Blender for biology
The making of Protein Expressions – Study N 2

Monica Zoppè[1*], Raluca Andrei[1,4], Stefano Cianchetta[1], Maria Francesca Zini[1], Tiziana Loni[2], Claudia Caudai[1], Marco Callieri[3]
[1]Scientific Visualization Unit, LTGM. Institute of Clinical Physiology, CNR Pisa. [2]Big Bang Solutions, Polo Tecnologico di Navacchio, Pisa. [3] Institute of Information Sciences and Technologies, CNR Pisa. [4] Scuola Normale Superiore, Pisa, Italy.

[*]Corresponding Author: mzoppe@ifc.cnr.it www.scivis.ifc.cnr.it

ABSTRACT
We presented in 2008 the initial steps of our project, to use Blender for scientific representation and molecular animation of life at the cellular level.
This report will focus on some developments in two major aspects: the use of Blender Game Engine in conjunction with other scientific programs to elaborate the movement of proteins and the development of a visual code based on texturing and particles special effects to represent the physical and chemical properties that confer to molecules an appearance that reveal useful information.
We also describe some details of the construction of specific scenes.

1 INTRODUCTION

The amount of information collected in the fields of cellular, molecular and structural biology and in protein dynamics is huge, and expanding at high speed.
In recent years several groups have been using part of this information to deliver a graphical view of the processes that occur at the nanometer scale of proteins, in both still and animated forms (1). This effort has also been facilitated by the availability of 3D computer graphics elaboration packages, such as Maya, 3DStudio Max (2) and Blender (3) which become progressively more versatile and sophisticated.
We have engaged in an effort to contribute to the field of molecular animations using Blender, a full-featured, integrated, 3D animation application that provides a complete workbench for all steps of video production. Blender (3) is open source, freely distributed, multi-platform, interoperable and supported; these features make it an ideal instrument for both animators and biologists, as well as other scientists, willing to 'show' their work in animated details.
During the 2008 Blender Conference (4), and in the relative printable report (5) we presented the general outline of the project and some of its early steps. Readers unfamiliar with concepts of cells, proteins, membranes and other biological subjects are referred to our previous article (5) where these subjects are presented for a readership of animators and Blender users in particular.
In this article we will discuss the details of two aspects: the motion of proteins and their surface rendering. We also describe some of the technical challenges of rendering and compositing some complex scenes in the making of the short movie Protein Expressions – Study N 2 (6). A description in biological terms of the movie was made available as 'Explanatory notes', released together with the movie itself (7).

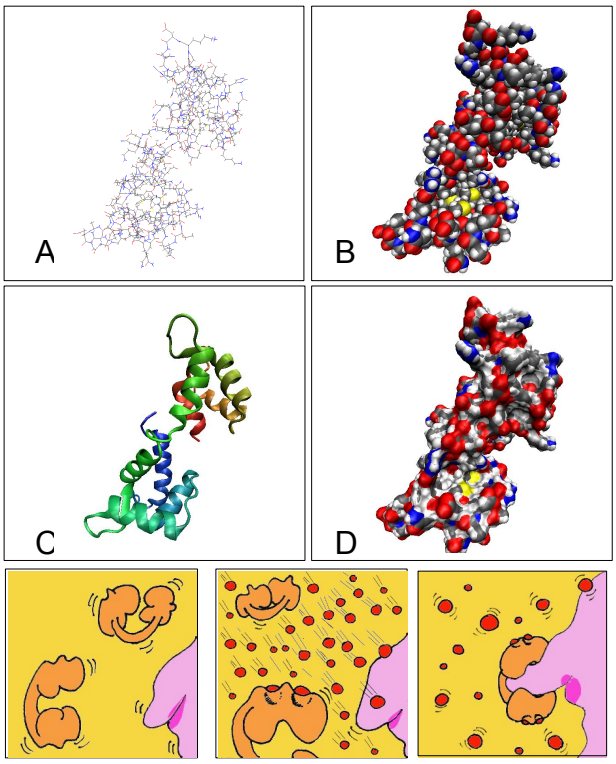

Figure 1. Calmodulin.
Different representations of the molecule, all 2262 atoms (including Hydrogen) with their connections. Images obtained with VMD (8).
***A**. All atoms, represented through their chemical bonds. Colors indicate different atoms (C dark grey, O red, N blue, S yellow, H white);*
***B**. All atoms, represented as Van der Waals surfaces, colors as in A;*
***C**. Structure representation. Helices are color coded from N to C (red to blue) terminal of the protein;*
***D**. Surface representation: colors as in A*
***Bottom row**. Cartoon style representation of the activity of CaM, idling, being associated with Ca ions, and bound to MLCK (see text). The actual resemblance of the real molecule to the drawing can be compared to the resemblance of mothers to their preschool kids' drawings.*

2 RESULTS

2.1 Moving proteins

Proteins are molecules composed of an organized ensemble of atoms, each one with specific connections to its neighbours, a.k.a. chemical bonds (9). Proteins move both in terms of vibrations (minor changes of atom coordinates around a defined position) and as the result of a larger scale, slower motion that changes the overall shape (and often activity) of the entire molecule. The precise distinction between these two types of motion is not clear-cut; however it is possible to ignore very small atomic motions and concentrate on the second type of motion, which is what we do.

The protein that we have used to elaborate the system in Blender is Calmodulin (CaM, 10), a very well studied small protein, composed of 1166 atoms (excluding Hydrogen) in 148 aminoacids, as illustrated schematically in Fig. 1.

CaM is organized in two globular heads (domains), each of which can bind two Calcium ions (Ca^{++}). When the local concentration of Ca^{++} rises, 4 ions fit into specific positions in the CaM, which is now called CaCaM. CaCaM thus activated acquires the ability to bind to other proteins (we use Myosin Light Chain Kinase, MLCK, as our specific case study) and set in motion a new chain of events.

