Two-Level Approach to Efficient Visualization of Protein Dynamics

Ove Daae Lampe, Ivan Viola *Member, IEEE Computer Society*, Nathalie Reuter and Helwig Hauser *Member, IEEE*

Abstract— Proteins are highly flexible and large amplitude deformations of their structure, also called slow dynamics, are often decisive to their function. We present a two-level rendering approach that enables visualization of slow dynamics of large protein assemblies. Our approach is aligned with a hierarchical model of large scale molecules. Instead of constantly updating positions of large amounts of atoms, we update the position and rotation of residues, i.e., higher level building blocks of a protein. Residues are represented by one vertex only indicating its position and additional information defining the rotation. The atoms in the residues are generated on-the-fly on the GPU, exploiting the new graphics hardware geometry shader capabilities. Moreover, we represent the atoms by billboards instead of tessellated spheres. Our representation is then significantly faster and pixel precise. We demonstrate the usefulness of our new approach in the context of our collaborative bioinformatics project.

Index Terms—Molecular visualization, hardware acceleration, protein dynamics.

1 Introduction

Proteins are often considered to be static objects. However, proteins are highly dynamic and their dynamics are often the key to their function [2]. For example, some proteins have an *open* and a *close* form and understanding the transition between both is crucial to be able to design efficient drugs [17, 28]. Of particular interest are the large amplitude movements of complex protein assemblies. Molecular dynamics simulations (MD) are widely used to study the conformational changes of protein structures along time, but very long MD simulations of molecular systems containing hundreds of thousands of atoms remain challenging. Normal modes analysis (NMA) with coarse grained models is much less computer demanding and appears to be a better approach; it has been successfully applied to predict the collective large amplitude motions of, for example, nanoengines or virus capsids [15].

Proteins are made of amino acids, referred to as *residues*. In this work, we exploit the fact that protein structures can be divided into two hierarchically separate levels. Although there are only 20 different standard amino acids, which contain between 7 and 24 atoms, also including the hydrogen atoms [16], some proteins need other reference constructs to make sense, such as water or other more complex molecules. To support such constructs, we need a model that allows mixing of two level structures, like proteins and one level structures such as water. All amino acids have 6 atoms in common (forming the so called *backbone*) and they differ by side chains of different form and complexity. Refer to Fig. 6 for more detail with respect to the structure of proteins. Amino acids are linked by amide bonds to form proteic chains. These chains often contain several hundreds of amino acids. A protein can be made of one or several chains. In living organisms, several proteins can interact with each other and aggregate to form a complex assembly whose function can differ from the originals. In fact, the biological structures we have to look at, for a better understanding of crucial biological processes, often contain a large number of atoms. With all the progresses made in our post-genomic era, the number of atoms in biomedical structures under investigation steadily increases.

One way of gaining insights into the dynamics of proteins is to analyze the geometric information arising from the NMA calculations and visualize the results using two-dimensional plots or numeric tables. A more intuitive approach which also enables a much better understanding of the spatial relationships is dynamic 3D visualization. Immersive three-dimensional visualization of protein dynamics, for example, has the potential to serve as a very good exchange platform between experimental and computational biologists. Experimental biologists are often not familiar with the methods employed by computational biologists (e.g., NMA [8]), which makes it difficult for them to analyze computational results and connect them with the protein structure and function. Interactive 3D visualization is more intuitive and results in significant benefits in this case.

There are several different visual representations in use for the visual depiction of protein structures. Each visual representation serves its own purpose. Some of them depict high-level information giving visual prominence to certain characteristics (e.g., the helical twist of a part of the backbone). Such a visual depiction is referred to as *cartoon* rendering among the domain scientists. A more traditional but still often preferred way of displaying the structure dynamics in proteins is showing the atoms (all but hydrogens) and investigating how they interact with each other. Rendering every atom in the protein with connections to the other atoms in the structure is known as *ball-stick* representation and rendering atoms as intersecting spheres is called *space-fill* representation (see Figure 1). When visualizing ever larger and complex structures, additional techniques like depth cueing, and depth aware silhouettes [27] are vital to extract structural information without loosing smaller details.

Interactive rendering of very large proteins (hundreds of thousands of atoms) using the *ball-stick* representation is computationally very demanding, even using the latest graphics hardware. In our project, this task gets even more challenging since protein dynamics should also be studied in an immersive environment using stereoscopic rendering on a high-resolution projection wall. The complex geometry of the protein representation is associated with dynamic changes in the atomic positions of the proteins. Sending new geometry for each frame will immediately drop the rendering performance to unacceptable frame rates because of constantly updated complex geometry calculations and I/O GPU bandwidth [18, 1]. Especially in the case of an immersive environment it is crucial that the performance does not drop under 24 fps to avoid nausea and disorientation [24, 31]. In summary, the more and more common scenario of molecular biologists aiming at an interactive 3D investigation of protein dynamics and the interaction of several proteins with each other, presents a substantial challenge for visualization technology and interaction.

To achieve interactive rendering in the above mentioned scenario, we now propose a new two-level approach for rendering dynamic proteins. The basic idea builds upon the natural hierarchical structure of

- *Ove Daae Lampe is with Christian Michelsen Research, Norway, E-mail: ove.lampe@cmr.no.*
- *Ivan Viola is with Department of Informatics, University of Bergen, Norway, E-mail: ivan.viola@uib.no*
- *Nathalie Reuter is with Computational Biology Unit/BCCS, University of Bergen, Norway, E-mail: nathalie.reuter@bccs.uib.no*
- *Helwig Hauser is with Department of Informatics, University of Bergen, Norway, E-mail: helwig.hauser@uib.no*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

