Dr. Hana Dobrovolny's Lab

Viral Agent-Based Model Interface
Vision and Scope

Version 1.0

Revision History

Date	Version	Description	Author
9/26/2025	1.0	Filled out all the sections initially	Norwood, Ellion

Table of Contents

Vision

1. Introduction

1.1 Background

Graduate students at TCU are currently using a complex agent-based model originally developed by a former student in CUDA C to study virus-mediated cell interactions. At present, running the model requires users to edit parameter files directly in code and execute the simulation through the command line, which makes it difficult for undergraduates and non-experts with limited coding experience to use. Researchers would like to make the tool more accessible for teaching and undergraduate research by providing a graphical user interface (GUI). This interface would allow users to adjust parameters through sliders and buttons, run simulations more easily, and visualize outputs such as virus spread, infected cells, and cell fusion over time.

The model itself has evolved from its original version focused on viral diffusion and cell infection to include additional biological processes, with ongoing plans to expand to more phenomena such as immune response and directed motion. Because the research group expects continued student involvement and code updates, they need the GUI to be built in a way that is well-documented, readable, and easy to extend. The product is also intended to streamline workflows by eliminating manual edits to header/parameter files, allowing researchers to save selected outputs, and simplifying access to GPU-enabled workstations where the simulations run.

1.2 References

[1] Client kickoff meeting recording, 2025-09-26.

[2] Email: required GUI features/parameters, 2025-09-26.

[3] CUDA/C++ ABM codebase (Baylor + current TCU fork), internal repository.

2. Business Requirements

2.1 Business Opportunity/Problem Statement

The problem of	*the current viral agent-based model can only be run by directly editing parameter header files and executing CUDA/C++ code from the terminal, which is inaccessible for students or researchers without significant coding background*
affects	*undergraduate students, new graduate students, and other researchers who want to use the model for class projects, research, or exploration but lack the technical expertise to modify and run the code.*
the impact of which is	*limited accessibility, a steep learning curve, and inefficient workflows. This prevents broader adoption of the model for teaching, reduces its value as a training and research tool, and slows progress in experimenting with new biological scenarios.*
a successful solution would be	*A graphical user interface (GUI) that allows users to:* *Set the number of cells and initial conditions (infected, eclipse, virus amount).* *Toggle cell-to-cell vs. cell-free transmission.* *Adjust key simulation parameters (infection rate, transmission probability, clearance rate, eclipse and infectious durations, distribution parameters).* *Display and select plots of uninfected, eclipse, infectious, dead cells, and virus over time.* *Save time-series and optional spatial data with configurable filenames.* *This solution would make the model accessible to non-coders, improve usability for classroom and research purposes, and create a maintainable platform for future extensions (e.g., immune responses, directed motion).*

2.2 Business Objectives

BO-1: Increase accessibility of the viral agent-based model by providing a GUI that enables parameter input and simulation control without editing code, available within 6 months of release.

- Scale: Count of model runs performed by minimal-programming users.
- Meter: Lab usage logs and feedback surveys.
- Past: 0-2 (no GUI available).
- Goal: ≥ 10 successful non-programmer runs within first semester.
- Stretch: ≥ 5 runs across multiple courses/research groups.

BO-2: Enable reproducible and configurable simulation output by supporting named file exports for both time-series and optional spatial data within initial release.

- Scale: Successful saving of simulation runs.
- Meter: Presence of output files and confirmation by test users.
- Goal: 100% of runs save correctly when requested.

BO-3: Support visualization of core model outcomes (cell and virus populations over time) directly in the GUI within 6 months of release.

- Scale: Availability of plots for uninfected, eclipse, infectious, dead cells, and virus.
- Meter: Functional GUI plotting tested against known model outputs.
- Goal: ≥2 plot types selectable and displayed correctly.

BO-4: Improve adoption of the model as a teaching and research tool by lowering technical barriers.

- Scale: Number of students/researcher runs using GUI instead of terminal runs.
- Meter: Instructor/researcher reports and surveys.
- Goal: At least 5 runs per graduate student using the GUI instead of the terminal.

2.3 Success Metrics

SM-1: At least 70% of previous users or new undergraduate/graduate users are able to run simulations and adjust parameters successfully using the GUI with no direct code editing, within 6 months following release.

SM-2: Users report ≥ 1.0 increase (on a 5-point scale) in perceived usability of the model compared to code-only access, measured in surveys 3 months after release.

SM-3: The GUI supports correct saving of time-series data in 100% of test cases, and spatial data when enabled, verified by project team testing.

SM-4: At least 2 types of line graphs (such as uninfected, eclipse, infectious, dead cells, virus) are selectable and display correctly in ≥ 90% of test runs within 6 months.

