Using High Performance Computing To Model Cellular Embryogenesis

Kison Osborne, Gerard Vanloo (Morehouse College)
Mentors: Chung Ng (Morehouse College), Kwai Wong, Ben Ramsey (University of Tennessee), Dali Wang (Oak Ridge National Laboratory)

Overview

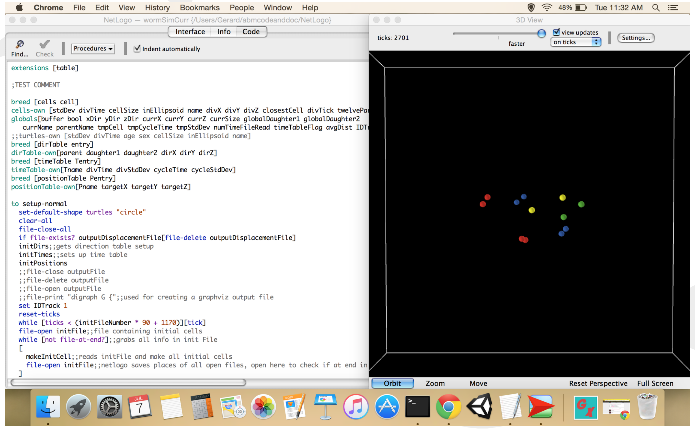
- Create a computer simulation of the *C. elegans*'s embryogenesis cycle to duplicate data from existing experimental data (collected from Memorial Sloan Kettering Cancer Center)
    - Port existing NetLogo simulation into RepastHPC[1]
    - Visually display results using VisIt[2]
- Create a graphical user interface (GUI) to run RepastHPC and VisIt from a single hub

C. Elegans

- Primitive multicellular organism (worm)
  - Shares many important biological characteristics that arise as complications within human beings[3]
- Used in development biology
  - Primarily for the study of cellular growth and organization in three dimensions
    - An abstract way of studying diseases such as cancer
- By using computer simulations, the same studies can be done without actually having to breed the worm

NetLogo

- NetLogo is an open-source, agent based modeling software written in Scala and Java
- Has two components
    - Interface/Code Window
    - Viewer
- Limitations
    - Can only execute in serial

