Mejia *et al. Visualization in Engineering* (2018) 6:6
https://doi.org/10.1186/s40327-018-0067-4

Visualization in Engineering

CASE STUDY Open Access

NemoViz: a visual interactive system for atomistic simulations design

Daniel Mejia*, Tillmann Kubis and Gerhard Klimeck

Abstract

Introduction: Given that the nanoscale regime has been reached, atomistic simulations are being used as predictive tools on a nanoscopic scale in nanoelectronics, materials science, and computational fluid dynamics. These simulations are generally supported by complex and specific simulation engines that require a deep knowledge of the engine as well as the field. The inputs required by these engines are usually described in long-text files, and their composition is error prone. These files contain details about the corresponding materials, geometries, algorithms, initial values, etc. The introduction of visualizations to the simulation creation process could alleviate some common user issues with simulation creation, in particular regarding the input-text files. In other fields, visual analytics have been used to facilitate user interaction with complex data. This work features a case study on the creation of a nanoelectronic device simulation and provides evidence of how the use of visual analytics reduces the cognitive complexity of defining an atomistic simulation.

Case description: NEMO5 is a tool designed to simulate the electronic properties of nanoelectronic devices on an atomistic level. In this work we introduce NemoViz, an interactive visualization tool that enables users to define the simulation inputs required by NEMO5. Regular NEMO5 users were exposed to NemoViz, and users' effectiveness and efficiency was measured while debugging the input for a simulation.

Discussion and Evaluation: The results of this work show that NemoViz reduced the time that users spent defining the inputs of a NEMO5 simulation. When using NemoViz, Expert NEMO5 users detected errors twice as fast as when NemoViz was not used, and non-expert NEMO5 users were able to detect errors as effectively and efficiently as expert users.

Conclusions: These results suggest that the use of visual analytics as a simulation design process tool reduces the cognitive load of complex simulators such as NEMO5. Users' interaction with visualization facilitates their understanding of output results and input descriptors, which may lead to new research codes from their widespread user base.

Keywords: Visualization in physical sciences and engineering, Software visualization, Visualization in education, Nanoelectronics

Introduction

Nowadays, complex simulations of a variety of processes are extensively used in academia and industry (Pedone 2009). For instance, since the 1960's, the drift-diffusion model complemented by the understanding of electron transport in transistors have driven work in computational electronics on the mesoscopic scale (Lundstrom 2015) . Most recently, other models, such as the Non-Equilibrium Green's Function (NEGF) formalism to simulate quantum transport, have been incorporated into codes to simulate phenomena on the nanoscopic scale. Applications of nanoscopic-scale atomistic simulations are commonly found in nanoelectronics, materials science, and computational fluid dynamics (Lundstrom 2015).

The developers of atomistic simulators usually deal with complex physical problems that require the use of intricate mathematical models and extensive numerical solutions. Commercial simulation engines are rare, and the

*Correspondence: dmejiapa@purdue.edu
School of Electrical and Computer Engineering, 610 Purdue Mall, West Lafayette, USA

development of tools in this realm is still ongoing. In contrast, scientists often opt to develop their own simulation codes and focus on modeling the physical system as closely as possible. The code users ultimately define their simulation parameters, usually in text files processed by the simulator. To properly create input text files, users need an in-depth understanding of various technical fields (Lundstrom 2015) and their relations, which adds additional layers of user cognitive processing (Wilensky and Resnick 1999). As a result, using text files as simulation input restricts the number of possible users, reduces the chances of correct simulation parameterization (Wilensky and Resnick 1999; Hmelo-Silver and Azevedo 2006), and generally slows down the new user learning curve.

This work integrates visual analytics to explore and modify the text files used as inputs in a nanoelectronics simulation tool. Properly-designed, interactive visual aids allow the user to interactively explore different key aspects of the simulation, and as consequence to rapidly understand the simulation tool and simulation results.

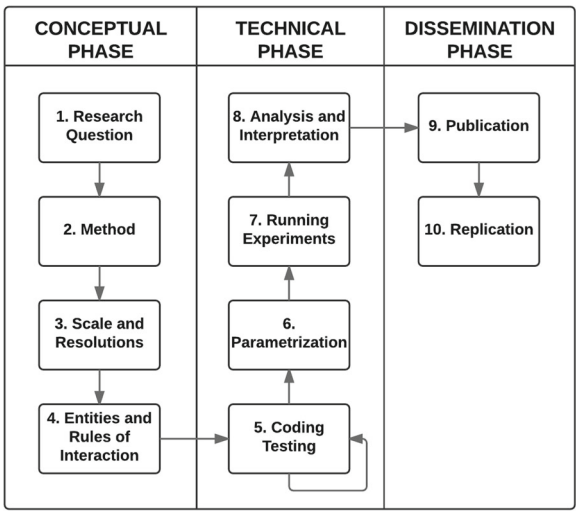

Fig. 1 Romanowska's ideal model development sequence

Background

Simulation process models

Three simulation design process models exist in the literature. The first documented definition of a simulation workflow model was proposed by Kruger (1970) (Krüger 1975). Kruger presented that the simulation workflow begins with the problem definition and proceeds to the stages of data collection and model building, validation, data analysis and interpretation, and documentation. Allen et al. (1990) proposed a scientific simulation workflow designed for fluid simulations (Allen and Tildesley 2017). This model highlighted the importance of conducting experiments in the real world and compared the results of the experiments with the simulation results using theoretical models. Most recently, Romanowska (2015) includes coding, testing, and result replication as part of the workflow (Romanowska 2015). This model classified the workflow steps into three main phases: the conceptual, technical, and dissemination phases shown in Fig. 1.

Users face different problems for each step of this process, and each step features specific cognitive challenges. In the nanoelectronics field, for example, the research question step typically begins with an understanding of the previously published results of experimental groups or compact models. Moreover, the data published as raw numbers usually misses background information. Thereafter, the experimental data is mapped onto a physical model with a specific device configuration. A visualization of the experimental results and device representation helps users to define questions about the device's dimensions and physical properties, including temperature and heat dissipation.

It has been shown that interactive visual representations amplify the cognitive process by incorporating external cognition (Scaife and Rogers 1996), the data is transformed into visualizations and users change these representations as they gain new insight. Visualizations have been widely used as graphical user interfaces that help users while they are running experiments as well as analyzing and interpreting data. In fact, visualizations can help users during the entire simulation process. The following two sections include examples of how visualizations have been used to support the simulation process

Commercial simulation visualization

Well-known commercial simulator packages (e.g., Simulink) use visualization to benefit their usability. Matlab introduced Simulink in 1992 (Simulink Simulation and Model-Based Design 2012) to support the coding and testing phases of the simulation process. Simulink represents complex blocks of code as visual boxes that can be parameterized. The corresponding outputs are visualized with different representations that have utility for the analysis and interpretation steps as well as the technical phase. Simulink's simplicity led Matlab to become one of the standard tools used in automatic-control and digital-signal processing.

Another visual aid example is the finite element simulation, in which solid models are represented as polygon meshes. For example, Comsol (COMSOL Multiphysics® Modeling Software 2005) included an interactive 2D/3D mesh representation of the model that helped users through steps 3 and 4 of the conceptual phase. Moreover, Comsol found that the spatially-resolved results overlapped with the three-dimensional model helped users during the analysis and interpretation steps.

Research simulation code visualization

Modeling in relatively new research fields -particularly research that involves the physical movements of atoms and molecules, or particles and atomic interactions— cannot always be simulated by commercial engines, and new tools are constantly being developed in conjunction with research simulation codes, evenly on the number of atoms that can be visualized and associated fields had seen a challenge (Qiao et al. 2005). Often, these codes are more concerned with solving scientific problems than their lacking usability.

Some developers are aware of this usability problem and have tried to address it via two approaches. The first approach included two graphical user interface layers that helped users with parametrization in addition to the analysis and interpretation step of the technical phase. For instance, MAPS (MAPS platform 2009) and Lammpsfe (Lammpsfe 2012) feature two visualization layers for LAMMPS, a well-known molecular dynamic code developed by Sandia National Laboratories. The first visualization layer is a wizard that enables users to create models; the second is a results visualization that follows the simulation. These aids have proven so invaluable to users that the inclusion of two visualization layers has been commercialized as research software. ATK applies visualization layers to a broad set of research codes, including LAMMPS, ABinit, and QuantumEspresso (Atomistix ToolKit (ATK) 2008).

Toolkits have been proposed to implement the simple two-layer graphical interface. (McLennan and Kennell 2010) and nanoHUB (Lundstrom and Klimeck 2006) provide a XML API that enables the graphical interface. Rappture allows users to not only describe the simulation models and visualize the simulation results, but also to execute experiments during the technical phase. For example, the Quantum Dot lab (Klimeck et al. 2006) employs NEMO5 to calculate electrical and optical properties of quantum dots; therefore, users can run experiments on different qdot sizes and materials. This tool presents the first layer of graphical control elements to set the simulation parameters. Then the results are displayed in a second layer that includes visualizations of the electron wave functions and optical properties. A limited number of visualizations are supported by Rappture, and additional visualizations require significant developments of the Rappture infrastructure (Zhao et al. 2017).

