EG UK Theory and Practice of Computer Graphics (2005)
L. Lever, M. McDerby (Editors)

Integrating Abstract and Physical Molecular Model Interaction

Dave Thorne[†], Steve Pettifer[†] and Terri Attwood[‡]

[†]Advanced Interfaces Group, School of Computer Science, The University of Manchester, UK
[‡]Sequence Analysis Group, Faculty of Life Sciences, The University of Manchester, UK

Abstract

Historically, bioinformaticians have carried out protein analysis in one of two ways: by concentrating on either the physical structural representation of the subject data or a more abstract sequential representation. This paper describes a system currently in development at The University of Manchester that attempts to unify these two paradigms. We discuss the use of high-end rendering techniques to greatly increase the level of detail and interactivity of molecular visualisation, and describe the requirements placed upon that visualisation by the relationship between the abstract and physical models.

Categories and Subject Descriptors (according to ACM CCS): J.3 [Life and Medical Sciences]: Biology and Genetics

1. Introduction

With the genomes of many organisms now decoded, focus has shifted from determining what a genome *is* to what it *means*. Understanding the specific function of genes individually and in combination, and knowing which proteins they code and what the biological function of those proteins is, opens up the potential for gene therapy and the creation of new drugs to help with currently incurable diseases. This problem has traditionally been tackled in one of two ways. First, by eschewing the physical structure of the molecule and considering only the sequential relationship between its component parts (base-pairs for DNA, amino acids for proteins etc), *sequence analysis* aims to determine function by finding patterns of biological significance along what is essentially a one-dimensional string of characters from a fixed alphabet (or in the case of multiple-sequence analysis, similarities between sequences drawn from several sources). Second, *structural analysis* looks for function determined by what are essentially the chemical properties of the molecular form, finding regions of atoms that are likely to have particular characteristics due to their relative position in the molecule's 3D space. Historically, these forms of analysis have been done independently, however recently, increasing interest has been shown in understanding the relationships between the results gleaned from the both forms: although structural and sequence analysis approach the problem from radically different perspectives, both are attempting to determine regions of biological interest, and therefore it is likely that results from one field have some bearing on the other. In this paper we describe a tool that enables analysis of proteins and DNA simultaneously as sequences and in their structural form, and discuss the issues involved in relating these two types of data, and the approach taken to accelerate the rendering of the 3D structure using current GPU technology.

Both sequence and structural analysis are well-established disciplines, and many tools already exist to aid the scientist in their investigations. Sequential analysis in particular, being for the greater part a text-based pattern-matching exercise, has a wide variety of tools available, the majority of which are algorithmic in nature and run as non-interactive batch jobs. Some aspects of sequence analysis, however, require human intervention and scientific intuition to help cut down the massive search spaces involved to a point where more brute-force algorithms can be employed. Tools such as Jalview [CCSB04], Belvu [Son99], ClustalX [TGP*99], Alscript [Bar93], SeaView [GGG96] and Xalign [WBS94] all visualise in one form or other multiple sequence alignments, enabling the user to move components of a sequence around in order to find regions of conservation. Similarly, tools such as Rasmol [SB92], PyMol [DeL02], JMol, Swiss

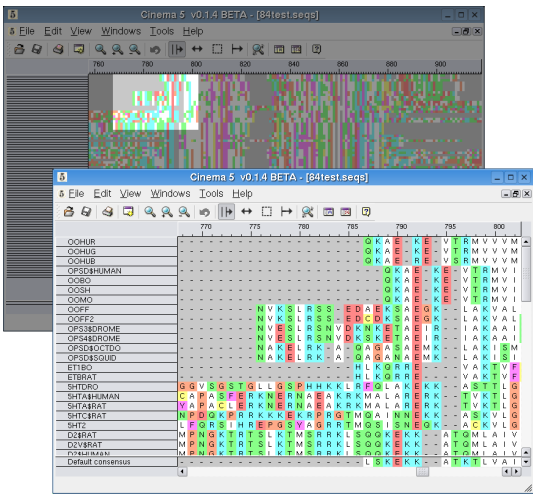


Figure 1: *The CINEMA sequence alignment tool.*

PDB Viewer [KL01] and Raster3D [MM94] all visualise the molecular structure of proteins in three dimensions. What is lacking is the ability to move seamlessly between these two paradigms. Here we describe two tools developed at The University of Manchester as part of the UTOPIA [PAS*02] project that aim to bridge this gap: the CINEMA sequence alignment viewer [PSA04] and the Ambrosia molecular structure viewer [Tho04].