2.4 Vision Statement

2.5

For	undergraduate and graduate students in biology, physics, and computer science
Who	need to run and analyze viral infection simulations without modifying complex CUDA and C code
The	Viral Agent-Based Model GUI is a graphical interface application
That	allows users to adjust simulation parameters, toggle transmission modes, set initial conditions, and visualize results in real time.
Unlike	the current system, which requires direct code modification and terminal execution,
Our product	provides an accessible, intuitive, and extensible interface that lowers the technical barrier for new users while remaining flexible enough for advanced research applications.

2.6 Business Risks

Low User Adoption: Undergraduate students may still find the system intimidating or too complex if the GUI is not intuitive. (Probability = 0.4, Impact = 7)
Mitigation: Use simple, consistent design (sliders, toggles, graphs), and conduct usability testing with target users.

Performance Limitations: GUI integration may slow down simulations or introduce incompatibilities with GPU-accelerated code. (Probability = 0.3, Impact = 8)
Mitigation: Keep backend C/CUDA performance untouched; ensure GUI primarily handles input/output and visualization.

Integration Complexity: Wrapping CUDA C code for use with the GUI may be more difficult than anticipated, leading to delays. (Probability = 0.5, Impact = 6)
Mitigation: Work closely with professor and prior student developer; build modular wrappers with clear documentation.

Maintenance Risk – Future students may struggle to extend the GUI if the codebase is poorly documented or overly complex. (Probability = 0.6, Impact = 8)
Mitigation: Provide clean, well-commented code and documentation to enable graduate students to maintain and expand the tool.

Scope Creep – As new biological features (e.g., immune response, syncytial formation) are added to the ABM, stakeholders may request major GUI extensions beyond the project timeline. (Probability = 0.5, Impact = 7)
Mitigation: Focus on modular, extensible design; prioritize “must-have” features for initial release.

2.7 Business Assumptions and Dependencies

AS-1: Students and researchers will have access to Linux workstations with GPUs capable of running CUDA simulations.

AS-2: The system will be used locally, not as a cloud/web service.

AS-3: Undergraduate users will have at least basic familiarity with biological terminology (infection rate, eclipse phase, etc.), so explanations within the GUI can remain concise.

AS-4: Graduate students will be available to maintain and extend the system after project completion.

AS-5: Input/output file structures for the ABM (parameter files, simulation outputs) will remain consistent across future versions of the model.

DE-1: Availability of the existing CUDA C ABM codebase (Baylor's and current TCU version) to serve as the simulation backend.

DE-2: GPU-enabled workstations maintained by the client's lab (e.g., Quadro 4000 machines) remain operational and accessible to student users.

DE-3: Python plotting libraries (e.g., Matplotlib) or equivalent tools will be available for graph generation.

DE-4: Continued support from the original developer (Baylor graduate) and current client for questions about model details.

DE-5: Future biological model expansions (immune response, directed motion) will be documented so that GUI parameters can be added with minimal friction.

3. Stakeholder Profiles and User Descriptions

3.1 Stakeholder Profiles

Stakeholder	Major value or benefit from this product	Attitudes	Major features of interest	Constraints	End user or not?
Faculty Research Advisors	Improved productivity in supervising and guiding simulation experiments; Improved productivity by avoiding custom coding, and allows for their researchers for no coding knowledge to run simulation	Supportive if GUI is easy to use and extendable	Easily setting up parameter values and easy usability of the GUI	Limited time to learn overly complex tools. Need access to the machines that have the software.	Yes
Graduate Student Researchers	Ability to perform new tasks without coding (streamlined process); improved usability	Enthusiastic about the usability of the GUI in order to increase productivity of their research	Easily setting up the parameters for viral agent-based model and exporting the resulting data	Need at least some conceptual understanding of the viral agent-based model. Need access to the machines that have the software.	Yes
Undergraduate Students	Improved usability and learning experience; automation of manual setup tasks	Generally positive, need clear instructions	Simple and intuitive interface with clear inputs and clear understandings of data outputs	Need some understanding of what the model is doing and what parameters to change. Need access to the machines that have the software.	Yes

3.2 User Environment

Number of people involved in completing the task:
Typically, one user (graduate or undergraduate student) runs a simulation at a time. The number of simultaneous users is not expected to change significantly.

Task cycle and time allocation:
A typical task cycle involves:

- Setting up initial conditions and parameters (5–15 minutes depending on complexity).
- Running the simulation (seconds to several minutes depending on system size and settings).
- Reviewing and saving outputs (5–20 minutes depending on the depth of analysis).
  These times are expected to decrease with a more streamlined GUI.

Environmental constraints:
The working environment is primarily academic and research-based located at designated workstations in labs.

System platforms in use today:
Currently, users rely to run the simulations on local Linux workstations equipped with NVIDIA Quadro GPUs that support CUDA.

Future platforms:
Future development may include compatibility with high-performance computing clusters or cloud-based platforms to enable faster simulation runs and larger-scale experiments.

Other applications in use:
Users commonly use spreadsheet software (Excel, Google Sheets) for organizing output data, statistical analysis software (R, MATLAB, or Python libraries), and visualization tools for creating publication-quality figures.

