USING REPAST AND JAVA TO ENHANCE THE NOM PROJECT: A REPORT

SUMMER 2003

DR. MADEY
RYAN KENNEDY
UNIVERSITY OF NOTRE DAME

Abstract

Natural organic matter (NOM) is a heterogeneous mixture of organic molecules. NOM is very important in many ecological processes, but is often tough and slow to study in nature. The NOM simulator is a stochastic model of agent-based NOM behavior in a specific environment. This web-based simulator allows scientists around the world to conduct research by enabling them to submit, run, and observe their custom simulations. Providing a graphical user interface (GUI) that allows one to view the molecules as they move and react is valuable to the scientists. The NOM GUI enables scientists to see colorful and dynamic results and also graphs that are updated as a simulation runs. This research paper tells how the GUI was created and provides insight for its operation.

Introduction

The NOM simulator has its roots in a software package called Swarm. A simple GUI had been created with Swarm, but had become unusable when the simulator was converted into Java. Once working in Java, integrating the RePast toolkit was straightforward and necessary to complete the new and improved GUI.

My strategy in creating the GUI was to familiarize myself with existing RePast GUIs and then to base the new GUI on them. Guidelines for the new GUI were discussed with a graduate student and a professor. Over the past few weeks, a powerful GUI has emerged that can be run on any system with Java and that effectively displays NOM interactions.