2. Multiple Sequence Analysis with CINEMA

Sequence data represents an abstract view of physical entities, whether those entities be proteins or nucleic acids. In ignoring all structural information, except the ordering of the constituent amino acids or nucleotides, the user can work with a simplified view of the domain in question. As with many other fields of study, this abstraction also provides the analyst new ways to work with that data. In proteomics, the greatest example is the ability of the analyst to ignore regions of a protein that may have evolved into or out of its structure by cutting up the sequence and re-aligning its constituent parts. This act is clearly impossible to do if only the physical representation was available. CINEMA (Colour INteractive Editor for Multiple Alignments) is a sequence alignment tool that provides visualisation of protein sequences and allows for such abstract techniques to be carried out.

All the other tools mentioned above represent their sequences as pages of text, and with the exception of Jalview, do not allow any editing of the sequences being visualised. CINEMA makes use of a GUI (Trolltech's QT) to display its sequences in a form that is not as restrictive as that of its contemporaries. This graphical view allows much more flexibility in how a sequence can be rendered: the size, colour and labelling of residues can be much more cleanly represented. Figure 1 shows an example of this flexibility in the form of multiple levels of zoom on a set of protein sequences. In some cases, such as being able to see a very high level view of a particular alignment, the benefit to the analyst is a functional one: it allows a scope of analysis otherwise unattainable. In other cases the benefits are more cosmetic, but nonetheless quite significant from a user-interface design perspective.

3. Structural Analysis with Ambrosia

Structural analysis takes a 'physical view' of data, that is it allows the analyst to view the actual physical structure of the proteins or nucleic acids in question. As these are physical entities, their structures provide much insight into the functions of their various constituents.

Ambrosia is a molecular structure viewer that forms part of the UTOPIA project. Ambrosia consists of a suite of reusable modules for importing, manipulating and then visualising molecular data. Preprocessing of residue data means that certain rendering formats are much quicker than they would have been, and Ambrosia makes use of the acceleration available on relatively inexpensive graphics cards to tackle the problems of rendering fidelity and interactivity and provide realistic visualisations. It is able to open and display very large data sets very quickly.

4. Interaction with Sequence and Structure

The comparisons available between the two forms of analysis (sequential and structural) are extremely useful, and have been in much use for quite some time. What has been missing is a direct form of translation between the two forms, such as is provided by the interaction of of CINEMA and Ambrosia.

With the two different visualisation techniques in question, there are a number of possible interactions that can take place, but in general, the analyst would want to very quickly relate artifacts from one visualisation to those of the other (i.e. highlight a region in the sequence view, and see this reflected in the 3D view, or vice-versa). The semantics of the communication between the abstract and physical models of the same underlying data is the real challenge. The requirements and challenges are considered in the following sections.

5. Requirements of Interaction

The following are the most important issues identified by this project with regard to integrating Ambrosia and CINEMA, and providing the fidelity and level of interactivity required.

- A data structure is required that can hold the models of both the abstract and physical views simultaneously.

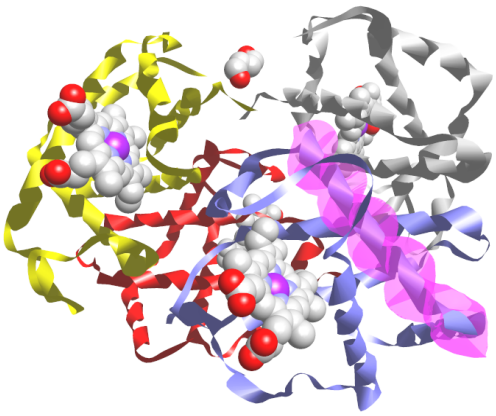


Figure 2: *Rendered image of the 1BIJ molecular model, with a highlighted helical region (in pink).*

- Interactive selection and highlighting are required both in the sequential and the strucural representations. 2D selection and highlighting is trivial, so we shall concentrate on Ambrosia's job of performing these tasks in 3D.
- Very large models are likely to require analysis and should be handled acceptably both in the level of detail supported and the speed of interactivity.
- Realism is key to the efficient use of the system; the closer the rendered image is to the perceived reality the easier it is for the user to gain a semantic understanding of the model.
- Rendering flexibility is essential to a successful molecular modeller due to the wide range of visualisation techniques currently in use.

