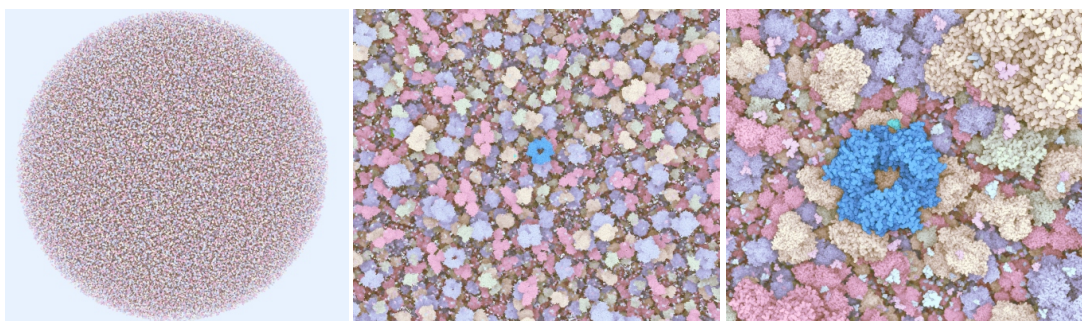


# Illustrative Visualization of Molecular Reactions using Omniscient Intelligence and Passive Agents

M. Le Muzic<sup>1</sup>, J. Parulek<sup>2</sup>, A.K. Stavrum<sup>2</sup>, and I. Viola<sup>1,2</sup>

<sup>1</sup>Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

<sup>2</sup>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

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 billboard 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  $25 \times 10^9$  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.

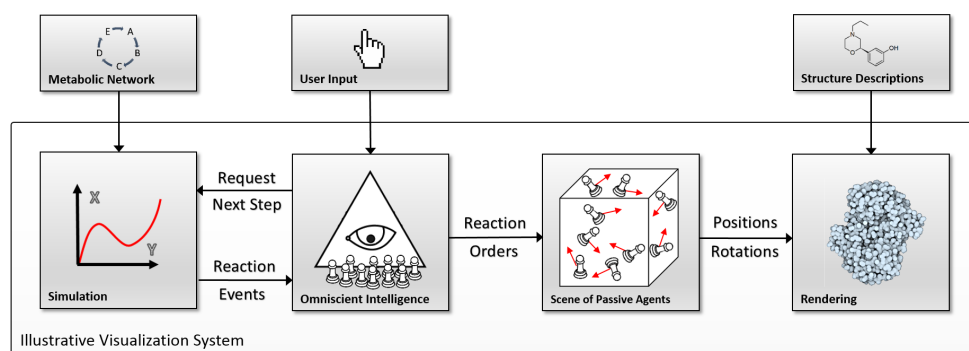
Our aim is to provide an environment for creating automated illustrations, yet there are also studies in the literature, which address more scientifically oriented schemes for solving and visualizing immersive molecular simulations. A powerful framework for visualizing large-scale atomistic simulations 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-



**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.

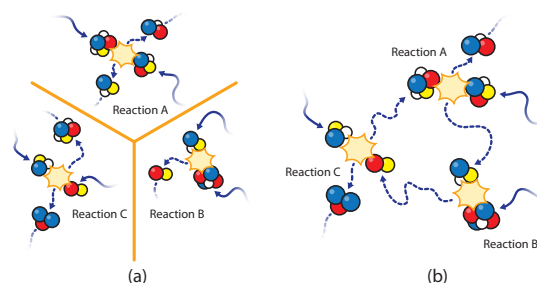
parison between a fully stochastic system, and a controllable system.

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 represent 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. Addi-



**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.

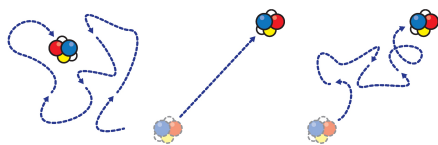
tionally, we highlight the reacting elements by increasing the color intensity and also display text-based information about the different actors of ongoing reactions in the focus.

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

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.

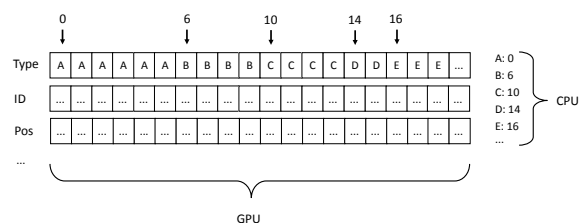
Each passive agent is composed of the following properties: *ID*, *Type*, *Position*, *Orientation*, *LinearVelocity*, *AngularVelocity*, *ReactionID*. The *ID*, is a unique ID given to every single molecule, the *Type* describes the chemical compound, such as *ATP* for example. The *Position*, *Orientation*, *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



**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 from 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 primi-

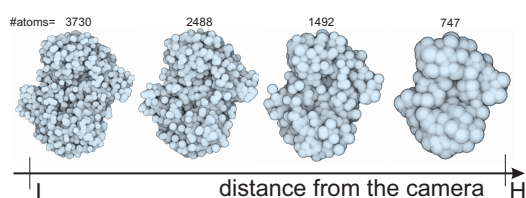
tives. This requires to specify two outer tessellation levels by  $gl\_TessLevelOuter[0] = gl\_TessLevelOuter[1] = \sqrt{\#atoms}$ . Then in the tessellation evaluation shader each atom  $id$  is defined by a combination tessellation variables as follows:

$$id = gl\_TessCoord.x \times gl\_TessLevelOuter[0] + \\ + gl\_TessCoord.y \times gl\_TessLevelOuter[0] \times \\ \times gl\_TessLevelOuter[1].$$

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

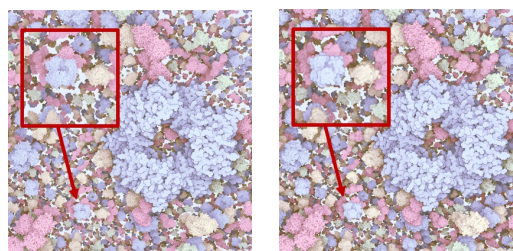
we skip more and more atoms, i.e., rendering only each  $n$ -th atom. As a result of this procedure, we remove always the same atoms from the molecule at a certain distance, which keeps a scene view more persistent than when performing removal, for instance, stochastically. Still this can be performed 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.

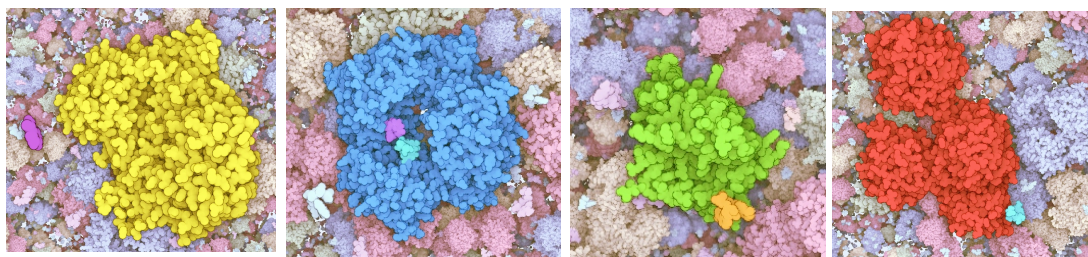
## 6 Results

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  $\alpha$ -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 mea-

sured 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 ( $3 \times 10^{10}$ ) 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. 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., KOEPL 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] GOODSSELL D.: *Illustrating Molecules*. 2003, ch. 15, pp. 267–270. 3
- [Goo09] GOODSSELL 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., GOODSSELL 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](http://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