Die approbierte Originalversion dieser Dissertation ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this thesis is available at the main library of the Vienna University of Technology.

http://www.ub.tuwien.ac.at/eng

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

From Atoms to Cells: Interactive and Illustrative Visualization of Digitally Reproduced Lifeforms

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Mathieu Le Muzic
Matrikelnummer 1326132

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Priv.-Doz. Dipl.-Ing. Dr.techn Ivan Viola

Diese Dissertation haben begutachtet:

Wien, 6. Oktober 2016

Mathieu Le Muzic

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

From Atoms to Cells: Interactive and Illustrative Visualization of Digitally Reproduced Lifeforms

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Mathieu Le Muzic
Registration Number 1326132

to the Faculty of Informatics

at the TU Wien

Advisor: Priv.-Doz. Dipl.-Ing. Dr.techn Ivan Viola

The dissertation has been reviewed by:

Vienna, 6th October, 2016

Mathieu Le Muzic

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Erklärung zur Verfassung der Arbeit

Mathieu Le Muzic
Apartment 15, The Needleworks 41-43, Albion Street, Leicester LE1 6GF, United Kingdom

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 6. Oktober 2016

Mathieu Le Muzic

Abstract

Macromolecules, such as proteins, are the building blocks of the machinery of life, and therefore are essential to the comprehension of physiological processes. In physiology, illustrations and animations are often utilized as a mean of communication because they can easily be understood with little background knowledge. However, their realization requires numerous months of manual work, which is both expensive and time consuming. Computational biology experts produce everyday large amount of data that is publicly available and that contains valuable information about the structure and also the function of these macromolecules. Instead of relying on manual work to generate illustrative visualizations of the cell biology, we envision a solution that would utilize all the data already available in order to streamline the creation process.

In this thesis are presented several contributions that aim at enabling our vision. First, a novel GPU-based rendering pipeline that allows interactive visualization of realistic molecular datasets comprising up to hundreds of millions of macromolecules. The rendering pipeline is embedded into a popular game engine and well known computer graphics optimizations were adapted to support this type of data, such as level-of-detail, instancing and occlusion queries. Secondly, a new method for authoring cutaway views and improving spatial exploration of crowded molecular landscapes. The system relies on the use of clipping objects that are manually placed in the scene and on visibility equalizers that allows fine tuning of the visibility of each species present in the scene. Agent-based modeling produces trajectory data that can also be combined with structural information in order to animate these landscapes. The snapshots of the trajectories are often played in fast-forward to shorten the length of the visualized sequences, which also renders potentially interesting events occurring at a higher temporal resolution invisible. The third contribution is a solution to visualize time-lapse of agent-based simulations that also reveals hidden information that is only observable at higher temporal resolutions. And finally, a new type of particle-system that utilize quantitative models as input and generate missing spatial information to enable the visualization of molecular trajectories and interactions. The particle-system produces a similar visual output as traditional agent-based modeling tools for a much lower computational footprint and allows interactive changing of the simulation parameters, which was not achievable with previous methods.

Kurzfassung

Makromoleküle, wie z.B. Proteine, sind die grundlegenden Bausteine von der Maschinen des Lebens und somit essentiell für das Verständnis von physiologischen Prozessen. Illustrationen und Animationen dienen häufig als Mittel für die Kommunikation in der Physiologie, weil sie einfach und mit wenig Hintergrundwissen zu verstehen sind. Ihre Entwicklung braucht jedoch oft Monate von teurer und zeitintensiver manueller Arbeit. Experten aus dem Bereich von Computational Biology produzieren jeden Tag riesige Mengen von öffentlich zugänglichen Daten, welche wertvolle Informationen über die Strukturen und Funktionen von Makromoleküle enthalten. Wir haben die Vision den Erstellungsprozess von illustrativen Visualisierungen im Bereich der Zellbiologie zu optimieren, wobei die gesamten verfügbaren Daten benutzt werden, anstatt auf rein manuelle Arbeit angewiesen zu sein.

Im Rahmen dieser Arbeit werden mehrere Ansätze vorgestellt, welche darauf abzielen diese Vision zu realisieren. Der erste Teil dieser Arbeit beschreibt eine Rendering Pipeline, welche interaktive Visualisierung von realistischen molekularen Datensätze ermöglicht mit Hunderten Millionen Markromolekülen. Die Rendering Pipeline ist in einer weit verbreiteten Spiele Engine entwickelt und bekannte Computergraphik Optimierungen wie level-of-detail, instancing und occlusion queries wurden adaptiert wurden um die molekularen Datensätze zu unterstützen. Im zweiten Teil dieser Arbeit wird eine neue Methode für die Erstellung von Cutaway Views präsentiert, welche die räumlichen Erkundung von überfüllten molekularen Landschaften verbessert. Die Methode basiert auf der Benutzung von Clipping-Objekten, welche manuell in der Szene platziert werden und einem Visibility-Equalizer, welcher den visuellen Feinabgleich von den verschiedenen Objekten in der Szene ermöglicht. Um solche Szene zu animieren wird häufig agentenbasierte Modellierung eingesetzt, welche Raumkurven produziert die mit mit strukturellen Information kombiniert werden können. Momentaufnahmen von den Raumkurven werden häufig im Schnellablauf abgespielt um die Länge der visualisierten Sequenz zu verkürzen, wodurch jedoch auch potentiell interessante Ereignisse nicht mehr sichtbar sein können. Im dritten Teil dieser Arbeit wird ein Ansatz präsentiert, der Zeitrafferaufnahmen von agentenbasierten Simulationen visualisiert, welcher versteckte Informationen aufdeckt, die nur mit eine höheren zeitlichen Auflösung sichtbar wären. Der letzte Teil dieser Arbeit beschreibt einen neuen Typ von Partikelsystemen, welcher quantitative Modelle als Input benutzt und fehlender räumliche Information generiert, um die Visualisierungen von Molekularen Raumkurven und Interaktionen zu visualisieren. Das Partikelsystem

produziert einen ähnlichen visuellen Output, wie traditionelle agentenbasierten Modelle, allerdings mit geringeren Rechenaufwand. Zusätzlich erlaubt das System interaktiv Simulationsparameter zu verändern, was mit bisherigen Methoden nicht möglich gewesen ist.

Contents

Abstract **vii**

Kurzfassung **ix**

Contents **xi**

I Overview **1**

1 Introduction **3**
- 1.1 Scope and Contributions 6
- 1.2 Contributions of Co-authors 11
- 1.3 Thesis Structure 12

2 Background and Related Work **13**
- 2.1 Visualization of Biological Structures 14
- 2.2 Visualization of Biological Systems 17

3 Rendering and Composition of Molecular Landscapes **23**
- 3.1 Level of Detail 26
- 3.2 Instancing 27
- 3.3 Occlusion Culling 29
- 3.4 Fibres Structures 30
- 3.5 Occlusion Management 32

4 Emulating the Machinery of Life **35**
- 4.1 Observing Multiple Time Scales Simultaneously 37
- 4.2 Quantity-Driven Particle Behaviour 40

5 Conclusion **45**
- 5.1 Summary 45
- 5.2 Lessons Learned 47
- 5.3 Future Work 48

II Publications 51

A Illustrative Visualization of Molecular Reactions using Omniscient Intelligence and Passive Agents 53

B Illustrative Timelapse:
A Technique for Illustrative Visualization of Particle-Based Simulations 65

C cellVIEW:
A Tool for Illustrative and Multi-Scale Rendering
of Large Biomolecular Datasets 75

D Visibility Equalizer:
Cutaway Visualization of Mesoscopic Biological Models 87

Bibliography 99

Part I

Overview

CHAPTER 1

Introduction

Biochemistry lies at the root of complex biological systems that describe the machinery of life. In order to understand how we work, we must first understand the complete cascade of events that begin at the atomic level. Because biological systems span several scales of space and time as well as scientific fields, such as biology, chemistry, mathematics, or medicine, it is crucial to communicate advances in biochemistry efficiently between experts with different scientific backgrounds. Moreover, there is also a growing interest from the general audience to understand how living organisms work.

Visual communication is undeniably efficient for educating a non-expert audience about how physiological processes function. A quick glance into physiological textbooks is enough to realize that they would be close to useless without any illustrations. Illustrations, such as the ones made by David Goodsell, are often used to convey information in such textbooks. The illustrator likes to depict entire sceneries on the mesoscale levels, as shown in Figure 1.1, which would be impossible to observe otherwise in such detail using current microscopy instruments. His paintings rightfully balance scientific accuracy and clarity, which makes them very popular because they are accessible to a large audience.

The realization of such illustrations, however, is very laborious and requires highly skilled individuals. The first step of the creation process consists of gathering knowledge from the scientific literature, in order to fully understand the process that is to be depicted. This task demands a thorough knowledge of biology, as scientific articles are intended to be read by experts and peers. Based on gathered information, the illustrator will decide how to compose the scene, i.e., which macromolecular structures should be present, where should they be located, in which quantities, and what behaviour should they exhibit.

The second step of the creation process is the drawing. It is important not to confuse the work of a scientific illustrator with the work of an artist. Although they both aim at conveying a message or an idea, the artist has the freedom to hide his message behind a curtain of abstraction. On the contrary, an illustrator has to convey a message as

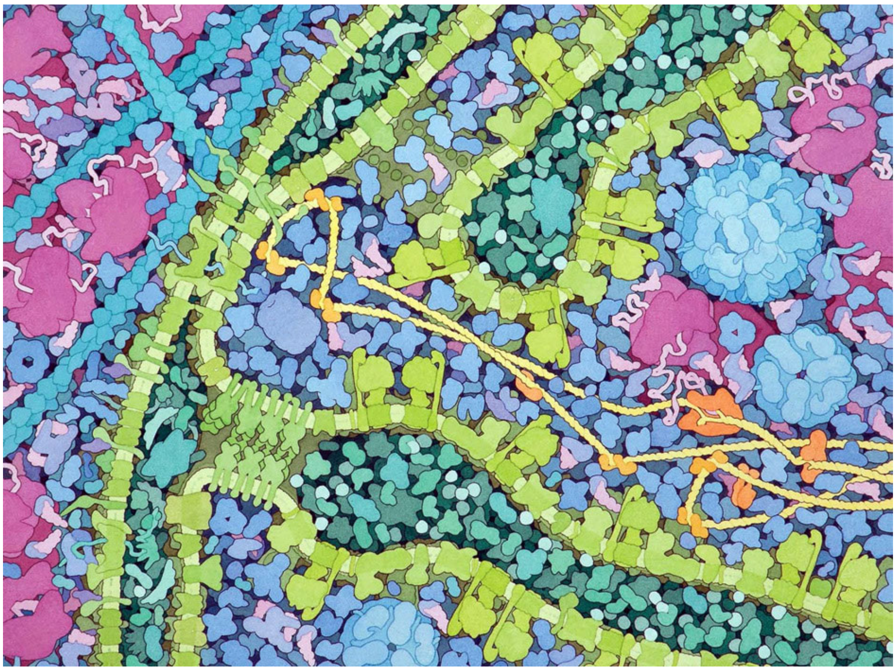


Figure 1.1: A painting made by David S. Goodsell of a cross-section revealing the internals of a mitochondrion. The view focuses on the outer and inner membrane of the organelle (green). The genome is shown in yellow, and the remaining proteins are both coloured according to their type and location. Note how the Goodsell arranged the scene to ensure that all the key elements are visible in a single image, and how the visual mapping of shapes and colors contribute to the clarity of the image.

clearly as possible in order to expose scientific concepts to an uninformed audience. This concretely means that the illustrator is bound to a set of logical guidelines in terms of composition, lighting, color-coding or storytelling.

While some illustrators prefer working with paper and pencil or 2D composition software such as Adobe Photoshop, the new generation of illustrators grew with 3D rendering and animation packages such as Autodesk Maya, Maxon Cinema4D or Blender, made mainstream by the popularization of digital effects in the movie industry. The use of such computer aided design tools greatly facilitates the drawing of three dimensional shapes. Perspective and lighting effects, for instance, are automatically handled by the software, thus leaving artists more time to work on other aspects such as material design, composition, or post-processing. Since less time is spent working on single images, it is also less cumbersome to produce animated stories. Consequently, over the last decade, 3D animation became an increasingly popular means of visual communication for cell biology.

A famous animated educational material is "The Inner Life of the Cell" [Bol06] realized in 2006 by the XVIVO medical illustration studio, which was appointed by Harvard

University. This short animated video beautifully depicts in less than 10 minutes, the logical sequence of events that describe complex biochemical processes of a living cell. The shape of macromolecules are based on real structural information available from public databases and their behaviour based on the most recent knowledge of cell biology. Furthermore, environments surrounding each event is also accurately depicted to provide important information about location and scale. This outstanding work took an entire team of experts and 14 months to produce, which is a good average production time for an educational material of this length. Unfortunately, on top of being time-consuming, the production of such films is also very costly, which limits the accessibility and availability of such visualization.

Another type of media that has a great potential for educational purposes, is interactive applications. Compared to movies, interactive titles, such as computer games, are able to keep the player engaged with educational content using the traditional reward system present in many computer games. Immune Attack [oAS08] and Sim Cell [Gam13] are two famous examples of edutainment titles whose goal is to reveal the functioning of living cells through accomplishment of actions that are part of physiological processes. Promising new VR devices have also emerged over the last years, and are now paving the way for more exciting and engaging user experiences that could have a great educational outreach. However, the production of high quality interactive content, similarly to animated films, is also a lengthy and costly enterprise, as technicians and programming experts are also needed in addition to the team.

Interactive applications may also have an educational purpose without necessarily introducing gameplay mechanisms or score-based rewards. Interactive map applications, such as Google Earth [Goo01], are a good example. Unlike static maps, these applications enable on-demand access to specific information. Through a set of 2D interactions such as zoom, rotate, and pan, or 3D interactions such as tilt, the user is free to navigate to whichever part of earth that he deems interesting. It also features multiple zoom levels from planets down to the size of building, houses or even cars. Another strong advantage of the platform is crowd-based collaboration. Three dimensional data obtained from scans of entire cities can be provided by the municipalities and added to the platform for in-depth city architecture exploration. Finally, the platform is not only bound to static representation of earth, dynamic data such as traffic or meteorology provided by third party applications can also be added in the platform. The final outcome is a system that enables omniscient and three dimensional observation of the planet and its dynamics, based on available data. The educational outreach of this software is undeniable as it transcend all types of media previously used, and provides unconstrained access to multiple types of information at once.

To our knowledge, the concept of reproducing an observable 3D virtual environment as such has not yet been transposed to the level of an entire cell. In order to achieve this enterprise one would need access to important data such as the three dimensional structures of macromolecules and greater ensembles that form the compartments and various organelles of the cells. Cells also carry important functions expressed in biological systems and represented as reaction networks between micro and macromolecules. This input information is used by scientists to model parts of living cells and produce datasets that describe the dynamic behaviour of molecules such as concentration charts, reactions events and 3D diffusion patterns. Fortunately, a large amount of biological knowledge is already available via online public databases. Concerning structural data, the Protein Data Bank [BKW⁺77], for instance, is a project that aims at grouping every known protein structures in a large public data base. Thanks to this resource, it is trivial to access the atomic structure of a very large number of proteins and use it for illustrative purposes. Concerning procedural data, Ecocyc [KCVGC⁺05], is another example of a public database that aim at gathering extensive procedural descriptions of the biological systems that are ongoing in the E. Coli bacterium. Based on these descriptions, biologists have also managed to run partial and whole-cell simulation of the systems occurring inside these species on super computers, and they also made the results publicly available [KPC14]. Similarly to satellite data, or traffic data in Google Earth, the data present in these databases is steadily updated with most recent scientific knowledge by a large crowd of researchers.

To address the limitations of traditional scientific animations workflows, which are time-consuming and expensive, we envision a new type of solution that would be more streamlined and relying on scientific data. Since the visualized output would directly derive from the data, it would become much less cumbersome to create new content or update the existing ones with new information based on recent scientific discoveries. Additionally, we also aim at providing the means to explore the 3D space in real-time across multiple scales, and also to interact with physiological properties of the environment in order to keep the subject even more engaged with the content. Despite the large quantity of available data, there is yet no solution that could generate a comprehensible digital cell based on this data, which would be both dynamic and multi-scale. What is truly needed is a solution that would collect and use all this data and enable real-time visualization and interaction with the showcased models. Although data-collection and real-time rendering are important aspects, this ambitious enterprise also comprises many interesting visualization-related challenges. In this thesis, we present the methodology that we employed to address the various challenges that we have encountered along the way.

1.1 Scope and Contributions

The central vision of this project is to develop a technology that would provide interactive visualization of digitally reproduced cellular lifeforms in order to spread state-of-the-art knowledge of cell biology more easily to the broad audience. Similarly to map navigation tools, we want to offer boundless exploration through the environment and massive zooming from the tiniest atom to the entire organism, so that the audience can query any region of a cell and learn about it on-demand, see Figure 1.2. We also envision the whole model to be dynamic in order to explain the complex machinery that allow the cells to sustain, function, and reproduce.

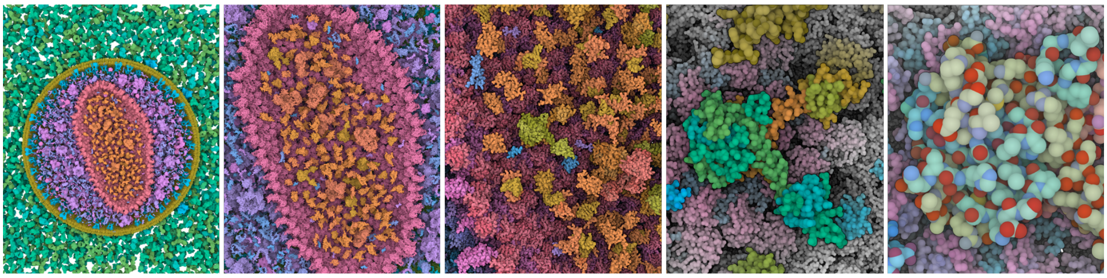
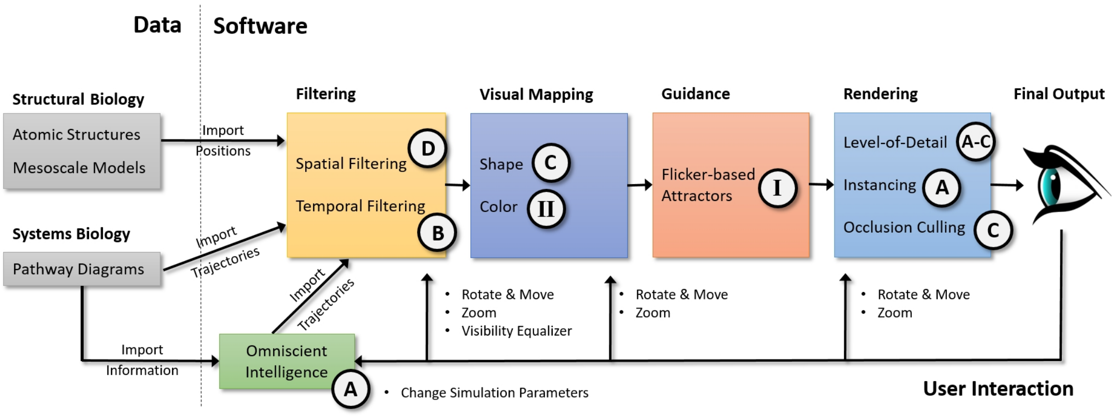


Figure 1.2: Multi-scale visualization of the HIV virus with cellVIEW, a program developed in the course of this PhD project. This image show the massive zooming capabilities from entire organisms down to single atoms, while also revealing intermediate structures such as organelles, compartments and macromolecules.

The scenario that explains how physiological processes work is far from linear, as these processes are designed to respond to multiple type of environmental changes. Therefore, we also wish to provide the means to interact with the functions of life in terms of changing environmental conditions in order to observe and learn how the life-form responds to such change. To achieve this vision, our approach is to integrate up-to-date biological knowledge from several online databases that contain frequently updated scientific findings related to a particular organism. For example, one database will provide us with the structural description on a microscopic spatial scale, another database will provide information on an atomic detail, yet another database will provide us with information on physiology such as reaction networks and life-cycle simulations. A good starting point for our enterprise would be to begin with small unicellular organisms such as E.coli bacterium or Mycoplasma mycoides. These are so-called model organisms because we already have an extensive knowledge about their functioning and structure, so the integration of this information into a visual depiction seems within reach.

In this thesis are presented different research projects that were all driven by this vision. To illustrate how the different contributions of this PhD project fit together, these are laid out in a flow diagram starting from input data to final visual output, and shown in Figure 1.3. The figure also contains a list of relevant first authorship publications referenced in the diagram. Second authorship contributions are also included in this diagram but are not discussed in this thesis. The contributions are encapsulated into blocks that correspond to the stages of a classic computer visualization pipeline, and which are described in the following sections. It is worth mentioning that all the pieces of software associated with these contributions were developed with the same visualization framework, cellVIEW, with the ambition to compile all the work we achieved in a unified solution. In 2016, cellVIEW was also awarded Best Technical Solution at the Austrian Computer Graphics Awards, during the PIXELvienna conference. Executable version and source code are freely available and can be downloaded here: https://www.cg.tuwien.ac.at/research/projects/illvisation/cellview/cellview.php.

Figure 1.3: The flow diagram describing the visualization pipeline which we designed for this PhD project. The contributions of this thesis are placed along the workflow, and the numbering corresponds to the publication listing below the diagram.



This thesis is based on the following publications:

A **Mathieu Le Muzic**, Julius Parulek, Anne-Kristin Stavrum, and Ivan Viola. Illustrative Visualization of Molecular Reactions Using Omniscient Intelligence and Passive Agents. In *Computer Graphics Forum*, pages 141–150, 2014.

B **Mathieu Le Muzic**, Manuela Waldner, Julius Parulek, and Ivan Viola. Illustrative Timelapse: a Technique for Illustrative Visualization of Particle-based Simulations. In *IEEE Pacific Visualization Symposium* (PacificVis), pages 247–254, 2015.

C **Mathieu Le Muzic**, Ludovic Autin, Julius Parulek, and Ivan Viola. cellVIEW: a Tool for Illustrative and Multi-scale Rendering of Large Biomolecular Datasets. In *Proceedings of the Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 61–70, 2015.

D **Mathieu Le Muzic**, Peter Mindek, Johannes Sorger, Ludovic Autin, David S. Goodsell, and Ivan Viola. Visibility Equalizer: Cutaway Visualization of Mesoscopic Biological Models. In *Computer Graphics Forum*, pages 161–170. 2016.

The following article are also related to this thesis:

I Manuela Waldner, **Mathieu Le Muzic**, Matthias Bernhard, Werner Purgathofer, and Ivan Viola. Attractive Flicker: Guiding Attention in Dynamic Narrative Visualizations. In *IEEE Transactions on Visualization and Computer Graphics*, pages 256–265, 2014.

II Nicholas Waldin, **Mathieu Le Muzic**, Manuela Waldner, Eduard Gröller, David S. Goodsell, Autin Ludovic, and Ivan Viola. Chameleon: Dynamic Color Mapping for Multi-Scale Structural Biology Models. In *Proceedings of the Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 53–62, 2016.

1.1.1 Preparing The Raw Data

In our particular use case, the input data is coming from various sources, and needs to be unified before being visualized. Structural information of large number of macromolecules, is publicly available via online databases. Additionally, it is also possible to obtain a valid spatial arrangement of large ensembles of proteins that form mesoscale models for entire cells. The trajectory and reaction history of individual particles for a given biological system may be obtained by modeling approaches called agent-based modelling. This data can thus be combined with structural information in order to reproduce a similar visual output as in animated movies.

However, particle-based modeling has a high computational footprint, which prohibits us from interacting with the simulation in real-time since the data must be precomputed. Quantitative modelling, such as kinetic modeling, is much more lightweight to compute but only provides quantitative and relational information, and the spatial component is entirely missing from the model description. Therefore, we have developed a new type of solution, which is able to generate 3D particle animations, driven by the results of a quantitative simulation, which is described in **Paper A**. The light computational footprint of quantitative models thus allows fast in-situ visualization of particle trajectories and interactions, which was not achievable up to that point.

1.1.2 Filtering

Input data processed by the visualization pipeline consists of positions of single atoms for the macromolecular structures, on the one hand, and the trajectory data for entire groups of molecules, on the other hand. Because of the chaotic nature of the forces driving the motion of molecules inside living cells, the raw visualization of such data often results in an overly complex visual output, which is challenging to comprehend. Thus, it is important to filter out irrelevant and redundant features such as excessively erratic motion and occluding elements to reveal important underlying information buried in the chaos.

Trajectory data is often visualized in fast-forward to shorten the viewing of overly long sequences, which also speeds up the motion of individual particles. Consequently, it is almost impossible to keep track of individual elements and observe key reaction events that enable that same process. Indeed, rates of physiological processes operate at a much larger time scale than the movement of individual particles. With direct visualization, one can observe only one of these scales, either the physiological process or the behaviour of individual particles. In **Paper B**, we have investigated how to simultaneously convey two phenomena that reside at different temporal scale levels. In particular, we have aimed at developing a technique that can show the complexity of diffusion in fast-forward and simultaneously allows viewers to see physiologically-relevant events that would not be observable at such temporal resolution.

Another issue that derives from densely crowded environments is the presence of a large number of bodies that may obstruct the view to key macromolecules and important reaction events. A characteristic of molecular bodies is that many of them actually share the same atomic structure. Our technique, called Visibility Equalizer, and described in **Paper D**, allows the user to see inside dense arrangements of proteins, by providing an explicit control over the visibility of entire groups of molecules sharing a similar structure. Rather than completely displaying or removing entire sets of proteins, we introduced the concept of fuzzy visibility, which allows reducing the concentration of visible elements of a given type, thus revealing the internals of a cell while preserving important contextual information.

1.1.3 Mapping

The next operation of the pipeline is the mapping, which determines visual properties such as shape or color. Because the data we want to visualize features multiple scales, from single atoms and up to entire cells, it is important to adapt the visual representation accordingly to ensure an optimal comprehension of the scene at any given zoom level. A level-of-detail scheme is an optimization technique often used in computer graphics and visualization to accelerate the rendering. The principle consists of progressively switching between simplified shape proxies as the camera distances itself from objects. In the case of molecular visualization, level of detail is two fold, on the one hand, it allows to speed up performance, and on the other hand it can also be utilized to filter out high frequency details, as molecules tend to have complex shapes, which may clutter the view when observing molecular landscapes in their entirety. In **Paper A** and **Paper C** we have explored the use of level-of-detail schemes, specifically designed for macromolecular structures.

Color is a strong visual cue that is extensively used in molecular visualization. In large scenes comprising many macromolecular elements, color coding can have multiple folds. It can either be used to discriminate atoms with different physio-chemical properties, such as charge or hydrophobicity, structural properties such as the type of atom, amino-acid, chain, protein, or even spatial properties such as the membership of element to a given subregion of an organism. Often these properties can only be optimally observed at one single zoom level. In **Paper II** we developed a novel dynamic coloring approach that optimises the color coding based on the current zoom level in order to ensure that only the most relevant information is revealed when exploring multi-scale atomic structures.

1.1.4 Dynamic Visual Guidance

Traditional focus+context approaches are commonly utilized to highlight key interactions between molecular agents when observing an ongoing biological system. However, these may fail when observing the results on large projection displays because they should be effective both for the foveal as well as the peripheral vision, especially when viewing large ensembles of particles that exhibit very chaotic motion patterns. Therefore, we have investigated how to guide the viewer to interesting events in a dense dynamic scene of interacting molecules that are presented on such type of display. In **Paper I**, we have developed a special type of dynamic guidance based on subtle flicker that is effective at guiding the viewer's gaze towards interesting events, is unobtrusive, does not use any visual variables that encode the data, and incurs only a minimal visual modification to the presented scene.

1.1.5 Rendering

The rendering is arguably the most crucial part of our visualization pipeline, as it is meant to display challengingly large and dynamic molecular datasets at a high frequency refresh rate. Realistic atomic structures of entire cells consist of thousands to millions of macromolecules, themselves composed of a few thousands atoms each, resulting in an memory footprint that exceeds the capacity of today's high end graphics hardware. Fortunately, many molecules present in these scenes share the same structure, which allows us to utilize the concept of instancing. Every structure of a particular protein macromolecule is stored on the graphics hardware (GPU) only once, while the positions and orientations of the molecules are stored separably. Instancing is useful for reducing the size of redundant datasets, but also helps to reduce the number of necessary draw calls, which tremendously accelerates the performance when rendering hundreds of thousands of elements. When it comes to the geometric representation of a single macromolecule, we model and render it as a set of atom spheres drawn using 2D impostors, which have a much lower vertex count than 3D tessellated spheres for the same visual output. We also utilize a level-of-detail scheme to dynamically switch between proxies according to the distance of molecules to the camera, which is presented in **Paper A**. In **Paper C**, further acceleration techniques have been designed to optimize the rendering speed even further, such as occlusion culling, as well as a new rendering pipeline dedicated to large and linear fibre structures.

1.2 Contributions of Co-authors

All manuscripts that constitute this thesis were written during the PhD project and the author of this thesis is also their main author. The first author has put the scattered thoughts of the entire research team and realized a concrete and meaningful technology out of it. He was responsible for the development of cellVIEW as well as the prototyped software of the technologies presented in the papers. Ivan Viola, the main supervisor of this thesis, coauthored all manuscripts. Viola is the primary investigator of the research team, and many interesting concepts presented here emerged from his imagination long before the project was even funded. He also provided indispensable mentorship, and high level ideas for a fruitful research direction thus contributing to the crystallization of ideas throughout the projects. **Paper A, B, C**, were all co-authored by Julius Parulek who helped with the conception and implementation of the rendering pipeline, and he also provided valuable advices and support during the writing of manuscripts. He also helped porting the prototyped technology to the Unity 3D game engine. **Paper A** received the participation of Anne-Kristin Stavrum who joined the project as a biologist and was given the task to conceive physiological models utilised to prototype the presented technology. **Paper B** was also co-authored by Manuela Waldner who helped designing and run the user studies, provided insightful inspiration, and also actively participated in the writing of the article. **Paper C** was co-authored by Ludovic Autin, who conceived the showcased models, helped with the development of cellVIEW, and also provided support and insightful advices. **Paper D** was co-authored by Peter Mindek as second first author since both authors were deemed to have contributed equally to the success of this article, this mention is also present in the publication. **Paper D** also received additional help from Johannes Sorger for the writing of the manuscript, and the design choices were also influenced by the feedback we received from Ludovic Autin and David S. Goodsell.

1.3 Thesis Structure

This thesis consists of two parts. The first part summarizes individual contributions and findings and also aims at describing how these single pieces fit together as part of a bigger picture. The second part contains the published articles. Chapter 2 follows the first introductory chapter with an overview of previous work related to the visualization of structural and systems biology models, with an emphasis on multi-scale visualization. A more detailed overview of related works is contained in the individual papers in the second part of this thesis. Following the related work, we provide in-depth details about the contributions, starting with the visualization of strictly static and structural information in Chapter 3. In Chapter 4, we present the work that is concerned with the visualization of large-scale structures enhanced with dynamic procedural information obtained from real scientific data. Finally, we conclude the first part of the thesis in Chapter 5.

CHAPTER 2

Background and Related Work

In biochemistry, there exists two distinct experimental protocols, respectively called dry and wet laboratories. Wet laboratories are where chemical agents are physically manipulated and then observed. Dry laboratories are where computational or mathematical methods are employed for the modelling and analysis of biochemical processes. Over the last decades, the use of *in-silico* experiments (dry laboratories) have significantly increased due to the development of new software and the decreasing costs of super-computers. Despite being often criticized for being too approximate, dry laboratory experiments represent a valuable source of information for researchers nonetheless.

In 2013 Martin Karplus, Michael Levitt and Arieh Warshel were awarded the Nobel prize in Chemistry for their work on theoretical modelling for complex chemical systems [Kar14]. Their work highlights the importance of theoretical modelling as a tool to complement experimental techniques as wet-lab experiments are usually complex and expensive to conduct. The analysis of theoretical modeling brings researchers the necessary guidance to formulate new hypotheses, which can be later on verified in wet laboratories, thus saving the time and money needed to run too many wet lab experiments. As a result of the increasing popularity of dry lab experiments, a significant amount of data has already been gathered and produced.

Data is often stored in digital format and may be shared with peers via online databases. Structural biology and Systems biology are branches of molecular biology that both heavily rely on computational methods. Structural biology informs us about how things look, i.e., what is the atomic structure of a protein, while systems biology informs us about how things work, i.e., what are the micro and macromolecular interactions that influence the functioning of living organisms.

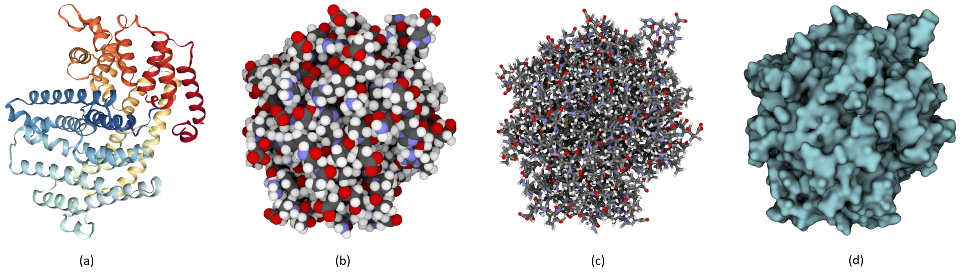


Figure 2.1: Different types of representation used in molecular visualization. (a) The ribbon diagram reveals structural information hidden inside the structure, such as the formation of sheets or helices along the protein chains. (b) The van der Waals representation, also called space filling, renders individual atoms as 3D spheres. (c) The stick model represents the bonds betweens two atoms with lines, but unlike space filling it does not encode the atomic radius. (d) The surface representation wraps the entire molecule with a tight hull that facilitates the detection of cavities that may host important reaction sites.

2.1 Visualization of Biological Structures

Structural biology is the branch of biology that is concerned by the structure of biological macromolecules, such as proteins or DNA, for example, and focuses on understanding the relationship between the structure of molecules and their function. Data acquisition methods, such as X-ray crystallography, are commonly used to read the atomic structure of proteins, i.e., the positions or atoms, their type, and the type of bonds between them. Acquired atomic structures are often stored in digital files and shared via public data bases, such as the Protein Data Bank [BKW⁺77], to facilitate collaboration among peers. This information is then processed to decrypt underlying important structural information, and also used to run molecular dynamics (MD) simulations, which aim at reproducing atomic interactions and forces to observe the actual behaviour of macromolecules over time. Visualization is an important component of this discipline, because atoms are arranged and assembled in three dimensional space and therefore, a visual representation is often required. Biologists developed several types of representation to illustrate molecular structures, and are supported by mainstream visualization packages such as VMD [HDS96] or PyMol [Sch15].

A popular representation among structural biologists is the secondary structure or ribbon diagram (Figure 2.1a), which is used to reveal properties of the protein backbones, such as sheets or helices. The van der Waals surface (Figure 2.1b) is probably the most commonly understood representation and simply shows atoms as spheres whose radius corresponds to the atomic radius. The simplest representation is the sticks model (Figure 2.1c), where each bond is represented as a line, and color coding is used to indicate the atom type at the line extremities or joints. Finally, the molecular surface representation (Figure 2.1d) is used to show a continuous surface that closely surrounds atoms of a protein, and that

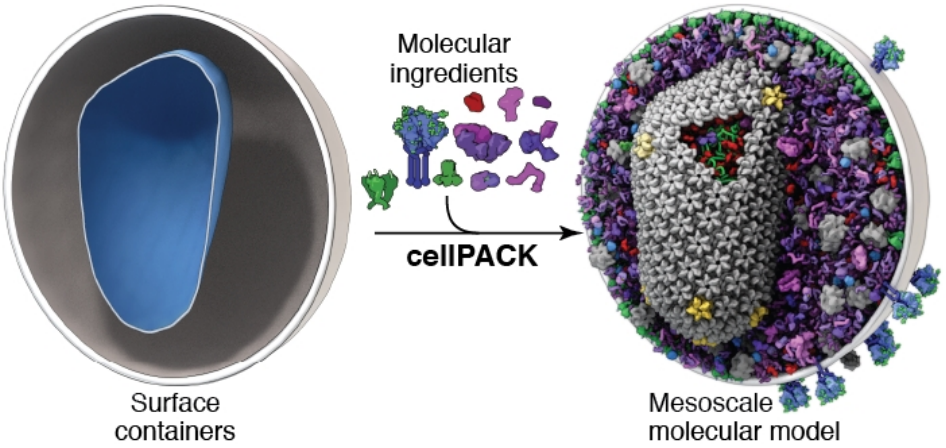


Figure 2.2: The principle of cellPACK, by courtesy of Ludovic Autin. Firstly, recipes are designed based on available information, such as protein structures, concentrations, distribution and compartment shapes. Then the software reads the recipe and generates a plausible assembly of molecules based on collision constraints, in order to generate a larger structure featuring structural information down to the level of atoms, which would not be possible to acquire otherwise with traditional methods such as X-ray crystallography.

also closes small holes between atoms that are not accessible by small solvent molecules. This method was first introduced to reveal information that is not salient enough with other types of representations, such as the presence of pockets and cavities buried in the protein structure that can potentially host important reaction sites. In scientific illustrations, the shape of a protein is an important aspect to convey as it is tightly related to its function. Therefore the surface or space-filling representations are often preferred, because they communicate shape information more efficiently. Furthermore, these models can easily be stored as polygon meshes, which are supported by 3D animation packages. BioBlender [ACZ$^+$12], Molecular Maya [Cla], ePMV [JAG$^+$11], are examples of plugins for animation packages that were specifically developed to ease the loading and rendering of molecular surface meshes in animation packages.

X-ray crystallography is limited because it cannot capture large and complex structures such as organelles, viruses, or cells in their entirety. Electron microscopy imaging, on the other hand, still does not offer enough resolution to capture individual atoms which make the segmentation task between proteins extremely challenging. So far, only little is known about spatial arrangement of proteins that form greater structures, and their manual modeling would be a cumbersome and time-consuming task. To fill the mesoscale gap between atoms and cells, scientists from the Scripps Research Institute have developed cellPACK [JAAA$^+$15], a tool to procedurally construct large mesoscale structures, such as entire viruses or cells, at atomic resolution. cellPACK incorporates the most recent knowledge obtained from biology to generate these models, such as protein structures obtained from crystallography, concentrations and spatial distribution observed *in vitro*, and 3D shape of compartments acquired from electron microscopy.

