ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

CellPathway: a Simulation Tool for Illustrative Visualization of Biochemical Networks

Matthias Reisacher
TU Wien, Austria
1125358@student.tuwien.ac.at

Ivan Viola
TU Wien, Austria
viola@cg.tuwien.ac.at

Mathieu Le Muzic
TU Wien, Austria
mathieu@cg.tuwien.ac.at

ABSTRACT

The molecular knowledge about complex biochemical reaction networks in biotechnology is crucial and has received a lot of attention lately. As a consequence, multiple visualization programs have been already developed to illustrate the anatomy of a cell. However, since a real cell performs millions of reactions every second to sustain live, it is necessary to move from anatomical to physiological illustrations to communicate knowledge about the behavior of a cell more accurately. In this publication we propose a reaction system including a collision detection algorithm, which is able to work at the level of single atoms, to enable simulation of molecular interactions. To visually explain molecular activities during the simulation process, a real-time glow effect in combination with a clipping object have been implemented. Since intracellular processes are performed with a set of chemical transformations, a hierarchical structure is used to illustrate the impact of one reaction on the entire simulation. The CellPathway system integrates acceleration techniques to render large datasets containing millions of atoms in real-time, while the reaction system is processed directly on the GPU to enable simulation with more than 1000 molecules. Furthermore, a graphical user interface has been implemented to allow the user to control parameters during simulation interactively.

Keywords: Molecular simulation, visualization system, collision detection, particle-based data, large data

1 INTRODUCTION

The usage of illustrative tools is an established approach to communicate knowledge of complex biochemical processes in cells to a broad audience. In the beginning, illustration artists had to create time consuming handmade animations combined with sophisticated visualization techniques to tell a structured story. The next step was to use software tools to create images showing complex molecular structures. While at the beginning, render processes took hours or days to complete, with increasing processing power it was possible to explore large scenes containing millions of atoms in real-time. However, millions of chemical reactions are performed every second in real cells to allow intercellular communication and to sustain living organisms. To communicate knowledge about complex intracellular processes which keep the cell alive, it is necessary to move from anatomical to physiological illustrations. Therefore, the next logical step is to use molecular reaction systems to simulate large scale reaction networks which are describing the physiology of a cell.

While this area has received a lot of attention lately, many tools to simulate and visualize molecules and reactions inside of a cell have been proposed in the last few years. Lately, particle-based simulators got more popular to imitate a realistic behavior of the molecules. Their general approach is to postpone most of the calculations and operations from the central processing unit to the graphics card through a GPU first approach by, for example, enabling GPU-to-GPU data flow. This is possible due to the modern, freely programmable GPUs. General-purpose GPU programming accelerates the performance of those systems immensely. That enables the simulation and visualization of large-scale scenes containing billions of atoms on an average computer. However, most of those approaches do not take global collision detection into account. This improves performance but leads to visible artifacts during the animation.

My goal is to extend a modern particle-based illustration tool with a basic molecular simulator and a collision detection system. Additionally, a visualization system to improve the user's awareness of biochemical processes and to display reaction networks inside of a specified area is implemented. Besides, the user should be able to interact with the system to optimize the learning effect. To implement those goals, more calculations per frame have to be executed, which has a significant impact on the performance. Especially the collision detection needs multiple processing steps. For load balancing, the simulation is only executed in a specific part of the scene, whereby the user can determine the size and the position of this area. Therefore, the program's requirements can be scaled down manually by the user to enable the simulation also on weaker computers.

WSCG 2016 conference proceedings
WSCG'2016, May 30 – June 3, 2016
Plzen, Czech Republic.

The system is based on the technique proposed by Le Muzic et al. [MAPV15]. While almost every aspect of the core visualization techniques are inherited, a simpler reaction system has been implemented. The quantitative simulation itself is calculated by the COPASI [HSG+06] API and the reaction system is working with an omniscient intelligence while using passive agents for dynamic simulation given by Kubera et al. [KMP10]. To enable a fast and easy change of the simulation system, the user is provided with a simple UI, whose implementation is inspired by the publication of Daniel Gehrer [D14].

