CellUnity - an Interactive Tool for Illustrative Visualization of Molecular Reactions

Daniel Gehrer*
Supervised by: Mathieu Le Muzic†, Ivan Viola‡

Institute of Computer Graphics and Algorithms
University of Technology
Vienna / Austria

Abstract

CellUnity is a tool for interactive visualization of molecular reactions using the Unity game engine. Current mesoscale visualizations commonly utilize the results of particle-based simulations, which account for spatial information of each single particle and are supposed to mimic a realistic behavior of the metabolites. However, this approach employs stochastic simulation methods which do not offer any control over the visualized output. CellUnity, on the other hand, exploits the results of deterministic simulations which are purely quantitative and in that way offering full user control over the spatial locations of the reactions in the visualization. The user is able to trigger reactions on demand instead of having to wait or search for a specific type of reaction event, while the quantities of displayed molecules would still be in accordance with real scientific data. CellUnity exploits the simulation results in real time and allows the user to freely modify simulation parameters while the system is running. The tool was realized in Unity, a cross-platform game engine that also comprises a free version with adequate functionality and therefore enables easy deployment of the project.

Keywords: Unity, visualization, molecular reactions, quantitative simulation, interactive visualization

1 Introduction

Biochemistry allows a deep insight into cells and the synergy of molecular processes. Without any visual explanation, biochemistry can be difficult to understand [1]. Hence, it is necessary to visualize these processes to gain a better and more intuitive understanding of what is happening inside a cell [2]. For learning and comprehension purposes it is also important to provide an interactive, gamelike environment in order that students can immediately experience the impact of modifications in a cellular environment [3]. Scientific illustrators usually utilize animated storytelling principles to visually explain molecular activities, e.g. a metabolic pathway. To achieve this, corresponding particles and reaction events have to be shown in a story-structured manner [1]. Available mesoscale visualization tools commonly utilize the results of particle-based simulations to generate illustrations depicting reactions of a given biochemical process. Particle-based simulations determine spatial information of each single particle and are supposed to mimic a realistic behavior of the metabolites. However, this approach do not offer any control over the visualized output [1]. In particle-based simulations it is extremely difficult to track a specific particle due to the chaotic motion. Also reactions cannot easily be observed in the complex environment [2]. Due to this problem, it is challenging to comprehend reactions describing a biochemical process. Even when single particles are tracked and brought to focus, there is still no guarantee that a desired or an interesting event will happen [1].

The goal of this project is to create a tool for interactive visualization of an illustrative molecular environment. The functionality and the implementation is inspired by the paper of Le Muzic et al. [1]. CellUnity exploits the results of deterministic simulations which are purely quantitative. This offers, in contrast to the existing approaches, full user control over the spatial locations of the reactions and avoids the chaotic diffusion motion [1]. The user is able to trigger reactions on demand instead of having to wait or search for a specific type of reaction event, while the quantities of the displayed molecules would still be in accordance with real scientific data. This makes it possible for the user to follow a specific reaction chain in a realistic environment, which is greatly valuable for the user's comprehension and for illustration purposes. Parameters such as reaction rates and particle quantities can be changed while the system is running. The impacts of these changes are immediately visualized.

Often, existing visual simulation environments like ZigCell3D are implemented as proprietary research prototypes that cannot be freely deployed on any machine [4]. Other tools like Molecular Maya or BioBlender are great for visualization but the created environments cannot be

*daniel.gehrer@student.tuwien.ac.at
†mathieu.muzic@tuwien.ac.at
‡ivan.viola@tuwien.ac.at

animated using a simulator [5][6]. The main contribution of this work is the implementation of such a visualization and simulation tool in Unity to enable easy and free deployment. Unity is a cross-platform game engine that also comprises a free version with adequate functionality [7]. CellUnity provides a user interface to create simple bio-molecular environments. It is possible to import molecular structures available from public databases, define molecule quantities, reaction rates, and even to locate individual particles in the environment. The settings can also be exported to bioinformatics standard formats for the usage in external applications.