They summarize all this data in structural descriptions which they call a recipe, which is then used as input to generate entire models of viruses and cells via a packing method based on collisions constraints. This concept is depicted in Figure 2.2. Their algorithm is designed to progressively insert molecules inside given compartments. They use a spatial partitioning scheme to detect overlapping structures and find an appropriate location to insert new shapes, guaranteeing no overlap with previously inserted elements. As an output, their tool generates a list that contains the position, rotation, and type of all the macromolecules that compose the organism. Additionally, their method also supports packing fibre data, such as DNA, or RNA, which is stored as spline control points in the resulting file.

The initial goal of cellPACK was to generate valid protein ensembles that form organisms and that also contain atomic data in order to serve as input for large-scale molecular dynamics simulations. Furthermore, the generated structures can also be loaded in 3D rendering and animation packages for illustration purposes. These large models are thus highly valuable to us, as they contain complex data that would have to be modeled manually otherwise. They are also publicly available and can be easily updated with the most recent knowledge of cell biology. However, the overwhelmingly large number of elements that may compose these mesoscale structures begin to truly challenge animation packages that were not designed with such constraints in mind. While it is possible to render still images in very high quality, real-time visualization of these models is simply not possible, even with simplified surface meshes. This affects the productivity of those who create the models, as well as those who are using it for illustration purposes, and it also compromises the transition to the next generation of interactive scientific illustrations.

Although the polygon mesh is currently the most common shape representation supported by animation packages and game creation software, it might not always provide the best performance for large and complex datasets. Indeed, the rendering of highly detailed meshes requires an overwhelmingly large number of polygons which can stress the rendering pipeline and video memory usage. Reducing the number of polygons for surface meshes can help improving performance significantly, but it also removes important high frequency structural details, and for larger scenes real-time performance requirements are often still not met. To keep up with the increasing size of atomic datasets, visualization experts developed new cutting edge techniques that do not rely on polygon meshes. Tarini et al. [TCM06] introduced a novel visualization technique inspired from 2D billboards, a popular concept in computer games. The technique consists of drawing camera-facing 2D sphere impostors rather than tessellated 3D spheres for rendering individual atoms. As a result, the drawing of a molecule comprising 1000 atoms, for example, requires only 4000 vertices —4 vertices per atom— to form the camera facing quads, while polygon meshes would require a number of vertices up to one or two orders of magnitude higher. Thus, they are able to interactively render large macromolecules with a much smaller computational and memory footprint.

Shortly afterwards, Lampe et al. [LVRH07] extended the billboard technique by leveraging the programmable GPU rendering pipeline to reduce memory bandwidth usage and GPU driver overhead. Instead of storing the entire atomic structure of a protein on the GPU, they only store the position of amino-acids which are the building blocks of proteins. Since there is a relatively low number of different amino-acid types, up to 20 different types, they take advantage of the multiple occurrences of these elements to reduce the number of overall bytes needed to render a protein. Alternatively, Grottel et al. [GRDE10] proposed to improve the rendering speed of large particle-based datasets by implementing occlusion culling to discard hidden particle chunks from the rendering pipeline based on the depth information obtained in the previous frame. Hence, only the sphere impostors that are guaranteed to be visible will be processed by the graphics pipeline, thus greatly increasing the rendering performance for dense particle datasets.

Lindow et al. [LBH12] subsequently presented a novel approach which relies on ray-casting instead. For each protein structure, they store the individual atoms in small and fitting 3D grids and upload the protein grid on the video memory. Upon rendering, they first draw the bounding box of the grid, and subsequently, in the fragment computing program, they cast a ray for each fragment in order to find the first hit with an atom sphere. The ray-tracing is thus performed locally for each macromolecule rather than globally for the entire scene, which means that ray-traversal routines could still be executed for proteins that are occluded and non visible in the final result. Their method supports rendering of very large structures with up to several billion atoms. Mesoscale landscapes usually feature a high number of individual proteins that share the same structure. In order to spare video memory usage, which is usually restricted in size on graphics device, they also use the principle of instancing. Instead of storing every atom of the scene on the video memory, they only upload the position and rotation of individual proteins to the video memory and upload unique protein structures only once.

Falk et al. [FKE13] extended this approach by introducing depth-based occlusion culling and used simpler grid traversal schemes to reduce computing for proteins that are located far away from the camera. They reported being able to render sparse cytoskeleton datasets for an entire cell, with up to 25 billion atoms at 3.6 fps on modern graphic hardware. While the presented methods only support the van Der Waals representation, a few techniques were also developed to improve the rendering of large and highly detailed molecular surfaces using GPU computing and efficient supporting structures instead of meshes [KSES12] [PV12] [PB13] [KGE11] [SI12]. However, none of these surface-based methods is yet able to compete in terms of performance with most recent van der Waals rendering methods presented by Lindow et al. [LBH12] and Falk et al. [FKE13].

2.2 Visualization of Biological Systems

Systems biology is the branch of biology concerned with computational or mathematical modeling of complex biological systems. The organization of biological systems spans several scales; on the level of single cells they typically describe signalling or regulatory

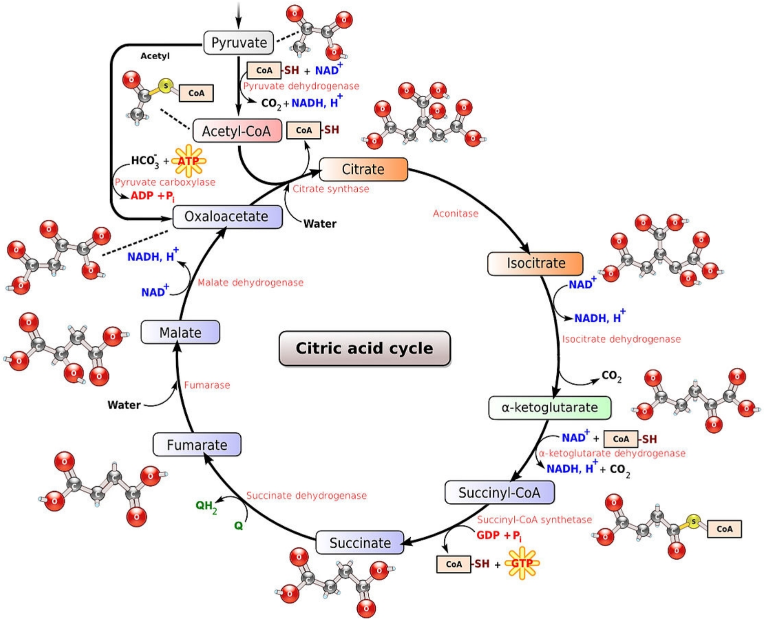


Figure 2.3: Pathway reaction cycle describing the process of energy production during aerobic respiration and which takes place inside mitochondria, also known as TCA or Krebs cycle [Kre08]. This type of procedural description informs us about the type of elements participating in metabolic processes and their role. Initially, these descriptions are used to build models and run scientific simulations. Alternatively they could also be utilized to generate the scenario of explanatory animations to inform the public.

functions of living cells, such as energy production, gene expression, and ability to divide or die. Such systems consist of a reaction network between molecular agents such as enzymes, metabolites, or proteins. The reaction network is denoted as pathway, an example is provided in Figure 2.3. Based on the pathway description, scientists reproduce the dynamics of a system *in silico*, via simulation tools, and observe the changes in species concentrations over time. The results of the simulation are then further analysed to predict and understand how these systems change over time and under varying conditions, and potentially develop solutions to health issues. The complex reaction networks are usually described with a custom markup language, such as SBML [HFS+03], and used as input for the simulation tools. Similar to protein structures, the system descriptions are often shared with peers via public online databases. Biologists have developed several methods to simulate the dynamics of a system. Depending on their modus operandi the modeling approach can either be deterministic or stochastic. Models may also feature spatial information or be purely quantitative.

Quantitative modeling, also known as kinetic modeling, relies on the use of differential equation systems to compute the species concentrations at a given time and is therefore deterministic. Results only vary according to the initial conditions such as concentrations and reaction rates that are predefined in the model. Additionally, the models may also feature spatial details such as location of a species in a subregion of a cell. This approach was the first one to be introduced and still remains very popular among systems biologists because it is reliable and computationally inexpensive. Another type of modeling is agent-based modeling. This method differs greatly from the strictly mathematical approach used in kinetic modeling. It aims at reproducing the original reaction-diffusion behaviour of biochemical agents in three dimensional space and is therefore stochastic. This technology was primarily developed to simulate and understand complex migration pattern among animal or human populations. The concept was then transposed to study the behaviour of chemical species as more capable computer hardware became available and affordable. With agent-based modeling, actors of systems are virtually represented as a 3D points in space and subject to constant random motion based on diffusion speeds observed in vitro. New elements are introduced or removed according to individual reaction events. Reaction events are triggered based on local proximity of potential reaction partners and reaction probabilities based on the reaction rates observed *in-vitro*.

Software, such as CellDesigner [FMKT03], TinkerCell [CBS09], and VCell [MSS$^+$08] are designed to facilitate the research process by providing a unique solution for modeling, simulation, and data analysis in a single framework. These tools usually cover non-spatial models (quantitative modelling), except VCell which also supports the use of an external agent-based simulation modules called Smoldyn [AABA10]. At this stage, scientists studying these models have very limited ways to see how these mathematical models of physiology behave. They can interact with the model by specifying input parameters to the simulation and the resulting visualizations are often time-concentration plots. Even when the simulation method produces spatial information that can be visualized, such as particle-based modeling, these tools will generally favour highly abstracted visualizations which expert users prefer and understand. Therefore, with such visual form, it is hard to relate the models to what is visually observed in wet-lab experiments. In interdisciplinary physiological sciences this might hamper communication of results. However, the underlying data present in the models contains thorough dynamic descriptions of how these biological systems work. These models inform us about the species present in a system, their quantities, location, diffusion speed, reaction partners, and reaction rates. When associated with corresponding structural information this data could potentially help to digitally reproduce an illustrative and dynamic model of a cell.

Biology, medicine, and other sciences can strongly profit from a visualization of physiology in order to gain, verify, and communicate the knowledge and the hypotheses in this field. While the visualization of spatial trajectory data is often not crucial for the study of metabolic pathways, in specific cases such as signalling pathways for example, such visualization might be informative to scientists that are interested in observing the spatial distribution of small signalling molecules over time. Therefore, a few specific tools were developed to allow three dimensional visualization of particle trajectories obtained from agent-based simulation results. CellBlender [cel16] is a software conceived as a plug-in for the 3D animation package Blender, which allows to model, design, and visualize

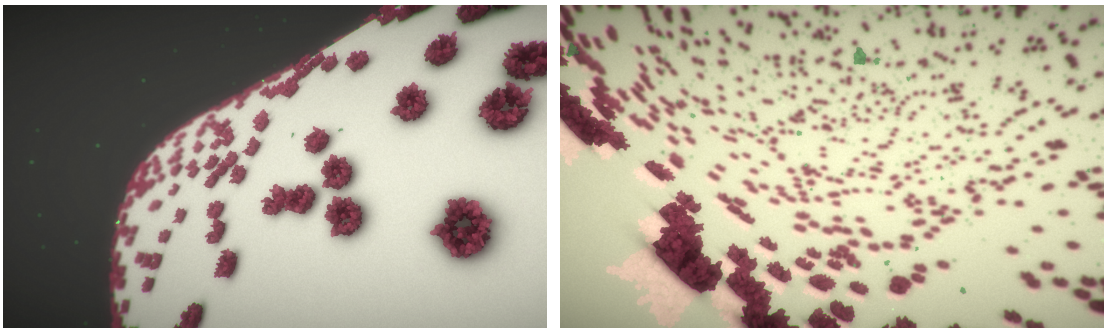


Figure 2.4: Playback of the trajectory data modeled with MCell, and rendered in cellVIEW. The model was only designed for demonstration purposes and depicts a mitochondrion from outside (left) and inside (right). The model features channel proteins diffusing on the outer membrane of the organelle (red), and small ATP molecules (green) exiting the organelle through the channel proteins. The process which only is partially depicted here is the production of energy (ATP) in the core of the organelle that is then released outside the matrix in order to be consumed elsewhere in the cell.

particle-based models computed with MCell [KBK$^+$08]. The cell compartments of a given model are represented as 3D meshes and can be modeled or loaded via the Blender interface. Via the custom interface of the tool, expert users can specify the model parameters, such as the species types, initial quantities, and diffusion speed.

MCell also supports 3D and 2D diffusion models for the particles. 3D diffusion is applied to elements diffusing freely inside a volume, while 2D diffusion is applied to elements that are embedded in a membrane and only diffuse along the compartment surface, such as channel proteins. Users must then input the reactions of the model by specifying the participants, the products and the reaction probability. Particles diffusing in 3D are also able to diffuse outside their initial compartment, and these crossing events must also be defined. The user interface also features a multitude of advanced parameters to fine tune the modeling. Finally, the user specifies the duration of a single step in nanoseconds, and the desired number of steps. The duration of one simulation step will determine the precision of the simulation. MCell then runs the computation offline based on the model properties previously set up in CellBlender, and produces large files that contain trajectory data for each single particle and for the given number of simulation steps. The trajectory data is then converted to a key-frame particle animation format which is readable in Blender. The simulation may then be played back for real-time exploration or rendered in movies. Additionally, it is also possible to use custom meshes to show the shapes of the molecules.

Although this type of modeling technique was invented for scientific purposes, we envision that the generated data could also be utilized to digitally reproduce the functioning of cells for explanatory purposes. Indeed, the resulting visualization carry important information as it allows the depiction of complex biological systems in the form of 3D diffusion and reaction animations, which can also be embedded in their environment, see Figure 2.4. Falk et al. [FKRE09] developed a framework to playback particle trajectories with additional overlaid information to trace the history of individual particles, such as trajectory and previously undergone reactions. A direct approach to visualizing raw trajectory data, however, may often result in an overly cluttered view due to an overwhelmingly large number of elements diffusing randomly in every directions, and may often be close to incomprehensible, even for expert users. To provide a clearer overview of the spatial information, Falk et al. [FKRE10] followed up their previous work and proposed a novel volume-based representation of the agents density to better observe migration pattern of a selected species. An advantage of this approach is that it significantly reduces visual clutter and highlights important spatial properties much more efficiently. However, this approach was only designed for a certain type of domain users, rather than for the laymen. Although it may help reducing overall visual clutter, it also removes individual particle behaviour, which would be crucial to showcase in order to ensure that the underlying information, i.e., the actual function of each actor of the system, is perceived by the viewer.

CHAPTER 3

Rendering and Composition of Molecular Landscapes

Up to this point, the rendering methods presented in the visualization literature have reached unprecedented levels of performance, in terms of size of supported datasets and rendering speed, thus enabling real-time rendering of large molecular structures generated with cellPACK [JAAA$^+$15]. The most recent presented solution is capable of rendering 25 billion atoms at 3.6 fps in HD resolution. However, the rendering approach fails to provide a comfortable user experience, as one expects between 24 and 60 Hz on average for interactive entertainment, and more than 75 Hz for VR content. The rendering should also leave enough resources free for eventual additional computation, such as the physics simulation of the molecular bodies for real-time animation, for example. Moreover, none of the techniques mentioned above have proved to efficiently support other types of molecular structures that exhibit a more complex organization, such as lipid membranes, nucleic acids or fibres, which ought to be taken into account for a precise depiction of molecular landscapes. These are indeed more challenging to render, because the assembling blocks of these structures are considerably smaller and also more numerous than with protein data.

In order to provide a truer depiction of micro-organisms and improve real-time user experience, we decided to investigate new rendering approaches that would address all these limitations. A shortcoming of the volumetric approach presented by Lindow et al. [LBH12], is that for each rendered macromolecule, additional expensive ray-traversal routines are required during the per-fragment processing, which may unbalance parallel thread execution and cause considerable bottlenecks, especially with dense scenes composed of small individual macromolecules. We opted for an impostor-based method for the design of our rendering pipeline, which requires more individual per-fragment thread execution per macromolecule, but is far more balanced and suitable for parallel processing and also does not require expensive volume sampling operation upon rendering.

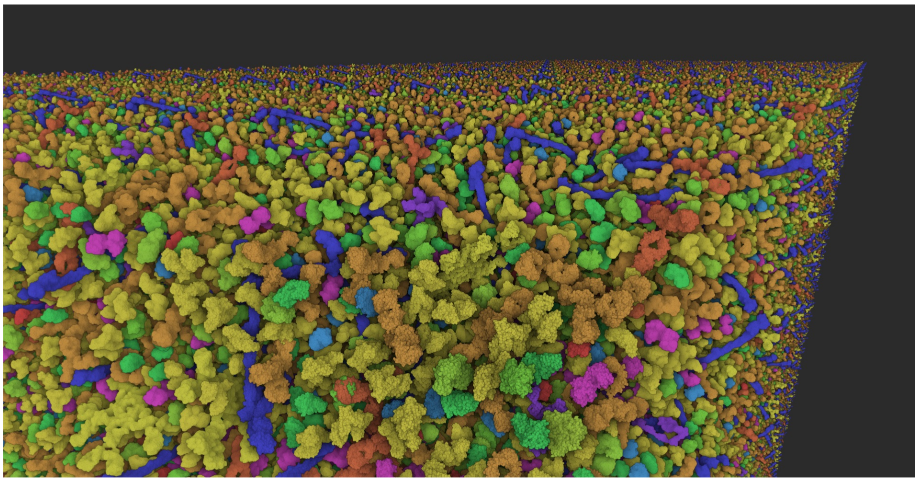


Figure 3.1: Rendering benchmark achieved with a virtual dataset made out of 250 smaller blood plasma datasets, and comprising up to 15 billion atoms in total. The software is capable of rendering the entire dataset at 60 frames per seconds, and from any point of view in the 3D scene. The level-of-detail scheme ensures a fast refresh rate when looking at the dataset in its entirety, and the occlusion queries optimize the performance when closely looking at a sub-region of the dataset.

The pipeline that we have designed directly follows-up the work of Lampe et al. [LVRH07]. It relies on GPU computing and aims at minimizing GPU driver-overhead caused when issuing too many draw commands to the GPU. We also optimized the rendering by adapting well-known computer graphics techniques, such as level of detail, instancing, and occlusion culling, allowing us to render up to several billion atoms with a steady 60 Hz refresh rate, see Figure 3.1. The rendering solution which we developed, dubbed cellVIEW, was implemented with a popular game engine, and the rendering pipeline was embedded in the core of the engine. The project received help from the researchers of the Scripps Research Institute, namely Ludovic Autin, who provided support to import cellPACK models in cellVIEW. Results of the visualization of cellPACK models with cellVIEW are shown in Figure 3.2.

A particularity of the scenes generated with cellPACK is that they aim at reproducing molecular crowding which can be observed *in vitro* and results in dense concentrations of macromolecules. So far, state-of-the-art methods that are able to render up to several billion atoms interactively have only showcased protein datasets with a low population density, which means that less macromolecules are present in the viewport at once. Unlike previous work, cellVIEW was specifically designed to render large scale and realistic datasets featuring accurate protein densities. Besides increasing the computational complexity of the rendering, denser scenes may also cause major occlusion issues because important internal structures, such as DNA for example, may be hidden by surrounding elements. Hence, we also propose a custom scene composition pipeline to adjust the visibility of proteins, while preserving important contextual information.

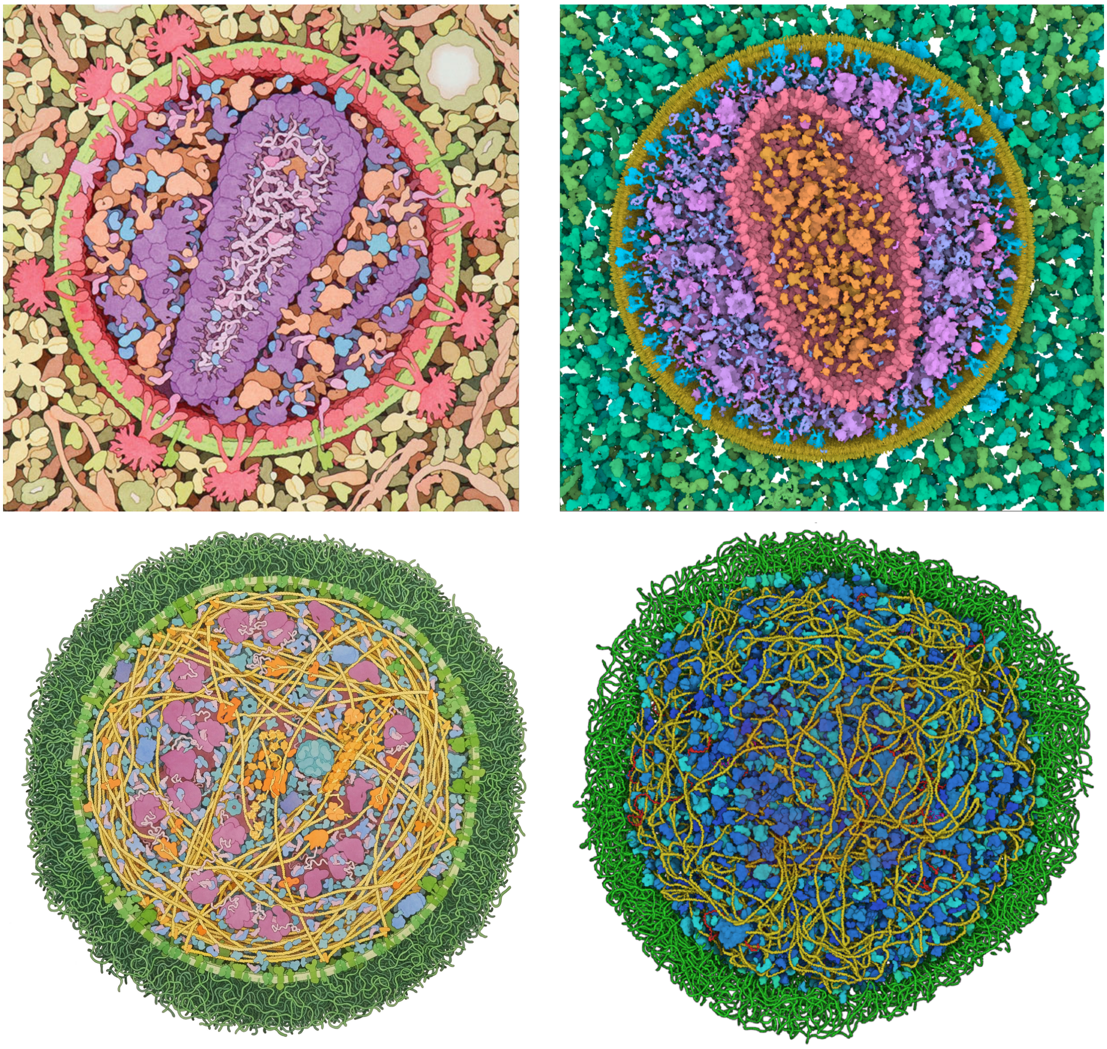


Figure 3.2: Handmade Illustrations of David S. Goodsell (left) next to images rendered with cellVIEW and captured in real-time. The depicted organisms are the HIV surrounded by blood plasma proteins (a) and Mycoplasma (b), which is a small bacterium. The datasets were modeled with cellPACK and are still work in progress, the RNA contained inside the HIV capsid, for example, is not yet present, as well as the bi-lipid membrane of Mycoplasma. Real-time rendering is very helpful during the modeling process because it allows to quickly verify and validate of the models, which renders the modeling process much less cumbersome. Additionally, it becomes less time consuming to create illustrations since the system also allows us to quickly compose the scene with the visibility equalizer to ensure that important elements are visible without manually modifying the position of the macromolecules.

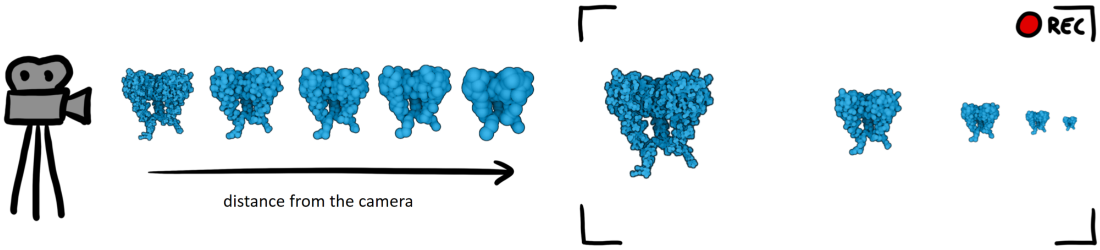


Figure 3.3: The level-of-detail concept (LOD) applied to macromolecules. Using a clustering algorithm, simplified atomic structures are generated, which comprise fewer and larger spheres instead of atoms, called *meta-atoms*. Impostors are used to generate the spheres and thus the computational footprint for rendering the proxies is greatly reduced, down to only a few dozen of spheres for the most simplified proxies.

3.1 Level of Detail

Level of detail (LOD) is a method often used in computer games to cope with the limited polygon budget of real-time applications. The principle consists of drawing simpler mesh representations for distant objects since they have a lower pixel coverage compared to objects nearer to the camera, and therefore less details can be shown. From an original 3D model with a high number of polygons, proxy models are generated to create an atlas of meshes with gradually simplified geometries, as visually explained in Figure 3.3. Upon rendering of a model, the LOD proxy is then selected based on the distance to the camera.

This concept was applied to molecular visualization by Parulek et al. [PJR+14] who presented a continuous level-of-detail scheme for molecular surface rendering. Molecular surfaces are useful for scientific exploration of cavities and pockets. Because computing the surface for dynamic molecular datasets in real-time is expensive, they propose to restrict the computation of high resolution surface details to a subset of the macromolecule located near the camera. For the most distant regions they simplify the atomic structure to reduce the computation times, which also reduces high frequency surface details. They use clustering algorithms to simplify the atomic structure of a protein with fewer and larger spheres, which we refer to as meta-atoms. We also use clustering to simplify protein structures, but we directly render the spheres resulting from the clustering as 2D impostors instead of computing the surface. Clustering allows a reduction of the number of spheres from 75% for the first LOD proxy up to 99% for the most simplified proxies. We also use different shading materials for original atomic structures and the proxies. For proteins closer to the camera and showing the entire atomic data, we highlight the surface details using high-frequency illumination. For proteins located further away that are showing only simplified structures, we only highlight low frequency shape details to make the meta-atoms less salient and the overall shape smoother.

A naive rendering strategy would be to issue a single draw command per macromolecule.

However, each draw command will cause a small latency due to GPU-driver overhead, regardless the number of rendered spheres. When dealing with complex scenes, the accumulation of the GPU-driver latency would cause a severe bottleneck, which would simply forbid real-time rendering. With legacy GPU-instancing, one may group proteins sharing a similar structure and LOD proxy in a single draw command. In most complex use-cases, however, there may be up to several thousand different macromolecule types, and half of dozen of LOD levels. Issuing that many draw operations would thus unnecessarily compromise the efficiency of the rendering pipeline. Therefore, we developed an optimized rendering pipeline which is able to render an entire set of macromolecules with different structures and LOD proxies in a single draw command, thus removing GPU-driver entirely.

3.2 Instancing

Instanced drawing is a concept widely used in computer graphics that aims at reducing the memory bandwidth and footprint, as well as reducing the GPU-driver latency caused by a large number of draw commands. This concept was applied by Lampe et al. [LVRH07] who used the geometry shader stage to instantiate the atomic structure of entire amino acids, also called residues, directly from the GPU rasterization pipeline. Amino-acids are the building blocks of proteins, and there are around 20 different types of amino acid. They initially store the atomic structures of each residue type in the video memory in a dedicated buffer, and they also store in separate buffers the position, rotation, and type of all amino acids that compose a protein. With their method, they are able to render an entire residue with a single initial per-vertex operation. During the vertex shader execution, amino acid properties such as position, rotation, and type, are read from the video memory and passed on to the next shader stage, i.e., the geometry shader. The geometry shader program then fetches the local atom positions for the corresponding residue from the video memory, which are then transformed with the residue position and rotation. For each residue atom, new triangles are injected around atom centroids, and then processed in the final per-pixel shader stage to form 2D sphere impostors, similarly to Tarini et al. [TCM06]. It is also possible to launch the execution of multiple vertex shader programs in a single draw operation, thus reducing the latency accumulation caused when sending multiple rendering commands to the GPU.

Given a protein composed of 4000 atoms and 250 amino acids, the memory footprint of a protein would thus be reduced from 16000 32 bits numbers (4000×3 floating-point numbers per atom position, and one integer for the type) to 2000 numbers (250×7 floating-point numbers per amino acid position and rotation and one integer for the type and excluding residue atoms, which may be are reused by other molecules). Although there is a finite number of amino-acid structures, there also exists an infinite number of possible rotational conformations, that may often change along a single protein chain. Therefore, it is rather challenging to accurately depict protein structures with this approach. To address this limitation, it would be preferable to apply the same concept of instancing to entire proteins instead of single residues. It would also help reducing the memory footprint of the scene as only the type, position, and rotation would suffice to describe a single protein (only 8 numbers needed per protein). However, the geometry shader causes a considerable latency as the number of injected geometries increases, and therefore it is only possible to instantiate a few dozen triangles efficiently with this method. Indeed, a single geometry shader program is responsible for transforming and injecting multiple geometries, but this operation is performed in serial, while it would be more efficient to distribute the workload across multiple threads instead.

The tessellation shader is a feature that is now available on most recent graphics hardware, and is also available on high-end mobile devices. Similarly to the geometry shader, this shader stage is designed to dynamically inject geometries in the rendering pipeline from a GPU program. While the geometry shader is only designed to inject a few dozens of triangles maximum, the tessellation shader is able to efficiently inject thousands of triangles by dispatching the work on multiple threads. Initially, the feature was developed to selectively increase the level of detail of sub-regions of meshes, based on the camera distance and surface curvature. The tessellation engine is also able to inject other types of primitives than polygons, such as lines or vertices. We use this feature to dynamically control the number of spheres to be rendered from the GPU program executed in the rendering pipeline. Similarly to Lampe et al. [LVRH07], only one initial vertex shader thread is required to draw an entire molecule, but the maximum number of vertices that can be injected per thread is up to two orders of magnitude greater with our approach.

The vertex shader, which is the first stage of the pipeline, is used to fetch protein information such as position, rotation, and type, from the video memory, and these will then be passed on to the next stages of the pipeline. Based on the protein type and distance to the camera, we select the corresponding LOD proxy, which informs us about the number of spheres required to draw the given protein. We then notify the tessellation engine about the number of vertices that we want to inject, i.e., one vertex per atom or meta-atom. The tessellation engine will then dispatch the execution of a thread batch comprising one thread per vertex. Each thread is initially given a unique ID, which is used to fetch the corresponding sphere information from the main video memory, such as the local position and radius of atoms or meta-atoms. After transforming the local sphere position with the protein position and rotation, the sphere centroid is passed on to the next shader stage, i.e., the geometry shader. Subsequently, the geometry shader is used to inject the remaining vertices around the sphere centroids to form the triangles of the 2D sphere billboards. The tessellation shader supports injecting of up to 4096 vertices maximum at once, and when combined with the geometry shader the pipeline is able to generate up to 8192 triangles from a single initial per-vertex program. An advantage of this technique is that it allows to render an entire scene comprised of several hundreds of thousands of proteins in a single draw command regardless of the protein type or LOD proxy, thus approaching zero driver overhead.

3.3 Occlusion Culling

Upon rendering macromolecular landscapes, a large number of geometries are processed through the rendering pipeline, which may overload the graphics hardware and cause bottlenecks. Per-fragment processing is, by far, the most computationally demanding stage of our pipeline. Because the scenes are very dense, a large number of spheres overlap each other at a single pixel location, which in turn results in an overly large number of concurrent depth-tests thus slowing down the rendering. An efficient way to reduce this computational load is occlusion culling, and is also widely used in computer games. The principle of occlusion culling is to exclude objects that are partially or completely hidden by other visible objects from the rendering pipeline, in order to spare computation. There exists on modern graphics hardware, a fixed functionality called hardware occlusion queries (HOQ), which allows computing the number of visible pixels of rasterized geometries based on an input depth texture. The depth texture may either be generated beforehand (depth-only pre-pass), or simply recycled from the previous frame.

Grottel et al. [GRDE10] presented an efficient rendering pipeline for large particle-based datasets, using HOQ to discard large chunks of the data that is hidden in the current viewport. They use a uniform grid to spatially partition the dataset and perform occlusion queries for each grid cell using cubes as bounding geometries. Each query result is individually read back to the application after a certain latency, which is due to the processing in the graphics pipeline. Additionally, since one draw command must be issued for each single query, the GPU driver overhead causes a small latency that accumulates with the number of queries. For the specific use-case demonstrated by Grottel et al. [GRDE10], this approach seems reasonable because of the relatively low number of queries needed (one query per grid cell with a $15 \times 15 \times 15$ resolution). In our use-case, we would need to perform one occlusion query for each individual macromolecule present in the scene. Averagely complex scenes generated with cellPACK may easily comprise up to millions of individual instances when small lipid molecules are present, and therefore the presented approach is guaranteed to perform poorly with such scenes.

A more suitable approach would have to be specifically designed to reduce execution times of individual queries and also to limit GPU-driver overhead. Hierarchical Z-Buffer (HiZ buffer) is a method introduced by Green et al. [GKM93] that is commonly used to perform visibility look-up's efficiently. Based on an input depth map, they compute mip-maps with a custom algorithm in order to only retain the deepest values for each texture down-sampling operation. Hence, a single pixel from a lower resolution mip-map will indicate the deepest value from the entire region covered by this pixel in higher resolution mip-maps. Subsequently, for each object to be queried, they compute a camera-facing bounding square. Based on the pixel coverage and position of the squares, they perform a series of look-up's at a specific resolution in order to minimize the number of texture accesses, which tend to slow down the computation. The use of camera-facing squares as bounding regions allows to cover the entire square with only 4 texture look-up's at the right mip resolution, thus guaranteeing a constant execution time, regardless of object sizes. Finally, in case the depth of a given square is deeper than all of the four values queried, the object is then guaranteed to be hidden and may safely be discarded from the rendering.

Without a proper query batching mechanism, however, the overhead caused by individual queries would simply be too high to allow real-time computation with our datasets. Therefore, we propose to adapt this method to run in parallel on graphics processing units, in order to perform multiple queries simultaneously, and thus speeding-up the operation. Because only one thread execution is needed to compute a single query, we are also able to dispatch the GPU computation of multiple queries in a single command, thus approaching zero GPU-driver overhead. With our most complex scene, we were able to save up to 50% of the rendering computation thanks to batched occlusion queries.

3.4 Fibres Structures

A crucial component of living organisms are fiber and polymer structures, such as DNA strands. These structures store and convey the genetic code of living organisms, and therefore, it is important to convey how they work, which is tightly tangled with how they look. These structures are composed of small monomers, called nucleotides, made out of a few dozen of atoms and assembled in long chains. One approach to render this type of data efficiently would be to use instancing, similarly to protein data, and to store the position, rotation, and type, of each nucleotide on the video memory. However, the large number of monomers contained in the entire genome, and the relatively small atomic size of individual monomers, will likely result in a considerable memory usage overhead that would limit the quantity of displayed information. Moreover, these macromolecules are dynamic entities, and therefore it is important to visualize structural changes over time to illustrate them faithfully. In order to display the trajectory of a molecular dataset, the data must transit from CPU to GPU, either beforehand, or streamed during the visualization. In the case of visualization of large genome datasets, the trajectory information is likely to be streamed on-the-fly to the GPU, because the data contained in all the simulation snapshots might not fit into the video memory. However, even on most recent graphics hardware, transferring data from CPU to GPU is still relatively slow. Although the video memory might be large enough to store millions of instances contained in a single frame, it does not guarantee that sequential simulation snapshots could be loaded and displayed in real-time.

Our solution to limiting the memory footprint and bandwidth of fibre structures is to only use the control points of a spline as input, and to position each individual nucleotide along the spline upon rendering, using GPU computing. There exist tools that can model the structure of DNA strands simply from control points [LO08] [HLLF13]. However, none of these methods are implemented on the GPU; the entire genome data is first computed on the CPU and must be transferred to the GPU in order to be rendered, which may stress the memory bandwidth and usage for very large datasets. The rendering pipeline which we developed for the fibres relies on the tessellation shader to dynamically

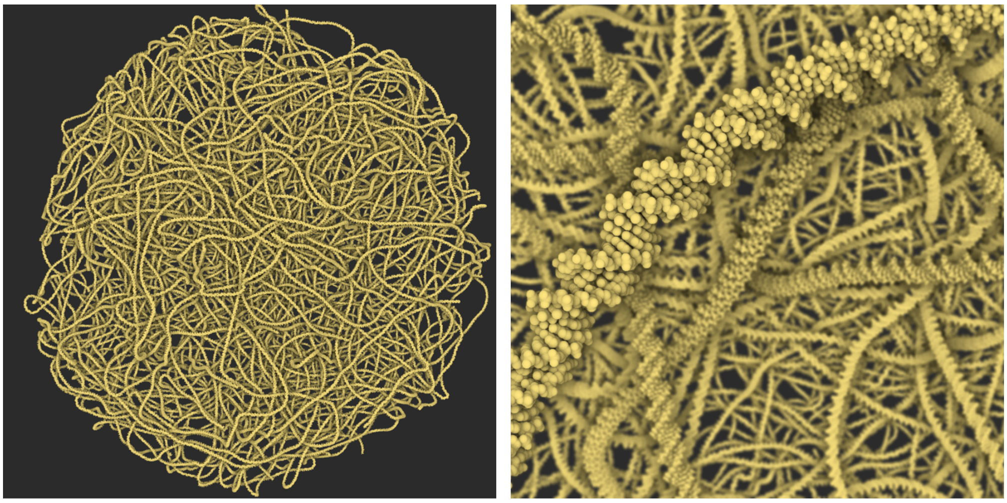


Figure 3.4: Screen capture of the genome of mycoplasma rendered in real-time with cellVIEW at more than 60 frames per second, and comprising over 1 million pairs of nucleotides. The control points of the DNA curve are uploaded to the GPU and the monomers are dynamically instantiated along the spline via a GPU program upon rendering.