A second approach uses the rise of e-Science as a new research practice. The importance of reproducibility for independent verification. Steps included in the *technical phase* of the simulation process can be described as scientific workflows. Scientific workflow systems such as Kepler (Altintas et al. 2004) and Pegasus (Deelman et al. 2005) aim to simplify scientific workflow management, provide infrastructure, and define the hierarchical models, dataflow constraints, and model execution frameworks. Edison (Suh et al. 2016) is a web services-based scientific workflow system that aims to support technology computer-aided design (TCAD) applications. VisTrails (Callahan et al. 2006) integrates visualization modules into its workflow system.

This research proposes a third approach based on the successful implementation of visual analytics as a big data analysis tools. Visual analytics can be used as a graphical user interface to guide users during simulation design. The following chapter describes an interactive, user-focused visualization system. The system enables users to explore simulation inputs as visual representations. This interactive visualization system concept has been applied to a research simulation code called NEMO5. NEMO5 is the fifth edition of the NanoElectronics Modeling Tool. This simulation tool incorporates the core concepts and experience gained from more than 20 years of development. NEMO5 is a multi-physics simulation tool used to design devices on an atomistic scale. NEMO5 is used as a research and educational tool in both industry and academia. NEMO5 powers nine of the most popular tools on nanoHUB.org. nanoHUB is used by more than 1.4 million users per year in approximately 172 countries (Sellier et al. 2012).

Interactive visualization system

In this work, we propose that interactive visualization systems should be designed based on Romanowska's (Fig. 1) simulation process and 3-phase functional requirements.

Conceptual phase

During this first phase, users upload raw data into the system. The data is processed and transformed into visual representations, such as heatmap or line plots. The users should be able to visualize the available physical models and access documentation for each specific model. The documentation should contain descriptive information on the assumptions, formulas, restrictions, and possible parameters of the physical model. Users should also be able to explore the database via material properties and change values.

Technical phase

Given a specific physical model and its restrictions to represent external results, users should be allowed to include simulation inputs and process the information in order to extract the spatial and abstract components of the simulation as well as the relations between these components. Spatial components should be visualized as 3D models and the abstract components should be listed with their relations. Users should be able to filter and reconfigure each visible component. When users define a specific experiment, the system should run the experiment and

display the results as visual representations. These visualizations allow users to interpret, analyze, and compare, the results.

Dissemination phase

After the set of experiments are conducted and new insights are found, the researchers should describe, parametrize, and export the new model inputs. These tools should be capable of running the model and visualizing the output results. Exposing these tools to the scientific community helps to replicate the findings and enables other researchers to explore new parameter options that can lead to the discovery of new insights.

System Implementation

We developed NemoViz as a modular interactive visualization system. The system was developed using HTML5, Webgl, Javascript, Python, C++, and NEMO5. Users can inspect the NEMO5 input text files (inputdecks), explore the three-dimensional models, visualize the simulation results, generate reproducibility tools, and export the three-dimensional models that generate high quality images. These components were designed to support users through the simulation design process. The following sections will introduce the actual user interface and examples of how users can interact with a NEMO5 simulation.

System structure

Figure 2 describes the NemoViz architecture and its utility to develop new visualizations. NemoViz is based on a Model-View-Controller MVC and client-server architectures. The client side is implemented with JavaScript; the control elements are based on the Dojo toolkit library (Dojo Toolkit 2008); the Webgl visualizations use a Three.js (three.js - Javascript 3D library 2010) core; and the D3 (Bostock 2012) library supports all other visualizations. The server side is implemented via C++ and supported by the Boost library (Karlsson 2005), and visualization plugins are implemented in Python.

All user interactions with the client are captured by a web browser as JavaScript events and passed to the Asynchronous Javascript and XML (AJAX) controller. This controller dispatches events in the Webgl model and refreshes the views that need to be changed. The AJAX controller can also demand information from web services as JSON requests. All of the client's requests are captured in the server by the Web Services (WS) Controller.

The WS Controller completes four main tasks. The first task is to call a NEMO5 kernel and execute small calculations using NEMO5's API. These calculations are triggered by the NemoViz Model and sent to the WS Controller. NEMO5's results are returned to the Model as well. Then, the controller receives the changes required by the client views and notifies the AJAX controller. Instead of calling the NEMO5 calculations, the second task executes NEMO5 database queries and processes information related to the NEMO5 parameters. The third task is to report users' interactions to the NemoViz models. For example, if the user hides layers of information or includes advance calculation of atoms' positions, this information is required by the model to be updated. The last task is to execute the python code from a specific plugin. The WS Controller captures the HTML representations of a visualization, compresses the text, sends the text to the client, and notifies the AJAX Controller.

User Interface

NemoViz is based on NEMO5's input structure that consist of blocks of properties. A NEMO5 simulation is described as a text file (inputdeck), and this inputdeck is written in a C-like format (similar to a STRUCT statement). Table 1 contains a simple example of a NEMO5 inputdeck, which is divided into groups and identified by keywords at the beginning of each curly bracket, henceforth this groups are going to be called *Blocks*. NemoViz

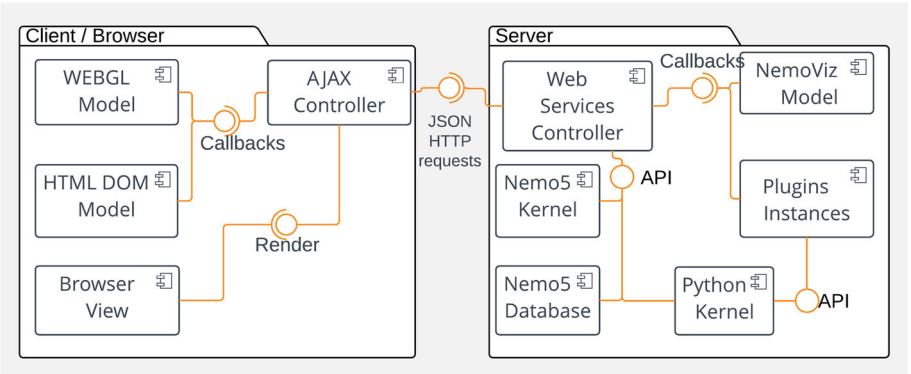

Fig. 2 The NemoViz system is implemented on a Model-View-Controller MVC and client-server architecture

Table 1 NEMO5 Gallium Arsenide Band-structure calculation inputdeck

```
Structure {
  Material {
    name = GaAs
    tag = substrate
    crystal_structure = zincblende
    regions = (1)
  }
  Domain {
    name = structure1
    type = pseudomorphic
    base_material = substrate
    dimension = (20,20,20)
    periodic = (true, true, true)
    regions = (1)
    crystal_direction1 = (1,0,0)
    crystal_direction2 = (0,1,0)
    crystal_direction3 = (0,0,1)
  }
  Geometry {
    Region {
        shape = cuboid
        region_number = 1
        min = (0,0,0)
        max = (5,5,5)
  }
  }
}

Solvers {
  solver {
    name = my_schroedi
    type = Schroedinger
    set {
        domain = structure1
        active_regions = (1)
        tb_basis = sp3d5sstar_SO
        job_list = (assemble_H, passivate_H, calculate_band_structure)
        output = (energies, eigenfunctions_VTK)
        charge_model = electron_hole
        automatic_threshold = true
        eigen_values_solver = krylovschur
        k_space_basis = cartesian
        k_points = [(0,0,0)]
  }
  }
  solver {
    name = my_overlap
    type = MatrixElements
    set {
        domain = structure1
        active_regions = (1)
        operator = overlap
        wf_simulation = my_schroedi
        output_file = matrix_elements
  }
  }
  solver {
    name = my_structure
    type = Structure
    set {
        domain = structure1
        active_atoms_only = true
  }
  }
}

Global {
  solve = (my_structure,my_schroedi,my_overlap)
  database = all.mat
}
```

was designed to show multiple visualizations of *Blocks* and relations found in the inputdeck. Figure 3. shows an example of an inputdeck loaded in NemoViz.

NemoViz layout consists of four main visual containers. These containers are synchronized according to each user action and enable additional visualizations as needed (e.g., if a domain is selected, the visualization details of the crystal should be displayed). Descriptions of the four containers are presented below.

