

BioBlender: a Software for Intuitive Representation of Surface Properties of Biomolecules

Raluca Mihaela Andrei^{1,2}, Marco Callieri³, Maria Francesca Zini¹, Tiziana Loni⁴, Giuseppe Maraziti⁴, Mike Chen Pan^{1,*} and Monica Zoppè^{1,§}

¹Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy

²Scuola Normale Superiore, Pisa, Italy

³Visual Computing Lab, ISTI, CNR of Italy, Area della Ricerca, Pisa, Italy

⁴Big Bang Solutions, Navacchio (Pisa), Italy

*Present address: University of British Columbia, Vancouver, Canada

§Corresponding author

Email addresses:

RMA: r.andrei@sns.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

In this and the associated article *BioBlender: Fast and Efficient All Atom Morphing of Proteins Using Blender Game Engine*, [1] we present BioBlender, a complete instrument for the elaboration of motion [1] and the visualization (here) of proteins and other macromolecules, using instruments of computer graphics.

The availability of protein structures enables the study of their surfaces and surface properties such as electrostatic potential (EP) and hydrophathy (MLP), based on atomic contribution. Recent advances in 3D animation and rendering software have not yet been exploited for the representation of proteins and other biological molecules in an intuitive, animated form.

Taking advantage of an open-source, 3D animation and rendering software, Blender, we developed BioBlender, a package dedicated to biological work: elaboration of proteins' motions [1] with the simultaneous visualization of chemical and physical features. EP and MLP are calculated using physico-chemical programs and custom programs and scripts, organized and accessed within BioBlender interface.

A new visual code is introduced for MLP visualization: a range of optical features that permits a photorealistic rendering of its spatial distribution on the surface of the protein.

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

Our system permits EP and MLP visualization of molecules and, in the case of moving proteins, the continuous perception of these features, calculated for each intermediate conformation. 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 hydrophobicity or charges.

Introduction

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 [2] 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 60.000 protein structures [3, 4].

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 [5, 6], SPDBViewer [7, 8], Chimera [9, 10], PyMOL [11, 12] 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 [13] or DelPhi [14]) and hydrophobicity (Kyte-Doolittle [15]). When present, these features are represented as field lines and/or as ranges of colours.

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, [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 www.molecularmovies.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 our purpose we use Blender, an open-source, free, cross-platform 3D application. Blender is a powerful instrument for 3D modelling, 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 modelling, 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 modelling 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 behaviour of the surface when illuminated, by specifying local informations like colour, 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 show internal motion of proteins obtained from structural data, visualizing molecules in a directly informative way. This task is done using BioBlender, 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's GE to elaborate the movement of proteins, starting from 2 or more conformations is described in Zini *et al.*. 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 behaviour, 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.

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

As the result of the visualization effort, 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 colour permits their simultaneous delivery in photorealistic images leaving the utilization of colour space for the description of other biochemical information. Here we describe the details of this process.

Results

The software/method presented here outputs the simultaneous visualization of EP and MLP on proteins. 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 as standard video speed).

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 1 and 4).

Protein Surfaces

Molecular surface of proteins [18] is calculated in PyMOL starting from the .pdb file, as shown in Figure 1, upper left. For series of conformations (obtained with Game Engine or derived from MD), 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 afflicted at low level by the problem of internal disjoint surfaces. In the 3D mesh used in the example reported in Figure 3A 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 1 upper right) is done using pyMLP.py [19, 20]. It 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 lipophilic atomic potential for every atom based on its chemistry, and several formulae for MLP calculation; however it does not support the Testa formula,

$$MLP(r) = \sum_i f_i \times e^{\frac{-|r-r_i|}{2d}}$$

an atom-based function using Broto fragment scheme and an exponential distance function, more appropriate for protein calculi [21]. 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 combination is a good compromise between MLP data, mesh triangulation, computer memory and time for calculation.

PyMLP outputs a .dx file in which the header defines the grid origin, the grid step and the number of points on each axis.

