PAPER

PhysiCell training apps: Cloud hosted open-source apps to learn cell-based simulation software

Aneequa Sundus[1,*], Furkan Kurtoglu[1,*], Kali Konstantinopoulos[1], Mary Chen[1], Drew Willis[1], Randy Heiland[1] and Paul Macklin[1,†]

[1]Indiana University Luddy School of Informatics, Computing and Engineering

[*]Contributed equally.
[†]Corresponding Author: macklin@iu.edu

Abstract

Cell-based tissue simulations require not only the ability to write new code in a simulation framework, but also an understanding of underlying mathematical models and parameters for each behavior of an agent. This entails a steep learning curve for interdisciplinary researchers joining computational biology research. We have created this suite of cloud-hosted open-source tools to separately explore key components of an agent-based cellular simulation framework. This creates an self-contained environment to learn and test functions of cells and micro-environment in modular fashion before creating more detailed, research-focused simulation models.

Key words: training; education; agent-based modeling; cloud hosted apps;open source, systems biology;cell based models

1 Introduction

Multicellular systems biology is an interdisciplinary field. It requires knowledge of basic biology and physics, combined with mathematical modeling, to convert that knowledge into a mathematical format. Next, those mathematical models are implemented in a variety of programming languages depending on the requirements of that model. Once these basic code bases are developed, new models can be developed—often without understanding all the knowledge and code behind it. Many system modeling researchers join the field from technically diverse backgrounds, ranging from biology and informatics to engineering and mathematics. While simulation software packages are essential tools for working work in systems biology and computational biology, they may present a steep learning curve for scientists entering the field from diverse backgrounds. There are discussions going on in academia to find better ways to train researchers in computational modeling field. Madamanchi et al. [1] focused on challenges for natural science students as they venture into the computational modeling world. They identified that most such students seek to use computational modeling to answer domain-specific research questions that cannot be answered in wet labs. It shows that many scientists with the minimum computational background are potential users of computational modeling and simulation tools. Madamanchi et al. [2] discussed the need for user-friendly tools in computational modeling training, along with the potential of use focused web applications ("microapps") for education and training. In this report, we demonstrate a suite of microapps to address the training needs of new multidisciplinary users of biological simulation software.

Agent-based simulation software (e.g., PhysiCell [3], Chaste [4], and CompuCell3D [5]) is widely used for creating cell-based simulations. (See Metzcar et al. [6] for a review of cell-based models.) These models can be more intuitive to understand than differential equation-based

Compiled on: June 24, 2022.
Draft manuscript prepared by the author.

models due to rule-based nature of assigning biological behavior to cell agents. Each cell of same cell type behaves independently even though they start at the same point with same rules; due to stochasticity introduced in code, they soon desynchronize and follow individual paths. Cell-based models generally simulate the tissue (micro)environment for these agents/cells by integrating diffusion solvers in the background, thus modeling the secretion, movement, and consumption of chemical substrates.

2 Statement of Need

Many complex simulation software packages used in academia are open source and are managed by small research teams. For such small teams, training new people outside their labs can become a bottleneck to dissemination and adoption. One common solution is to arrange workshops to train new people, such as PhysiCell's 2021 Virtual Woprkshop and Hackathon [7], recent CompuCell3D workshops [8], and the overall *Software Carpentry* approach [9]. There have been many efforts to create interactive materials for systems biology simulations. One earlier example of such training materials is *bugbox* [10]; this desktop-based software application must be downloaded and installed to explore different parameters of predator-prey models. In recent years there have been some efforts to create training materials more tailored to specific simulation software packages. In recent BrainIAK tutorials [11], the authors created learning materials for functional MRI analysis. They created Jupyter notebooks for users to run on their own to learn different concepts with open source data. They also created training materials around these notebooks to explain the concepts in fuller detail.