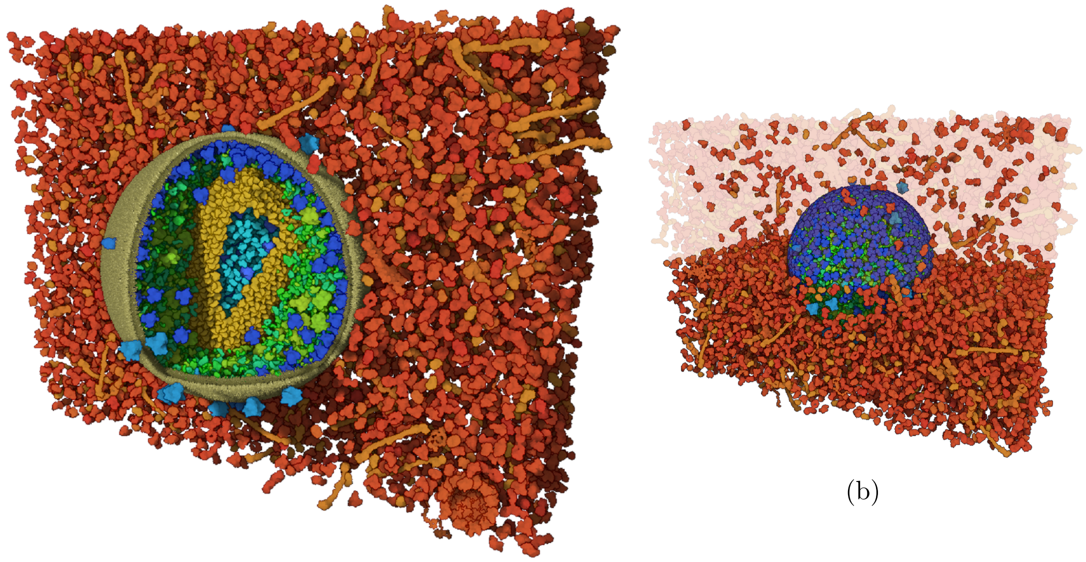
inject atoms of the nucleotides along the fibre curve segments. An advantage of our fibre rendering pipeline is that the previous technologies that we have developed to optimize the rendering of protein data, such as LOD or occlusion culling, may also be seamlessly used with the fibre data. Additionally, since only the curve data is needed as input, the animation of the fibre structures becomes much more trivial to compute. Indeed, mainstream physics animation tools that are GPU-based may be used instead of CPU-based scientific tools, which guarantees a considerable performance boost. In order to support different types of fibres, it is important to first understand how their structures are organized. In the case of DNA fibres, a well-known structure is the B-DNA, which exhibits the recognizable double-helix pattern. The structure is also very simple to model: a spacing of 3.4Å and a rotation of 34.3 degrees between adjacent nucleotides. We also define an average number of 12 nucleotides per segments. Based on this knowledge, we are able to generate long and continuous B-DNA strand, using only control points and building rules as input information.

The pipeline is designed to render one segment of the curve (12 nucleotides) for each invocation of the per-vertex program. The vertex program accesses the main video memory to read the control points information for the current segment as well as the building rule of the current fibre structure. The building rules and the segment curvature are used to compute the position and rotation of the nucleic acids along the curve segment, and this information is then passed on to the next shader stages. Based on the building rules of the segment and its distance to the camera, we estimate the number of spheres required to draw the fibre segment. We then notify the tessellation engine about the number of vertices that we want to inject, i.e., one vertex per sphere. The tessellation engine will dispatch the execution of a thread batch comprising one thread per sphere. Upon processing individual spheres in the tessellation stage, the corresponding sphere information is fetched from the main video memory, such as the local position and radius of atoms or meta-atoms. After transforming the local sphere position with the position and rotation of the corresponding nucleic acid, the sphere centroid is passed on to the next shader stage, i.e., the geometry shader. Subsequently, the geometry shader is used to inject the remaining vertices around the sphere centroids, in order to form the triangles of the 2D sphere billboards. A final render of our fast genome rendering pipeline is shown in Figure 3.4.

3.5 Occlusion Management

The phenomenon of molecular crowding is used to describe a solution when macromolecules are present in high concentrations. Such conditions occur routinely in living cells; cellular interiors are 20–30% volume-occupied by macromolecules, which corresponds to an approximate range of 200–300 mg/mL. The cytosol of E. Coli, for instance, contains about 300–400 mg of macromolecules per millilitre. Therefore, an accurate depiction of the internal structure of cells often results in a very dense arrangement of macromolecules. Without efficient means to selectively remove occluding objects, it would therefore be impossible to observe hidden internal structures that play essential roles in the functioning of biological systems. In scientific illustrations and visualizations, cutaway views are often employed as an effective technique for occlusion management in densely packed scenes. However, a limitation of strict cutaway views is that important contextual information is removed. Illustrators must make sure that the essential information, such as the proportions of molecular agents present in the system, are rightfully represented and not simply clipped away. While mainstream visualization and illustration software feature cutaway tools, they do not provide the means to easily perform advanced scene composition for such scenes.

We propose a novel method for solving occlusion problems due to molecular crowding. In contrast with existing techniques, we take advantage of the characteristics of complex molecular landscapes such as the scenes modeled with cellPACK. These models usually contain hundreds of thousands of individual macromolecules, however, many of them share the same structure. Therefore, by reducing the concentration of elements sharing a similar structure, we may reveal hidden features and also preserve important information such as the type of contextual elements and their location. This concept of a "fuzzy visibility" resembles the “screen-door transparency” technique popular in the early days of computer graphics, and where transparency is achieved by placing small holes in a polygon to reveal what is present behind. When reducing the concentration of macromolecules to reveal structures that are located behind, we also ensure that elements are removed uniformly across the scene to preserve the proportions of the correct spatial distribution. A potential limitation of this approach is that it will communicate erroneous quantitative information to the viewer. In order to avoid misconceptions, we optionally propose to



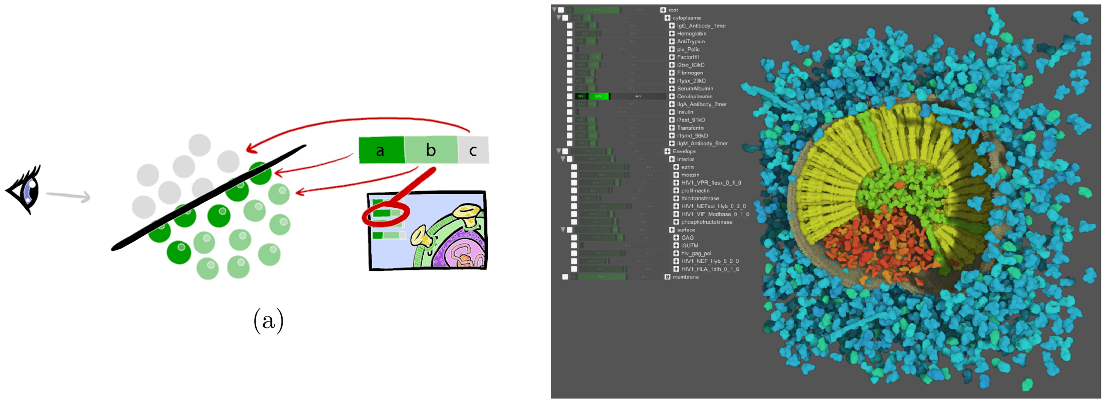
(a)

(b)

Figure 3.5: Cut-awy view of the HIV + blood plasma dataset authored with the Visibility Equalizer. (a) The inside of the HIV capsid (clear blue proteins surrounded by yellow capsid proteins) is set as object of interest, which ensures occluders to be removed dynamically from any point of view. (b) Demonstration of the "fuzzy visibility" concept. Above the manually positioned clipping plane the user can decide to override the concentration of displayed elements. Additionally, we also propose to render the ghost of the removed elements, to reduce visual clutter only the first layer of removed molecules is drawn.

display the removed elements with a ghosting effect such as transparency or contours only. To achieve this effect, discarded elements are first drawn as opaque geometries with the depth-test enabled in a separate texture before being composed on top of visible elements with alpha blending. Therefore, only the first layer of removed elements are shown, which helps minimizing visual clutter compared with full scene traversal transparency. Screen captures of scenes composed with the Visibility Equalizer are shown in Figure 3.5.

To assist the scene composition, we additionally display a bar chart over the view that provides real-time quantitative feedback about the visibility of each macromolecular type. A single bar comprises three distinct regions that indicate the visibility properties for a given species, as shown in Figure 3.6. The grey region bar indicates the number of macromolecules that are currently discarded from the rendering. The dark green region indicates the proportion of rendered molecules that are currently visible, while the light green region indicates the proportion of them that are currently occluded by other macromolecules. The bar chart thus provides an overview of the visibility properties that are helpful for scene composition. For instance, if the user wishes to observe a particular species, he will immediately be informed about the quantity of this species



(a)

(b)

Figure 3.6: The concept of Visibility Equalizer. For each group of similar macromolecules a stack of bar is displayed, which informs us about their visibility. The gray bar tells us the percentage of instances that have been clipped, either by a clipping plane or the "fuzzy visibility". The dark green bar informs us about the number of instances currently visible on the screen, and the light green bar about the number of instances rendered but occluded by other molecules. By dragging the bar the user can thus easily modify the visibility of entire groups of proteins and intuitively compose and inspect a given dataset only with a limited set of user interactions.

that are occluded by other structures, and will adapt the visualization accordingly. The modulation of the proportion of visible elements is also done via the quantitative view by simply dragging the edge between the grey and green regions.

Another useful functionality for scene composition is the selection of species of interest. When selecting a particular macromolecular type as species of interest, occluding elements of a different types are then guaranteed to be discarded in order to provide full-visibility for the species of interest. Since this method is view-dependent, it is necessary to compute occluding elements routinely when the camera position is updated. Therefore, we developed a custom culling pipeline using GPU computing to guarantee a smooth user experience even with scenes featuring a large number of objects. Macromolecules that are set as species of interest are first rendered in a separate off-line depth texture and used as input to perform image-based occlusion queries for each of the remaining elements and detect occluders. Additionally to our quantitative approach for occlusion management, we also provide support for traditional clipping objects such as planes, cubes, or spheres that can be placed in the 3D scene manually.

CHAPTER 4

Emulating the Machinery of Life

Our grand vision is to digitally reproduce the complex mechanisms ongoing inside living organisms and expose them to a larger audience. An essential part of this vision is to provide an interactive visualization technology that would allow the viewers to directly interact with the showcased content. Another important aspect is the use of computational biology data to minimize manual creation of digital assets and show what is currently known and where are the borders of our knowledge. So far, we collected static models of entire viruses and cells and developed new methods to efficiently visualize them in real-time with a multi-scale approach. However, to fully accomplish our vision, the next challenge is to provide the means to animate macromolecules, in order to depict the story which is associated with their function. In animated movies conceived for public dissemination of cell biology, the actors of the machinery of life are traditionally animated with standard animation methods such as key frame animation. Because these animations must be conceived and authored manually, the creation process in usually very expensive and time consuming. Furthermore, the work has to be updated frequently if one wishes to keep the material up to date with state-of-the-art research in cell biology. Experts in computational biology constantly produce large amounts of data for their research, which contain valuable information that could also be used to automatize the creation process of animated content.

Structural biology already provides us important information about how things look, e.g., the atomic structures of key macromolecules. They also utilize this information to set-up dynamics simulations (molecular dynamics or MD) and reproduce the physical behaviour of atom in three dimensions, in order to understand the role they play in the machinery of life. Unfortunately, this type of modeling is computationally demanding and is greatly limited in terms of size and length by the power of modern computers. Systems biology, on the other hand, is the branch of biology concerned by the study of complex biological systems using computational or mathematical modeling. A typical systems biology model consists of a reaction network between molecular agents that characterizes its functioning on the molecular level, also known as a pathway. The reaction network describes the complex cascade of reactions that enables a given process, such as gene expression, energy production from nourishment, reproduction, or destruction of a cell. A model also contains quantitative information such as initial concentrations of species and reaction rates that are observed *in vitro*. This information is then used to initialize simulation programs that aim at modeling the variation in species concentration over time. Based on simulation results, experts are able to formulate hypotheses, which they can verify in wet laboratory experiments. Systems biology experts are generally interested in analysing quantitative information and merely concerned about viewing the actual structure of the individual molecular agents that are simulated. However, data produced by these experts contains valuable details about complex processes that could be used to efficiently produce dynamic 3D illustrations and would significantly improve the way visual communication of cell biology is traditionally done. Indeed, these models already contain enough details to procedurally lay out the scenario of an animated sequence, such as which elements should react together, in which region, in which order, and at which rate.

Standard computational models usually employ a strictly quantitative approach, which are computationally inexpensive but do not provide spatial information other than the sub-region of the cell or organism in which elements are located. Particle-based modeling is a computational modeling approach which has recently emerged. The principle behind this technique is very singular as it aims at reproducing the reaction-diffusion behaviour of each individual molecule contained in the system. In order to reduce computational costs, the reaction-diffusion process is extremely simplified compared to traditional molecular dynamics. To summarize this principle, each molecule is represented as a 3D point and subject to a simple random walk motion to simulate the diffusion in a crowded environment, and reactions between agents are triggered upon collision between the bounding spheres of two particles. Particle-based modeling is thus more appealing to us compared to other modeling approaches, as it provides additional particle trajectories that can simply be played back in a virtual scene to automatically generate animated sequences. In this chapter, we investigate the use of systems biology data combined with structural information to generate comprehensive visualization of simulated processes. In the first part, we will show how to overcome a major drawback of playing-back particle trajectory data, which is how to deal with the prevailing chaos that is naturally present in living cells when observing the reaction-diffusion process for a large number of particles. Another limitation of particle-based modeling is that the simulation must be precomputed beforehand due to its high computational cost, which prohibits interactive changes of the simulation parameters during the visualization. Compared to particle-based modeling, quantitative modeling, such as kinetic modeling, has a very light computational footprint. In the second part, we will explore the use of an hybrid approach using particle systems and kinetic modeling to produce a similar visual output as particle-based modeling with a much lighter computational cost, thus enabling interactive changes of the simulation parameters in real-time.

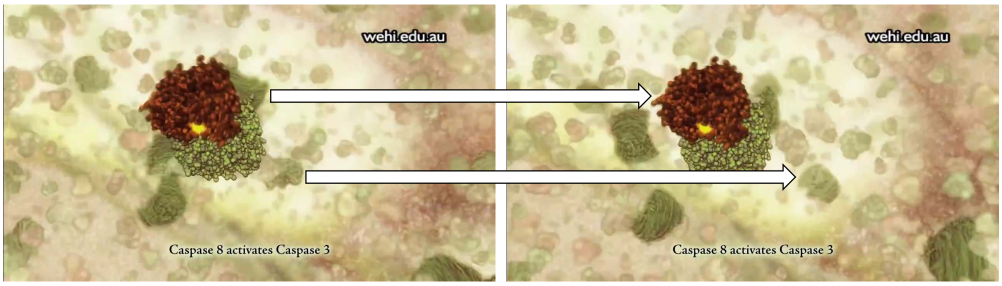


Figure 4.1: Screenshot of frame t and t+3 of a molecular animation *Apoptosis* made by Drew Berry [Ber06]. Mind how the main reaction is forced to remain fairly stationary (upper arrow), while molecules in the background move very quickly (lower arrow). Illustrative timelapse is inspired by this effect, and aims at reproducing it based on real scientific data only.

4.1 Observing Multiple Time Scales Simultaneously

In particle-based modeling, also known as agent-based modeling, the reaction-diffusion is explicitly simulated for each individual particle present in the system. In a single simulation step, the simulation routine performs a random displacement for each particle to mimic the diffusion phenomenon in a crowded environment, and also trigger reaction events upon collision of two neighbouring particles. Although the primary use case for this type of modeling is to provide experts quantitative information, the modeling approach is also capable of producing 3D trajectory data and reaction log as output, which can be used to visualize the underlying spatial information [FKRE09]. Particle-based simulations, on one hand, traditionally operate at a very small frequency, typically of the order of nanoseconds for a single simulation step. The duration of the simulation, on the other hand, is usually several orders of magnitude larger, up to several seconds, thus resulting in a very large number of simulation steps. Hence, there exists a large time-scale discrepancy between the life of individual particles and the life of a given simulated process.

Because of the overwhelmingly large number of simulation steps, it is sensible to only show simulation frames spaced at a constant given interval to reduce the viewing length. This is also referred to as fast-forward or timelapse. A major challenge with the visualization of particle trajectories in fast-forward, is the strong visual clutter caused by the fast erratic motion of individual particles diffusing in every direction. When viewing such results, one can observe the overall motion of swarming particles, but it is impossible to notice individual particles reacting. However, to provide a clear visual explanation of a given process, it is crucial to showcase individual reaction events to illustrate the role of each key macromolecule present in the system. But in order to visualize individual reactions, it is necessary to have a close-up on reacting elements and also to reduce the playback speed to the minimum to allow visual tracking of individual particles. We are therefore facing the following conundrum: we can either observe the entire process in a reasonable amount of time without seeing individual reactions, or we can slow down the simulation and observe individual particles thus increasing greatly the viewing length. A complete overview of the process would therefore require to constantly change the temporal resolution back and forth, as there is currently no technique that would enable us to see two phenomena occurring at two different time scales in a single view. To address this issue, we propose a new type of visualization for particle-data inspired from animated illustrations of cell biology and that respects the following guidelines:

1. The viewing length should be minimum and therefore the trajectory data should be played back in fast-forward.
2. Individual particles should move slow enough to be traceable with human eyes, especially those with a high degree of interest.
3. Events of interest, such as reactions, should be salient and last enough time in order to be noticeable by a human eye.
4. Despite eventual alterations to respect the previous guidelines, the visualization should be as realistic as possible to avoid misconceptions.

The last requirement was formulated in accordance with the results of a study by Jenkinson et al. [JM12], which demonstrated that a more realistic depiction of a process can enhance the viewer's understanding compared to a highly abstracted one, we also observed the same principle in animated movies as shown in Figure 4.1.

4.1.1 Speed Reduction

To reduce the viewing length of a simulation, the trajectories of the particles are played-back in fast-forward. To achieve this, we only display snapshots of the trajectory at a fixed interval. Thus, the spatial information between two displayed snapshots is simply omitted, resulting in larger position leaps between consecutive frames and faster particle motion. Because of the increased speed of the particles, it might be difficult to keep track of individual particles between two consecutive frames, which prevents the viewer from observing key reaction events. In accordance with our guideline rule Number 2, we apply a speed reduction filter to shorten the trajectories of individual particles. The filter mechanism consists of reducing displacements between the current position of particles and their original trajectory positions sampled from the dataset at a given time. The displacements are reduced according to the speed reduction factor as shown in Figure 4.2. Particles thus follow their original trajectories but are displaced over shorter distances and their speed is decreased. As a result, the particle motion is much smoother and easier to follow by human eyes while preserving a fast playback speed. The speed reduction filter also causes a small delay between particles and their original trajectories, however, because of the chaotic nature of the Brownian motion particle, this delay is hardly noticeable. Indeed, diffusing molecules travel much slower than entities with a more

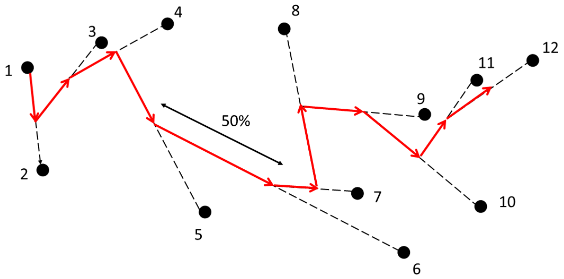


Figure 4.2: The speed reduction principle. The black dots represent the non-filtered trajectory and the red line the reduced one. When sampling the next position from the raw trajectory, we downscale the displacement that would originally move the particle to the sampled position. This approach has the advantage of smoothing the trajectory while causing a minimum amount of latency. The displacement toward the raw-trajectory is downscale by a the speed reduction factor which controls the smoothness of the motion.

linear motion, and therefore even if the particle lags behind its original trajectory it is almost certain to be located in its vicinity.

4.1.2 Reaction Highlighting

In particle-based simulation of biochemical reaction networks, reactions are punctual events that occur upon collision of two potential reaction partners. In the event of a reaction, the resulting products will immediately be injected in the system as the operation is performed in a single simulation step. Therefore, it might be difficult to observe single reactions. Additionally, when viewing the simulation in fast-forward, many important reaction events might simply be omitted due to frame dropping. Without showing reaction events, molecular agents will appear and disappear from the visualization without providing the necessary visual clues to understand what might have caused it. We therefore prolong the duration of the reactions to make them stand out and to inform the viewer about the nature of these events. Shortly before each reaction event, we apply attraction forces to steer the reactants towards the reaction location. During this procedure the original motion of the particle is overwritten by attraction forces. To emphasize reaction events, reacting elements are highlighted with vivid colors to contrast with the rest of the scene. The slow attraction animation and the color highlighting thus guarantees that the viewer is informed that a reaction is about to happen. Once a reaction is accomplished, the products are introduced and their original motion is restored.

4.1.3 Lens Effect

Because of the slow movement of the particles compared to their original trajectories, the viewer may be misguided about the true diffusion properties of molecular agents. To preserve a realistic impression of molecular motion despite trajectory reducing, we

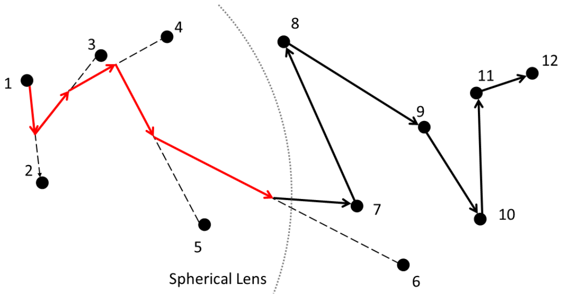


Figure 4.3: The spherical object-space modifies the speed reduction of the particles based on their distance to the centre of a virtual sphere. Inside the sphere the trajectories of the particles are reduced so that the viewer can visually keep track of them. Outside the lens the particles are subject to their original chaotic motion thus conveying a true impression of chaos and stochastic motion.

introduce a temporal focus+context technique that applies visual abstraction of molecular trajectories solely in the foreground (focus), while showing more realistic —yet often untraceable— motion in the background (context). By showing the actual particle trajectories in the background we want to create an illusion of chaos, thus informing the viewer about their real diffusion speeds while allowing him to observe important reaction events in the same view. We propose two methods to achieve this effect. The first method operates in object space and is described in Figure 4.3. We define a region anchored in front of the camera in which molecular agents will be subject to a significant speed reduction. Outside of this region we progressively decrease the speed reduction until reaching the background zone where the original particle trajectories are shown. The second method we propose operates in image space. We limit the displacement of particles to a distance defined in pixel-space and based on the smooth pursuit eye-movement limit used in psychology research. When using a perspective projection in the visualization, elements close to the camera appear to move much slower than those further away in z direction. In fact, their screen space velocity is approximately the same, this illusion is due to the differences in pixel coverage of particles that are located in the background. This effect results in a smooth and continuous transition between the motion of front and background particles, which is more tedious to obtain with the world-space approach because it requires additional fine tuning.

4.2 Quantity-Driven Particle Behaviour

In systems biology, there exist various modeling approaches that are either deterministic, such as quantitative modeling, or stochastic, such as agent-based modeling. One advantage of agent-based models, as discussed above, is the presence of spatial information that can easily be displayed in a 3D view. Although the visualization may often be overly cluttered, we previously demonstrated that it is possible to apply filtering methods to improve the clarity and expose essential parts to the viewer. However, the computational footprint of particle-based models is very high compared to quantitative models, because it must account for the reaction and diffusion of each single reacting entities, which are usually present in large quantities. The simulation is therefore often precomputed prior to the visualization. As simulations are complex to setup and to compute, the creation and exploration of multiple scenario is thus slowed by this workflow. Additionally, with large and complex models, the resulting trajectory files may be challenging, over tens of gigabytes of data for 100000 snapshots and only 5000 individual particles, which is cumbersome to manipulate when dealing with a multitude of different scenario.

A more practical workflow for our use case would be to visualize the particle positions on-the-fly as the simulation runs. With such an approach, it would also become possible to influence the course of a simulation, for instance, increase the temperature, or introduce new species, and to directly observe the impact of these changes without having to run another simulation or deal with overly large files. Most-efficient simulation tools, such as Smoldyn [AABA10], are able to compute single simulation steps in a matter of seconds, thanks to efficient parallel algorithms running on GPU. A few tools additionally support *in-situ* visualization of particle-based models computed in real-time with such an approach. However, the computation is still too demanding to deliver a smooth user experience even on high-end commodity hardware, which prohibits the use of these tools for interactive and explanatory applications.

Additionally, to visualize and understand a biological system, one must be able to observe the logical cascade of reactions described in the reaction network, as depicted in scientific animations. With particle-based modeling, it is impossible to influence where and when reactions will occur. Therefore, the viewing of an entire cascade of reactions may be very tedious because it requires manual spatial exploration and waiting times between successive reactions. To overcome this issue, we would have to gain control over where reactions might occur. This can only be achieved by decoupling the quantitative information from the spatial information. In other words, we would need to use a modeling approach that would allow us overwriting spatial information without influencing the course of the simulation. Quantitative modeling methods, such as kinetic modeling, do not feature particle trajectory data. They are also much faster to compute and could easily be simulated along with a complex 3D visualization with high frequency refresh rates. To address the previous limitations of particle-based modeling, we thus chose to explore the use of quantitative information to drive 3D particle animations in real-time.

With particle-based modeling, the behaviour of individual particles directly influences the concentration of species over time, as reactions are only triggered by collision events. Given a reaction $A+B \Rightarrow C$, for example, when two reactants A and B are closely located next to each other, then a reaction is likely to occur. In the event of a reaction, elements A and B will then fuse to produce element C, thus increasing the concentration of element C and decreasing the concentration of elements A and B. In particle-based simulations, individual agents are thus actors of their own fates. We propose a new type of particle system where the behaviour of particles is purely driven by quantitative information. We

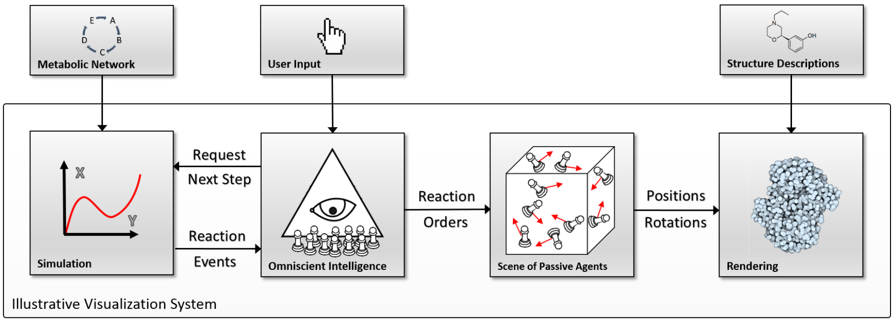


Figure 4.4: Workflow of the passive agent-base system. A biological process is represented as a metabolic network model, which is numerically simulated. The controlling module, called Omniscient Intelligence, (OI) routinely reads the results from the quantitative simulation, and dispatches reaction events to the agents. Agents are diffusing in the 3D scene, passively waiting for reaction order from the OI. Upon reaction events the target agents are steered towards each others and forced to react. Finally the position and rotation of the molecules are passed to the rendered which renders the set of particles according to their structural information. Via user input it is also possible to interact with the simulation parameters and affect the outcome of the visualization.

introduce a new type of agent, whose behaviour mostly depends on external parameters, and which we dub *passive agents*. As the quantitative simulation advances, the system will query the number of reactions to occur at a given time, and subsequently force passive agents to react. Hence, passive agents do not play any role in the simulation process, their only function is to act as a visualization proxy to show quantitative information in three dimensions. While the visual output is hardly distinguishable from an actual particle-based simulation, the benefits of this approach is two-fold. Firstly, the simulation of the reaction network is very lightweight to compute, which makes this approach much more usable for interactive applications. Secondly, the spatial information is completely decoupled from the simulation, which means that we can have control over where reactions will occur. This approach grants us the freedom to dynamically trigger a cascade of reactions directly in the viewport in order to facilitate storytelling of complex reaction networks. This would not be possible with an agent-based simulation.

The workflow of our passive-agents system is shown in Figure 4.4 and is laid out as follows: On the one hand, we have a biological system described as a reaction network and modeled with a scientific simulation software using a kinetic modeling approach. On the other hand, we have a 3D scene filled with potential reactants, and which is set according to the initial conditions of the model, such as the initial quantity and location of each species. Each passive-agent additionally undergoes a 3D random walk motion to simulate the diffusion process, similarly to traditional particle-based simulation. We then routinely query the quantitative simulation to fetch information about the number of reactions that are meant to occur at time t. Assuming a reaction of type $A + B \Rightarrow C$, for example, the particle system will first randomly select two elements A and B based on their spatial proximity. The two particles will then be subject to attraction forces until they enter in collision. Upon collision of the reactants, the reaction is performed, products are injected and the reactants are removed from the system. Finally, the reaction products return to a normal diffusion behaviour until they get chosen for a subsequent reaction defined in the network. By default, the reaction participants are randomly chosen in the 3D scene, but it is also possible to set a reaction priority to a few elements present in the viewport, in order to show cascades of reactions without having to browse through the entire scene.

CHAPTER 5

Conclusion

The initial aim of this PhD project was to develop a visualization platform that would keep the audience and multidisciplinary experts informed about state-of-the-art knowledge in cell biology, using interactive 3D visualization as means of communication. We intended to use as much available scientific data as possible to reduce the amount of manual work traditionally needed for creating digital assets for such content, and make this enterprise possible only with a small team of researchers. We also aimed at providing interaction, which is not usually present in traditional visual communication of cell biology, although it is clear that it would have a strong and beneficial impact on the learning experience. Therefore, the presented techniques have a strong emphasis on GPU-computing in order to deliver a responsive user experience with high-frequency refresh rates. To achieve this ambitious goal, we first identified and isolated the most challenging parts of the pipeline and focused on addressing them individually. In some cases, these challenges where addressed by improving current state-of-the-art techniques, such as large scale molecular rendering, but we also presented novel concepts to solve visualization problems that have not yet been investigated in the context of cell biology, such as the concepts of Omniscient Intelligence, Illustrative Timelapse or Visibility Equalizer.

5.1 Summary

We first proposed a novel technique to render large sets of proteins, while preserving atomic-scale details when zooming in. Previous methods utilized the concept of instancing, but also relied on ray casting, which fails to deliver high frequency refresh rates. We proposed to utilize the rasterization pipeline instead because it features a powerful tessellation engine that allows us to dynamically control the level of detail and reduce GPU-driver overhead. Fiber structures, such as DNA for example, are present in large quantity inside living cells, but tend to have a more complex structure than proteins. Therefore, as a follow-up, we extended this technique to support these structures. We also increase the rendering speed by developing an occlusion culling scheme inspired by techniques traditionally used in graphically advanced computer games.

Up to this point, the scenes used to showcase rendering techniques were made up for benchmarking purposes and not realistically modelled to reflect the correct density of proteins present inside living cells. Our technique has proven to deliver fast frame rates, even when using realistic whole-cell atomic structures provided by the researchers of the Scripps Research Institute, and modelled with cellPACK. These realistic models aim at reproducing a natural arrangements of proteins inside living cells and viruses, which are subject to molecular crowding and therefore densely packed next to each others. As a result, when displaying the entire datasets, only the most frontal atoms are visible, thus hiding important structural information buried internally. We also proposed a novel solution to solve occlusion problems when visualizing whole-cell structural models. We utilize the fact that many macromolecules present inside the cells actually share a similar atomic structure to reduce the concentration of displayed elements of a given type, and clear the view to reveal hidden structures. While the exact number of displayed elements is overridden, important information about the location of these structures and the proportions are preserved.

So far, we have only addressed challenges that are concerned with the visualization of static datasets. However, to truly express the function of these structures, we must animate them to communicate where they travel and how they interact with their surroundings. In computational biology, there is a modeling technique called particle-based modeling that can reproduce biological systems on the scale of entire cells while providing spatial information such as 3D trajectory and reaction history. When directly viewing the results of such simulation, it is very difficult to observe the behaviour of individual particles because of the large number of elements present in the system, and because of their fast and erratic diffusion motion. A solution to this problem would be to increase the temporal resolution of the visualization to observe individual particles, and to constantly go back and forth between two temporal scales to observe either the simulated process or single particles. To improve the visualization of multi-scale processes, we propose a novel time-lapse method that is designed to visualize the results of a simulation at the temporal resolution of the system, while applying a speed reduction filter to particles so that their motion can comfortably be perceived by human eyes. The filtering is only applied in the focus region while the original motion is preserved in the context area. This partition helps to preserve the original impression of chaos and prevent us from misguiding the viewer. This illusion was first introduced by the scientific illustrator Drew Berry in one of his animated films [Ber06], we then formalized this concept and transposed it to scientific visualization.

A major downside of particle-based modeling is the large computational footprint, which obliges us to precompute the simulation before visualizing it. This constraint prohibits us from providing a truly interactive experience where the viewer would be able to interact with the simulation parameters and instantly observe the changes. Kinetic modeling is a strictly quantitative modeling approach, which is much faster to compute, but does not provide spatial information. In order to utilize kinetic modeling to generate 3D spatial animations, we invented a new type of particle system that is able to convert the quantitative information into a spatial representation enhanced with structural information. We introduced the notion of passive agents, which unlike traditional agents, have their behaviour dictated by the omniscient intelligence, a controlling mechanism that utilizes the quantitative information to dispatch reaction order to the agents. The particle system relies on GPU computing to routinely perform diffusion motion displacement and neighbouring search, allowing us to emulate the behaviour of several million molecules in real-time, while providing a similar visual output as genuine particle-based modeling. Since spatial data and simulation are decoupled, we thus gain control over what particular elements are going to react and when. With particle-based modeling, this behaviour is stochastic, which means that when focusing on a given particle, we have no guarantee that the particle will undergo a complete reaction cycle described in the pathway in a reasonable amount of time, which is impractical for story-telling. With passive agents, we gain the ability to focus on a single element and to force it to undergo an entire reaction cycle, thus facilitating the viewing of relevant physiological events.

5.2 Lessons Learned

In the course of this thesis, we have learned that biological entities tend to feature a large number of elements sharing a similar structure, and that we can often use this specificity to our advantage when addressing visualization challenges. For example, cells are composed of multiple sub-cellar entities, such as Mitochondria or Golgi apparatus, which are present in multiple instances. The building blocks of cells are organelles are the proteins, in the case of Erythrotytes (red blood cells) for instance, about ⅔ of their structure is made out of haemoglobin proteins. The internal structures of proteins reveal long chains of smaller links, called amino acids, and it only exists a few tens of different types, which are mostly composed of 6 distinct atomic elements (C, N, O, H, S, Se). The presence of so called "building block", or structures (n) repeated several times (N), is thus a phenomenon that is occurring on several magnification levels, from atoms up to entire cells. In our contributions, we utilize this "n « N" specificity in the following contexts:

Rendering: we can render one object many times, we can use instancing and reduce the GPU driver overhead resulting into an immense speed-up if n « N.

Memory Management: instancing naturally reduces the memory footprint for storing the same object only once instead of multiple times, which is crucial for rendering a scene using consumer level graphics hardware.

Occlusion Management: we can utilize this characteristic when we want to cut away some part of the data. We can remove some instances of a particular building block as long as we keep few of them in the scene so that the viewer understands their spatial distribution in the cut-away area from the visible ones.

Visual Story-telling: we can tell a story about a process using any structural building block, we do not need to use a particular one and and follow the course of a realistic particle behaviour that may not always reveal important information. This gives us freedom for designing explanatory storytelling visualization techniques.

5.3 Future Work

The models employed to showcase the dynamics of biological systems are only simple yet realistic demonstration models, which do not truly compare in complexity with the large systems that we initially wanted to showcase such as en entire and functioning E.Coli bacterium. Although we showed that the techniques we designed are able to support real scientific data, we yet have to combine structural and physiological composition for entire cells instead. This ambitious goal could only be fully achieved with a tight collaboration with biology experts. To facilitate the collection and validation of information from different online sources, it would also be interesting to develop automated means to collect this information and compile it into a meta-model comprising both structural and procedural information. Hence, once new information is available, the system would be able to scan all the sources and update the model accordingly, thus saving us from tedious and cumbersome manual operations.

We also need to figure out how to scale upward with the models that we are able to render. So far, we are able to render viruses or prokaryotic cells such as HIV, Mycoplasma mycoides or Escherichia Coli. Eukaryotes are the cells that compose living organisms such as plants and animals, therefore it is important to explain their mechanism to understand how we work. These cells, however, are much bigger than prokaryotes as they contains membrane bound organelles such as Mitochondria or Golgi apparatus, which may be as large as a bacterium and are therefore much more challenging to render. To render these larger structures we would need new ways of representing shapes other than spheres while preserving the massive zooming continuum. We would also need to work even closer with the scientists to help them gather and extract important information from the raw data they collect.

An important aspect which we have not yet explored and would be worth investigating in future work is accurate real-time collision detection for large molecular datasets. Although particle-based modeling or passive agents provides us the 3D trajectory of macromolecules, these approaches only consider molecules as a simple 3D point and the notion of shape or collision with surrounding bodies is simply ignored. Thanks to the democratization of graphics hardware it is now possible to perform large-scale rigid-body collision-detection for up to several million of individual bodies, in real-time and on commodity hardware. It would be worthwhile to integrate such system with the visualization of molecular landscapes, not only it would correct overlapping bodies when viewing dynamic data, but it would also allow us to generate large structures similarly to the ones modelled with cellPACK but in real-time. Indeed, the computational footprint of cellPACK does not allow the creation of models in real-time, however, with such technology we would be able to overcome this limitation and to progressively model a sub-region of a cell that is visible in the viewport as we navigate though the scene. This would also allow us to discard structural information which is not currently visible and address the memory space limitation of graphics hardware, thus enabling the visualization of even larger structures than the ones currently supported with cellVIEW.

