Scientific Visualization Unit. IFC – CNR
www.scivis.ifc.cnr.it April 2010

PROTEIN EXPRESSIONS

Study N. 3D

The video **PROTEIN EXPRESSIONS - Study N. 3D** is the third (and last) re-elaboration of the movie on which we have been developing our studies in the last two years. Produced as 3D stereo vision and in high definition (HD, 1080 x 1920) it has been a major technical challenge, beside the incorporation of a new scene in which we elaborate on protein-protein interaction, a very important aspect of research in cellular and molecular biology.
The following text covers a large part of the description prepared on the occasion of the previous movies, with some additional information, and including the new scenes introduced.
More technical and/or scientific details can be found on our website and has been submitted for publications to the scientific literature.

What follows is the description of some of the technical and scientific aspects of the video, scene by scene. We are aware of the fact that the terminology used can be difficult for non-professional biologists or graphic experts. Hopefully the interested reader will take occasion to get some insights from many excellent cell biology and 3D computer graphics textbooks.

1. Red Blood Cells with a White Blood Cell (T cell)

The set is a fine capillary. It is populated by red blood cells, platelets and a few white blood cells. The scene was built in Maya. The capillary is a cylinder on whose surface (inside normals) is mapped a microscopic photograph (manually made seamless) of epithelial cells grown to confluence in a Petri dish (photo kindly provided by G. Ratto, Instituto of Neuroscience, CNR) . The number of cells in the lumen of the capillary is lower than it is in reality: they are so crowded that it would be impossible to see in between them. All objects (red cells, white and platelets) are present in dimensional and numerical proportion.
Red cells are all instances of a simple polygonal sphere deformed to become biconcave; these instances are set as rigid body objects and sent through the vein at constant speed. The Blender Game Engine evaluates collision between cells and with the capillary walls.
The white cell, is a mesh modelled by extrusions and moved by means of both a skeletal system and an animated displacement texture.

2. Flight over cell surface.

The membrane was obtained imposing properties derived from membrane biochemistry to several particle systems, in Maya: single lipids, groups of lipids (rafts) and groups of proteins distributed on the surface. This was converted to a grey scale texture and attributed (in Blender) to the displacement node of the surface using Blender SuperTexture Node and Texture Plug-in.
The objects flying with us over the surface are modelled on the largest objects (besides cells and platelets) found in human plasma: High and Low Density Lipo-proteins (also known as good and bad cholesterol), and IgMs, large multiprotein complexes that look like 'snow-flakes'.
They are added to the scene as polygonal meshes in 3 different depth layers (each one containing few thousands objects). Their motion was calculated using the Game Engine with spherical bounding boxes actors, with initial random impulses attributed via Python script.

3. The membrane.

Based on the texture above, the rafts were constructed as a Game Engine structure in Blender, with proteins loosely connected with rigid body joints. Proteins in the raft were modelled (at low resolution) on the basis of pdb files: CD4 (1WIO), Thy-1/CD59 (2UX2), TCR (1KCG), CD3 trimer (1XIW, 1SY6, 1SY6), Toll Like Receptor (2Z7X), ICAM-1 (1IAM for domains 1 and 2, and 1P53 for domains 3, 4 and 5). The details of the surface of each protein are imposed as texture map, derived from a high resolution model of each protein.

The group of proteins was set to follow the centre of a raft. This motion was recorded and provided the 2D surface sliding of proteins on the membrane. The Z (vertical) motion to follow the wavy surface was obtained by dropping the proteins at each frame on the distorted plane. An additional rotatory motion was attributed to each protein with a random generator. Each protein also has a skeleton that governs randomly the inner motion. Also oligosaccharide chains are modelled on the basis of pdb files and moved with an internal skeleton by another random motion generator, faster and with wider angles relative to proteins.
The Potassium channel is modelled according to the structure in pdb file 1LNQ (side chains added with Swiss PDBviewer). It is shown releasing K ions (small spheres) in bursts of few hundreds each: however, the rate of ion release by real channels can reach 10 millions of ions/second.