The former two of these issues will be dealt with in sections 5.1 and 5.2. The latter three issues are somewhat related in their solutions, and so are introduced and dealt with in sections 6 and 7 respectively.

5.1. Relating Structural and Sequence Data

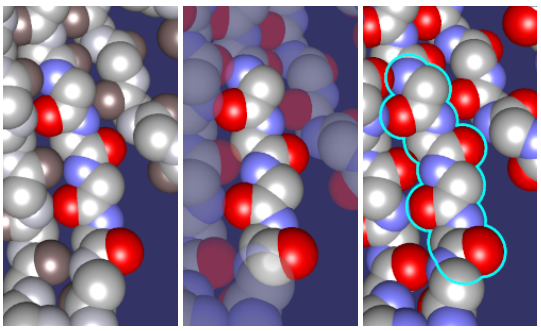
In order to allow meaningful communication between the abstract and physical molecular models, some way of storing the relationships between the two is required. Currently under development is an object oriented data model that can store both types of data, in conjunction with the metadata and provenance that comprise the relationships. The data itself is held in generic molecular data structures that provide standard interfaces for querying and modifications, with the different visualisation applications plugging into those interfaces in order to perform their functions. Designing and populating such a model is more complex than may initially be imagined; unlike the physical sciences where information is typically axiomatic (e.g. values generated from formulae), biological knowledge is often 'fuzzy', being experimental, hypothetical or incomplete. Representing this uncertainty in a data structure is a complex task, and to do this completely is beyond the scope of this work. For our purposes we work with a simplified model specified by a hierarchy of metadata and provenance data describing what the relationships are, who decided that relationship exists, and the degree of certainty of that relationship.

The primary example in this case is the basic relationship between a sequence and a structure: obtaining sequence information is by today's scientific standards straightforward, and there are many databases of such information that are growing in size on a daily basis (the SWISPROT database alone contains 168,297 entries at the time of this paper); by comparison, there are very few reliable descriptions of the molecular structures of proteins in the PDB database, since these biological molecules generally do not respond well to the process of crystallization necessary to determine their form. Scientists are therefore either limited to analysing the relationship between only those proteins that have been crystalized (too restrictive a set in practice), or to working with sequences that have 'near matches' in the the structural database. Thus, unless a convenient match exists between an interesting protein sequence and a crystalized molecule, fuzzy matching techniques must be employed to find a suitably similar candidate. In our case this is performed using the BLAST [TM99] (Basic Local Alignment Search Tool) algorithm, which searches for similarities between sequences, with potential candidates then being queried against the PDB database. Clearly working with similar rather than identical molecules in sequence and structrual format introduces interesting challenges in terms of relating the components for simultaneous visualistion.

5.2. Selection and Highlighting Techniques

When interactively viewing molecular models, it is desirable to both select and emphasise regions of particular structural importance to the user (i.e. that form part of the semantic link between abstract and physical respresentations). Although selection can be achieved by utilising secondary views of the 3D data (such as an 'explorer' that displays the hierarchical structure in a form similar to that of most filesystem explorers) it is favourable to select directly from the 3D representation. Whereas single visible objects are relatively easy to select in three dimensions, selecting arbitrary structural regions can be made more difficult due to occlusions and the complexity of the structure. Selection mechanisms must therefore be both flexible and robust to be useful.

Structural emphasis is surprisingly difficult to achieve, since colour, shape, and position in space are already heavily utilised to give biological meaning to the rendered artefacts. In CINEMA's 2D sequence view, traditional techniques such as marquee-boxes and highlighting and lowlighting regions



(a) (b) (c)

Figure 3: *Rendered images of the 3EBX model, using three different highlighting techniqes to emphasise the same region of the molecule: (a) varying colour saturation, (b) using transparency and (c) outlining the region.*

of interest work well. In Ambrosia's 3D view, the problem is more complex.