2 State of the Art

There are already tools for visualizing molecular reactions depicting a biological pathway. All of them have been designed for a slightly different purpose. Yet they all share the goal to provide insight into biological processes by visualizing a cellular environment at mesoscale levels. In this chapter various available tools are examined in respect to their visualization capabilities for scientific correct mesoscopic storytelling to explain cellular activities.

2.1 Tools for Molecular Visualization only

Some tools like Molecular Maya, BioBlender and ePMV focus solely on visualizing cellular environments [5][6][8]. These tools are all implemented as plugins for 3D computer graphics software and use the host application for visualization. They are all capable of importing molecules from the RCSB Protein Data Bank and they are all available for free [9]. Molecular Maya uses Maya as host application, BioBlender uses Blender and ePMV can be used with Blender, Cinema4D and Maya [5][10][6][11][8][12].

Molecular Maya supports various representation forms and also enables the user to easily extend structures, for example creating surface meshes and biological units [5]. BioBlender can animate transitions of conformations and visualize various molecular features, e.g. the electrostatic potential (EP) and the molecular lipophilicity potential (MLP). This kind of representation makes the nano scale world more understandable and is making it easier to conceive invisible phenomena such as hydropathy or charge [6]. The embedded Python Molecular Viewer (ePMV) does not only import molecules from different file formats but also keeps the link between structure file and the model. That way changes that are applied to the structure file after the import, are also applied to the model. The generated model is not just a static structure but can also be manipulated by the 3D host program or by python scripts that interact with ePMV.

All mentioned tools can be used to illustrate molecules but the models do not convey information about its function. To illustrate a cellular environment, the illustrator has to model the molecular processes manually using the host applications default tool set, which is a time-consuming and expensive task, taking up to months or years [1].

2.2 Tools for Molecular Visualization and Simulation

2.2.1 Visualization of Signal Transduction Processes

Falk et al. developed a visualization framework to explore simulation data of a virtual cellular environment [2]. The goal of the work was to highlight events of interest in the confusing environment. It especially helps Biologists to follow signaling molecules through the cell. The user can interactively select individual molecules and zoom into the virtual cell. It is possible to visualize individual molecules, their tracks, or reactions. The work is suitable for detailed, realistic, spatial simulations, where each molecule is an independent agent. A simulation usually covers several hundred frames. The user can step through each frame by keystrokes. The work also includes a virtual microscope to create images which can be compared with results from wet lab experiments. While the analyzing tool is interactive, the simulation is not [2]. The user has to perform the simulation again before it is possible to see the effects of the changes made. Also the tool is not openly available and therefore only used by a small set of users [2].

2.2.2 MCell and CellBlender

MCell (Monte Carlo Cell) is referring itself as micro physiological simulator [13]. It is a program to simulate the movements and reactions of molecules within and between cellular regions. For simulation, MCell uses spatially realistic 3D models and specialized Monte Carlo algorithms. It is intended to realize as realistic simulations as possible. The model can contain multiple compartments, which represent enclosed parts, e.g. organelles [13]. The meshes can be obtained from segmented volumetric imaging data or from CAD (computer-aided design) software [14]. MCell is free of charge and available for Linux, OS X, and Windows [13]. The model and the simulation conditions are defined in modular, human-readable text files, using a model description language [14].

CellBlender is an add-on for the free and open-source 3D computer graphics software Blender [11] [15]. The add-on is closely linked to MCell and enables the user to perform integral modeling tasks in Blender. It is possible to create, edit and visualize cellular models for the use in MCell. The simulation results generated by MCell, again, can be visualized in Blender, including the locations and states of participating meshes and molecules [15].

2.2.3 ZigCell3D

ZigCell3D is a software for modeling, simulating and visualizing an entire cell. The visualization covers several

orders of magnitudes, the full range from the cell surface to organelles and molecules down to the atomic level. The system also includes a virtual fluorescence microscope. For simulation two different approaches are used. On the one hand particle-based Brownian dynamics simulation, and on the other hand simulations based on Reaction Diffusion Master Equations (RDME), which have less spatial resolution but better performance [4].

