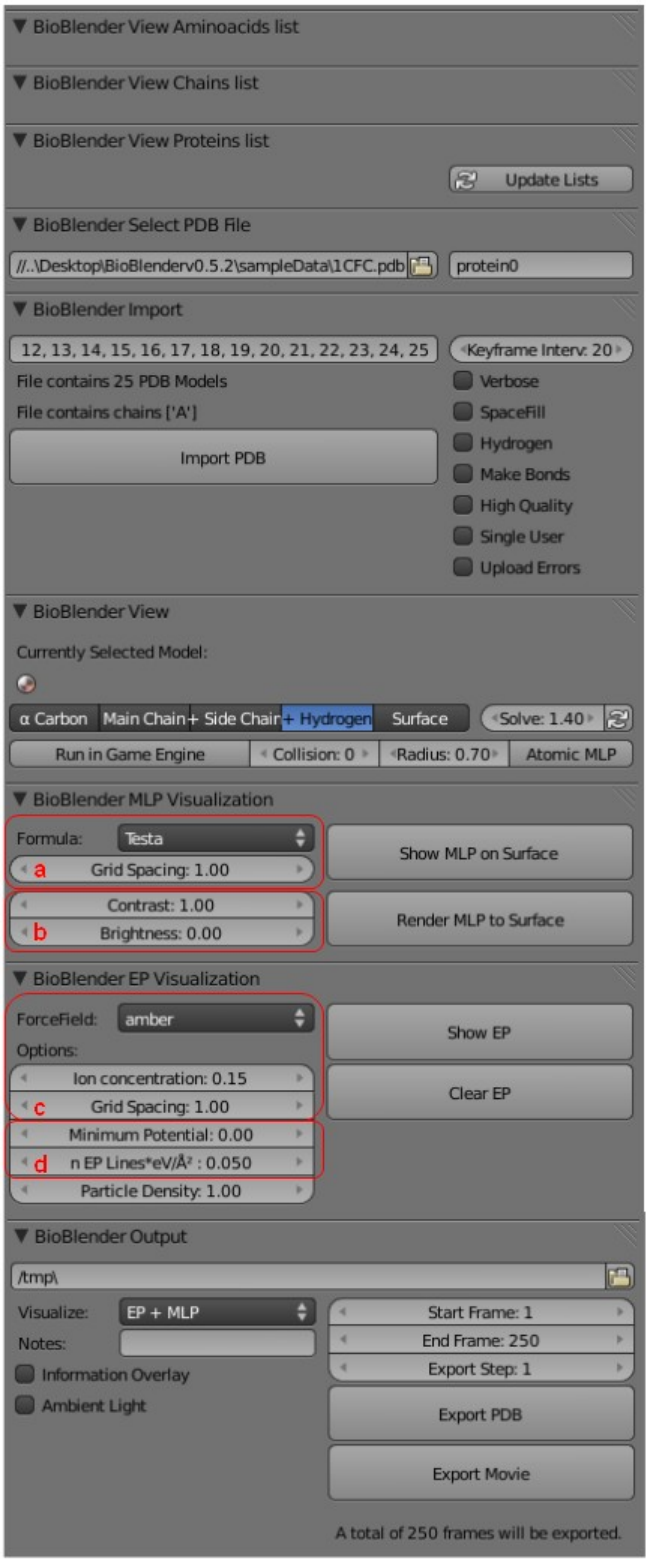

BioBlender View Aminoacids list
BioBlender View Chains list
BioBlender View Proteins list
Update Lists
BioBlender Select PDB File
//..\Desktop\BioBlenderv0.5.2\sampleData\1CFC.pdb
protein0
BioBlender Import
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
Keyframe Interv: 20
File contains 25 PDB Models
File contains chains ['A']
Import PDB
Verbose
SpaceFill
Hydrogen
Make Bonds
High Quality
Single User
Upload Errors
BioBlender View
Currently Selected Model:
α Carbon
Main Chain
+ Side Chain
+ Hydrogen
Surface
Solve: 1.40
Run in Game Engine
Collision: 0
Radius: 0.70
Atomic MLP
BioBlender MLP Visualization
Formula:
Testa
a
Grid Spacing: 1.00
Contrast: 1.00
b
Brightness: 0.00
Show MLP on Surface
Render MLP to Surface
BioBlender EP Visualization
ForceField:
amber
Options:
Ion concentration: 0.15
c
Grid Spacing: 1.00
Minimum Potential: 0.00
d
n EP Lines*eV/Å² : 0.050
Particle Density: 1.00
Show EP
Clear EP
BioBlender Output
/tmp\
Visualize:
EP + MLP
Notes:
Information Overlay
Ambient Light
Start Frame: 1
End Frame: 250
Export Step: 1
Export PDB
Export Movie
A total of 250 frames will be exported.