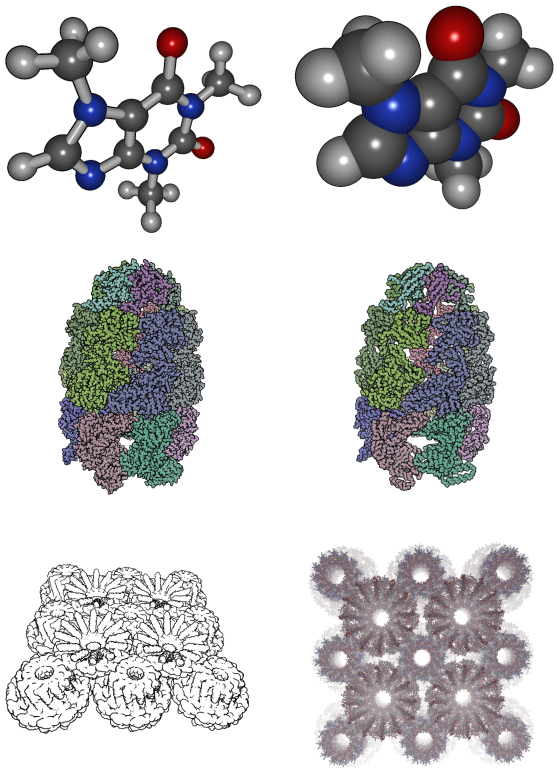

Fig. 1. Different visual representations are needed when visualizing atom structures. Top: left Caffeine using ball-stick, right Caffeine using space-fill. Middle: left GroEL using separate chain color and cartoon borders, right GroEL backbone only. Bottom: left Bacteriophage phi-29 [32] connector array using cartoon Sketch, right same using depth cue (fog).

proteins. Accordingly, we decompose the rendering pipeline into two levels: (a) dynamic changes are applied to high-level structures of the proteins such as the backbone; (b) the geometry of low-level structures (residues) is generated on-the-fly for atom-by-atom rendering.

The remainder of this paper is organized as follows: In the next section we review related work. The main idea of two-level rendering and its individual steps are described in Section 3. Section 4 demonstrates the use of our new approach in the context of our cooperative project. In Section 5 we report performance and quality measures. Finally, we summarize our contribution and draw conclusions in Section 6.

2 Related Work

Visualization of protein dynamics is an active research area. One of the latest approaches in the field resulted in the publicly available framework known as VMD (Visual Molecular Dynamics) [12]. This framework has become very rich in features over the years which makes it popular for computational biologists. However, when it comes to the visualization of large protein structures, the rendering performance is not usable in immersive environments [31].

The high computational costs of rendering complex protein geometries have been partly alleviated by reducing the number of dynamic graphics elements to high-level protein structures using so-called cartoon rendering [26, 11]. Displaying the simple geometry of high-level structures also allows to study molecular dynamics in virtual environments [21].

More recent approaches are focusing on displaying the dynamics of every atom in the molecule. Hao et al. [6] achieve interactive framerates for mid-sized proteins (ca. 10,000 atoms) in a desktop environment through simplifying geometry that represents a single atom.

Reducing geometry complexity and increasing performance by depicting the details using image-based representations is a very powerful technique in various fields of computer graphics [13]. Recently, billboard-based techniques have been applied for molecular visualization such as Qutemol [27]. Billboarding results in very high framerates even for reasonably sized non-dynamic molecules. Our rendering of single atoms extends this billboarding technique. As Qutemol is limited to orthogonal projections only, we use the technique proposed by Gumhold [5] for rendering perspectively correct spheres. In the context of the ball-stick representation we use a very similar concept [25, 23] to render the bonds.

Our two-level rendering approach is especially designed to support the dynamic visualization of large atomic structures for stereo projection. In addition to the pre-computed animations of protein structures, we utilize focus+context techniques [7] for interactive exploration such as 3D magic lens [29, 4] or view dependent disptortion [3].

In this paper, we demonstrate how the newest graphics hardware capabilities are utilized to come up with an elegant two-level approach to interactive visualization of large and at the same time dynamic protein structures. Additionally, we improve and integrate the above reviewed techniques for billboarding, perspective correction, and explorative interaction to better suit our purposes.

3 Interactive Dynamic Proteins

To understand the spatial relationships in the interaction amongst several dynamic proteins, biologists need to study such behavior using stereo projection and on high-resolution projection walls. To satisfy their needs, user interactivity cannot be compromised by the extent of the scene complexity. In the following we describe a new rendering approach which enables this required interactivity, even for very large protein structures. Position updates for every single atom in large proteins in every single frame would normally drop the rendering performance unacceptably due to the CPU time needed to translate atoms and due to limited I/O GPU bandwidth. By exploiting the fact that the NMA simulation only calculates vectors for backbone elements and then afterwards applies them to all atoms, we can send the initial vector to the graphics card and have it applied to the atoms in the corresponding residue on the graphics card.

In the first level of our new rendering pipeline, dynamic changes in the protein therefore are applied only to the backbone *control points*, where each residue is represented by a one control point, i.e., by a single vertex. These comparably small numbers of vertices are then transferred to the GPU. In the second rendering level we dynamically generate the atoms which are contained in the residues, utilizing the geometry shader as featured in the latest graphics hardware generation [18]. For every atom we emit four vertices from the control point of the residue and represent the atoms by perspective-correct billboards. The shading of the spheres which represent the atoms is done in the fragment shader resulting in pixel-precise spheres. The entire rendering pipeline is illustrated in Figure 2 and the individual steps are described in detail in the following subsections.

3.1 Two-Level Rendering Pipeline

Prior to rendering protein structures, we parse the input files [20] that define the protein and generate the protein hierarchy. All residues in the proteic chain are identified and attached to the backbone by their control point. The control point is a particular carbon atom which is present in every residue and which is denoted in the backbone as Cα. This atom serves as the origin of the local coordinate system of the residue. The atoms in the protein are assigned to respective residues and their relative position to the control point is computed. This transformation is defined as translation from the origin of the residue.

For each residue we store the following information:

- control point position (X,Y,Z) – residue position array
- rotations ϕ,θ,ψ – residue rotation array
- index of first atom Λ – atom offset array