1 ***Outline* View:** The outline view represents the hierarchical data and defines the simulation. It allows for viewing the inputdeck in two different ways: a hierarchical tree structure or plain text editor. A hierarchical tree structure is a natural way to represent the inputdeck blocks (see Fig. 3a). Users can collapse and expand a block by clicking on the folder icon. Users can transition from the tree-like view to the plain text editor using the "Input Deck Text" mode.
2 ***Properties* View**: The properties visualization represents the state of a block. Each block represents a set of parameters that configure part of the simulation. When a user selects a block, the properties view table appears populated with the block's name-value pairs (see Fig. 3b). Users can edit the table values and see the changes visualized in the main view and the relations view visualization containers. Also, the system alerts users about the errors detected in the parameters of a particular block via tooltip display. The properties view also includes a toolbar where users can hide or display the visual representations of the selected block and identify the possible block parameters and documentation.
3 ***Main* View:** This container is a set of visualizations. The main visualization is called the "workspace," which features post-processing visualization plugins. Each plugin is loaded in a different container (tab) with a unique name. The tabs can be renamed and closed. All visualizations can be accessed by clicking on the tab identifier name. The workspace visualization cannot be closed.
The workspace visualization consists of three-dimensional models that represent different aspects of the simulation. Any block that contains information about a spatial representation (e.g. Regions, Domains, Boundary Conditions, Finite Element Domains, etc.) is featured as a three-dimensional model in the main view (see Fig. 4). All models have an opacity level that enables users to visually detect the overlapped elements. Traditional zoom/pan interaction techniques are enabled so the user can visualize the models from different points of view. Information on the model elements is visible as a tooltip when the mouse is nearby the element.

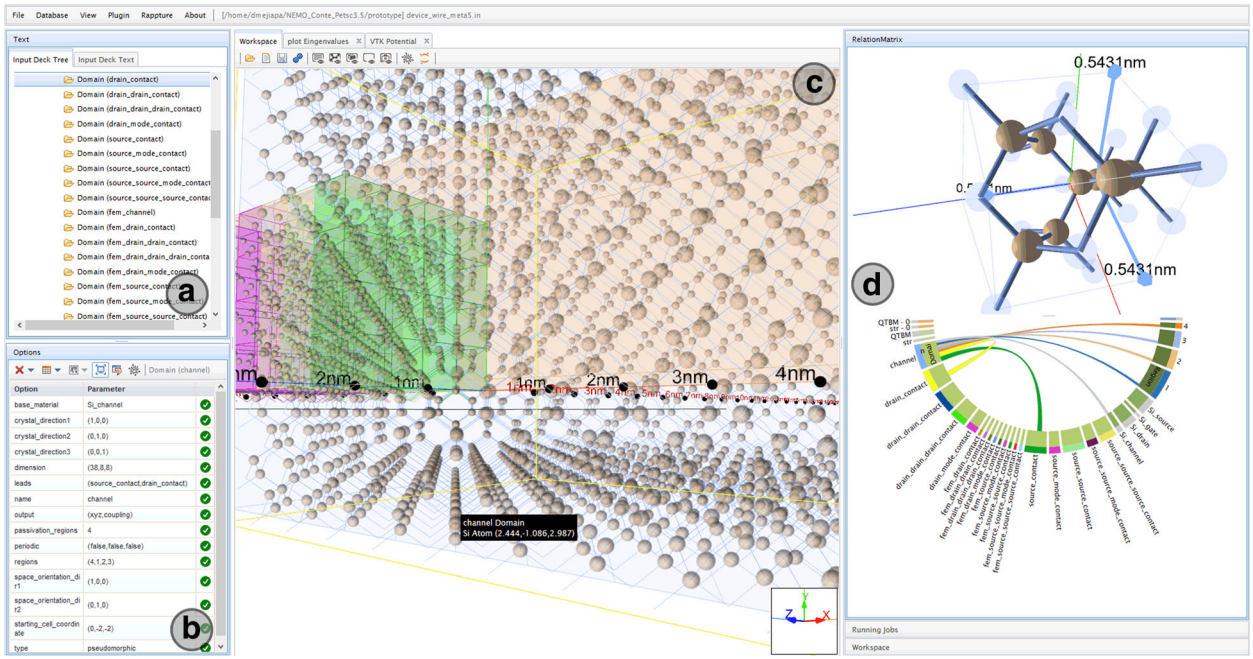

Fig. 3 The NemoViz client layout consists of four main visualization containers: **a** the outline view of the hierarchical data structures; **b** the properties view of editable options for a selected block; **c** the main view, a three-dimensional representation of the input-defined model; and **d** the relations view, an abstract representation of a selected block and its relationship with other blocks

4 ***Relations* View:** The Relations visualization represents a simulation block and its relations. This visualization is a hybrid of a bottom quadrant chord diagram (see Fig. 5) and a three-dimensional canvas. The chord diagram is represented in the bottom quadrant only, and displays the detected relations between inputdeck blocks. The relations view also represents blocks as geometrical models, a mesh, or an atomistic structure. The user navigates the different inputdeck blocks by clicking on the block's name in the chord diagram; thereafter, all other visualizations are synchronized accordingly.

NemoViz plugins

Plugins are post-processing visualization scripts that represent data. They are mostly used to show simulation

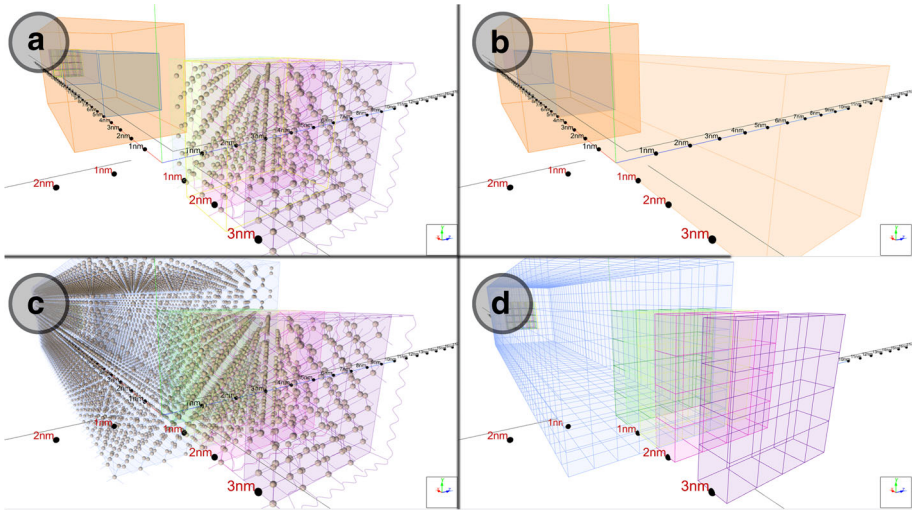

Fig. 4 In the NemoViz main view, blocks are visualized via spatial representation. Users can filter the overlapping models as follows: **a** different block types enabled for different device parts; **b** geometrical representation; **c** atomistic representation; and **d** mesh representation

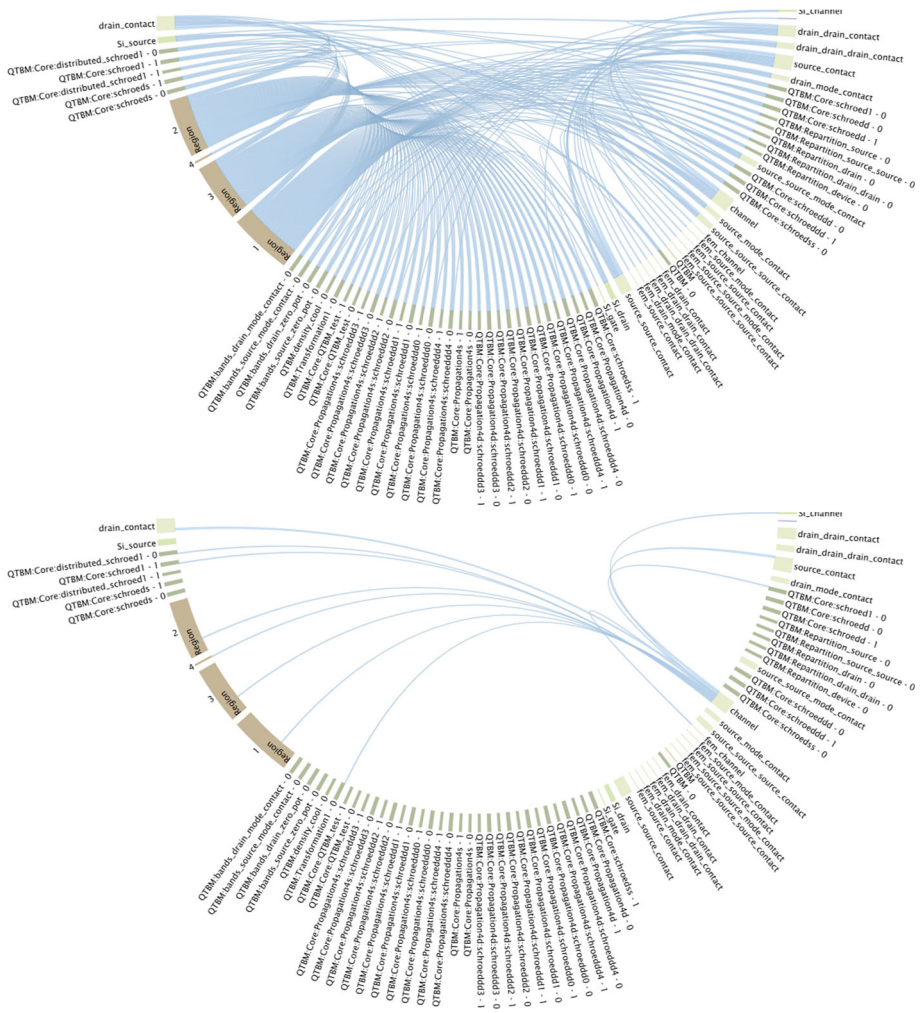

Fig. 5 Relations detected for an example quantum transport on a silicon nanowire. Top: Relations found in the inputdeck. Bottom: Relations detected involving region 1