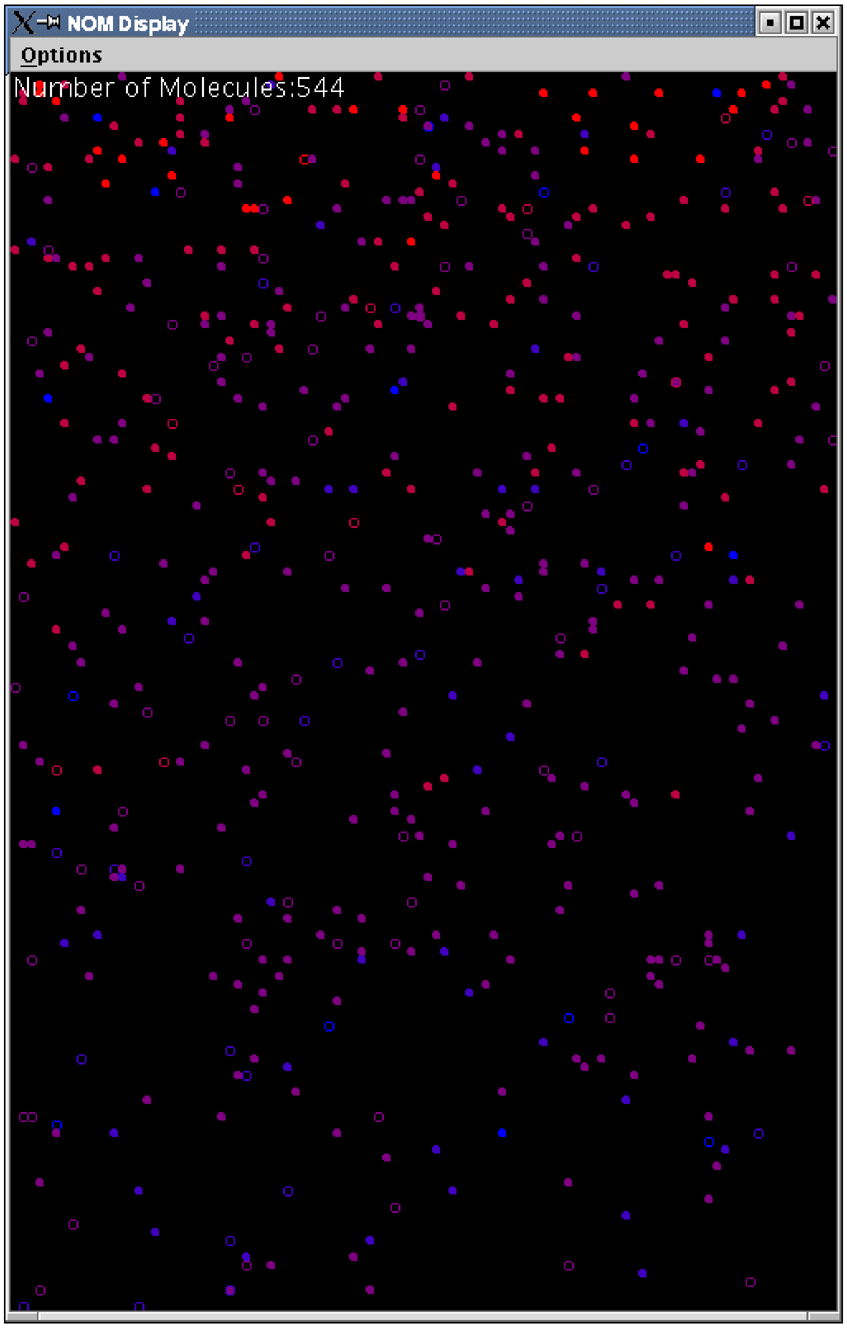
Methods

My work on the NOM project began with a restructuring and cleaning of the simulation code. This was done to make the integration of the RePast toolkit simpler and also to remove all ties to the Swarm package. Modifications to the random number generator used by the simulator were most tedious; however, switching from the Swarm method *getDoubleWith$withMax()* to the RePast method *getNextDoubleFromTo()* was very straightforward. Initial attempts to convert to a Java method to generate the next random number proved unsuccessful.

Once the code had been properly restructured, I began learning RePast. Reading the documentation on RePast's website was vital to my proper understanding of RePast. Also, carefully examining the provided demonstrations and source codes further helped me increase my grasp of RePast. After I had become fairly comfortable working with RePast, I began my work on the NOM GUI.

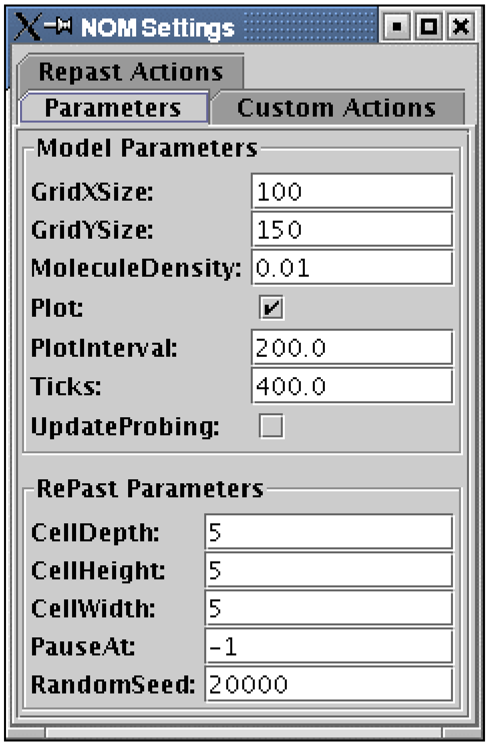
I started the GUI by closely following some of the provided source code for the HeatBugs and Life simulations. Little by little, I pieced together the necessary components to launch a primitive GUI. I implemented *SimModelImpl* to take advantage of some common display features. The molecules inhabit an *Object2DGrid* on an *Object2DDisplay,* titled *NOM Display*. In the code, this display is simply called *moleculeDisplay*. Basically, this means that the molecules move on a bounded grid. I implemented the molecules as rounded rectangles that are color-coded by molecular weight. Heavier molecules are more red and lighter ones are more blue. I wrote the color-coding mechanism to assign colors based on how many shades were needed and then generated the colors via their RGB value. This mechanism is dependent on the molecules being entered in the simulation in order of increasing molecular weight. The GUI is able to support shading for up to 256 molecule types. Molecules that are hollow are desorbed and solid ones are adsorbed. Adsorbed means that the molecules are stuck to a surface and desorbed means that they are freely moving. There is also a counter on the display called a *TextDisplay* that displays the total number of molecules in the system. Figure 1 shows the display for the molecules. A legend, Figure 3, is displayed along with the main display that shows the molecule name and molecular weight associated with each color.

Figure 1



Number of Molecules:544

Figure 2



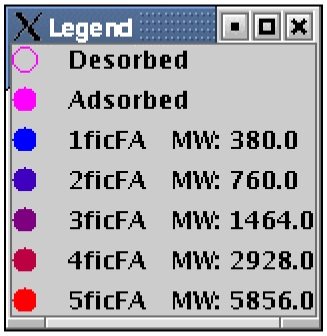
Model Parameters

GridXSize:	100
GridYSize:	150
MoleculeDensity:	0.01
Plot:	✓
PlotInterval:	200.0
Ticks:	400.0
UpdateProbing:	☐