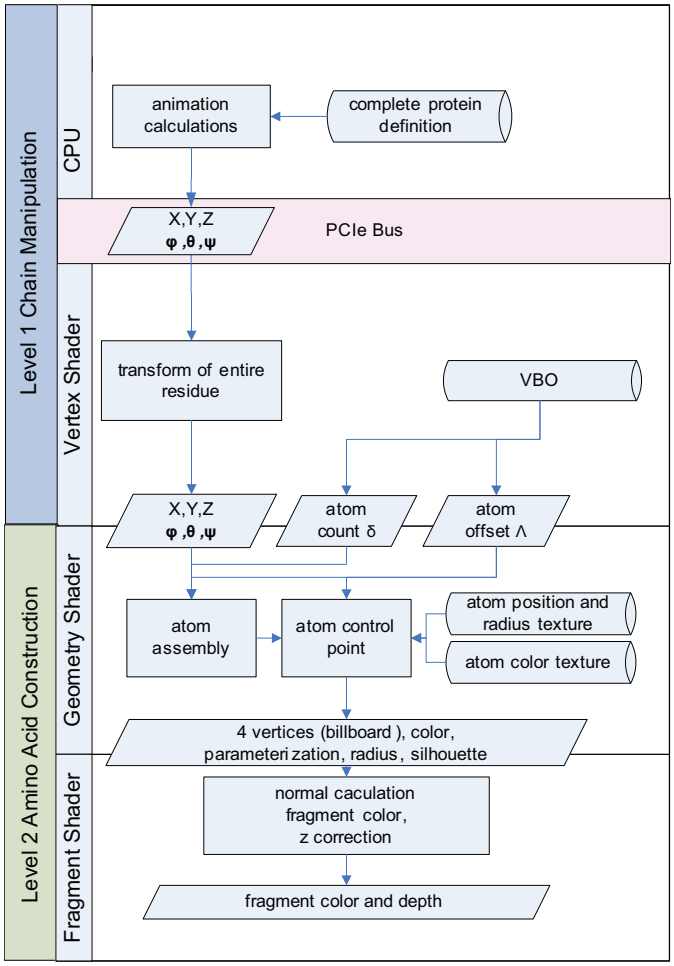

Fig. 2. A data flow diagram showing the different stages in our pipeline, from the CPU through the vertex shader, the geometry shader, to the fragment shader. The data sent, noted by arrows, are multiplied up by the number of residues in all process blocks under *Level 1* and by atoms in *Level 2*

- number of atoms δ – atom count array

The information stored for every residue is also listed in Table 1 with the associated data type.

X	Y	Z	ϕ	θ	ψ	Λ	δ
float	float	float	float	float	float	int	int

Table 1. Residue Composition

All rotations are initially set to zero. We have selected Euler angles as our rotation representation because of their low bandwidth footprint, i.e., 3 floats. This could be implemented by use of quaternions for more efficient calculation and better handling of interpolation.

We reduce the bandwidth requirements during animations by uploading all elements that are static during animations to the graphics memory once. We create two textures, one containing the colors and one containing the positions and radii of all the atoms. Next, we associate the Λ and δ values with the residues vertex array and upload them to the graphics memory as vertex buffered objects (VBO). The atom information is now stored in textures, and the residue information stored in vertex arrays, some residing in main memory, and some in texture memory.

To render the residues, we bind the four above mentioned arrays and the two textures, and then render *points* by sending indices. This transports our values from our two main memory residing arrays, joining them with the other two from graphics memory, and sending them to the vertex shader. Transformations, which are applied to this one *point*, done by the vertex shader, then affect the entire residue (and not only Cα).

To render all the atoms which are associated to a residue, the geometry shader fetches the atom information, starting at Λ, and iterating until $\Lambda+\delta$. The maximum of atoms this algorithm can output is then limited by the hardware platform. By the time of writing the upper limit of the G80 was set to 1024 scalars. This means that our algorithm could theoretically output 64 atoms per geometry call. To resolve the location of such an on-the-fly created atom we need to translate the iterated value to texture u and v values where the color, the position (relative to the origin of the residue), and the radius are stored. This is done by exploiting the newest features in the shader specification *textureSize* and *texelFetch*.

To place the atoms correctly, the geometry shader creates a matrix from the Euler angles (ϕ,θ,ψ) and the position provided (X,Y,Z). The atom positions previously fetched from texture memory must then be multiplied by this matrix, before they are sent to the next step which creates the actual geometry that is sent to the fragment shader.

We represent atoms by billboards (see Section 3.3 for a detailed discussion). To render one atom, the geometry shader generates four vertices in a billboarded *quad* (in fact in a small tri-strip since the geometry shader cannot output quads), the radius of the atom, the atom color, and other information which the fragment shader requires to accurately portray the intended sphere that represents the atom.

One call to render a residue consists of 6 floats · 4 bytes/float = 24 bytes. Rendering one residue is a factor of $24bytes/12bytes = 2$ times bigger than rendering a single atom. However, residues are of size 7–24 atoms and on average about 10 [16]. Therefore we observe that the bandwidth usage is reduced (on average) by about $10/2 = 5$ times. Ignoring the hydrogen atoms, all residues consist of at least 4 atoms each (NCCO, as in glycine). Even if we only render the backbone of a protein (containing the four minimal atoms per residue), we still save half of the bandwidth with our approach.

3.2 Dynamic Scene Rendering and Exploration

Animations often adhere to certain *skeleton* restrictions. This is also the case with proteins. Especially in NMA simulations; the backbone of the protein is animated and the side-chains follow this movement. To dynamically update the scene, we first send the movement of the backbone to the graphics card. Every transformation of a control point of the backbone, defining the origin of a particular residue, is defined through a new position (X,Y,Z) and new Euler angles (ϕ,θ,ψ). We then apply a local coordinate system to render the residue relative to the control point in the backbone. In the second level of rendering we reassemble the entire residue (7–24 atoms) by providing the graphics card with one translation and one orientation description.

With our rendering pipeline we support two different types of dynamics, i.e., the animation sequences as resulting from the simulation of protein dynamics and the displacements of protein elements according to explorative interactions such as the 3D fisheye lens distortion.

The simulation of protein dynamics is usually a computationally very expensive process. Therefore, this is done in a pre-processing step, separated from the rendering pipeline [10]. In order to achieve the highest possible compatibility with all available tools, the result of such a simulation is stored using a standard file format [20] for every frame. To reconstruct the backbone animation, we parse the simulated frames prior to rendering. We extract the position changes and orientation angles, which then define the transformation of the residue in the second level of our rendering approach (see Section 4 for a specific example).

In addition to these simulated protein animations, we also support procedural animations which increase the understanding of the complex spatial arrangements of atoms in the proteins. Motion cues are of significant advantage when it comes to 3D space perception, even in the case when combined with stereo projection [30]. The overall structure of a complex protein can be well studied using global movements