Although R is not frequently used to build simulation models, the Shiny package [12] can be used to transform R-based models into web applications for user-friendly sharing. For example, Handel [13] used R/Shiny to create an online, interactive systems immunology model. Their small, lightweight app allows users to explore different parameters in a model of infection and immune response. In "A virtual lab for teaching physiology" by David Granjon [14], created an R/shiny app to dynamically explore the effects of different genes on calcium and phosphate homeostasis in rats and humans.

Generally, there are fewer such platforms for transforming Python-, JavaScript-, and C++-based computational models to shareable web apps. Artistoo [15] is a framework based on JavaScript that lets users build interactive, explorable simulation models of cells and tissues in a web browser. They use the cellular Potts model [16] to simulate cells. Heiland et al. [17] developed *xml2jupyter* as a general framework to automatically create Jupypter-based graphical user interfaces (GUIs) for PhysiCell models built in C++, which can subsequently be shared as cloud-based web models on the nanoHub platform [18]. Similarly, CompuCell3D was recently adapted to hosting on nanoHUB [19]. Most of these apps provide specific examples for users to explore as a learning project. QUBES [20] also hosts a collection of training materials for qualitative biology. NetLogo[21] is another example for exploring mathematical models online by adjusting different parameters. It is more generalized and not tailored to biology.

Most of these apps do not explicitly link theoretical biological knowledge with programmable application. Instead, they generally assume that the user is already knowledgeable in those concepts or frameworks. Moreover, most of these frameworks are not physics-based (and instead restrict cells to non-realistic lattice positions with uniform size), while the cellular Potts-based models (e.g., CompuCell3D and Artistoo) are framed around interpreting biological behaviors as energy terms, which can be non-intuitive. There are unmet needs for online simulation frameworks where the cell behaviors are grounded in more realistic physics, cell behaviors can readily be associated with biologically meaningful parameters, and the models (and the underlying biology) can be taught through online interactive models that do not require software installs and additional coding.

3 Training Applications

We filled these unmet needs in training materials by creating a suite of small standalone apps that each focus on a different feature of cell-based models. A critical feature of our cloud-hosted interactive apps is that they are accompanied by training material to help the user understand the biological concept that each app presents. We have also designed our apps such that all parameters unrelated to a particular concept are fixed. This divide-and-conquer approach helps the user focus on the function of one parameter at a time. This way, we have divided basic modeling framework into seven apps. Students will train on each aspect in each app and then see how all of these concepts work in relation to each other in a final simulation.

This suite serves both to introduce key biological concepts of cell-based modeling while also specifically training users to run PhysiCell [3]: an open-source physics-based cell simulation framework. It provides key cell functions that can be used to write complex simulations of cells in their environment. PhysiCell provides support to run these simulations on desktops as well as on supercomputers, thus making it useful for multiple-scale simulations. The core code of Physicell is written in C++, with input files provided in XML and output files stored as XML and MATLAB files. Beyond the core framework is a growing software ecosystem, including packages like xml2jupyter [17] that that can help run the compiled models in a Python-based Jupyter Notebook. With this wide array of programming languages, file formats, and packages, PhysiCell can present a steep learning curve not only for students new to systems biology, but also those who are experienced with systems biology but unfamiliar with this specific software.

Our training apps are hosted on NanoHUB: an online platform to host simulations and other software tools [18]. We make our apps available for anyone to access and run. We selected seven apps because they capture the basic functionality of PhysiCell. While these introductory apps do not completely capture the breadth of functionality offered by PhysiCell, they are sufficient to train a user to create complex multicellular simulations.

Each training app has an graphical user interface (GUI), as depicted in Figure 1. In the `About` tab, we describe the biological and mathematical background of the app, along with some suggested parameter sets for exploration. This is followed by the `Config Basics`, `Microenvironment`, and `User Parameters` tabs. These three tabs provide different customization options to users, including parameters that can be modified to explore the given simulation. We have minimized the number of changeable parameters to focus on the selected modeling aspect for each app. Details about the theory and application of these apps are described below.