ZigCell3D provides a real time interactive environment, where model parameters can be changed and the resulting effect can be analyzed in the 3D visualization of the cell. The user can select particles and can analyze the underlying rules and mathematical expressions that are responsible for the creation of the selected component or molecule. That frees the user from guessing the possible origin and bridges the divide between quantitative sciences and math-free wet-lab biology [4].

3 Methods

In CellUnity, the user has to set up the molecular environment before it can be simulated. The setup consists of importing different molecule species, setting the initial quantities, defining the reaction equations and setting the compartment size. The tool is interactive, so that the user is able to explore and immerse into a virtual 3D cellular environment. It is possible to track molecular compounds and also trigger reactions manually, while respecting quantities obtained via scientific simulation. In this chapter, the concepts and tools used in CellUnity are introduced. In the next chapter the concrete implementation is explained.

3.1 Development Environment

The project is built in Unity, a cross-platform game engine. Unity is not only a game engine but also includes an integrated development environment (IDE). Unity was chosen as framework because it is easy to use, quick to learn and there is also a free version with adequate functionality to realize this tool. Moreover, this enables the project to be easily shared and deployed, and allows the user to modify or extend the project with little effort. Additionally, Unity provides built-in methods for visualizing 3D objects and has a built-in physics engine. These features speed up development and avoid that the project has to be created from scratch. Furthermore, the Unity editor can be extended easily to include custom plugins, which seamlessly integrate into the Unity interface [7].

3.2 Visualization of Molecules

Molecules are visualized at atomic resolution. CellUnity can import molecule species from the file system using PDB files or can download the structure information automatically from the PDB webserver using a given PDB ID [9]. The representation of a molecule is automatically created using the atom definitions in that PDB file. The imported molecules are displayed as bunch of spheres, each representing an atom. The locations of the atoms are in accordance with the PDB file. The size of each atom corresponds with the Van der Waals radius of the associated element. This representation is often referred as Van der Waals representation [1]. An example is shown in figure 1.

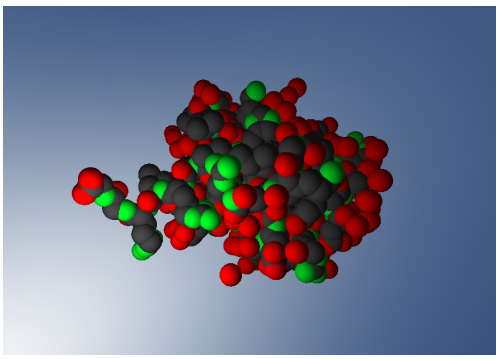

Figure 1: Visualization of an ubiquitin [16] molecule in CellUnity

3.3 Simulation

CellUnity is coupled to a simulator to mimic a realistic behavior in the visualization. For simulation the biochemical simulator COPASI is used [17].

As soon as the simulation is started, the user defined initial state is transmitted to the simulator. Since CellUnity only needs the number of reactions occurred, the simulation is purely quantitative. The simulation is performed step by step. After each step, the results are transmitted back to CellUnity and the reactions are performed in the visualization. It is also possible to modify simulation parameters after each step. The duration of such a step is adjustable by the user.

Reaction events are solely triggered by an omniscient intelligence (OI), like proposed by Le Muzic et al. [1]. In this system, molecules are passive agents, according to the definition by Kubera et al. [18]. Unlike in spatial-based simulation methods, molecules in CellUnity are unable to initiate reactions but can only receive reaction orders from the OI. The OI is influenced by the simulator and controls the molecules accordingly. Thus reactions in CellUnity do not just happen but are actively forced. The OI uses the current simulation state and takes action to achieve the same state in the visualization. Concretely, the OI reads out the quantity of reactions that occurred in the simulation for each reaction type, and forces the same quantity of reactions to happen in the visualization. Therefore the simulation and the visualization are quantitatively synchronized [1].