We have experimented with the techniques for highlighting regions of the molecule:

5.2.1. Varying Colour Saturation

One of the simplest techniques is to lower the colour saturation of each part of the model that does not comprise the highlighted region. This has the effect of 'greying out' the uninteresting sections and subconsciously causes the user to treat those grey areas with less importance, and so emphasising the highlighted region(s). See figure 3(a).

With relatively small molecules, or conversely with relatively large highlighted regions, this approach is easy and quite effective. For molecules where the highlighted region is occluded however (maybe due to it being inside the molecule, and not visible on its surface) this highlighting method ceases to work effectively. Because of this disadvantage, this method of highlighting is suited to more static images, such as high-resolution snapshots of a model.

5.2.2. Transparency

This is similar to the previous method, and indeed can be used in conjunction with that method, but does afford us a greater degree of flexibility with regard to occluded regions. By alpha blending the unimportant regions we purposefully reduce the user's attention on those areas, emphasising only the opaque regions we have highlighted. As this method does not suffer with the occlusion problem of the previous method, it is much more suited to interactivity in the models. See figure 3(b).

Alpha blending is one possible technique that can provide transparency, but at its most basic level it is just a method of accumulating colour values in the frame buffer of a graphics card. This means that alpha blending is unsuited to true transparency, being unable on its own to provide the reflection, refraction and caustics that are exhibited in the 'real world'. Due to the restrictions of the alpha blending mechanism, producing more impressive transparency effects can best be done using multipass rendering techniques. Ideally we do not want every single sphere to be blended separately as this would cause a 'cloud' of blended colour to appear instead of a transparent molecule. To overcome this we render only the directly visible 'surface' of the molecule, so no blending clashes occur. The practical method for achieving this using OpenGL requires three rendering passes to be made during every frame.

Pass 1 : Draw the highlighted regions, with enabled depth and colour buffers.

Pass 2 : Draw the unhighlighted regions, with an enabled depth buffer and a disabled colour buffer.

Pass 3 : Draw the unhighlighted regions again, with a disabled depth buffer and an enabled colour buffer, drawing only those fragments that have a depth equal to that in the depth buffer (only draws the visible surface).

5.2.3. Blended 'Bendy Tube'

This polygonal highlighting technique simply extrudes a tube along the backbone of a protein, enveloping the portions of that backbone that are interesting (see figure 2). Of course, this 'bendy tube' would have to be alpha blended to make sure that the highlighted regions were still visible, and this could be done in a similar way as in section 5.2.2.

This method is graphically rather attractive, but suffers from some practical pitfalls when viewing certain rendering formats; for example, the bendy tube might be occluded by the molecular model that it is supposed to be highlighting.

Because of the inherent flexibility of a polygonal approach, this technique provides the greatest freedom of expression with regards to how highlights are rendered (e.g. multiple highlights can be rendered using different colours or different tube radii to distinguish between them).

5.2.4. Projected Outline

The final highlighting method mentioned here is that of geometric outlining. This multipass rendering technique makes use of the stencil buffer as well as the depth and colour buffers to perform both stencilling and compositing of various elements into the final image. The objects that are highlighted are rendered with a thin outline of stark colour to delimit their silhouette. See figure 3(c).

Depending on the precise use of the OpenGL buffers, this outline can be rendered in a number of ways: the outline can silhouette only the unoccluded regions of the highlighted object, although this suffers once again from the problem of occluded highlights (as in section 5.2.1); or the outline can

silhouette the whole of the highlighted object, regardless of subsequent occlusion.

The outlining of occluded portions of an object is rather counter-intuitive, making this form of highlighting unsuitable for anything other than directly visible objects. In many cases, where the highlighted object is often (or always) occluded, the outline of that object does not provide enough visual information to the user in order to do their job correctly. For static highlighting however, where the regions of interest are visible, this is a very attractive form of emphasis that lends itself well to adding metadata to an image in the form of callouts or a legend.

6. Visualisation Obstacles

Certainly for our purposes, though arguably for all, high rendering fidelity and acceptable interactivity are important for the effective use of our visualisation software.