results; however, they can also be used to visualize simulation memory consumption, time tracing, or object lifetimes. Plugin types represent a wrapper around specific python libraries. Currently, NemoViz supports five plugin types based on the popular visualization libraries: Bokeh, Plotly (Fig. 6a-c), Paraview (Fig. 6e), X3Dom and HTML/D3 (Fig. 6d). Each plugin can be parametrized and configured. Meta variables allow users to directly change the parameters from the NemoViz client and execute the same script with various inputs. Visualizations vary from simple line plots or histograms to heat-maps and three dimensional surfaces. They also include different layers of information (e.g., contour visualizations).

Dissemination tools

NemoViz includes two main components that can help researchers disseminate their findings. The first component allows users to export three dimensional models as a Threejs scene. Threejs describes all elements included in the scene as a JSON files, including cameras, lights, and geometrical definitions. These files can be imported into Blender using a Blender add-on called the NemoViz loader. Blender allows researchers to create high quality pictures of their models..

The second component is the reproducibility exporter, which allows researchers to define parameters to the inputdeck and export these parameters for publication via one of the two following options. The first option exports the parameters as a Rappture tool that can be published on nanoHUB.org. This tool consists of a XML file that describes the graphical user interface and a python script that utilizes the parameters from the graphical user interface, creates a valid NEMO5 inputdeck, and executes NEMO5. By default, the Rappture tool captures all the output from the simulation as a log file, but researchers

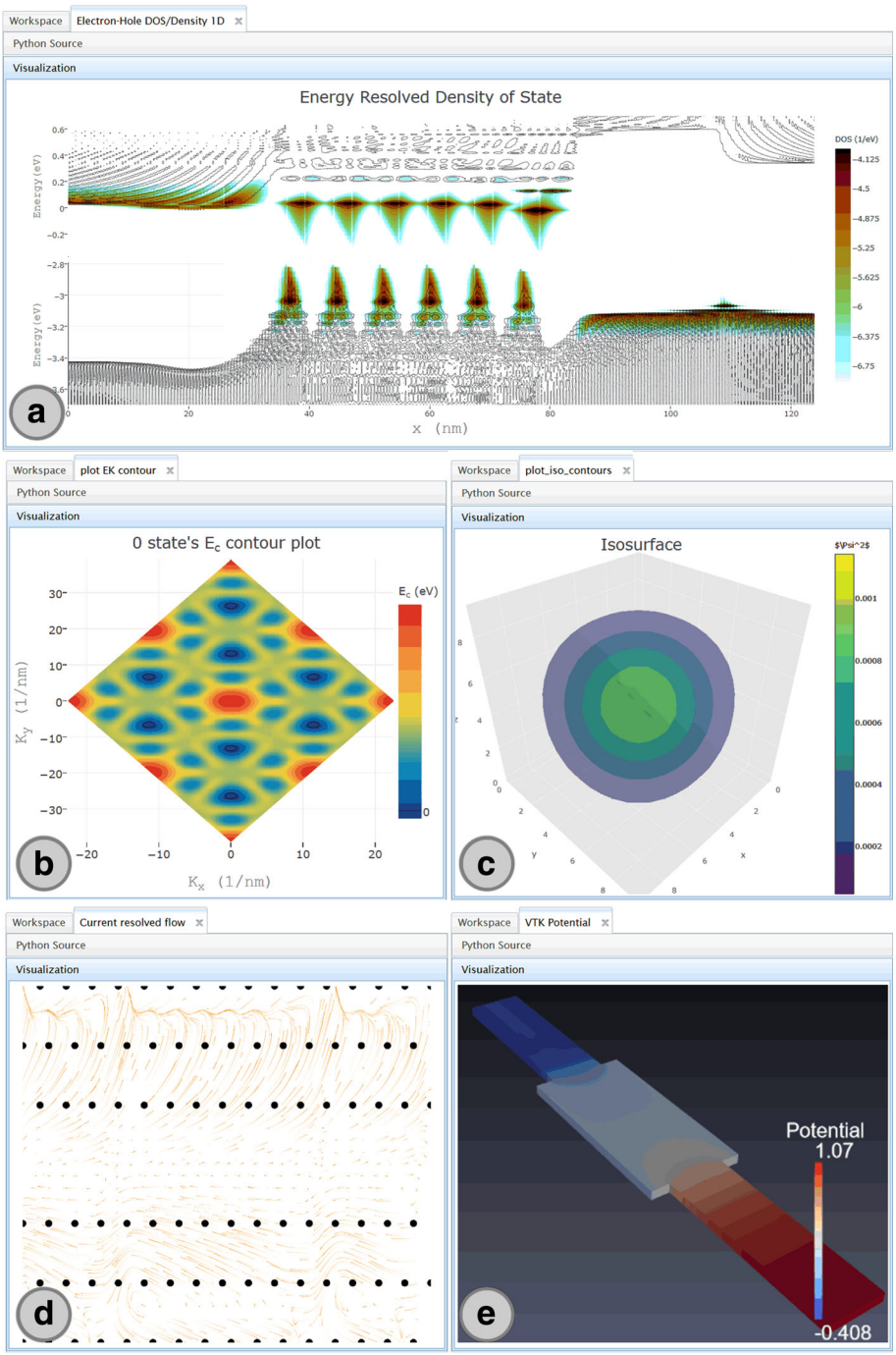

Fig. 6 The NemoViz plugins: **a** a light-emitting diode simulation with density of states represented as a heat-map with different energy values; **b** a GaAs simulation with density of states mapping the Brillouin zone; **c** a quantum-dot simulation with eigenfunctions of the ground state represented as 3D contours; **d** a transport simulation of MoS2 sheets with current flows represented as streamlines between atoms; and **e** a silicon ultra-thin-body (UTB) simulation with potential fields represented as a mesh surface

can modify the python script to include a visualization as part of the output as well.

The second option allows users to export the input-deck as a Jupyter notebook. In this case, researchers have the option to predefine the parameters and include visualization plugins. The exported Jupyter notebook translates all selected parameters as Jupyter widgets and plugins code as functions. The Jupyter notebook also includes button

widgets that triggers the execution of those functions. Jupyter notebooks can be downloaded and published on nanoHUB.org as public tools.

Case description

Here, we show how our interactive visualization tool NemoViz facilitates the simulation design process in NEMO5. For this purpose, we consider the case of a scientist who is searching for materials as potential candidates to replace silicon in electronic devices and is designing a simulation to help him in this search. The ultimate goal of the simulation is to show the electronic properties of a selected material and geometry, which are represented as energy levels and wave functions. As we describe this case study we will explain how NemoViz supports the selection of the simulation parameters. We will describe the energy value calculations of a simple quantum dot (a cube of 5 mm,) of Gallium-Arsenide with a sp3d5s*SO tight-binding model.

The process of designing a simulation begins with the conceptual phase, during which researchers pose research questions and define their methods. The first research question that a scientist using NEMO5 might pose is: which material could be a potential candidate to replace silicon? In other words, which material do I choose to run the simulation(s), and what is the best model to describe that material's properties? NemoViz includes visualizations that support this part of the conceptual phase by helping users to explore the material databases, sets of parameters, models, and methods available in NEMO5. Figure 7 displays the database exploration interface. The information is displayed as a treetable that can be expanded by selecting a specific material. Users can also query the database parameters. The calculations required by a parameter are displayed on the screen, which is useful because some material parameters are defined as functions that depend on other parameters.