Current techniques in mesoscale visualization of biochemical processes are including collision detection only partially or they are ignoring it at all for the sake of performance. Tools like ZigCell3D [CKMK13] or MegaMol [GKMRE14] are great for visualization but because the molecular participants do not collide and therefor don't interact with each other beside during a reaction, they are not able to showcase realistic animation of molecular crowding. Furthermore, visible artifacts occur. The main contribution of this work is to implement a three dimensional collision detection system which is able to detect the intersections of two or more objects at the level of single atoms. Additionally, an illustration technique using two adjustable cone-cut-objects in combination with a real-time glow effect is implemented to make complex processes visible even in dense scenes but without losing the impression of depth. Further, a hierarchical structure is used to illustrate intracellular process, by showing the impact one reaction has on the entire simulation. Since this project is based on the work proposed by Le Muzic et al., it also uses the Unity3D [WSUnity] engine. Unity is a cross-platform game engine which is also available for free in a limited, but still functional, scope. The provided user interface is also implemented in Unity. Simulated molecules can be downloaded from the public PDB database [WSPdb] and afterwards imported through the user interface.

2 RELATED WORK

We structure the prior work review in two parts. Firstly, we refer to agent-based simulation systems using a game-based environment. In the second part we relate with research techniques that employ with Monte Carlo-based simulation systems.

2.1 Agent-based Simulations in Game-like Environments

Using game-like environments to reduces the software development work-load is getting more and more popular among the visualization community [MAPV15]. A recent work on visualizing the anatomy of a cell in a multiscale approach has been proposed by Muzic et al. [MAPV15]. cellView has been implemented in the Unity3D [WSUnity] game engine and uses macromolecular datasets, which are modeled with the cellPACK [JAAGS15] tool. cellPACK is publicly available and enables the generation of large biomolecular structures. By using advanced GPU programming and acceleration techniques, such as hierarchical Z-buffer occlusion culling and a twofold level-of-detail approach, it is possible to render scenes containing billions of atoms. But since no molecular dynamics are supported, cellView can not be used to illustrate the physiology of a cell.

A tool to visualize agent-based simulations has been proposed by de Heras Ciechomski et al. [CKMK13]. An accurate simulation system is implemented in an interactive and game-like 3D environment to illustrate complex chemical processes at various zoom levels. Cellular reactions are modeled with a GUI and are represented as an SBGN [WSSbgn] network diagram. However, the rendering module does not use GPU programming and therefore, no real-time processing of large scenes is possible.

To enable real-time rendering of billions of molecules, the necessary calculations have to be done in parallel using advanced GPU programming. Such visual explanation tools have been proposed by Le Muzic et al. [MPSV14] and with CellUnity[D14]. A particle-based simulation system is used in combination with passive agents and an omniscient intelligence to simulated biochemical reactions in a story telling manner. Nevertheless, only a limited collision detection system has been implemented and the tool is not publicly available. Another simulation system using this approach is CellUnity [D14] It has been implemented in Unity3D and uses its physics engine to apply a fully collision detection during simulation. However, since the project is implemented entirely on the CPU, only small scenes containing a few hundred molecules can be used for simulation.

2.2 Monte Carlo based Simulation Systems

By using specialized Monte Carlo algorithms, MCell [SB01] is a popular tool to simulate chemical reactions in multiple compartments. The visualization tool CellBlender [WSCB], which is an addon for the open-source 3D computer graphics software Blender [WSB], enables a simple and fast way to model and edit the molecule designs of a simulation. Although, the simulation settings can be changed directly in Blender, no interactive storytelling approach can be used to present the outcomes.

Illustrative timelapse is a cross-platform simulation tool with the focus on multi-scale temporal illustrative visualization techniques. MCell is used to model and simulate biochemical processes while the visualization

part is implemented in Unity3D using the technique proposed by Le Muzic et al. [MPSV14] A combination of interactive temporal zooming and visual abstraction is used to communicate reaction processes in a storytelling manner. However, no collision detection has been implemented.

3 SIMULATION

In this section we describe the reaction system and the simulation algorithm, as well as implemented techniques for load balancing.

3.1 Global Simulation

Our molecular reaction system is based on the technique proposed by Le Muzic et al. [MPSV14]. While the individual molecules who are participating in the simulation process are implemented as passive agents, an omniscient intelligence (OI) is used to control molecular interactions. Passive agents are unable to start reactions autonomously, instead they can only receive reaction orders from an OI, which is tightly coupled with the quantitative simulation [MPSV14]. The system uses the COPASI API [HSG+06] as simulation engine, which is responsible to initiate new reactions in regular time intervals, whereby the interval length is specified by the user. Since each reaction has to be processed separately on the GPU, the number of created reactions per frame is limited to 20 to increase the performance. For each initiated reaction the reaction system searches for appropriate reactants, which are not already included in an open reaction. While the first reactant is picked randomly, the other molecules of the specific reaction are selected by their distance to the first molecule. Only the molecules closest to the first reactant are assigned to the reaction. Additionally, every reaction type can be linked to protein by the user. Thus, the reactants need to enter a random protein to perform the specific reaction.