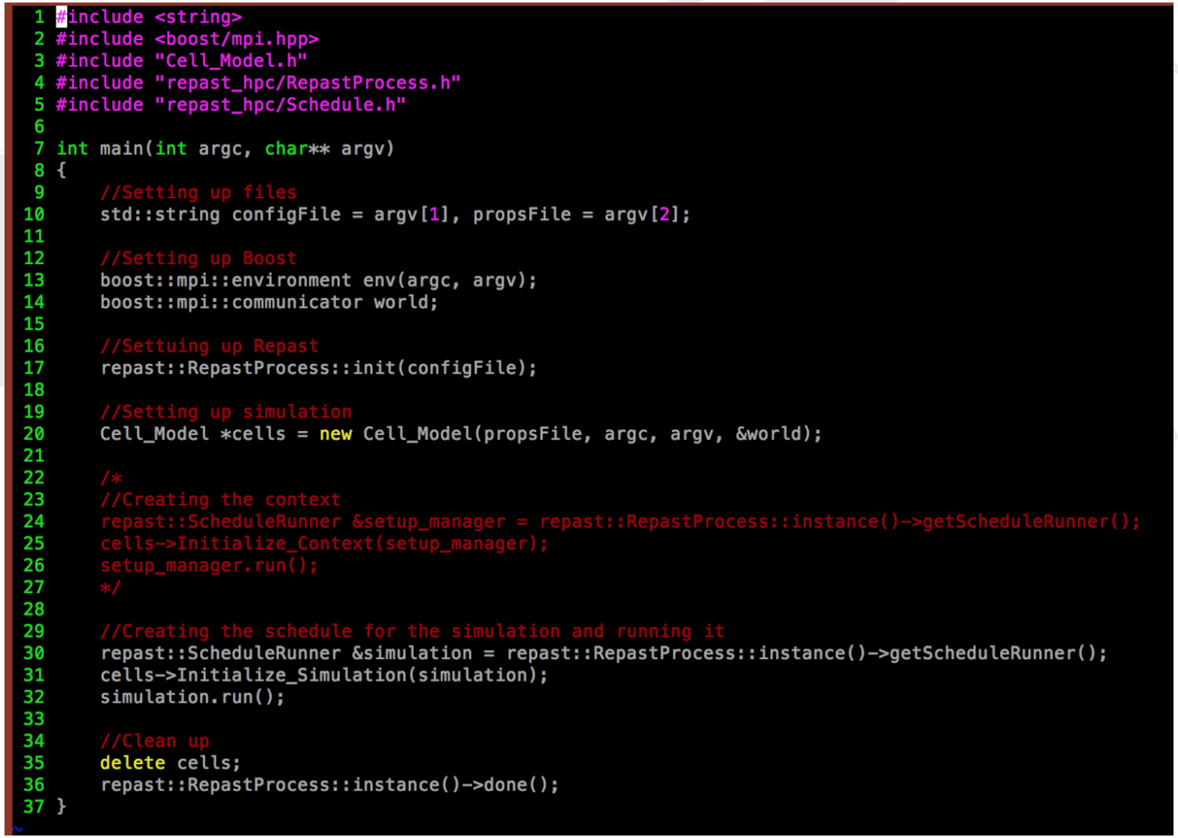


```
extensions [table]

;TEST COMMENT

breed [cells cell]
cells-own [stdDev divTime cellSize inEllipsoid name divX divY divZ closestCell divTick twelvePa
globals[buffer bool xDir yDir zDir currX currY currZ currSize globalDaughter1 globalDaughter2
   currName parentName tmpCell tmpCycleTime tmpStdDev numTimeFileRead timeTableFlag avgDist IDTr
;;turtles-own [stdDev divTime age sex cellSize inEllipsoid name]
breed [dirTable entry]
dirTable-own[parent daughter1 daughter2 dirX dirY dirZ]
breed [timeTable Tentry]
timeTable-own[Tname divTime divStdDev cycleTime cycleStdDev]
breed [positionTable Pentry]
positionTable-own[Pname targetX targetY targetZ]

to setup-normal
  set-default-shape turtles "circle"
  clear-all
  file-close-all
  if file-exists? outputDisplacementFile[file-delete outputDisplacementFile]
  initDirs;;gets direction table setup
  initTimes;;sets up time table
  initPositions
  ;;file-close outputFile
  ;;file-delete outputFile
  ;;file-open outputFile
  ;;file-print "digraph G {";;used for creating a graphviz output file
  set IDTrack 1
  reset-ticks
  while [ticks < (initFileNumber * 90 + 1170)][tick]
  file-open initFile;;file containing initial cells
  while [not file-at-end?];;grabs all info in init File
  [
    makeInitCell;;reads initFile and make all initial cells
    file-open initFile;;netlogo saves places of all open files, open here to check if at end in
  ]
```