4. Calmodulin.

Once inside the cell (after falling into the K channel), we meet Calmodulin (CaM).
The movement of CaM is the most scientifically relevant part of the entire video. CaM is a small protein (148 aminoacids, 1166 atoms, excluding Hydrogen) structurally organized in two parts (heads, or globular domains) connected by a flexible linker. The two parts can associate with Calcium ions, and when this happens the lobes open and become reactive towards other proteins, thus transducing the signal delivered by the Calcium wave.

We first meet apoCaM (Calmodulin without Calcium). Protein movements were obtained elaborating on an NMR structure collection (1CFC) using the Game Engine of Blender according to a flow as follows:
- all atoms of the protein (excluding hydrogens) are imported in Blender (we wrote specific scripts to import and export pdb files) in two different positions, chosen among the 25 present in 1CFC.pdb file as the farthest apart in RMSD, separated by X frames (say 500). Atoms are connected with rigid body joints (corresponding to chemical bonds).
- Blender Game Engine interpolates between conformations, evaluating joints and collisions.
- intermediate conformations so generated are exported as new .pdb files, and compared (RMSD) with other conformations in the original NMR collection. Any NMR conformation closer than 2 Å is imposed as intermediate keyframe in the correspondent time between the initial positions, and the process is repeated.
- when no more conformations can be added to a group, the sequence of .pdb files is exported and adjusted using the GROMOS force field implemented in Swiss-PDB Viewer.

With few cycles of this procedure we have generated a continuous motion that covers a geometrical distance of 21 Å in steps of <1 Å, 'visiting' a dozen conformations experimentally derived. These are set in Blender at 50 frames distance, and again interpolated by the GE. Finally, each frame of the global movement is exported and sent to biophysical programs that introduce some necessary minor adjustments to the position of atoms and introduces Hydrogens.

For the visualization of chemical and physical properties, we have developed a visual code applied to the surface of the molecule without making use of colors.
Each frame from the animation (.pdb file), is converted to .pqr (using a program that introduces proper charge to every atom) and elaborated with a series of libraries and other programs and scripts (see List of Programs) that calculate electrostatic and lipophilic potential (EP and MLP respectively) values. Using a home-made program, these fields are mapped on the protein surface (Connolly surface, obtained as a .wrl file from PyMOL) and associated to all vertices of the mesh for MLP or transformed in a series of lines with a direction (for EP). The values of MLP are expressed as scale of smooth-shiny-clear to bumpy-dull-dark using a series of scripts and progrms. The property of EP is represented as particles running along the filed lines, from positive to negative.

5. Microtubules.

After observing Calmodulin for a while, we change our gaze towards a bundle of microtubules. These are built with an array of tubulin dimers (pdb file 1TUB) displaced by the appropriate distance to form a 13 filaments ring. The ring is repeated (another array) to form the pipe.
Running over the central microtubule, in the direction of the positive end, we see Kinesin 8, one of the many proteins that populate the cytoskeletal structures in the cell. Its shape is modelled on the pdb 3KIN, and its motion was obtained by a system of bones (moved with IK and using stride bone) based on the motion of Kinesin, as described in the scientific literature.
Other proteins in the scene are NFkB (1LKN), Diaphanous (1Z2C) and Aurora (2BFX), rendered according to our procedure and moved through a similar mechanism as the proteins over the cell surface (scene 2). We travel for a short while along the microtubules and then we take a hyper-leap to the periphery of the cell, where we meet a

completely different environment.

6. MLCK and the plasma membrane.

With the hyper-leap, we cover the huge distance of several micrometers and reach the vicinity of the membrane, seen (this time) form inside. The place is still very crowded, protein coming and going everywhere, but juxtaposed to the membrane we see a different structure, made of thin (actin) and thick (myosin) filaments interconnected. The structure is part of the contractile ring, a transient ensemble that is built at the time of cellular division, when the cellular content has been duplicated and the cell divides into two daughters. The contractile ring is assembled at the equatorial perimeter, associated with the membrane: at the appropriate time it will start to contract until the cell will finally split. Contraction is due to the relative motion of actin and myosin filaments, powered by ATP and driven by a number of accessory proteins, including MLCK (Myosin Light Chain Kinase).

