Eurographics Conference on Visualization (EuroVis) 2014
H. Carr, P. Rheingans, and H. Schumann
(Guest Editors)

Volume 33 (2014), Number 3

Illustrative Visualization of Metabolic Networks using Omniscient Intelligence and Passive Agents

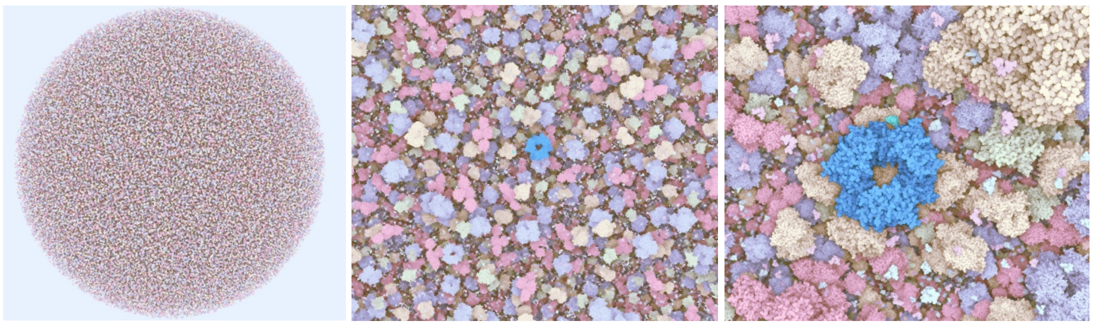

Figure 1: Demonstration of our system capabilities. A user can perform zooming in towards the scene (from left to right). The scene contains 10^6 molecules, where a given pathway, its reactions, can be followed (right). Our scene is fully dynamic, which incorporates the simulation and rendering, where we achieve 30 FPS.

Abstract

In this paper we propose a new category of a particle systems, designed for illustrative visualization of processes, in particular for visualization of metabolic networks. Previous visualizations of molecular processes were exploiting the results of agent-based modeling. Such modeling aims at reproducing accurately the stochastic nature of molecular interactions, but at the same time it is impossible to expect events of interest happening at a certain time and location. To obtain the means of controlling molecular interactions, we propose to govern passive agents with an omniscient intelligence rather than being able to initiate reactions themselves. This allows to generate intended animated stories that communicate how molecular machinery works. The rendering allows for interactive framerates of massive amounts of data, based on a new level-of-detail scheme. Finally, we report informal expert feedback we obtained from the potential users.

1 Introduction

Biology is difficult to understand without any visual explanation. For this reason biological processes, such as DNA repair, are being communicated by illustration via still images or movies. To produce scientific illustrations, content creators utilize real scientific data and 3D animation packages. They are using structural descriptions of molecular compounds to design their shape. They also have tools to ease the import, animation, and rendering of those structures such as ePMV [JAG*11] and MolecularMaya [mol13]. Although structural information is needed to explain the form, it does not convey explicit information about its function. This information has to be added by the illustrator manually through complex tasks, such as key-frame animation. Consequently, the creation process is time-consuming and expensive, taking up to months or years. Computational biology, beside structural models, also offers procedural descriptions of the function of biological processes. These are currently unused in the content creation. Illustrators could, however, greatly benefit from automated means to generate animated stories out of such procedural descriptions.

In the field of visualization, attempts have been made to visualize physiological processes, with the aim of providing a clear understanding of molecular biology phenomena. An example are biological networks, such as metabolic or signaling networks. Agent-based simulations, also known as particle-based simulations, are utilized for this purpose, because they provide spatial information of individual molecules. Such simulations are stochastic, each molecule is represented as an entity diffusing freely in space and reacting with other entities, based on the state of the environment and reaction rules. A naive way to visualize agent-based simulations, is to display each single entity on the screen and to animate them according to the positions from the simulation. The visual complexity of such generated output is usually high, due to the large number of displayed elements and the anarchical behavior of the agents.