From the scientific literature and the Protein Data Bank (PDB, 11) we retrieved a collection of different conformation of CaM in its native form (ApoCaM, 1cfc), a partial conformation of the transition state between ApoCaM and CaCaM (1g7e), another collection of conformations of CaCaM (1x02) and a single structure of CaCaM associated with a small piece of MLCK (1cdl).

The 25 conformations of ApoCaM in 1cfc can be assimilated to single shots of a sequence (which contains many more), except that they are given in random order; therefore the first task is to organize them so that, by interpolating between different poses, a smooth movement can be produced.

In order to obtain such smooth sequence, we developed the strategy outlined in Fig. 2. We started with the most different conformations (highest value of root mean square deviation, RMSD) in the collection, and used our importer to load them in Blender at 1000 frames distance.

Atoms were imported as spheres, made rigid body actors with collision radius similar to their Van der Waals radius, and their chemical bonds were set as rigid body joints. The transition between the two conformations is calculated by the Game Engine (GE), which considers joints, collisions and a slight bounce factor when collisions occur. As described in the flow chart, 20 intermediate conformations were retrieved from Blender and compared to the remaining poses in the original .pdb file. Those conformations that were found within a mean distance (RMSD) of 2 Å or less were inserted in the sequence and the procedure was repeated until no more conformations could be introduced. In this way we obtained a sequence of 15 poses that could be animated smoothly. Some details of this procedure were presented as a poster at the International Society of Computational Biology meeting 2009 (12), and will be published shortly (Zini *et al.*, manuscript in preparation).

Precise adjustment of the conformations was refined using Swiss-PDBviewer (13), a program that implements the GROMOS force field to adjust atomic positions according to chemical and physical forces.

The scripts used to import atomic files into Blender (reading standard .pdb format) open an interface that

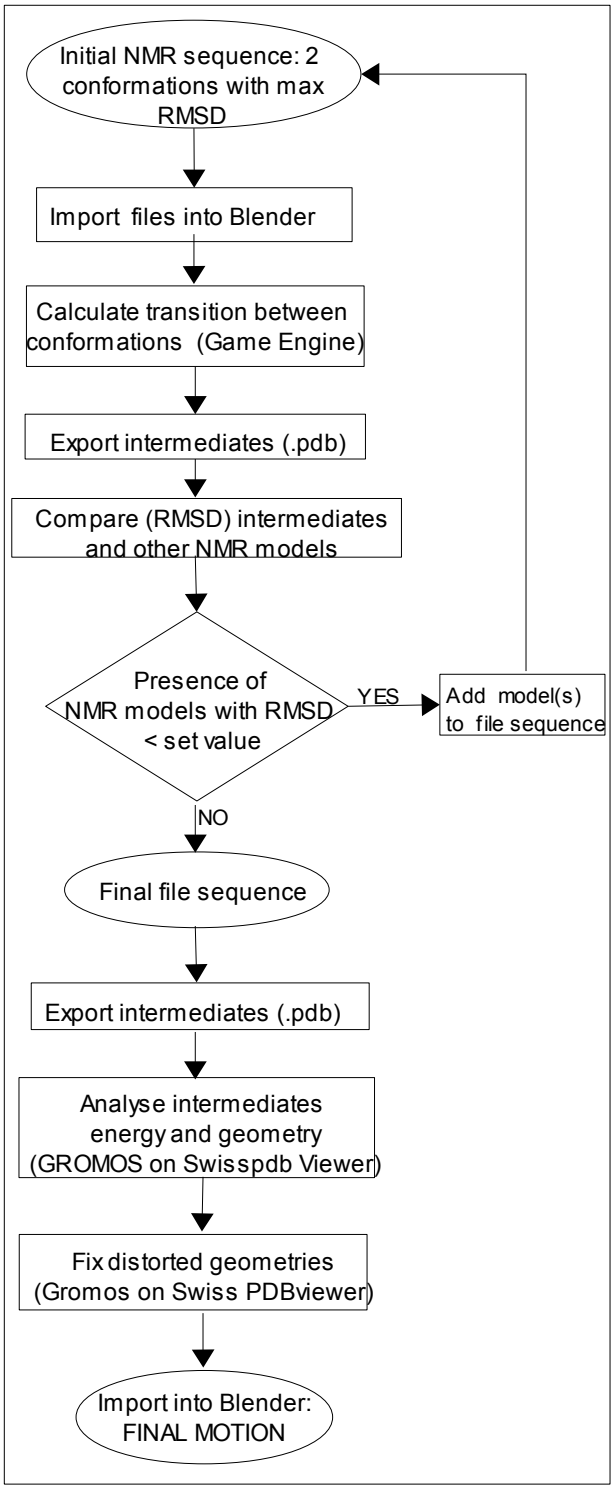

Figure 2. Flow for motion elaboration on a set of conformers.

First an RMSD table is built and the two most distant conformations are selected. These are elaborated in Blender, and intermediate conformations are exported and compared (by RMSD) to all other models in the original collection. The models whose RMSD is lower than 2 Å are inserted in the sequence and the procedure is repeated until no more conformations can be inserted in the sequence.

If necessary the procedure is repeated starting with another pair of conformations; in this way a network of transition paths connecting all conformations in a collection can be identified.

allows the user to choose how many conformations to import, which ones, in which order, the distance between them (in frames), and the selection of atoms to be imported (Alpha Carbons, Main chain atoms, Main chain and side chains or all atoms including Hydrogen). A similar interface is used for the export procedure, where the user is prompted for details of saving options.
Using the procedure outlined above, we have obtained a 'navigation map' that permits to visit all the conformations in the original .pdb file, defining a few paths compatible with the physical and chemical conditions present in animal cells.
The example of Calmodulin movements can be seen in a dedicated page in Proteopedia (a wiki dedicated to proteins, 14) and was incorporated in to the movie Protein Expressions (6). The material used can be downloaded from our website.