We can distinguish MLCK, associated with the actin/myosin bundle, as a protein in part attached to one of the myosin heads, and with a long moving tail. The very long 'tail', has not been completely characterized at atomic detail. We have assembled several 'pieces' and have modelled its movements using a system of bones and random motion. The tail of MLCK shares with many other proteins the property of being 'disordered' or 'unstructured', a term that indicates that its shape is not fixed, due to the inherent flexibility of its components. We show random motions of the tail, just to give an idea of this feature.

MLCK, being bound to the contractile ring, has the primary role to activate the myosin movement by adding a Phosphate group to one of its subunits. All movements are coordinated and they are mediated by Calcium and Calmodulin: the signal arrives in the form of Calcium ions, which bind and activate Calmodulin, which in turn can go and activate MLCK (as well as other components, not visible here).

While we get close to the contractile ring, we can see a set of Calcium wave.

7. ApoCaM to CaCaM transition.

Once again, we focus on Calmodulin, which we see transiting form its unbound (ApoCaM) form to the conformation that assumes when 4 Calcium ions are bound (CaCaM). The transition was obtained by morphing the protein from Conformation 21 of 1CFC (ApoCaM) to 11 of 1X02, which collects 20 conformations of CaCaM. The elaboration followed a process similar to the one described for the transition between different conformations of the same form, except that here the change is much more pronounced, and required a larger number of refinement and adjustment steps. When bound to Ca, the protein exposes a patch of hydrophobic surface at the interior of each globular head. This patch facilitates binding of CaCaM to its partner proteins, one of which is MLCK.

8. Binding of CaCaM to MLCK and MLCK activation.

Both proteins are shown in full atomic details, the head of MLCK modelled on the basis of pdb file 1CDL, that describes the conformation of CaCaM associated to a small peptide of MLCK, and 1KOA, containing detailed conformation of another protein that phosphorylates myosin (twitchin kinase). The homology between twitchin and MLCK is sufficient to build an accurate model of MLCK, which we have done using several scientific programs.

Calmodulin is seen moving and bumping around MLCK, moved by random vibration. When the reciprocal interacting surfaces of the two proteins face each other in a productive way, the two become attached, and a further movement of CaCaM closes on a peptide of MLCK, inducing a new conformational transition in the latter and activating its enzymatic activity which is the phosphorylation of myosin. Now myosin can start the contraction, sliding its head in a coordinated way relative to actin.

9. The contractile ring.

The scene is the same as before: during cytokinesis, when the contractile ring pulls the membrane that will eventually divide the cell in two.

The underlying membrane is a plane converted to cloth, contracting along the middle line through pinning to a cylinder scaling in time, and covered with 50.000 particles (hair) that follow the plane geometry to simulate presence of many proteins spanning in the membrane. To create the actin filaments, we started from the high resolution crystal data of globular actin (1ATN), and positioned the monomers as described in the PDB file 1ALM (crystal structures). This model, containing 5 monomers, was imported in the Blender scene via Chimera. In Blender an array modifier was created, to multiply the single actin model by values derived by carefully adjusting the offset and rotation settings of the monomers in the short filament crystal. The array-based actin filament can be elongated in a snap.