After defining the appropriate materials and method, the scientist needs to choose a geometry, general solution(s), and corresponding restrictions. These steps correspond to steps 3 and 4 of the conceptual phase described in Fig. 1. In this case study, the scientist intends to simulate a simple representation of the selected material (i.e. a box full of atoms defined as three-dimensional, stacked unit-cells). This structure represents a well-known quantum mechanical problem also known as the simple quantum dot (Klimeck et al. 2006). NEMO5 solves the particle in a box system using the Schröedinger equation. Wave functions are calculated for different energy levels, and the solution's eigenfunctions are described for the 3D atom space.

All previously described details must be translated into a NEMO5 input text file (inputdeck). Table 1 shows the inputdeck that describes this simulation experiment. As mentioned in the user interface section, the inputdeck is organized in blocks. The first block of Table 1 is called *Structure*, which contains information about the device structure and materials. Additional information about the material, atomic composition, and nonstandard material parameters is contained in the Materials block. The Geometry block specifies the geometric shapes of individual regions. The Domain blocks define which regions are aggregated within a domain. In this particular example, one material type (i.e. GaAs) is defined in the Material block; a single region (i.e. a cuboid) is defined in the Region block; and a single instance of the material (Domain) is defined in the Domain block

Having the option to graphically represent the inputdeck helps users during the scaling, resolution, rules of interaction, and parametrization steps. NemoViz helps the scientist to understand the complexity by generating visual representations of the blocks' spatial information. Figure 8 is a visualization of the inputdeck described in Table 1. The main view (Fig. 8a) defines three-dimensional representations of all atoms, and a 5nm cuboid. All atoms

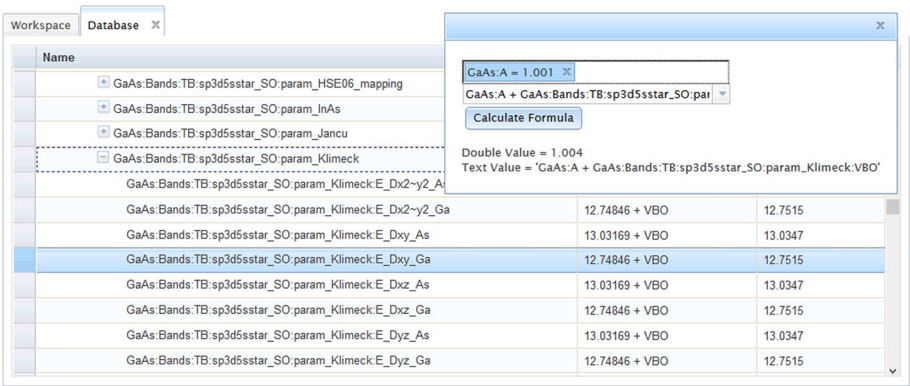

Fig. 7 Database explorer visualization interface, this interface allows users to explore Nemo5 material database values and their definitions

are located in their expected spatial position. The atoms' information is displayed and the bonds between atoms are also visualized. This information is defined by crystal parameters. The scientist can also change the visual representations to display the simulation atoms only (Fig. 8b).

The inputdeck shown in Table 1 also contains definitions of three simulation steps, or the so-called solvers. The Solvers block contains information about the simulation types. Each simulation has a set of options specific to its task. In this example the scientist defines: 1) a solver for the output of the domain to a VTK file ("my_structure" solver), 2) a solver for the Schröedinger equation ("my_schroedi" solver); and 3) a solver for the calculation of optical properties based on the eigenfunctions of the Schröedinger equation ("*my_overlap*" solver). This latter solver illustrates interrelations between solvers. However, solvers are not only connected to other solvers; solvers are also connected to other inputdeck blocks. For example, "wf simulation" and "my schroedi" solvers are related to the domain "structure1" and region number 1. This region is also related to the domain structure1 and the material substrate.

The aforementioned relations are detected by NemoViz and represented in the relations view shown in Fig. 8c. Figure 8d-e show the relations view when a domain or a region is selected. If the user selects the domain "structure1," the arcs connected with this block are shown and a representation of the basic unit cell is visualized in the upper portion of the visualization. Similarly, if the user selects region 1, the cuboid appears and the relations of region 1 are displayed as well (Fig. 8e).

After the parametrization of the model, the scientist defines a set of experiments (step 7), in which each experiment is a specific modification of the input deck and contains a fixed set of values. NemoViz's visual interface keeps track of different experiment sets. This interface contains all possible parameters required to submit the simulation to the predefined computer clusters. Users can also visualize the status of running simulations. As a result of running the simulation described in Table 1, the electronic band-structures are calculated for different momentum-points, and the corresponding eigenfunctions are sampled for all atom locations and written to the disk.

When these files need to be examined and compared, the scientist enters the analysis and interpretation steps. NemoViz allows the scientist to use Plotly plugins to visualize the bands as traditional line plots and plugins represent the eigenvalues as three-dimensional iso-surfaces.

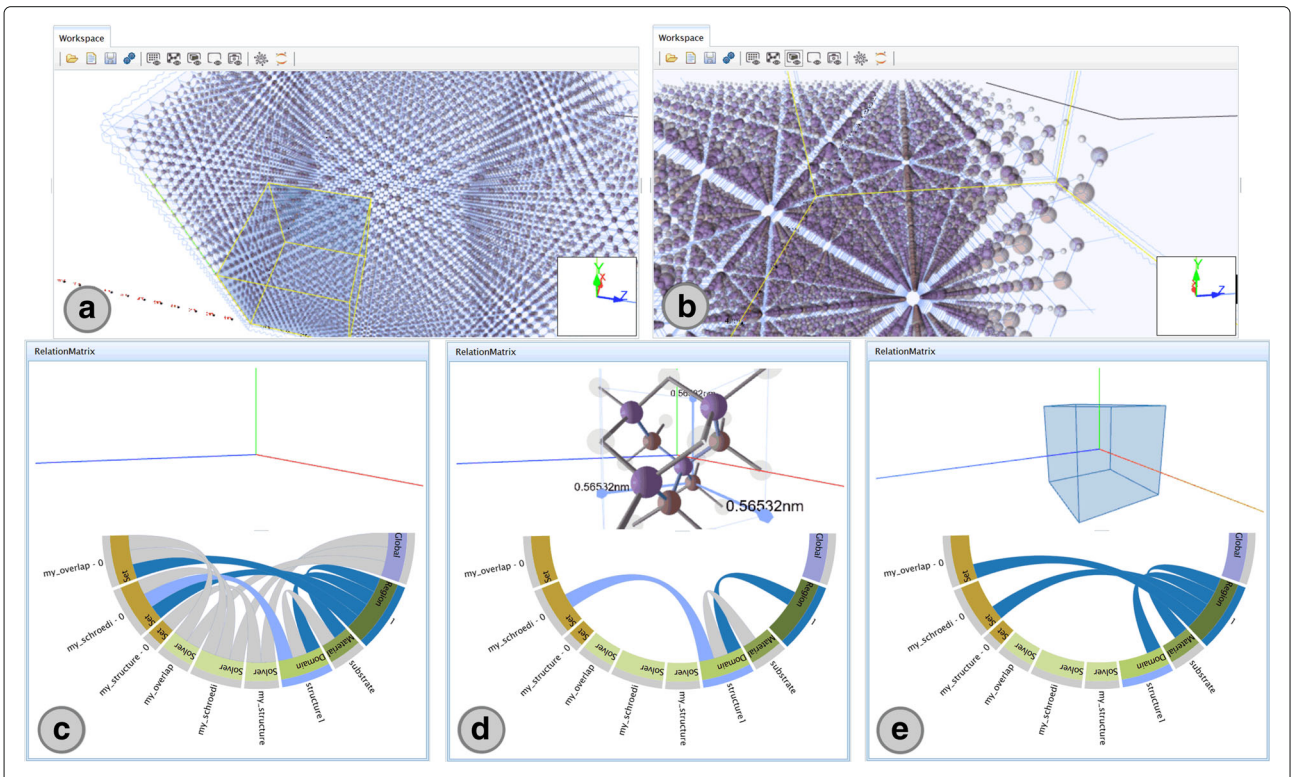

Fig. 8 NemoViz representations of a GaAs cuboid example. **a** main view of geometrical regions and atomistic structures. **b** main view of the simulation's active atoms. **c** relations view of all identified block relations. **d** relations view showing of domain structure relations. **e** relations view of region 1

Figure 9 shows two plugin instances that visualize the eigenvalues of two different energy levels. Each visualization represents the different trajectories a particle can follow in the confined box. These visualizations help the scientist identify important energy levels that match the experimental results.

Finally, the scientist proceeds to the Dissemination phase. The inputdecks and plugins can be parametrized and exported as Jupyter notebooks that describe scientific workflows. Notebooks define the parameters required to replicate and reproduce the results as well as options to change some parameters predefined by the scientist. These details can be shared with other scientists to replicate the findings. The notebooks can be expanded to include additional documentation and support different documentation formats, including html, pdf, latex, images, and videos.