While the transition between conformations that are quite close (RMSD < 5 Å) easily results in a seamless transition, major movements are more demanding and Blender GE cannot solve them using the simple procedure described above. It becomes therefore necessary to explore other possibilities to obtain a movement using different tools.
One possibility is the introduction of more links (such as H bonds), that stabilize naturally occurring rigid structures. This would imply the import of all Hydrogens (about doubling the number of spheres), the development of an algorithm capable of recognizing the H bonds when present, and the introduction of a different set of rules, as H bonds behave different from chemical covalent bonds.
Another possibility is the introduction of additional structural features to make rigid some parts of the protein such as helices and sheets. Again, this would be a manual process, requiring users to know the details of protein structural features, and it would be difficult to handle them while the protein changes its conformation, sometimes quite dramatically.
Finally, it might be possible to use a different system to handle atoms and bonds.

2.2 Rendering proteins

Proteins are, as we have remarked, invisible, as their size is beyond the limit of visibility by humans. Several devices have been implemented by scientists to outreach from these limits, from microscopes to modern techniques of X-ray and single molecule imaging; yet the possibilities for seeing objects of few nm size are still not straightforward. Our work is to be intended as the representation of some of the features of these invisible objects: namely the shape, the Molecular Lipophilic Potential (MLP) and the Electrostatic Potential (EP). These three properties are the major features that determine protein behavior.

2.2a Shape-Mesh

The Shape of a protein can be defined as the volume occupied by the atoms that compose the protein. This can be calculated through one of several systems: we have chosen the Solvent Accessible Surface Area (SASA), which means the limit of the space that can be reached by the molecules of water in which the protein is typically dissolved (other solvents can be used).
To calculate this surface and to obtain a mesh, we utilize the scientific program Chimera (15) or PyMOL (16), which can read the file .pdb (atomic coordinates) and export the mesh as a .wrl file. Both programs allow users to choose the level of details of the mesh generated: the greater the detail, the larger the file (see Fig. 3). Scripts that automatically extract pseudo-.pdb files from Blender (obtained through the process of motion generation described above), send them to PyMOL and re-import the corresponding meshes have been written in Python.

2.2b Molecular Lipophilic Potential

The hydrophilic (and its converse, lipophilic) potential of a substance is a measure of its capability of interaction with water. The word contains the root *hydro-* (water) or *lipo-* (grease) and *-philic* (friendly), to indicate the tendency of a substance of becoming dissolved in water, or, conversely in lipid such as oil. As an example,

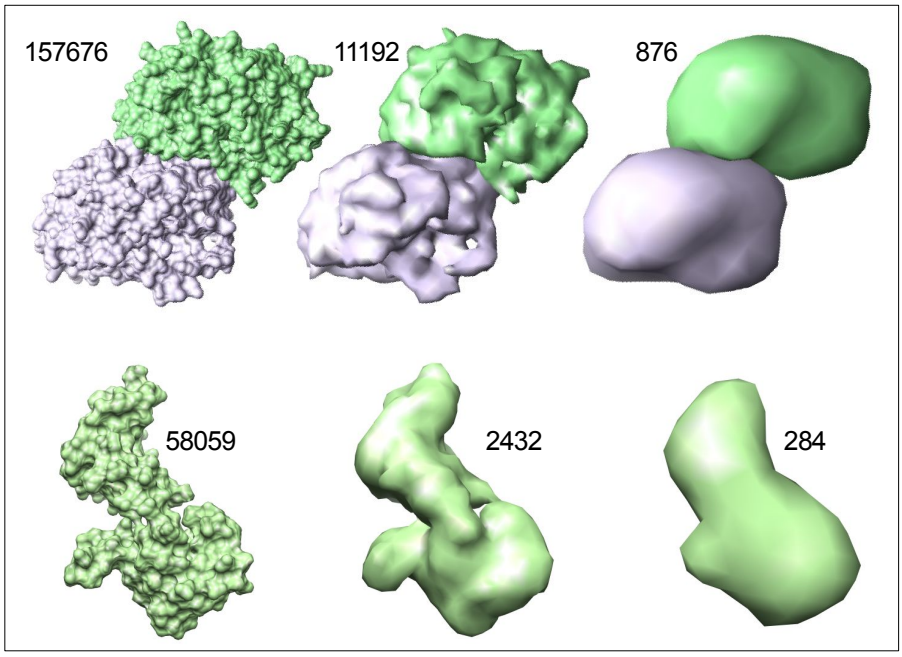

Figure 3. Meshes of proteins as obtained from Chimera.

Protein meshes were calculated and imported at high, medium and low resolution. In the top row is a Tubulin dimer, composed of 867 aminoacids, bottom row Calmodulin (148 aa).

The two proteins are not at the same scale.

Numbers beside each image indicate the N of polygons of the mesh.