Myosin filaments were created using of PDB files 1I84 and 1SR6, (two extreme conformations of the heavy chain motor domain) and 1NKN (coiled-coil tail structure from two intertwined myosins), which were loaded into Chimera, together with 1ALM (5 aligned actin monomers). The latter file also contains a myosin head domain bound

to the actin filament. This gave us the opportunity to align the myosin head group to the actin filament. The other myosin structures were aligned to the 1ALM file and the surface models were created and exported as described. The models also include the Myosin Essential Light Chain and Regulatory Light Chain molecules, wrapped around the neck domain of the Myosin Heavy Chain molecule. Once imported into Blender, the myosin motor domain movement was animated using an armature for the pre-power stroke model. The armature was connected and subsequently posed to closely match the myosin surface model in the power stroke position. Care was taken to not only animate the large displacement of the head group, but also the more subtle displacements of smaller subdomains of the proteins, through an additional armature into the bumps of the protein surface model. The resulting rigged model could now be moved from the pre-power stroke conformation to the power stroke conformation by moving from one pose of the armature to the other and back. The final movement of the myosin molecule is composed of five partly overlapping steps:
1- power stroke (first 20 frames); the motion of the myosin head while it is attached to an actin filament; this motion causes the displacement of the actin filament.
2- release and lift (frame 20-40). The myosin head is released from the actin filament and the head is lifted away from the actin.
3- prepare for power stroke (frame 29-55). The myosin head is rotated around the neck to reposition it for the next power stroke.
4- drop (frame 41-66). The myosin head moves towards the actin filament.
5- connect (frame 66-67). The myosin again grabs an actin filament; to visualize this, the motion of the bones stops for just one frame.
The tail domains of two myosin heavy chains twist around each other to form a coiled coil. The 1NKN PDB file was used to model such a coiled coil. This model, however, only contains 89 amino acids per strand, whereas the tail domain of a myosin heavy chain molecule extends over hundreds of amino acids. Therefore, the model was imported into Blender and multiple copies of the small model were carefully aligned and connected manually until the correct tail length was reached. Myosin is active as bipolar filaments, which are composed of 10-20 pairs of myosin molecules. To mimic this, a bundle of myosin tails was created by joining a number of tail models together. Twenty myosin head models were placed pairwise at the two ends of this filament model. The conformational movement of the head groups was subsequently shifted with respect to each other to create a more realistic, shuffled animation. Finally, the acceleration of the movement by Calcium waves that lead to MLCK activation had to be animated. For this, each armature object (which drive the movement of the myosin head groups) was given a so-called time-ipo. This allows for acceleration or deceleration of the movement of the head groups. As all the head groups were given the same time-ipo, the speed of movement can be globally changed easily.

OTHER NOTES

Time Considerations. The speed at which biological activities are performed is such that nothing could be followed by human eyes. In this video we make no attempt to follow timing considerations.

Music and Sound. Are molecules silent? Or do they produce sound? Just as the scenes we show are, in reality, invisible to the human eye, they produce no sound audible to the human hear. The music accompanying the video was created by Antonio Dell'Aquila, a friend of the lab, a talented artist and musician who interpreted the images and the biological information according to his taste and art.

Why no colors? Is inner life all black and white?
Since we describe objects that are invisible (smaller than the wavelength of light, and of colored light), the idea of 'real color' cannot be applied. What we show is a *representation* of molecules, and specifically the representation of their shape, their molecular lipophilic potential (MLP) and their electrostatic potential (EP). We have chosen to avoid the use of colors, as they are not necessary at this moment, favouring instead a photorealistic tactile impression, which can be easily grasped by any viewer without the necessity of direct explanations or of a decoding legend. This also leaves the use of color space available for the representation of other chemical and physical features, such as pH, redox potential, osmotic pressure, temperature and other features that characterize the different cellular environments, and for which we have no direct and intuitive visual reference.