To create the impression of chaotic behavior, the molecular movement during a reaction is created by interpolating direct motion with Brownian motion. This prevents linear pathways and enables the simulation of molecular trajectories more realistically. The molecule's position, rotation and the calculated movement vector are passed to the collision detection algorithm, to find a collision with a protein. Since proteins are much larger than the reactants, a collision has no influence on their movement. On the other hand, the movement vector of a reactant is shortened in case of a collision. If a reactant needs to enter a specific protein to perform a reaction, collision between the protein and the reactant is ignored.

Reactions are processed when all included molecules are colliding. When a reaction is executed, the reactants are deleted and the created products are placed at the location of the reaction. To minimize the number of molecules stored in a compute buffer, the buffer positions of deleted molecules are saved. This way, whenever a product is created, deleted molecules can be overwritten.

For load balancing, not all molecules are included in the reaction system. Instead, the simulation area is reduced to a spherical compartment, which is placed by the user at the location of an arbitrary protein. Only reactants inside of the compartment are included in the simulation process, while the others are moved by Brownian motion.

3.2 Compartment Simulation

Since the reaction system and the collision detection algorithm are processed in real time, three techniques are used inside of the simulation compartment to reduce the number of overall calculations during simulation. By using spatial subdivision, counting sort and the fast fixed-radius nearest neighbor algorithm, the spatial position of individual objects can be included during the processing of molecular interactions.

Spatial subdivision is an approach where objects in a three dimensional space are ordered by their position. The space is partitioned in a uniform grid, such that a cell is at least as large as the largest object [Nygu07]. The objects are sorted with their corresponding cell ID by using the counting sort algorithm. Additionally, an array called Bin-Counter is used to keep track of the number of objects inside of every single cell. The Bin-Counter in combination with the list of ordered objects allows to identify all objects contained in a specific cell by the cell index.

Fast fixed-radius nearest neighbors is an algorithm to find all objects inside of a sphere with a radius R. When the sphere is centered at the position of a specific object and the radius corresponds to the length of the movement vector, the algorithm can be used to find all relevant objects during collision detection. Since the time complexity of a brute force attempt to find all neighbors of all objects is $O = (n^2)$, the spatial partitioning method is applied first. To minimize the number of cells who are overlapping with the sphere without having too many objects per cell, an additional requirement is established, which states that the minimum cell size during spacial partitioning has to be at least as large as the radius R. This way, only objects in neighboring cells have to be searched, which reduces the average complexity to $O(n * log(n))$ [WSNT]. Those objects can easily be found by combining the spatially sorted objects and the Bin-Counter values.

4 VISUALIZATION

While the technique to represent molecules in a level-of-detail manner proposed by Le Muzic et al.[MPSV14] is inherited, three additional visualization techniques, called real-time glow, cone clipping

and reaction tree, are implemented to communicate knowledge of complex biochemical processes in cells. Since the reactions are distributed throughout the compartment and can occur simultaneously, it is difficult for the user to realize when and where a reaction is completed. Therefore, every created product and proteins included in a completed reaction are highlighted with a real-time glow effect for approximately one second. The glowing objects are copied in a separate texture with a compute shader and blurred by using a two-step operation called a *separable convolution* [Fer04]. This way, the two-dimensional convolution kernel is divided into two separate one-dimensional convolutions, one in each axis, which greatly reduces the computation costs [Fer04]. To increase the brightness of the glow effect, a non-uniform convolution kernel is used.

When illustrating dense scenes with millions of atoms placed near each other, reaction processes are easily covered by larger protein structures. With the cone clipping method, the user is able to remove disturbing protein structures, which are located between the camera and the center of the simulation compartment, to get a better view of the ongoing reactions and molecular interactions during simulation. To allow the user to change the amount of clipped objects, the cone angle can be set interactively to a value between 1 and 89 degrees.
While increasing the visibility of simulation participants is the main goal of this visualization technique, the three dimensional spatial depth impression should be retained. Additionally, a semi-transparent area is used to create a continuous transition between the clipped area inside of the cone and the shown objects outside of it. Thus, a second cone is implemented and placed at the same position as the clipping cone, whereby the angle of the second cone is twice as large as the angle of the clipping cone. Objects located between the inner and outer cone are represented partially transparent. To create a continuous transition between the clipped area and the opaque objects, the amount of transparency for a specific object depends on the location. While molecules located next to opaque objects are rendered with less transparency, the value increases when located closer to the clipping cone.