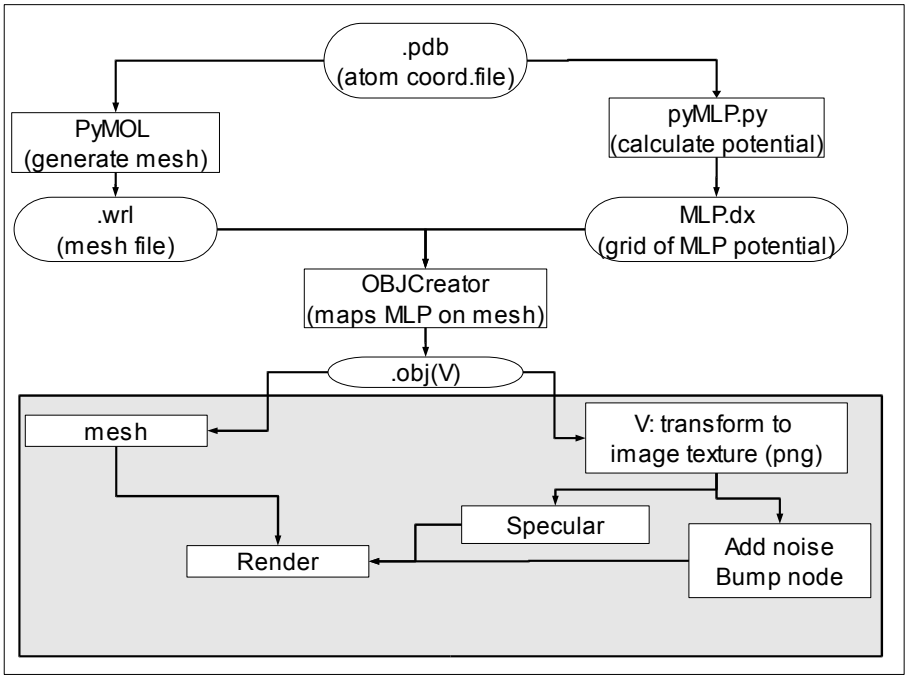

Figure 4. Procedure for calculating and rendering the Molecular Lipophilic Potential of proteins.

On the left the flow to generate and import molecular surface mesh; on the right the steps for MLP calculation and rendering.

The steps within the grey box are elaborated in Blender.

See text for further details.

sugar and salts are typically very hydrophilic, while oil, butter etc. are lipophilic/hydrophobic. In the case of proteins, the same molecule can have patches of the surface with different affinity for water and oils. This property is the result of the property of each atomic group exposed on the surface, and can be calculated according to one of few formulae.

Figure 4 describes the flow of information necessary to calculate and represent MLP.

To calculate the lipophilic potential, starting from the .pdb file (the list of atoms composing the protein with their space coordinates) we have chosen to apply the Testa formula (17), which considers a lipophilic value attributed to every single atom (derived from a library), its links and neighbours, and is applied by the script pyMLP.

This script outputs a .dx file listing the distribution of MLP values in a cubic grid containing the protein. These values are mapped on the surface mesh by OBJCreator, a program created in house, that combines mesh data and MLP data to write another .obj file. This file lists all vertices defining the mesh (x,y and z coordinates) and the numerical value (in the 'v' column) of MLP for that vertex.

Once imported in Blender, the v (MLP) value, already a numerical attribution to each vertex, is converted into vertex color with same value for RGB, generating a grey scale texture; the mesh is unwrapped and the texture map is saved as a .png image. The .png is the basis for manipulating the grey scale with a ramp, it is converted into a bump map through a bump node using a noise texture and it is enhanced with another ramp to create a specular map. All these operations will result in a surface with a white, smooth and reflective appearance where the surface is very lipophilic, and a darker, rough and dull material in the more hydrophilic areas.

As the protein moves, the relations between its atoms change slightly, and the MLP value of the surface changes accordingly. For this reason the procedure has to be reiterated at every frame of the animation. For long animations this is done automatically, using yet another python script. An example of a single frame is shown in Fig. 5.

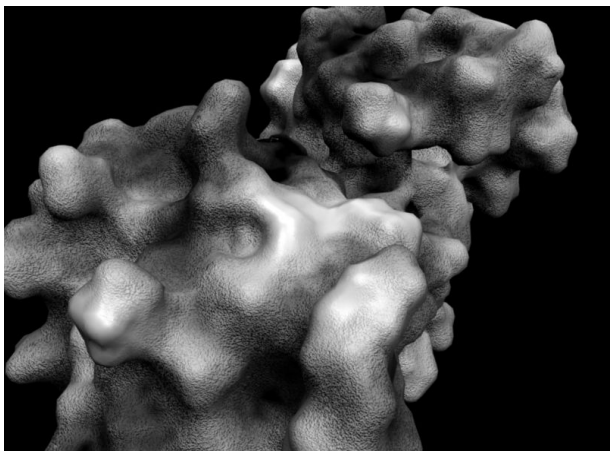

Figure 5. Calmodulin portrait.

The rendering is taken with a single spot light that casts no shadows. All grades of grey are a representation of the degree of lipophilicity presented by Calmodulin surface.

2.2c Electric Potential

For a definition and the importance of EP in the life of proteins, please see our previous Report (5). Also for the EP the basis is the .pdb file, as delineated in the flow of Figure 6.

The PDB2PQR program (18) assigns to every atom a specific (partial) charge; next the APBS (19) module calculates the distribution of potentials in the grid space by solving the Poisson-Boltzmann equation, and generates a .dx file similar in nature to the one described above for MLP.

The home-made program SciVis uses this .dx file and the mesh (.obj) to calculate curves that represent the field lines, allowing the user to choose the minimal value of the EP from which curves are generated. The user can also decide how many curves to extract: whichever their amount, they will be distributed proportionally to the charge at every vertex of the mesh.