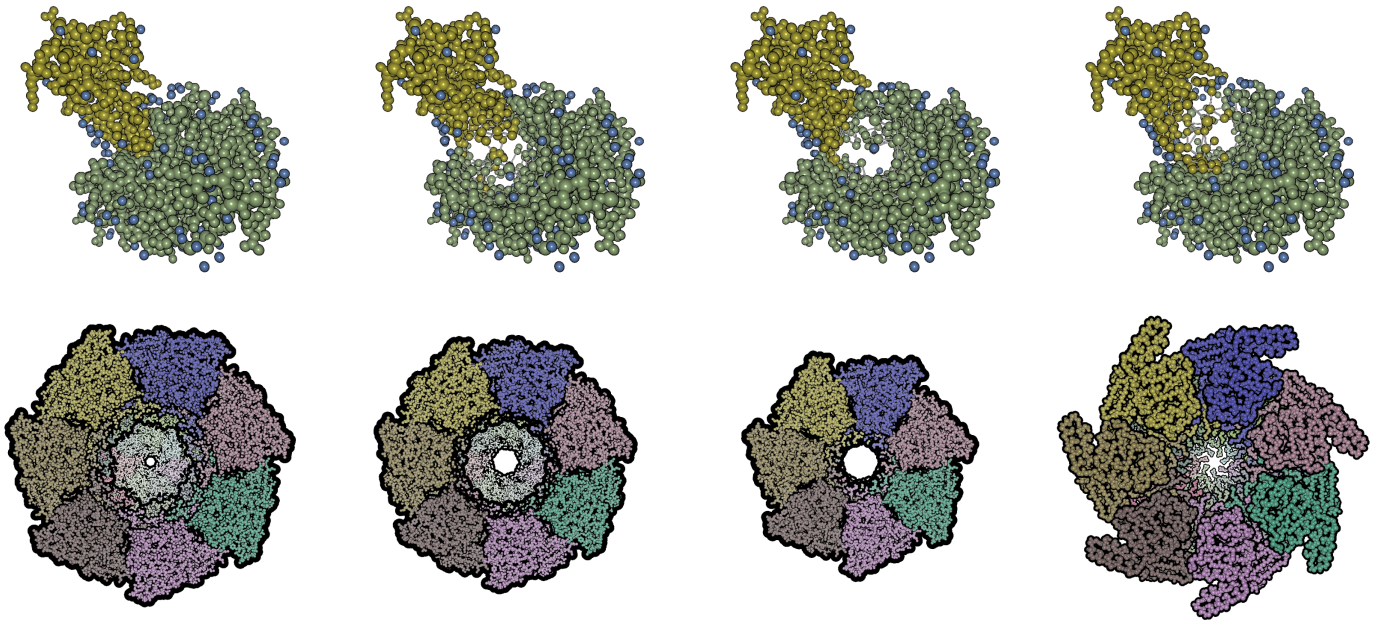

Fig. 3. Top row: Complex (proteinase/inhibitor) visualized with separate color per chain and fisheye to study weak bonds connecting the two chains. Bottom row: Shows procedural pulsation, where the global symmetries become well visible in the GroEL protein visualized here. See also the accompanying videos.

such as a *pulsation* from the center of the protein or along a protein symmetry axis. For demonstration we perform a pulsation procedural animation on the GroEL protein along its rotational symmetry axis (see Figure 3 top row). The displacement function is in this case defined as:

$$\vec{d} = -\begin{bmatrix} p_x/\sqrt{p_x^2+p_y^2} \\ p_y/\sqrt{p_x^2+p_y^2} \end{bmatrix} \tag{1}$$

$$\Delta = \sqrt{p_{x0}^2+p_{y0}^2}\cdot r\cdot(1+sin(\varphi+k\cdot p_{z0})) \tag{2}$$

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} p_{x0}+\Delta\cdot\vec{d}_x \\ p_{y0}+\Delta\cdot\vec{d}_y \\ p_{z0} \end{bmatrix} \tag{3}$$

To calculate a distortion effect that creates a pulsating movement around the z axis, i.e., the axis of rotational symmetry, we first calculate the displacement direction $\vec{d}$ that will point away from the z axis. The pulsation Δ is proportional to the distance from the z axis. Furthermore, we define Δ to vary with a sine wave depending on a time-varying parameter φ and the z height p_{z0}. To control the phase of the sine wave in z we also introduce phase shift k and sine amplitude r. This pulsation effect is as a result of its large relative distortion along its center axis, only suitable for a class of proteins that have either rotational symmetries or a cavity along this axis.

In case of tightly clustered structures it may become difficult to correctly understand the spatial arrangement. Using the small movements in the structure, the depth perception can be significantly strengthened. For this purpose we use a *jittering* procedural displacement. This is demonstrated in the bottom row in Figure 3. The displacement function is defined as:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \vec{p}_0 + r\cdot\begin{bmatrix} \sin(\varphi+p_{x0}) \\ \sin(\varphi+p_{y0}) \\ \sin(\varphi+p_{z0}) \end{bmatrix} \tag{4}$$

where φ is the time-varying parameter, $\vec{p}_0$ the origin of the atom, and r the distortion amplitude.

When the structures in the protein are very tightly clustered it is difficult to understand the connectivity and *jittering* helps. Another useful opportunity is a user-steered rearrangement by means of focus+context visualization [7]. To do so, we apply local distortions utilizing a fisheye lens approach. Our lens function is defined as

$$f(d,r) = \begin{cases} \left(\sin(\pi+\frac{2d\pi}{r})+1\right)/2, & \text{if } \mathrm{d}\leq\mathrm{r} \\ 0, \text{ifd} > \mathrm{r} \end{cases} \tag{5}$$

where $f(d,r)$ is the force magnitude applied to atoms, with r being the radius of the area to be affected by the distortions, and with d being the point-line distance from ray $(\vec{p},\vec{d})$ when defined by viewer position $\vec{p}$ and direction $\vec{d}$ of the mouse pointer. While d defines the magnitude, force $\vec{F}$ defines the direction of the force as given below (with $\vec{A}$ being the position of the atom to be tested).

$$\vec{f} = \vec{d}\times\left(\vec{d}\times(\vec{p}-\vec{A})\right) \tag{6}$$

$$\vec{F} = \vec{f}/|\vec{f}| \tag{7}$$

3.3 Atom Rendering

Our visual representation uses spheres of different size and color to depict the atoms in the protein. Rendering spheres by the use of billboards is a technique that significantly increases speed performance as compared to tessellated primitives. Billboarding is often used as a load balancing method of increasing speed while loosing visual quality. Using advanced z-buffer correction on these billboards and per-pixel shading calculations accurately displays pixel-precise spheres. Our rendering approach using billboards for atoms and bonds builds upon existing works in the field [27, 5, 25].