In this project, a hierarchical structure is used to illustrate biological networks created by complex interactions between different molecules and proteins in a particular time span. Starting with only one reaction, a dynamic tree structure is build by illustrating the path of the reaction products with lines. When those molecules are included in another reaction, a node at the location of the reaction is included in the structure, while the outgoing branches are connected with the new products. Therefore, the tree structure is growing over time, showing the influence of the starting reaction on the whole simulation. The tree structure is created during post-rendering by using a geometry shader. This enables to visualize dynamic lines for reactants and products of ongoing reaction and the static structure created by already processed reactions.

1: **Input**: $diff$, *rm* and *sumRadii*.
2: **Output**: The new vector length or -1.
3: float $dist = length(diff)$
4: float $cd = dot(normalize(rm), diff)$
5: **if** $cd <= 0$ **then**
6: **return** -1
7: float $f = dist * dist - cd * cd$
8: float $sumRadiiSquared = sumRadii * sumRadii$
9: **if** $f >= sumRadiiSquared$ **then**
10: **return** -1
11: float $t = sumRadiiSquared - f$
12: float $newMovementLength = cd - sqrt(t)$
13: **return** $newMovementLength$

Algorithm 1: Collision Detection

5 IMPLEMENTATION

In this section, we are going to discuss implementation details about the used data structure and the collision detection algorithm.

5.1 Data Structure

Data objects are stored in a two-parted data structure to enable efficient data access and to minimize the needed amount of storage. Passive data contains all information needed to render molecules and is uploaded to the GPU when a scene is loaded and when the reactants are placed. Structural details of the molecular types, such as the atom count, the position of single atoms and the color are stored as passive data, as well as the position and rotation of every single molecule. Active data on the other hand stores the following structural information needed for the simulation process:

- *Reactant*: Is created for every reactant participating the simulation process and contains a reference to the passive data, a specific reaction, the movement vector and the cell id of the compartment.
- *Reaction*: Is created for every initiated reaction and contains a reference to all reactants, the calculated reaction position, which products are created and if the reaction is part of the reaction tree.
- *Collision*: A reference to the colliding objects and their movement vectors.
- *Reaction Tree-Line*: The start and end point and the molecular color.

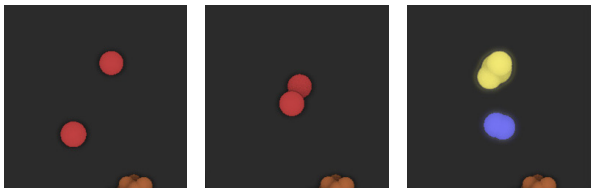

Figure 1: Snapshots from the processed reaction $H + H \rightarrow H_2 + H_2O$. (a) The two reactants approach each other. (b) The objects are colliding and the reaction is processed. (c) The reactants have been removed and the products are placed.

5.2 Collision Detection

Our algorithm is structured in a two-phase approach: a broad phase followed by a narrow phase [Nygu07]. During the broad phase, collision between molecules is determined by using minimal bounding spheres. Those tests are fast and cheap to calculate. However, since approximated bounding volumes are too imprecise for more complex shapes, errors occur in the collision tests. Therefore, potentially colliding molecules must be further processed in the narrow phase. By testing collision between two objects precisely, the single atoms of both objects have to be tested against each other. Since both phases are comparing spherical objects the same test algorithm, shown in Algorithm 1, can be used.

The algorithm calculates if two objects will collide during the next movement step and shortens the movement vector when necessary, so that both objects come to rest just at the point of contact. As input, the relative movement vector $rm = movement1 - movement2$, the spatial distance *diff* and the sum of the radii *sumRadii* are used. The radius refers to the minimum bounding sphere in the broad phase and to the atom radii during the narrow phase. The closest distance *cd* between the objects during the movement is calculated in line 4. For $cd <= 0$, the objects are not moving towards each other and no collision can occur. When the squared length of the closest distance vector is larger than the square of the summed radii compared in line 9, the objects are not colliding during their movement. At last, the Pythagorean theorem is used to shorten the movement vector in line 12 and the results are returned. After the narrow phase, the shortest calculated movement length of all pair-wise atom tests is used for the next movement step.