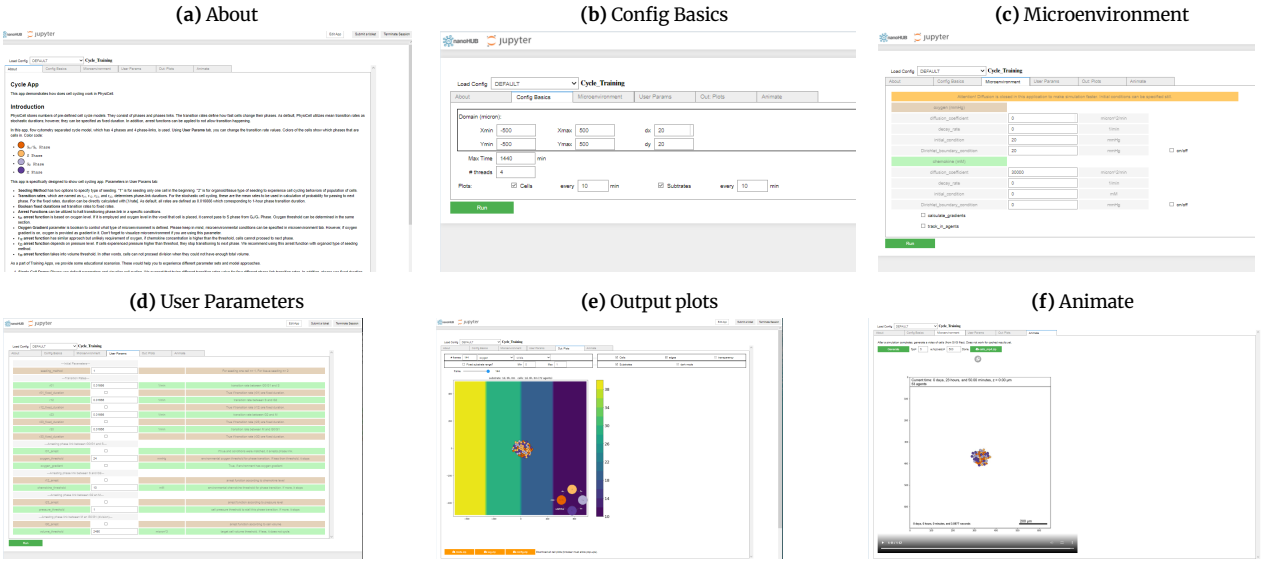

Figure 1. User Interface of each Training App (a) The `About` tab includes an introduction to the app and some suggested scenarios for the user to get familiar with app and its educational concepts. **(b)** The `Config Basics` tab is where the user can set the domain size of simulation and how often data is being stored and displayed. **(c)** The `Microenvironment` tab displays chemical substances present in the environment along with their diffusion and decay rates. Users can also set different boundary conditions for these substrates. **(d)** The `User Parameters` tab is the main focus point for most apps. Users can set parameters related to the module under discussion and then run the model by clicking on "Run" button. **(e)** In `Plots`, the user can observe the behavior of cells and substrate diffusion in the environment over time. **(f)** In `Animate` tab, users can make small animation of all outputs for rapid dynamic visualization.

3.1 Chemical Diffusion

PhysiCell [3] uses BioFVM [22] to simulate the chemical environment of cells. BioFVM [22] is a diffusion transport solver that allows substrate diffusion models to be run on modern desktop computers as well as supercomputers. In this app, we demonstrate the chemical environment using two substrates: oxygen and ChemicalA. Users can modify the diffusion and decay rates of each substrate to obtain different diffusion length scales ($\sqrt{D/\lambda}$). Users can also set the boundary conditions for each substrate conditions to either von Neumann (zero flux) boundary conditions or Dirichlet (fixed value) boundary conditions. Within the Dirichlet boundary parameters, there is an option to fine-tune the values for individual boundaries. Users can also add one Dirichlet node (a location where the substrate value is held fixed, similarly to a boundary condition) anywhere in the domain. Figure 2a shows the distribution of oxygen in the microenvironment after 6 hours and 30 minutes of simulated time, with Dirichlet conditions of 100 on the positive x-axis and 0 on the negative x-axis, and zero flux (von Neuman) conditions on the remaining boundaries. There is also a Dirichlet node present in the domain at (100,100) with a concentration of 100 on it. No chemical is being absorbed by any cell in the environment in this demonstration app, allowing users to focus solely on the impact of diffusion, decay, and boundary conditions. The Chemical Diffusion app is accessible at https://nanohub.org/resources/microenvnmtr.