Curves thus generated are written in a .txt file, which is then imported in Blender. The beginning (+ end) of every curve is an emitter point of a particle system that

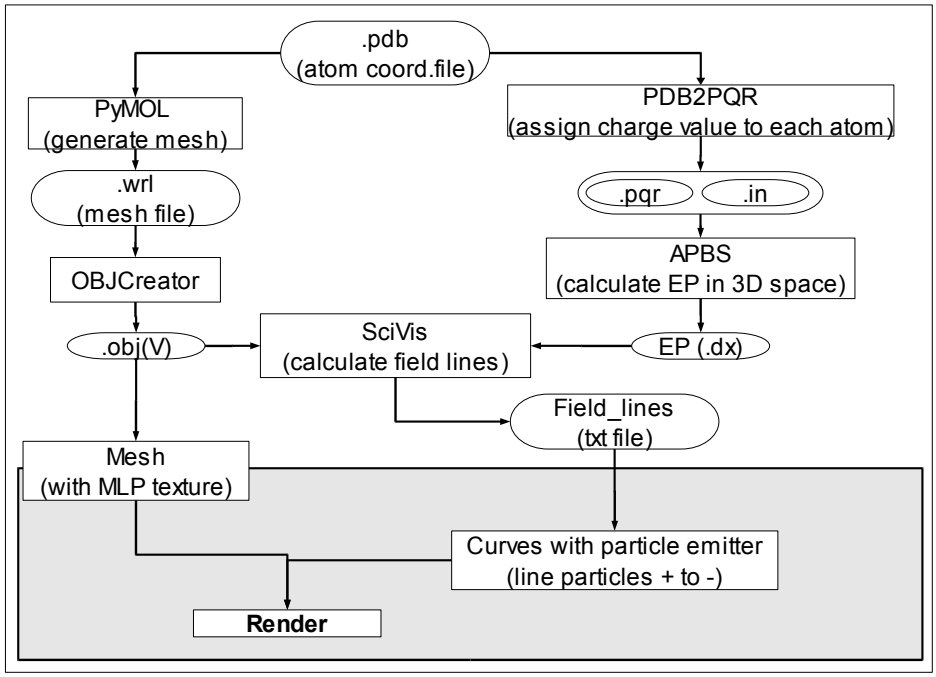

Figure 6 Procedure for calculating and rendering the Electrostatic potential lines generated by proteins.

On the left the flow to generate and import molecular surface mesh and the MLP; on the right the steps for EP calculation and rendering.

The steps within the grey box are elaborated in Blender.

See text for further details.

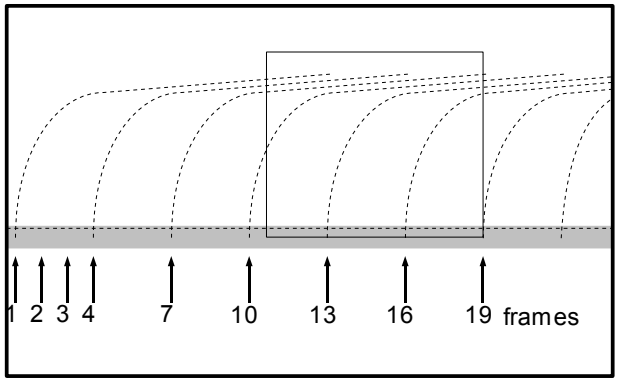

Figure 7. Particle emission time sequence for the representation of EP field lines.

emits 1000 line particles that will travel along the curves from positive to negative. A new particle system is created every 3 frames and has a lifetime of 12 frames, so that the movie is practically in steady state from the tenth frame onward, as schematically shown in Figure 7.

2.3 Selected scenes

2.3a Microtubules and Kinesin-8 (Kip3)

The scene (Figure 8) shows a bundle of microtubules (the long 'cables') in the cytoplasm (the material inside the cells), together with a number of floating proteins (NFκB, Diaphanous and Aurora) and one of the many microtubule associated motor proteins, specifically Kinesin-8 (formerly known as Kip3).

In Blender, the microtubules are built starting from the mesh of a single tubulin dimer (pdb file 1tub) imported as mesh with a procedural texture similar to its MLP texture. Two of these are arranged according to their relationship in nature. The displacement is used in an array, repeated 13 times, leading to the building of a spiral ring. This ring is then repeated with a second array modifier along a line path, thus giving rise to the final tubule. Each dimer mesh, as first imported from the scientific program (see above) consists of 48 thousands polygons, making it impossible to build the entire microtubule without overloading Blender. For this reason, several versions were made, with decreasing number of polygons, to be used in the background. The high poly version of the tubule is “anchored” to the camera and moved by 8 B.U. steps (the pitch of the helix, i.e. the distance from one dimer to the corresponding one in the next spire of the helix) to follow the camera, via script, so that the well defined model is always near the camera.

Kinesin-8 (20) is a protein that belongs to the family of kinesins, motor proteins that travel along the microtubules in a specific direction (those that travel in the opposite directions are quite different, and are called dyneins). Kinesin-8 is among the fast runners, possibly used by the cellular biochemical system to keep under control the extent of the microtubule network. We modelled it on the basis of one of its close homologues (Kinesin-1, pdb 3kin), as there are no atomic coordinate data specific to Kinesin-8. Studies of kinesin dynamics (21) were also the basis for Kip movement, which was modelled using a system of bones to control the different parts of the protein. The two feet are connected by a neck linker, which functions as a pivot for the rotation of one foot at a time. The movement was prepared as a walk cycle followed by a translation along a path over the microtubule.