6 RESULTS

In this section we present results of the proposed algorithm regarding the quality of the created images and the performance of the simulation system. The simulation system is tested with a dataset created with the cellPACK [JAAGS15] modeling tool, showing a human blood serum surrounding HIV virus. Additionally, 50000 reactants of five different molecular types have been placed throughout the scene. By using a compartment radius of 300 nm, about 350 reactants have been included in the reaction system, processing four reaction types.

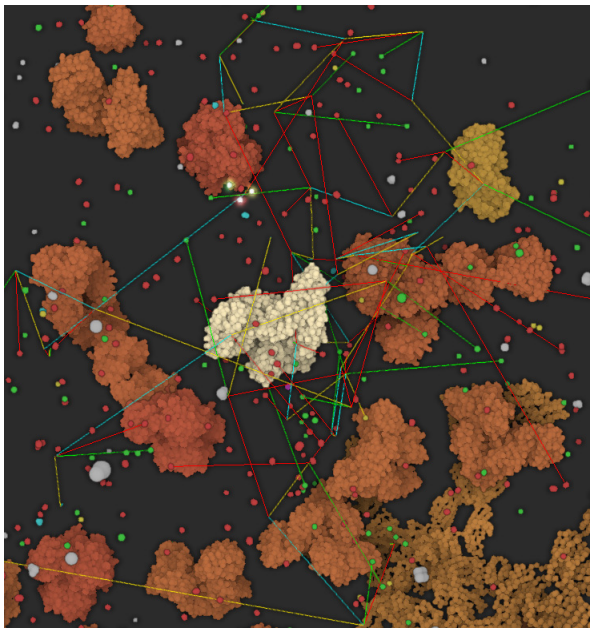

Figure 2: Snapshots from a reaction tree structure while simulating four different reaction types with approximately 350 reactants.

Figure 1 shows snapshots from the reaction process of the type $H + H \rightarrow H_2 + H_2O$. In Figure 1(a) the two reactants of type A, which are included in the reaction, are moving to the calculated reaction location. The collision of the molecules is shown in Figure 1(b). Since reactions are triggered by contact, the reaction system removes both reactants and creates the products B and C. Figure 1(c) shows the placement of the newly created products. To avoid overlapping molecules, the collision detection algorithm is used to find free areas around the reaction location.

A screenshot from the reaction tree visualization technique where the reactants are processed by four reaction types is shown in Figure 2. After initiating the starting reaction it took approximately 20 seconds of the simulation time until the tree structure has reached the illustrated size. It can be seen that created products are included in further reactions, which are initiated afterwards. By using the color of a molecular type for single lines it is possible for the user to determine which reaction has been processed at a specific location and which products have been included.

An example of the real-time glow effect is given in Figure 3. The camera is positioned in such a way that the entire compartment is shown. By highlighting the created products, the user is able to determine when and where a reaction is processed during the simulation. In Figure 3(b), a close-up of individual glowing molecules

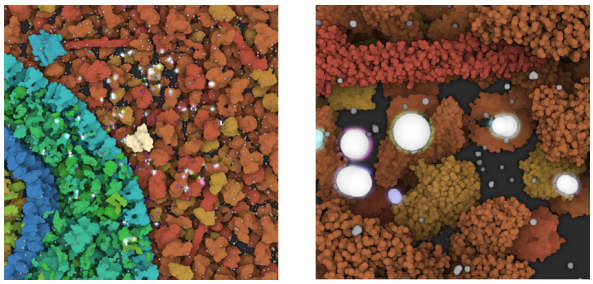

Figure 3: Snapshots from the processed reaction $A + A \rightarrow B + C$. (a) The two reactants approach each other. (b) The objects are colliding and the reaction is processed. (c) The reactants have been removed and the products are placed.

is shown. By comparing the glow radius in Figure 3(a) and (b), it can be seen that the size of the glowing effect depends on the distance of the camera to the specific object. When the camera is close to the highlighted product, the size of the glow effect around the molecule is reduced to avoid superposition of molecular structures.