Another aspect which is worth investigating is dynamic story-telling to visually explain biological networks that are usually described as cryptic diagrams and only comprehensible by domain experts. While we have successfully accomplished the task of showing how elements of a system move and interact with other elements, it is now clear to us that showcasing only a realistic view of a system might not suffice to efficiently and visually communicate this type of information. With the concept of Omniscient Intelligence, we have shown how to distort the normally stochastic behaviour to present a complete reaction cycle in a reasonable amount of time. We now have to start thinking about new methods to morph this approach with more traditional 2D information and possibly provide intermediate views and transitions between realistic 3D representation and static 2D reaction diagrams.

Part II

Publications

PAPER A

Illustrative Visualization of Molecular Reactions using Omniscient Intelligence and Passive Agents

Published in [LMPSV14] by the Eurographics Association.

DOI: 10.1111/cgf.12370
Eurographics Conference on Visualization (EuroVis) 2014
H. Carr, P. Rheingans, and H. Schumann
(Guest Editors)

Volume 33 (2014), Number 3

Illustrative Visualization of Molecular Reactions using Omniscient Intelligence and Passive Agents

M. Le Muzic[1], J. Parulek[2], A.K. Stavrum[2], and I. Viola[1,2]

[1]Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria
[2]Department of Informatics, University of Bergen, Norway

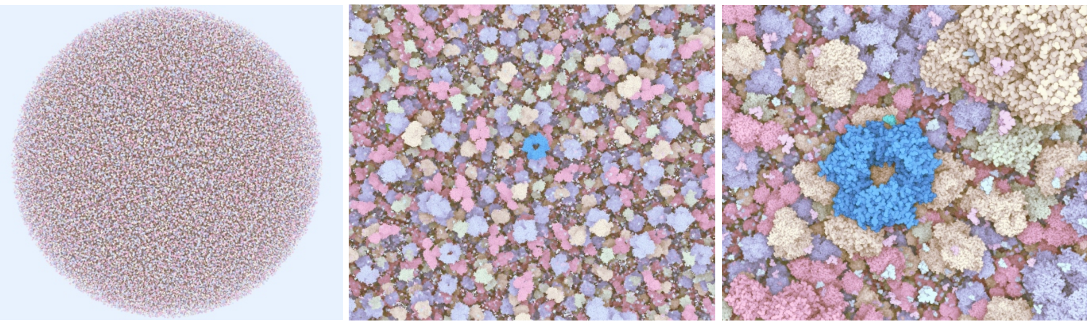


Figure 1: *Demonstration of our system capabilities from three different zooming levels (left to right). We showcase a scene containing* 10^6 *diffusing and reacting molecules in real-time at 30 FPS.*

Abstract

In this paper we propose a new type of a particle systems, tailored for illustrative visualization purposes, in particular for visualizing molecular reactions in biological networks. Previous visualizations of biochemical processes were exploiting the results of agent-based modeling. Such modeling aims at reproducing accurately the stochastic nature of molecular interactions. However, it is impossible to expect events of interest happening at a certain time and location, which is impractical for storytelling. To obtain the means of controlling molecular interactions, we propose to govern passive agents with an omniscient intelligence, instead of giving to the agents the freedom of initiating reaction autonomously. This makes it possible to generate illustrative animated stories that communicate the functioning of the molecular machinery. The rendering performance delivers for interactive framerates of massive amounts of data, based on the dynamic tessellation capabilities of modern graphics cards. Finally, we report an informal expert feedback we obtained from the potential users.

1 Introduction

Biochemistry is difficult to understand without any visual explanation. For this reason, processes, such as cell division, DNA transcription, or DNA repair are often being communicated via animated movies. In comparison to static representations, animated movies allow explaining more efficiently how the physical appearance of molecules influences the functioning of the process. To produce such movies, scientific illustrators employ 3D animation packages along with real scientific data. They are using structure descriptions to design the visual aspect of molecular compounds. They also have tools to ease the import, animation, and rendering of those structures such as ePMV [JAG*11] and Molecular-Maya [mol13].

Although structural information is needed to explain the form of a molecule, it does not convey explicit information about its function. This information has to be added by the illustrator manually through complex tasks, such as

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

delivered by
EUROGRAPHICS
DIGITAL LIBRARY
www.eg.org
diglib.eg.org

key-frame animation. Consequently, the creation process of animated content is time-consuming and expensive, taking up to months or years.

Our goal is to improve visual communication in the field of biochemistry by generating illustrative visualizations of molecular reactions involved in biochemical processes. Computational biology provides description of structural and procedural models replicating the function of biological processes. We suggest to merge these two data sources and to automatically produce visualizations communicating both structure and function of the molecular machinery.

Current techniques in mesoscale visualization already provide ways to visualize biochemical processes by exploiting the results of agent-based simulations. However, those techniques are heavy to compute and challenging to represent at the level of individual molecules. Indeed, due to the stochastic nature of the simulation and visual complexity, it is challenging to keep track of the reactions describing a biochemical process. The resulting visualizations manage to show realistic views of a given process, but they do not really succeed at communicating its function.

We propose a new procedural abstraction technique that selectively distorts the reaction-diffusion process, to visualize the function and structural characteristics of the studied metabolic system. In contrast to direct visualization of agent-based simulations, our technique shows reactions events directly in the viewport and composing explanatory animations sequences related to the visualized process. Just like cutaways or exploded views visually abstract the structure to convey how things look, our technique visually abstracts the process to convey how things work. Below are the two main contributions we present in this paper in order to achieve our goal:

- A novel particle system approach using passive agents, controlled by omniscient intelligence.
- A novel rendering technique for instant visualization of a vast amount of molecules.

2 Related work

We structure the prior work review along our two contributions, namely the abstraction of processes and structure.

2.1 Visualization of Biological Networks

Visualization is essential for the analysis of metabolic networks as it provides a clear picture of relationships among metabolites. It is often employed when e.g., depicting pathways, which are stored in the KEGG [KG00] database. Traditional visualization in the domain of biology is based on graph representations. These allow for step-wise analysis of a particular process, but also graph aspects such as centrality, cardinality, degree, etc. The visualization community has contributed to enrich pathway visualizations by, for example, defining requirements for pathway visualization [SND05] and by depicting the pathway in the context of related information [LPK*13].

Agent-based simulations provide the means of representing the dynamics of metabolic networks in their natural embedding of the 3D world. They compute the positions of particles that are supposed to mimic a realistic behavior of the metabolites. By exploiting the results of agent-based simulation software, such as ChemCell [PS03] or Smoldyn [AB04], it is possible to produce videos or even real-time visualizations of metabolic processes. The output of such visualizations, however, lacks any means for focusing on events of interest.

Falk et al. [FKRE09] presented a tool which reads the results of an agent-based simulation and enables interactive visual exploration of the results. The aim of their visualization is to describe the process of signal transduction on both mesoscopic, and molecular level. The individual molecules are represented in 3D space as spherical glyphs, and their positions are updated over time according to the value stored in the agent-based simulation. The tool also allows the user to track specific particle inside a cell. The trajectory is represented as a trail in 3D space providing information about directions and reactions.

The visualization of the raw agent-based simulation using individual particles, however, suffers from high visual complexity. The large number of displayed particles and their chaotic organization, due to diffusion motion, makes the understanding of the visualized processes difficult. To tackle this issue, Falk et al. adopted a volume-based rendering approach [FKRE10], using aggregation to convert the particle data into a density volume, which is then rendered via ray casting. This visualization filters out the prevailing chaos and offers a more intuitive representation of the general particle motion.

A more recent work on visualization of agent-based simulations by de Heras Ciechomski et al. introduces a system for designing and visualizing cellular models [dHCKMK13]. Their visualization framework aims at providing a biophysics research and exploration tool within a 3D computer-game environment. The tool allows the user to render the results of an agent-based simulation with 3D representations of the corresponding molecular structures employing ray-tracing. The rendering module, however, does not take any advantage of GPU computing and does not achieve real-time performances.

Reviewed approaches of individual-based visualization of biological networks are all aimed at the same goal, which is to give insight on how biological processes actually work. Whether we see videos or real-time visualizations, the process is usually not, or only partially, understood because of the stochastic nature of the simulation results. Even when tracking single elements and bringing them into focus, there is still no guarantee that interesting events will happen. One can, for instance, use volumetric rendering to visualize the spatial distribution of particular molecular quantities, but this does not provide enough details about the process itself, such as the key steps along a reaction chain, known as pathway. We address this problem by providing automated means for illustrative visualization of the molecular reactions involved in a given pathway.

2.2 Visualization of Molecular Structures

The second aspect we relate our work to is interactive molecular visualization. Visual and geometrical molecular representations are covered extensively in the scientific literature. They are traditionally employed for analytical tasks, such as binding site analysis, where surface-based representations are utilized, e.g., solvent excluded surfaces [GB78], minimal molecular surfaces [Ede99], or surfaces based on the summation of Gaussian densities [Bli82]. Nevertheless, these representations are often difficult to compute and are tailored towards the analysis of small molecular compounds. Instead, we employ a representation that is frequently used by cellular modeling systems [FKE13], the so-called Van der Waals (VdW) representation. It represents each atom of a molecule as a set of spheres.

To speed up rendering of spheres, Tarini et al. presented a real-time algorithm for visualizing molecules by means of VdW representations [TCM06]. They employed 2D sphere billboarding for rendering individual atoms of molecular systems. Daae Lampe et al. [DVRH07] have introduced an even faster rendering of VdW atoms by exploiting billboards of proteins. The molecules were generated utilizing geometry shaders. In our approach, we go one step further by exploiting the tessellation shader to generate the entire molecule, not limited to the proteins as it was the case in the technique proposed by Daae Lampe et al.

Most VdW molecular models refer to the original work done by David Goodsell [Goo03], who has developed a simplistic, but expressive style for representing molecules through space filling. This illustration approach has been recently adopted by Falk et al. [FKE13] to depict large mesoscopic cellular models. Their scene is visualized via ray casting performed efficiently on the underlying grid structures. Additionally, each molecule is stored in its own supporting grid, which is then traversed in a level-of-detail (LOD) manner. The authors initially achieved at least 3.6 frames per second (fps) for scenes with 25×10^9 atoms. The work of Falk et al. has been a follow-up approach to a technique proposed by Lindow et al. [LBH12]. The ray casting was performed after the bounding boxes of all molecules were rendered. Again, a supporting grid storing the molecular atoms is still required, leading to at least 3 fps for 10^9 atoms. We employ an optimized approach using a straightforward LOD scheme, but importantly, we are not bound to any grid-based supporting structure. Our rendering technique forms visualization elements on the fly, which additionally allows us to alter the positions of molecular atoms dynamically.

Our aim is to provide an environment for creating automated illustrations, yet there are also studies in the literature, which address more scientifically oriented schemes for solving and visualizing immersive molecular simulations. A powerful framework for visualizing large-scale atomistic simulations was presented by Reda et al. [RFK*13]. They employ a ball-and-stick model combined with volumetric surfaces to convey the uncertainty in molecular boundaries. Moreover, they utilize GPU-based ray-casting, implemented in an immersive CAVE environment, providing the means to render atomistic simulations of millions of atoms. Recently, a CPU-based solution for visualization of large molecular simulations was proposed by Knoll et al. [KWN*13]. Again, the ball-and-stick representation is combined with volume data to provide a compound visualization. Moreover, an efficient preprocessing scheme allows to perform fast ray-casting implementation on multicore CPUs. Stone et al. [SVS13] introduce a visual system to analyze petascale molecular dynamic simulations. The system employs GPU compute nodes and VMD scene data structure to visualize molecular surfaces obtained from very large biomolecular complexes. In comparison to these systems, our work is realized through causal PCs with modern graphics card supporting tessellation shader capabilities. Moreover, our focus lies on illustrative aspects while the actual simulation analysis is excluded.

3 Passive Agents and Omniscient Intelligence

Scientific illustrators are explaining biology through visual storytelling, e.g., a sequence of events illustrating a reaction chain along a metabolic pathway. In order to produce illustrative visualizations of metabolic processes, we would also need to show reaction events in a story-structured manner. So far the tools and techniques developed to visualize such processes were solely based on agent-based models, and reactions are only initiated according to probabilities and collision events. Due to the stochastic nature of the reaction events, we cannot predict the time nor the location of these events, which is clearly impractical for storytelling. In this section we describe how our system is capable of selectively distorting the reaction-diffusion process, to convey the function and structural characteristics of the studied metabolic system. A schematic overview of the system is also given in Figure 2.

3.1 Overview

Instead of using agents as simulation entities, we propose to use passive agents to reflect the dynamics of quantitative simulations. According to the agent classification given by Kubera et al. [KMP10], passive agents are entities that can undergo actions, but not initiate them. In our system, passive agents are unable to start reactions autonomously, they can only receive reaction orders from an omniscient intelligence (OI), which controls their behavior. Figure 3 shows a com-

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

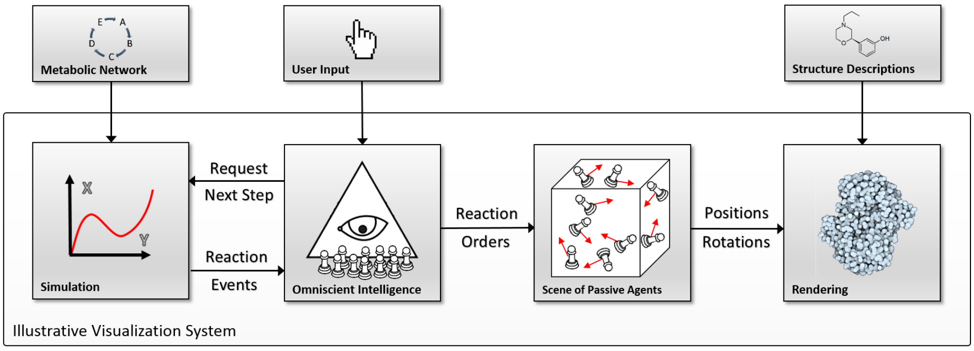


Figure 2: *Overview of our system: A biological process is represented by a metabolic network model, which is numerically simulated. The Omniscient Intelligence module routinely reads the results from the quantitative simulation, and dispatches reaction events to the agents. The final stage of the pipeline represents elements with their structural models and renders the entire scene at interactive framerates.*

parison between a fully stochastic system, and a controllable system.

The OI is tightly coupled with the quantitative simulation, which runs in parallel. The simulation module reads and simulates procedural descriptions of biological processes. Its role is to provide information about the number of reaction events for a given reaction type in real-time. These results are then used by the OI, which is responsible for triggering reaction orders of the passive agents. More details about the quantitative simulation are given in Subsection 3.3.

Agent-based simulations result in motion of chemicals that is hard to observe by viewer when displayed directly. Either it is too fast to perceive it, or when the simulation is slowed down to observable speed, the probability that the viewpoint will display a reaction is extremely low. Therefore, we showcase a motion of molecular compounds which does not represents the results of a simulation. Instead, we employ an illustrative type of molecular motion that is perceivable by human observer, while showing a scene where reactions are frequently happening.

When triggering a reaction, the OI chooses reaction partners according to their spatial vicinity and drives the reaction using steering forces. The OI also chooses the reaction candidates to assure that reactions are well distributed in space. We provide technical details concerning the OI in Section 4. Additionally, when reactions occur, the random motion of particles is blended with the reaction steering forces. This prevents linear trajectories, which would otherwise result in unnatural motion (Fig. 4).

We render our passive agents according to molecular structure descriptions. Additionally to the structure, we assign to the molecules distinct colors in function of their species type. We developed a custom rendering pipeline capable of displaying large amounts of structures at interactive framerates, which we describe in details in Section 5. Additionally, we highlight the reacting elements by increasing the color intensity and also display text-based information about the different actors of ongoing reactions in the focus.

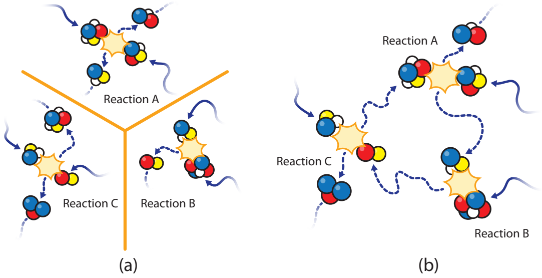


Figure 3: *Comparison between a fully stochastic system (a) and our system (b). In a fully stochastic system we can observe reaction happening in the scene at random locations, and without any follow-up according to the pathway. On the other hand, in our system it is possible to visualize reaction immediately and to build reactions sequences.*

3.2 Story Composition

At any time, the user may set focus on any molecule shown in the scene. Once in focus, the camera starts following the actor. Moreover, the actor is prioritized over other molecules of the same type to undergo the next reaction. This gives us the freedom to trigger a reaction at a desired location, which we use to build molecular stories depicting chains of reactions.

After the reaction of the focused element is completed, the focus is automatically shifted to one of the products. This facilitates tracking of the reactions described in the pathway. It happens often in biological networks that one reaction produces several products which are susceptible to take part in

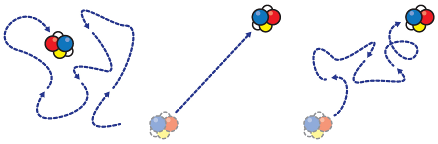


Figure 4: *Description of the reaction motion, we blend a random walk motion (left), with linear interpolation (middle) in order to get a consistent attraction motion (right).*

reactions of different types. This results in forks in the reaction network. The same principle holds for species that are involved in different types of reactions. In such case, the system will either assign a default choice, or requests a decision from the user.

In addition to animations, reactions are also responsible for creating/removing molecules. The reaction dispatching guarantees that a correct number of elements is formed to match with the quantitative simulation. Consistency is essential to preserve for visually communicating the pathway of entire units such as mitochondria or E.coli, for instance. There, the availability of molecular structures will affect whether a reaction can be shown or if not enough chemical elements are available to carry out the reaction.

3.3 Quantitative Simulation

The OI is tightly coupled with the quantitative simulation. Our system uses COPASI API [HSG*06] as simulation engine. The engine reads and simulates biological network files stored in the SBML format [HFS*03]. The SBML format is a bioinformatics format based on XML syntax. It describes, among many other properties, the type of molecular species involved in the process, their initial concentrations, as well as the different reactions which constitutes the process. Below are the reaction categories we show in our visualization system:

1. $0 \rightarrow A$
2. $A \rightarrow B$
3. $A \rightarrow B + C$
4. $A + B \rightarrow C$
5. $A + B \rightarrow C + D$
6. $A + E \rightarrow B + E$
7. $A + E \rightarrow B + C + E$
8. $A + B + E \rightarrow C + E$
9. $A + B + E \rightarrow C + D + E$

where the zeroth order reaction (1) represents the injection of a new element into the system, unimolecular reactions $(2,3)$ involve one molecular entity, bimolecular reactions $(4,5)$ involve two molecular entities, enzymatic reactions $(6-9)$ involve additional catalyzers.

4 Agent Computation

The OI executes two different routines, prior to the rendering, in order to compute the positions and orientations of the passive agents. In this section we provide implementation details about these two routines, which we name respectively reaction dispatching routine and agent updating routine. To be able to handle a large number of elements, we perform the computation in parallel on the GPU, using CUDA. The implementation of our particle system is inspired by the GPU version of agent-based simulation from Dematte [Dem10], where we adjusted his approach to be able to introduce the omniscient intelligence. Large memory buffers are allocated on the GPU memory and hold the individual properties of the passive agents in an Array-of-Structure (AOS) manner.

Each passive agent is composed of the following properties: ID, $Type$, $Position$, $Orientation$, $LinearVelocity$, $AngularVelocity$, $ReactionID$. The ID, is a unique ID given to every single molecule, the $Type$ describes the chemical compound, such as ATP for example. The $Position$, $Orientation$, $Linear$ and $AngularVelocity$ are all physics related properties, and the $ReactionID$ is an ID, which is unique for each reaction and assigned to all the participating structures. The $ReactionID$ of a non-reaction element is always null.

4.1 Reaction Dispatching

This routine ensures the communication between the simulation and the passive agents. At the beginning of the routine, prior to the first call, we uniformly populate the scene space with molecules. The molecules are added according to the initial concentrations defined in the model. The initial concentrations will guarantee that enough potential reaction partners are present in the scene before starting to initiate the individual reactions. Then, on each call, the routine requests the simulation engine to compute a new integration step and dispatches the resulting reaction events to the passive agents.

In the case of a reaction of category (1) $0 \rightarrow A$, new elements are simply inserted at the end of the buffers. All buffers must be sorted prior to this step so that empty slots are kept at the end of the buffers. Other categories of reactions (unimolecular, bimolecular and enzymatic), work on a similar basis. Given a single reaction of category (4) $A + B \rightarrow C$, the modus operandi is the following: First, the OI selects a candidate of type A from the scene. To select an element of given type efficiently, the buffers are previously sorted according to the type of the molecule. The sorting is done using fast GPU sorting operations [BH11] and must be repeated for each routine call. We also keep track of the number of molecules of each type on the CPU. This allows us to determine the parts of the sorted buffers, which only contain elements of a given type. Since elements are randomly distributed in the scene, we consider any element of the buffer to have a random location. Therefore we simply select the first element of the section containing only elements of type A to undergo the reaction. Figure 5 represents how CPU book-keeping allows us to select elements of a given type from the GPU sorted buffers.

Once the first element of the reaction is selected, the scene

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

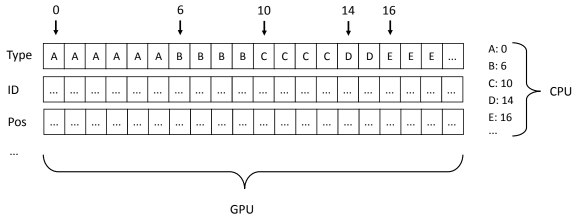


Figure 5: *Representation of the Array-of-Structure allocated on the GPU memory, each index value corresponds to one molecule and can be used to access the properties form the different buffer. On the CPU a book-keeping is constantly maintained in order to access elements of a given type, when the buffers are all sorted according to the molecule type*

is traversed in order to find the closest element of type B. When dealing with a large number of particles it is important to provide means for an efficient lookup into the scene. In order to rapidly search across large scene of elements we employ a GPU-based subdivision technique based on the work of Le Grand [LG07], which was also employed by Dematte [Dem10]. The technique computes a uniform 3D grid of the scene, where each grid cell contains a list of elements located inside. The search is performed by browsing the surrounding cells for the closest partner element. The grid resolution is chosen according to the density of displayed elements. In case the partner search fails, we enlarge the searching radius in order to browse more surrounding cells, and thus improve the chance to successfully find a reaction partner. Once the reaction partner has been found, elements A and B will both have their reaction IDs set to the same ID, and will no longer be available for reactions until their ongoing reaction is over.

4.2 Agent Updating

This routine is responsible for the motion of the agents and must therefore be called every frame in order to guarantee smooth animations. It is also responsible for changing substrates into products at the end of a reaction. There are two types of motion which can influence the trajectory of a given particle; the diffusion and the reaction motion. We apply motion to the agents using physical forces, based on the rigid-body physics simulation scheme introduced by Baraff [Bar01]. Each particle carries attributes, such as position, rotation, linear and angular velocity. The current position and rotation values are integrated using the Euler method.

The reaction motion consists of attraction forces represented as 3D vectors that point towards the direction of the reaction partner. Since the buffers are constantly reorganized due to the sorting operations of the first routine, we need an easy way to group reaction partners in order to apply the attraction forces. We use the reaction ID, common to each partner of a reaction, to sort our buffers. During an ongoing reaction the routine also checks for collision between reaction partners, using a simple collision detection scheme defined by Le Grand [LG07]. Once all the partners of a reaction are in contact with each other, the routine starts a timeout. At the end of the timeout the reactant types are either changed into new ones (products), remain unchanged (enzymes), or are removed entirely from the scene, depending on the type of the reaction. In the aftermath, the reaction ID of the participants is reset, and they may again diffuse freely within the scene and take part in other reactions.

The Brownian motion that brings the molecular machinery into chaotic motion, results from a constant bouncing from the crowd of surrounded molecules. In order to mimic its mechanics, we apply impulse forces in a stochastic manner. We generate random force vectors at regular intervals using cuRAND, which we apply to the particles. This can be efficiently performed in parallel since the diffusion motion of each element is independent from the others. The magnitude of the random force and the size of the intervals can be modified in order to tweak the resulting motion. Since forces are simply represented by directional vectors, it is straightforward to minimize the linear steering motion by linear interpolation of two driving forces.

5 Rendering

Each molecule is visualized by means of the VdW representation, where atoms are defined as spheres of given radii. Instead of having meshes of spheres, we use billboards rendered in the fragment shader. We additionally perform z-buffer correction [DVRH07]. This technique, as it has been demonstrated previously [TCM06, PRV13], significantly increases performance as compared to tessellated primitives. One of the challenging tasks is to render millions of dynamic atoms, $O(10^6)$, molecules involving $O(10^4)$ atoms at interactive frame rates. One potential solution is to upload all the atoms information for each frame to the GPU, which would require a large CPU to GPU transfer bandwidth. In our approach we generate a texture buffer containing all the required atom positions, defined in the PDB file format.

We invoke the rendering of the molecules by using a single vertex as an input, i.e., the center of the molecules. This point is accompanied with the rotational quaternion representing the current orientation of the molecule. The data is obtained by reading from the CUDA buffers used by the OI in Section 4, thus saving transfer cost from CPU to GPU. In the next stage, we exploit tessellation and geometry shaders to emit the individual atoms of the molecules. In an ideal case, the total number of tessellation levels equals the number of atoms. Using this approach we are able to form maximally $4K$ atoms due to current hardware limitations. Therefore, we also utilize the capabilities of the geometry shader that can produce another 64 atoms per output created by the tessellation evaluation shader. As a result we are able to produce up to 262144 atoms per one vertex call. In the tessellation control shader, we exploit isolines as patch primitives. This requires to specify two outer tessellation levels by $gl\_TessLevelOuter[0] = gl\_TessLevelOuter[1] = \sqrt{\#atoms}$. Then in the tessellation evaluation shader each atom id is defined by a combination tessellation variables as follows:

$$\begin{aligned} id \;=\; & gl\_TessCoord.x \times gl\_TessLevelOuter[0] + \\ & + gl\_TessCoord.y \times gl\_TessLevelOuter[0] \times \\ & \times gl\_TessLevelOuter[1]. \end{aligned}$$

Based on this id, the necessary atom is fetched from the texture buffer holding the atom coordinates and the radius.

Afterward, each atom is processed via the geometry shader to generate a billboard. In case when $\#atoms$ exceeds $4K$, we specify $\#atoms = 4K$ and perform the generation of the remaining atoms, multiplies of $\#atoms/4K$ in the geometry shader. For example, when we would like to form $12K$ atoms, each geometry shader pass will produce three atoms, i.e., three billboards, instead of one. We apply also a view frustum culling in the vertex shader to avoid tessellating of molecules outside the viewing frustum.

5.1 Level of Detail

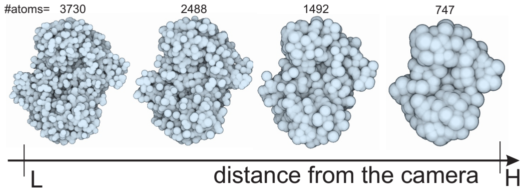


Figure 6: *An example of our molecular Level-of-Detail. With increasing distance to the camera, more atoms are skipped. The radii of the remaining atoms are scaled accordingly.*

When rendering large molecular scenes, there is no need to visualize all the molecules in full detail, i.e., with the full-atom count. Especially, this is evident when the screen space area is occupied with molecules, which span over only few image pixels. This is the case of molecules that are located far away in the current view. We exploit the primary task of tessellation shaders to lower the number of the tessellation levels according to an increasing camera distance. The vertex shader decides on the number of atoms to be generated, i.e., $\#atoms$ parameter in the tessellation control shader. This is achieved by setting up two discrete boundaries, L and H, where the number of atoms being created is interpolated between both boundaries (Fig. 6).

In order to achieve as smooth transition as possible between neighboring LODs, and as well to preserve the molecular structure close to the original form, we propose the following strategy. For each molecule, the atoms are sorted by an increasing distance from the center of the bounding box that encapsulates the molecule. Based on the molecular center and its position within $[L,H]$, we decide on the number of atoms to be skipped. When approaching the H boundary we skip more and more atoms, i.e., rendering only each n-th atom. As a result of this procedure, we remove always the same atoms from the molecule at a certain distance, which keeps a scene view more persistent than when performing removal, for instance, stochastically. Still this can be performed in a more elaborated way, e.g., ordering the molecular atoms by a certain importance criterion, and removing atoms that are less important. Our method is simple and straightforward, and produces visually pleasing results (Fig. 7). In addition to the atom skipping, we scale the radius by a value $k \in [1, max\_scale]$, $new\_radius = radius * k$. Value k increases linearly within $[L,H]$, i.e., $k = max\_scale * \left(\frac{depth-L}{H-L}\right)$. This will close the gaps that can appear after the removal of atoms.

5.2 Shading

Our shading model employs a set of visual effects that enhance shape and depth information. The entire shading scheme is inspired by the approach presented by David Goodsell [Goo09]. We use his system of visual cues, i.e, constant shading, contour and depth enhancement, which he employs in molecular illustrations. We apply these visual cues in the focus and context style, where the focus is represented by a selected pathway. The reacting elements of the pathway are visually enhanced by employing more saturated colors (Fig. 8). On the other hand, the context molecules are displayed via pale colors.

6 Results

The visualization of described particle-based system is showcased on two different molecular pathways. The first pathway is a simplified version of the NAD pathway (Fig. 8). Nicotinamide adenine dinucleotide (NAD) is found in all living cells, and has a number of important functions. It is well known for its involvement in redox reactions, where it acts as an electron carrier. In addition it is used as a substrate for post translational modifications of proteins. When NAD is

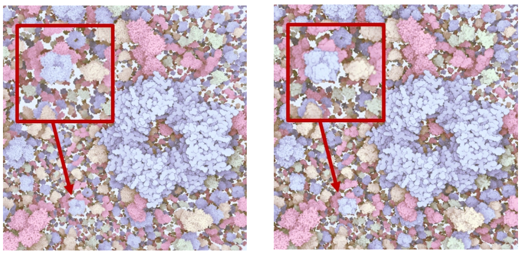


Figure 7: *An example of LOD application. The same scene is rendered without (left,* 43 948 997 *atoms) and with LOD (right,* 7 227 817 *atoms). Notice that for the molecules that are further in the back (the closeups), only negligible differences are visible.*

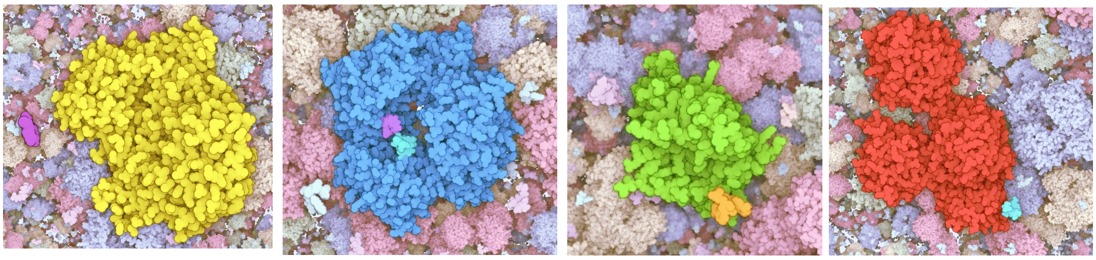


Figure 8: *Snapshots from the NAD pathway. The examples show four extracted frames from four reactions involved in NAD pathway. We see four enzymes (yellow — NAMPT_ATP, blue — NMNAT1_NMN, green — NNMT, red — PARP1) that catalyze these reactions. The camera follows metabolites that take part in reactions (small molecules). The navigation is done fully automatically, where in a case that there are two reaction products a user can interactively select which one to follow.*

Agents	NAD cycle [ms]	glyoxylate cycle [ms]
20000	9.5	10
100000	12	13
200000	16	18
500000	23	25

Table 1: *Performance results of the agent computation (one frame) for the NAD and TCA cycle with approximately* 50 *and* 500 *reactions triggered per second.*

used as a substrate, it gets broken down into nicotinamide (Nam) and ADP-ribose. ADP-ribose is used in various processes, while Nam enters the NAD salvage pathway to regenerate NAD.

The second one is the glyoxylate cycle. The glyoxylate cycle is used by plants, bacteria and fungi to produce carbohydrates. Parts of the cycle overlap with the tricarboxylic acid (TCA) cycle. The split between the two cycles happens when ICL converts isocitrate to glyoxylate and succinate in the glyoxylate cycle, while ICD converts isocitrate to α-ketoglutarate in the TCA cycle. This cyclic pathway allows cells to obtain energy from fat.

6.1 Agent Computation Performance

We provide a performance analysis of the computation of the particles, i.e., the OI routines. We demonstrate the interactive performance of our system using the two different network models, both showing different characteristics such as initial quantities and reaction rates. In this analysis the two models also showcase a different number of reactions triggered by second (approximately 50 reactions per second for the NAD cycle and 500 reaction per second for the glyoxylate cycle). This number simply reflects the differences in reaction rates between the reactions of the two networks. Since the two OI routines are not running at the same pace, we provide an average computation time, accounting for both routines, for a given frame (see Table 1). The performance was measured on an Intel Core i7-3930 CPU 3.20 GHz coupled with a GeForce GTX Titan GPU.

6.2 Rendering Performance

Just like many molecular rendering techniques, the performance strongly depends on the position and orientation of the camera, the size and number of displayed elements, and in our case, on the level-of-detail parameters. Therefore, we limit the evaluation of our rendering to a stress rendering test of a large data set, analogously to the evaluation method as in Falk et al. [FKE13]. We set up a scene containing 4 million instances of large molecules, of four different types, from 2000 up to 12000 atoms each. This gives us an effective number of 30 billion (3×10^{10}) of atoms for the dataset. The molecules are populated randomly in space inside a spherical volume. We center the camera and ensure that the entire scene fits into the screen. The size of the viewport for this measure is 1920 × 1080 (HD). The testing hardware was identical to performance analysis of our particle system. The number of atoms emitted through the tessellation and geometry shaders is actually smaller than the amount of effective atoms in the scene, because of the dynamic level of detail. We render the scene with an average computation time of 80 ms per frame. The rendering speed is approximately 10 fps for a scene containing 30 billions of effective atoms, where 13092630 atoms were actually emitted by the rendering pipeline. When decreasing the number of molecules to 1 million, we achieve 30 fps (Fig. 1).

7 User Feedback

We have demonstrated the outcome of our framework to several experts in the area of biology education, dissemination of biological research, and molecular illustration.

The molecular illustrator had specific reservations to the outcome video we showed him. His critical remarks were directed towards an apparently direct motion of the particles; there was according to him still lack of randomness in the particle motion. This is a valid point, the main reason was to provide clarity when showing reactions, which would not be case when using dense scenes. This critical point, however, can be solved by giving higher prominence to the random walk motion component than to the motion triggered by the reaction event. We have increased the amount of randomness in the accompanying video. The second critical point was related to the vast space our animation was depicting, i.e., the molecular crowding was not present in the video sequence. This can be partially solved by increasing the number of particles in the scene. However, as we at this point do not detect collisions, in a very crowded scene the lack of collisions will lead to visible artifacts in the animation. The collision detection thus will have to be integrated into our particle system to support animations with molecular crowding.

The professor teaching molecular biology praised the entire framework for the provided interactivity. Still, she raised several critical points. In order to be used for learning purposes, the environment could be more physically accurate. Again, we were advised to increase the density of elements. Additionally, we were suggested to perform zooming-in action when a binding event is happening, in order to achieve better focus for the viewer. Another suggestion was to employ metadata, describing the current scene view, pathways, and molecules, which should accompany the 3D view. The major complaint was about the speed of the reactions. She suggested to slow down the animation when a reaction is happening. Many of the aforementioned ideas are very relevant suggestions, and we will consider these in follow-up work on advanced visual guidance in molecular machineries.

The experts on dissemination of biology were on the other hand very positive concerning the demonstration of our new technology. Their current workflow is frame-based and the output is a linear video. They have raised a strong interest in including our system in their workflow and expressed several functionalities that would enable the integration into their work processes. Experts imagine to employ our system in visual communication of intracellular signaling cascades, which are based on the interaction of two or more macromolecules. The most crucial functionality to them seems the exporting feature of spatial and temporal subparts as well as camera paths of the animation into common formats of 3D modeling packages. While their workflow will still include keyframe animation of major actors of a molecular story, our system can demonstrate the contextual environment in which the story is embedded in.

8 Conclusions and Future Work

We have developed a novel concept of particle system, tailored for creation of interactive molecular illustrations. The concept integrates scientific data from structural and systems biology. While autonomous agents do not allow us specifying the visualization intent, our method allows for directing the behavior of passive agents through the newly introduced omniscient intelligence.

Based on the idea of representing the simulation quantitatively with a particle-based visualization, we have proposed a new way to represent the form and the function of a given biochemical process. The same concept could also be potentially translated to a larger family of dynamical systems where visualization might be the right tool to explore and analyze them, such as population dynamics, migration patterns, or crowd simulations.

Our visualization, however, does not allow to display complex processes, such as protein transport, and assumes that the reactions are solely taking place in a single compartment. To be able to explain biochemical process in their entirety we would firstly need to address this issue. Secondly, the collision detection scheme we adopted is certainly too trivial to showcase realistic animation of molecular crowding, which would require collision-detection at the levels of single atoms.

Finally, current means that guide the viewer are based on the simple approach of a camera following the actor and text-based description of the reaction participants. In future work, we wish to integrate graph-based network visualization to our system, in order to provide a deeper understanding of the pathway of a process at the glance.

9 Acknowledgments

This project has been funded by the Vienna Science and Technology Fund (WWTF) through project VRG11-010 and additionally supported by EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680. Additionally, this work has been carried out within the PhysioIllustration research project #218023, which is funded by the Norwegian Research Council.