For the parallel projection, the sphere/atom billboards are drawn orthogonal to the view plane. Then the correct z is computed in the fragment shader as follows:

$$r^2 = z^2+l^2 \;\Rightarrow\; z = \sqrt{r^2-l^2} \tag{8}$$

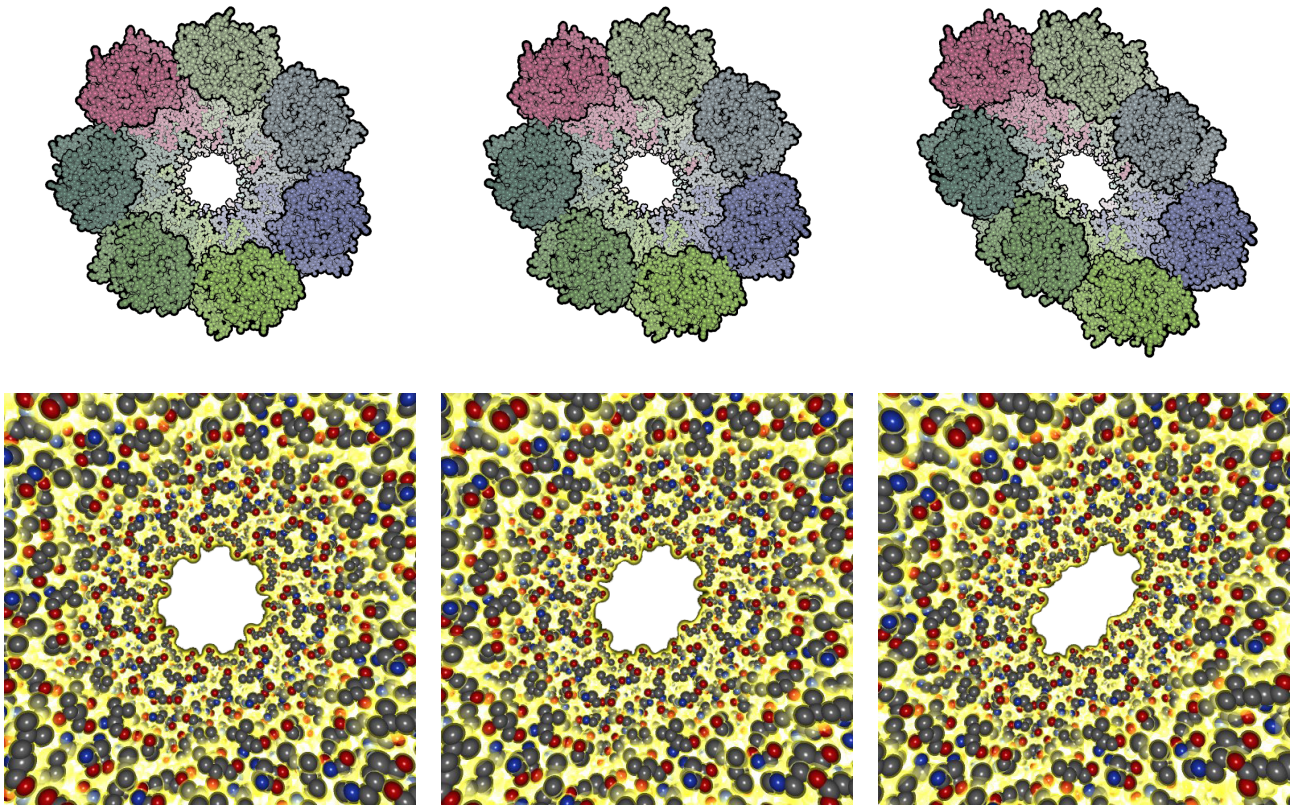

Fig. 5. Interactive visualization of the chaperonin GroEL protein dynamics. Top row: frames showing dynamic changes in the arrangement of the entire structure. Bottom row: frames showing the exploration of structure movements inside the protein.

In Equation 8, r is the radius, l is the distance from the spheres origin, and z is the height correction that needs to be applied.

Using the perspective projection, the above equation gets a bit more complex since there are two important aspects which one has to take care of. First, the radius, or the circumference will be smaller, since the viewing ray will intersect the sphere before the ray hits the billboarded circle. Therefore, the correct intersection depth has to be computed as illustrated in Figure 4. The points on the circle where the view ray is orthogonal to the circle tangent are $s_{\perp}^{1}$ and $s_{\perp}^{2}$. These points define the plane displacement of the billboard. The second issue is that the z correction is not parallel to $\tilde{m}$ either, but will follow the ray. In both orthographic and in perspective projection the normal calculation is the same once you have identified the intersection point on the sphere. If the intersection point is $\vec{i}$ and the origin of the sphere is $\vec{o}$, then the normal is $\vec{n} = \vec{i} - \vec{o}$.

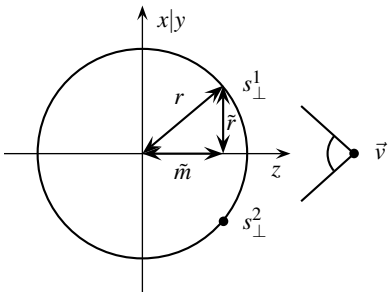

Fig. 4. Silhouette calculation. $\vec{v}$ is eye position. $\tilde{r}$ is the corrected billboard radius, according to projection.

Previous billboarding techniques [5, 25] do not make use of the latest graphics hardware capabilities. For each vertex in the billboard they require to perform calculations for the silhouette, the camera direction, and the cross product to define the orthogonal plane to the camera direction. We utitilize the geometry shading capabilities to reduce these calculations to one vertex only, then *emit* four vertices (one per billboard corner) and reuse the calculations from the original vertex. This implies nearly four times performance gain for rendering a single atom.

4 Protein Analysis

In this part of our paper, we briefly report from our cooperative bioinformatics project and demonstrate how the new two-level rendering approach helped to improve our insights in complex and dynamic protein structures. In our research we aim at studying interaction of multiple proteins also considering their dynamics. Furthermore, we are investigating re-structuring of existing proteins for efficient drug design.

When analyzing large molecular structures such as the trypsin protein with its inhibitor BPTI (Bovine Pancreatic Trypsin Inhibitor) or the chaperonin GroEL protein and their dynamics (see Figs. 6 and 5), we need to consider overall and detailed effects. Accordingly, it is very useful to visually consider proteins on different levels of their internal structure.

The chaperonin GroEL protein is a very interesting and an especially stable molecule which has been extensively researched. It plays an important role in conjunction with the folding of many other proteins, including prokaryotes, chloroplasts, and mitochondria. We investigate this protein and its function by taking a close look at its intra-protein dynamics. For this purpose, we have previously developed a

web-based tool to calculate large amplitude movements of a protein, starting from the Cartesian coordinates of its Cα atoms [10]. This tool, based on the Molecular Modeling Tool Kit [9], is called WEBnma and performs Normal Modes Analysis (NMA) of proteins [8]. Among other things, this tool allows to create an animation of the movements which a protein is susceptible to undergo. The animation can be visualized within the web browser or downloaded and visualized with VMD. However, in both cases we display only the Cα and still the animation is relatively slow.