The scene also shows the crowding of proteins in the cell. According to scientific indications, cellular space is even more crowded than this, although it might be better organized. However, while we need to pass the message that there is little if any empty space, and that proteins are so crowded that they can't move without bumping into each other all the time, if we were to fill the entire space with proteins, we would not be able to see anything beyond the first two or three proteins in front of us. The scene contains about 4000 floating protein meshes of 3 different kinds. These were modelled as usual, with meshes of 1870 to 2657 polygons each. To

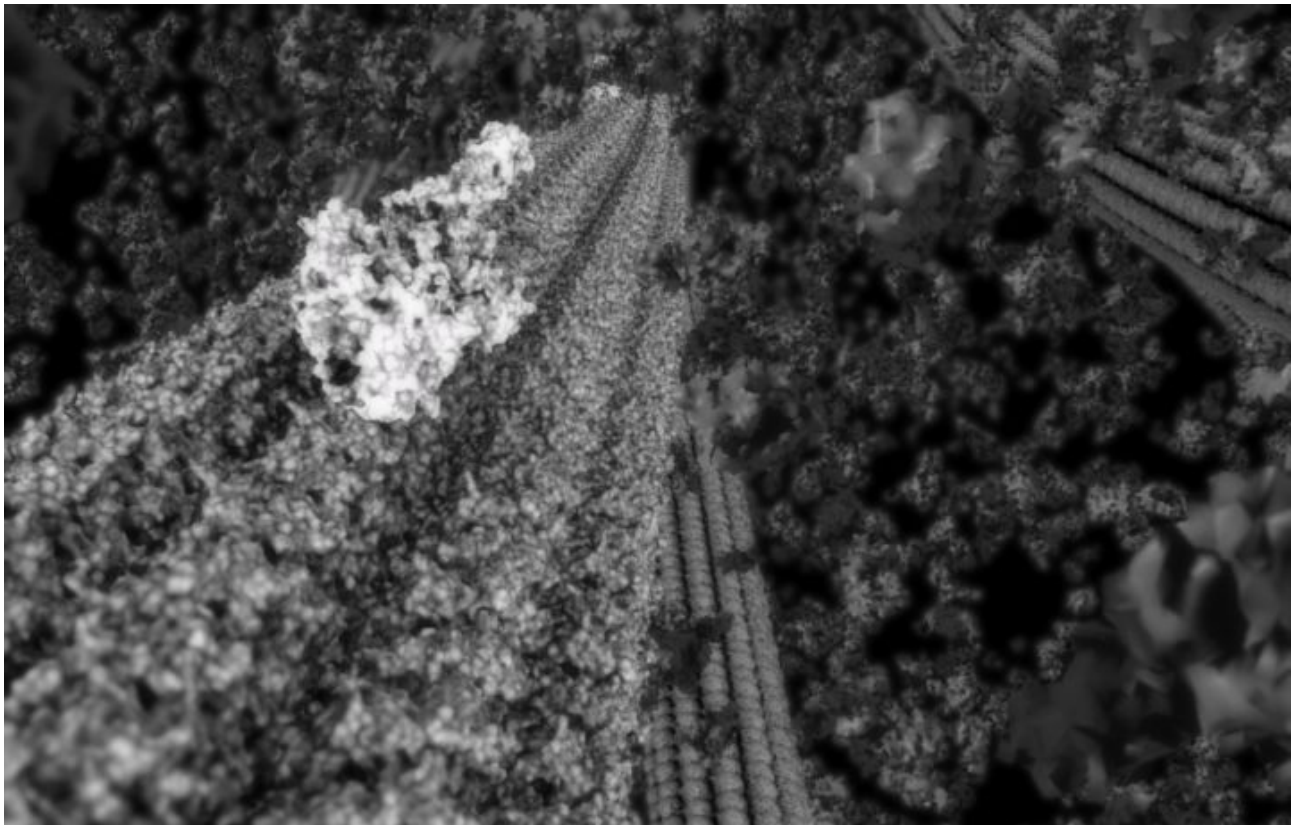

Figure 8. The microtubule scene. *Shot from the movie: the long 'cables' are microtubules surrounded by a large number of other proteins; Please note that the cellular environment is much more crowded than this, although possibly better organized.*

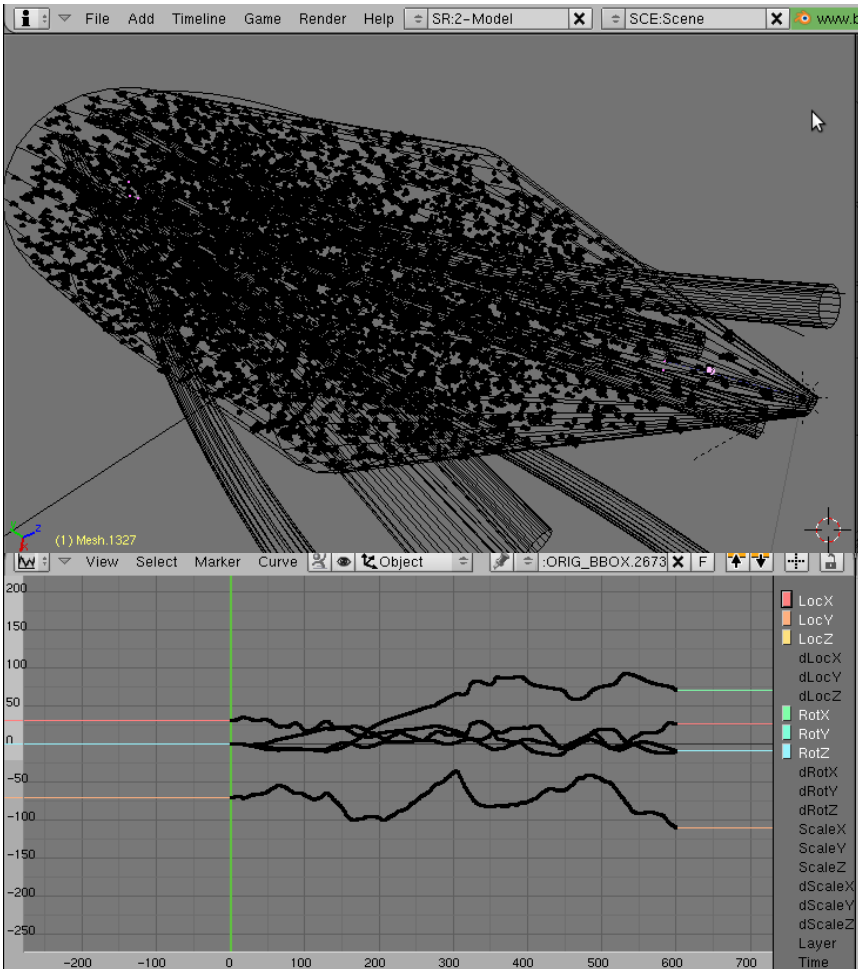

Figure 9. The microtubule scene.