The MLP values (typically between -3 and 1) are mapped on the surface of the molecule by assigning values of MLP to the mesh. For every vertex of the mesh, the correspondent grid-cell is identified and the value of potential is calculated using trilinear interpolation. This process is very fast and the mesh vertex density is high enough to represent smoothly the potential spatial transition. The vertices positions and their corresponding MLP value are saved in an .obj file. This format stores the basic 3D structure (vertices and faces of the mesh) and provides two additional data fields to save further information, i.e. the texture coordinates (U and V). The V field of this file is used to store

the MLP value associated with each vertex of the mesh. The MLP values are next converted into vertex colour (the same value for each RGB channel, obtaining levels of grey). For the conversion we normalize the range of the MLP values ([-3,1]) to the range of grey scale ([0,1], and set value 0 of MLP to correspond to the value 0.5 of the grey scale). In this way the hydrophathy of the protein is visualized in Blender as levels of grey: bright areas representing hydrophobicity and dark areas hydrophilicity (Figure 3B). 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).

MLP rendering

The representation of MLP as levels of grey (i.e. values between 0 and 1) is the basis for the photorealistic visualization. The code for the representation of hydrophathy that we propose is a range of optical features that go from smooth-shiny surface (hydrophobic) to rough-dull (hydrophilic), as shown in Figure 3C.

Data elaboration for rendering is done in 3 steps (Figure 1, lower part)

1. *Creation of the first image texture.* The mesh is unwrapped to generate a texture parametrization and the per-vertex colour 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. The 2D image is also called image texture or UV map, where U and V are the texture axes.

2. *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, adding a Gaussian noise to the texture image, which produces an image with a strong noise over the black regions, gradually reduced on grey regions to reach a level of no noise on white. In the rendering process this noise is converted to bump, as explained below.

3. *Addition of specularity and roughness.* In the final rendering step, the image obtained in the first step (grey 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 represented as reflective and smooth, while the more hydrophilic ones as duller and rougher (Figure 3C).

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 [22, 23] which outputs 2 files: .pqr and .in. These files store information on the size and the 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 [24], 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, 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 local values into a grid of gradients*
3. *Selection of 'interesting' 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.

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.

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 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 \times $\text{eV}/\text{\AA}^2$). 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 the value set by the user).

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.

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 the animation the particles are generated every 5 frames and have a life-time of 20 frames. This means that the system is in steady state after the sixteenth frame (see the scheme in Figure 5). 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 so that the sequence of all images produces a representation of molecules and their features that is coherent from frame to frame.

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 the invisible field lines.

Design and Implementation

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 colours and managing various scientific programs as described (www.bioblender.net). BioBlender is distributed as Open Source software, under Creative Commons license.

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 [11], used for visualization of .pdb files (proteins, nucleic acids, other macromolecules). 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 a format (.wrl) easily read by 3D software tools.

PDB2PQR-1.6.0 – [22, 23] 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 in PDB format to PQR format. It assigns partial atomic charge to every atom in the .pdb file according to different force fields (AMBER 94, CHARMM 27 and 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 the 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) – [24] 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 Electrostatic Potential grid (.dx) calculated by the APBS plug-in. 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 [25]. 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 [26]); it contains a library of lipophilic atomic potential for every atom based on its chemical position 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, 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).

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 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 aminoacids, 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.

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 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. We implemented in Blender 2.5 an engine with a user interface adapted for biological visualization.

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 have presented 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 hydrophathy 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 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. Example of the use of these codes can be seen for a single protein in the Proteopedia page [27] and for a complex in our movie Protein Expressions – Study N3 [28].

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) 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 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 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 aminoacid, using the Kyte-Doolittle scale. This scale was elaborated almost 30 years ago [15] 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 [29] which employs the Brickman 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 negative (partial) charges: 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.

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 hands 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 plugins, by writing an appropriate visualization program using the OpenGL syntax. Using a javascript support library, SpiderGL [30], we built an interactive visualization scheme [31] which accepts as input the same meshes, curves representing the field lines and the texture images for MLP 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.

Availability and Future Directions

Project name: BioBlender

Project download page: www.scivis.ifc.cnr.it, www.bioblender.net

Operating system: Windows

Requirements: install PyMOL, python and numpy (they can be found also in the Installer folder)

We are currently developing more complete version of the software that will include:

- surface properties visualization also for sugars and lipids,
- libraries for importing and working with nucleic acids and other molecules
- Linux compatibility (now it can be run on Linux using Wine).

Acknowledgments

We thank the PDB2PQR, APBS, PyMol teams, late Warren DeLano and the Blender users and developers community for kind answers to our many questions.