When a reaction is initiated, the OI selects random or user selected candidates according to the species of the

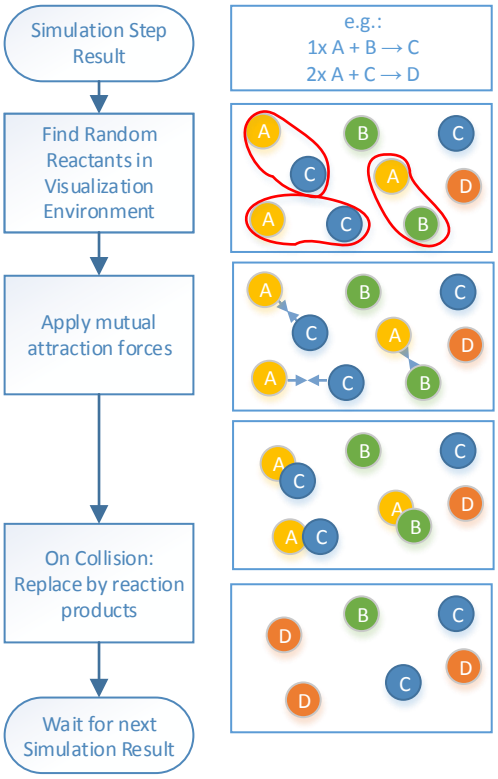

Figure 2: Example: Apply simulation results in visualization.

reactants and applies mutual attraction forces to force reaction partners to meet. As soon as they collide, the reactants are replaced by the reaction products. An example is shown in figure 2. The motion of molecules and also the collision detection is realized using Unity's built-in physics engine. Depending on if colliding molecules should react, the reaction takes place, if not, they repel each other via bouncing motion [1].

3.4 Navigation and Storytelling

To allow the user to navigate through the molecular environment, a navigation similar to a first-person-game is enabled. The user can turn around using the mouse and move with the W, A, S, D keys. Clicking on a molecule selects it. The user can adjust the view to automatically follow the selected molecule in the space or tag it for a reaction. If a molecule is tagged for a reaction, its priority will be set to react first when the OI initiates a new reaction. This allows the user to easily follow a reaction chain along a metabolic pathway. This is an easy way to comprehend reaction chains without having to wait until reactions happen on themselves, and is useful for storytelling.

3.5 Data Persistence

Unity usually serializes game data into so-called assets to persistently save them [7]. CellUnity uses this feature to save environmental data like molecule species and reactions. Molecules are implemented as GameObjects and therefore can be saved and restored in scenes when Unity is in edit mode. The position of every molecule is also preserved that way. In game mode the scene cannot be saved but the current state can be exported to an SBML file. The export functionality is available in edit mode as well. The SBML functionality is acquired by an external library.

4 Implementation

CellUnity is implemented as a project inside Unity. Custom editors are used to allow the user to configure the molecular environment. CellUnity's implementation is divided into individual classes. It is heeded that responsibilities of each class is well defined, to ensure coherent program modules that are as independent as possible. The CellUnity Environment (CUE) implements the model of the cell, the custom editor serves as controller for this model and Unity provides the visualization. Together these components form a model-view-controller.

One custom editor window is implemented for CellUnity. Via this editor the user can model and modify the environment. It is possible to import molecule species from PDB, to add and remove reactions from the system, configure simulation properties and export the environment to an SBML file. The target of the changes made in the editor is the CUE, which holds all the environment properties and definitions.

4.1 CellUnity Environment

The CellUnity Environment (CUE) is the class that holds all environmental properties and definitions. The entire system can only contain one instance of a CUE, therefore it is implemented as singleton. The CUE contains the defined molecule species and reactions, the volume of the compartment, a molecule manager, a reaction manager and a simulation manager. Each manager focuses on a separate task. They are described in detail in later sections.