One particular model we have used during our development is that of a ribosomal subunit that consists of 51,743 distinct atomic coordinates (PDB Model ID: 1J5E - Structure of the Thermus Thermophilus 30S Ribosomal Subunit). Using the 'spacefill' visualisation format, in which each atom is represented as a sphere with a radius equal to its Van der Waals radius, then we must render 51,743 spheres. In a polygonal rendering environment such as OpenGL even the most rudimentary sphere would require eighty polygons, leading to a polygon soup of more than four million polygons to represent the entire model (using a icosahedron edge subdivision tessellation technique, a rough approximation to a sphere can be represented by 42 vertices and 80 triangular polygons: $80 \times 51,743 = 4,139,440$).

Of course such approximation of spheres brings with it its own set of problems related to the detail and fidelity of the rendered molecule. If Gouraud shading is to be employed in a diffuse lighting model then for anything approaching true spherical shading, each sphere should have at least half as many polygons again, preferably more. As with most virtual environments, our aim should be to make the molecule look as realistic as possible, thereby allowing the user to carry out their analysis without having the burden of consciously translating the rendered image into a cognitive model. All geneticists are familiar with the shiny plastic ball and stick model sets that are used to recreate the structures of simple chemical compounds, and so the most comfortable form of three-dimensional image would be one that recreates this look and feel on the computer screen. This means that ideally specular shading should also be employed as this is a much more accurate (read 'realistic') lighting model in which our plastic balls can be reproduced as faithfully as possible. For the specular model to work using Gouraud shading techniques however, there has to be a far greater degree of tessellation in the polygonal model due to the linear interpolation between vertices.

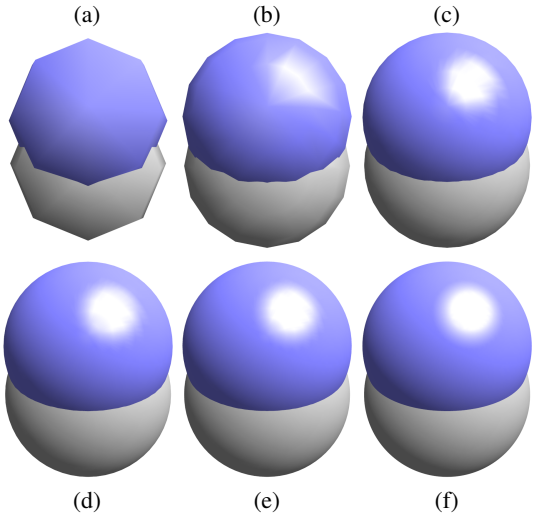


Figure 4: *Rendered images of a diatomic molecule at six different levels of detail: (a) 32, (b) 128, (c) 800, (d) 1,800, (e) 3,200 and (f) 12,800 polygons per sphere. Notice that the specular highlight tends toward 'realistic' the higher the level of detail used.*

Investigations we have undertaken show that in order to depict a relatively shiny sphere using Gouraud shaded specular highlights in OpenGL, a sphere has to have at least 4,000 polygons. This is fifty times as many polygons as the previous example; far too many for even the most advanced graphics cards available to handle acceptably. Figure 4 shows five images of the same diatomic molecule using different levels of detail. We found that a 32 polygon approximation of a sphere is one of the lowest levels of detail at which we can imagine a spherical shape. Unfortunately the intersection is wildly inaccurate and there is practically no specular reflection. The intersection of the two approximations of spheres becomes acceptably accurate when between 800 and 1,800 polygons are used depending on the size of the rendering and scale of the spheres. Luckily, as most molecular models are quite large, each individual atom must be rendered to quite a small scale, meaning this problem is not quite so severe; most high-end graphics cards can easily cope with the tessellation needed to resolve this.

The problem of producing acceptable lighting is much more acute however. For specular highlights to look correct our results show that we need to approximate our spheres with between 1,800 and 12,800 polygons, again depending on the size and scale of the rendered image. The shinier the surface (the greater the value of n in equation (1)) the greater the problem and the higher the degree of tessellation needed to solve it. The rendered images in figure 4 use a 'shininess' of $n = 30$; a greater value of n would require a greater num-

ber of polygons to accurately render the highlight due to the tighter nature of the boundary between the specular and diffuse highlights.

$$I_s = K_s(R \cdot V)^n \quad (1)$$

7. Acceleration Techniques

With the issues detailed above, it can be seen that optimisation and acceleration are essential if any sort of usable interactivity is expected; scalability, fidelity and flexibility simply cannot be attained using standard techniques. Techniques to tackle these problems range from software optimisation to hardware acceleration depending on the exact nature of the problem.