NemoViz also supports these documentations by allowing scientists to generate high quality images (see Fig. 10). Figure 10a image was generated after loading a NemoViz three-dimensional model into the Blender software to render the scene. This particular image didn't require any modification of the Blender system before the rendering process; however more advanced visualizations, such as those reported by the Nemo Group on Blue Waters annual reports, require additional pre-/post-processing (Fig. 10b).

This case has shown how NemoViz can be used as an interactive visualization system to support the simulation design process, in particular the design of atomistic simulations. It is important to clarify why step 5 (Coding and Testing) was not mentioned in this section. Simulations do not always require the development of new models. Nemo 5 models are developed in C++ and Python. Various integrated development environment software (IDE) are available to develop the code. Nevertheless, NemoViz includes plugins that help users to debug new models. These plugins process NEMO5 output logs in order to extract and visualize information such as memory use, object lifetime, or time tracing.

Evaluation and discussion

The evaluation of new visualizations and visualization systems is a principal challenge of the visual analytics field (Thomas and Kielman 2009). So far, evaluation methods are not well established. However, accuracy, utility, and efficiency are the most accepted criteria (Zhu 2007) to evaluate a visualization solution. We conducted two types of assessments: 1) an informal evaluation test designed to assess the perceived usability of NemoViz; and 2) a quantitative evaluation in a controlled environment to assess NemoViz's effectiveness and efficiency of identifying simulation input deck errors.

Informal evaluation test

First, NemoViz was assessed through an informal evaluation. Previous research suggests that usability studies require groups between five and twenty users to have a

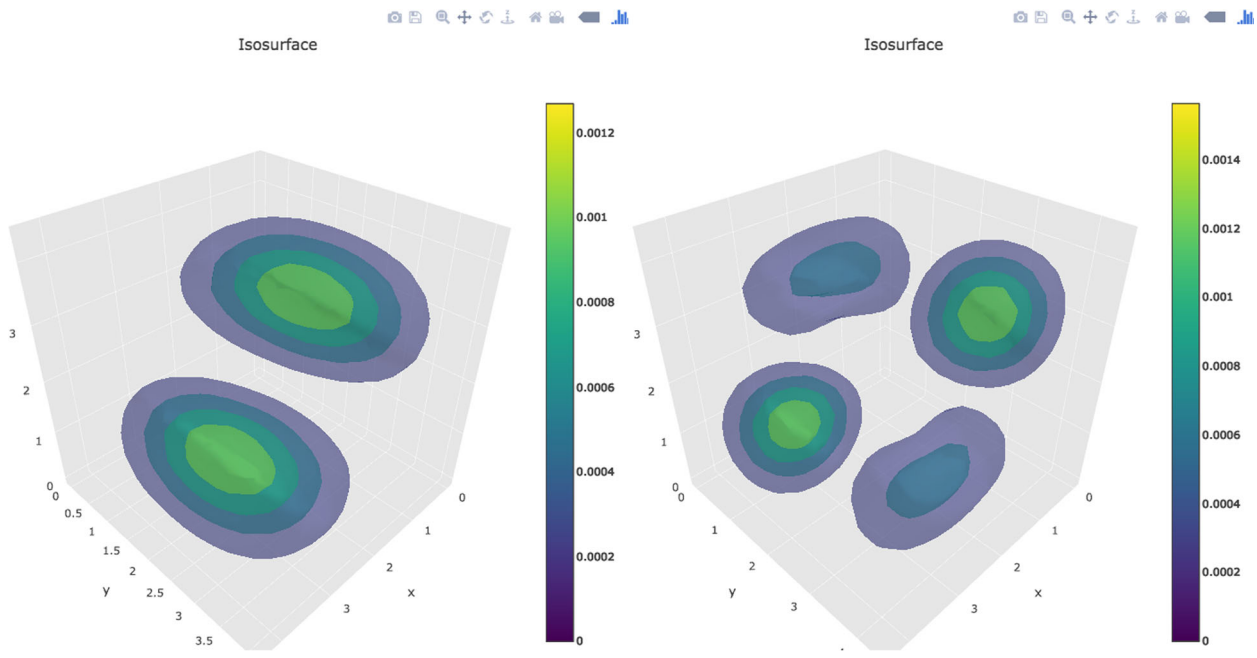

Fig. 9 3D Particle in a box simulation with electronic-wave function representations of two different energy levels and the corresponding orbital confinements

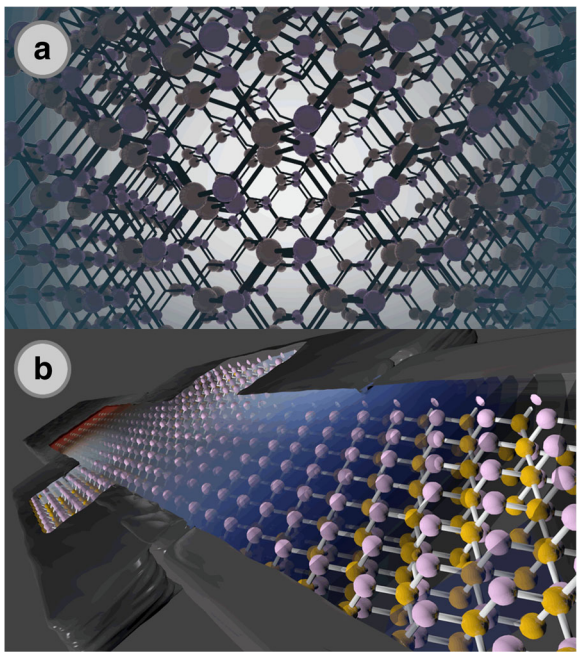

Fig. 10 Exported image after importing NemoViz scene into Blender. **a** Basic render, atoms are colored based on the CPK convention extended by Jmol. **b** Ultra Thin Body (UTB) transistor and the amount of electric potential along the transistor

Table 2 Percentages of participants that answer the questionnaire going from 1 (strongly agree) to 5 (strongly disagree)

To what extend to you agree with the following statements about NemoViz	1	2	3	4	5
NemoViz supports the user well in inspecting a NEMO5 inputdeck	100%	0%	0%	0%	0%
NemoViz structures and summarizes the atomistic representation of a model defined in a NEMO5 inputdeck	43%	57%	0%	0%	0%
NemoViz structures and summarize relations between different elements in a NEMO5 inputdeck	43%	43%	14%	0%	0%
NemoViz is easy to use and self-explaining	43%	43%	14%	0%	0%
NemoViz saves you time when modifying NEMO5 inputdecks	72%	14%	14%	0%	0%

good ratio of error detection (Macefield 2009). Users also has to be familiar with the environment that the tool is running, in this case been able to define simulations that involve atomistic structures. In order to fulfill this criteria, this evaluation was sent to 15 users of NEMO5, 3 with a background in Physics, 9 with a background in Electrical Engineering and 3 with backgrounds in other areas of engineering. All of them were familiar with NemoViz.

The test consisted of 5 questions designed to measure NemoViz's usability and capacity to save user's time, structure spatial information from an input-deck, and enable users to explore the inputdeck. The questions were designed based on work of (Beck et al. 2016) and modified for this NemoViz evaluation. The specific questions are presented in Table 2. Nine (9) NemoViz users answered the questions using a Likert-type scale from 1 (strongly agree) to 5 (strongly disagree). The questionnaire, created with the Qualtrics survey tool, was sent as an online form. Most of the participants (57%) were expert users of NEMO5, 29% were intermediate users, and 14% said they just started using NEMO5.

The results from the informal evaluation test showed that 100% (strongly agree + agree) of the participants agree that NemoViz provides support when exploring Nemo 5 inputdecks and captures the atomistic representation of the model defined in the NEMO5 inputdeck. The exploration of the inputdecks is directly related to the visualizations presented in the outline view of NemoViz. Some users also found the relation view useful to explore relationships among blocks. The representation of the atomistic structure of the simulation is mainly shown in the main view of NemoViz. Some characteristics of the structure were also located in the property and relation views.

Regarding the ability of NemoViz to summarize relations between inputdeck blocks, 86% of the participants perceive that NemoViz accurately represents these relations (Table 2). Only 14% of the participants responded that NemoViz does not add value to the structure and relations found in the text file of the inputdeck. Regarding the usability, 86% of participants reported that NemoViz was easy to use and intuitive. We hypothesize that elements such as its simple design, extensive use of visualizations, real-time synchronization of the multiple views, and widgets contribute to the usability. These elements were intentionally incorporated into the design of NemoViz for this purpose.

Finally, and most importantly, a very significant number of users (86%) reported that NemoViz helps them to save time when modifying the inputdecks. Modifying and running the input decks are the most commonly-performed tasks by users of any simulator tool given that a single modification of the input parameters is equivalent to a new experiment. The other 14% reported spending the same amount of time modifying the NEMO5 inputdecks using NemoViz or using a text editor. It is worth noting that 100% of the expert users reported a decrease in the time spent modifying inputdecks when using NemoViz.