4.2 Saving

Because all the environmental information is stored in the CUE, it makes sense to simply serialize the instance to persistently save the entire model when Unity is closed. Unity already provides automatic serialization methods. However, a few specific characteristics must be considered when used. Multiple references to one instance of a class are serialized multiple times, therefore, for every reference a new instance is created after restoring. This behavior is not satisfactory for species and reaction instances. Therefore these classes are derived from ScriptableObject. ScriptableObjects are serialized only once and multiple references are restored correctly [7].

4.3 Compartment

The CUE currently only supports one compartment and it has to be in the shape of a sphere. To keep the molecules inside the compartment, collisions with the compartment wall must be detected and the particles must bounce off. The physics engine of Unity does not support inverted colliders, which would be required. Therefore the desired behavior must be implemented by user code [7]. For each molecule, the distance to the compartment center is calculated. If the distance exceeds the radius, the distance is set to the value of the radius and the velocity of the molecule is inverted.

4.4 Molecules

In CellUnity, molecule representations are GameObjects with a Molecule-script attached. The Molecule-script applies molecular behavior to the GameObject. Each molecule is an instance of a specific molecule species. To make it easy to insert new molecules, each species has a prefab asset. A prefab is a Unity asset type that allows to store a GameObject with all components and properties. It acts as a template from which it is possible to create new instances in the environment [7]. The prefab is automatically created when a new species is imported.

4.5 PDB Import

CellUnity can either import PDB files from the file system or download them directly from the PDB website. PDB files are text files which contain 3D structure information of biological macromolecules [19]. For the molecule representation, only the atom positions in the file are considered. To gather this information, a simple PDB parser was written.

When a new molecule species is imported, at first, a new MoleculeSpecies-instance is created and added to the CUE. Then an empty GameObject is created, which serves as the main object of the molecule. The main object gets the Molecule-script attached and the newly created species-instance assigned. All atoms defined in the PDB file are now created as sphere-primitives and are added as sub-objects to the main object. To get a Van der Waals representation of the molecule, the size, location and color of each sphere is set accordingly. For performance reasons only one spherical collider that is considered by the physics engine is used for the whole molecule. The newly created molecule is then saved to a new prefab asset and assigned to the new species as the template for the molecules.

4.6 Molecule Manager

The assignment of the MoleculeManager is to keep track of all molecules in the system. When the play mode in Unity is activated, each molecule registers itself to the MoleculeManager of the CUE. In the manager, all molecules are organized in separate lists, depending on their species and whether they are free or already in use for a reaction. These lists are implemented as double linked lists. One molecule can only be in one list at a time. This enables to find free molecules quickly and efficient, which is important for the preparation of reactions. The organization of the molecule manager is depicted in figure 3.

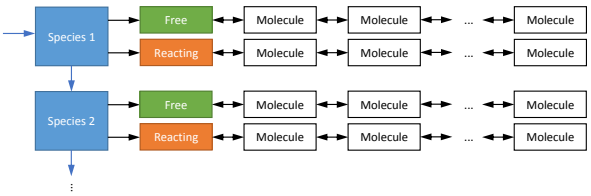

Figure 3: Schematic Diagram of the Molecule Manager

When a new reaction is initiated, the reaction manager asks the molecule manager to find a set of molecules of specific species that are near together. Due to the organization in the molecule manager, the nearest free molecule of a specific species can be found in $O(n+m)$ where n is the number of species, which is usually very small, and m is the number of molecules which are "free" and of the defined species, meaning only a fraction of all molecules. When a molecule's state changes from "free" to "reacting", it has to be removed from the "free" list and added to the "reacting" list. This state change can be performed in $O(1)$.