RePast Parameters

CellDepth:	5
CellHeight:	5
CellWidth:	5
PauseAt:	-1
RandomSeed:	20000

Figure 3



○	Desorbed	
●	Adsorbed	
●	1ficFA	MW: 380.0
●	2ficFA	MW: 760.0
●	3ficFA	MW: 1464.0
●	4ficFA	MW: 2928.0
●	5ficFA	MW: 5856.0

When the GUI is loaded, there are many parameters that the user can modify in the *Parameters* tab of the *NOM Settings* window, Figure 2. Among them are molecule density, grid size, and whether to plot real-time results of the simulation. Enabling the plot feature is as simple as clicking a checkbox. When the model is initialized with the plot enabled, the display window, the legend, and the plot window, Figure 4**,** will all appear on the screen. If the plot is disabled, it will not display. It displays Molecular Weight vs. Number using datasets for adsorbed and desorbed molecules. The plot updates and rescales itself at the interval specified by the user in the *NOM Settings* window. The default is set to every 40 ticks (timesteps) because plot generation can be slow on some computers.

The *Object2DDisplay* that the molecules inhabit is also specified as a probeable surface. This means that an user can click on an agent (molecule) in the display to produce another window, Figure 5. This window displays specific information about that particular agent. I specified each window to display information such as the current coordinates of the molecule and its molecule identification number. By default, the probe windows do not update themselves automatically. The user can either click the checkbox before initializing the simulation in the *NOM Settings* window or enable it through the *RePast Actions* tab in the *NOM Settings* window.

Figure 4

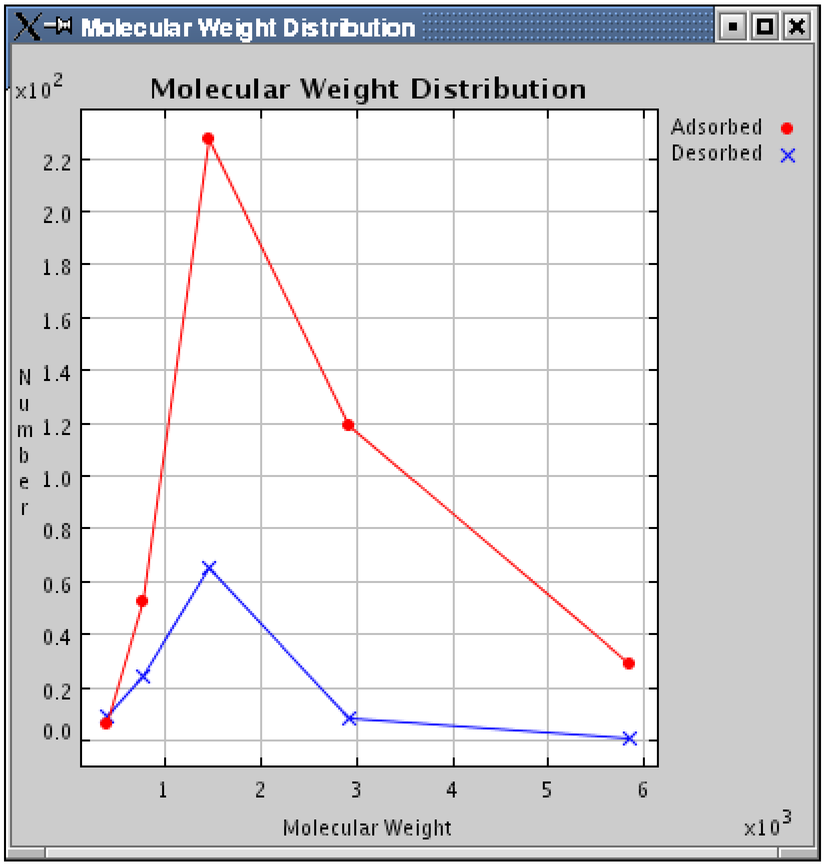
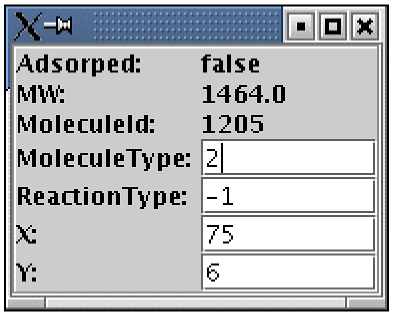


Figure 5



Adsorped:	false
MW:	1464.0
MoleculeId:	1205
MoleculeType:	2
ReactionType:	-1
X:	75
Y:	6