7.1. Model Scalability

The sheer number of polygons required to accurately render a complex molecule is enough to cause most graphics cards to behave poorly; sending the required polygonal information across the AGP bus each frame is too time consuming. There are a number of software and hardware acceleration techniques that can be employed to speed this process up; some 'client-side' in which the optimisation takes place on the CPU and in main memory, and some 'server-side' in which the optimisation takes place in the memory and GPU of the video card:

display lists can be used to cache the polygonal data for the model in main memory (client-side);

vertex arrays are a similar concept to display lists but allow for more flexibility in how the data is cached and used (client-side);

vertex buffer objects can be used to cache the polygonal data in the video card's memory (server-side), drastically reducing the data transfer across the AGP bus (to almost nothing) and so affording a substantial increase in rendering speed.

The use of vertex buffer objects gives us by far the greatest increase in speed because of the resultant per-frame data transfer across the AGP bus. It is conceivable that all the required polygonal information could be transferred to the GPU's memory space before rendering begins, in which case the only data that must be transferred per frame are the model view matrices and the actual rendering commands. Unfortunately, even with today's high-end graphics cards, restrictions in the amount of graphics card memory available can mean that it is impossible to store all the polygonal information at one time.

Each vertex could consist of three pieces of information: a position, a normal and a colour. For a model consisting of 50,000 atoms, with each atom consisting of a sphere of around 800 polygons (1602 separate vertex references when tessellated in a latitudinal-longitudinal fashion), this gives us $1602 \times 50,000 = 80,100,000$ vertices. Assuming each position and normal are represented using tuples of 4-byte floating point numbers (x, y and z) and that the colour is represented in RGBA byte form, then each vertex would require $(3 \times 4) + (3 \times 4) + 4 = 28$ bytes of data for its representation. This equates to more than two gigabytes of data: far too much for any graphics card to handle without paging the data into the GPU's memory space during every frame.

Model	Atoms	Vertices	Memory (MB)
1J5E	51,743	102	120.80
1A5U	31,824	146	106.35
168L	6,445	786	115.95
1F88	5,069	1,026	119.04
1A08	2,058	2,706	127.46
3EBX	569	9,606	125.10

Table 1: *For six models of varied size, this table shows the maximum number of vertex references per sphere that can be held in the memory of the graphics card (assuming a maximum server-side memory of 128MB).*

We can lower this number considerable by realising that in a model of 50,000 atoms, each atom will be on a fairly small scale. Rendering spheres with a much lower level of detail is acceptable, as long as the viewpoint is sufficiently far from the molecule in question. Secondly, as frequency of colour changes required in a model can be very low, storing colour information for every vertex can be counter productive. In most models the cost of sending the colour changes across the now almost unused AGP bus is negligible, and so by keeping the colour changes client-side we can free up four bytes per vertex.

In general, we can tailor the level of detail depending on the size of the model; larger models can handle a lower level of detail than smaller models because the scale of the rendered image elements will be smaller. Table 1 shows some calculations regarding the highest rendering level of detail that can be held on a 128 megabyte graphics card when different models are used. Figure 5(a,c) shows parts of the two rendered models 3EBX and 2GLS using the level of detail from table 1. The rendering of the 3EBX model has perfectly acceptable sphere intersections and lighting due to the greater level of detail allowed. Although the intersections of the larger 2GLS model are also acceptable because of the smaller scale of the atoms, the lighting is not nearly accurate enough to provide the user with a sense of realism. Luckily, lighting is not a problem as this can be dealt with separately as detailed in the next section.

7.2. Rendering Fidelity

As shown in the previous sections, the lighting of the model is the most difficult aspect to get right. Limitations in the