RepastHPC

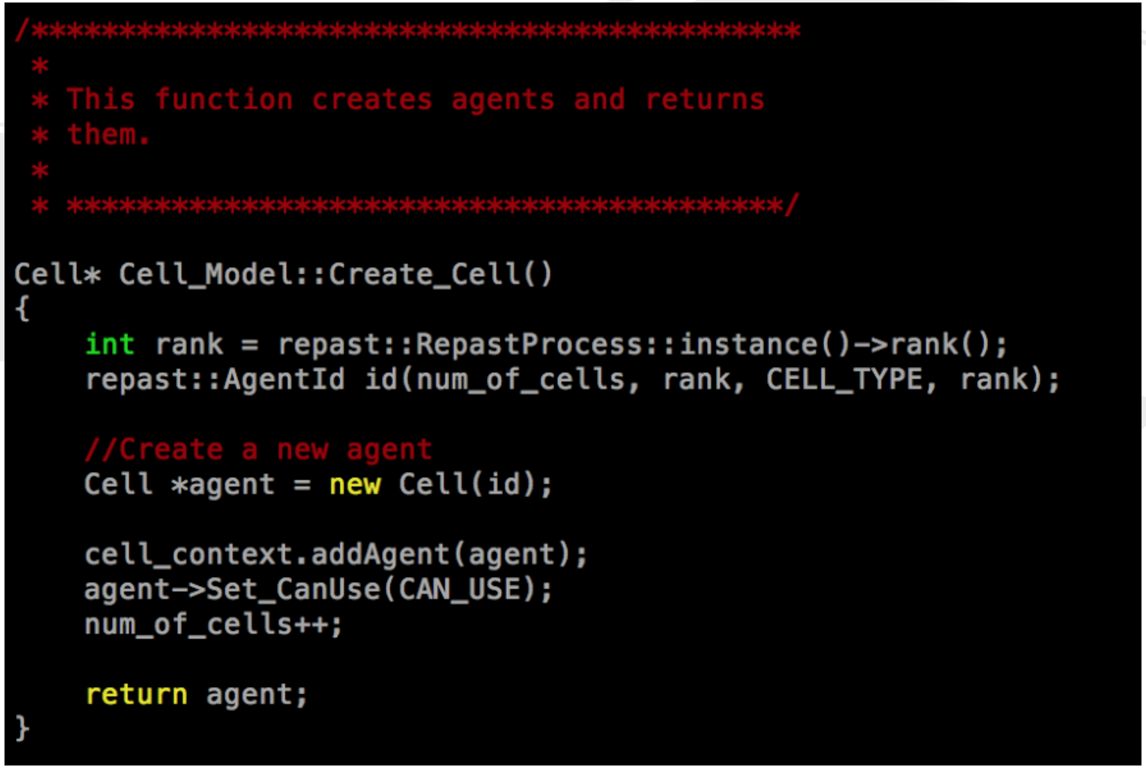
- RepastHPC is an open-source, cross-platform, agent based modeling toolkit written in C++
    - Released (v2.1.0) by Argonne National Laboratory on May 8, 2015
- Unlike NetLogo, it is created specifically for high performance computing (parallel programming).
    - Runs on clusters, supercomputers, and workstations
- Limitations
    - No visual features

Preparing the Simulation



```cpp
#include <string>
#include <boost/mpi.hpp>
#include "Cell_Model.h"
#include "repast_hpc/RepastProcess.h"
#include "repast_hpc/Schedule.h"

int main(int argc, char** argv)
{
    //Setting up files
    std::string configFile = argv[1], propsFile = argv[2];

    //Setting up Boost
    boost::mpi::environment env(argc, argv);
    boost::mpi::communicator world;

    //Settuing up Repast
    repast::RepastProcess::init(configFile);

    //Setting up simulation
    Cell_Model *cells = new Cell_Model(propsFile, argc, argv, &world);

    /*
    //Creating the context
    repast::ScheduleRunner &setup_manager = repast::RepastProcess::instance()->getScheduleRunner();
    cells->Initialize_Context(setup_manager);
    setup_manager.run();
    */

    //Creating the schedule for the simulation and running it
    repast::ScheduleRunner &simulation = repast::RepastProcess::instance()->getScheduleRunner();
    cells->Initialize_Simulation(simulation);
    simulation.run();

    //Clean up
    delete cells;
    repast::RepastProcess::instance()->done();
}
```