Images in Figure 5, on the other side, are frames from our new interactive visualization of the chaperonin GroEL dynamics.

The GroEL protein [33] contains 53,822 atoms and the animation consists of 150 different structures. The visualization allows us to see not only the Cα on which the calculation is made but also the side chains. This is an important level of information. We get a better view of the formation of cavities and the deformation of the surface. We in addition incorporate halo effect around individual atoms to improve the depth cueing. The animation also is much faster and smoother than with other tools such as VMD [12].

In the case of the trypsin protein, for example, we are interested in its associated inhibitor BPTI (2ptc) [22]. An inhibitor is a molecule (protein or not) which prevents a protein from functioning. Inhibitors are important regulative factors with respect to the activity of certain proteins. An inhibitor binds to its partner following a *key-lock model*, meaning that they have complementary structure (also referred to as *molecular docking*). The investigation of the detailed structure of each partner helps us understand whether they couple and, if yes, how they couple. When a disease is caused by a malfunctioning protein, for example, we aim at inhibiting its activity. In drug design, it is therefore often important to find compounds which would have the necessary characteristics to be inhibitors of the malfunctioning protein. To do so, an in-depth 3D analysis of the structural aspects is very important during the search for structural complementarity.

The new two-level rendering approach enables us to investigate the proteins in their full complexity (with all atoms), even if they are composed of hundreds of thousands of atoms, interactively in 3D (see Fig. 6(a)). We are able to investigate their detailed shapes while they are moving smoothly. When interactively operating in 3D space, however, we regularly aim at temporarily reverting to a more overview kind of view for the purpose of mentally re-registering the overall structure of the molecule to the current view – due to its complex structure and large number of atoms it is not always immediately obvious which parts connect to which others along the backbone. With our new rendering approach we can very easily suppress the unfolding of residues and thereby only show the backbone of the proteins (Fig. 6(b)). In some situations, however, for a very detailed view, we temporarily disable the rendering of all atoms but just those of one selected residuum (Fig. 6(c–f)).

Another protein we are interested in is the goose lysozyme (153L) [14]. It has a relatively complex structure and showing it with all atoms results in a pretty packed visualization. For an overall analysis, we again make use of the option to not show residues but only the backbone. To better explore the structure of the backbone we apply an interactive 3D fisheye distortion (with care, however, i.e., moving the fisheye around slowly). The fisheye lens locally separates parts of the backbone and thereby reveals details relations (see Fig. 8).

We have performed initial tests of our application in our VR lab. First impressions indicate that immersive visualization is very useful for protein dynamics investigation. The spatial arrangement of protein structures is conveyed much clearer from stereo projection as opposed to desktop environment. Figure 7 demonstrates interactive visualization session of trypsin protein with its inhibitor BPTI (Bovine Pancreatic Trypsin Inhibitor). Additional material is available on http://www.cmr.no/research/protein_dynamics/.

5 Performance Analysis

We have tested the performance of our two-level rendering approach on the chaperonin GroEL (1AON) consisting of $58,884$ atoms for NMA and several other (see Table 2) with fisheye force calculations. While rendering the GroEL NMA analysis we achieve 29 FPS with a 3200×1200 resolution in the stereo mode (or 58 fields per sec). Further results are presented in Table 3 and Figure 9 . These results were made using a NVIDIA GF 8800 GTX graphics card. The stereo results were made on the NVIDIA Quadro architecture.

By implementing force calculations on the GPU (once per entire amino acid), we save an average 7 force calculations. One other significant factor in our gain in performance lies in that with geometry shaders we can reduce the calculations for billboarding and silhouette offset to once per atom vs four times per atom without the geometry shader.

We compare the bandwidth load between our approach and one-level approach also using billboards for object representation but not exploiting geometry shaders for geometry generation. Table 2 shows that the average atom count per residue lies between $7-9$ for proteins used in our comparison. The bandwidth load is depicted in Table 3 where we use the following formula for the bandwidth in bytes per pass:

One-level rendering:

$$\left(atoms \cdot \frac{vertex}{atoms} \cdot \frac{floats}{vertex} \cdot \frac{byte}{float}\right) \Rightarrow (atoms \cdot 4 \cdot 3 \cdot 4) \qquad (9)$$

Two-level rendering:

$$\left(residues \cdot \frac{floats}{residue} \cdot \frac{byte}{float}\right) \Rightarrow (residues \cdot 6 \cdot 4) \qquad (10)$$

It turns out that our method uses approx 15 times less bandwidth as compared to a one-level approach. Moreover, this comparison does not include rendering of hydrogen atoms as they are usually neglected in molecular visualization. In cases when protein investigations would require rendering of protein including hydrogens, the atom count will drastically increase, while the residue count will stay the same. This means that the bandwidth and calculation load for one-level approach will be at least doubled, whereas our load will stay the same.

6 Summary and Conclusions

We have designed a two-level rendering approach for immersive visualization and exploration of protein dynamics. Our approach performs better than that of a One Level with approximate 60% and consumes significantly less bandwidth as compared to a one-level approach; that makes it superior in scenarios with frequent positional change of the backbone in proteins.

During studies of proteins, biologists often like to switch between different levels of abstraction in proteins, e.g., first we might be interested in getting the big picture and seeing the backbone structure only,

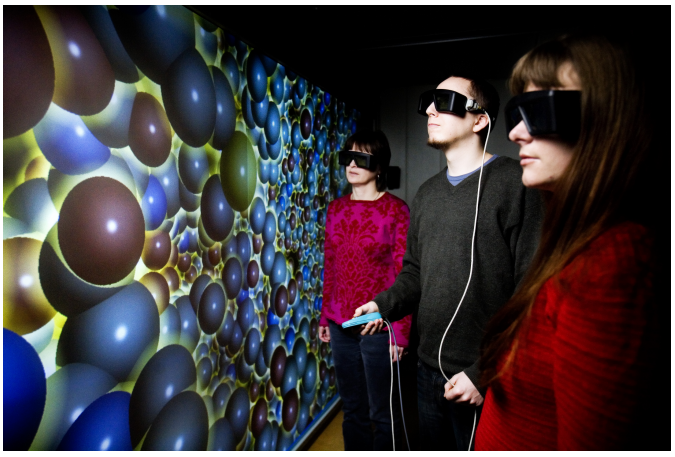

Fig. 7. Photograph taken from a visualization of structures of trypsin protein with its inhibitor BPTI (Bovine Pancreatic Trypsin Inhibitor) in our immersive environment . Photo ©Bjørn Erik Larsen, www.bel.no