A screenshot of the cone clipping effect with a 15° angle is given in Figure 4. For the simplified transparency effect, the color of a translucent object is combined with the color of concealed opaque objects or the background color, but not with other transparent objects. Although, excluding translucent objects during the alpha blending process increases the rendering performance, artifacts are created when molecules are located at the edge of the scene. An example of those artifacts is shown in the top right corner of Figure 4. When no opaque objects are located behind a transparent protein, the object color is mixed with the background color black. This leads to a wrong perception of depth, because transparent objects which are located closer to the clipping cone are shown darker than the objects behind it, which are positioned further away.

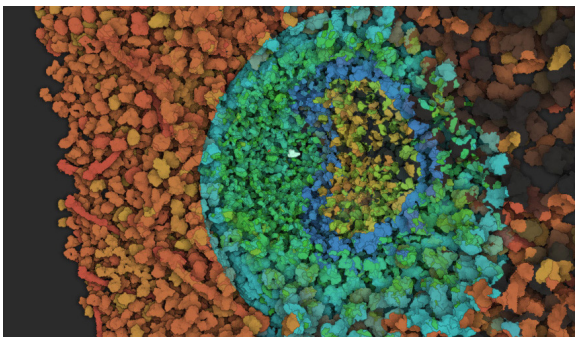

Figure 4: Snapshots from a reaction tree structure while simulating four different reaction types with approximately 350 reactants.

Performance Analysis

Since the implemented visualization techniques require a processing time of approximately 1*ms*, only the performance of the simulation system is discussed in this section. The performance of two simulations, containing 322 and 1138 reactants, has been measured on an Intel Core i7-3930 CPU 3.20 GHz coupled with a GeForce GTX Titan X graphics card using the Unity3D profiler. The results of both tests are shown in Table 1. While reactions are generated and executed when needed, only the movement step is processed per frame. Therefore, the average processing time during approximately 130 frames was measured in both tests to give a more realistic representation of the overall processing time. This time interval of about 130 frames corresponds with the adjusted reaction cycle of the COPASI API, in which new reactions are initiated. Since the reaction system stores the list of new reactions given by COPASI and processes them over time by initiating only 20 reactions per frame, the processing time of the reaction generation step fluctuates between 0ms and approximately 160ms. While the processing time of the movement step was consistent in both tests, the reaction execution step fluctuated as well. During the first test the performance to execute completed reactions was in a range between 0ms and 2ms. Since more reactions have to be executed over time with an increased number of reactants participating in the simulation system, the processing time of the reaction execution step increased and fluctuated between 0ms and 15ms during the second test. Those immense fluctuations of the processing time during reaction generation and execution lead to non-stable frame rates and possible stuttering during the simulation.

Simulation Steps	Test 1 [ms]	Test 2 [ms]
Reaction Generation	3	12
Movement	13	33
Reaction Execution	1	7

Table 1: Average performance results of the single steps of the reaction system during approximately 130 frames. The first test included 322 reactants and was executed at 45 frames per second, while the second test contained 1138 reactants and was performed at 18 frames per second.

7 CONCLUSION AND FUTURE WORK

We have introduced a tool to simulate and visualize biochemical reactions and biological networks in a large and complex multiscale structural model. Due to the collision detection algorithm, which is able to work at the level of single atoms, the implemented reaction system is able to simulate molecular interactions. For load balancing, advanced GPU programming was used for

data manipulation as well as optimization algorithms to minimize the number of calculations per frame and to enable simulation with more than 1000 reactants participating in the reaction process. Due to the size and complexity of cells and their inner life, containing billions of atoms, it is necessary to visually communicate the proceedings during the simulation to the user. Therefore, three visualization techniques have been implemented. A real-time glow effect in combination with a conical clipping object are used to point out interesting areas where reactions occur. The third implemented visualization technique, a hierarchical structure called reaction tree, is used to illustrate a biological network, by illustrating the impact of one reaction on the entire reaction system.

Although this reaction system is able simulate more realistic molecular behavior it also encloses some limitations: Firstly, the explanatory visualization of biochemical processes is limited by the overall amount of ongoing reactions during the simulation. Currently, it is not possible that the user can follow a specific molecule during the reaction process. Therefore, a combination of a leaded camera and a slow motion technique could be implemented to improve the way how the processing of individual reactions are communicated to the user. Furthermore, this approach would allow the user to follow a specific molecule through out the scene and to illustrate the molecule's reaction pathway. The second limitation refers to the simplicity of the reaction process. In CellPathway, an arbitrary protein can be assigned to single reactions. Thus, the reaction position is moved to the location of the specific protein, whereby the molecular type of the protein is not taken into account. Furthermore, proteins are not synthesized or broken apart during the reaction process. In further versions, a more advanced reaction system could be implemented to simulate changes of the cell structure. Finally, the simplicity of the visualization tree makes it difficult for the user to analyze the created structure. Therefore, additional information like the direction of a molecule's path could be visualized by arrows or a color transition of individual lines.