Hence, it is a challenging task to visualize the relevant reactions from a suitable view directions in order to convey the dynamic process of the underlying metabolic network. In this paper we tackle this problem by the following contributions:

- A novel particle system approach using passive agents, controlled by omniscient intelligence
- A novel rendering technique for visualization of vast amount of molecules without any precomputation
- A comprehensive visualization of metabolic networks using real-time 3D animations

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 stored in the KEGG [KG00] database. Traditional visualization coming from the domain of biology is a graph based. 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].

However, a graph representation is omitting all the dynamics of a system, such as reaction fluxes, which are an important aspect of the biological functions. These are commonly represented by time-concentration function plots. 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, suffers from the lack of guidance and means for exploration.

Falk et al. [FKRE09] presented a tool which reads the results of an agent-based simulation and allows for 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 information about directions and reactions.

The visualization of the raw agent-based simulation using individual particles, however, suffers from a high 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. Since their rendering is based on a module that does not take advantage of GPU computing, the visualization suffers from latency.

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 of a pathway. We address this problem by providing automated means for visually communicating metabolic networks in an interactive manner.

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. In our approach, we deal with tens of thousands of dynamic molecules, which prevents interactive surface computation, i.e., computing thousands of molecular surfaces for each frame. Instead, we employ a representation that is frequently used by cellular modeling systems [FKE13], the so-called Van der Waals (VdW) representation. It represents molecules as a set of spheres, or atoms.

To speed up rendering of spheres and tubes, Tarini et al. presented a real-time algorithm for visualizing molecules by means of VdW representations [TCM06]. Their primary goal was to improve the depth perception, but they have for the first time employed billboarding for rendering static molecular systems. We also exploit VdW representations in rendering of tens of thousands of molecules at interactive frame rates. Lampe et al. [DVRH07] have introduced an even faster rendering of VdW atoms by exploiting billboards of proteins. The amino acids were generated utilizing geometry shaders. In our approach, we go one step further by exploiting also the tessellation shader to generate the entire molecule, not limited to the proteins as it was the case in the technique proposed by Lampe et al.

Many of the VdW molecular models go back 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 also employ an optimized approach using a new LOD scheme, but, importantly, we are not bound to any supporting structure.

3 Passive Agents Principle

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.

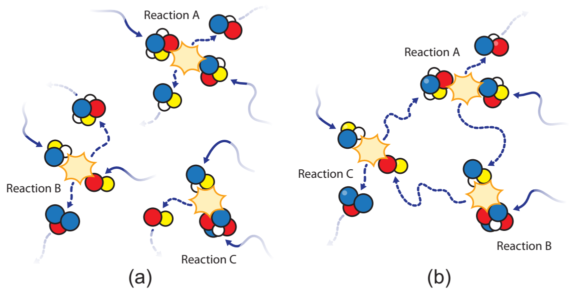

Figure 2: Comparison between a fully stochastic system (a) and our system (b). In a fully stochastic system we can observe reaction happening in the scene at random locations, and without any follow-up reactions forming a pathway. On the other hand, in our system it is possible to compose reaction sequences and visualize the reaction pathway immediately.

So far the tools and techniques developed to visualize such processes were mostly based on agent-based models, which are impractical for storytelling, because of the stochastic nature of the simulation results. In this section we describe an illustrative procedural abstraction that selectively distorts the reaction-diffusion process, to convey the function and structural characteristics of the studied metabolic system. Figure 2 shows a comparison of the reaction behavior between a fully stochastic system, and a controllable autocratic system.

In agent-based modeling, agents are computed entities evolving in time and space. They can accomplish actions according to their current state, the state of the surrounding, and various other parameters. As a consequence, each decision they make is taken by the agents' own mind, as an individual thinking entities. They do obey to certain rules, but in the case of biological networks simulation, they do not take any direct orders from an omniscient intelligence. Since we wish our agents to be controlled, we must be able to give them orders, in our case, reaction orders, to observe reactions happening in the current viewport, and to be able to build reaction sequences.