Figure 2.

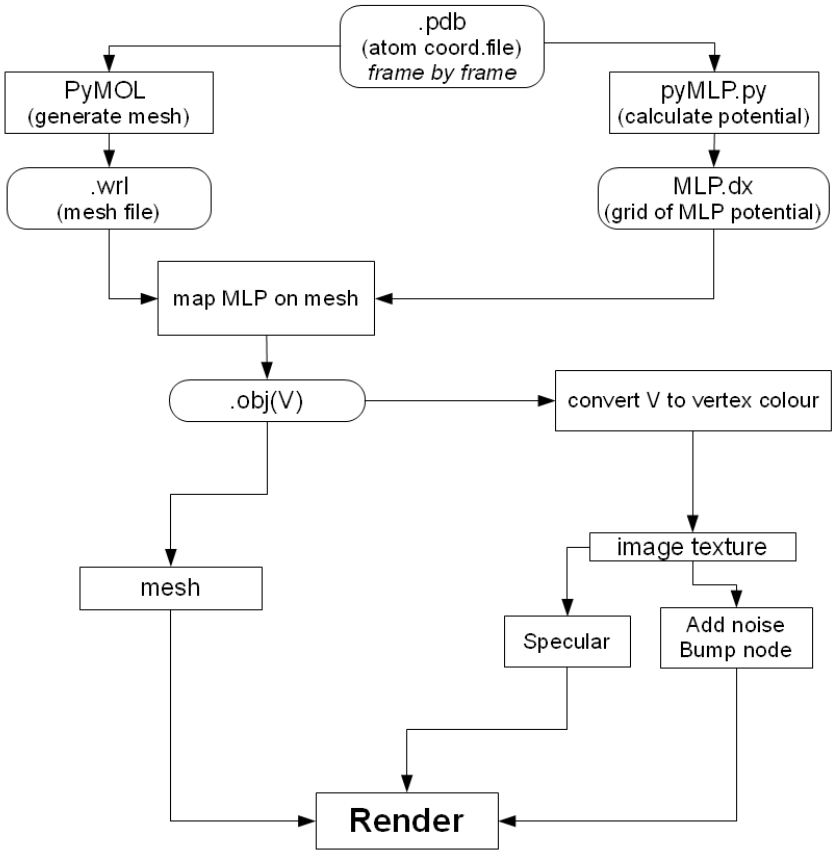

.pdb
(atom coord.file)
frame by frame
PyMOL
(generate mesh)
pyMLP.py
(calculate potential)
.wrl
(mesh file)
MLP.dx
(grid of MLP potential)
map MLP on mesh
.obj(V)
convert V to vertex colour
image texture
mesh
Specular
Add noise
Bump node
Render

Figure 3.