3.2 Cell Division

Cell division is modeled in PhysiCell through different cell cycle models. Each cell cycle model consists of several phases (e.g., G_0/G_1, S, G_2, M), and cells transition from one phase to the next based on user-provided transition rates. These rates can be coupled with some external factors to control the cell cycle. PhysiCell also provides some arrest functions that can lock the cell in a particular phase. During the final stage, the cell divides into two daughter cells. In this app, we use PhysiCell's built-in `flow cytometry separated` cycle model to explore cell division. In the app, four factors control the four transition rates in the model. Users can control target concentrations of substrates, volume, and pressure to control transition from one phase to the next. Figure 2b shows the output of the cell cycle app, where cells cells are prevented from transitioning from G_0/G_1 to S if the local oxygen value is less than 25 mmHg. We can see that cells left half of the domain with oxygen concentration higher than 25 mmHg keep going through the cell cycle phases and dividing, while cells on the right are arrested in G_0/G_1. Also notice that the cells present on left side are of same type, yet they are not synchronously progressing through the cycle transitions. This is due to stochastic behavior introduced in the code to observe more realistic behavior. It is important to note that diffusion is not enabled for this app to demonstrate this phenomenon. The Cell Division app is accessible at https://nanohub.org/resources/trcycle.

3.3 Cell Death

PhysiCell supports two models of cell death: apoptosis and necrosis. Apoptosis is programmed cell death, which contributes to regular growth and function. In contrast, necrosis is un-programmed cell death due to injury or disease. The apoptosis model has only 1 phase, called Apoptotic. The apoptosis model does not involve any growth in cell size, though cells do shrink afterward. The necrosis model has 2 phases: the swelling phase and the lysed phase. The cell enters the swelling phase and grows in size until it reaches a specific volume, after which it lyses (bursts). After lysis, the cell shrinks (and potentially calcifies), entering the lysed phase. Figure 2c shows cell death with the apoptosis model. Necrotic brown cells are going through apoptosis. They swell before rupture. Green cells have not yet undergone apoptosis. This also shows the stochastic behavior in the model, since cells do not go through phases of apoptosis all at once. The Cell Death app is accessible at https://nanohub.org/resources/trdeath.

3.4 Cell Motility

Cell motility is active cell locomotion, either randomly or in response to some external stimulus (e.g., via chemotaxis). In PhysiCell (as well as many other cell-based models), motility is modeled via a biased random walk: a cell chooses a direction (either along a migration bias direction, randomly, or a combination based on its bias), move along that direction for a duration governed by its migration persistence time, and then choose a new direction of travel. In this app, users can control cell motility by setting the migration bias, migration speed, migration bias angle, and persistence time. We have displayed the each cell's path in this training app using a trail of a non-diffusing chemical substance in the microenvironment. At the end of the simulation, the effect of different parameters on cell movement can be observed by studying the cell paths. Figure 2d shows the path of three cells in the domain when they move toward a 30-degree x-axis gradient with a bias of 0.6. The Cell motility app is accessible at https://nanohub.org/resources/trmotility.