In the case of agent-based simulation of biological networks, one can not simply couple omniscient intelligence with active agents. Indeed, the results of the simulation depends on the behavior of the agents, and if a controlling entity, such as an omniscient intelligence, would disturb their behavior the results of the simulation would consequently become biased, and the resulting number of agents would be wrong.

We suggest a new approach, where agents are no longer simulation entities, but only visualization entities. Instead of using simulated agents, we employ a quantitative simulation technique, completely independent from the agent behavior and from the visualization. The agents are made passive, by removing their ability to make decisions on their own, while an omniscient intelligence module is introduced

(OI), coupled with the quantitative simulation, which dispatches reaction orders to the passive agents. We use the term passive to describe agents that can only undergo actions, according to the agent classification given by Kubera et al. [KMP10].The quantitative simulation is based on a kinetic model and provides information about the number of reactions events which must be initiated at a given time. If a reaction has to be triggered the OI simply selects an element in the scene and forces it to react. According to the type of the reaction, the neighborhood of the candidate is queried in order to find the closest reaction partners.

Reaction candidates can be chosen based on a multitude of criteria. The OI manages the selection to assure that reactions are well distributed in space, that reactions are happening in the viewport and that a selected chemical compound can be followed along the metabolic pathway. If an element, is selected by the user, it can be set to react as soon as a reaction event is delivered by the quantitative simulation to the intelligence. Once the selected element has reacted, the focus is automatically given to one of the reaction products, now awaiting in his turn for a reaction order. In case a selected element can undergo more than one reaction, the system will either assign a default reaction, or requests a decision from the user.

In addition, our system provides visualizations of quantitative characteristics of metabolic processes. One could zoom into the magnification level of a single reaction, to observe how molecules interact with each other. At this stage only few molecules are visible and the quantitative aspects are simply not observable. When zooming out from individual molecules, the more quantitative characteristics, i.e, the displayed amount of each species, become relevant. Integration of these two magnification levels allows for multi-scale visualization of metabolic processes in single compartments, and possible extension to more complex visualizations, using smooth transitions between the different compartments and magnification levels. Our proposed system features continuous transitions between two views, where the first gives a global overview of the quantitative simulation and the second one provides details-on-demand.

4 System Overview

In this section we provide an overview of our particle system. It consists of four modules, Figure 3 provides a schematic representation of the system.

The simulation module is an external module which can load and simulate descriptions of metabolic networks using the standard bioinformatics format. The results of the simulation are purely quantitative, they provide information about the initial quantities of each chemical species and the number of reaction events. The simulation is not precomputed, the system integrates each step progressively, while displaying the results in at interactive rates. This offers the possibility to modify the parameters of the simulation, such as current quantities or reaction rates, and to see the impact of the changes immediately in the visualization. The current version of our system, does not include yet the user-interface for changing the simulation parameters during the visualization.

The OI, is a centralized controlling mechanism acting as a bridge between the quantitative simulation and the passive agents. This module comprises a routine, which requests the next integration step to the simulator, and reads the amount of reaction events in order to dispatch them to the agents. For each reaction event the OI selects a candidate reactant and searches its neighborhood for the closest reaction partners. The user is able to select elements from the scene in order to force them to react. When an element is selected, the intelligence will give it the priority and trigger a reaction as soon as the intelligence gets a corresponding reaction event from the simulation.

All passive agents in the scene, undergo diffusion motion, supposed to represent the constant bounding due to molecular crowding. When a reaction order is being dispatched to an agent, its diffusion motion will slightly be deviated toward the other partners, in order to proceed to the reaction. Once the elements are in contact, the reaction is carried out and the type of the reactants change into products. The participating chemical compounds may diffuse freely again, and take part to other reactions consequently.