Quantitative evaluation

The most common problem that users faced when modifying an inputdeck is the detection of errors. In other

words, users had to debug an inputdeck. In order to determine NemoViz's effectiveness and efficiency in debugging inputdecks, first we needed to identify which were the most relevant errors encountered by NEMO5 users.

The most relevant errors were identified based on the previous experience of current NEMO5 users. These previous experience measurements were designed as a survey to ask NEMO5 users about their own perception of debugging each type of error. Three different aspects were measured: frequency, complexity and difficulty. NEMO5 users answered the following questions using Likert-type scales: How often do you face this error? from 1 (never) to 4 (very often); How difficult is the process of fixing this error? from 1 (easy) to 4 (very difficult); and how long does it usually take you to solve it? from 1 (few minutes) to 3 (more than an hour). The most common input-deck errors were classified as follows: domain sizes are not correctly defined (structure dimension), domain positions are not correctly defined (structure position), geometrical regions are not properly defined (spatial definitions), crystal orientation is not well defined (crystal information), missing connections between solvers (solution-solution relation), missing connections between domains (domain-domain relation), missing relations between domains and solvers (solution-structure relation), missing relations between regions and solvers (solution-spatial relation), or inconsistency between the domains and geometrical regions (spatial-structure relation). We quantified each error type as well.

Fifteen (15) NEMO5 (not necessarily exposed to NemoViz) users answered this new survey following the same format as the first evaluation. The results for each aspect were normalized and the score values (between 0 and 1) were assigned (see Fig. 11). These results show that the most frequent errors were related to the structure definitions and relations. The most complex and frequent errors were related to crystal information. However, these latter errors were far less common. The results also show that errors involving relations between regions and solutions were easily solvable.

A measurement instrument was developed based on the error types with a high score on all aspects. Four error types were selected from the previous threshold: structure-structure relations (StrStrRel), spatial definitions (SDef), spatial-structure relation (SStrRel), and structure position (StrPos). However, based on the assumption that the most frequent errors have a larger impact, we decided to include the structure-dimension error (StrDim) in our evaluation. NemoViz expands the user's cognitive process in different ways and detecting the inputdeck errors can be defined as two cognitive processes: spatial representations that involve spatial cognition processes and well-defined mental models as well as abstract representations that require new mental models to represent the relations.

We validate the spatial representation errors by analyzing users' understanding of the main view, and the abstract representation by users' understanding of the relations view. We chose a widely used quantum transport calculation of a Silicon nanowire. For each error type, the text inputdecks were created, the corresponding errors introduced, and NemoViz visualization snapshots taken. Users

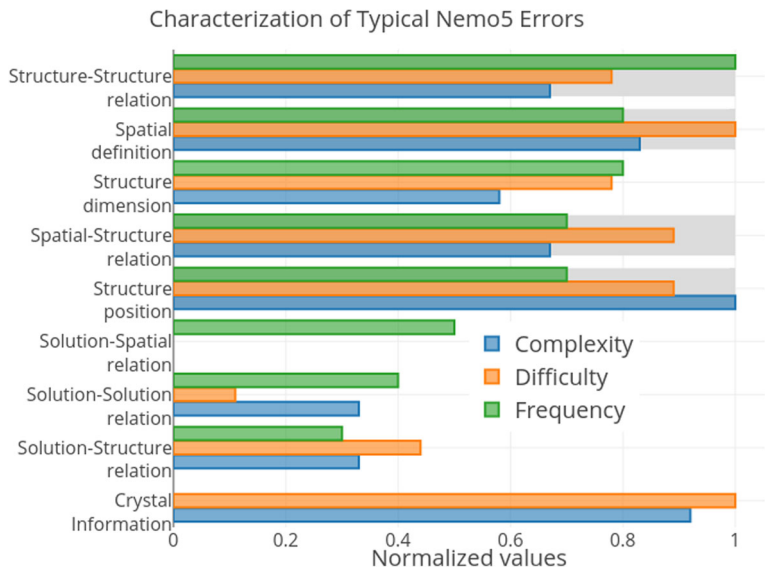

Fig. 11 Normalized scores for each type of error. The results are ordered by frequency. The gray areas denote scores of 0.6 and higher in complexity, difficulty and frequency

were asked to classify the correct type of error based on a visualization or segments of a text inputdeck. In order to avoid biased results, the users were asked to schedule an appointment in controlled space with a prepared desktop to answer the survey. Before the survey was taken, users had to take a small tutorial on how NemoViz represents inputdecks.

The time taken by each respondent to answer each survey question and the number of correct answers were measured. The effectiveness of users to detect errors using either the visualization or the text was calculated as the percentage of correct answers (accurately identified errors) over the total number of questions. The effectiveness using the visualization or text inputdecks was compared. Similarly, the efficiency of the visualization and text input deck was calculated as the time it took the user to correctly identify an error.

Results

In general, the users improved their effectiveness of error detection by 7% and were able to detect errors twice as fast using NemoViz compared to simply analyzing the text inputdecks. The left box plot of Fig. 12 shows the increased average accuracy. The standard deviation dramatically shrank to less than 10% while using NemoViz. The right box plot shows how the average time users took to detect errors was cut in half when interactive visualizations were used. The standard deviation was reduced even further.

We also analyzed data on users' expertise and question type. Non-expert users improved their detection times threefold when using NemoViz (bottom right of Fig. 13 shows detection times reduced three times in average). Expert users also showed a twofold improved detection time when using NemoViz (bottom left of Fig. 13 shows detection times reduced two times in average). The average times were calculated by taking the response times of all the NemoViz, or text inputdeck, questions of all 15 subjects and finding the arithmetic mean. Detection times for the structure-structure relations and structure dimension questions did not show any dramatic improvement when using visualizations. However, spatial structure-relations showed extraordinary improvements in time and efficiency (Top of Fig. 13).

In summary, the results presented above suggest that NemoViz enhances the cognitive process during error detection as follows: 1) it improves user efficiency and effectiveness in debugging NEMO5 inputdecks; and 2) it accelerates the learning curve of novice users by enhancing their effectiveness to the level of expert users.

Conclusion and future work

This work presents NemoViz as an interactive visual analytics system for NEMO5 inputdecks and shows evidence that interactive visualizations reduce the cognitive load of the simulation designing process. Complex simulation engines such as NEMO5 can be easily adopted by end users. The results of the evaluation tests suggest that interactive visualizations add value to end users by saving time and facilitating the cognitive processes associated with the understanding of spatial information and relations among different simulation entities. The results also suggest that during the

Fig. 12 Box plots of users' NemoVis effectiveness and efficiency score averages and quantiles. The blue boxes show the visual input results and orange boxes display the base input results

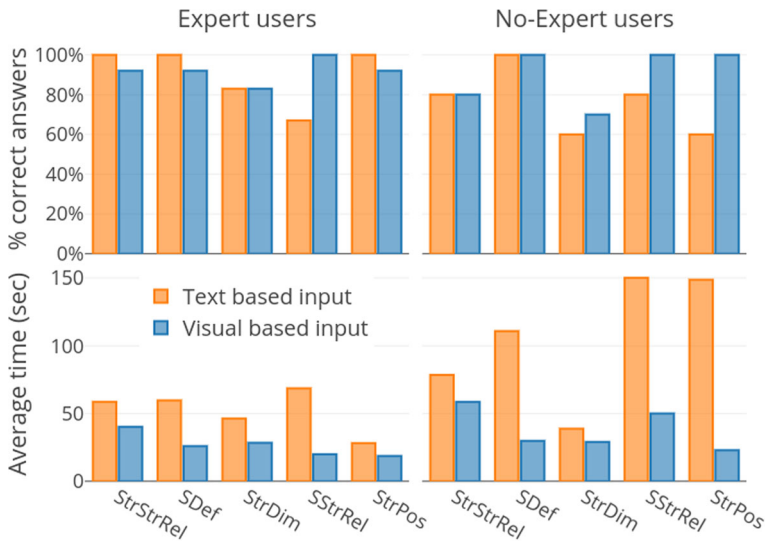

Fig. 13 Comparison of expert and non-expert users. Top: Average percentage of correct answers. Bottom: Average time to correctly detect errors

task of detecting errors in an inputdeck, non-expert users achieve a similar effectiveness as expert users if they use NemoViz.

NemoViz not only allows users to create, load, and visualize inputdecks in a graphical environment, but it also graphically represents relations between input blocks and creates three-dimensional representations of the spatial entities of blocks (e.g. atomistic structures with geometry descriptions). This work featured a simple case study to illustrate the power of NemoViz. More complex inputdecks e.g., the simulations of quantum transport in nanowires shown in Fig. 5, and complex three-dimensional devices shown in Fig. 4 are also supported by NemoViz. A simulation of quantum transport calculation on a Silicon nanowire was used as part of the evaluation design.