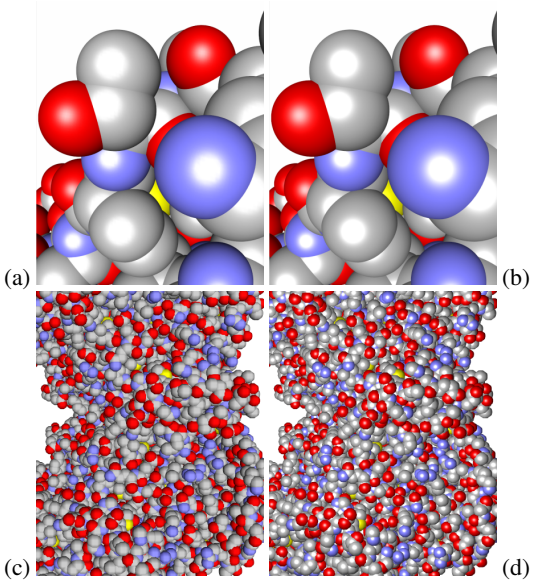


Figure 5: *Rendered images of two models, 3EBX and 2GLS, using a level of detail that would maximise the memory use of a 128MB graphics card. Images (a) and (c) use the standard OpenGL lighting calculations, while (b) and (d) employ fragment shading techniques.*

number of polygons available to medium and large molecular models means that the standard lighting functions of OpenGL are not good enough to provide high fidelity images. The linear interpolation of colour from one vertex to the next is not accurate enough, as can be seen in figure 4: images (a) and (b).

To remedy this we can make use of programmable GPUs to specify our own lighting algorithms. Using a combination of a vertex program and a number of fragment programs, we can bypass the normal lighting stage of the OpenGL pipeline and perform more accurate lighting. Instead of using the normals to each vertex to calculate a colour, and then linearly interpolating that colour, we can use 'fragment shading' techniques to linearly interpolate the normal to each vertex across a polygon, and then use that interpolated normal to calculate the colour of each fragment.

Using such techniques can lead to drastically more accurate lighting of even the most badly tessellated spheres. Figure 5(b,d) shows the same parts of the rendered images as in figure 5(a,c) but using vertex and fragment programs to perform the lighting calculations. The power of the fragment shading approach is that very few vertices are required for the lighting of the sphere to be almost exactly spherical: a sphere of only fifty polygons can be fragment shaded with a higher fidelity than a Gouraud shaded sphere of 12,800 polygons. With even a small increase in the number of polygons used, the lighting would very quickly tend towards perfectly realistic lighting.

7.3. Visualisation Flexibility

There are a great many ways to visualise the same molecular structure, and all (or most) of these ways should be available to the analyst through their visualisation software. Broadly, the following visualisations should be available:

- the physical extent of the constituent atoms, or more commonly known as the 'spacefill' format, in which each atom is represented by a sphere;
- an indication of the atomic bonds within the molecule, usually rendered as a line or lozenge between the two bonded atomic coordinates;
- an abstract view of the molecule's secondary structure, usually differentiating between the many classifications of structure (e.g. alpha helices, beta sheets, turns etc.);
- an abstract view of the 'surface' of the molecule.

Each of these visualisations has its own set of variants, differing on factors such as how structures are rendered and the colouring scheme used. On top of this, within any given molecular model these different visualisation types can be used simultaneously, either for different parts of the molecule or for 'ghosting' one visualisation on top of another.

This flexibility only serves to complicate matters, especially from the point of view of model scalability. Everything mentioned in section 7.1 was based on the assumption that the model was being rendered in spacefill format only, but in some cases the user might want to see a representation of the backbone of the molecule *inside* a translucent spacefill shell of the molecule. Such visualisation options can cause the number of required vertices to almost double in some circumstances, and so must be taken into account when calculating optimal levels of detail.

8. Future Work

8.1. Visualisation Acceleration

In the area of acceleration, there is yet more experimentation to perform. All of the methods described in this paper are to some extent broad techniques that can be 'fine-tuned' to suit any application. These tunable factors include such things as vertex buffer lengths, ordering of render passes in multipass rendering, geometric optimisations such as primitive depth ordering and dynamic level of detail implementations. Amongst other things, future work will most certainly involve the identification of these factors' optimal values, and how much they are affected by the different types of model that can be displayed. Spatial management techniques for large scale virtual environments that are being developed in our group might also be of help with our goal of interactivity.

8.2. Incomplete Translations

Possibly the most difficult obstacle to meaningful communication between the two types of model is that the proteomic analyst does not necessarily have a direct one-to-one translation between a known sequence and its known structure. For almost all recorded structures there is an associated sequence available, but the reverse relationship is not nearly so complete. This void can be filled by lengthy experimentation, but the time and expense of conducting NMR scans for example means that circumstances will most likely remain the same for quite some time.