Finally, the information of passive-agents, are passed to the renderer in order to compute the current frame. We employ a custom rendering pipeline that allows to display large amounts of structures at interactive framerates. We represent the structures using the VdW representation via the glyph-impostors technique. We also employ rendering techniques, such as ambient occlusion, in order to enhance the depth cues and the visual quality of the output.

Further details about the different steps of our particle system are delineated in the following section below. We describe the computation of the passive agent positions and the rendering in two dedicated sections.

5 Particle Computation

In order to compute the positions of the passive agents, prior to the rendering, the system executes several steps which we describe in this section. Let us assume that the passive agents are defined as a set $A = \{A_0, A_1, \ldots, A_n\}$. Please note that the number of agent, n, changes over the simulation time. Each agent A_i is composed of the following properties: $A_i = \{Type, Position, Orientation, LinearVelocity, AngularVelocity, ReactionType, ReactionID\}$.

The $Type$ property describes the chemical compound, such as for example NAD or ATP. The $Position$, $Orientation$, $Linear$ and $AngularVelocity$ are all physics related properties, and the $ReactionID$ and $Type$, are properties which give unique ID to every reaction and describe

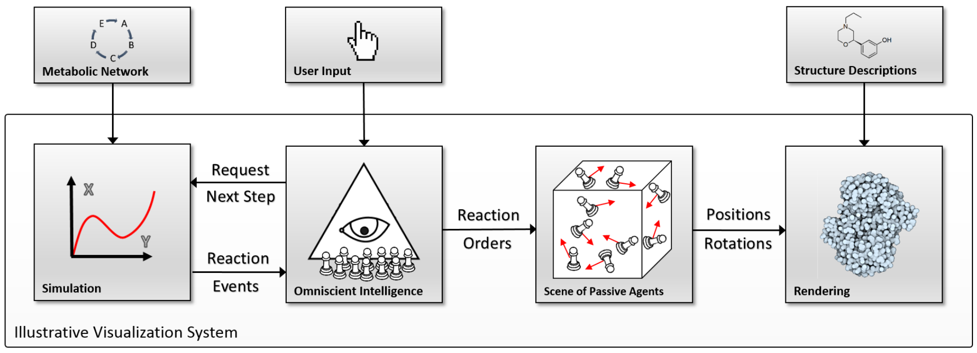

Figure 3: Overview of our system: A biological process is represented by a kinetic model, which is numerically simulated. The Omniscient intelligence module fetches the amount of reactions and based on particular rules it selects which particles undergo a reaction. The final stage of the pipeline represents elements with their structural models and renders the entire scene at interactive framerates.

which reactants, products, and enzymes participate in the reaction.

The agents are updated by two different routines, the OI routine and the agent routine. The OI routine is responsible for requesting the next integration step from the simulation module and for dispatching reaction events to the agents. The agent routine, on the other hand, is responsible for the agents behavior such as the motion. The agent routine is coupled with the display routine, since it is strongly dependent on the motion of the agents. The OI routine consists of dispatching punctual events and it does not have to be called as frequently as the agent routines. Here only a few calls per second are sufficient to guarantee a visualization of reactions without causing any noticeable delay.

5.1 Omniscient Intelligence Routine

The omniscient intelligence acts as a bridge between the simulation and the agents. The simulation, based on the description of the metabolic network, defines the initial quantities, the type of the passive agents and with the reactions that they undergo.

The reactions represent a key ingredient in our modeling system. We define the following reaction types:

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

where the zeroth order reaction (1) represents the input of an 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.

For each simulation step, we obtain a set of reactions $R = \{r_1, \dots, r_l\}$, where the r_i type is always one of the 9 reactions. Essentially, the set R transforms the current concentration values represented by the amount of agent types, into new concentrations specified by new agent types obtained by execution of the reaction R. The reactions are obtained by the simulation results. In the case of enzymatic reactions, we also include the catalyzer as an actor of the reactions. In order to integrate the enzymes into the scene, we adjust the number of enzymes to ensure that enzymatic reactions can be initiated from any location at any given time.