Top. *Wireframe scene showing the box containing the spheres.*

Bottom. *A typical ipo curve for one sphere, as recorded after GE evaluation. The roughness of the ipo curve indicates the numerous objects and bounces that this sphere encountered.*

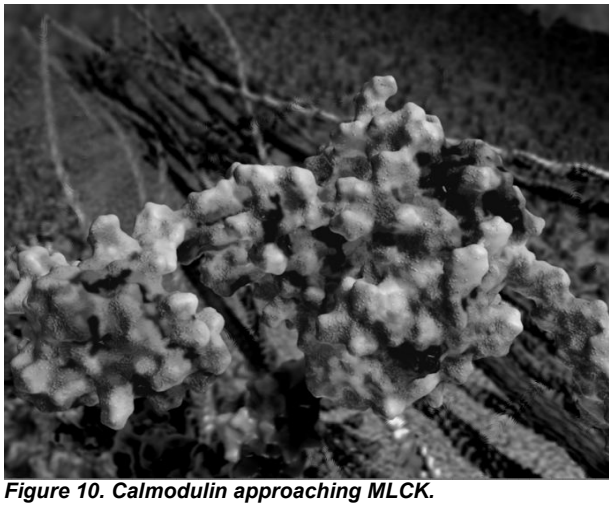

Figure 10. Calmodulin approaching MLCK.
Shot from the movie.

fill the space around the microtubule bundle, we built in the Blender scene (see Fig. 9, top) a volume containing the tubular cylinders, the camera equipped with a collision cone, and 4000 spheres, made rigid body actors for evaluation by the GE. A python script was assigned to execute during GE runtime, that attributed to each sphere a force generated randomly at every frame, producing a disordered movement. The script also attributed an initial torque force for rotation to each sphere. Thus the GE evaluated collisions, bounces, and the forces in play. The simulation was recorded as IPO curves, which were finally attributed to the various proteins.
For rendering, the scene was divided in several parts, each one rendered separately using Multilayer .exr format: the close up microtubules, other microtubules, Kinesin-8 (for this we used speed vector for the motion blur effect), and the other proteins. The final composition also included effects such as DOF, fog and the dark background.

2.3b MLCK scene

A more challenging compositing effort was made in the composition of the scene, towards the end, in which we see MLCK being bound by CaCaM (Figure 10). The scene set is the periphery of the cell, very close to the inner leaflet of the cellular membrane. Here we see, already assembled, the membrane, with all the inner parts of proteins that span its thickness, the contractile ring, MLCK and Calmodulin. The contractile ring is responsible for the separation of one cell into two daughter cells, at the end of the complex process of cellular division (called mitosis). It is comprised of filaments of actin (the clear, twisted cables) and myosin (the darker rigid structures), which will slide along one another during ring contraction. MLCK is an enzyme (Myosin Light Chain Kinase) that by adding a phosphate group to a specific spot on the Myosin complex, enables it to start the sliding motion. MLCK is in turn activated by Calcium-Calmodulin as we see in the scene.
The compositing window is shown in Figure 11.
Panel A. The membrane basis is rendered in 2 layers: the membrane itself, with the shadows of the embedded proteins and the proteins, combined in Z. The actin and myosin bundle is color adjusted and, again, combined in Z with the previous image.
Panel B. For the farthest background, an image was created using a procedural texture (noise), adjusted for color and brightness, blurred, and placed in the extreme Z layer.
Panel C. The resulting combined plate was passed through a Blur node controlled along Z, obtaining a DOF effect.
Panel D. MLCK is a very large and complex protein. It is