Running a Simulation

Running a simulation is very simple with the GUI. In the directory where the source files are compiled and stored, one can simply type *java StartMolecule 222 gui* to invoke simulation number 222. The *RePast Control Toolbar*, Figure 6, and the *NOM Settings* windows will be the first things you see. The simulation is loaded with parameters specific to simulation 222, but the user can edit a few of them in the *NOM Settings* window, Figure 2. Clicking the initialize button will display the molecule display window, *NOM Display*, and the legend. If the plot feature is enabled, the plot will be displayed too. Once initialized, the simulation can begin and will run for the specified number of ticks. Clicking the step button will run one timestep for every click. To change parameters for the simulation, the user must first stop the running simulation and then click the setup button to break down the current simulation and begin the framework for the next. After clicking setup, the user can initialize the next simulation. Clicking the red X exits the GUI. Running the simulator in batch mode is accomplished by typing *java StartMolecule 102 batch* from the same directory. Text will be displayed in the same window in which the simulator was invoked, Figure 7. This is the same display that is displayed when the simulator is run in GUI mode.

Figure 6

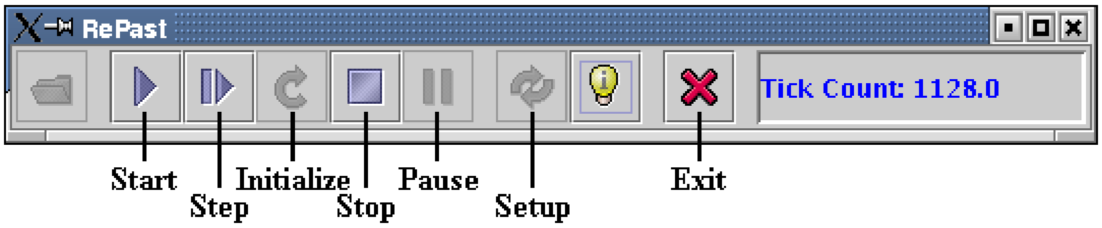
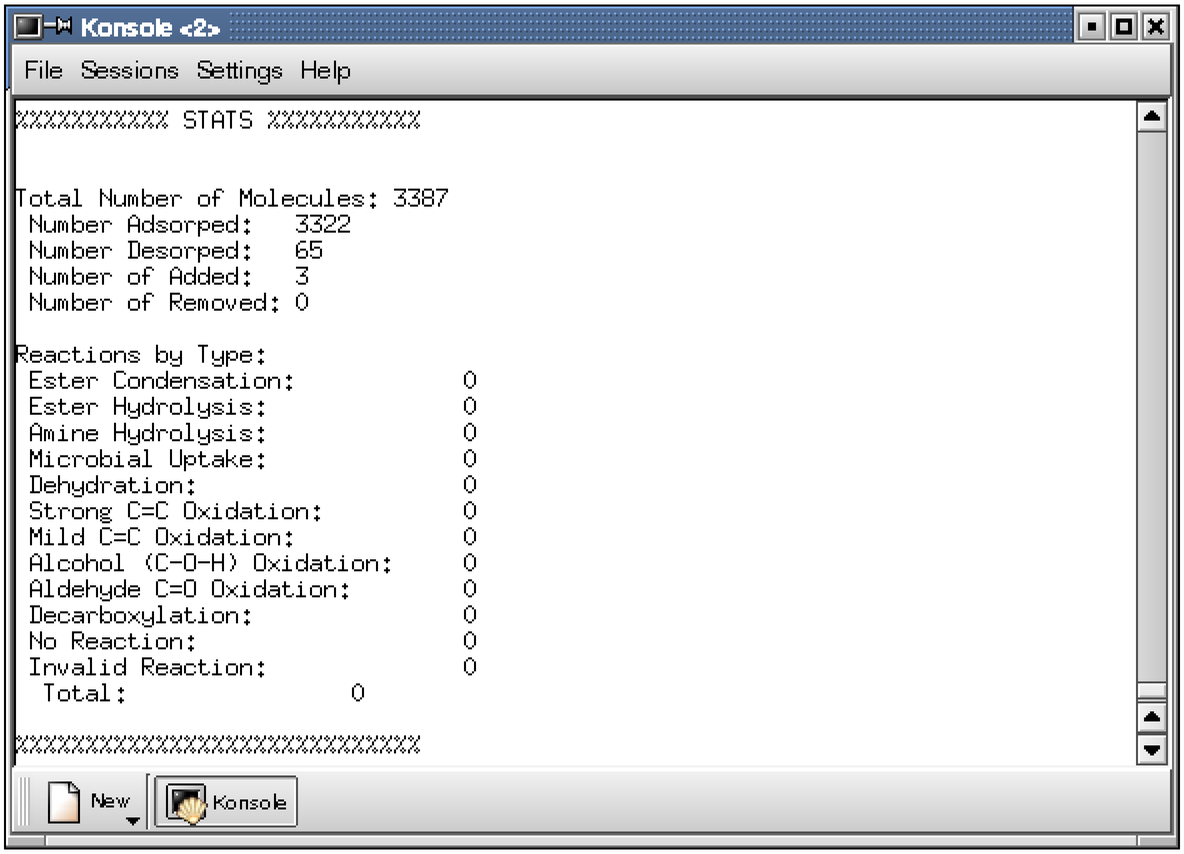