At the beginning of the simulation, prior to the first routine call, OI module populates the scene with agents 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 reactions.

For a given reaction event, like for instance reaction (4) A + B → C, the OI module selects a candidate, A, from the scene. Then, the scene is searched for the closest reaction partner, B. When dealing with large number of particles it is important to provide means for an efficient lookup into the scene. We describe implementation details about the partner search in Section 5.3. Once the partner search is done, the OI module will assign a reaction ID to the reaction participants. This ID, initially set to NULL, will change the state of the agents and force them to start the reaction animation.

Metabolites, i.e., the elements of a metabolic network, are often organized in a cascade of reactions in a pathway. A pathway is a network, defining connections of reactions. A pathway P can be described as a chain of reactions; $P = \{(r_i \to r_k), (r_k \to r_m), (r_m \to r_q), \dots\}$. As such, any given pathway can be easily converted to our reaction model. Our framework uses Copasi [HSG*06] as the simulation engine. Copasi can read SBML [HFS*03] files, which means that models described in this format can be integrated into our framework.

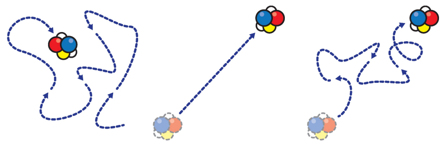

Figure 4: Description of the reaction motion, we blend a random walk motion (left), with linear interpolation (middle) in order to get a consistent attraction motion (right).

5.2 Agent Routine

An update of the agent routine affects the motion of the agent. There are two types of motions, which can influence the trajectory of a given particle; the diffusion and the reaction motion. We apply the motion to the agents using physical forces. This is based on the rigid-body physics simulation introduced by Baraff et al. [Bar01]. Each particle carries attributes, such as position, rotation, linear and angular velocity, and the current position and rotation are integrated using the Euler method.

In order to mimic the mechanics of Brownian motion, which results of a constant bouncing from the crowd of surrounded molecules, we apply impulse forces in a random direction to the particles at given time intervals. These intervals, as well as the magnitude of the force can be interactively specified by the user to match with a diffusion speed which is slow enough to be trackable by human eye. It is important to mention that our aim is not to reproduce the accurate Brownian motion mechanic, but rather to illustrate its effect.

The reaction motion consists of attraction forces represented as 3D vectors that point towards the direction of the reaction partner. The attraction forces are added to the diffusion forces. This accumulation of forces prevents linearization of trajectories, i.e., particles following straight lines, resulting in *unnatural* motion (Fig. 4). Once all the partners of a reaction are in contact, the reaction is over and the reactant types are either changed into new ones (products), or removed entirely from the scene. Subsequently, each remaining participant may again diffuse freely within the scene, or can take part on another succeeding reaction in the pathway.

In case of enzymatic reactions, we do not involve attraction forces to define the motion trajectory of enzymes, due to the very different sizes of large enzymes in comparison to small metabolites. The enzymes diffuse freely while in space waiting for its reaction partner.

5.3 Implementation

In order to provide fast computation of large number of elements, we exploit the computational capabilities of modern GPUs, using CUDA. All the data concerning passive agents is stored in and computed on the GPU memory, and can be transferred to the rendering stage without any performance costs. We allocate a large buffer on the GPU memory, in which we store all the individual properties of the computed entities. The properties are stored in an Array-of-Structure (AOS) manner, in order to take advantage of the GPU computation paradigms. The use of a large buffer also allows us to use fast GPU operations such as sorting, or spatial subdivision.

When dispatching a reaction order, the OI module selects firstly a candidate according to its type. We sort the agents (particles) within the agent buffer according to their type. This will allow a fast and efficient access to elements of a given type, i.e., without having the need to browse the entire buffer. We use a buffer management library for this purpose which allows us to sort large buffer on the GPU extremely efficiently with radix sort [BH11]. When attracting reaction partners together, we sort the agent buffer according to the reaction ID, which groups all the reacting agents of the same reaction together and allows for fast memory access.