Authors' contributions

RMA performed research, wrote and tested software; MC, MFZ, GM contributed programming help; MC, MCP contributed scivis.exe and BioBlender interface, respectively; TL, MZ contributed visual elaboration with Blender; MZ conceived research; RMA, MZ wrote paper

References

1. Zini, M F; Porozov, Y; Andrei, R M; Loni, T; Caudai, C; Zoppè, M (2010) BioBlender: Fast and Efficient All Atom Morphing of Proteins Using Blender Game Engine. (under review)
2. 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
3. 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
4. Protein Data Bank [www.pdb.org]
5. Humphrey, W; Dalke, A; Schulten, K (1996) VMD: visual molecular dynamics. J Mol Graph 14: 33-8, 27-8
6. Visual Molecular Dynamics [<http://www.ks.uiuc.edu/Research/vmd/>]

7. Guex, N; Peitsch, M C (1997) SWISS-MODEL and the Swiss-PdbViewer: an environment for comparative protein modeling. *Electrophoresis* 18: 2714-23
8. Swiss-PdbViewer [<http://www.expasy.org/spdbv/>]
9. Pettersen, Eric F; Goddard, Thomas D; Huang, Conrad C; Couch, Gregory S; Greenblatt, Daniel M; Meng, Elaine C; Ferrin, Thomas E (2004) UCSF Chimera--a visualization system for exploratory research and analysis. *J Comput Chem* 25: 1605-12
10. UCSF Chimera [<http://www.cgl.ucsf.edu/chimera/>]
11. DeLano, WL, The PyMOL Molecular Graphics System, 2002
12. The PyMOL Molecular Graphics System, Version 1.2r3pre, Schrödinger, LLC.
13. Holst, MJ; Baker, NA; Wang, F (2000) Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. Algorithms and examples. *J. Comput. Chem.* 21: 1319-1342
14. Rocchia, Walter; Sridharan, Sundaram; Nicholls, Anthony; Alexov, Emil; Chiabrera, Alessandro; Honig, Barry (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. *J Comput Chem* 23: 128-37
15. Kyte, J; Doolittle, R F (1982) A simple method for displaying the hydrophobic character of a protein. *J Mol Biol* 157: 105-32
16. Autodesk website [www.autodesk.com]
17. Blender website [www.blender.org]
18. Connolly, M L (1983) Solvent-accessible surfaces of proteins and nucleic acids. *Science* 221: 709-13
19. 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
20. 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
21. Testa, B; Carrupt, P A; Gaillard, P; Billois, F; Weber, P (1996) Lipophilicity in molecular modeling. *Pharm Res* 13: 335-43
22. Dolinsky, Todd J; Nielsen, Jens E; McCammon, J Andrew; Baker, Nathan A (2004) PDB2PQR: an automated pipeline for the setup of Poisson-Boltzmann electrostatics calculations. *Nucleic Acids Res* 32: W665-7
23. Dolinsky, Todd J; Czodrowski, Paul; Li, Hui; Nielsen, Jens E; Jensen, Jan H; Klebe, Gerhard; Baker, Nathan A (2007) PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. *Nucleic Acids Res* 35: W522-5
24. Baker, N A; Sept, D; Joseph, S; Holst, M J; McCammon, J A (2001) Electrostatics of nanosystems: application to microtubules and the ribosome. *Proc Natl Acad Sci U S A* 98: 10037-41
25. Python website [www.python.org]
26. pyMLP.py [<http://code.google.com/p/pymlp/source/browse/trunk/pyMLP.py>]
27. Calmodulin motion on Proteopedia [http://proteopedia.org/wiki/index.php/Calmodulin_in_motion]
28. SciVis website [<http://www.scivis.ifc.cnr.it/index.php/videos>]

29. Steinkellner, Georg; Rader, Robert; Thallinger, Gerhard G; Kratky, Christoph; Gruber, Karl (2009) VASCo: computation and visualization of annotated protein surface contacts. BMC Bioinformatics 10: 32
30. 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
31. 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

Figure Legends

Figure 1 – 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 specular effects.