References

[AB04] ANDREWS S. S., BRAY D.: Stochastic simulation of chemical reactions with spatial resolution and single molecule detail. *Physical biology 1*, 3 (2004), 137. 2

[Bar01] BARAFF D.: Physically based modeling: Rigid body simulation. *ACM SIGGRAPH Course Notes 2*, 1 (2001). 6

[BH11] BELL N., HOBEROCK J.: Thrust: A 2 6. *GPU Computing Gems Jade Edition* (2011), 359. 5

[Bli82] BLINN J.: A generalization of algebraic surface drawing. *ACM Transactions on Graphics 1* (1982), 235–256. 3

[Dem10] DEMATTÉ L.: Parallel particle-based reaction diffusion: a gpu implementation. In *Parallel and Distributed Methods in Verification, 2010 Ninth International Workshop on, and High Performance Computational Systems Biology, Second International Workshop on* (2010), IEEE, pp. 67–77. 5, 6

[dHCKMK13] DE HERAS CIECHOMSKI P., KLANN M., MANGE R., KOEPPL H.: From biochemical reaction networks to 3d dynamics in the cell: The ZigCell3D modeling, simulation and visualisation framework. In *Proceedings of IEEE BioVis* (2013), pp. 41–48. 2

[DVRH07] Daae Lampe O., Viola I., Reuter N., Hauser H.: Two-level approach to efficient visualization of protein dynamics. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1616–1623. 3, 6

[Ede99] Edelsbrunner H.: Deformable smooth surface design. *Discrete & Computational Geometry 21*, 1 (1999), 87–115. 3

[FKE13] Falk M., Krone M., Ertl T.: Atomistic visualization of mesoscopic whole-cell simulations using ray-casted instancing. *Computer Graphics Forum 32*, 8 (2013), 195–206. 3, 8

[FKRE09] Falk M., Klann M., Reuss M., Ertl T.: Visualization of signal transduction processes in the crowded environment of the cell. In *Proceedings of IEEE PacificVis 2009* (2009), pp. 169–176. 2

[FKRE10] Falk M., Klann M., Reuss M., Ertl T.: 3D visualization of concentrations from stochastic agent-based signal transduction simulations. In *Proceedings of IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI '10)* (2010), pp. 1301–1304. 2

[GB78] Greer J., Bush B. L.: Macromolecular shape and surface maps by solvent exclusion. *Proceedings of the National Academy of Sciences of the United States of America 75*, 1 (1978), 303–307. 3

[Goo03] Goodsell D.: *Illustrating Molecules*. 2003, ch. 15, pp. 267–270. 3

[Goo09] Goodsell D.: *The Machinery of Life*. Springer, 2009. 7

[HFS*03] Hucka M., Finney A., Sauro H. M., Bolouri H., Doyle J. C., Kitano H., Arkin A. P., Bornstein B. J., Bray D., Cornish-Bowden A., et al.: The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics 19*, 4 (2003), 524–531. 5

[HSG*06] Hoops S., Sahle S., Gauges R., Lee C., Pahle J., Simus N., Singhal M., Xu L., Mendes P., Kummer U.: Copasi - a complex pathway simulator. *Bioinformatics 22*, 24 (2006), 3067–3074. 5

[JAG*11] Johnson G. T., Autin L., Goodsell D. S., Sanner M. F., Olson A. J.: ePMV embeds molecular modeling into professional animation software environments. *Structure 19*, 3 (2011), 293–303. 1

[KG00] Kanehisa M., Goto S.: Kegg: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res 28*, 1 (2000), 27–30. 2

[KMP10] Kubera Y., Mathieu P., Picault S.: Everything can be agent! In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems* (2010), pp. 1547–1548. 3

[KWN*13] Knoll A., Wald I., Navrátil P. A., Papka M. E., Gaither K. P.: Ray tracing and volume rendering large molecular data on multi-core and many-core architectures. In *Proceedings of the 8th International Workshop on Ultrascale Visualization* (New York, NY, USA, 2013), UltraVis '13, ACM, pp. 5:1–5:8. 3

[LBH12] Lindow N., Baum D., Hege H.-C.: Interactive rendering of materials and biological structures on atomic and nanoscopic scale. *Computer Graphics Forum 31*, 3 (2012). 3

[LG07] Le Grand S.: Broad-phase collision detection with cuda. *GPU Gems 3* (2007), 697–721. 6

[LPK*13] Lex A., Partl C., Kalkofen D., Streit M., Gratzl S., Wassermann A. M., Schmalstieg D., Pfister H.: Entourage: Visualizing relationships between biological pathways using contextual subsets. *IEEE Transactions on Visualization and Computer Graphics*, 12 (2013), 2536–2545. 2

[mol13] Molecular Maya 1.3 website: www.molecularmovies.com/toolkit, 2013. 1

[PRV13] Parulek J., Ropinski T., Viola I.: Seamless abstraction of molecular surfaces. In *Proceedings of SCCG* (2013), pp. 120–127. 6

[PS03] Plimpton S. J., Slepoy A.: *ChemCell: a particle-based model of protein chemistry and diffusion in microbial cells*. United States. Department of Energy, 2003. 2

[RFK*13] Reda K., Febretti A., Knoll A., Aurisano J., Leigh J., Johnson A., Papka M., Hereld M.: Visualizing large, heterogeneous data in hybrid-reality environments. *Computer Graphics and Applications, IEEE 33*, 4 (July 2013), 38–48. 3

[SND05] Saraiya P., North C., Duca K.: Visualizing biological pathways: requirements analysis, systems evaluation and research agenda. *Information Visualization 4*, 3 (2005), 191–205. 2

[SVS13] Stone J. E., Vandivort K. L., Schulten K.: Gpu-accelerated molecular visualization on petascale supercomputing platforms. In *Proceedings of the 8th International Workshop on Ultrascale Visualization* (New York, NY, USA, 2013), UltraVis '13, ACM, pp. 6:1–6:8. 3

[TCM06] Tarini M., Cignoni P., Montani C.: Ambient occlusion and edge cueing to enhance real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1237–1244. 3, 6

PAPER B

Illustrative Timelapse: A Technique for Illustrative Visualization of Particle-Based Simulations

Published in [LMWPV15] by IEEE.

Illustrative Timelapse: A Technique for Illustrative Visualization of Particle-Based Simulations

Mathieu Le Muzic[1] Manuela Waldner[1] Julius Parulek[2] Ivan Viola[1]

[1]Vienna University of Technology, Austria
[2]University of Bergen, Norway

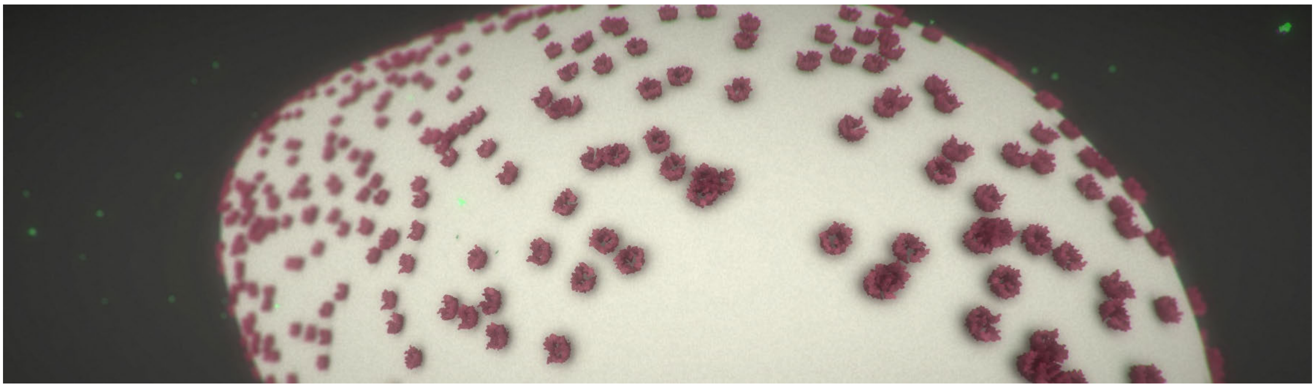


Figure 1: Screen capture of an outer mitochondrial membrane featuring adenosine triphosphate (ATP) molecules (in green) crossing larger voltage-dependent anion channel (VDAC) proteins (in dark red). Our real-time technique lets us observe results of mesoscale particle simulation in fast-forward while showcasing perceivable moving molecules and a scene full of stochastic motion and interactions

ABSTRACT

Animated movies are a popular way to communicate complex phenomena in cell biology to the broad audience. Animation artists apply sophisticated illustration techniques to communicate a story, while trying to maintain a realistic representation of a complex dynamic environment. Since such hand-crafted animations are time-consuming and cost-intensive to create, our goal is to formalize illustration techniques used by artists to facilitate the automatic creation of visualizations generated from mesoscale particle-based molecular simulations. Our technique *Illustrative Timelapse* supports visual exploration of complex biochemical processes in dynamic environments by (1) seamless temporal zooming to observe phenomena in different temporal resolutions, (2) visual abstraction of molecular trajectories to ensure that observers are able to visually follow the main actors, (3) increased visual focus on events of interest, and (4) lens effects to preserve a realistic representation of the environment in the context. Results from a first user study indicate that visual abstraction of trajectories improves the ability to follow a story and is also appreciated by users. Lens effects increased the perceived amount of molecular motion in the environment while trading off traceability of individual molecules.

Index Terms: I.3.7 [COMPUTER GRAPHICS]: Three-Dimensional Graphics and Realism—Animation; I.6.3 [SIMULATION AND MODELING]: Applications—

1 INTRODUCTION

In molecular biology, visual explanations such as still images and animated movies are often employed to describe how things work to a lay audience. Empirical studies have also shown that animations are helpful for a better understanding of molecular biology on the academic level [14]. Over the last years, animated movies have become increasingly popular thanks to tools such as *ePMV* [15] or *Molecular Maya* [2], easing the import of molecular structures into 3D animation packages. But despite these special-purpose modeling tools, scientific movies are still created via hand-crafted key-frame animation, which is tedious, time consuming and can be very expensive, too. The lack of automated techniques clearly hampers the communication of the outcome from biological sciences to the general audience.

In order to improve the way how visual communication is traditionally being made, visualization scientists have experimented with the use of data from computational biology to reproduce dynamic and 3D mesoscale environments. A commonly employed technique is mesoscale particle-based simulation, since it features information about the type and the location of each single molecule over a time-course. This technique was initially developed for systems biologists to study spatial and quantitative properties of biological pathway models.

Direct visualization of the simulation results consist of displaying particle positions on screen and consecutively for each simulated step. Because particle simulations operate at very small time steps, typically of the order of nanoseconds, the resulting number of steps can often be very large. Observing an entire biochemical process can therefore lead to overly long visualization sequences. To shorten the duration of the visualization, it is common to only display simulation frames spaced at a constant given interval. This is usually referred to as *fast-forward* or *timelapse*. However, when using large lapses, the scene may exhibit visual clutter due to the artifacts of the frame-dropping and the very fast motion of the particles. As a result, one can hardly keep track of individual elements throughout consecutive frames, which may be an unpleasant experience and also cause a misinterpretation of the scene.

We present *Illustrative Timelapse*, a novel technique to reduce visual clutter caused by dynamic fast-forward visualization of molecular simulation data on the mesoscale level. Illustrative Timelapse supports seamless temporal zooming and ensures that

---

on any zoom level the motion trajectory of scene elements can be followed by the gaze of human observers. Inspired by hand-crafted animated illustration techniques, Illustrative Timelapse shortens the trajectory of particles so that each element is below the maximum velocity that can be followed and tracked by smooth pursuit eye movements. As a result, we obtain a slower and easily perceivable motion of individual molecules, while remaining in the vicinity of their original motion in accordance to underlying scientific data (Figure 1). We also observed that some artists made the choice of blending abstracted views together with realistic views in order to enhance the viewer's comprehension (see, for instance, work by Drew Berry [5]). Illustrative Timelapse therefore uses a lens system that seamlessly merges the abstracted dynamic visualization with the raw data to lower the risk of misinterpretation in diffusion speed and we assess its effectiveness through a user study.

We envision scientific utility of our newly developed technique in two scenarios, where both scenarios include dynamics spanning over multiple temporal scales. Being able to simultaneously perceive events whose timespan is several orders of magnitude apart from each other, has been considered as a very useful technology by our cooperating scientific partners from the field of biology. The first scenario would be the authoring phase when a stochastic model is being developed. Here our Illustrative Timelapse technique can be part of a *visual debugger* framework that enables the author to see whether the system behaves as intended. The second scenario would be the explanatory exploration of the simulation content, where the intended audience would be interactively exploring dynamics in various parts of the 3D scene. By applying Illustrative Timelapse to a particular part in the scene, the viewer will be able to perceive the physiologically relevant content, while the rest of the scene would not be distorted and will convey in which dynamics the physiological process is embedded in.

2 RELATED WORK

We structure the prior work review in two parts. In the first part we relate with research about biology and techniques that employ computational biology to generate illustrative visualizations. In the second part we provide a broader overview of techniques used in different time-dependent visualization and video-based graphics research areas.

2.1 Mesoscale Cellular Visualization

There are several tools described in the literature which aim at modeling signaling pathways on the mesoscale level via particle-based simulations. The most popular ones are *MCell* [23], *Smoldyn* [4] or more recently the *Cellular Dynamic Simulator* [6]. They all feature graphics modules to showcase the results of the simulation. Among those tools, MCell has one of the most advanced visualization modules, *CellBlender* [1], which is a plug-in for *Blender*, a 3D animation package commonly used by 3D artists. The module was introduced to ease the creation of models for MCell by bridging 3D modeling and biological modeling in one single powerful tool. It allows for direct visualization of the particles embedded in their cellular structures represented as 3D meshes. However, direct visualization of particle-based simulation is mostly intended as a visual support for expert users in their modeling task. The outcome exhibits high visual complexity due to the large number of visualized particles and their diffusion driven chaotic organization, which is impractical for dissemination of biological sciences to a broader audience. Moreover, in case of large temporal simulations it is quite cumbersome to loop over all the frames to locate and understand the events of interest.

Falk et al. [8] presented a tool which reads the results of a particle-based simulation and enables interactive visual exploration of the results. The aim of their visualization is to describe the process of signal transduction on both mesoscopic and molecular level. The individual molecules are represented in 3D space as spherical glyphs and their positions are updated over time according to the simulation values. The tool also allows the user to track specific particles inside a cell. The trajectory is represented as a trail in 3D space providing information about directions and reactions. In addition, they employ depth-of-field and depth cues, encoded as a color gradient, to emphasize molecules and their trajectories. This represents a useful focus+context visualization technique, which helps to guide attention in crowded cellular environments. In our work, we employ focus+context techniques not only to filter the visual appearance, but also the level of detail of dynamic properties. In a follow-up work, Falk et al. [9] developed a new technique to improve visualization of large mesoscopic cellular models to observe particle simulations together with real molecular structure data. The entire scene is processed by means of ray casting of spheres, which is efficiently performed on the underlying grid structures holding the molecular positions. Additionally, each molecule has its own supporting grid, which is then traversed based on the given level of detail manner. The authors achieved 3.6 frames per second for scenes containing 25×10^9 atoms.

More recently, Le Muzic et al. [20] proposed a new type of particle systems designed for illustrative visualization of molecular reactions. The main idea behind this work was to show sequences of molecular reactions describing a molecular pathway, and embedded in their cellular environment. They claim this task to be to cumbersome using a particle-based approach due to the stochastic behavior of the simulation. Instead, they propose to use quantitative simulations, to control the behavior of molecules and force them to react directly in the viewport. They also distort the real diffusion values in order to perceive moving elements more easily. While the location of elements and reaction is chosen based on illustrative considerations, the quantities and reactions rates remain in accordance with real scientific data. Moreover due to lightweight computation of quantitative models, the simulation is able to run simultaneously with the visualization, whereas traditional particle-based approaches have to be precomputed prior the visualization.

The later work adopted a new approach, more oriented towards storytelling rather than strict scientific exploration. However the fact that diffusion rates and thus spatial location of particles are not depicted realistically may lead to misconceptions in audience non familiar with the basic underlying phenomena. Only particle-based simulation tools are able to provide accurate spatial information and yet there is still no elegant solution available that could automatically generate explanatory visualizations out of this type of data. This work is an attempt to fill the gap left in this domain; with Illustrative Timelapse we aim at providing an interactive and guiding tool for spatio-temporal exploration of mesoscale particle-based simulations. By adopting illustration techniques from animations and videos, we provide a new way to present complex molecular scenes from particle-based simulations, to achieve a maximum degree of understanding for a broad audience.

2.2 Time-dependent Visualization

In general, time-dependent data can be visualized using either a static or a dynamic representation [3]. While in the information visualization domain static representations are far more common, we explore challenges arising when using dynamic representations as dynamics is a fundamental property of molecular interactions. Wolter et al. [26] presented a model for time-varying visualizations utilizing dynamic representations. In their model, they distinguish between *user time* (the real time we perceive), *simulation time* (time changes in the simulated process) and *visualization time* (a normalized time frame for showing the complete simulation). They demonstrate their model on scientific simulation data that can be visualized with interactively changeable temporal resolutions. However, they do not abstract the resulting dynamic visualization in case the processes cannot be followed by human observers any more, due to large velocities in low temporal resolutions.

While there is little related work on visual abstraction techniques for dynamic visualizations, video-based graphics research addresses a conceptually similar challenge to create short video synopses out of long input videos. *Adaptive fast-forward*, for instance, chooses an optimal playback speed based on the estimated information density in the input video to create compact surveil-

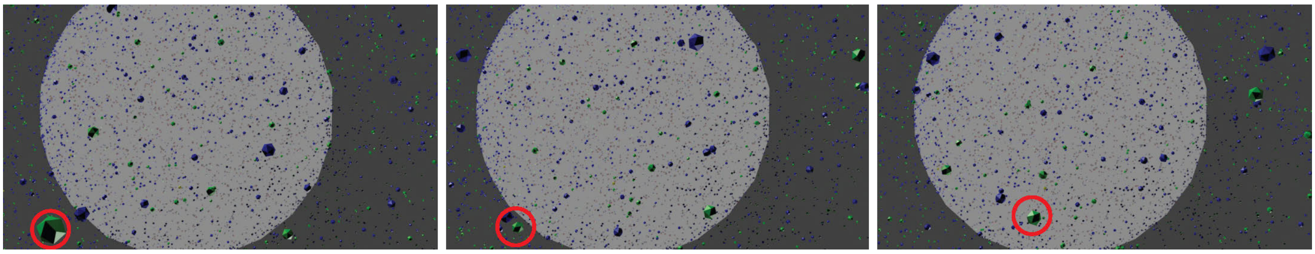


Figure 2: Three consecutive frames of a MCell based simulation visualized using CellBlender. When directly visualizing the molecular positions obtained from the simulation data, it is very challenging to track the molecules in consecutive time steps (cf., red circle).

lance camera streams [11]. Instead of dropping uninformative frames to create a video synopsis, *Space-Time Video Montage* [17] and *Dynamic Video Synopsis* [22] extract and reassemble informative space-time portions into a smaller video space-time volume. These techniques work well if the input video contains longer temporal sequences or spatial regions of low information density. However, this is not the case in mesoscale molecular simulations, where the entire environment is filled with constantly moving elements.

An extreme case of video synopsis is the compression of a short input video into a single still image. This can be achieved by merging multiple frames with salient content into a static image [24], by visualizing clustered movement trajectories of tracked elments in surveillance streams [12], or a combination of these two approaches [19]. Nevertheless, our goal is to preserve the dynamic representation so users can obtain a deeper understanding of the temporal properties of biochemical processes, as opposed to static representations commonly found in biology text books.

To reduce visual clutter in fast-forward videos, Höferlin et al. [13] investigated different methods to represent elements with high velocities. According to their empirical findings, simple dropping of frames without any visual abstraction led to the best performance when users were asked to detect a specific object in the surveillance video as well as subjectively rate the motion perception of objects in the video. The videos used in their study were sped-up by a factor of 10 and 20, respectively and scene elements like pedestrians or cars could still be easily traced without any visual abstraction. In contrast, in mesoscale molecular simulations, movement speed of individual elements and the duration of a sequence of reactions involved in biochemical processes often differ in a factor of 100 and more. With these speed differences, simple frame dropping as well as conventional visual abstraction methods like motion blur, fail to preserve traceable object trajectories.

First-person hyper-lapse videos create stabilized fast-forward videos of helmet camera recordings by reconstructing and smoothing the 3D camera trajectory [18]. In our work, we adopt the concept of spatio-temporal trajectory smoothing to create smooth fast-forward visualizations of long input data. However, we smooth the trajectories of the scene elements instead of the camera path. In addition, we ensure that the user is provided with a visual reference to the original speed to prevent extensive misinterpretations of the movement properties of the involved actors.

3 REQUIREMENT ANALYSIS

Visualization of particle-based molecular simulations on the mesoscale level aims at providing a better understanding of cell biology by means of realistic representations [8]. Despite the overall chaos of the output, it reveals important features, such as quantities, spatial distribution, scales, and motion. Most complex biochemical processes also comprise at least two temporal scales: local (e.g., the behavior of a single molecule at a certain time) and global (the entire process that is simulated). While molecules are constantly moving by a constant displacement in every simulation frame, on the other hand, physiologically meaningful interactions may only be happening every 100th step, for instance. So in the case of large simulations ($> 10^5$ frames), it would be quite inefficient and cumbersome to observe the entire sequences of frames. However, speeding up the animation so that more interesting events are happening within a reasonable time-frame would result in a chaotic and unwatchable animation. Indeed, individual molecules are moving so quickly between two consecutive frames, that it becomes impossible for a human observer to follow them, as indicated in Figure 2.

Instead of employing realistic simulations, scientific illustrators hand-craft animated visualizations to ensure that the essential information is intuitively presented to the viewer. For instance, in the movie *Apoptosis* made by Drew Berry [5], which is depicted in Figure 3, exemplary reactions in the foreground are slowed down and emphasized, while elements in the background are moving with very high velocity. When watching the video one gets the impression of a very fast overall particle motion, but in reality this impression is a carefully crafted illusion. Assuming the constancy among the shown molecular species, the viewer adopts an impression that the foreground elements are behaving in the same way as the same species in the background and perceives these as moving similarly fast.

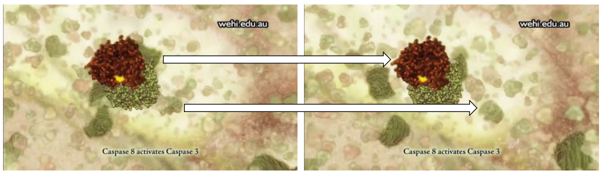


Figure 3: Screenshot of frame m and $m+3$ of a molecular animation [5]. Mind how the main reaction is forced to remain fairly stationary (upper arrow), while molecules in the background move very quickly (lower arrow).

Inspired by such hand-crafted animations, we formulated four requirements for the automatic creation of illustrative visualizations from particle-based molecular simulations:

R1: The visualization should show the simulated process in a reasonable amount of time, irrespective of the number of simulation frames. The duration of the visualization can either be predefined (for instance, to be part of a scientific movie with predefined length) or interactively chosen by the user.

R2: Secondly, it has to be possible to keep track of individual elements in the visualization—especially those with a high degree of interest at a certain time. This means that there should be a maximum velocity of elements in the resulting visualization, so that human observers can visually trace them.

R3: Thirdly, visibility of events with high degree of interest (e.g., molecular reactions or channel protein transfers) should be guaranteed. Reactions are usually represented as punctual events in particle simulations and therefore might be difficult to perceive because they do not last longer than a simulation step or might simply be skipped due to frame dropping.

R4: Fourthly, the visualization should be as realistic and detailed as possible. This requirement is in accordance with the results of a study by Jenkinson et al. [14], which demonstrated that a more realistic depiction of a process can enhance the viewer's understanding compared to a highly abstracted one.

Clearly, these requirements partially contradict each other: A realistic representation of a visualization with a low temporal resolution will lead to large displacements—and as a consequence, velocities—of individual elements, violating R2. Hand-crafted animations, like the one shown in Figure 3, have demonstrated that there are ways to visually trade fidelity against an appealing representation to create animations that are valuable for a general audience. What is missing is a formalism to apply these artistic principles to visualizations that are automatically generated from scientific simulation data. The goal of Illustrative Timelapse is therefore to find a trade-off to at least partially satisfy all four requirements for the automatic production of illustrative visualizations from particle-based molecular simulations.

4 ILLUSTRATIVE TIMELAPSE

Illustrative Timelapse is an illustrative visualization technique for semantic zooming [21] in the temporal domain. Its concept is based on the deformation of time exemplified, for instance, in the animation of Drew Berry [5] shown in Figure 3. The motion pattern is selectively simplified in order to convey a particle's trajectory in a clearly understandable motion, rather than representing the entire particle path, which could not be traced by human observers anyway. This selective abstraction can be applied globally to the entire scene, within a dedicated region in the scene, or as a function of eye-space depth. Illustrative Timelapse performs four operations to fulfill the requirements formulated above:

1. **Temporal zooming** allows users to interactively change the temporal resolution of the visualization (Section 4.1).
2. **Visual abstraction** reduces the speed of particles according to the temporal resolution (Section 4.2).
3. **Emphasizing** ensures that important reactions are visualized for a minimum duration and visually stand out (Section 4.3).
4. **Lens effects** preserve non-abstracted movements in the context, so that a more realistic impression of molecular motion is depicted (Section 4.4).

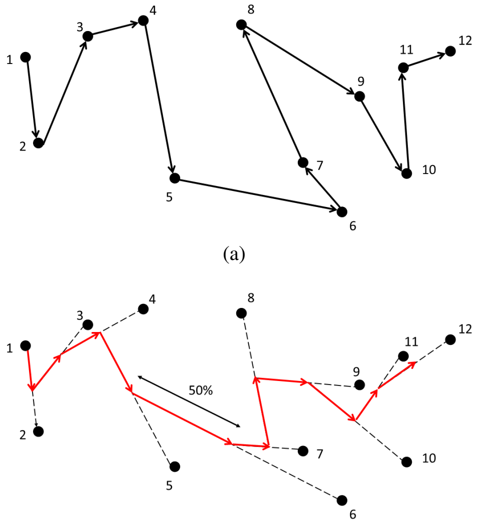
With these operations, Illustrative Timelapse supports visual exploration of complex phenomena in a user-defined time frame (→ R1), trading visualization realism (→ R4) against good visibility of important events (→ R3) and easily perceivable velocities (→ R2).

4.1 Temporal Zooming

As input, our technique uses particle-based simulation data obtained from computational biology. Let us assume that the input particle simulation contains N frames, where each frame represents a discrete step in the simulation of Δt. Each frame contains a set of molecules (particles) given by their type and position. In addition, information about the reactions (type, time, location, and participants) is provided by the simulation tool, which defines the changes in molecular states between frames.

To visualize such a simulation, we map these N simulation frames into a new set of $M \leq N$ visualization frames. The number of visualization frames can be chosen so that $M = d \cdot f$, where d is the desired duration of the output visualization in seconds and f is the number of visualized frames per second. The smaller the desired duration d, the lower the temporal resolution of the visualization and the larger the displacement of elements between consecutive visualization frames. Alternatively, users can directly change the temporal resolution during run-time—for instance when first watching the simulation in a low temporal resolution to get an overview and subsequently zooming in to observe parts of the simulation in more detail. The number of visualization frames would then be $M = \frac{N}{z}$, where $z \geq 1$ is the temporal zoom factor.

For each visualization frame $m \in \{1, \ldots, M\}$, the corresponding simulation frame $n = m\frac{N}{M}$ is obtained from the simulation data. When directly visualizing all the simulation steps so that $M = N$, each visualization element's position $p[m]$ directly corresponds to the position $b[n]$ from the simulation data. Otherwise, we use nearest-neighbor interpolation to determine $p[m]$ from the simulation data, which is equivalent to simple dropping of simulation frames.



(a)

(b)

Figure 4: A sample movement trajectory over 12 frames: (a) the original trajectory ($\rho = 1$) and (b) a reduced version visualized in red ($\rho = 0.5$) which is equivalent to a displacement reduction of 50% of the distance between the current position and the next sampled point on the trajectory.

4.2 Visual Abstraction

From psychology research, it is well known that there is a lower and an upper velocity limit a human observer is able to follow with an uninterrupted, smooth eye movement (*smooth pursuit*). For the upper limit, this threshold lies at 50° to 70° of visual angle per second for a single target with predictable trajectory [7]. Assuming a visualization frame rate of 60 Hz, a monitor resolution of 110 ppi and an optimal distance between user and monitor of ~ 60 cm, elements should be displaced less than ~ 40 pixels per frame to stay below this maximum velocity threshold. However, this threshold also varies between individuals [7] and elements with unpredictable trajectories or surrounded by distractor elements generally need to move slower to be trackable [27]. Experimental findings also suggest that smaller targets are harder to track than larger ones [16]. Since our visualization consists of a large number of molecules subject to unpredictable Brownian motion, the amount of per-molecule displacement per frame should therefore be significantly below this 40-pixel threshold if we want to ensure that viewers are able to follow individual elements with smooth pursuit eye movements. We reduce the speed of the particles, using a straightforward temporal filtering technique. This technique allows to shorten the motion path of the particles while preserving them in the vicinity of their original trajectories. We opted for this solution because it is light to implement and to compute and offers a lot of control over the desired speed of particles. Such method also is commonly used in eye tracking research to improve the stability of gaze-controlled cursors [28].

We formulate this as an Infinite Impulse Response (IIR) low-pass filter in the following form:

$$p[m] = (1-\rho)p[m-1] + \rho b[n], \qquad (1)$$

where m refers to the current visualization frame, n to the corresponding simulation frame, $p[m-1]$ is the filtered 3D position of the particle in the previous visualization frame and $b[n]$ is the discrete 3D trajectory position obtained from the simulation. The displacement magnitude $\rho \in [\rho_{min}, 1]$ is chosen according to the selected temporal resolution, where ρ_{min} corresponds to an arbitrarily defined minimum value of ρ. The smooth signal thus is obtained by blending the result of the previous output of the filter (i.e., $p[m-1]$) with the Brownian motion specified by the simulation (i.e., $b[n]$), as illustrated in Figure 4(b).

4.3 Reactions Emphasis

In particle-based simulations, changes in molecular states, e.g., creation or consumption, are the results of molecular reactions. As a consequence, such events can greatly enhance the viewer's comprehension of the underlying process. This is why they are often depicted very explicitly in illustrated movies (see, for instance, the connected brown and green macromolecules in Figure 3).

However, in the simulation, reactions are represented as punctual events that only take place in a single simulation step, which is very short and can therefore be hard to perceive. Additionally, when creating a visualization in low temporal resolution (fast-forward), reactions will likely be filtered out because they often take place between two visualized lapses. As a result in the visualization, molecules will suddenly change state, without providing any hint about how it occurred or which molecules it reacted with.

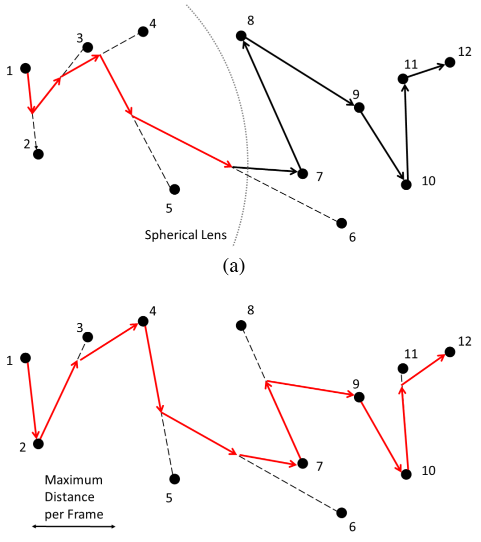
We therefore prolong the duration of the reactions to make them stand out and to inform the viewer about the nature these events. Information about reaction locations, times and participants is first read out from the simulation data in order to anticipate reaction events during the visualization. We use this information to associate new or missing particles between consecutive frames to their corresponding reactions, thus the birth or death of particles will only occur as a result of a reaction event. Then, a few seconds before each individual reaction and until the end of the reaction, participating reactants are attracted toward the reaction location. It is worth mentioning that during this operation particles are not longer subject to their original trajectories. In order to ensure all reactants to meet each other at the right time, we apply a per-frame displacement to the particles in direction of the reaction location. We also adapt the magnitude of these displacements according to the distance between particles and reaction sites in order to obtain uniform reaction durations. To emphasize reaction events, the color of reacting elements is highlighted, thus contrasting with the rest of the scene. Once a reaction is accomplished, the elements are subject to their respective original trajectories again.

4.4 Lens Effects

To preserve a realistic impression of molecular motion despite trajectory reducing, we introduce a temporal focus+context technique that applies visual abstraction of molecular trajectories solely in the foreground (focus), while showing more realistic—yet often untraceable—motion in the background (context). With a lens effect, parameter ρ (Eq. 1) is dynamically adjusted for each element according to some lens-specific rules. We introduce two lens effects: the **world lens** (Section 4.4.1) adjusts ρ based on the world-space distance to a movable spherical lens, while the **maximum velocity lens** (Section 4.4.2) reduces ρ dynamically, based on an element's displacement in screen space between two consecutive frames.

4.4.1 World Lens

We represent the world lens as a two-layered sphere with an inner radius surrounding the focus region and an outer radius beyond which we define the context region. The area between the inner and outer radius represents a smooth transition between both regions. We fixed the center of the sphere to the camera position, so that objects with increasing distance from the camera and screen center, respectively, are less abstracted. The world lens concept can be compared with well-known focus+context techniques from the spatial domain, like *fisheye views* [10], which increase the spatial zoom level locally, while preserving an overview in the periphery. Note that the world lens does not affect the temporal zoom level itself, but rather acts as a spatial filter on the amount of visual abstraction applied on the movement trajectories. Figure 5(a) illustrates the effect of the world lens on the visualized particle trajectories.



(a)

(b)

Figure 5: Illustration of two lens effects. a) Visual abstraction (red trails) within the world lens (illustrated with inner radius only, without transition area). b) Dynamic displacement reduction based on screen-space distance.

4.4.2 Maximum Velocity Lens

We define the second lens effect by limiting the displacement of molecules between consecutive frames. The user defines a maximum displacement threshold in screen space (d_{max}), which should typically lie clearly below the maximum smooth pursuit velocity threshold (cf., Section 4.2). The displacement magnitude ρ is then defined as follows:

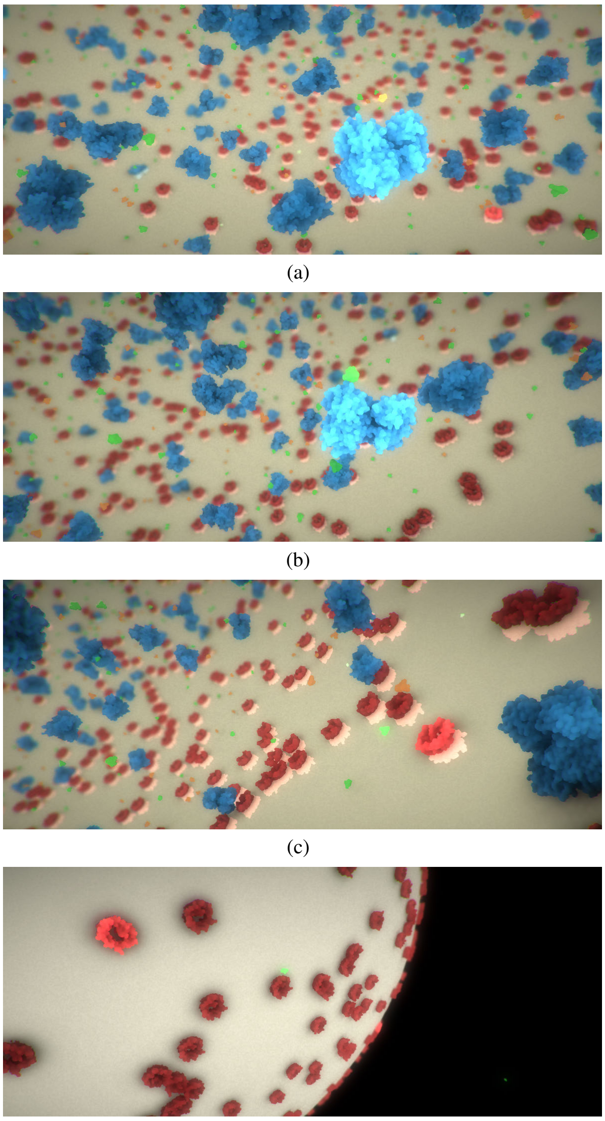
$$\rho = min\{\frac{d_{max}}{||b^*[n] - p^*[m-1]||}, 1\}, \qquad (2)$$

where $p^*[m-1]$ and $b^*[n]$ are the particle's previous position in the visualization and current position in the simulation, respectively, mapped to 2D screen space. In contrast to the world lens, the amount of visual abstraction is hereby defined solely by the element's trajectory projection onto the screen, not its position in the world space (see Figure 5(b)).

When using a perspective projection in the visualization, this maximum velocity lens automatically leads to the effect that elements close to the camera appear to move much slower than those further away in z direction. In fact, their screen space velocity is approximately the same. This illusion is due to the differences in size and displacement of particles that are located in the background. Since the particles move at the same screen velocity, we can achieve a more harmonious effect than with the world lens.

While the world lens requires educated guesses for the maximum amount of visual abstraction as well as the radii of the spherical lens for different temporal resolutions, the maximum velocity lens is defined by the single value of d_{max}, which could be derived empirically from psychophysics experiments. With this approach, it is possible to interactively change the temporal resolution, while ρ is automatically adjusted by the maximum velocity lens. In the simple definition of Eq. 2, visual abstraction is not affected by the element's distance to the screen center on the x-y-plane, as for the world lens. To achieve such an effect, d_{max} can be dynamically increased with increasing distance between $p^*[m-1]$ and the screen center.

It is worth mentioning that the maximum velocity lens may induce an accumulating delay between the current position of particles and their original trajectories, especially with very linear motion. However, in the real world due to Brownian motion, molecules do not exhibit linear trajectories, they rather move in every directions and spread relatively slowly, even in fast forward. Therefore, such motion characteristics guarantees that there will not be much significant delay when applying this approach to molecular simulations.