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]. The technique consists of computing a uniform 3D grid of the scene, where each gird cell contains a list of elements located inside. The search is done by browsing the surrounding cells for the closest partner element. The grid resolution is chosen according to the density of displayed elements. In the case a partner search fails, we enlarge the searching area in order to browse more surrounding voxels and improve our chances to successfully find a reaction partner.

For generating the random numbers needed for the Brownian motion, the cuRAND library is used.

5.4 Particle System Performance

We provide a performance analysis of the computation of the particles, i.e., the positions and rotations. We demonstrate the interactive performance of our system using the two different networks models, each one showing different characteristics such as quantities and reaction rates.

The first model we employ is a simplified version of the NAD pathway (Fig. 5). 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 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 TCA cycle also called glyoxylate cycle. The glyoxylate cycle is used by plants, bacteria and fungi to produce carbohydrates. Parts of the cycle overlaps with the tricarboxylic acid (TCA) cycle. The split between

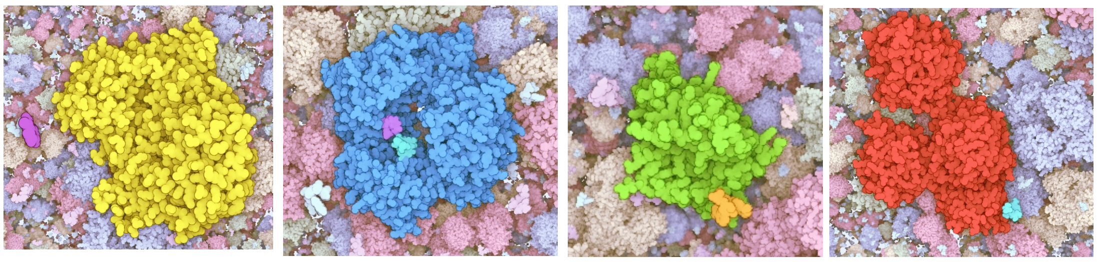

Figure 5: Snapshots from the NAD pathway. The examples show four extracted frames from four reactions involved in NAD pathway. We see four enzymes (yellow — NAMPT_ATP,blue — NMNAT1_NMN,green — NNMT,red — PARP1) that catalyze these reactions. The camera follows metabolites that take part in reactions 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.

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. The TCA cycle allows cells to obtain energy from fat.

Since the OI routine and the agent routine are not coupled together, we provide an average computation time, accounting for both routines, for a given frame (see Table 1). The performance were measured on an Intel Core i7-3930 CPU 3.20 GHz coupled with a GeForce GTX Titan GPU.

6 Particle Rendering

Each molecule is visualized by means of the VdW representation, where atoms are defined as spheres of given radii. Instead of having tessellated spheres, we use billboards to render them 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 the way to render millions of dynamic atoms, $O(10^5)$, molecules involving $O(10^3)$ atoms at interactive frame rates. One potential solution is to upload all the atoms per frame to the GPU, which would however 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 molecule by just rendering a single point, i.e., the center of the molecule. This point is accompanied with the rotational quaternion representing the current orientation of the molecule. In the next stage, we exploit tessellation and geometry shaders to invoke the rendering of all the molecular atoms. 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 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. Thus, as an input for the vertex shader, we use the molecular ID, i.e., the address with the texture, position, obtained from the simulation, and rotational quaternion. In the tessellation control shader, we exploit isolines as patch primitives. This requires to specify two outer tessellation levels that both equal to $\sqrt{\#atoms}$. Thus each atom is defined by a combination tessellation variables, `gl_TessCoord.x` and `gl_TessCoord.y`, obtained in the tessellation evaluation shader. Based on this id, the necessary atom is fetched from the texture buffer holding the atom coordinates and the radius.