Figure 7



```
%%%%%%%%%%% STATS %%%%%%%%%%%


Total Number of Molecules: 3387
 Number Adsorped:   3322
 Number Desorped:   65
 Number of Added:   3
 Number of Removed: 0

Reactions by Type:
 Ester Condensation:          0
 Ester Hydrolysis:            0
 Amine Hydrolysis:            0
 Microbial Uptake:            0
 Dehydration:                 0
 Strong C=C Oxidation:        0
 Mild C=C Oxidation:          0
 Alcohol (C-O-H) Oxidation:   0
 Aldehyde C=O Oxidation:      0
 Decarboxylation:             0
 No Reaction:                 0
 Invalid Reaction:            0
  Total:              0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

WHAT YOU SEE

When a simulation is invoked, the *NOM Display* window will display the appropriate number of molecules for the simulation, listing the number of molecules in its upper left corner. The molecules will be color-coded according to their molecular weight, with the desorbed molecules being hollow and the adsorbed ones being solid. Desorbed molecules move freely through the system, from top to bottom, while being bounded on the left and right. As they proceed through the system, they are able to adsorb, or stick to a surface. Molecules with higher molecular weights are more likely to stay stuck once they stick, but are not as likely to become stuck as the lighter molecules. Molecules are also able to react. The reactions that take place are displayed in the window where the simulation was invoked.

Notable Methods Added

ModelSwarm.java

printStats()
zeroStats()

Molecule.java

draw(SimGraphics g)
getProbedProperties()

StartMolecule.java

begin()
buildDisplay()
buildLegend()
buildSchedule()
plotPoints()
populateNumAdsorbedX()
setup()
step()

Notable Methods Edited

ModelSwarm.java

buildModel()
move()
react()
sorption()

Molecule.java

move()
react()
sorption()

StartMolecule.java

main(String [] args)

Results

I consider my work this summer to be a success. Learning Java, RePast, and some Swarm was not easy, but was very rewarding. I did not encounter any major or unusual difficulties, but I did have my share of troubles. This is likely because of my lack of experience in the aforementioned environments. However, at summer's end, I had produced a robust GUI that suited the needs of the scientists and was a much better programmer.

Conclusion

Working with RePast was a very pleasant experience. It is very straightforward and powerful. In addition, it works on top of the already popular Java language. RePast was designed to work for agent-based simulations, which is highly evident in the plethora of tools it provides to simplify the creation of a simulation or GUI. Although I did not work much with Swarm, I consider RePast to be a better suited environment for agent-based simulations, especially for the web-based NOM simulator.

Future Work

Future additions to the GUI include integrating RePast's methods for *Object2DGrid* to find the nearest neighboring agents (molecules) and supporting more than 256 molecule types, while still allowing each type to have a distinct color. Additional plots could also be interesting additions, as well as enabling specific molecules to emit a trail as they work through the simulation. Lastly, modifying the simulation to properly color-code the molecules regardless of how they were entered into the simulation would be very beneficial.

REFERENCES AND TOOLS

Arthurs, Leilani.

Cabaniss, Steve. *White Papers: Modeling and Stochastic Simulation of NOM Reactions*. University of Notre Dame, 2002.

Deitel & Deitel. *Java: How to Program (Third Edition)*. Upper Saddle River, NJ: Prentice Hall, 1999.

Huang, Yingping. *Infrastructure, Query Optimization, Data Warehousing and Data Mining for Scientific Simulation*: Master's Thesis. University of Notre Dame, 2002.

Madey, Gregory R., Ph.D..

Maurice, Patricia A., Ph.D..

Xiang, Xiaorong. *Agent-Based Scientific Applications and Collaboration Using Java*: Master's Thesis. University of Notre Dame, 2003.

Java Version 1.4.1

RePast Version 2.0.1

http://java.sun.com/j2se/1.4.1/docs/api/

http://repast.sourceforge.net