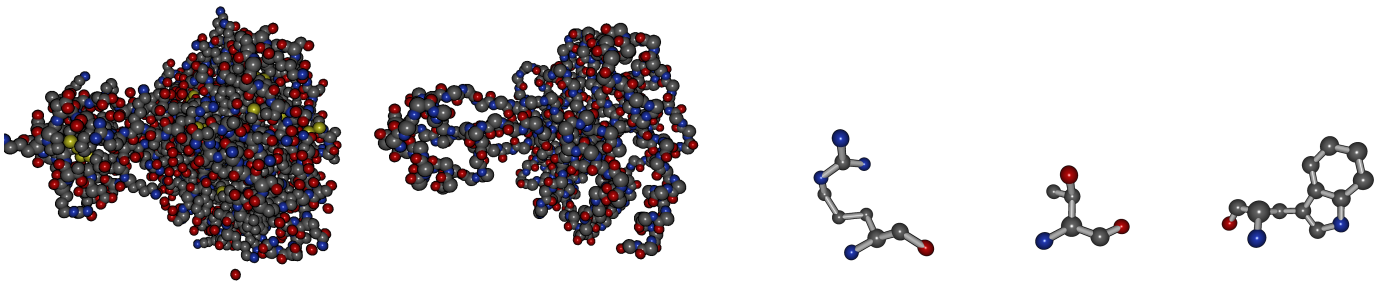

Fig. 6. Hierarchical structures in the trypsin protein with its inhibitor BPTI (Bovine Pancreatic Trypsin Inhibitor): entire proteic chain, backbone without side-chains (proteic chain build only out of glycine), individual residues arginine, threonine, and tryptophan.

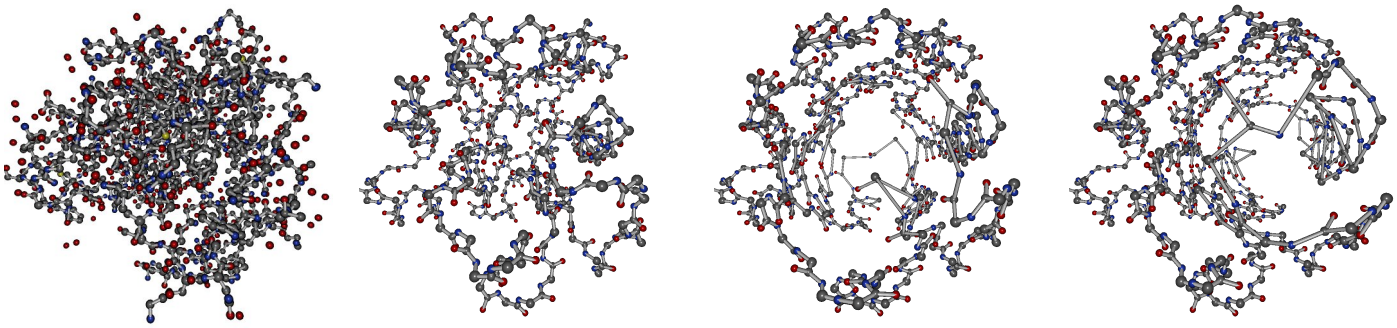

Fig. 8. The protein lysozyme in its original arrangement, only the backbone, and with applied fisheye distortion by moving the lens from the center and rightwards.

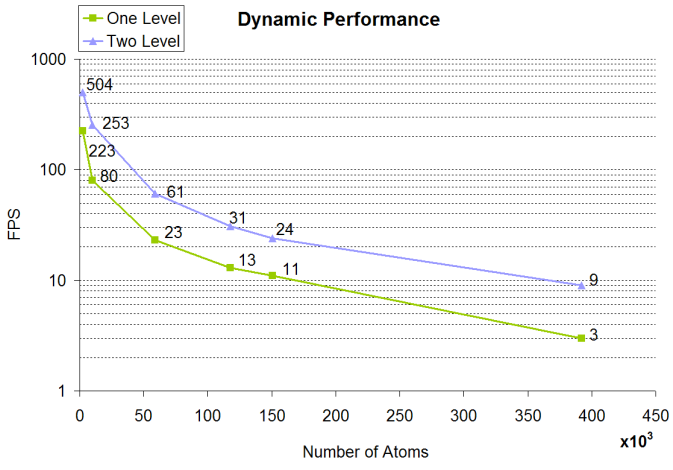

Fig. 9. Performance in FPS comparing One Level rendering to Two Level rendering of dynamic proteins.

Atom Name	Two Level FPS	One Level FPS	Atom Count	Two Level Bandwidth	One Level Bandwidth
2PTC	504	223	2241	6,672	107,568
10K Atoms	253	80	10000	32,904	480,000
1AON	61	23	58884	192,360	2,826,432
1AON*2	31	13	117768	384,720	5,652,864
1VRI	24	11	150720	444,960	7,234,560
Phi-29	9	3	391872	1,156,896	18,809,856

Table 3. Frame-rate and bandwidth comparison of our test cases. Frame-rate presented for dynamic visualizations.

after identifying interesting regions we need to see the entire structure, or the most prominent part of the structure only etc. As our approach is aligned with the hierarchical structure of proteins, change in visual complexity of the scene comes out straightforward from our algorithm.

Proteins are very complex structures and to visually understand them may sometimes become difficult. Here immersive environments coupled with advanced motion and depth cues convey much better structural information than the desktop-based visualization.

Proteins are usually treated as static structures, however their dynamics are often the key to effective drug design. Therefore interactive visualization of dynamics will play an important role in future protein investigations.

ACKNOWLEDGEMENTS

Funding for Nathalie Reuter was provided by FUGE (Norwegian functional genomics program) through the technology platform for bioinformatics. Lars Skjærven is acknowledged for having provided the coordinates files of the NMA on GroEL. We also acknowledge CMR

Atom Name	PDB Code	Atoms	Residues	Bond count	Chains	Avg. Residue Size
Hydrolase(o-glycosyl)	153L	1,614	185	1,460	1	7.74054
Complex (proteinase/inhibitor)	2PTC	2,241	278	2,134	2	7.49281
Complex (groel/groes)	1AON	58,884	8,015	59,304	21	7.32227
Bacteriophage phi-29 upper 5 monomers	1VRI	150,720	18,540	161,164	60	8.12945
Bacteriophage phi-29 connector array	1VRI, 1RVJ, 1YWE	391,872	48,204	418,922	156	8.12945

Table 2. Characteristics of test data sets.