Figure 2 – BioBlender interface

The interface is structured in 6 panels: *select .pdb file* – upload from user defined path, or access directly from PDB.org specifying the 4 letter code); *import* – select various parameters, including covalent/Van der Waals radius, include/exclude Hydrogens and others); *view* – visualization in 3D working space, activation of Game Engine (see [1]); *MLP visualization* – Parameters for MLP, **a**: choice of formula and grid spacing; **b**: contrast and brightness control; *EP visualization* – parameters for EP, **c** and **d**: calculation and representation, respectively; *output* – export of .pdb files and rendered frames.

Figure 3 – 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 gradient of MLP distribution over the molecular surface.

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 – Particles generation and representation for moving proteins

Field lines are imported as curves every 5 frames (0.2 seconds). Particles have a lifetime of 20 frames. After the sixteenth frame (0.6 seconds) the system is in ready-state (square).

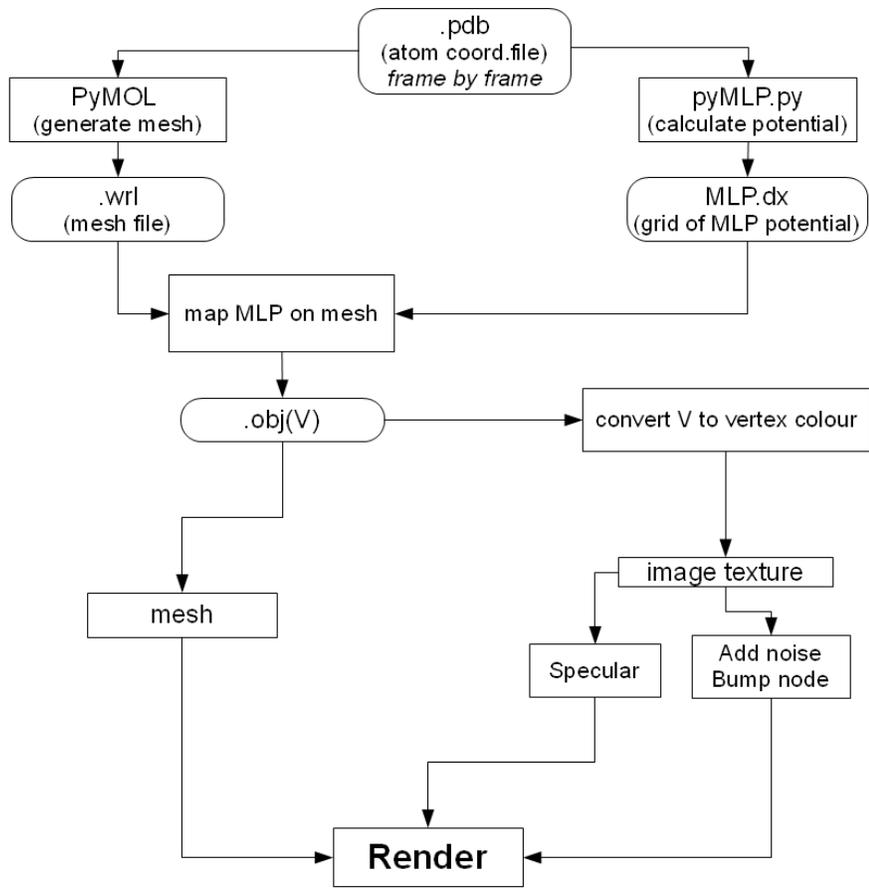


Figure 1

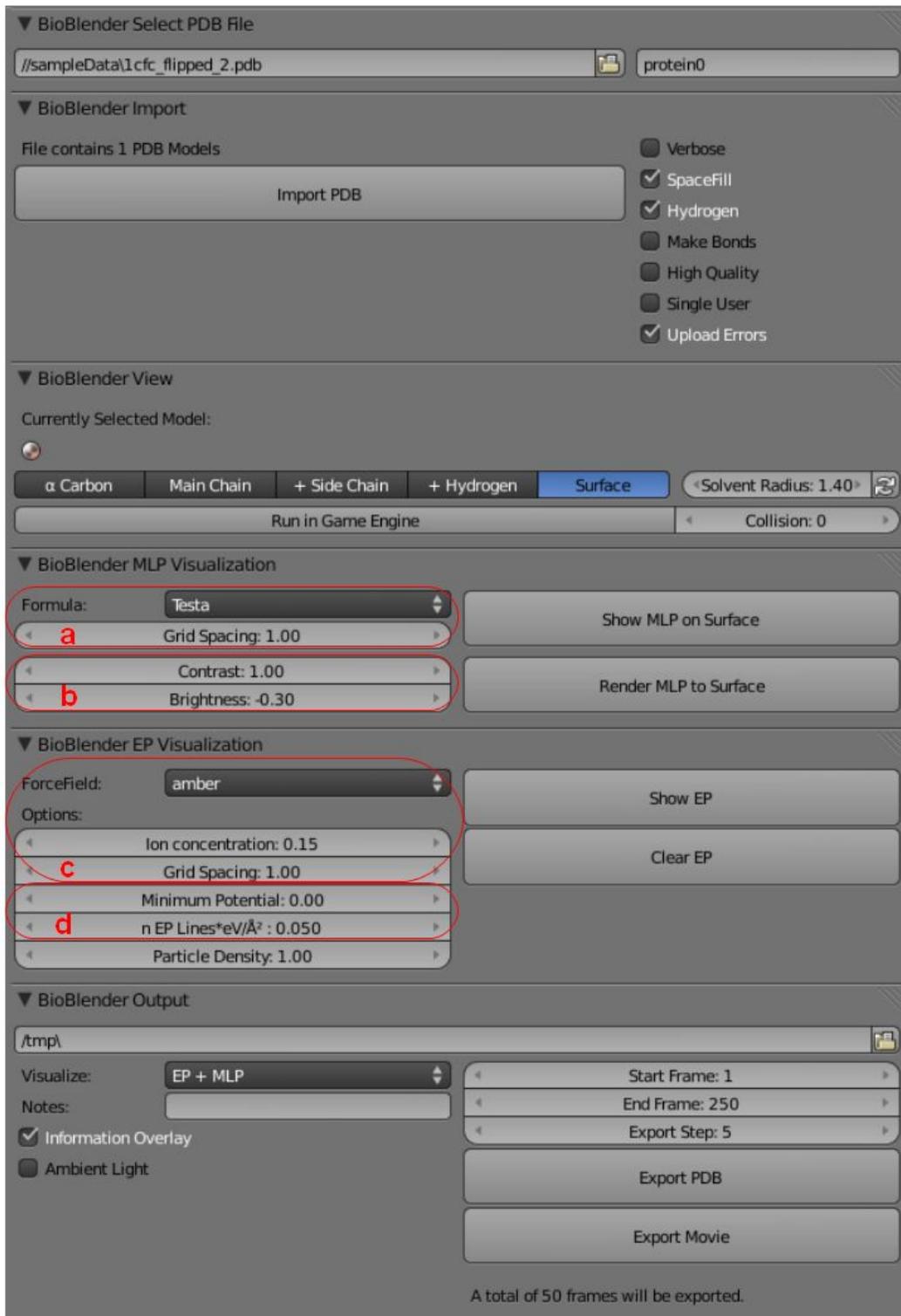


Figure 2

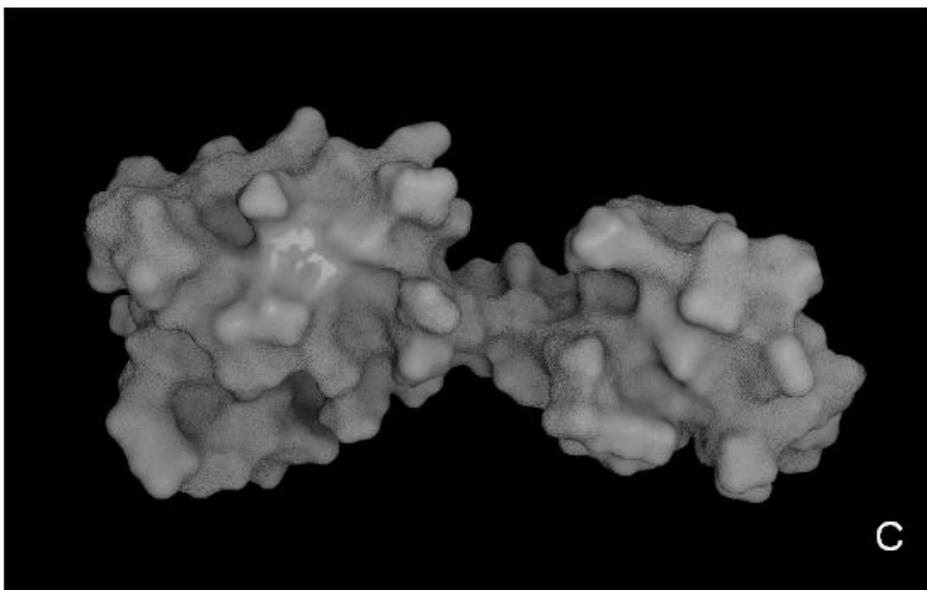
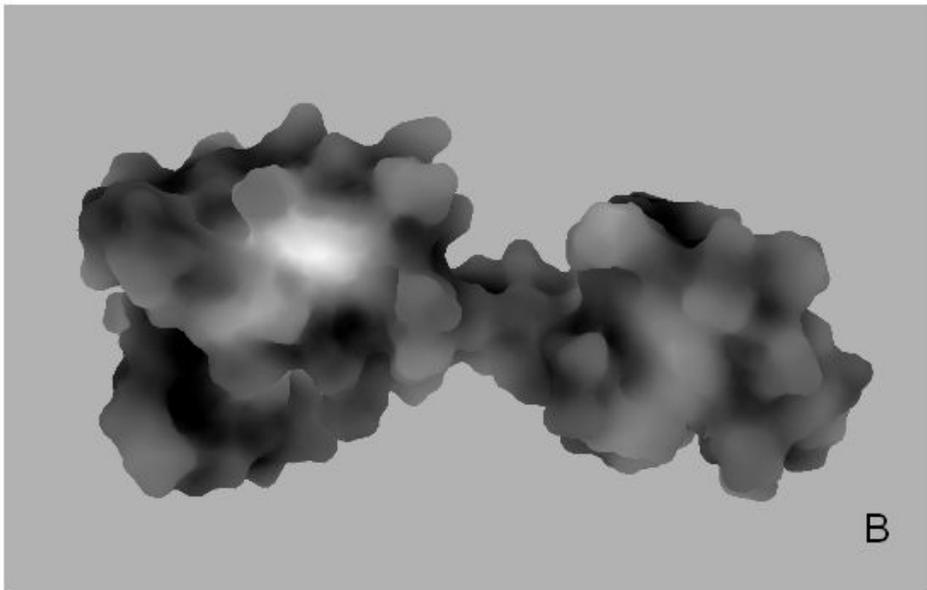
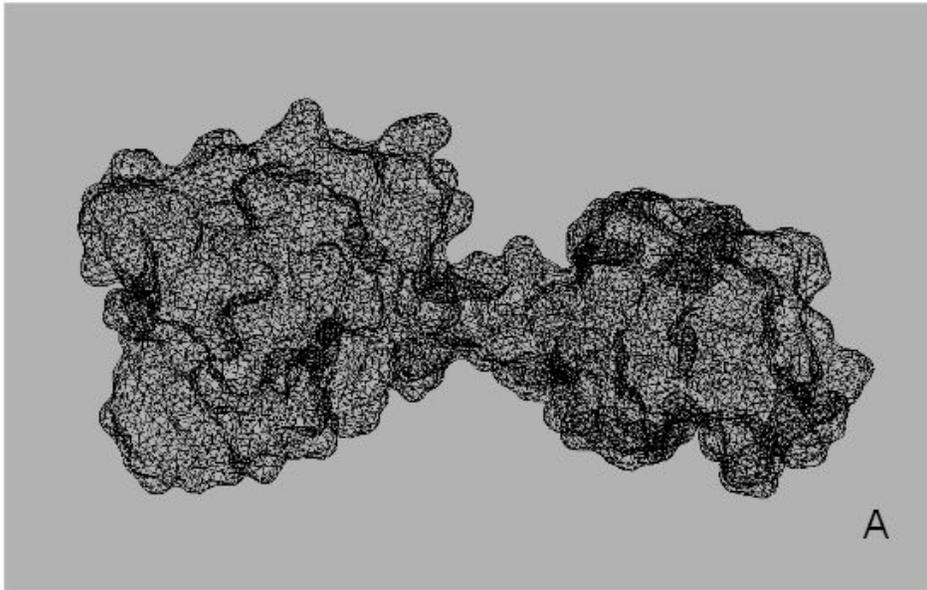