(a)

(b)

(c)

(d)

Figure 6: Screen capture of our Illustrated Timelapse system which features 25000 particles reacting and diffusing smoothly at more than 50 frames per seconds. (a) Succinate thiokinase (blue) reacting with ADP (orange) inside mitochondrion (b) ATP (green) is produced from Succinate thiokinase (c) ATP about to exit the organelle via VDAC (red) (d) ATP outside mitochondrion.

5 IMPLEMENTATION

In our work, we employ MCell –previously introduced in Section 2– for the modeling and simulation of biochemical processes and we use Unity3D as visualization framework. Computation with MCell can be quite slow depending on the complexity of the model and is simply not suitable for real-time simulation, therefore we precompute the data prior to the visualization.

It is also worth mentioning that MCell does not take collisions between molecular entities into account during the simulation and this is usually the case in particle-based mesoscale simulation. As the result, when visualizing a dense simulation of large molecules they are very likely to intersect, whether the particle trajectories are abstracted or not. To solve this issue one could simply perform collision detection between particles, via a physics engine for instance, and offset the particle position accordingly in order to avoid overlapping molecules.

To enable real-time rendering of a large number of high quality molecules, we also developed a module based on the technique proposed by Le Muzic et al. [20]. This module allows us to load molecular structures stored in the PDB format and to display their Van der Waals surface in real-time using tessellation shaders.

6 USE CASE

Thanks to our illustrative technique we are able to observe a perceivable fast-forward visualization of the molecular machinery while also showing a scene full of stochastic motion together with molecular reactions. We evaluated our technique via a user study on a very simple model of adenosine triphosphate (ATP) molecules diffusing throughout the mitochondrial outer membrane towards the cytosol. Mitochondria are cellular components present in most cells of living species, which is mainly responsible for the energy management of the cell. Its size usually varies between 0.5 and 10 micrometers. On the outer membrane, it features voltage-dependent anion channels (VDAC) proteins, acting as pores and responsible for the passage of molecules across the membrane. Those are around $20-30$ Å large, which is sufficient to let small molecules, such a ATP, pass through while restraining larger molecules from crossing the membrane. VDAC transport proteins are found in very large quantities across the membrane and are also very densely packed together. As computation time and simulation output data size greatly increase with the number of frames and particles, we reduced the amount of molecules to a total number of 5000 particles in this example since it already featured a fairly large number of frames (10^5). The visualization was running smoothly at more than 60 frames per seconds on a desktop equipped with an Inter Core i7-3930 3.20 GHz CPU processor coupled with an NVidia GTX Titan GPU processor.

We also enriched the previous model in order to demonstrate the scalability of our technique. This second model additionally features ADP and succinate thiokinase molecules. Succinate thiokinase is an enzyme which diffuses freely and reacts inside the mitochondrial matrix and is responsible for the production of ATP molecules from ADP, a small ligand previously produced by the other actors of the Krebs cycle. This example comprised a total number of 25000 particles and was running smoothly at more than 50 fps on the same machine we used in the previous example. The diffusion constants we used were $10^6\ cm^2/s$ for ATP and ADP and $10^7\ cm^2/s$ for VDAC and succinate thiokinase, those values were identical in the previous example. We hereby presented a more complex depiction of the mitochondrial machinery by showing molecules and reaction taking place inside the mitochondrial matrix. It is worth mentioning that although enriched, this model still contains only a minimal fraction of processes in the respective organelle. We were also able to observe a much more densely packed scene as previously and we started to foresee occlusion problems that might occur in such conditions. A series of screen captures which depict the sequence of events we describe in this model is shown in Figure 6.

7 USER STUDY

We performed a preliminary user study to investigate the effect of an increasing amount of visual abstraction as well as the lens effect, on the ability to follow individual events, perceived movement speed of the molecules (i.e., the velocity of Brownian motion) and the subjective visual appeal of the resulting visualization. For this purpose, users had to subjectively rate these three aspects after watching videos of the scene described in Section 6. Since our research questions did not involve any interactive exploration of the scene and the desired measurements were solely subjective, the study was performed online.

7.1 Setup

26 users finished the study (aged 24 to 58, four females). Except for two users, all were related to computer sciences and had no extensive knowledge in biology. Two users reported red-green weakness, but produced consistent results and where therefore included in the analysis. We employed a 3×2 within-subjects design with the following factors:

- the amount of **visual abstraction** on three levels: *none* (no trajectory reduction, but simple fast-forwarding instead), *minimal* (slight trajectory smoothing so individual elements could be partially tracked with a considerable amount of effort) and *smooth* (strong trajectory smoothing so individual elements could be tracked effortlessly) and
- whether or not the visual abstraction was shown only within a spatially limited **lens**.

Since the lens has no impact when using no visual abstraction, this results in five different visualization conditions users had to watch and rate: no abstraction, minimal abstraction with and without lens as well as smooth visualization with and without lens. The ρ values for minimal and smooth visual abstraction were obtained in an informal pilot experiment where we showed the results of a simulation sped up a 100 times and three users were subjectively selecting a value within a range of $\rho = 0$ to $\rho = 1$, so it was easily possible to track an individual element ($\rho_s = 0.05$) or possible only with considerable effort ($\rho_m = 0.1$). For the world lens effect, we had an inner radius of $r_1 = 10$ and an outer radius of and $r_2 = 20$ in world units ($1 \times 10^{-8} nm$), while the mitochondrion shown in the video had a maximum length of $100 \times 10^{-8} nm$.

Since absolute judgment of visual appeal, traceability, and speed is difficult, we used comparative questionnaires to obtain subjective ratings: We showed all pair-wise combinations of the five videos in a side-by-side view, resulting in ten comparisons with two videos each. It is worth mentioning that video where not actually shown side-by-side but where superposed instead, because of the large frame width we obtained at 720p resolution. The videos were presented in embedded *YouTube* players with 720p resolution and users were free to watch the videos as many times as they wished—synchronously or one after the other. For each comparison, users were asked to select one of the two videos (or “no difference”) for the three questions listed in Table 1. In addition, users could leave a comment at the end of the study. The order in which the video pairs were presented was randomized.

Q1	Which video was more visually pleasing to watch?
Q2	In which video was it easier to follow molecules diffusing through the membrane?
Q3	In which video did the molecules move faster?

Table 1: The three questions for each pair-wise comparison.

We expected to observe that visual abstraction would lead to improved traceability, but also reduce the perceived molecular speed dramatically. With the world lens, we expected that users would report that elements move faster compared to global abstraction, but that the capability to trace individual elements is hardly affected.

7.2 Results

For each vote for a particular video, we assigned the value 1 to the respective condition, while we assigned 0.5 for both conditions if the users selected “no difference”. We used the *Case V Scaling* model by Thurstone [25] to convert all obtained pair-wise comparison values into interval scale. The values were transformed so that the minimum obtained z-value for each question corresponds to zero.

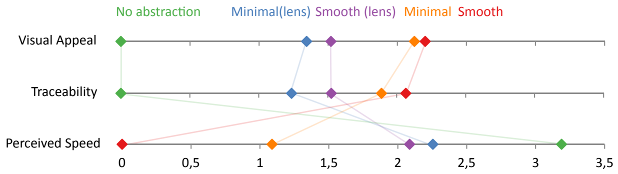


Figure 7: Average z-scores per visualization condition (color-coded) for the three questions listed in Table 1. Higher values are better.

As expected, both global visual abstraction methods (minimal and smooth) led to improved traceability of particles in the visualization (orange and red mark in Figure 7, center), as compared to the non-abstracted version (green mark). Similarly, the perceived speed of particles was much lower in the abstracted visualizations than in the non-abstracted one (see orange, red, and green mark in Figure 7, bottom).

The world lens could not fully preserve the traceability of particles compared to the globally abstracted version (compare blue and purple marks to orange and red marks in Figure 7, center). However, the increased velocity in the context indeed led to a faster perception of overall particle movement (see blue and purple marks in Figure 7, bottom).

In terms of visual appeal ratings, the obtained intervals are very similar to the traceability ratings (compare Figure 7 top and center). In other words, the smoother the overall motion, the more visually appealing the users rated the visualization.

Some users commented that tracing individual elements – especially the small ATP molecules – was generally hard, even in the abstracted versions of the visualization. A few suggested using stronger visual highlight effects to make reactions more evident. Also, one user criticized the fact that reactions were also highlighted in the *“blurred out-of-focus regions”*, where reactions were *“not at all”* visible—presumably especially in the lens conditions, where the blurred regions were also visualized with non-abstracted motion. One user also reported that *“the different movement speeds for particles nearer and further away in some videos feel weird”*, which was probably caused by the world lens effect.

7.3 Discussion

Our preliminary user study results indicate that with increased trajectory smoothing, the ability to trace individual elements is better than if observing a non-abstracted visualization. Unsurprisingly, our study also showed that the perceived motion speed was decreased due to the decreased velocity of the elements. The world lens used in our experiment could alleviate this underestimation effect to some extent, but in turn leads to more difficulties when keeping track of individual elements as well as decreased aesthetics ratings.

Due to these observations, together with selected user statements collected during the study, we iterated the lens design and created the maximum velocity lens, described in Section 4.4.2. In addition, the reaction highlighting should clearly be coupled with the lens effect: Only if elements can be easily traced by the observer’s gaze, a visual highlight should attract the user’s attention to those elements that are about to react. Otherwise, the attention is directed towards contextual scene regions, whose primary purpose is actually to generate a realistic impression of the environment, but not to communicate the story of the visualized process in focus.

Nonetheless, illustrative techniques always compromise realism when providing an explanatory visual description to enhance the viewer's understanding of selected aspects. It is therefore up to the visualization designer to define priorities and choose the level of visual abstraction accordingly—just like illustrations and visual storytelling.

It is worth mentioning that in order to fully evaluate our tool a few points would need to be further evaluated. Firstly, our study did not include the maximum velocity lens, which would be needed to judge its usability and effectiveness more objectively. Secondly, the fact that our technique provides means for interactive exploration also calls for further investigations, requiring users to accomplish a set of tasks instead of simply rating a video. And finally we ought to conduct more empirical research to obtain the best candidates for the parameters ρ and the maximum traceable velocity d_{max} in such chaotic systems.

8 CONCLUSION AND FUTURE WORK

With Illustrative Timelapse we introduced a new technique for interactive and explanatory exploration of mesoscale particle-based simulation on multiple temporal scales. Our system is capable of showing traceable moving molecules, when viewing processes in fast-forward which is important for an artistic depiction of biological sciences. While abstracting original motion from the simulation we also provide a more accurate representation of the molecular diffusion in the context area, similarly as in animated movies. In future work, it might be worthwhile investigating whether our proposed technique would also apply to other types of simulation, such and molecular dynamics for instance.

Although this technique has a great potential for dissemination of biological sciences it also comprises a few limitations: Firstly, the simulation has to be pre-computed prior to the visualization which disallows user interaction with the simulation outcome. This problem can be tackled by executing the mesoscale simulation on accelerated hardware such as supercomputers or graphics processing units and thereby enable interactivity. The second limitation is about potential occlusion problems that can occur with very dense scenes. Indeed, molecules are usually subject to molecular crowding inside their cellular compartments. Therefore, in order to provide a more faithful representation to the viewer we would have to develop new visualization techniques aiming at providing a high degree of understanding despite clutter due to molecular crowding. In addition, we also plan to extend the perceptual study in order to cover the points that were not yet evaluated and which we discussed in Section 7.3. We also wish to compare our the IIR filter with other techniques such as B-splines for instance in order to find out which technique offers the best visual abstraction. Finally we will reinforce cooperation with domain experts, in order to develop more complex models and provide and better interactive depiction of the machinery of life via our technique.

ACKNOWLEDGEMENTS

This project has been funded by the Vienna Science and Technology Fund (WWTF) through project VRG11-010 and additionally supported by EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680. Additionally, this work has been carried out within the PhysioIllustration research project 218023, which is funded by the Norwegian Research Council. We would also like thank the collaborators of this project, Stefan Bruckner, Helwig Hauser, Matthias Ziegler and Matthias Bernhard.

REFERENCES

[1] Cellblender. `http://code.google.com/p/cellblender/`. Accessed: 2014-12-01.

[2] Molecular maya. `http://www.molecularmovies.com/toolkit/`. Accessed: 2014-12-01.

[3] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Springer Science & Business Media, 2011.

[4] S. S. Andrews and D. Bray. Stochastic simulation of chemical reactions with spatial resolution and single molecule detail. *Physical biology*, 2004.

[5] D. Berry. Apoptosis. `http://youtu.be/DR80Huxp4y8?t=1m50s`, 2006. The Walter and Eliza Hall Institute.

[6] M. J. Byrne, M. N. Waxham, and Y. Kubota. Cellular dynamic simulator: an event driven molecular simulation environment for cellular physiology. *Neuroinformatics*, 2010.

[7] J. D. Enderle. *Models of Horizontal Eye Movements: Early models of saccades and smooth pursuit*. Morgan & Claypool Publishers, 2010.

[8] M. Falk, M. Klann, M. Reuss, and T. Ertl. Visualization of signal transduction processes in the crowded environment of the cell. In *Proceedings of IEEE PacificVis 2009*, 2009.

[9] M. Falk, M. Krone, and T. Ertl. Atomistic visualization of mesoscopic whole-cell simulations using ray-casted instancing. *Computer Graphics Forum*, 2013.

[10] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1986.

[11] B. Hoeferlin, M. Hoeferlin, D. Weiskopf, and G. Heidemann. Information-based adaptive fast-forward for visual surveillance. *Multimedia Tools and Applications*, 2011.

[12] M. Hoeferlin, B. Hoeferlin, D. Weiskopf, and G. Heidemann. Interactive schematic summaries for exploration of surveillance video. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, 2011.

[13] M. Hoeferlin, K. Kurzhals, B. Hoferlin, G. Heidemann, and D. Weiskopf. Evaluation of fast-forward video visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2012.

[14] J. Jenkinson and G. McGill. Visualizing protein interactions and dynamics: evolving a visual language for molecular animation. *CBE-Life Sciences Education*, 2012.

[15] G. T. Johnson, L. Autin, D. S. Goodsell, M. F. Sanner, and A. J. Olson. *ePMV* embeds molecular modeling into professional animation software environments. *Structure*, 2011.

[16] C. W. Johnston and F. J. Pirozzolo. *Neuropsychology of Eye Movement*. Psychology Press, 2013.

[17] H.-W. Kang, Y. Matsushita, X. Tang, and X.-Q. Chen. Space-time video montage. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[18] J. Kopf, M. F. Cohen, and R. Szeliski. First-person hyper-lapse videos. *ACM Transaction on Graphics*, 2014.

[19] A. Meghdadi and P. Irani. Interactive exploration of surveillance video through action shot summarization and trajectory visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2013.

[20] M. L. Muzic, J. Parulek, A.-K. Stavrum, and I. Viola. Illustrative visualization of molecular reactions using omniscient intelligence and passive agents. *Computer Graphics Forum*, 2014.

[21] K. Perlin and D. Fox. Pad: An alternative approach to the computer interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, 1993.

[22] A. Rav-Acha, Y. Pritch, and S. Peleg. Making a long video short: Dynamic video synopsis. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[23] J. R. Stiles, T. M. Bartol, et al. Monte carlo methods for simulating realistic synaptic microphysiology using mcell. *Computational neuroscience: realistic modeling for experimentalists*, 2001.

[24] L. Teodosio and W. Bender. Salient stills. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2005.

[25] L. L. Thurstone. *The measurement of values*. Univer. Chicago Press, 1959.

[26] M. Wolter, I. Assenmacher, B. Hentschel, M. Schirski, and T. Kuhlen. A time model for time-varying visualization. *Computer Graphics Forum*, 2009.

[27] J.-J. O. d. Xivry and P. Lefèvre. Saccades and pursuit: two outcomes of a single sensorimotor process. *The Journal of Physiology*, 2007.

[28] X. Zhang, X. Ren, and H. Zha. Improving eye cursor's stability for eye pointing tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008.

PAPER C

cellVIEW: A Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets

Published in [LMAPV15] by the Eurographics Association.

Eurographics Workshop on Visual Computing for Biology and Medicine (2015)
K. Bühler, L. Linsen, and N. W. John (Editors)

cellVIEW: a Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets

Mathieu Le Muzic[1] and Ludovic Autin[2] and Julius Parulek[3] and Ivan Viola[1]

[1]Institute of Computer Graphics and Algorithms, TU Wien, Austria
[2]Department of Integrative Structural and Computational Biology, The Scripps Research Institute, La Jolla, California, USA.
[3]Department of Informatics, University of Bergen, Norway

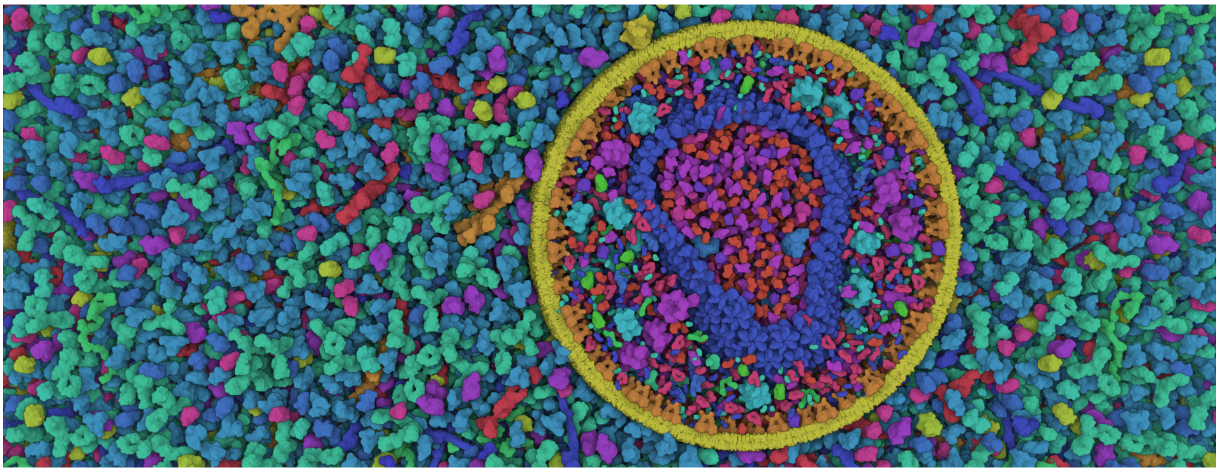


Figure 1: *Real-time screen-shot of an illustrative cross-section of the HIV virus surrounded by blood plasma. Our rendering tool is directly integrated in the Unity3D game engine and is able to render datasets with up to 15 billion atoms smoothly at 60Hz and in high resolution. Because these datasets exhibit high visual complexity, we opted for an illustrative rendering style to improve shape perception, inspired by the style of scientific illustrators.*

Abstract

In this article we introduce cellVIEW, a new system to interactively visualize large biomolecular datasets on the atomic level. Our tool is unique and has been specifically designed to match the ambitions of our domain experts to model and interactively visualize structures comprised of several billions atom. The cellVIEW system integrates acceleration techniques to allow for real-time graphics performance of 60 Hz display rate on datasets representing large viruses and bacterial organisms. Inspired by the work of scientific illustrators, we propose a level-of-detail scheme which purpose is two-fold: accelerating the rendering and reducing visual clutter. The main part of our datasets is made out of macromolecules, but it also comprises nucleic acids strands which are stored as sets of control points. For that specific case, we extend our rendering method to support the dynamic generation of DNA strands directly on the GPU. It is noteworthy that our tool has been directly implemented inside a game engine. We chose to rely on a third party engine to reduce software development work-load and to make bleeding-edge graphics techniques more accessible to the end-users. To our knowledge cellVIEW is the only suitable solution for interactive visualization of large bimolecular landscapes on the atomic level and is freely available to use and extend.

© The Eurographics Association 2015.

delivered by
EUROGRAPHICS
DIGITAL LIBRARY
www.eg.org diglib.eg.org

DOI: 10.2312/vcbm.20151209

1. Introduction

Computational biology already offers the means to model large structural models of cell biology, such as viruses or bacteria on the atomic level [JGA*14] [JAAA*15]. Visualization of macromolecular structures plays an essential role in this modelling process of such organisms. The most widely known visualization softwares are: VMD [HDS96], Chimera [PGH*04], Pymol [DeL02], PMV [S*99], ePMV [JAG*11]. These tools, however, are not designed to render a large number of atoms at interactive frame-rates and with full-atomic details (Van der Walls or CPK spherical representation). Megamol [GKM*15] is a state-of-the-art prototyping and visualization framework designed for particle-based data and which currently outperforms any other molecular visualisation software or generic visualization frameworks such VTK/Paraview [SLM04]. The system is able to render up to 100 million atoms at 10 fps on commodity hardware, which represents, in terms of size, a large virus or a small bacterium. Larger bacteria, however, such as the well known *E. coli*, made out of tens of billions of atoms, which is two orders of magnitude bigger than what the highest-end available solution is able to render.

According to our domain experts, responsive visual feedback is of a great value for the modelling process of such organisms. However, none of the currently available solutions are able to serve the ambitions of our domain experts, which is to model large macromolecular structures such as *E. coli*. Related works have already presented bleeding-edge techniques that can render large datasets with up to billions of atoms at interactive framerates on commodity graphics hardware [LBH12] [FKE13] [LMPSV14]. However, to our knowledge, the tools which implemented these techniques were either not publicly available, or remained in the prototyping stage. Indeed, a very cumbersome task for researchers is releasing and maintaining a usable version of the source code once the article has been published. The presented techniques are often a proof-of-concept that would require substantial software development work to ensure a maximum degree of accessibility. Unfortunately, this is often omitted because of a busy research schedule and is simply left in the hand of interested third party developers. Consequently, if this task remains unachieved, end-users are unlikely to use state-of-the-art techniques in their work.

cellVIEW is a new solution that enables fast rendering of very large biological macromolecular scene. Unlike Megamol, which is designed for generic particle-data, cellVIEW is primarily designed for large biomolecular landscapes, and thus, exploits the repetitive nature of such structures to improve the rendering performance. While the main function of this tool is to assist our domain experts in their modelling task, the visualization of these datasets could also serve an educational purpose. By interactively showcasing the machinery of life in science museums, for instance, we could also improve the understanding of basic cell biology of the laymen audience.

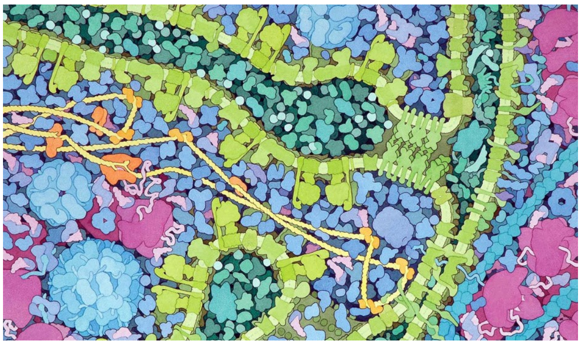


Figure 2: *An illustration of David Goodsell depicting a cross section of a Mitochondrion. Given the complexity of the scene the artist deliberately chose to render molecules with highly abstracted shapes.*

cellVIEW is built on top of state-of-the-art techniques, and also introduces new means to efficiently reduce the amount of processed geometries. The approach we demonstrate in cellVIEW improves rendering performance compared to related work by introducing efficient occlusion culling and robust level-of-detail schemes. Our level-of-detail scheme also abstracts the shape of macromolecules efficiently, thus reducing visual clutter, as seen on the artistic depictions of David Goodsell in Figure 2. We showcase our tool with real, large-scale scientific data such as the HIV virus and Mycoplasma bacterium, which were provided by our cooperating domain scientists. Their datasets, not only contain information relative to the location of individual macromolecules, but also provide the path of nucleic acids strands, which is stored in the form of control points. We additionally extend our method to procedurally generate DNA strands on-the-fly via the GPU tessellation shader, thus reducing the modelling effort as well as GPU transfer times and memory space. Our system is implemented using a user-friendly and popular game engine. The ease of use of the engine guarantees our tool a maximum degree of accessibility, thus bridging the gap between bleeding-edge techniques and actual use in real applications. Additionally, since game engines are gaining in popularity among the visualization community, we anticipate third-party users adopting our tool, and thereby breaking the barriers caused by heterogeneous toolset usage across research departments.

2. Related Work

Large-scale Molecular Visualization Lindow *et al.* [LBH12] have first introduced a method capable of quickly rendering large-scale atomic data consisting of several billions of atoms on commodity hardware. Rather than transferring the data from CPU to GPU every frame, they store the

© The Eurographics Association 2015.

structure of each type of molecule only once and utilize instancing to repeat these structures in the scene. For each type of protein a 3D grid structure containing all the atoms is created and then stored on the GPU memory. Upon rendering, the bounding boxes of the instances are drawn and individually raycasted, similar to volumetric billboards [DN09]. Subsequently Falk *et al.* [FKE13] presented a similar approach with improved depth culling and hierarchical ray casting for impostors that are located far away and do not require a full grid traversal. Although this implementation features depth culling, their method only operates on the fragment level, while they could have probably benefited from a culling on the instance level too. With their new improvement they managed to obtain 3.6 fps in full HD resolution for 25 billion atoms on a NVidia GTX 580, while Lindow *et al.* managed to get around 3 fps for 10 billions atoms in HD resolution on a NVIDIA GTX 285. Le Muzic *et al.* [LMPSV14], introduced another technique for fast rendering of large particle-based datasets using the GPU rasterization pipeline instead. They were able to render up to 30 billions of atoms at 10 fps in full HD resolution on a NVidia GTX Titan. They utilize tessellation shaders to inject atoms on-the-fly into the GPU pipeline similar to the technique of Lampe *et al.* [LVRH07]. In order to increase the rendering speed they dynamically reduce the number of injected atoms according to the camera depth. To simplify the molecular structures they discard atoms uniformly along the protein chain and increase the radius of remaining atoms to compensate for the volume loss. This level-of-detail scheme offers decent results for low degrees of simplification, but it does not guarantee preserving the initial shape of the molecules, resulting in poor image quality with highly simplified shapes.

Occlusion Culling A key aspect when rendering large and complex scenes is efficient occlusion culling. Grottel *et al.* [GRDE10] presented a method to perform coherent occlusion culling for particle-based datasets, which is closely related to Deferred Splatting [GBP04] and relies on temporal coherency. Their particle data is stored in a uniform grid, and they operate the culling at two-levels: at the level of grid cells first, and at the atomic level afterwards. Individual atoms are rendered via 2D depth impostors, because they have a much lower vertex count than sphere meshes for the same results. At the beginning of each frame they render an early depth pass with atoms that were visible during the previous frame. This pass results in an incomplete depth buffer that they utilize to determine the visibility of the remaining particles. For the coarse-level culling they determine the visibility of the grid cells by testing their bounding boxes against the incomplete depth buffer via hardware occlusion queries (HOQ). For the fine-level culling they test the visibility of individual atoms in the final render using the well known hierarchical Z-buffer (HZB) visibility technique [GKM93]. They construct the HZB from the incomplete depth buffer beforehand, and during the final render, they discard fragment operations from the vertex shader if the visibility test fails, thus compensating for the lack of early fragment rejection with depth impostors.

Illustrative Molecular Visualization When rendering large structures the speed of execution is not the only concern. As the structures increase in size, they are also increasing in complexity, and it is necessary to display the data in the most suitable way. Ambient occlusion, for instance, has been shown to play an essential role when dealing with large molecular structures, as it provides important depth cues which increase shape perception [GKSE12, ESH13]. But the rendering style is not the only means to define visual encoding, geometric abstraction should be applied as well. Parulek *et al.* [PJR*14], demonstrated a continuous level-of-detail scheme for molecular data. Their object-space approach offers detail-on-demand in the focus area while applying gradual shape simplification schemes elsewhere. At the finest level of detail they were showcasing solvent excluded surface (SES) representation and abstracted molecular shape for distant molecule. They introduced an interesting abstraction approach, other than molecular surfaces, based on union of spheres obtained via clustering methods. Several common clustering methods are compared and evaluated.

Modelling of Nucleic Acids Chains DNA plays a key role in cell biology, and thus is an important part of our datasets. Therefore, as with protein data, we shall also provide the means for efficient rendering of this type of structure. There are several scientific modeling tools [MC98, LO08, HLLF13] designed to generate DNA strands from a simple set of control points. These techniques, however, are all performed on the CPU, which means that geometry data must be uploaded on the GPU prior to the rendering. Because of the cost of transferring data from CPU to GPU, such approach would likely perform poorly when rendering and animating long DNA strands. Therefore, we introduce a new GPU-based approach which relies on dynamic instancing of DNA base-pairs along a curve. This approach is similar to the work of Lampe *et al.* [LVRH07], who use the geometry shader to dynamically instantiate residues along the protein backbone. The major difference here is the introduction of procedural building rules based on scientific data and the use of the tessellation shader, which offer a much greater bandwidth of injected primitives. Moreover, by changing the building rules, our approach can also be extended and applied to fibres or repetitive objects that are present in cellular environment (actin filaments, microtubules, lipoglycane, etc.).

Game Engines and Biomolecular Visualization Game engines are becoming increasingly popular in the molecular visualization community. Shepherd *et al.* [SZA*14] have developed an interactive application to showcase 3D genome data using a game engine. Their visualization is multi-scale and is able to render a large amount of data thanks to the implementation of a level-of-detail scheme. Various works on interactive illustration of biological processes have also mentioned using game engines to interactively visualize biomolecular processes in 3D, such as polymerization [KPV*14] and membrane crossings [LMWPV15]. Similarly to our work, Baaden *et al.* [LTDS*13] developed a molecular viewer which offers artistic and illustrative rendering methods based on the Unity3D game engine. Their primary intention was to democratize biomolecular visualization thanks to the use of a more intuitive and user friendly framework. Their tool has managed to prove that game engines are also useful in serious visualization projects. One noticeable technical difference between cellVIEW and UnityMol, other than the scale of the supported datasets, is that our tool is fully integrated in the "What you see is what you get" (WYSIWYG) editor of the Unity3D engine. Thus, our tool coexists with the engine toolset which provides a rich set of functionalities that can be directly used to enhance the quality of our visualization.

3. Efficient Occlusion Culling

The overwhelmingly increasing size of structural biology datasets calls for efficient means for reducing the amount of processed geometries. Our rendering pipeline is based on the work of Le Muzic *et al.* [LMPSV14], which relies on the tessellation shader to dynamically inject sphere primitives in the pipeline for each molecule. However, without proper occlusion culling, the injection of sphere primitives would still be performed, even if a molecule is completely hidden behind occluders. The presented occlusion culling method is inspired by the work of Grottel *et al.* [GRDE10]. We have revisited their technique to provide efficient occlusion culling for macromolecular datasets that are several orders of magnitude larger than the ones showcased with their method.

3.1. Temporal coherency

We developed a custom visibility technique, implemented with compute shaders and using the well-known hierarchical Z-buffer (HZB) occlusion culling. This solution has the advantage to reduce GPU driver overhead compared to HOQ used by Grottel *et al.* [GRDE10], since multiple queries can be performed in a single call. The approach rely on the use of an item-buffer to precisely determine the visibilty of the molecules at the end of a frame. Then at the beginning of the next frame, the previously visible molecules are firstly drawn. This will result in an partially complete frame, in case of eventual camera motion. The next step is to determine the remaining visible elements in order to complete the frame. We generate the HZB from the partially complete depth buffer and we compute the visibility information for the remaining molecules. The remaining visible molecules are finally drawn and we use the item buffer to determine which molecules are present on the screen at the end of this frame. The sequential steps of our occlusion culling method for a given frame are laid down as follows:

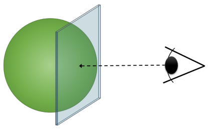


Figure 3: *Depth conservative sphere impostors, in order to benefit from early depth culling for depth impostors we must guaranty that the output depth will be greater than the depth of the billboard.*

1. Clear HZB and depth buffer
2. Draw visible molecules at the previous frame
3. Generate HZB from the depth buffer obtained in step 2
4. Compute HZB-visibility for the remaining molecules
5. Draw HZB-visible molecules from step 4
6. Find visible molecules via item-buffer for the next frame

3.2. Accelerating Texture Writes

Individual atoms are rendered via 2D sphere impostors, because they have a much lower vertex count than sphere meshes for the same results. The depth of the sphere impostors is corrected in the fragment shader in order to mimic a spherical volume. Upon drawing the atoms, many of them are actually occluded by other atoms of the same or surrounding molecules. These atoms would normally be processed, as a well-known limitation of graphics hardware, so far, has been the lack of early depth fragment rejection for depth impostors. Thanks to advances in graphics hardware however, it is now possible to activate early depth rejection when a fragment is modifying the output depth value. Hence, thanks to this feature, we may now easily avoid fragment computation for hidden atoms. This feature is called conservative depth output. Once activated, in order for conservative depth output to work, we must output a depth which is greater than the depth of the 2D billboard. This way the GPU is able to tell if a fragment will be occluded beforehand by querying the visibility internally. A description of the depth conservative output sphere impostor is given in Figure 3. Additionally, to limit the number of texture writes, we only output the id of the molecules to the render texture upon rendering. The colors are fetched afterwards in post-processing by reading the molecules properties from the id.

4. Twofold Level-of-Detail

Proteins are key elements of biological organisms, and thus it is important to visualize them in order to understand how these work. They are also present in fairly large quantities, which is challenging to render interactively without proper level-of-detail schemes (LOD). Additionally, their complex shapes might cause a high degree of visual clutter, which may render overly complex images. We propose a twofold LOD scheme which provides rendering acceleration and of-

© The Eurographics Association 2015.

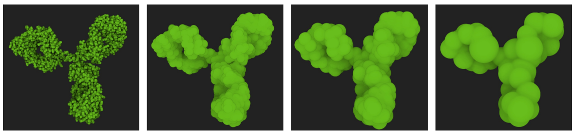


Figure 4: *Our level-of-detail scheme allows to reduce the number of sphere primitives from 10182 to 50 while preserving the overall shape of the protein. From left to right, the protein is shown with (i) full-atomic detail, (ii) only 15 percent of the overall sphere count, (iii) 5 percent and (iv) 0.5 percent.*

fers a clearer depiction of the scene using smoothly abstracted shapes. Our technique also offer a seamless continuum between the different levels of abstractions from highly detailed to highly abstracted.

Our rendering pipeline is based on the work of Le Muzic *et al.* [LMPSV14], where LOD was dynamically determined during the tessellation stage. To reduce the number of spheres, atoms were periodically skipped along the protein backbone, and the radii of remaining atoms were increased to compensate the volume loss. This technique, although fully dynamic, offers poor results for highly decimated molecules since it does not guarantee to preserve the overall shape. We employ clustering methods instead, similar to the technique of Parulek *et al.* [PJR*14], to simplify the shape of the molecules and reduce the number of primitives to render. Atoms corresponding to one cluster are replaced by a single sphere with a radius that approximates the size of the cluster. Clustering offers a very good decimation ratio as well as accurate shape abstraction, because it tends to preserve low-frequency details. With higher shape accuracy we are also able to switch to simpler LOD proxies closer to the camera, thus gaining in render speed without compromising image quality.

The clustering of the molecules is precomputed and results in a set of spheres which are stored in the GPU memory. We compute our LOD levels using a GPU-based K-means clustering algorithm. In our tests, we deemed that four levels were sufficient with our current datasets. The compression factor of each level was manually chosen to obtain the best performance/image quality ratio. The results of the clustering of our four levels is shown in Figure 4. These parameters can be easily changed via the editor interface. A side-by-side comparison between our illustrative LOD compared to full atomic detail is provided in Figure 5.

5. Dynamic DNA Generation

Animating individual molecules is fairly straightforward because modifying the atomic structure may not be required. In the case of DNA, the positions of the control points of the DNA path highly influence its structure, namely the positions and rotations of the individual nucleic acids. As a result, each modification of the control points of the DNA path requires a new computation of the strand. Current approaches are only performed on the CPU, [HLLF13, LO08, MC98] which means that the whole nucleic acids chain has to be transferred to the GPU upon re-computation. While this approach is viable for low to mid sized DNA strands, it is likely to perform poorly for large and dynamic DNA paths featuring a large number of control points.

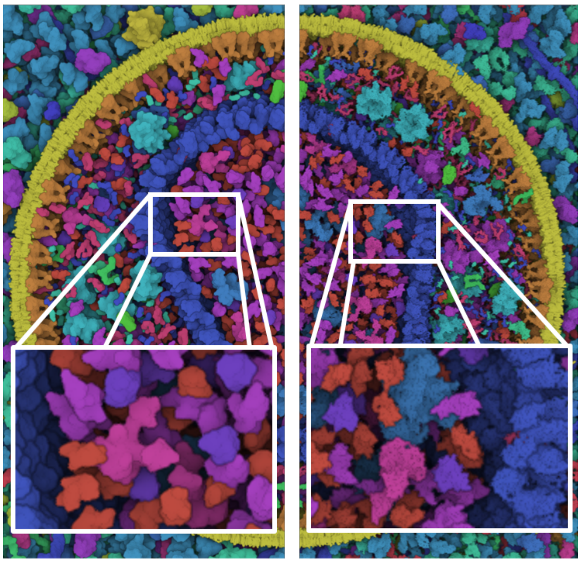


Figure 5: *Side-by-side comparison of our illustrative LOD compared with full atomic details. Our illustrative LOD provides smoother and elegant shapes, while also reducing the processing load.*

We propose to use the dynamic tessellation to leverage the generation of nucleic acid strands. So far we have only used tessellation to instantiate data stored in the GPU memory. However it is possible to include building rules characteristic to the DNA's well known geometry to procedurally generate a double helix structure simply based on control points. Thus, data transfers as well as GPU memory space can be dramatically reduced.

Similar to GraphiteLifeExplorer [HLLF13], our goal is more illustrative than strict biomolecular modeling. Therefore we privilege rendering performance over accuracy, and we provide only a limited array of folding types. Although the study of DNA structures has revealed many different types of folding, requiring complex modeling algorithms, the most commonly recognizable shape that of B-DNA that exhibits a regular structure which is simple to model: a spacing of 3.4Å and a rotation of 34.3° between each base. Based on these rules we are able to procedurally generate B-DNA strands based on path control points via GPU dynamic tessellation. The workflow which we employ is described as follow:

© The Eurographics Association 2015.