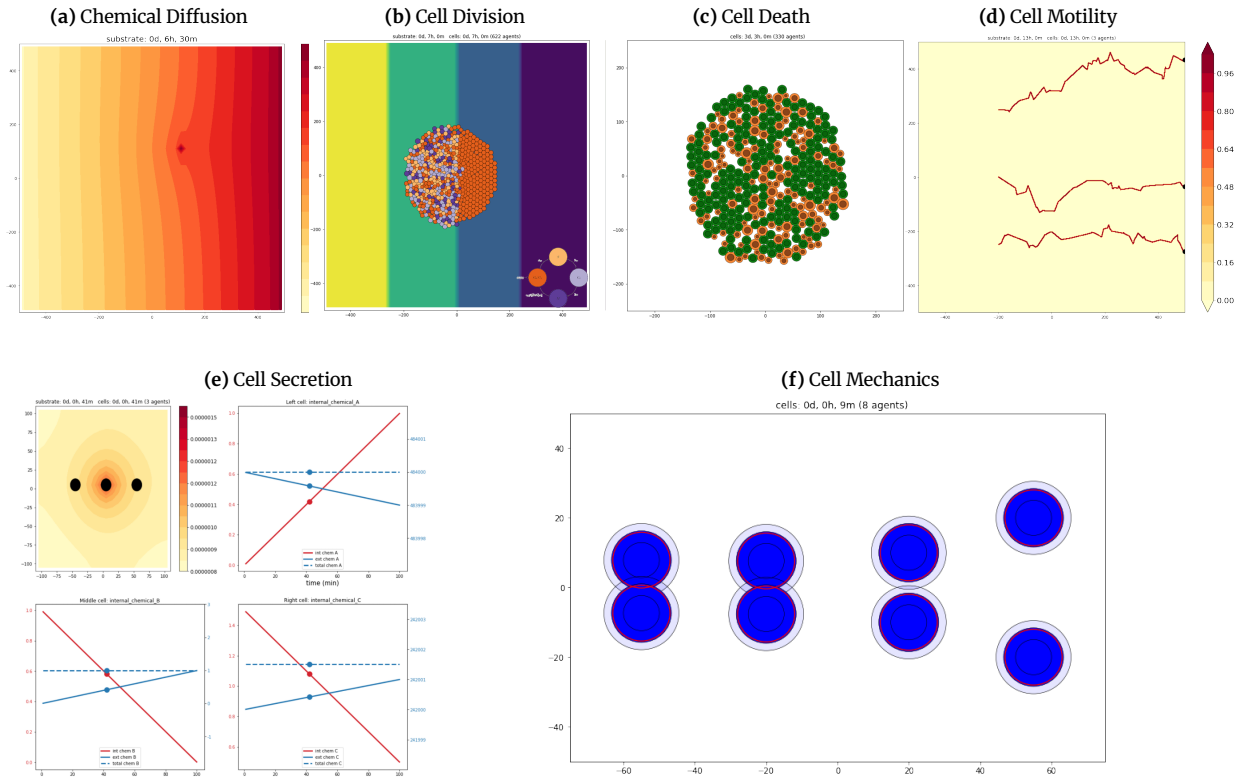

Figure 2. Typical Outputs for the Applications (a) Chemical Diffusion shows bands of chemical gradient formed by introducing Dirichlet boundary Conditions on the x-axis. **(b)** Cell Division displays cells going through different color coded-cell cycle phases according to their microenvironment **(c)** Cell Death shows an apoptosis cell death model introduced on a blob of cells **(d)** Cell Motility shows the biased random path of three motile cells **(e)** Cell Secretion displays three cells with their relative chemical secretion and absorption. Graphs display the amount of chemicals inside and outside the cell for each one **(f)** Cell Mechanics shows four pairs of cells placed at different distances resulting in different adhesive and repulsive forces for each pair.

3.5 Cell Secretion And Uptake

This app demonstrates the secretion and uptake functionality of cells in the PhysiCell environment. Cells remove chemicals from the microenvironment based on their uptake rates. Cells also secrete chemicals in the microenvironment. Mass conservation is guaranteed to keep the total sum of chemical within the cell and in the microenvironment constant. It is very challenging to visually show the amount of chemicals in cells and outside due to difference in scales of the amount present inside and outside of cells. We have solved this difficulty by separately plotting the amount of chemicals inside cells and outside in the environment. Figure 2e shows a sample output of the application. There are three cells in the simulation. The left cell uptakes Chemical A from the environment, the middle cell secretes Chemical B, and the right cell secretes Chemical C. Three graphs show the amount of chemicals present in each cell and the microenvironment. We can see that the total amount of chemicals remains constant due to mass conservation. Parameter values used for this output can be seen in scenario 2 of the app About page. The cell Secretion app is accessible at https://nanohub.org/resources/trsecretion.