Creating Cells

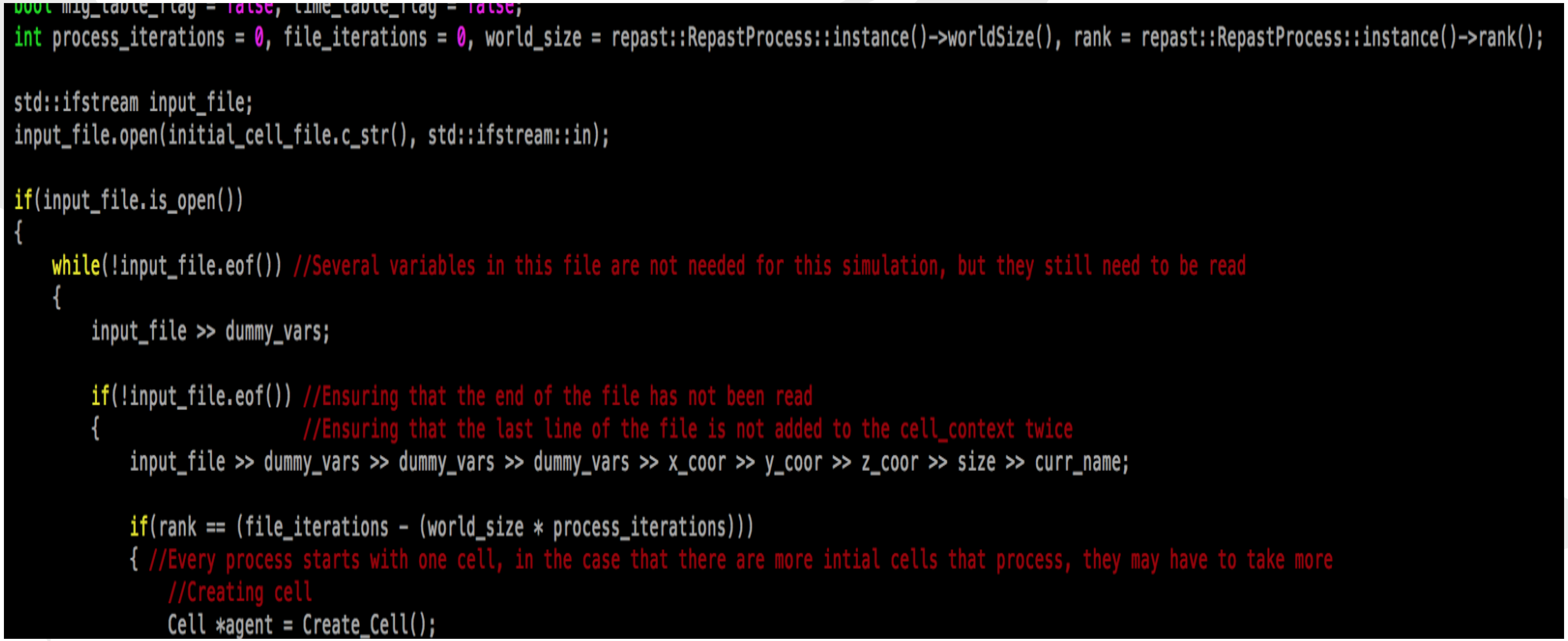


```
/********************************************
 *
 * This function creates agents and returns
 * them.
 *
 * *****************************************/

Cell* Cell_Model::Create_Cell()
{
    int rank = repast::RepastProcess::instance()->rank();
    repast::AgentId id(num_of_cells, rank, CELL_TYPE, rank);

    //Create a new agent
    Cell *agent = new Cell(id);

    cell_context.addAgent(agent);
    agent->Set_CanUse(CAN_USE);
    num_of_cells++;

    return agent;
}
```

Boosting With RepastHPC

- RepastHPC make implicit use of the Boost library
- Boost is a parallelization library created for C++
    - More specifically, it is a layer of abstraction over MPI
- In particular, Boost is used with the transfer of agents between processes
    - It does this by serializing an agent package—the actual object that is passed between the processes

Splitting the Work



```
bool mig_table_flag = false, time_table_flag = false;
int process_iterations = 0, file_iterations = 0, world_size = repast::RepastProcess::instance()->worldSize(), rank = repast::RepastProcess::instance()->rank();

std::ifstream input_file;
input_file.open(initial_cell_file.c_str(), std::ifstream::in);

if(input_file.is_open())
{
    while(!input_file.eof()) //Several variables in this file are not needed for this simulation, but they still need to be read
    {
        input_file >> dummy_vars;

        if(!input_file.eof()) //Ensuring that the end of the file has not been read
        {                     //Ensuring that the last line of the file is not added to the cell_context twice
            input_file >> dummy_vars >> dummy_vars >> dummy_vars >> x_coor >> y_coor >> z_coor >> size >> curr_name;

            if(rank == (file_iterations - (world_size * process_iterations)))
            { //Every process starts with one cell, in the case that there are more intial cells that process, they may have to take more
                //Creating cell
                Cell *agent = Create_Cell();
```

All cells are distributed between the processes

RepastHPC Agents in Parallel

- Agents can be requested, copied, and/or moved between processes
- In this simulation, RepastHPC automatically handles these processes due to the use of the spatial network (a grid)
  - More specifically does...
    - Moving agents across processes
    - Copying agents across processes
    - Cancelling of non local agents (copies)

Algorithms

- Wander
- Divide
- Save progress
    - X, Y, Z Coordinates
    - Name
    - Size
    - Various IDs