1. Resample control points (on the CPU).
2. Compute smooth control point normals (on the CPU).
3. Upload control point data to the GPU
4. Draw all the path segments in one pass, one vertex shader per segment
5. Read the control points and adjacent points needed for smooth cubic interpolation. (In vertex shader, for each segment)
6. Do uniform sampling along the cubic curve segment to determine the positions of the bases. (In vertex shader, for each segment)
7. Pass the position of the bases to the tessellation shader. (In vertex shader, for each segment)
8. Compute normal vector of each base using linear interpolation between the control points normals (In tessellation shader, for each base)
9. Inject atom, then translate and rotate accordingly (In tessellation shader, for each atom of each base)
10. Render sphere impostor from injected atom (In geometry & fragment shader, for each atom of each base)

5.1. Smooth Normals Computation

A well known challenge when dealing with 3D splines is to determine smooth and continuous frames along the whole curve. Any twists or abrupt variation in frame orientation would cause visible artifacts due to irregularities in the DNA structure, which should be avoided at all costs. We perform the computation of the smooth and continuous frames primarily on the CPU. We first determine the normal direction for every control point of the path. Then, we sequentially browse the control points and rotate the normal direction vector around the tangent vector in order to minimize the variation in orientation compared to the previous control point normal. The recalculated normals are then uploaded to the GPU along with the control points positions. Then, during the instantiation of the nucleic acids, we obtain the normal vector of a nucleic acid by linear interpolation between the two normal vectors of the segment.

5.2. Double Helix Instancing

When instancing individual pairs of nucleic acids in the tessellation shader, we first fetch the nucleic acid atoms, position them along the curve, orient them toward the normal direction and then rotate then around the tangent vector in order to generate the double helix. We always orient the first base of a segment according to the normal direction only, while the subsequent bases are all oriented towards the normal direction first and then rotated with an increasing angular offset of 34.3° around the tangent of the curve. The angular offset of a given base is defined as follows: $\alpha = i \times 34.3$, where i corresponds to the index of the base inside a segment. The last base of a segment must therefore always perform an offset rotation of $(360 - 34.3)^\circ$ around the tangent vector. This way it connects smoothly to the first base of the next segment, which is oriented towards the normal vector only. The result of the procedural generation of B-DNA is given in Figure 6 as well as a visual explanation of the different steps.

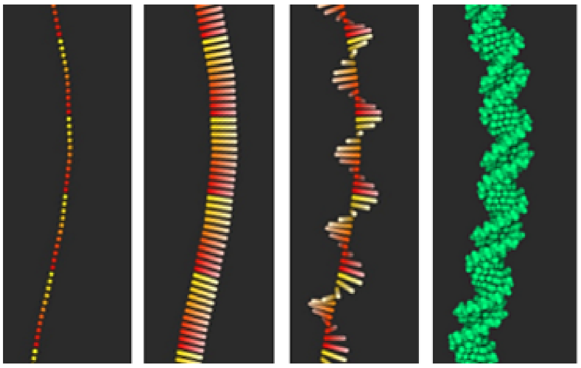


Figure 6: *Procedural generation of B-DNA structures via GPU dynamic tessellation. In the first image we can see the position of the individual bases. The color gradient highlights the individual segments. In the second image we draw the smooth normals along the curve, the color desaturation shows the direction of the vector. The third image shows the rotation offset of the normal vector along the tangent, and the last image shows the final result.*

5.3. Control Points Resampling

Given that the bases of a segment must perform a revolution to connect smoothly to the next segment, it is trivial to determine the number of bases per segment as follows: $n = 360 \div 34,3$. From the number of bases per segment we can easily deduce the required size of a segment as follows: $s = n \times 3.4$ Å, which results in a segment length of approximately 35 Å . This constraint implies that all control points be spaced uniformly with a distance of 35 Å. However, it may be the case that control points obtained via modelling software have arbitrary spacing. Therefore, we must resample the control points along the curve to ensure a uniform spacing before uploading it to the GPU. Although we resample the control points according to the B-DNA build rules, the length of the interpolated curve segments will always be slightly greater because of the curvature. We did not find this to be visually disturbing, mostly because consecutive segments in our dataset did not showcase critically acute angles, so the overall curvature of individual curve segments remained rather low.

6. Results

We have tested our tool with several datasets of different nature and sizes. The datasets were modelled with cellPACK [JAAA*15], a modelling tool for procedural generation of

© The Eurographics Association 2015.

Dataset	Size	Raw	LOD	O. Culling	LOD + O. Culling
HIV	15M	80	130	110	140
HIV + blood plasma	60M	25	90	60	120
HIV + blood plasma x 250	15B	<1	15	<1	60
Mycoplasma DNA	12M	70	n/a	n/a	n/a

Table 1: *Performance comparison for each dataset used in our study. During our tests we have monitored the rendering speed with various camera settings, from far-out to close-up and from many angles. The measured performance represents the slowest render speed obtained, in frame per seconds at full HD resolution. The first column shows the size of the dataset in terms of number of atoms, then from left to right: without optimizations, with LOD only, with occlusion culling only and finally with LOD and occlusion culling.*

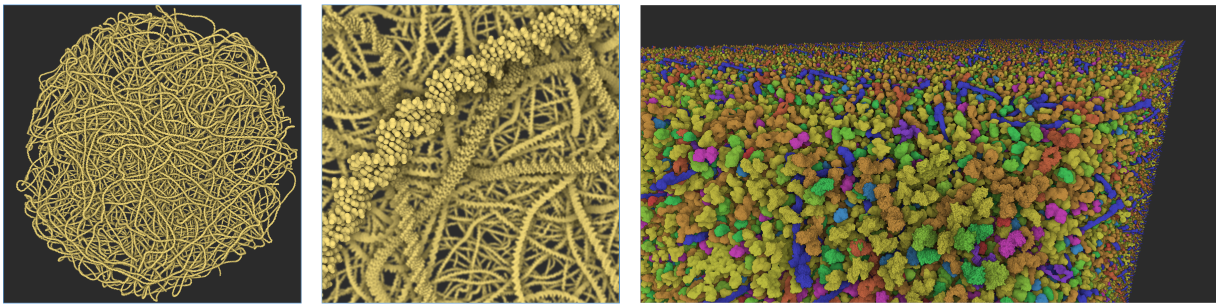


Figure 7: *The results of our rendering test, showing DNA from Mycoplasma on the left and HIV in blood plasma on the right. The first dataset has approximatively 11 million atoms and the second one approximatively 15 billion*

large biomolecular structures. cellPACK is developed and used by our domain experts, it is publicly available and offers to anyone the means for experimenting and creating their own models. Our program reads the files that are generated by cellPACK and is able to reconstruct and display the scene in a multiscale approach. The generated files comprise of a list of elements with their properties such as name, position, rotation, and PDB identifier that indicates the atomic structure [SLJ*98]. The structural data is directly fetched online from the Protein Data Bank via the PDB identifier. In case an entry is not present or refers to a custom PDB file, we load the protein information from a dedicated repository provided by the domain experts. The generated files also include control points for the linear or repetitive type of structures such as DNA, unfolded peptide, lypoglycane, etc.

6.1. Use cases

HIV Virus + Blood Plasma The first dataset we showcase is a combination of two datasets: the HIV virus [JGA*14] surrounded by blood plasma. The HIV is a retrovirus and thus only contains RNA, which features much more complex modeling rules than DNA and forbids dynamic procedural generation. For this specific case the atomic structure of RNA would have to be modelled ad-hoc with a third party tool before being loaded in cellVIEW. Without the genomic information, the dataset comprises a total of 60 millions atoms consisting in 40 different types of molecules. For the purpose of benchmarking, we periodically repeat this dataset to reach an overall number of 15 billion atoms.

Mycoplasma To demonstrate the use of our dynamic building rules for DNA, we use the data from Mycoplasma mycoide, one of the smallest bacteria with a genome of 1,211,703 baise pairs. Mycoplsama has been widely studied by biologists, and was the first organism to be fully synthetized. For this dataset we only showcase a preliminary model built with cellPACK and containing only a quarter of the total genome. This dataset comprises a set of 9617 control points defining the overall path of the DNA and the PDB reference of the nucleic acid base pairs. The pairs are instanced along the path resulting in an overall number of 11,619,195 atoms. We were able to procedurally generate and render the entire dataset at 70 fps without any culling nor LOD schemes. With this test we simply wanted to show the raw computation time in order to demonstrate the efficiency of our technique. Naturally, when using LOD and culling schemes like with protein data, the performance would considerably increase and would not impact the rest of the computation. The results of the two datasets are shown in Figure 7, and a preliminary render of the Mycoplasma is shown in Figure 8.

6.2. Performance Analysis

It is rather challenging to precisely evaluate the performance of our tool, as the speed of execution depends on many fac-

© The Eurographics Association 2015.

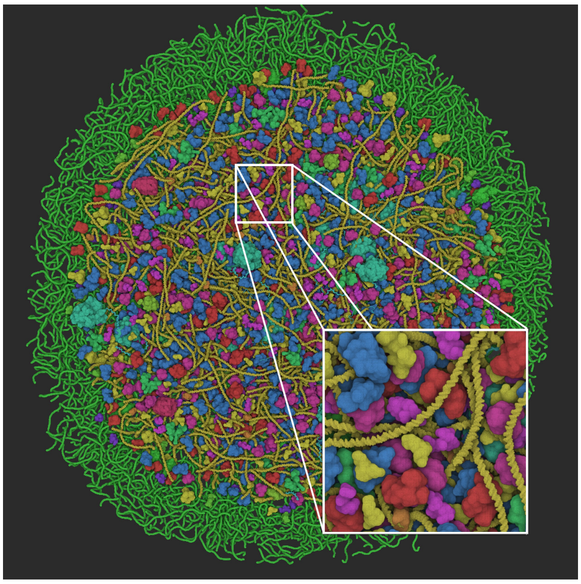


Figure 8: *Preliminary results of the Mycoplasma model. The model additionally features proteins, RNA and lycosomes.*

tors, such as camera position or level-of-detail parameters, and which are arbitrarily chosen. It is also worth mentioning that available software solutions are not able to deal with the amount of data presented in our largest datasets. Therefore we did not perform a thorough comparison with available software solutions and related work. We perform an intra-performance evaluation instead, using our different datasets.

Table 1 provides a descriptive listing of the rendering performance for each dataset, with and without our optimizations. The rendering tests were performed on an Intel Core i7-3930 CPU 3.20 GHz machine coupled with a GeForce GTX Titan X graphics card with 12GB of video RAM. During our tests we have monitored the rendering speed with various camera settings, from far-out to close-up and from many angles. The measured performance represents the slowest render speed obtained, in frame per seconds at full HD resolution. The LOD parameters were carefully tuned in order to obtain the best ratio between performance and image quality. From these results we can clearly see the impact of the LOD in terms of performance for all datasets. We can also observe that the culling greatly improves the rendering speed when displaying a very dense dataset. Additionally, our tool is able to render datasets which are equivalent in size to the ones showcased in related work at higher framerates (> 60fps).

It is worth mentioning that it would always be possible to render larger datasets at more than 60 fps using more aggressive LOD settings and thus trading image quality. However, one could question the utility of this approach to display datasets that would be one or several orders of magnitude larger. Indeed, when viewing our largest dataset in its entirety, the view starts to exhibit graining artefacts due to the very small screen-size of individual molecules. These artefacts create unwanted visual clutter, and therefore another type of approach rather than the particle-based one should be considered in this case.

7. Expert Feedback & Discussion

Domain experts who have experimented with cellVIEW have responded favorably and with great enthusiasm. One of our domain experts, a core actor of the cellPACK project, wrote:

> *Prior to cellVIEW, visualizing this type of data was cumbersome for the experts and as the scale increased, it was often not possible to view large models with all structures turned on with a standard computer. cellVIEW now provides state-of-the-art techniques to accomplish this task. Some experts were dismayed that cellVIEW could not yet be implemented in their lab's preferred or homemade visualization toolsets (i.e., not simply a python or C++ library they could access), but most had some experience working with the Unity 3D framework, so the transition to this standalone tool was sufficient. For large biological structures, such as Mycoplasma mycoides, the cellPACK viewers are currently unable to visualize the complete models produced by the packing algorithm. Because cellVIEW can handle Mycoplasma and larger models in atomic detail and with ease, it is evident that cellVIEW will become a critical tool for cellPACK users who wish to explore multi-scale modeling extremes such whole bacterial cells and ultimately whole mammalian cells.*

cellVIEW is open source, free to use, and available online, as well as the datasets modelled with cellPACK (https://github.com/illvisation/cellVIEW). With cellVIEW we wanted to guarantee the maximum degree of accessibility as possible. Therefore, we opted for a well-known, generic, and universal development framework to encourage third-party users to experiment and also to contribute to our project, such as visualization scientists, scientific illustrators, biologists or students. Although this solution might not have been the most preferred one for our main users at first, the ease of use of the tool has shown to be very valuable to them. The main advantage to us is that the development and maintenance of the core platform is already taken care of. This allows small teams of researchers to allocate their resources more efficiently and to focus on developing the actual technologies more quickly. However, the engine also presents a few drawbacks which would need to be addressed in the future in order to become a stronger contender as a visualization framework. Firstly, the advanced GPU programming features we use to develop cellVIEW are based on DirectX 11, which makes our tool only available to Windows platforms, at least until Unity3D supports advanced GPU programming with OpenGL. Another major drawback is that

© The Eurographics Association 2015.

the source code of the core of the engine is not yet publicly available, which may be critical in case a missing core feature would need to be manually coded.

8. Conclusions and Future Work

We have introduced cellVIEW, a tool for real-time multi-scale visualization of large molecular landscapes. Our tool is able to load files generated by cellPACK a powerful modeling tool for representing entire organisms at the atomic level. cellVIEW was engineered to work seamlessly inside the Unity3D game engine, which allows us to prototype and deploy quickly and to leverage performance via advanced GPU programming. The method which we presented also features notable improvements over previous works. We provide the means for efficient occlusion culling, which is crucial when dealing with such large scale datasets. We also implemented a level-of-detail scheme, which allows both acceleration of rendering times and provides a clear and accurate depiction of the scene. Finally, we demonstrated the use of dynamic tessellation to generate biolmolecular structures on-the-fly based on scientific modeling rules.

In future work we would like to tighten the collaboration with domain experts and achieve interactive viewing of more complex organisms and bacteria such as *E. coli*. As the scale increases the view exhibits highly grainy results due to the very small size of molecules. In the future we would like to focus on better representation for this case, and perhaps find new semantics that could be integrated in our level-of-detail continuum. We also would like to use our rendering to experiment with in-situ simulations as a visual exploration tool for scientists, and also as an educational tool to showcase the machinery of life to a lay audience.

Acknowledgement

This project has been funded by the Vienna Science and Technology Fund (WWTF) through project VRG11-010 and also supported by EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680. Additionally, this work has been carried out within the PhysioIllustration research project 218023, which is funded by the Norwegian Research Council. Autin,L. received support from the National Institutes of Health under award number P41GM103426.

References

[DeL02] DeLano W. L.: The pymol molecular graphics system. 2

[DN09] Decaudin P., Neyret F.: Volumetric billboards. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 2079–2089. 3

[ESH13] Eichelbaum S., Scheuermann G., Hlawitschka M.: Pointao improved ambient occlusion for point-based visualization. 3

[FKE13] Falk M., Krone M., Ertl T.: Atomistic visualization of mesoscopic whole-cell simulations using ray-casted instancing. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 195–206. 2, 3

[GBP04] Guennebaud G., Barthe L., Paulin M.: Deferred splatting. In *Computer Graphics Forum* (2004), vol. 23, Wiley Online Library, pp. 653–660. 3

[GKM93] Greene N., Kass M., Miller G.: Hierarchical z-buffer visibility. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), ACM, pp. 231–238. 3

[GKM*15] Grottel S., Krone M., Muller C., Reina G., Ertl T.: Megamol a prototyping framework for particle-based visualization. *Visualization and Computer Graphics, IEEE Transactions on 21*, 2 (2015), 201–214. 2

[GKSE12] Grottel S., Krone M., Scharnowski K., Ertl T.: Object-space ambient occlusion for molecular dynamics. In *Pacific Visualization Symposium (PacificVis), 2012 IEEE* (2012), IEEE, pp. 209–216. 3

[GRDE10] Grottel S., Reina G., Dachsbacher C., Ertl T.: Coherent culling and shading for large molecular dynamics visualization. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 953–962. 3, 4

[HDS96] Humphrey W., Dalke A., Schulten K.: Vmd: visual molecular dynamics. *Journal of molecular graphics 14*, 1 (1996), 33–38. 2

[HLLF13] Hornus S., Lévy B., Larivière D., Fourmentin E.: Easy dna modeling and more with graphitelifeexplorer. *PloS one 8*, 1 (2013), 53609. 3, 5

[JAAA*15] Johnson G. T., Autin L., Al-Alusi M., Goodsell D. S., Sanner M. F., Olson A. J.: cellpack: a virtual mesoscope to model and visualize structural systems biology. *Nature methods 12*, 1 (2015), 85–91. 2, 6

[JAG*11] Johnson G. T., Autin L., Goodsell D. S., Sanner M. F., Olson A. J.: epmv embeds molecular modeling into professional animation software environments. *Structure 19*, 3 (2011), 293–303. 2

[JGA*14] Johnson G. T., Goodsell D. S., Autin L., Forli S., Sanner M. F., Olson A. J.: 3d molecular models of whole hiv-1 virions generated with cellpack. *Faraday discussions 169* (2014), 23–44. 2, 7

[KPV*14] Kolesar I., Parulek J., Viola I., Bruckner S., Stavrum A.-K., Hauser H.: Illustrating polymerization using three-level model fusion. *arXiv preprint arXiv:1407.3757* (2014). 4

[LBH12] Lindow N., Baum D., Hege H.-C.: Interactive rendering of materials and biological structures on atomic and nanoscopic scale. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 1325–1334. 2

[LMPSV14] Le Muzic M., Parulek J., Stavrum A.-K., Viola I.: Illustrative visualization of molecular reactions using omniscient intelligence and passive agents. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 141–150. 2, 3, 4, 5

[LMWPV15] Le Muzic M., Waldner M., Parulek J., Viola I.: Illustrative timelapse: A technique for illustrative visualization of particle-based simulations. In *Visualization Symposium (PacificVis), 2015 IEEE Pacific* (2015), IEEE, pp. 247–254. 4

[LO08] Lu X.-J., Olson W. K.: 3dna: a versatile, integrated software system for the analysis, rebuilding and visualization of three-dimensional nucleic-acid structures. *Nature protocols 3*, 7 (2008), 1213–1227. 3, 5

© The Eurographics Association 2015.

[LTDS*13] LV Z., TEK A., DA SILVA F., EMPEREUR-MOT C., CHAVENT M., BAADEN M.: Game on, science-how video game technology may help biologists tackle visualization challenges. *PloS one 8*, 3 (2013), 57990. 4

[LVRH07] LAMPE O. D., VIOLA I., REUTER N., HAUSER H.: Two-level approach to efficient visualization of protein dynamics. *Visualization and Computer Graphics, IEEE Transactions on 13*, 6 (2007), 1616–1623. 3

[MC98] MACKE T. J., CASE D. A.: Modeling unusual nucleic acid structures. 3, 5

[PGH*04] PETTERSEN E. F., GODDARD T. D., HUANG C. C., COUCH G. S., GREENBLATT D. M., MENG E. C., FERRIN T. E.: Ucsf chimera a visualization system for exploratory research and analysis. *Journal of computational chemistry 25*, 13 (2004), 1605–1612. 2

[PJR*14] PARULEK J., JÖNSSON D., ROPINSKI T., BRUCKNER S., YNNERMAN A., VIOLA I.: Continuous levels-of-detail and visual abstraction for seamless molecular visualization. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 276–287. 3, 5

[S*99] SANNER M. F., ET AL.: Python: a programming language for software integration and development. *J Mol Graph Model 17*, 1 (1999), 57–61. 2

[SLJ*98] SUSSMAN J. L., LIN D., JIANG J., MANNING N. O., PRILUSKY J., RITTER O., ABOLA E.: Protein data bank (pdb): database of three-dimensional structural information of biological macromolecules. *Acta Crystallographica Section D: Biological Crystallography 54*, 6 (1998), 1078–1084. 7

[SLM04] SCHROEDER W. J., LORENSEN B., MARTIN K.: *The visualization toolkit*. Kitware, 2004. 2

[SZA*14] SHEPHERD J. J., ZHOU L., ARNDT W., ZHANG Y., ZHENG W. J., TANG J.: Exploring genomes with a game engine. *Faraday discussions 169* (2014), 443–453. 3

© The Eurographics Association 2015.

PAPER D

Visibility Equalizer: Cutaway Visualization of Mesoscopic Biological Models

Published in [LMMS+16] by the Eurographics Association.

DOI: 10.1111/cgf.12892
Eurographics Conference on Visualization (EuroVis) 2016
K.-L. Ma, G. Santucci, and J. van Wijk
(Guest Editors)

Volume 35 (2016), Number 3

Visibility Equalizer
Cutaway Visualization of Mesoscopic Biological Models

M. Le Muzic[†][1], P. Mindek[1], J. Sorger[1,2], L. Autin[3], D. S. Goodsell[3], and I. Viola[1]

[1]TU Wien, Austria [2]VRVis Research Center, Vienna, Austria [3]The Scripps Research Institute, La Jolla, California, USA

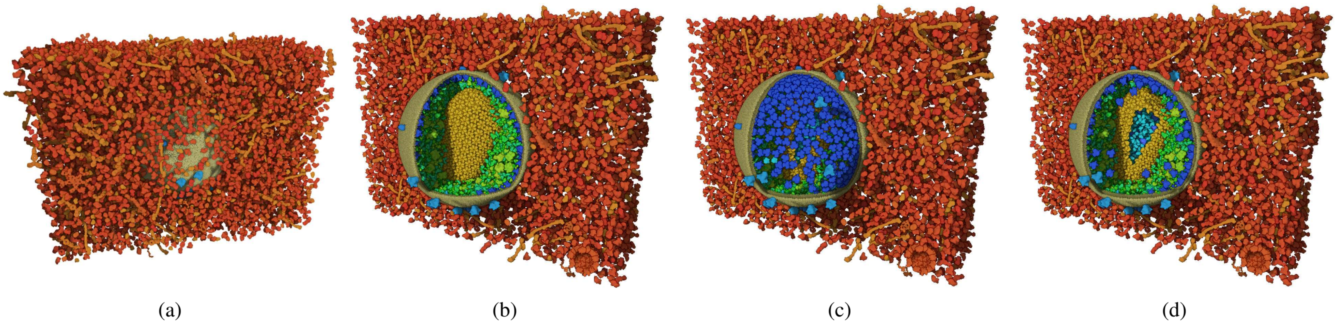


Figure 1: The workflow of our method for a model of HIV surrounded with blood plasma proteins. (a) The entire dataset is shown. The blood serum (shown in red) is occluding the virus. (b) Clipping objects are added to selectively clip molecules to reveal the HIV capsid. (c) The illustrator decides to show more of the matrix proteins (shown in blue), so their clipping is disabled. However, they are now occluding the view of the capsid. (d) The probabilistic clipping has been used to selectively remove those matrix proteins occluding the capsid, but some of them are left in the scene to indicate the presence of this type of protein on the virus membrane. The capsid has been clipped with view space clipping to reveal its internal structure.

Abstract

In scientific illustrations and visualization, cutaway views are often employed as an effective technique for occlusion management in densely packed scenes. We propose a novel method for authoring cutaway illustrations of mesoscopic biological models. In contrast to the existing cutaway algorithms, we take advantage of the specific nature of the biological models. These models consist of thousands of instances with a comparably smaller number of different types. Our method constitutes a two stage process. In the first step, clipping objects are placed in the scene, creating a cutaway visualization of the model. During this process, a hierarchical list of stacked bars inform the user about the instance visibility distribution of each individual molecular type in the scene. In the second step, the visibility of each molecular type is fine-tuned through these bars, which at this point act as interactive visibility equalizers. An evaluation of our technique with domain experts confirmed that our equalizer-based approach for visibility specification is valuable and effective for both, scientific and educational purposes.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms

---

[†] The first two authors contributed equally.
Contact: {mathieu | mindek}@cg.tuwien.ac.at

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

1. Introduction

Molecular biology is an emerging field that is characterized by rapid advances of the current state of knowledge. New discoveries have to be communicated frequently to a large variety of audiences. However, due to their structural and phenomenal complexity, it is challenging to convey the discoveries of molecular phenomena. On a mesoscale level, where thousands or millions of macromolecules form a given structure, this challenge is amplified by the presence of multiple spatio-temporal scales. Currently, illustrations are the most widely-used form of communicating mesoscale structures. To reveal the internal composition of mesoscale structures, such as viruses or bacteria, illustrators often employ clipping, section views or cutaways in their work.

Considering the rapid evolution of knowledge in the field of biology, it is necessary to adapt the traditional illustration pipeline so that new knowledge can be easily added into illustrations, instead of tediously redrawing the entire structure. Virtual 3D models of cells and other mesoscale molecular structures can be utilized for these purposes. Biologists have designed tools, such as *cellPACK* [JAAA*15], to procedurally generate 3D models that represent the structure of micro-organisms such as viruses, or entire cells at atomic resolution. Based on a set of input parameters, individual molecules are algorithmically assembled into these complex organic static structures. The parameter set consists of a specification of molecular types, concentrations and spatial distribution that define where the instances are distributed in a given compartment. The resulting 3D models, in the most complex cases, may consist of thousands of various molecular types, which in turn, may result in millions of molecules and billions of atoms. The instances are densely packed within the predefined compartments, to replicate the actual molecular crowding phenomena prevailing in living organisms. Due to the high density of these scenes, inner structures that are essential for conveying the function of the organism remain hidden. It is therefore important to develop visualization techniques that would procedurally reproduce the occlusion management methods used in traditional illustration. Currently, this is achieved by placing clipping objects in the scene, which remove specified parts of the displayed model. However, illustrators have to make sure that the essential information, e.g., the ratio of multiple molecular ingredients, is represented and not either hidden in the volume or clipped away (Fig. 2a). To do this, they would need to visually inspect the presence of each single ingredient in the resulting visualization (Fig. 2b).

To alleviate this process, we present our first contribution: a method that quantifies the overall visibility of the model contents. We display a stacked bar for each molecular type that encodes the ratio of visible, clipped, and occluded instances of the respective type for the current viewpoint and clipping setting. During the process of placing clipping objects in the scene, these bars continuously reveal molecular types that are underrepresented or overrepresented. This enables the illustrator to modify the placement of the clipping objects in such a way that every molecular type is adequately represented in the scene. We call this the *coarse level* of the visibility specification process.

To preserve important structures that would be removed by clipping objects such as cutting planes, traditional illustrations also often reintroduce parts of the removed structures in front of the revealed cross sections. In Figure 2b, for instance, the glycoproteins (yellow molecules) of the HIV particle that are not occluding the object of interest, in this case the capsid containing the RNA, are left in the illustration to communicate their presence on the surface of the virus (Fig. 2a). In this way, the main components of the virus particle can be illustrated in a single image. The process of fine-tuning the visibility is extremely time-consuming, as the illustrator has to manually pick individual molecular instances to be reintroduced or removed from the scene.

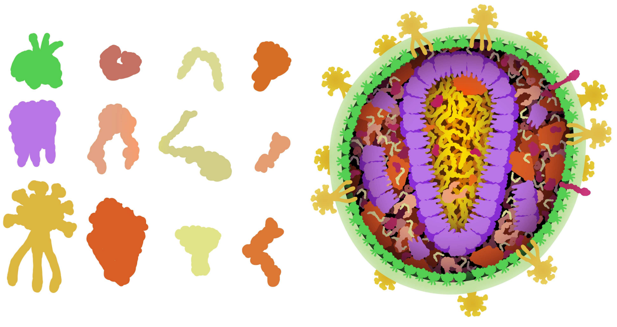

(a) (b)

Figure 2: (a) Various types of protein molecules and (b) their embedding into a mesoscale model. It is a demanding task to determine which molecular types are visible and how many of their instances are shown.

To speed up this visibility fine-tuning process, we propose our second contribution: a novel method for managing the visibility of instances. Instead of binary enabling or disabling the presence of all instances for a given molecular type, we offer the possibility to show a desired number of them. The main purpose of this approach is to increase the visibility of hidden structures by removing redundant parts of occluding instances while preserving some of them. An analogy of this approach can be drawn with "Screen-Door Transparency", a trivial way to obtain transparency in computer graphics by placing small holes in a polygon to reveal what is present behind. Additionally, we propose the novel metaphor of the *visibility equalizer* for controlling this effect efficiently. To explain its role, we use the metaphor of hi-fi sound reproduction where the *volume control* is used for adjusting the output sound *uniformly* on all frequencies. Such a mechanism corresponds to our coarse level visibility specification of the clipping objects, where all molecular types are uniformly removed from the clipped regions. However, hi-fi sound systems also allow users to fine-tune the volume levels of individual frequency bands through an *equalizer*. Following this metaphor, the visibility levels of molecular types are represented as stacked bars that form our visibility equalizer. All visibility bars are interactive, thus allowing the user to adjust desired levels just as with a sound equalizer. After drawing relation to previously proposed visibility management techniques in the next section, we will discuss the technique of visibility equalizers in further detail in the remaining sections of the paper.

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

2. Related Work

Related work can be categorized into occlusion management techniques and molecular visualization. We will concentrate on the former, according to the focus of this paper.

2.1. Occlusion Management

Related occlusion management techniques can be categorized into object centric approaches and transfer function based approaches. In object centric approaches, the geometry or parts of the volume that are obstructing one or more particular objects of interest are (partially) removed. In transfer function based approaches, the user assigns importances to intervals of the volume data values.

Object Centered Approaches. Cutaway and ghosting techniques were first introduced by Feiner & Seligmann [FS92] in 1992. Their work inspired several follow-up approaches [DWE02, DWE03, WEE03, VKG04, VKG05, KTH*05] that were later summarized in the survey by Viola & Gröller [VG05] under the collective term of *smart visibility* techniques. They coined this term to describe expressive visualization techniques that smartly uncover the most important features of the displayed data, i.e., cutaway views, ghosted views, and exploded views.

Krüger et al. [KSW06] developed a system that applies transparency and shading to enable focus&context visualization in volume data sets with a simple point&click interface. Li et al. [LRA*07] propose a cutaway design based on the geometry of the occluder in contrast to previous approaches that were based on the occludee. Burns & Finkelstein [BF08] applied the concept of importance-driven cutaways for volume data to polygonal models. Lawonn et al. [LGV*16] extend this approach to present a composite technique that combines the visualization of blood flow with the surrounding vessel structures. Baer et al. [BGCP11] published a perceptual evaluation of smart visibility techniques for two ghosted view approaches in comparison to semi-transparent approaches. The results clearly favored the ghosted view techniques. Sigg et al. [SFCP12] propose an approach for automatic cutaway box placement with optimized visibility for target features that are specified as degree-of-interest functions during interactive visual analysis of the volume data. Lidal et al. [LHV12] defined five design principles for cutaway visualization of geological models. The approach by Diaz et al. [DMNV12] preserves the relevant context information in volume clipping by allowing the user to extrude segmented surfaces such as bone structures from the clipping plane.

Since these approaches are object centered, they deal with (partial) occlusion of individual objects. For our data, partial or even complete occlusion of individual molecules is not an issue. The data is not composed of large singular entities such as polygonal or segmented volumetric objects where each single one has a semantic meaning. Instead, there are thousands or hundreds of thousands of instances that stem from only a couple of dozen molecule types. Therefore it does not matter if individual instances are occluded, as long as the structures that they form are preserved. Our approach is therefore fundamentally different from existing occlusion management approaches as it combines principles from object centered and transfer function based approaches.

Transfer Function Based Approaches. Since our molecule data is composed of a dense point cloud that resembles volumetric data on a nonregular grid, our approach is also related to transfer function based approaches. However, instead of a wide range of attribute values distributed over voxels, our data features a comparably smaller number of molecule types. This characteristic enables our visibility equalizer approach. The context-preserving volume rendering model [BGKG05] uses a function of shading intensity, gradient magnitude, distance to the eye point, and previously accumulated opacity to selectively reduce the opacity in less important data regions. Contours of surfaces that would be removed due to opacity remain visible, as the amount of illumination received is taken as a measure whether a point should be visible or not. Burns et al. [BHW*07] propose a multimodal approach that combines CT scan data and real-time ultrasound data. Importance driven shading is used to emphasize features of higher importance that have been revealed through the ghosting.

In his PhD thesis [Vio05], Viola presents an optimization strategy for automatically assigning visual mapping to voxels so that segmented objects in the volume are visible as specified by the user. Correa et al. [CM11] used a similar approach for applying visibility directly to voxels, without the notion of segmented objects. In our approach, we control visibility by interacting with the stacked bars of the visibility equalizer to modify the clipping object properties for each icndividual molecule type. Ruiz et al. [RBB*11] propose an approach for automatic transfer function optimization by minimizing the informational divergence between a user specified target distribution and the visibility distribution captured from certain viewpoints.

Transfer function based approaches are well suited for volumetric data that contains segmentable structures, such as the organs or bones in a medical scan. For molecular data this only holds partially true, as some types of molecules do indeed form continuous structures that could be made visible with a transfer function (e.g., membranes, nucleus). On the other hand, within these structures there is a more noise-like distribution of these molecules that cannot be segmented into solid structures.

2.2. Multi-Scale Visualization of Molecular Structures

Lindow et al. [LBH12] were the first to introduce a fast method for the real-time rendering of large-scale atomic data on consumer level hardware. They utilize instancing on the GPU to repeat these structures in the scene. For each molecule type, a 3D grid of the atoms is created and stored on the GPU. Falk et al. [FKE13] further refined the method with improved depth culling and hierarchical ray casting to achieve faster rendering performance for even larger scenes.

A novel and more efficient approach for rendering large molecular datasets was introduced by Le Muzic et al. [LMPSV14], which is based on brute-force rasterization rather than ray-tracing. To reduce the number of drawn geometries they rely on the tessellation shader to perform dynamic level-of-detail rendering. Their approach allows switching between different degrees of coarseness for the molecular structures on the fly, thus allowing the entire scene to be rendered in a single draw call efficiently, approaching

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

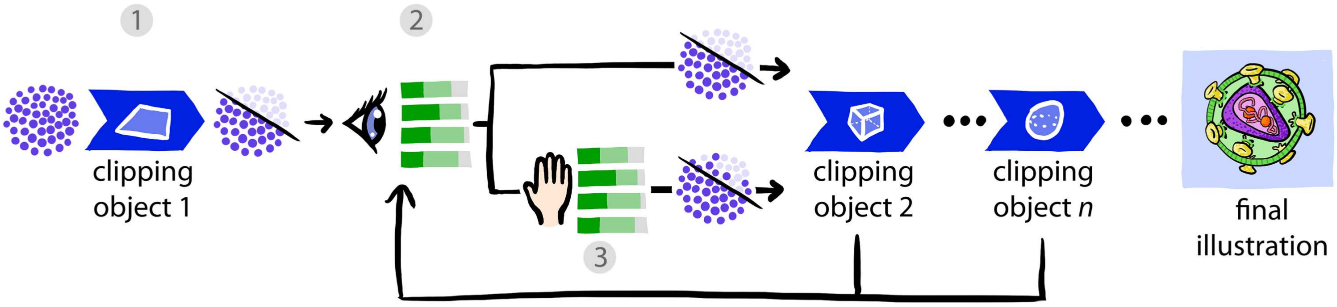


Figure 3: An illustration of the workflow with the visibility equalizer. (1) Clipping objects filter out elements in the data based on their type and location. (2) The clipping is applied in serial, i.e., the output of a clipping object constitute the input of the next one. The visibility information of the entire scene is routinely collected and updated in the visibility equalizer to keep the viewer informed about the current state of the data. (3) The clipping parameters of a given clipping object can later on be refined by interacting with the bar charts of the visibility equalizer to offer more control on the clipping, such as fuzziness.

zero driver overhead. As a follow up, they developed and released cellVIEW [LAPV15], a tool designed for rendering large-scale molecular scenes, which was implemented with Unity3D, a popular 3D engine. cellVIEW was primarily developed to showcase large molecular structures generated with cellPACK [JAAA*15], a tool developed to procedurally generate accurate and multi-scale models of entire viruses and cells.

Our visibility equalizer technique is built upon cellVIEW to improve the navigation and exploration of large molecular scenes generated with cellPACK. cellVIEW leverages GPU computing and parallel programming to enable real-time rendering and therefore, to provide a smooth and responsive user-experience, the visibility equalizer was developed with the same programming paradigm.

3. Overview

The two main components of our method are the *clipping objects* and the *visibility equalizer*. The workflow of our method is depicted in Figure 3. We distinguish between object space object clipping and view space object clipping. Object space clipping discards elements according to their distance to a geometric shape. View-space clipping discards elements according to whether or not they occlude certain objects in the current viewport. The role of the visibility equalizer is two fold: to provide important information about the visibility of molecular species, and to let users override the behavior of the clipping objects by directly manipulating the equalizer values. Each set of stacked bars show three types of quantitative information for a given type of ingredient: a - the ratio of visible instances to the total number of instances; b - the ratio of occluded instances to the total number of instances; c - the ratio of dicarded instances to the total number of instances. The visual encoding of the visibility equalizer is illustrated in Figure 4. By dragging the light green bar, the proportion of instances affected by the object-space clipping is modified, while dragging the dark green bar modifies the proportion of instances affected by the view-space clipping.

4. Object Space Clipping

Clipping objects define how instances shall be discarded depending on their location or type. They can operate either in object space or in view space. In this Section, we will explain in detail how clipping objects operate in object space.

4.1. Clipping Object Distance

Clipping objects are associated with geometric shapes to specify a region of the domain that is influenced by the clipping. Our system currently supports the following set of primitive shapes: plane, cube, sphere, cylinder and cone. We compute the distance between the instance centroids and the clipping region to identify instances that lie inside that region. To accelerate the computation, we solve the problem analytically using a mathematical description of the clipping region as a 3D signed distance field (SDF).