3.6 Cell Mechanics

Cell mechanics in Physicell are achieved by simulating adhesive and repulsive forces, similarly to to the Leonard-Jones potential in atomic and particle physics. The range and strength of these forces ultimately create an equilibrium position for a cell at a certain distance from its neighbors. In the training app, we created four pair of cells seeded at different distances from each other. Users can modify the adhesive and repulsive forces as well as the maximum interaction distance of these forces. The interaction radius of each cell is displayed using a blue field around it, while the (parameter-dependent) equilibrium distance is denoted by red circles. Any cell pairs that are within their interaction

radii (i.e., their blue circles overlap) will pull each other towards the equilibrium spacing, while any cell pair beyond that interaction distance will not. When two cells have reached equilibrium, their red circles will just touch but not overlap. Notice that if any two cells are closer than their equilibrium spacing, they will push each other apart. Here in this snapshot 2f, after just 9 minutes of simulation two right pair of cells are pushing each other. The third pair from right is getting attracted while the leftmost pair is not moving since both are beyond the interaction distance. Cell Mechanics app is accessible at https://nanohub.org/resources/trmechanics.

3.7 Cell Volume

Cell volume in PhysiCell is used to capture liquid and solid fractions in the cytoplasm and cell nucleus. The cell will grow or shrink towards its steady-state (or "target") volume, based on its individual stored parameters; its current volume is stored as a state. In this app, we have used a red circle to denote the cell's steady state "target" size as determined by its parameters. Cells are color-coded according to a fluid fraction of cell cytoplasm and cell nucleus. Users can perturb the cell's state and watch it evolve back towards its steady state, or adjust parameters to change the steady state and thus direct the dynamics. The Cell Volume app is accessible at https://nanohub.org/resources/volumetr.

4 Discussion

Our goal for this work was to create user-friendly modules to train novice researchers on a cell-based modeling framework. We created this suite of seven open-source and made them broadly accessible with cloud hosting. Each app can be run independently, allowing users to select and focus on a single topic. We endeavored to make these apps faster by removing any functionality unrelated to the topic under discussion. Thus, these apps run fast, and we can see results almost instantly. We also provide a guide for the exploration at front page of all apps, directing users to explore and outlining what what they can expect from changing different parameters.

During the development of these apps, we worked in close collaboration with the principal developer of the software and the principal developer of user a interface for cloud-hosted apps. This close in-house collaboration resulted in improving not only training apps but also the underlying software. We identified the need in the software to quickly set Dirichlet conditions of each boundary, which was then implemented and added to the next release of PhysiCell. During development and testing of these apps, we found and reported bugs in PhysiCell's implementation of older function definitions, thus showing the integral role that the development of training software can play in quality control. We also worked with the developers of the Jupyter interface, which led to the addition of extra usability-focused features, such as adding separation tabs to separate different sets of parameters, adding an animate tab, and showing cell paths. We shared our initial versions of apps in our weekly lab meetings. Feedback from other lab members led us to explore some features in color template selection for apps. It is essential for applications like the cell cycle, where around half a dozen colors represent different stages in the cell cycle. Feedback from lab members led us to refine the usability and explore colorblind-friendly color schemes. We decided to go with color palettes described in [23] for these apps. This showed the power of integrating development of training applications and user interfaces directly with day-to-day lab operations.

We used these apps as teaching aids in an introductory class to systems biology modeling, and to introduce agent-based modeling to undergraduate researchers. These apps helped in translating theoretical concepts to tools that could drive hands-on exploration by students. These apps were also included as teaching materials for the PhysiCell workshop [7], and they helped accelerate the learning curve for participants in the one-week program. These apps were heavily used in training sessions about micro-environment set up and cell phenotype. All the participants of workshop were able to create their own PhysiCell models by the end of workshop. These apps can also be used in multiple settings, such as high school level biology classes to introduce concepts like diffusion, chemotaxis, and mechanical forces. We note that such apps could present an additional resource to include with documentation of other scientific software user guides, particularly for interdisciplinary fields where ideas form different disciplines intersect in one model.