Afterwards, each atom is processed via the geometry shader to generate a billboard. In case when $\#atoms$ exceeds $4K$ atoms, we setup $\#atoms = 4K$ 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 also apply view frustum culling in the vertex shader to avoid rendering of molecules outside the viewing frustum.

Agents	NAD cycle [ms]	TCA 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.

6.1 Level of Detail

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 occupied with molecules, which span over only few image pixels. This is the case of molecules that are far in the back of 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.

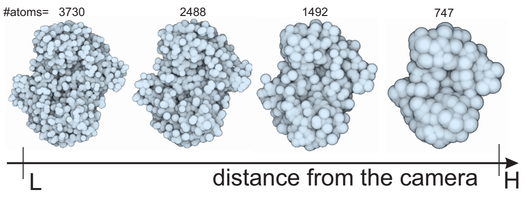

Figure 6: An example of our molecular Level-of-Detail. With increasing distance to the camera, more atoms are skipped. The radii of the remaining atoms are scaled accordingly.

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. 5). On the other hand, the context molecules are displayed via pale colors.

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 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 $\times$ 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.

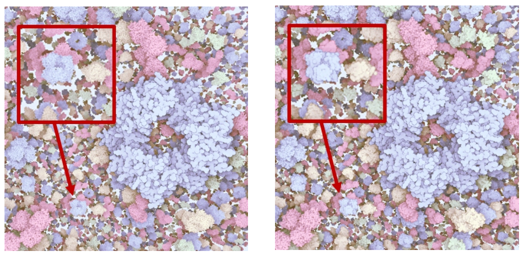

Figure 7: An example of LOD application. The same scene is rendered without (left, 43 948 997 atoms) and with LOD (right, 7 227 817 atoms). Notice that for the molecules that are further in the back (the closeups), only negligible differences are visible.

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. The subtitles in the video sequence, where the chemical reaction is described, contributed to the comprehension of the biological process depicted in 3D. Still, experts would appreciate more advanced means for visual guidance as the processes are happening in a fast pace. A legend depicting the most relevant molecules has been suggested to further improve the comprehension. Furthermore, advanced methods for selective visualization have been requested, to easily determine the main actors from the context. As the strongest criticism they mentioned the *anticlimactic* reaction in the shown video. As storytellers they would like to see a build-up of a story into a punch line.

8 Conclusions and Future Work

We have developed a novel concept of an autocratic particle system, tailored for creation of interactive molecular illustrations. While in the agent-based particle systems the autonomous agents do not allow for specifying the visualization intent, our method allows for directing the behavior through the newly introduced omniscient intelligence. The physically-plausible behavior of molecular interactions based on diffusion is thus distorted and traded with the ability to control the process. From a perspective of illustrative visualization techniques, we can see our contribution as an abstraction technique; 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.

We have demonstrated the utility of the new particle system on the domain of computational biology. Based on the simple idea of representing the simulation quantitatively with an particle-based visualization, we have proposed a new way to represent metabolic processes. Our particle system integrates scientific data from structural biology and systems biology. This might potentially have impact on how to approach the integration challenge of individual models developed in reductionistic scientific disciplines into larger integrated models by means of visualization. Our particle system is in some degree generic and does not apply only for the domain of molecular biology. 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, migrations patterns, or crowd simulations.

Current means that guide the viewer are based on the simple approach of a camera following the actor. To provide deeper understanding how molecular processes work, advanced visual guidance needs to be developed, which modifies temporal aspects as well. While diffusion of quantities within larger area can be in interactive, to convey a docking detail, one would need to slow down the time by several orders of magnitude. The visual guidance would certainly benefit from tailored visualization designs that convey structural or quantitative details. Finally we would like to improve the visual aspect of the visualization by adding atom-based collision responses between all the participating elements in the scene.

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. 6

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

[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, 7

[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. 8

[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. 4

[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. 7

[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

[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

[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, 7