Due to the architecture of the rendering technique which is employed, the instances information (position, rotation, type, radius, etc.) is already stored large buffers stored on the GPU memory. To speed up the clipping operation and avoid data transfer costs, the distance is computed in parallel in a compute shader program, prior to the rendering, with one thread allocated per instance. The position, rotation, scale and geometry type of every clipping object must be additionally uploaded to the GPU in order to define the correct clipping region SDF. In a single thread and for each clipping object, the required information is fetched from the video memory and is used to compute the signed distance between an instance and the clipping object. When an object needs to be clipped, a dedicated flag in a GPU buffer is updated. The flag is then accessed during the rendering to discard the clipped instance.

4.2. Clipping Filtering

A clipping object also comprises a set of parameters that override the clipping state of instances based on their type. This filtering operation is performed if an instance is located inside the clipping region, in the same compute shader program described in Section 4.1. The first parameter controls the percentage of discarded instances

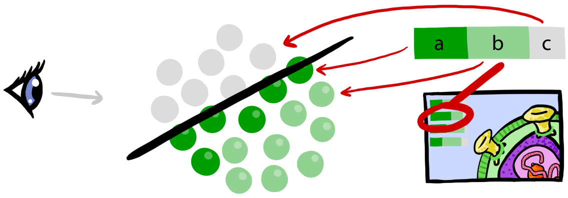


Figure 4: Illustration of the visibility equalizers. Each molecular ingredient has its own stacked bar showing (a) instances visible from the current viewpoint, (b) occluded instances, (c) instances clipped away by the clipping objects.

in the clipping region for a given type. We refer to this value as *object space clipping probability*. This value can be increased or decreased by dragging the light green bar in the visibility equalizer. It is important to mention that with multiple clipping objects, interaction with the light green bar in the visibility equalizer will only affect the clipping probability of the selected clipping object.

Upon start-up of the system, each instance is given a uniformly distributed random number between 0 and 1, and which will remain unchanged. Then, for each instance, we compare this value with the clipping probability in the computer shader program. If the constant random number is higher than the clipping probability, the instance is marked as discarded, and will not be rendered. For example, if the clipping probability value is equal to zero, all instances in the clipping region will be clipped, whereas if the value is equal to one, no instances will be clipped. A value between zero and one thus controls the degree of fuzziness of a clipping, as explained in Figure 5.

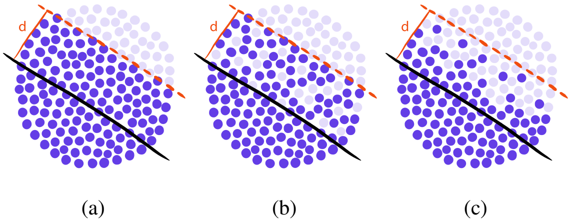
The other parameters allow users to control the clipping based on properties such as the size of the molecules (weight) and their overall number (concentration). Via the user interface and for a given clipping object, the user defines ranges that correspond to the desired concentration and molecular weight. Instances whose properties lie outside these ranges are simply discarded and will not be rendered.

4.3. Falloff Function

To increase control over the object space clipping, we use a falloff function. The falloff function gradually modulates the effect of the clipping with respect to the distance from the clipping shape as illustrated in Figure 5. The farther away from the clipping surface an instance is, the higher its clipping probability will be. The object space clipping probability of a molecule on the 3D position p is multiplied by the falloff function $f(p)$. The falloff function is defined as follows:

$$f(\vec{p}) = 1 - min(1, (d(\vec{p})/m)^c) \tag{1}$$

where $d(p)$ is the distance to the clipping surface from the point p. The function is parametrized by m and c, where m is the maximum distance up to which the object-space clipping probability takes effect, and c specifies the exponent of the falloff function. It is important to mention that the falloff function will not preserve the user-defined clip-ratio displayed in the visibility equalizer.



(a) (b) (c)

Figure 5: Illustration of the falloff function mechanism: (a) elements located further than distance d from the clipping object are clipped, (b) elements between the clipping object and d are uniformly clipped, (c) elements are removed gradually based on their distance to the clipping object to further customize its behaviour.

5. View Space Clipping

While object space clipping with primitive shapes allows for a high degree of flexibility, it may also require cumbersome manual operations for more complex set-ups. We therefore provide additional functionalities to selectively remove occluding instances in front of a set of ingredients set focus, to ensure them a maximum degree of visibility. The focus state can be manually specified via the visibility equalizer by ticking a dedicated checkbox in front of the stacked bars.

5.1. Occlusion Queries

When an ingredient type is set as focus, occluding instances of a different type may be selectively removed to reveal the occludees. To identify occluding instances, we perform occlusion queries. Nowadays, modern graphics hardware is able to perform occlusion queries easily using the fixed-functionality. This method requires one draw call per query, which may induce driver overhead with several thousands of queries. We compute the queries manually instead, using a custom shader program, because it allows the queries to be computed in a single GPU draw call, thus approaching zero driver overhead. In-depth technical details about this approach are described by Kubisch & Tavenrath [KT14].

We first render all the focused ingredients to an off-screen texture. This texture will then serve as a depth-stencil mask for the occlusion queries. There can be several ingredient types constituting the object of focus for a given clipping object. Thereafter, we render the bounding spheres of the potential occluders in a single draw call using instancing, on top of the depth-stencil mask. Thanks to early depth-stencil functionality, available on modern graphics hardware, fragments that will pass the test and be executed are garanteed to belong to an occluder. We then update the clipping state of the occluding instance by updating its corresponding occlusion flag stored in the main video memory directly from the fragment program.

5.2. Clipping Filtering

Similarly to the object space clipping, we provide an additional parameter to control the degree of fuzziness of the view-dependent

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

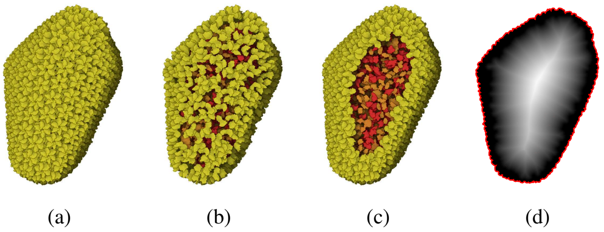


Figure 6: View-space clipping, (a) shows the full HIV Capsid, (b) shows the uniformly distributed clipping, (c) demonstrate the aperture effect and (d) shows the results of the 2D distance transform of the clipping mask.

clipping, which we refer to as *view-space clipping probability*. This value is set by the user for each ingredient type, and is modified by dragging the dark green bar in the visibility equalizer. The view-space clipping probability is evaluated after an instance is flagged as occluder in the same shader program mentioned in Section 5.1. We compare the clipping probability with a random number, initially defined and described in Section 4.2. If the random number is higher than the clipping probability, the instance will remain as clipped, otherwise it will be displayed. This will however result in a uniformly distributed number of visible occluders over the object of focus. Such a distribution might not always be the best design choice, because it fragments heavily the overall structure of the occluders and makes it difficult to see the occludees, as shown in Figure 6b.

We propose an alternative technique for fuzzy removal of occluding instances, which we dub the *aperture effect*. We define an additional parameter, the aperture coefficient, which controls the 2D distance from the instance to the edges of the occludees mask below which occluding instances shall be clipped. A example of the aperture effect is shown in Figure 6c. To enable this effect we compute the 2D distance transform of the occludees mask which we store in a separate off-line texture. We use the GPU Jump Flooding Algorithm by Rong & Tan [RT06] to interactively recompute the 2D distance field every frame. After the computation of the distance transform, the texture stores the distance to the contours of the shape for each pixel. Then, while computing occlusion queries in the fragment shader, we simply look-up the distance for the current fragment, and discard instances according to the user-defined aperture coefficient.

5.3. Contextual Anchoring

When observing still images of cut-away scenes, it might be challenging to perceive the depth of objects correctly, despite using lighting-based depth cues. We propose an additional method for depth guidance, which we call *contextual anchoring*. The concept is to override the results of clipping, to preserve elements located in proximity to non-clipped elements that would normally be clipped. This principle in shown in Figure 7, where we can observe parts of the green membrane anchored around channel molecules, and which indicate that they are located on the surface of the object.

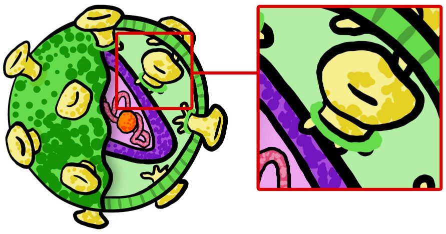


Figure 7: Illustration of contextual anchoring with an HIV particle. Despite the cutaway, some of the glyco-proteins (in yellow) are displayed and their surrounding lipid molecules (green) is preserved as contextual information.

We were able to procedurally reproduce this effect by applying a depth bias to the occlusion queries computation for selected focused molecules. This bias will ensure that contextual elements will no longer overlap the focus and will therefore be preserve as illustrated in Figure 8.

6. Equalizing Visibility

The visibility equalizer comprises a series of stacked bars that convey important visibility information for each ingredient type. The three colors correspond respectively to the number of visible, occluded and clipped instances, as explained in Figure 4. In order to fill the stacked bars with correct values, we must count the number of clipped and visible instances, and this operation must be repeated on every update.

6.1. Counting Clipped Instances

We perform the counting of the clipped instances on the GPU, in a dedicated compute shader program, since all the data already reside in the video memory. We previously declare a dedicated buffer on the GPU to hold the number of clipped instances for each ingredient type, and which shall be cleared before each counting operation. Counting the clipped instances is a rather straightforward task since the clipping state of each instance is routinely computed and stored in a dedicated GPU buffer. Once the clipping routine is performed, we simply browse through all the instances, and if an instance is flagged as clipped, we increase the counter in the GPU buffer that corresponds to the number of clipped instances for the given type. It is important to use an atomic increment operation for the counting to avoid concurrent accesses to the same counter value from different threads.

6.2. Counting Visible Instances

In order to count the number of visible instances for a given viewpoint, we first need to generate an instance buffer, which is a texture that contains, for each pixel, the unique instance id of the rendered molecule. We first start to flag visible instances in a post-processing shader, by browsing all the pixels of the instance buffer. In case an instance is present in the instance buffer, it is guaranteed to have at least one pixel visible on the screen, and it is therefore flagged as

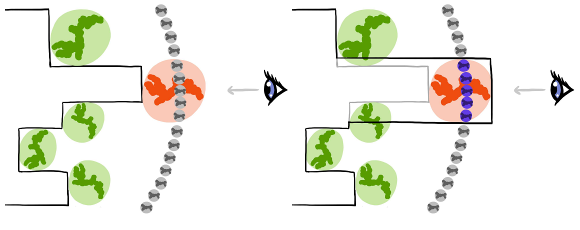

(a) (b)

Figure 8: The principle of the depth-bias used for contextual anchoring. The dark bars represents the depth values of the mask from the side, in one dimension. Elements in grey correspond to potential occluders, while elements in red and green correspond to occludees. The red type is subject to contextual anchoring. (a) Without contextual anchoring, the depth of occluders (grey) is overlapping the depth of the mask and will therefore be discarded. (b) With contextual anchoring, the depth of the occludees (red) is shifted so that context elements (purple) no longer overlap the focus and remain unclipped.

visible in a dedicated GPU buffer. To store the number of visible instances per type, we also need to declare an additional GPU buffer, which must be previously cleared each time visible instances are counted. In a second stage, similarly to the counting of the clipped instances, we browse through all the instances in a dedicated compute shader, while fetching the visibility information which was previously computed. Should an instance be flagged as visible, the counter that corresponds to the number of visible instances for the given type will be increased using an atomic increment operation. Once the information about the number of visible and clipped instances is obtained, the data is then transferred to the CPU and used to update the visibility equalizer.

7. Results and Performance Analysis

To showcase the capabilities of our method, we applied it to three different mesoscale molecular scenes. For the rendering, we used cellVIEW [LAPV15], a tool designed to efficiently render large molecular scenes on the GPU and implemented with Unity3D. The different datasets have been generated by the domain experts with cellPACK [JAAA*15], a modeling tool for procedural generation of large biomolecular structures. cellPACK summarizes and incorporates the most recent knowledge obtained from structural biology and system biology to generate comprehensive mesoscale models. Based on experimentally obtained data (such as proteins structure, concentration and spatial distribution), the tool is able to generate entire models of viruses and cells via a packing method based on collision constraints.

The first dataset is a model of an HIV particle in blood serum that contains 20502 instances of 45 different protein types and 215559 instances of lipid molecules. In Figure 9a, we show an example of a single clipping plane used to reduce the concentration of the blood serum molecules, so that the HIV proteins are visible. However, to avoid misleading the viewer about the actual concentration of the blood molecules, we render clipped proteins with a ghosting effect. This communicate true information about the concentration, while reducing visual clutter caused by the dense arrangement of blood serum proteins. Figure 1 shows sequential step for production a comprehensible cut-away illustration with the HIV dataset.

The second dataset is a model of *Mycoplasma mycoides* that contains 5380 proteins of 22 different types. Figure 9b shows how fuzzy-clipping is used to reduce visual clutter to illustrate the positions of the ribosomes (shown in blue) within the cell.

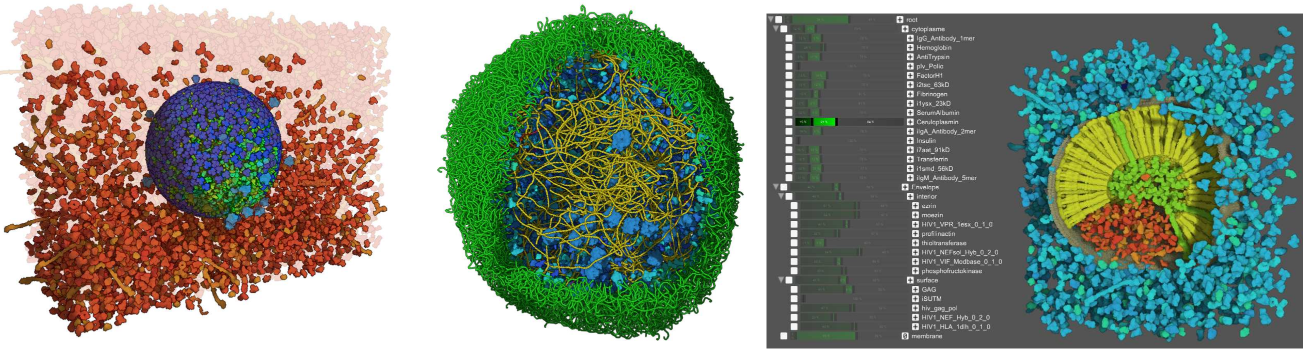
The third dataset, shown in Figure 9c is a model of an immature HIV which contains 1081 instances of 13 different protein types. We applied several clipping objects to reveal the internal structure of the virus. The blood serum (blue) has been preserved around the particle using the fuzzy clipping to illustrate how it encloses the HIV particle. The visibility equalizer is displayed as well, showing the ratios of visible and clipped instances of the individual molecular ingredients. The white boxes to the left of each stacked bar are used to mark the given ingredient or compartment as focus.

Figure 10 shows the mature HIV dataset clipped with a single plane. The contextual anchoring is applied to reintroduce parts of the clipped membrane (grey) around the envelope proteins (blue).

The visibility equalizer is designed to limit the computational overhead in order to offer a fast and responsive user experience. To demonstrate the responsiveness of our method, we measured the computation time for the object-space clipping, view-space clipping and 2D distance transform, respectively. The application was running on a machine equipped with an Intel Core i7-3930 CPU 3.20 GHz machine coupled with a GeForce GTX Titan X graphics card with 12GB of video RAM. The computation of the object-space clipping, compared to the rendering task performed by cellVIEW, is very lightweight and does not impact the overall performance too much. It took **0.3 milliseconds** to evaluate the 236061 instances of the HIV + blood dataset without clipping any of them. It took **0.5 milliseconds** in total to slice the dataset in half and **0.6 milliseconds** to clip it entirely. The increasing cost corresponds to the writing operations to the video memory, which are performed when an instance is clipped. It is important to mention that neither the shape of the clipping object nor the number of clipping objects have a meaningful influence on the performance.

The view-space clipping, however, requires more computational work that could impact the responsiveness. Indeed, for computing occlusion queries, occluders and occludees must be additionally rendered, which adds extra work to the rendering pipeline. For this reason, only the bounding spheres of the molecules are rendered instead of their entire structures, which may consist of hundreds or thousands of spheres, in order to guarantee a minimal computational overhead. We measured **0.07 milliseconds** for rendering the depth-stencil mask with 12142 instances (HIV proteins), and **0.57 milliseconds** for the computation of the 223919 occlusion queries corresponding to the remaining objects of the scene (blood proteins + lipid residues). Additionally, the 2D distance transform that is needed for the aperture effect also requires additional computation. It took **0.15 milliseconds** for computing the distance transform of the previous depth-stencil mask at a resolution of 512 by 512 pixels. Unlike object-space clipping, the view-space clipping computation cost will keep increasing with additional operations. Therefore, it

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.



(a) (b) (c)

Figure 9: Advanced clipping options in real test systems. (a) A falloff function is used to gradually clip serum molecules (red) from bottom to top to reveal the HIV capsid, with ghosting to give cues about the overall concentration. (b) Selective clipping is used to reveal the location of ribosome (blue) in a model of Mycoplasma mycoides. (c) Internal structures of a immature HIV model are shown by several clipping objects. On the left, the visibility equalizer is shown.

is a good strategy to keep a low number of view space clipping objects, especially with very large scenes.

8. User Feedback

We evaluated the usefulness of our tool by collecting informal user feedback from domain experts in biological illustration and molecular biology. In both cases, we did a remote walk-through introduction of our software, while collecting first impressions. Additionally, we gave them an executable version of the software and asked them to write a short summary of their experience after trying the tool by themselves. We first sent an early version of our tool to a biomedical illustrator with a strong academic background in chemistry. His overall feedback was very positive, he enjoyed the responsiveness of the tool, and the novel concept of fuzziness and gradient clipping. Here is a quote from his written feedback:

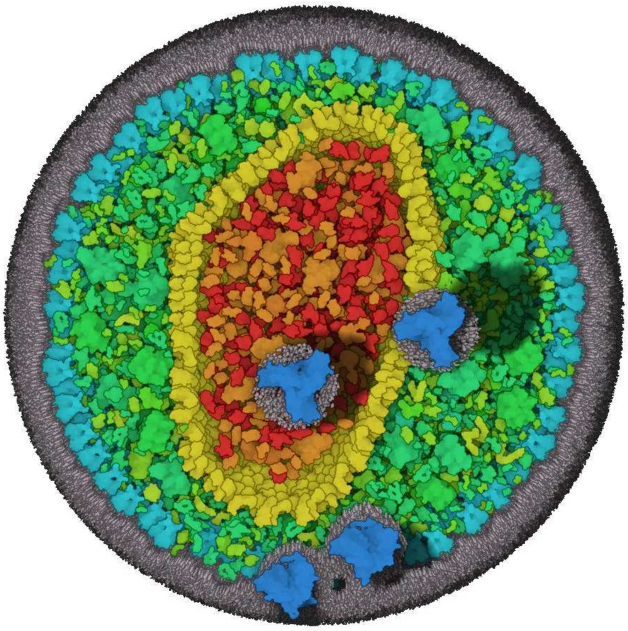


Figure 10: HIV clipped with a plane. Contextual anchoring is used to indicate the proximity of envelope proteins (dark blue) with the lipid membrane (grey) The dark spots represent shadows projected into interior proteins.

> *"...in my opinion it can be a very useful toolkit for an illustrator in the biomedical field...It also seems very promising for interactive teaching and also for animation purposes... One very useful feature of the software is the possibility to "cut" planes of interest of a particular space, and keeping the information of all "layers" by creating a "gradient" of concentration of the ingredients of the displayed molecular recipe. A visualization that resembles an "exploded model" but for biological assembly and it can be achieved without manually selecting every instance you would like to hide."*

Secondly, we interviewed an expert in the domain of molecular biology and visualization. For this second interview, the overall feedback was also quite positive. He greatly enjoyed how easy and fast it was to perform clipping, and also enjoyed the user interface for manipulating the cut object parameters. He also wished for several additional features to improve the usability of the tool, such as filtering based on biomolecular properties and rendering the ghosts of the clipped instances. These features have since been implemented in the current version of the software, as seen in Figure 9a. Here is a quote from the written feedback we collected:

> *"...The aperture cutting feature is especially useful for exploring a feature or object in the context of a crowded molecular environment. The ability to retain a subset of the clipped objects ("fuzzy clipping") is an interesting feature that could be very useful under certain circumstances. The feature is useful if one wants to get an impression of reducing the concentration of some of the molecular ingredients, or of what a gradient of certain molecular ingredients would look like."*

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

9. Conclusion and Future Work

In this paper, we present a novel method for authoring cutaway illustrations of mesoscopic molecular models. Our system uses clipping objects to selectively remove instances based on their type and location. To monitor and fine-tune the process, we introduce the visibility equalizer. It keeps the user informed about the number of molecular instances removed by the clipping objects, or occluded in the current viewport. Moreover, the visibility equalizer allows the users to directly override the behaviour of the clipping objects in order to fine-tune the visibility of molecular ingredients within the scene.

The visibility equalizer concept demonstrates a scenario where a visualization metaphor, such as the stacked bar chart, can serve as a user interface for performing a specific task, in our case to manipulate 3D data to authorize cutaways. The method allows users to create comprehensive illustrations of static biological models in realtime. This was confirmed by gathering feedback from domain experts. While the concept was applied to a specific domain, we also wish to develop other examples where the (information) visualization would act simultaneously as an interface to steer the view.

There are also several follow-up ideas which we would like to focus on in the future, to strengthen data exploration and showcasing with cellVIEW. Firstly, we would like to explore automatic clipping mechanisms to assist the user with the placement of clipping objects based on the nature of the scene and shape analysis. Secondly, we would also like to try our visibility equalizer concept with time-dependent datasets and enhance it to provide the means for authoring illustrations of dynamic datasets.

Our Visibility Equalizer method is built on top of cellVIEW and Unity3D, which are both free to use for non-commercial use, the source code is publicly available, as well as the showcased scenes modelled with cellPACK (https://github.com/illvisation/cellVIEW).

Acknowledgement

This project has been funded by the Vienna Science and Technology Fund (WWTF) through project VRG11-010 and also supported by EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680. Johannes Sorger has been partially supported in the scope of the FWF-funded project P24597-N23 (VISAR) and the COMET K1 program of the Austrian Funding Agency (FFG). Ludovic Autin received support from the National Institutes of Health under award number P41GM103426. We would like to thank Aysylu Gabdulkhakova and Manuela Waldner for insightful comments.

References

[BF08] Burns M., Finkelstein A.: Adaptive cutaways for comprehensible rendering of polygonal scenes. In *SIGGRAPH Asia* (Singapore, 2008), ACM, pp. 154:1–154:7. 3

[BGCP11] Baer A., Gasteiger R., Cunningham D., Preim B.: Perceptual evaluation of ghosted view techniques for the exploration of vascular structures and embedded flow. *Computer Graphics Forum 30*, 3 (2011), 811–820. 3

[BGKG05] Bruckner S., Grimm S., Kanitsar A., Gröller M. E.: Illustrative context-preserving volume rendering. In *Proceedings of the Seventh Joint Eurographics / IEEE VGTC Conference on Visualization* (Leeds, United Kingdom, 2005), EUROVIS'05, pp. 69–76. 3

[BHW*07] Burns M., Haidacher M., Wein W., Viola I., Gröller M. E.: Feature emphasis and contextual cutaways for multimodal medical visualization. In *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization* (Sweden, 2007), EUROVIS'07, pp. 275–282. 3

[CM11] Correa C., Ma K.-L.: Visibility histograms and visibility-driven transfer functions. *Visualization and Computer Graphics, IEEE Transactions on 17*, 2 (Feb 2011), 192–204. 3

[DMNV12] Díaz J., Monclús E., Navazo I., Vázquez P.: Adaptive cross-sections of anatomical models. *Computer Graphics Forum 31*, 7 (2012), 2155–2164. 3

[DWE02] Diepstraten J., Weiskopf D., Ertl T.: Transparency in interactive technical illustrations. *Computer Graphics Forum 21*, 3 (2002), 317–325. 3

[DWE03] Diepstraten J., Weiskopf D., Ertl T.: Interactive cutaway illustrations. *Computer Graphics Forum 22*, 3 (2003), 523–532. 3

[FKE13] Falk M., Krone M., Ertl T.: Atomistic visualization of mesoscopic whole-cell simulations using ray-casted instancing. *Computer Graphics Forum 32*, 8 (2013), 195–206. 3

[FS92] Feiner S., Seligmann D.: Cutaways and ghosting: satisfying visibility constraints in dynamic 3d illustrations. *The Visual Computer 8*, 5-6 (1992), 292–302. 3

[JAAA*15] Johnson G. T., Autin L., Al-Alusi M., Goodsell D. S., Sanner M. F., Olson A. J.: cellPACK: a virtual mesoscope to model and visualize structural systems biology. *Nature methods 12*, 1 (Jan. 2015), 85–91. 2, 4, 7

[KSW06] Krüger J., Schneider J., Westermann R.: Clearview: An interactive context preserving hotspot visualization technique. *Visualization and Computer Graphics, IEEE Transactions on 12*, 5 (Sept 2006), 941–948. 3

[KT14] Kubisch C., Tavenrath M.: Opengl 4.4 scene rendering techniques. *NVIDIA Corporation* (2014). 5

[KTH*05] Krüger A., Tietjen C., Hintze J., Preim B., Hertel I., StraussG.: Interactive visualization for neck-dissection planning. In *Proceedings of the Seventh Joint Eurographics / IEEE VGTC Conference on Visualization* (Leeds, United Kingdom, 2005), EUROVIS'05, pp. 295–302. 3

[LAPV15] Le Muzic M., Autin L., Parulek J., Viola I.: cellVIEW: a tool for illustrative and multi-scale rendering of large biomolecular datasets. In *Eurographics Workshop on Visual Computing for Biology and Medicine* (Sept. 2015), pp. 61–70. 4, 7

[LBH12] Lindow N., Baum D., Hege H.-C.: Interactive rendering of materials and biological structures on atomic and nanoscopic scale. *Computer Graphics Forum 31*, 3 (2012), 1325–1334. 3

[LGV*16] Lawonn K., Glasser S., Vilanova A., Preim B., Isenberg T.: Occlusion-free blood flow animation with wall thickness visualization. *Visualization and Computer Graphics, IEEE Transactions on 22*, 1 (Jan 2016), 728–737. 3

[LHV12] Lidal E. M., Hauser H., Viola I.: Design principles for cutaway visualization of geological models. In *Proceedings of Spring Conference on Computer Graphics (SCCG 2012)* (May 2012), pp. 53–60. 3

[LMPSV14] Le Muzic M., Parulek J., Stavrum A.-K., Viola I.: Illustrative visualization of molecular reactions using omniscient intelligence and passive agents. *Computer Graphics Forum 33*, 3 (2014), 141–150. 3

[LRA*07] Li W., Ritter L., Agrawala M., Curless B., Salesin D.: Interactive cutaway illustrations of complex 3D models. In *SIGGRAPH '07* (San Diego, California, 2007), ACM. 3

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd.

[RBB*11] RUIZ M., BARDERA A., BOADA I., VIOLA I., FEIXAS M., SBERT M.: Automatic transfer functions based on informational divergence. *Visualization and Computer Graphics, IEEE Transactions on 17*, 12 (Dec 2011), 1932–1941. 3

[RT06] RONG G., TAN T.-S.: Jump flooding in gpu with applications to voronoi diagram and distance transform. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2006), I3D '06, ACM, pp. 109–116. 6

[SFCP12] SIGG S., FUCHS R., CARNECKY R., PEIKERT R.: Intelligent cutaway illustrations. In *Visualization Symposium (PacificVis), 2012 IEEE Pacific* (Feb 2012), pp. 185–192. 3

[VG05] VIOLA I., GRÖLLER E.: Smart visibility in visualization. In *Computational Aesthetics in Graphics, Visualization and Imaging* (2005), The Eurographics Association. 3

[Vio05] VIOLA I.: *Importance-Driven Expressive Visualization*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, June 2005. 3

[VKG04] VIOLA I., KANITSAR A., GRÖLLER M. E.: Importance-driven volume rendering. In *Proceedings of the Conference on Visualization '04* (Washington, DC, USA, 2004), VIS '04, IEEE Computer Society, pp. 139–146. 3

[VKG05] VIOLA I., KANITSAR A., GRÖLLER M. E.: Importance-driven feature enhancement in volume visualization. *Visualization and Computer Graphics, IEEE Transactions on 11*, 4 (July 2005), 408–418. 3

[WEE03] WEISKOPF D., ENGEL K., ERTL T.: Interactive clipping techniques for texture-based volume visualization and volume shading. *Visualization and Computer Graphics, IEEE Transactions on 9*, 3 (July 2003), 298–312. 3

Bibliography

[AABA10] Steven S Andrews, Nathan J Addy, Roger Brent, and Adam P Arkin. Detailed simulations of cell biology with smoldyn 2.1. *PLoS Comput Biol*, 6(3):e1000705, 2010.

[ACZ$^+$12] Raluca Mihaela Andrei, Marco Callieri, Maria Francesca Zini, Tiziana Loni, Giuseppe Maraziti, Mike Chen Pan, and Monica Zoppè. Intuitive representation of surface properties of biomolecules using bioblender. *BMC bioinformatics*, 13(4):1, 2012.

[Ber06] Drew Berry. Apoptosis. `http://youtu.be/DR80Huxp4y8?t=1m50s`, 2006. The Walter and Eliza Hall Institute.

[BKW$^+$77] Frances C Bernstein, Thomas F Koetzle, Graheme JB Williams, Edgar F Meyer, Michael D Brice, John R Rodgers, Olga Kennard, Takehiko Shimanouchi, and Mitsuo Tasumi. The protein data bank. *European Journal of Biochemistry*, 80(2):319–324, 1977.

[Bol06] David Bolinsky. The inner life of a cell. XVIVO, 2006.

[CBS09] Deepak Chandran, Frank T Bergmann, and Herbert M Sauro. Tinkercell: modular cad tool for synthetic biology. *Journal of biological engineering*, 3(1):1, 2009.

[cel16] Cellblender. `https://github.com/mcellteam/cellblender`, 2016. Accessed: 2016-08-08.

[Cla] Clarafi. Molecular maya. https://clarafi.com/tools/mmaya.

[FKE13] Martin Falk, Michael Krone, and Thomas Ertl. Atomistic visualization of mesoscopic whole-cell simulations using ray-casted instancing. In *Computer Graphics Forum*, volume 32, pages 195–206. Wiley Online Library, 2013.

[FKRE09] Martin Falk, Michael Klann, Matthias Reuss, and Thomas Ertl. Visualization of signal transduction processes in the crowded environment of the cell. In *2009 IEEE Pacific Visualization Symposium*, pages 169–176. IEEE, 2009.

[FKRE10] Martin Falk, Michael Klann, Matthias Reuss, and Thomas Ertl. 3d visualization of concentrations from stochastic agent-based signal transduction simulations. In *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1301–1304. IEEE, 2010.

[FMKT03] Akira Funahashi, Mineo Morohashi, Hiroaki Kitano, and Naoki Tanimura. Celldesigner: a process diagram editor for gene-regulatory and biochemical networks. *Biosilico*, 1(5):159–162, 2003.

[Gam13] Strange Loop Games. Sim cell, 2013.

[GKM93] Ned Greene, Michael Kass, and Gavin Miller. Hierarchical z-buffer visibility. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 231–238. ACM, 1993.

[Goo01] Google. Google earth, 2001.

[GRDE10] Sebastian Grottel, Guido Reina, Carsten Dachsbacher, and Thomas Ertl. Coherent culling and shading for large molecular dynamics visualization. In *Computer Graphics Forum*, volume 29, pages 953–962. Wiley Online Library, 2010.

[HDS96] William Humphrey, Andrew Dalke, and Klaus Schulten. Vmd: visual molecular dynamics. *Journal of molecular graphics*, 14(1):33–38, 1996.

[HFS$^+$03] Michael Hucka, Andrew Finney, Herbert M Sauro, Hamid Bolouri, John C Doyle, Hiroaki Kitano, Adam P Arkin, Benjamin J Bornstein, Dennis Bray, Athel Cornish-Bowden, et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[HLLF13] Samuel Hornus, Bruno Lévy, Damien Larivière, and Eric Fourmentin. Easy dna modeling and more with graphitelifeexplorer. *PloS one*, 8(1):e53609, 2013.

[JAAA$^+$15] Graham T Johnson, Ludovic Autin, Mostafa Al-Alusi, David S Goodsell, Michel F Sanner, and Arthur J Olson. cellpack: a virtual mesoscope to model and visualize structural systems biology. *Nature methods*, 12(1):85–91, 2015.

[JAG$^+$11] Graham T Johnson, Ludovic Autin, David S Goodsell, Michel F Sanner, and Arthur J Olson. epmv embeds molecular modeling into professional animation software environments. *Structure*, 19(3):293–303, 2011.

[JM12] Jodie Jenkinson and Gaël McGill. Visualizing protein interactions and dynamics: evolving a visual language for molecular animation. *CBE-Life Sciences Education*, 11(1):103–110, 2012.

[Kar14] Martin Karplus. Development of multiscale models for complex chemical systems: from h+ h2 to biomolecules (nobel lecture). *Angewandte Chemie International Edition*, 53(38):9992–10005, 2014.

[KBK⁺08] Rex A Kerr, Thomas M Bartol, Boris Kaminsky, Markus Dittrich, Jen-Chien Jack Chang, Scott B Baden, Terrence J Sejnowski, and Joel R Stiles. Fast monte carlo simulation methods for biological reaction-diffusion systems in solution and on surfaces. *SIAM journal on scientific computing*, 30(6):3126–3149, 2008.

[KCVGC⁺05] Ingrid M Keseler, Julio Collado-Vides, Socorro Gama-Castro, John Ingraham, Suzanne Paley, Ian T Paulsen, Martín Peralta-Gil, and Peter D Karp. Ecocyc: a comprehensive database resource for escherichia coli. *Nucleic acids research*, 33(suppl 1):D334–D337, 2005.

[KGE11] Michael Krone, Sebastian Grottel, and Thomas Ertl. Parallel contour-buildup algorithm for the molecular surface. In *Biological Data Visualization (BioVis), 2011 IEEE Symposium on*, pages 17–22. IEEE, 2011.

[KPC14] Jonathan R Karr, Nolan C Phillips, and Markus W Covert. Wholecellsimdb: a hybrid relational/hdf database for whole-cell model predictions. *Database*, 2014:bau095, 2014.

[Kre08] The krebs cycle. `https://en.wikipedia.org/wiki/Citric_acid_cycle`, 2008.

[KSES12] Michael Krone, John E Stone, Thomas Ertl, and Klaus Schulten. Fast visualization of gaussian density surfaces for molecular dynamics and particle system trajectories. *EuroVis-Short Papers*, 1:67–71, 2012.

[LBH12] Norbert Lindow, Daniel Baum, and H-C Hege. Interactive rendering of materials and biological structures on atomic and nanoscopic scale. In *Computer Graphics Forum*, volume 31, pages 1325–1334. Wiley Online Library, 2012.

[LMAPV15] Mathieu Le Muzic, Ludovic Autin, Julius Parulek, and Ivan Viola. cellview: a tool for illustrative and multi-scale rendering of large biomolecular datasets. In *Proceedings of the Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 61–70. Eurographics Association, 2015.

[LMMS⁺16] Mathieu Le Muzic, Peter Mindek, Johannes Sorger, Ludovic Autin, David S Goodsell, and Ivan Viola. Visibility equalizer cutaway visualization of mesoscopic biological models. In *Computer Graphics Forum*, volume 35, pages 161–170. Wiley Online Library, 2016.

[LMPSV14] Mathieu Le Muzic, Julius Parulek, Anne-Kristin Stavrum, and Ivan Viola. Illustrative visualization of molecular reactions using omniscient intelligence and passive agents. In *Computer Graphics Forum*, volume 33, pages 141–150. Wiley Online Library, 2014.

[LMWPV15] Mathieu Le Muzic, Manuela Waldner, Julius Parulek, and Ivan Viola. Illustrative timelapse: A technique for illustrative visualization of particle-based simulations. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, pages 247–254. IEEE, 2015.

[LO08] Xiang-Jun Lu and Wilma K Olson. 3dna: a versatile, integrated software system for the analysis, rebuilding and visualization of three-dimensional nucleic-acid structures. *Nature protocols*, 3(7):1213–1227, 2008.

[LVRH07] Ove Daae Lampe, Ivan Viola, Nathalie Reuter, and Helwig Hauser. Two-level approach to efficient visualization of protein dynamics. *IEEE transactions on visualization and computer graphics*, 13(6):1616–1623, 2007.

[MSS$^+$08] Ion I Moraru, James C Schaff, Boris M Slepchenko, ML Blinov, Frank Morgan, Anuradha Lakshminarayana, Fei Gao, Ye Li, and Leslie M Loew. Virtual cell modelling and simulation software environment. *IET systems biology*, 2(5):352–362, 2008.

[oAS08] Federation of American Scientists. Immune attack, 2008.

[PB13] Julius Parulek and Andrea Brambilla. Fast blending scheme for molecular surface representation. *IEEE transactions on visualization and computer graphics*, 19(12):2653–2662, 2013.

[PJR$^+$14] Julius Parulek, Daniel Jönsson, Timo Ropinski, Stefan Bruckner, Anders Ynnerman, and Ivan Viola. Continuous levels-of-detail and visual abstraction for seamless molecular visualization. In *Computer Graphics Forum*, volume 33, pages 276–287. Wiley Online Library, 2014.

[PV12] Julius Parulek and Ivan Viola. Implicit representation of molecular surfaces. In *Visualization Symposium (PacificVis), 2012 IEEE Pacific*, pages 217–224. IEEE, 2012.

[Sch15] Schrödinger, LLC. The PyMOL molecular graphics system, version 1.8. November 2015.

[SI12] László Szécsi and Dávid Illés. Real-time metaball ray casting with fragment lists. In *Eurographics (Short Papers)*, pages 93–96, 2012.

[TCM06] Marco Tarini, Paolo Cignoni, and Claudio Montani. Ambient occlusion and edge cueing for enhancing real time molecular visualization. *IEEE transactions on visualization and computer graphics*, 12(5):1237–1244, 2006.