Although these apps encompass the basic modules of a small cellular simulation, they cannot fully capture some of the more advanced features in PhysiCell that are used to write complex multi-scale simulations. While it is difficult to write narrowly-tailored training apps for these features, intermediate-level users (after having learned the essential features in this training suite) can continue learning from numerous intermediate-level and advanced projects available on nanoHUB. We recommend using cancer bio-robots [24] and PhysiCell invader-scout-attacker system simulation [25] on nanohub for intermediate-level exploration of capabilities of PhysiCell. Three-Type Multicellular Simulation Lab [26] shows a very detailed and customized simulation in which almost all the parts are available to customize. Thus, this provides an advanced-level application for exploration. Apps presented in this paper will serve as fundamental building blocks to move towards more complex simulations using PhysiCell platform.

5 Availability of source code and requirements (optional, if code is present)

- Project name: Physicell Training Apps
- Project home page: https://nanohub.org/members/221278/collections/physicell-training-apps
- Operating system(s): Platform independent
- Programming language: C++, Python,XML, MATLAB
- Other requirements:none
- License:BSD 3-Clause License

6 Declarations

6.1 Competing Interests

The authors declare that they have no competing interests.

6.2 Funding

We thank the Breast Cancer Research Foundation, Jayne Koskinas Ted Giovanis Foundation for Health and Policy, National Science Foundation (1818187 and 1720625), and National Cancer Institute (1U01CA232137-01) for generous support.

6.3 Author's Contributions

AS and FK led planning, development and refinement of the apps. KK, MC, and DW contributed to app development. RH provided technical support and refinement for Jupyter interfaces. AS, RK, and RH co-mentored undergraduate researchers. PM conceptualized and directed the work, obtained funding, and provided technical support for PhysiCell. AS led manuscript preparation, and all contributed to manuscript revision.

7 Acknowledgements

We want to acknowledge and thank fellow Ph.D. students; John Metzcar, Yafei Wang, post docs; Heber Rocha and Michael Getz: and past and present undergrad members of MathCancer lab for their feedback and suggestion during the development of these apps. Their feedback on earlier versions of these apps was key in improving these apps and shaping them into their present form.