4.7 Automatic Molecule Placement

Since it is impractical to place each molecule manually, CellUnity offers to place a defined initial quantity of molecules automatically and randomly. The initial quantity can be set in the species editor. A problem that can possibly occur is that due to the randomness two or more molecules are placed too near together so that their colliders intersect. When this happens in Unity, particles repel each other with an unusually strong collision response which we ought to avoid. In CellUnity the problem is solved using the physics engine itself. The initial drag of the molecules is set to a very high value. As a result, colliding molecules repel each other gently until they do not intersect each other any longer and then remain steady next to each other. This procedure is only performed once, when the molecules are placed.

4.8 Reactions

Reactions that are possible in the system must be defined to the CUE. They are defined as ReactionType-instances. A reaction type consists of one or more reactants, zero or more products and a reaction rate. Reactants are the molecule species that are needed to perform a reaction.

Products are molecule species that are produced when a reaction was performed. The rate defines how many reactions should happen in a given amount of time. The reaction law is by default "Mass action (irreversible)" and cannot be changed in the current version of CellUnity.

4.9 Reaction Manager

The assignment of the ReactionManager is to initiate reactions and to perform them when all reactants collide. Reactions are usually initiated by the simulation manager. When a reaction of a specific reaction type is started, a new ReactionPrep (short for reaction preparation) object is created, which stores all important information about the reaction. Then the reaction manager asks the molecule manager to choose some free molecules of the species of the reactants for the reaction. The user can influence which molecules are chosen next by selecting them. If no molecule is selected, the reactants are picked randomly. If not enough molecules are available, the reaction is noted and delayed until enough molecules are available. This is important to guarantee correct molecule quantities on the long run. Such a delay can happen when the visualization is slower than the simulation, for example when the reacting molecules are located far apart, or when they hit obstacles which slow them down before reacting. However, if enough molecules are available, they are linked with the ReactionPrep object. Every molecule can only be linked to one ReactionPrep object at a time. Molecules linked to the same ReactionPrep instance and therefore are reactants of the same reaction, attract each other. This ensures that they will collide. When two molecules collide, they inform the reaction manager. The reaction manager checks if they belong to the same reaction, if yes, they are both tagged as "ready". As soon as all reactants are ready, the reaction is performed. The reactants are then replaced by new product molecules.

When a reaction is initiated, the reactants attract each other and accelerate towards each other. Due to their physical properties they possibly do not collide immediately but start to orbit the common barycenter. This can result in an endless circulation with the molecules never collide. To avoid this, a drag is set in the environment for all molecules. As a consequence orbiting molecules slow down and collide after a short time.

4.10 Simulation

CellUnity is coupled with COPASI, a tool for quantitative modeling and simulation. COPASI is used to simulate the user defined molecular environment. The communication is enabled via the C# application programming interface (API) provided by COPASI [17]. The simulation is started and administered by the simulation manager. The manager is also responsible for the data transfer with the simulator as well as the utilization of the simulation results.

4.11 Simulation Manager

Prior to the real-time simulation of the environment, the CellUnity model has to be transferred to COPASI. The compartment, the species and the reactions from the CUE are added to COPASI via the API. The initial quantities of the species in COPASI are set to the count of the species currently located in the CUE. When the model is changed, it is re-transferred to COPASI. Because CellUnity only pursuits of quantitative correctness, everything needed from the simulator are the number and types of reactions performed in the simulation. To gather this value for each reaction type, a "global quantity" model value is added. The value is defined as the ParticleFlux of the particular reaction. The type of the model value is set to "ode", so the value is the total value of performed reactions of this type.

The simulation is performed in steps. In CellUnity, there is a time for the "visualization step" and a time for the "simulation step" that the user can define. The "visualization step" is the real time interval of a step. The "simulation step" is the time simulated in such a step. After each simulation step, the ParticleFlux of each reaction is compared with the value before that step. The difference is the number of reactions performed during this step. The same number of reactions is then initiated in the visualization.