Figure 3

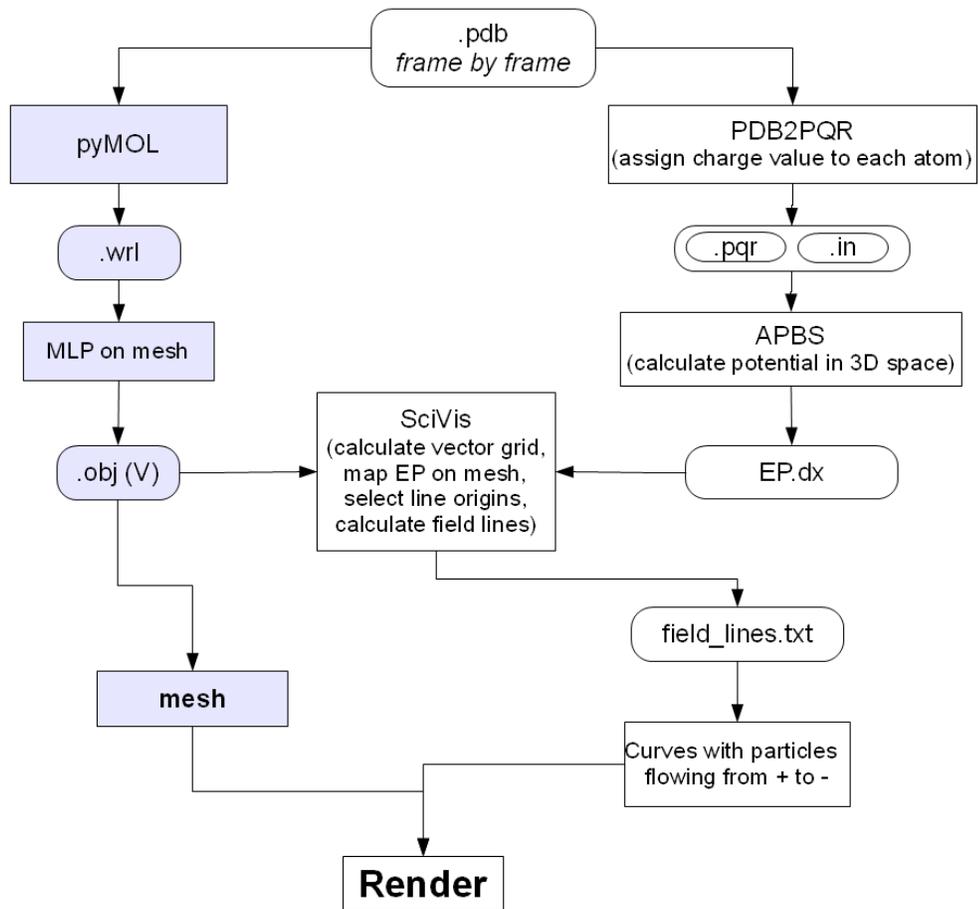


Figure 4

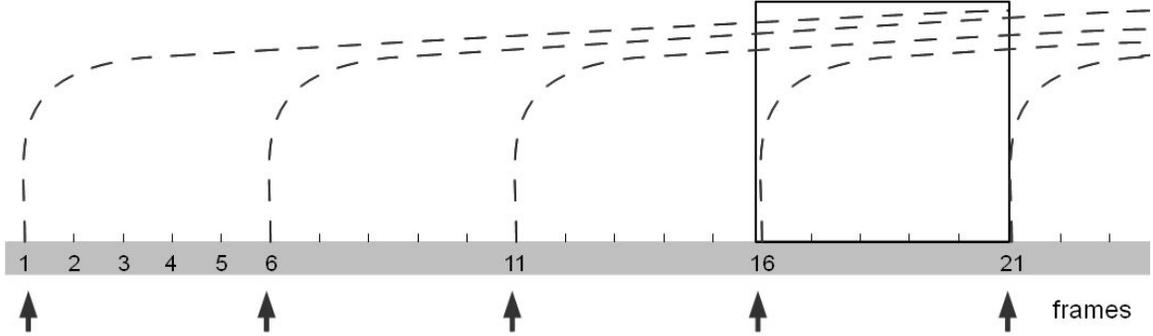


Figure 5