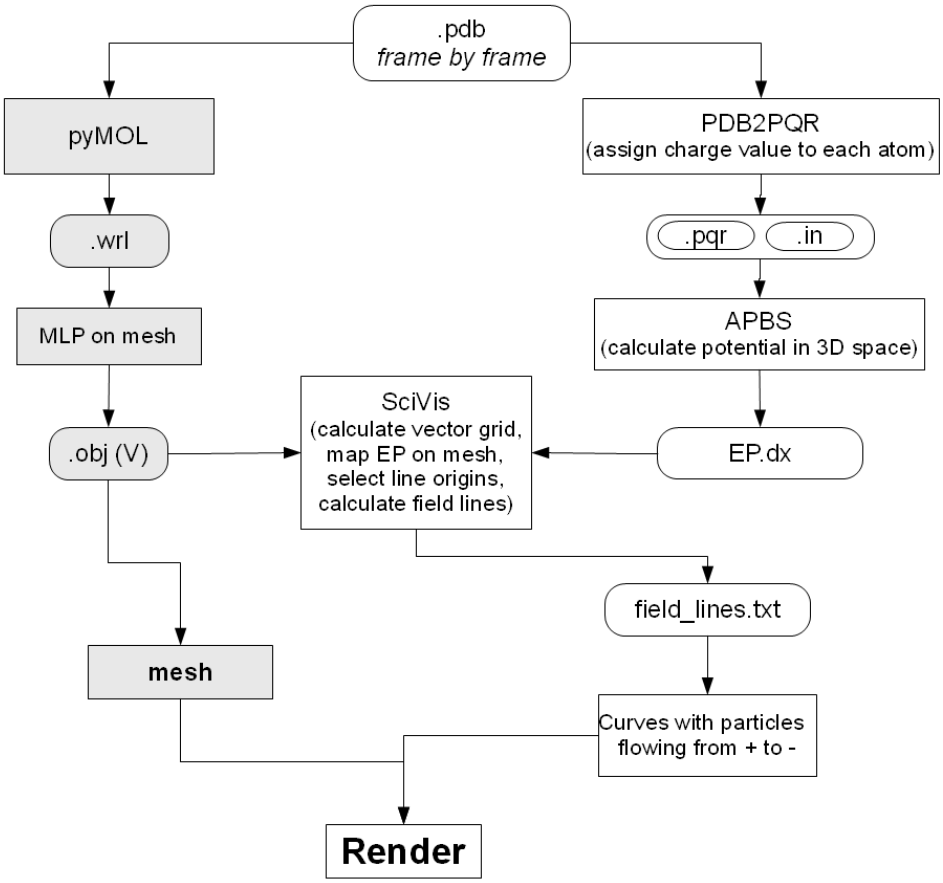

.pdb
frame by frame
pyMOL
PDB2PQR
(assign charge value to each atom)
.wrl
.pqr
.in
MLP on mesh
APBS
(calculate potential in 3D space)
SciVis
(calculate vector grid,
map EP on mesh,
select line origins,
calculate field lines)
.obj (V)
EP.dx
field_lines.txt
mesh
Curves with particles
flowing from + to -
Render

Figure 4.