5 Summary

This paper presents an interactive tool for illustrative visualization of molecular reactions. It enables the user to build a simple molecular environment and simulate it in real time. It is possible to import molecular structures available from public databases, define reactions, and locate molecules in a compartment. Existing visualization tools commonly utilize particle-based simulations to generate illustrations depicting reactions. This approach provides highly realistic visualization, however, it does not offer any control about the visual output. Due to this, it can be hard to follow a specific chain of reactions, because reactions occur randomly and it is not guaranteed that anything interesting will happen in the user's sight. CellUnity, on the other hand, allows the user to trigger reactions and can automatically follow molecular reactions along a metabolic pathway. Even though the user interacts with the environment, the visualization remains in accordance with real scientific data. This enables the user to experience and comprehend metabolic processes. The model created in CellUnity can be exported as SBML file and used in other applications. Another advantage is that only free software is used to develop CellUnity. Hence, CellUnity can be easily deployed, modified and extended by everyone.

References

[1] Mathieu Le Muzic, Julius Parulek, Anne-Kristin Stavrum, and Ivan Viola. Illustrative visualization of molecular reactions using omniscient intelligence and passive agents. *Computer Graphics Forum*, 33(3):141–150, June 2014.

[2] Martin Falk, Michael Klann, Matthias Reuss, and Thomas Ertl. Visualization of signal transduction processes in the crowded environment of the cell. In *Proceedings of the 2009 IEEE Pacific Visualization Symposium*, PACIFICVIS '09, pages 169–176, Washington, DC, USA, 2009. IEEE Computer Society.

[3] Marc Prensky. Computer games and learning: Digital game-based learning. *Handbook of computer game studies*, 18:97–122, 2005.

[4] P. de Heras Ciechomski, M. Klann, R. Mange, and H. Koeppl. From biochemical reaction networks to 3d dynamics in the cell: The zigcell3d modeling, simulation and visualisation framework. In *Biological Data Visualization (BioVis), 2013 IEEE Symposium on*, pages 41–48, Oct 2013.

[5] Molecular Maya Toolkit website. http://www.molecularmovies.com/toolkit/. Accessed: 2014-08-21.

[6] Raluca Mihaela Andrei, Marco Callieri, Maria Francesca Zini, Tiziana Loni, Giuseppe Maraziti, Mike Chen Pan, and Monica Zoppè. Bioblender: A software for intuitive representation of surface properties of biomolecules. *CoRR*, 2010.

[7] Unity Technologies. http://www.unity3d.com/. Accessed: 2014-08-21.

[8] Graham T. Johnson, Ludovic Autin, David S. Goodsell, Michel F. Sanner, and Arthur J. Olson. epmv embeds molecular modeling into professional animation software environments. *Structure*, 19(3):293–303, 2014.

[9] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[10] Autodesk Maya. `http://www.autodesk.de/products/maya/`. Accessed: 2014-08-21.

[11] Blender Online Community. Blender - a 3d modelling and rendering package, http://www.blender.org. Accessed: 2014-08-21.

[12] Maxon Cinema4D. http://www.maxon.net/de/products/cinema-4d-studio.html. Accessed: 2014-08-21.

[13] MCell website. http://mcell.org/. Accessed: 2014-08-21.

[14] Rex A. Kerr, Thomas M. Bartol, Boris Kamubsky, Markus Dittrich, Jenchien Jack Chang, Scott B. Baden, Terrence J. Sejnowski, and Joel R. Stiles. Fast monte carlo simulation methods for biological reaction-diffusion systems in solution and on surfaces. *Nucleic Acids Research*, 2008.

[15] CellBlender website. https://code.google.com/p/cellblender/. Accessed: 2014-08-21.

[16] PDB ID: 1UBI
Ramage, R. and Green, J. and Muir, T.W. and Ogunjobi, O.M. and Love, S. and Shaw, K. Synthetic, structural and biological studies of the ubiquitin system: the total chemical synthesis of ubiquitin.

[17] Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jrgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. Copasia complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, 2006.

[18] Yoann Kubera, Philippe Mathieu, and Sébastien Picault. Everything can be agent! In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, pages 1547–1548, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[19] wwPDB. 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`.