SUPPLEMENTARY MATERIAL

A video showing the basic functionality of the project can be found at `https://youtu.be/dKFYqH1RNW0`. The source code of CellPathway is publicly available at `https://github.com/UnityDevTeam/CellPathway`.

ACKNOWLEDGEMENTS

This work was supported through grants from the Vienna Science and Technology Fund (WWTF) through project VRG11-010 and by the EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680.

REFERENCES

[MPSV14] Le Muzic M., Parulek J., Stavrum A.K., Viola I. Illustrative visualization of molecular reactions using omniscient intelligence and passive agents. *Computer Graphics Forum*, Vol.33, No.3, pp.141–150, June 2014.

[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*, Vol.22, No.24, pp.3067–3074, 2006.

[KMP10] Kubera Y., Mathieu P., Picault S. Everything can be agent! *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, Vol.1, pp.1547–1548, 2010.

[MAPV15] Le Muzic M., Autin L., Parulek J. and Viola I. cellVIEW: a Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets. *The Eurographics Association*, 2015.

[JAAGS15] Johnson G. T., Autin L., Al-Alusi M., Goodsell D. S., Sanner M. F., Olson A. J. cellpack: a virtual mesoscope to model and visualize structural systems biology. *Nature methods*, Vol.12, No.1 , pp.85–91, 2015.

[WSUnity] Unity Technologies. https://unity3d.com/. Accessed: 2016-02-11.

[WSPdb] PDB. Protein data bank contents guide: Atomic coordinate entry format description version 3.30. ftp://ftp.wwpdb.org/pub/pdb/doc/format_descriptions/Format_v33_A4.pdf. Accessed: 2016-02-11.

[CKMK13] Ciechomski P.H., Klann M., Mange R., Koeppl H. From biochemical reaction networks to 3D dynamics in the cell: The ZigCell3D modeling, simulation and visualisation framework. *Biological Data Visualization (BioVis)*, IEEE Symposium, pp.41–48, 2013.

[WSSbgn] SBGN wbsite. http://www.sbgn.org/Main_Page. Accessed: 2016-02-11.

[D14] Gehrer Daniel. CellUnity - an Interactive Tool for Illustrative Visualization of Molecular Reactions. *Institute of Computer Graphics and Algorithms, University of Technology*, 2014.

[SB01] Stiles, Joel R. and Bartol, Thomas M. and others. Monte Carlo methods for simulating realistic synaptic microphysiology using MCell. *Computational neuroscience: realistic modeling for experimentalists*, CRC Press, Boca Raton, FL, pp.87–127, 2001.

[WSCB] CellBlender website. https://code.google.com/p/cellblender/. Accessed: 2016-02-11.

[WSB] Blender Online Community. Blender - a 3d modelling and rendering package, http://www.blender.org. Accessed: 2016-02-11.

[MWPV15] Le Muzic M., Waldner M., Parulek J., Viola I. Illustrative Timelapse: A Technique for Illustrative Visualization of Particle-Based Simulations. *Visualization Symposium (PacificVis)*, 2015 IEEE Pacific, Vol., pp.247–254, 2015.

[GKMRE14] Grottel S., Krone M., Müller C., Reina G., Ertl T. MegaMol - A Prototyping Framework for Particle-Based Visualization. *Visualization and Computer Graphics*, IEEE Transaction, Vol.21, No.2, pp.201–214, 2014.

[HGVV16] Hermosillaa P., Guallarb V., Vinacuaa A., Vazqueza P.P. High quality illustrative effects for molecular rendering. *Computers & Graphics*, Vol.54, pp.113–120, 2016.

[Nygu07] Nguyen H. GPU Gems 3, Chapter 32. *Addison-Wesley Professional*, 2007.

[WSNT] Nvidia - GPU Technology Conference. http://on-demand.gputechconf.com/gtc/2014/presentations/S4117-fast-fixed-radius-nearest-neighbor-gpu.pdf. Accessed: 2016-02-11.

[Fer04] Fernando R. GPU Gems, Chapter 21. *Addison-Wesley Professional*, 2004.