PRINCIPAL AUTHORS:
Monica Zoppè Scientist, IFC – CNR, Pisa. *Corresponding author*
Raluca Andrei PhD student, Lab of Molecular Biology, Scuola Normale Superiore and IFC – CNR, Pisa

Stefano Cianchetta	3D Artist, IFC – CNR, Pisa
Tiziana Loni	3D Artist, Big Bang Solutions, Navacchio (Pisa)
Maria Francesca Zini	Scientist, IFC – CNR, Pisa
Giuseppe Maraziti	Programmer, Big Bang Solutions, Navacchio (Pisa)
Yuri Porozov	Scientist, IFC – CNR, Pisa
Leonard Bosgraaf	Dept. of Biochemistry, Univ. of Groeningen, Netherland
Marco Callieri	Scientist, Visual Computing Lab, ISTI – CNR, Pisa
Ilaria Carlone	Scientist, IFC – CNR, Pisa
Claudia Caudai	Scientist, IFC – CNR, Pisa
Maria Antonietta Psacali	Scientist, IFC – CNR, Pisa
Antonio Dell'Aquila	Musician, Pisa

FUNDING
Regione Toscana, Programma POR 3
Scuola Normale Superiore, Pisa, Italy
Consiglio Nazionale delle Ricerche (CNR of Italy)

PROGRAMS USED
Graphics and image processing:

- Blender www.blender.org
- Maya Autodesk http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=7635018
- The Gimp www.gimp.org
- Djv_view http://djv.sourceforge.net/djv_view.html
- ImageMagick www.imagemagick.org
- FarmerJoe http://blender.formworks.co.nz

Scientific

- VMD http://www.ks.uiuc.edu/Research/vmd/
- Swiss-PDB Viewer (Gromos43B1) http://spdbv.vital-it.ch/
- Chimera http://www.cgl.ucsf.edu/chimera/
- Reduce - MolProbity http://molprobity.biochem.duke.edu/
- PDB2PQR http://pdb2pqr.sourceforge.net/
- PyMLP.py http://code.google.com/p/pymlp/
- APBS: http://sourceforge.net/projects/apbs
- PyMOL http://www.pymol.org/

Home made scripts and programs (available in open format from our website)

OBJCreator – software application developed in ANSI C/C++. It maps MLP values on the mesh and exports an .obj file. It takes as inputs mesh data (.wrl) created by PyMOL and the values of MLP (.dx), created by pyMLP.py. For every vertex of the mesh, the correspondent grid-cell is identified and the local potential is calculated by trilinear interpolation. The .obj file stores information of the position of each vertex of the mesh and its correspondent MLP value.

MLP.py – a Python script to be 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 to vertex colors (levels of gray) assigning a color to each vertex. The 3D surface is then unwrapped to obtain a UV texture parametrization. Finally, a texture map is built by baking the computed per-vertex color values in the texture image.

texture.py – a Python script used within Blender, that builds a basic material for the 3D mesh imported into the scene and assigns to this material the texture images generated by MLP.py.

SciVis.exe – a software application written in C++ used to calculate the field lines and to export them in an ASCII file to be imported in Blender. This tool uses the 3D surface (.obj) generated by OBJCreator and the EP grid (.dx) calculated by APBS. The computation of the field lines involves: EP values mapping on the 3D surface, gradient field calculation, automatic selection of areas with high values of EP and computation of the corresponding field lines using the gradient field.

import_curves.py – a Python script used in Blender to read the ASCII file generated by scivis.exe; it creates curves (spline curves, defined by control points) in the Blender scene and associates a particle emitter to the positive end of every curve.

cycle.sh – a shell script for Linux that reads the .pdb files and execute external programs and scripts (PyMOL, PDB2PQR, APBS, pyMLP.py and OBJCreator), saving the two .dx files for EP and MLP and the .obj file with the

MLP values.

Render.py – another Python script used in Blender during the final step of rendering. For each frame it selects the corresponding mesh (imported by MLP.py), it assigns the appropriate image texture and renders the scene, according to the scene settings (camera, lights etc.). It finally saves every rendered image as a .png file.

Other References

Protein Data Bank http://www.pdb.org

Antonio Dell'Aquila (music) http://www.antoniodellaquila.com/

We have used wherever possible open source programs. All our scripts and programs will be packaged in user friendly form, and made available in open format. Some are already available for download from our website.
For more information, or comments, please email us mzoppe -at- ifc . cnr . it

Visit our website to stay updated with the latest news www.scivis.ifc.cnr.it