(Christian Michelsen Research) for their funding in this project. Christian Sigg, Simon Green and Stefan Guthe from NVIDIA for providing expertise on the NV80. Vegar Kleppe and Anonymous IEEE Reviewers for excellent feedback, and Mona Nijhof for correcting grammar.

REFERENCES

[1] ATI web page on hardware specs. http://www.ati.com/, 2007.

[2] C. L. Brooks, M. Karplus, and M. Pettitt. *Proteins, a theoretical perspective of dynamics, structure and thermodynamics*. John Wiley and Sons, 1988.

[3] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Distortion viewing techniques for 3-dimensional data. In *Proceedings of IEEE Symposium on Information Visualization*, pages 46–53, 1996.

[4] P. Cignoni, C. Montani, and R. Scopigno. Magicsphere: an insight tool for 3d data visualization. In *Computer Graphics Forum, 13*, pages 317–328, 1994.

[5] S. Gumhold. Splatting illuminated ellipsoids with depth correction. In *Proceedings of Vision, Modelling and Visualization Workshop*, pages 245–252, 2003.

[6] X. Hao, A. Varshney, and S. Sukharev. Real-time visualization of large time-varying molecules. In *Proceedings of High-Performance Computing Symposium*, 2004.

[7] H. Hauser. *Scientific Visualization: The Visual Extraction of Knowledge from Data*, chapter Generalizing Focus+Context Visualization, pages 305–327. Springer-Verlag, 2005.

[8] S. Hayward. Normal mode analysis of biological molecules. In *Computational biochemistry and biophysics*, pages 153–168, 2001.

[9] K. Hinsen. The molecular modeling toolkit: a new approach to molecular simulations. *Journal of Computational Chemistry*, 21:79–85, 2000.

[10] S. Hollup, G. Sælensminde, and N. Reuter. WEBnma: a web application for normal mode analyses of proteins. *BMC Bioinformatics*, 6(1):52, 2005.

[11] H. Huitema and R. van Liere. Interactive visualization of protein dynamics. In *Proceedings of IEEE Visualization*, pages 465–468, 2000.

[12] W. F. Humphrey, A. Dalke, and K. Schulten. VMD - visual molecular dynamics. *Journal of Molecular Graphics and Modelling*, 14:33–38, 1996.

[13] S. Jeschke, M. Wimmer, and W. Purgathofer. Image-based representations for accelerated rendering of complex scenes. State of The Art Report Eurographics, 2005.

[14] L.H.Weaver, M.G.Gruetter, and B.W.Matthews. he refined structures of goose lysozyme and its complex with a bound trisaccharide show that the "goose-type" lysozymes lack a catalytic aspartate residue. *JMol Biol*, 245:54–68, 1995.

[15] J.-J. L. N. Reuter, K. Hinsen. Transconformations of the serca1 ca-atpase: A normal mode study. *Biophysical Journal*, 85:2186–2197, 2003.

[16] D. L. Nelson and M. M. Cox. *Lehninger. Principles of Biochemistry, 3rd edition*. Macmillan Press Worth Publishers, 2000.

[17] S. Noskov and B. Roux. Importance of hydration and dynamics on the selectivity of the KcsA and NaK channels. *Journal of General Physiology*, 129(2):135–143, 2007.

[18] NVIDIA web page on hardware specs. http://www.nvidia.com/, 2007.

[19] Opengl web site. http://www.opengl.org/, 2007.

[20] An information portal to biological macromolecular structures. http://www.pdb.org/, 2007.

[21] J. F. Prins, J. Hermans, G. Mann, L. S. Nyland, and M. Simons. A virtual environment for steered molecular dynamics. *Future Generation Computer Systems*, 15(4):485–495, 1999.

[22] R.Huber and J.Deisenhofer. The geometry of the reactive site and of the peptide groups in trypsin, trypsinogen and its complexes with inhibitors. *acta crystallogr*, 39:480, 1983.

[23] M. Schirski, T. Kuhlen, M. Hopp, P. Adomeit, S. Pischinger, and C. Bischof. Efficient visualization of large amounts of particle trajectories in virtual environments using virtual tubelets. In *Proceedings of the ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 141–147, 2004.

[24] SGI techpubs library, http://techpubs.sgi.com/. http://techpubs.sgi.com/, 2007.

[25] C. Stoll, S. Gumhold, and H.-P. Seidel. Visualization with stylized line primitives. In *Proceedings of IEEE Visualization*, pages 695–702, 2005.

[26] J. E. Stone, J. Gullingsrud, and K. Schulten. A system for interactive molecular dynamics simulation. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 191–194, 2001.

[27] M. Tarini, P. Cignoni, and C. Montani. Ambient occlusion and edge cueing to enhance real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Visualization 2006)*, 12(5):1237–1244, 2006.

[28] D. P. Tieleman. Computer simulations of transport through membranes: Passive diffusion, pores, channels and transporters. *Clinical and Experimental Pharmacology and Physiology*, 33(10):893–903, 2006.

[29] J. Viega, M. J. Conway, G. Williams, and R. Pausch. 3d magic lenses. In *Proceedings of ACM Symposium on User Interface Software and Technology*, pages 51–58, 1996.

[30] C. Ware and G. Franck. Evaluating stereo and motion cues for visualizing information nets in three dimensions. *ACM Transactions on Graphics*, 15(2):121–140, 1996.

[31] S. D. Young, B. D. Adelstein, and S. R. Ellis. Demand characteristics in assessing motion sickness in a virtual environment: Or does taking a motion sickness questionnaire make you sick? *IEEE Transactions on Visualization and Computer Graphics*, 13(3):422–428, 2007.

[32] Y.Y.Guo, F.Blocker, F.Xiao, and P.Guo. Construction and 3-d computer modeling of connector arrays with tetragonal to decagonal transition induced by prna of phi29 dna-packaging motor. *J Nanosci Nanotechnol*, 5:856–863, 2005.

[33] Z.Xu, A.L.Horwich, and P.B.Sigler. The crystal structure of the asymmetric groel-groes-(adp)7 chaperonin complex. *Nature*, 388:741–750, 1997.