Integration needs:
The GUI primarily needs to output data in common file formats (e.g., CSV, TXT) to ensure compatibility with external analysis tools.

3.3 Alternatives and Competition

Current Situation / No Competitors:
Currently, there are no known competitors; the main alternative perceived by stakeholders is to continue using existing code-based workflows without a dedicated GUI.

Continue Using Existing Code-Based Workflows:

- *Description*: Run the viral agent-based model entirely through scripts without a graphical interface.
- *Strengths*:
    - No additional development required.
    - Maximum flexibility for technically skilled users.
- *Weaknesses*:
    - Steep learning curve for less experienced users.
    - Higher risk of coding errors and inefficiency.
    - Slower for repetitive experiments or classroom use.

4. Scope and Limitations

4.1 Product Perspective

Our product serves as a self-contained front-end system that provides a user-friendly environment for interacting with existing Viral Agent-Based Model (ABM) simulation engines. While the core ABM is implemented in C++ and accelerated by CUDA, with Python scripts for data analysis, the proposed product will act as a graphical shell around this back-end code.

The interface does **not** replace the simulation code; instead, it communicates directly with it by allowing:

- Parameter adjustments
- Simulation controls
- Visualization of outputs

This approach eliminates the need for users to modify the source code, lowering the barrier of entry and allowing larger groups of people—such as undergraduate researchers, students, and non-programmer users—to explore infection dynamics.

Since the ABM will continue to evolve with additional biological complexity, the GUI must be modular and extensible, supporting:

- Easy integration of new parameters
- Addition of switches
- New visualization modes

External interfaces include:

- The simulation engine written in C++ and CUDA for computation
- Analysis scripts for data formatting and visualization
- File system access for saving time-series data, spatial snapshots, and user-defined filenames

4.2 Major Features / Scope

FE-1: Configure initial simulation state

FE-2: Set and control model parameters and transmission modes

FE-3: Manage simulation execution

FE-4: Create, visualize, and export time series results

FE-5: Manage spatial data output

FE ID	Priority	Benefit	Effort	Risk
1	Must Have	High	Medium	Low
2	Must Have	High	Medium	Medium
3	Must Have	High	Medium	Low
4	Must Have	High	Medium	Low
5	Must Have	High	Medium	Low

4.3 Deployment Considerations

The system will be deployed within the TCU Astrophysics research lab environment, primarily on local Linux workstations equipped with NVIDIA Quadro GPUs that support CUDA. Users will include graduate students, undergraduate researchers, and instructors. Access is limited to a small group of users who will either run the program locally or access the lab machines via SSH. Time zone distribution is not a concern, as all users are

affiliated with TCU and will operate in the same location.

The GUI must integrate seamlessly with existing CUDA C code and handle parameter input/output through user-friendly sliders, buttons, and file save options. Infrastructure changes are minimal, as the required GPU-enabled workstations are already available, but the deployment will require a clean installation of CUDA, proper environment setup, and GPU drivers on the designated machines.

Training and process documentation are essential to ensure usability by students with limited coding experience, offering an overall user-friendly experience. A simple user manual and walkthrough will be developed to demonstrate how to (1) launch the GUI, (2) adjust parameters, (3) run simulations, and (4) save and interpret results. Additionally, deployment should account for possible future extensions, so that graduate students can add new parameters, buttons, or visualization features without full redesign.

5. Other Product Requirements

Standards and Platforms:

- Must run on Linux-based workstations with CUDA-capable GPUs.
- Written in a language compatible with C/CUDA backend (e.g., Python, C++, or Java for GUI).
- Should comply with TCU research lab security and access protocols.

Hardware and Performance:

- Requires GPU-enabled systems (e.g., NVIDIA Quadro 4000 or newer).
- Simulations should handle grids up to one million cells in a few minutes of runtime.
- GUI should support responsive interaction without noticeable lag when adjusting parameters or launching runs.

Quality Attributes:

- **Usability:** GUI must be intuitive for users with little or no coding experience.
- **Robustness & Fault Tolerance:** The interface should gracefully handle inputs (e.g., disallow out-of-range values).
- **Expandability**: Future parameters, processes, and visualization options must be easy to integrate.
- **Stability:** Once deployed, the system should reliably run on designated workstations with minimal troubleshooting.

Documentation:

- A concise user manual.
- Online help text/tooltips within the GUI explaining how to manipulate parameters.
- Developer notes describing the backend structure to help graduate students maintain and extend the system.

Constraints and Dependencies:

- Dependent on CUDA toolkit and GPU driver versions.
- GUI must interact with existing C/CUDA simulation code without major rewrites.
- Python may be required if additional visualization or analysis scripts are used.

Priorities:

- **High Priority:** Usability, performance on GPU workstations, expandability, and robustness.
- **Medium Priority:** Visualization features beyond basic line plots (e.g., movies, graphics).
- **Low Priority:** Cross-platform portability to non-Linux systems