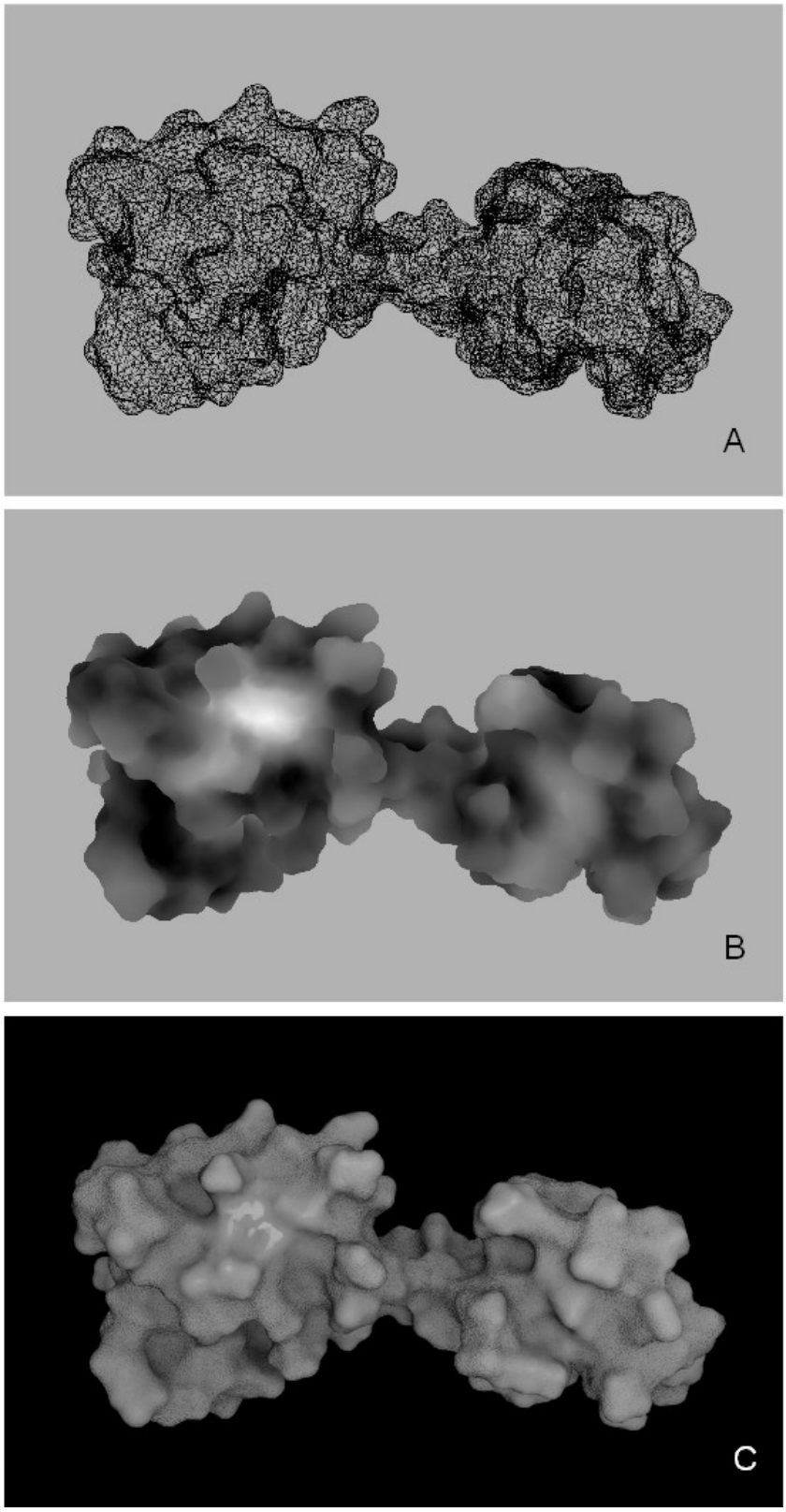


Figure 5.

ACKNOLEDGEMENTS

It is a pleasure to thank many people who made this thesis possible. It is with immense gratitude that I acknowledge the support and help of my tutor Dr. Monica Zoppè. I thank Monica for her proposal for this PhD project, her advices, our daily discussions, her patience during the long process of forming myself as a scientist, helping me to learn biology, write scientific articles and encouraging me to develop my artistic side to represent visually the magnificent world of biology.

I am grateful to Scuola Normale Superiore for this opportunity and especially to Prof. Arturo Falaschi who strongly believed in this project. Also, I would like to thank Prof. Antonino Cattaneo for assuring the continuity of my project in the Molecular Biology lab. Special thanks to Lucia Monacci, Marina Barsotti and Ambra Vettori.

I am very indebted to my colleagues from the multidisciplinary group Scivis for the fruitful collaboration, their precious help and their friendship: Mauro Lorusso and Stefano Cianchetta for helping me with Maya, Tiziana Loni for introducing me to Blender and helping me solving any problem with Blender, Mike Pan, Giuseppe Maraziti and Maria Francesca Zini for helping me with scripting, Ilaria Carlone, Maria Antonietta Pascali, Claudia Caudai and Davide Cornolti. I would like to thank my tutor for encouraging us to work in team. I thank Marco Callieri for his patience during our work, for our interesting discussions and his friendship.

The words are not enough to express how much it meant my brother's love, support and encouragement in my 'pepsi moments'. I thank Madi for her 'ready for battle' words and for being beside me for all these years despite the distance. I am deeply grateful to Mitică for his encouragement to follow my dream to study abroad. I thank Mirela for her friendship despite the distance and for her positive attitude.

I would like to thank Betta, Vincenzo, Massimo, Denise, Patrizia for their friendship and Italian “classes”, Simona for her help during my first days in Italy. I owe my PhD also to Anda and Florin who encouraged me to apply to SNS. I would like to thank Miriam for her friendship, support and for teaching me some words in Sicilian dialect.

It was a pleasure to meet Awatef in the Molecular Biology lab and I thank

her for her precious advices.

I would like to thank PyMOL, APBS, PDB2PQR, VMD, Blender, Blender Artists, Kino3D forums and mailing lists for helping me finding the answers to some questions that arose during my project.

I thank my grandmother and dna Dorina for their precious life advices.

I thank my parents Doina and Ştefan for who I am.