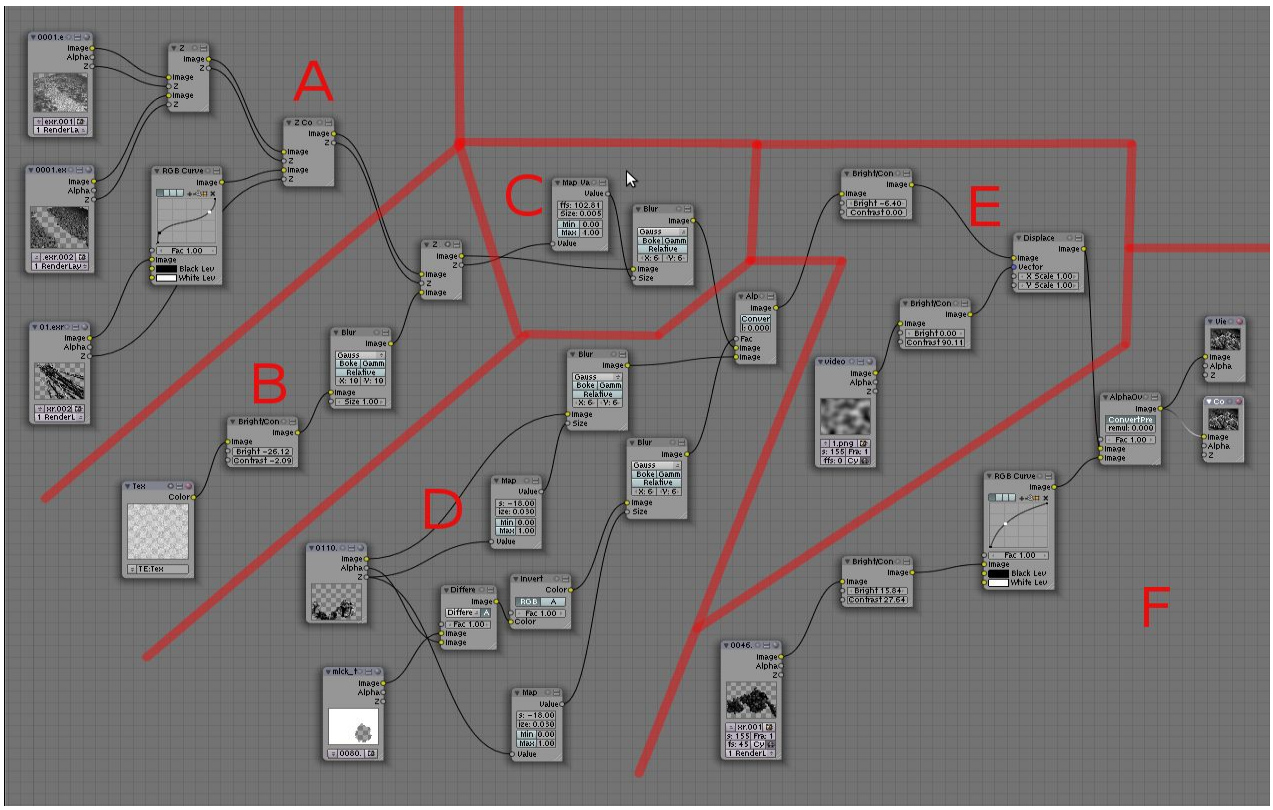

Figure 11. The MLCK scene. *Compositing panel.*

composed of an enzymatic part (the 'head', which we have studied at the atomic level) and a long flexible 'tail', whose complete structure and dynamics are not yet elucidated. In the movie we show the tail part moving behind the head; however the motion was obtained by a system of bones with the sole purpose of conveying an idea of its flexibility and dimension at the beginning of the scene. At the end, when Calmodulin activates the enzyme (both proteins are modelled at atomic level) the tail is seen as part of the background. Because the first modelling included the head, when the atomic model was composited, the previous head had to be hidden. This was obtained by cutting a shape subtracted via alpha channel, and compositing the tail onto the background, again with alpha over node.
Panel E. Before adding the MLCK and Calmodulin in the foreground, we introduced a mild distortion effect driven by an animated texture, recorded on a plate using a procedural Stucci.
Panel F. Finally the major actors, MLCK and Calmodulin, were added, after minor adjustment of color/brightness, using once again alpha over node.

4 ACKNOWLEDGEMENTS

We are indebted to the Regione Toscana that supported the entire work presented here.
We also thank the Blender community for patient help, through the forums (www.kino3d.com/forum and www.blendernation.com/).
Finally, we acknowledge the Blender Foundation, that gave us the opportunity to present our work, and our movie, at the annual Conference in Amsterdam 2009.

5 REFERENCES

1. Inner life, http://multimedia.mcb.harvard.edu/anim_innerlife.html and Malaria Lifecycle, http://www.wehi.edu.au/wehi-tv/movies/malaria.html
2. Autodesk Maya and 3D studio Max, accessible from Autodesk website.
3. Blender (http://www.blender.org/features-gallery)
4. Blend Conference 2008, http://river-valley.tv/media/conferences/blender2008/0102-Monica_Zoppe/
5. Zoppè, M., Porozov, Y., Andrei, R., Cianchetta, S., Zini, M.F., Loni, T., Caudai, C., Callieri, M. (2008) *Using Blender for molecular animation and scientific representation*. Proc. of the Blender Conference, Amsterdam. Download
6. Protein Expressions. Study N 2. View here or download here
7. Protein Expressions – Study N 2. Explanatory notes. Download
8. Humphrey, W., Dalke, A. and Schulten, K. *VMD - Visual Molecular Dynamics*, J. Molec. Graphics, 1996, vol. 14, pp. 33-38. http://www.ks.uiuc.edu/Research/vmd/
9. Page on proteins from Wikipedia
10. Johnson, C.K. (2006) *Calmodulin, conformational states, and calcium signaling. A single-molecule perspective.* Biochemistry 45:14233.
11. 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.
12. Zini M.F., Andrei R., Loni T., Cianchetta S., Callieri M., Zoppè M. (2009) *Blender for biology: Game Engine for molecular animation and special effects for chemical and physical behavior*. Poster presented at the International Society of Computational Biology meeting, Stockholm. Download
13. Swiss Guex, N. and Peitsch, M.C. (1997) *SWISS-MODEL and the Swiss-PdbViewer: An environment for comparative protein modeling.* Electrophoresis 18:2714. Access here
14. Calmodulin dedicated page on Proteopedia.
15. Pettersen E.F. *et al.* (2004) *UCSF Chimera--a visualization system for exploratory research and analysis.* J Comput Chem. 13:1605.
16. DeLano, W.L. (2002) *The PyMOL Molecular Graphics System* DeLano Scientific, Palo Alto, CA, USA. http://www.pymol.org
17. Testa, B, Carrupt, P A, Gaillard, P, Billois, F, Weber, P (1996) *Lipophilicity in molecular modeling.* Pharm Res 13:335.
18. 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 Res 35:W522 .
19. 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.
20. Varga, V., Leduc, C., Bormuth, V., Diez, S., Howard, J. (2009) *Kinesin-8 motors act cooperatively to mediate length-dependent microtubule depolymerization.* Cell 138:1174.
21. Asbury, C.L., Fehr, A.N., Block, S.M. (2003) *Kinesin moves by an asymmetric hand-over-hand mechanism.* Science 302:2130