From a sequence analysis perspective, this is not such a problem; the most important requirement is that selected subsequences such as motifs and fingerprints can be visualised, not necessarily entire sequences. As most motifs can be found in multiple structures, the user can be offered alternative structures that exhibit those motifs in the hope that insight can still be gained. The system can use tools such as BLAST to do this motif-matching task, and depending on the length and obscurity of the motifs required can return the closest structural matches in rank order for the user to choose from.

This user-level heuristic approach is something that the UTOPIA project has identified as important for the bioinformatician; in many cases an experienced geneticist's intuition can give more valuable insights than a rigid algorithm could.

8.3. Component Communication

Underpinning the communication of semantics between the two research paradigms is the communication between the different components of the system. In our current working application there is one CINEMA component and any number of Ambrosia components, although as the project continues it is likely that this setup will grow to include other representation components, centralised data storage or even remote control components for external devices (e.g. 6DoF mice or headmounted displays). In order to accommodate these architectures, and to provide the freedom of distributability, a lightweight networking library is being developed named the eXtensible Networking Engine (XNE) that is designed to meet the demands of distributed visualisation systems.

9. Conclusion

By far the most difficult aspect of the interaction we are seeking is the choice of model. Holding both the abstract and physical data is a challenge in itself, relating that data and storing the provenance and metadata required only compounds the problem. Our work on this is still ongoing. The representation of three-dimensional highlights is another difficult aspect to get right, but we believe we have enough alternatives to do this successfully, and acceptably.

References

[Bar93] BARTON G. J.: ALSCRIPT: A tool to format multiple sequence alignments. *Protein Engineering 6*, 1 (January 1993), 37–40.

[CCSB04] CLAMP M., CUFF J., SEARLE S. M., BARTON G. J.: The jalview java alignment editor. *Bioinformatics 20* (2004), 426–427.

[DeL02] DELANO W. L.: The PyMol molecular graphics system, 2002. DeLano Scientific, San Carlos, CA.

[GGG96] GALTIER N., GOUY M., GAUTIER C.: SEAVIEW and PHYLO_WIN: Two graphic tools for sequence alignment and molecular phylogeny. *Bioinformatics 12* (1996), 543–548.

[KL01] KAPLAN W., LITTLEJOHN T. G.: Swiss-pdb viewer (deep view). *Briefings in Bioinformatics 2*, 2 (May 2001), 195–197.

[MM94] MERRITT E. A., MURPHY M. E. P.: Raster3d version 2.0 - a program for photorealistic molecular graphics. *Acta Crystallographica Section D 50* (1994), 869–873.

[PAS*02] PETTIFER S., ATTWOOD T. K., SINNOTT J. R., VASS N., CORPAS M.: UTOPIA: User-friendly Tools for OPerating Iformatics Applications. Proceedings of 1st All-Hands Meeting of the UK e-Science Programme, September 2002.

[PSA04] PETTIFER S., SINNOTT J. R., ATTWOOD T. K.: CINEMA - A UTOPIAn sequence editor. CCP11 Newsletter, Jan 2004.

[SB92] SAYLE R., BISSELL A.: RasMol: A program for fast, realistic rendering of molecular structures with shadows. In *Proceedings of the 10th Eurographics UK* (1992).

[Son99] SONNHAMMER E.: Belvu. Online document – active on January 2005, 1999. `http//www.cgb.ki.se/cgb/groups/sonnhammer/Belvu.html`.

[TGP*99] THOMPSON J. D., GIBSON T. J., PLEWNIAK F., JEANMOUGIN F., HIGGINS D. G.: The ClustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research 25*, 24 (1999), 4876–4882.

[Tho04] THORNE D.: *Ambrosia: Interactive Visualisation of the Structure of Protein Fingerprints*. Tech. rep., The University of Manchester, May 2004.

[TM99] TATUSOVA T. A., MADDEN T. L.: Blast 2 sequences, a new tool for comparing protein and nucleotide sequences. *FEMS Microbiology Letters 174* (March 1999), 247–250.

[WBS94] WISHART D. S., BOYKO R. F., SYKES B. D.: Constrained multiple sequence alignment using XALIGN. *Comput Appl Biosci. 10*, 6 (December 1994), 687–688.