In this work we presented a framework for model development to identify critical functionalities in a research code and implement visualizations as aids to support researchers in the process of defining new simulations. This methodological procedures can be used in any research field, but particularly on fields where atomistic databases or atomistic structures are critical, such as nanoelectronics and molecular dynamics. An specific case of study showed how NemoViz, a visualization module, was incorporated as an additional module to Nemo5, a text-based research simulation code. A similar approach can be taken by research code developers in other fields to incorporate visualization aids in already existing simulations. The next step in the integration of NemoViz as a visualization aid for Nemo5 is to offer it as a free tool on Nanohub, and expect to expand its capabilities to support additional DFT research codes LAMMPS and QuantumEspresso.

Abbreviations

AJAX: Asynchronous Javascript and XML; DFT: Density functional theory; IDE: Integrated development environment; NEGF: Non-equilibrium green's function; NEMO5: NanoElectronics MOdeling tool fifth edition; TCAD: Technology computer-aided design; UTB: Ultra-thin-body transistor; WS: Web services

Acknowledgements

The authors wish to thank Professor Niklas Elmqvist of the University of Maryland and Professor David Ebert of Purdue University for their valuable discussion of this project.

Funding

Financial support from nanoHUB.org is gratefully acknowledged. The use of nanoHUB.org computational resources operated by the Network for Computational Nanotechnology funded by the US National Science Foundation under Grant Nos. EEC-0228390, EEC-1227110, EEC-0228390, EEC-0634750, OCI-0438246, OCI-0832623 and OCI-0721680 is gratefully acknowledged. NEMO5 developments were critically supported by an NSF Peta-Apps award OCI-0749140 and by Intel Corp.

Availability of data and materials

NemoViz is incorporated as an additional module of the NEMO5 toolkit and released under the NEMO5 Commercial/Academic license. NEMO5 licenses can be requested on-line (Beck et al. 2016).

Authors' contributions

DM developed the system and conducted the evaluations. DM also drafted the manuscript, analyzed the results, and completed the literature review. TK and GK were responsible for the critical revision of the manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Received: 24 April 2018 Accepted: 1 November 2018
Published online: 29 November 2018

References

Allen, M.P., & Tildesley, D.J. (2017). *Computer Simulation of Liquids*: Oxford University Press.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S. (2004). Kepler: An extensible system for design and execution of scientific workflows, In *16th International Conference on Scientific and Statistical Database Management* (pp. 423–424).

Atomistix ToolKit (ATK) (2008). Simulation software for nanoscience. http://quantumwise.com/. Accessed 06 May 2016.

Beck, F., Koch, S., Weiskopf, D. (2016). Visual Analysis and Dissemination of Scientific Literature Collections with SurVis. *IEEE Transactions on Visualization and Computer Graphics*, *22*(1), 180–189. https://doi.org/10.1109/TVCG.2015.2467757.

Bostock, M. (2012). D3.js - Data-Driven Documents. https://d3js.org/. Accessed 16 May 2016.

Callahan, S.P., Freire, J., Santos, E., Scheidegger, C.E., Silva, C.T., Vo, H.T. (2006). VisTrails: Visualization Meets Data Management, In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD '06*. https://doi.org/10.1145/1142473.1142574. Accessed 07 Jan 2016 (pp. 745–747). New York: ACM.

COMSOL Multiphysics® Modeling Software (2005). https://www.comsol.com/. Accessed 28 Apr 2016.

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., Laity, A., Jacob, J.C., Katz, D.S. (2005). Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. https://doi.org/10.1155/2005/128026. https://www.hindawi.com/journals/sp/2005/128026/cta/. Accessed 14 Sept 2017.

Dojo Toolkit (2008). http://dojotoolkit.org/. Accessed 16 May 2016.

Hmelo-Silver, C.E., & Azevedo, R. (2006). Understanding Complex Systems: Some Core Challenges. *Journal of the Learning Sciences*, *15*(1), 53–61.

Karlsson, B. (2005). *Beyond the C++ Standard Library: An Introduction to Boost*: Pearson Education. Google-Books-ID: IFfuYJ0OeZkC.

Klimeck, G., McLennan, M., Mannino, M., Korkusinski, M., Heitzinger, C., Kennell, R., Clark, S. (2006). NEMO 3-D and nanoHUB: Bridging Research and Education, In *Sixth IEEE Conference on Nanotechnology, 2006. IEEE-NANO 2006, vol 2*. https://doi.org/10.1109/NANO.2006.247682 (pp. 441–444).

Krüger, S. (1975). *Simulation: Grundlagen, Techniken, Anwendungen*. Berlin, New York: De Gruyter.

Lammpsfe (2012). https://www.scifes.com/index.php/product. Accessed 06 May 2016.

Lundstrom, M. (2015). Drift-diffusion and computational electronics - still going strong after 40 years!, In *2015 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*. https://doi.org/10.1109/SISPAD.2015.7292243 (pp. 1–3).

Lundstrom, M., & Klimeck, G. (2006). The NCN: Science, Simulation, and Cyber Services, In *2006 IEEE Conference on Emerging Technologies - Nanoelectronics*. https://doi.org/10.1109/NANOEL.2006.1609779 (pp. 496–500).

Macefield, R. (2009). How to specify the participant group size for usability studies: a practitioner's guide. *Journal of Usability Studies 5.1*, *5*(1), 34–45.

MAPS platform (2009). Current version is 4.0 | Scienomics. http://scienomics.com/products/molecular-modeling-platform. Accessed 06 May 2016.

McLennan, M., & Kennell, R. (2010). HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering. *Computing in Science Engineering*, *12*(2), 48–53. https://doi.org/10.1109/MCSE.2010.41.

Pedone, A. (2009). Properties Calculations of Silica-Based Glasses by Atomistic Simulations Techniques: A Review. *The Journal of Physical Chemistry C*, *113*(49), 20773–20784. https://doi.org/10.1021/jp9071263. Accessed 31 Mar 2017.

Qiao, W., Ebert, D.S., Entezari, A., Korkusinski, M., Klimeck, G. (2005). VolQD: direct volume rendering of multi-million atom quantum dot simulations, In *VIS 05. IEEE Visualization, 2005*. https://doi.org/10.1109/VISUAL.2005.1532811 (pp. 319–326).

Romanowska, I. (2015). So You Think You Can Model? A Guide to Building and Evaluating Archaeological Simulation Models of Dispersals. *Human Biology*, *87*(3), 169–192.

Scaife, M., & Rogers, Y. (1996). External cognition: how do graphical representations work? *International Journal of Human-Computer Studies*, *45*(2), 185–213. https://doi.org/10.1006/ijhc.1996.0048. Accessed 21 Apr 2016.

Sellier, J., Fonseca, J., Kubis, T.C., Povolotskyi, M., He, Y., Ilatikhameneh, H., breakJiang, Z., Kim, S., Mejia, D., Sengupta, P., et al (2012). Nemo5, a parallel, multiscale, multiphysics nanoelectronics modeling tool, In *Proc. SISPAD*. http://in4.iue.tuwien.ac.at/pdfs/sispad2012/P-18.pdf. Accessed 21 Apr 2016 (pp. 1–4).

Simulink Simulation and Model-Based Design (2012). http://www.mathworks.com/products/simulink/?requestedDomain=www.mathworks.com. Accessed 28 Apr 2016.

Suh, Y.K., Ryu, H., Kim, H., Cho, K.W. (2016). EDISON: A Web-Based HPC Simulation Execution Framework for Large-Scale Scientific Computing Software, In *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. https://doi.org/10.1109/CCGrid.2016.31 (pp. 608–612).

three.js - Javascript 3D library (2010). http://threejs.org/. Accessed 16 May 2016.

Thomas, J., & Kielman, J. (2009). Challenges for Visual Analytics. *Information Visualization*, *8*(4), 309–314. https://doi.org/10.1057/ivs.2009.26. Accessed 17 May 2016.

Wilensky, U., & Resnick, M. (1999). Thinking in Levels: A Dynamic Systems Approach to Making Sense of the World. *Journal of Science Education and Technology*, *8*(1), 3–19. https://doi.org/10.1023/A:1009421303064. Accessed 10 May 2016.

Zhao, L., Song, C.X., Kalyanam, R., Biehl, L., Campbell, R., Delgass, L., Kearney, D., Wan, W., Shin, J., Kim, I.L., Ellis, C. (2017). GABBs - Reusable Geospatial Data Analysis Building Blocks for Science Gateways, In *9th International Workshop on Science Gateways* (pp. 19–21).

Zhu, Y. (2007). Measuring effective data visualization. *Advances in Visual Computing*, 652–661. http://www.springerlink.com/index/H474878RR850R168.pdf. Accessed 21 July 2017.