References

1. Madamanchi A, Poindexter SV, Cardella ME, Glazier JA, Umulis DM. Qualitative Findings from Study of Interdisciplinary Education in Computational Modeling for Life Sciences Student Researchers from Emerging Research Institutions. In: 2018 IEEE Frontiers in Education Conference (FIE); 2018. p. 1–5.
2. Madamanchi A, Thomas M, Magana A, Heiland R, Macklin P. Supporting Computational Apprenticeship Through Educational and Software Infrastructure: A Case Study in a Mathematical Oncology Research Lab. PRIMUS 2021;0(0):1–22. https://doi.org/10.1080/10511970.2021.1881849.
3. Ghaffarizadeh A, Heiland R, Friedman SH, Mumenthaler SM, Macklin P. PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems. PLOS Computational Biology 2018;14(2):1–31. https://doi.org/10.1371/journal.pcbi.1005991.
4. Mirams GR, Arthurs CJ, Bernabeu MO, Bordas R, Cooper J, Corrias A, et al. Chaste: An Open Source C++ Library for Computational Physiology and Biology. PLOS Computational Biology 2013 03;9(3):1–8. https://doi.org/10.1371/journal.pcbi.1002970.
5. Swat MH, Thomas GL, Belmonte JM, Shirinifard A, Hmeljak D, Glazier JA. Chapter 13 - Multi-Scale Modeling of Tissues Using CompuCell3D. In: Asthagiri AR, Arkin AP, editors. Computational Methods in Cell Biology, vol. 110 of Methods in Cell Biology Academic Press; 2012.p. 325–366. https://www.sciencedirect.com/science/article/pii/B9780123884039000138.
6. Metzcar J, Wang Y, Heiland R, Macklin P. A Review of Cell-Based Computational Modeling in Cancer Biology. JCO Clinical Cancer Informatics 2019;(3):1–13. https://doi.org/10.1200/CCI.18.00069.
7. 2021 PhysiCell Workshop and Hackathon – PhysiCell;. http://physicell.org/ws2021/.
8. Workshop21 - CompuCell3D;. https://compucell3d.org/Workshop21.
9. G GW. Software Carpentry: lessons learned. F1000 Research 2016;3:62. https://doi.org/10.12688/f1000research.3-62.v2.
10. Ledder G. An Experimental Approach to Mathematical Modeling in Biology. PRIMUS 2008;18(1):119–138. https://doi.org/10.1080/10511970701753423.
11. Kumar M, Ellis CT, Lu Q, Zhang H, Capotă M, Willke TL, et al. BrainIAK tutorials: User-friendly learning materials for advanced fMRI analysis. PLOS Computational Biology 2020;16(1):1–12. https://doi.org/10.1371/journal.pcbi.1007549.
12. RStudio, Inc. Easy web applications in R.; 2013, uRL: http://www.rstudio.com/shiny/.
13. Handel A. A software package for immunologists to learn simulation modeling. BMC Immunology 2020 Jan;21(1):1. https://doi.org/10.1186/s12865-019-0321-0.
14. CaPO4 Teaching App;. https://dgranjon.shinyapps.io/entry_level/.
15. Wortel IMN, Textor J. Artistoo, a library to build, share, and explore simulations of cells and tissues in the web browser. eLife 2021 apr;10:e61288. https://doi.org/10.7554/eLife.61288.
16. Graner Fmc, Glazier JA. Simulation of biological cell sorting using a two-dimensional extended Potts model. Phys Rev Lett 1992 Sep;69:2013–2016. https://link.aps.org/doi/10.1103/PhysRevLett.69.2013.
17. Heiland R, Mishler D, Zhang T, Bower E, Macklin P. xml2jupyter: Mapping parameters between XML and Jupyter widgets. Journal of open source software 2019;4(39):10.21105/joss.01408. https://pubmed.ncbi.nlm.nih.gov/31342010, 31342010[pmid].
18. Madhavan K, Zentner L, Farnsworth V, Shivarajapura S, Zentner M, Denny N, et al. nanoHUB.org: cloud-based services for nanoscale modeling, simulation, and education. Nanotechnology Reviews 2013;2(1):107–117. https://doi.org/10.1515/ntrev-2012-0043.
19. Shaikh A, CompuCell3D; 2018. https://nanohub.org/tools/compucell3d.
20. Donovan S, Eaton CD, Gower ST, Jenkins KP, LaMar MD, Poli D, et al. QUBES: a community focused on supporting teaching and learning in quantitative biology. Letters in Biomathematics 2015;2(1):46–55. https://lettersinbiomath.journals.publicknowledgeproject.org/index.php/lib/article/view/45.
21. NetLogo Web;. https://www.netlogoweb.org.
22. Ghaffarizadeh A, Friedman SH, Macklin P. BioFVM: an efficient, parallelized diffusive transport solver for 3-D biological simulations. Bioinformatics 2016;32:1256 – 1258.
23. ColorBrewer: Updates;. http://www.personal.psu.edu/cab38/ColorBrewer/ColorBrewer_updates.html.
24. Heiland R, Macklin P, PhysiCell cancer biorobots simulation; 2019. https://nanohub.org/tools/pc4cancerbots.
25. Macklin P, Heiland R, PhysiCell: invader-scout-attacker system; 2019. https://nanohub.org/tools/pcisa.
26. Macklin P, Heiland R, Three-Type Multicellular Simulation Lab; 2020. http://nanohub.org/tools/pc3types.