Wander Algorithm: Closer Look

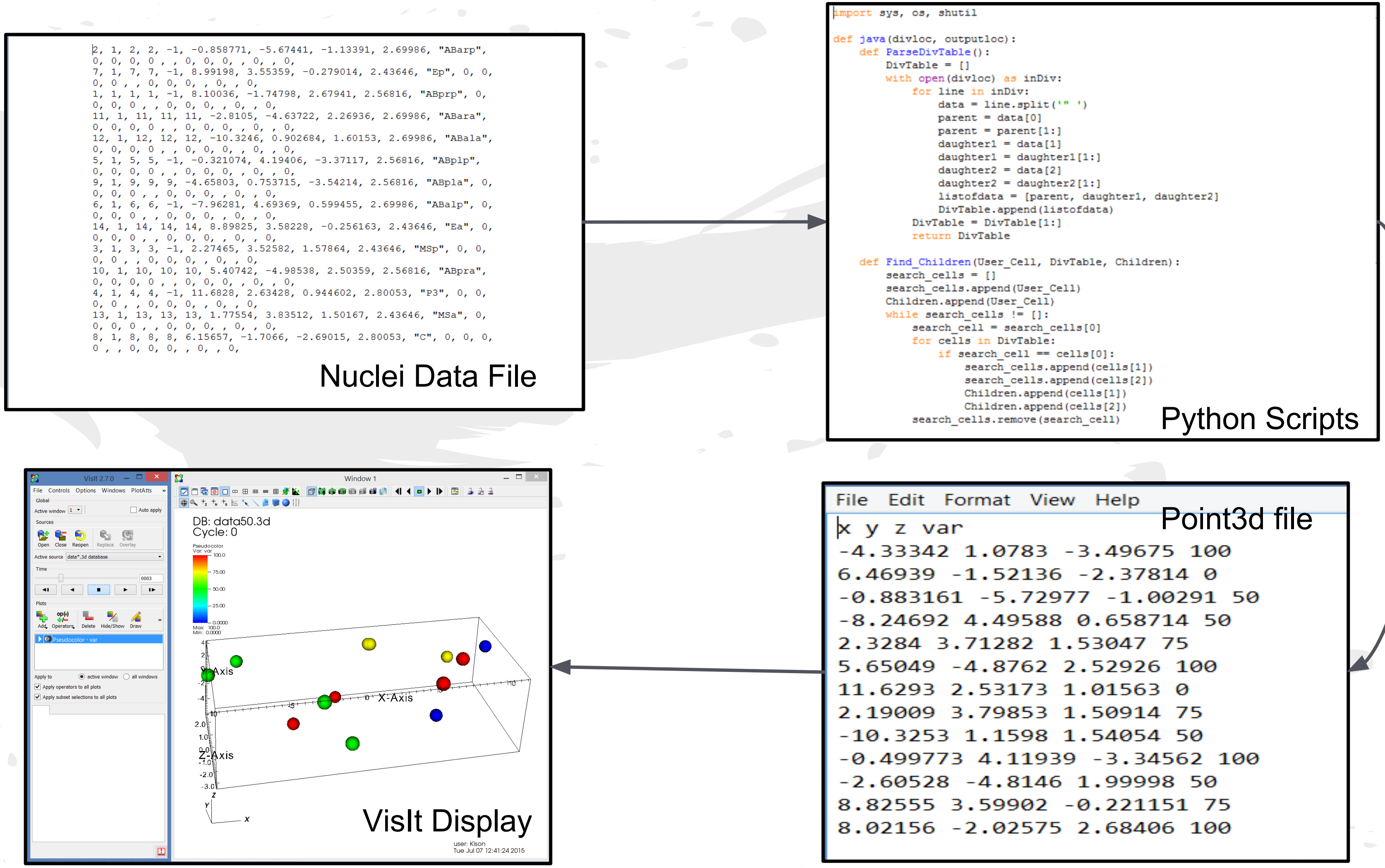
- Move in a linear path
  - Cells execute a different movement path after X amount of ticks have occurred during the simulation
  - Mainly, cells normally move in the following manner
    - Current Position + ((Target Position – Current Position) / (Division Time – Current Number of Ticks))

Division Algorithm: Closer Look

- If parent cell is ready to divide
  - Create new cell, which becomes first daughter, and check experimental data
    - Determine target coordinates
    - Determine division time
    - Determine the initial speed of the cell’s movement
  - Parent cell becomes daughter cell and check experimental data
    - Determine division time
    - Determine target coordinates
- If parent has outlived division cycle
  - Create a new cell; this becomes the first daughter
  - Parent cell becomes the second daughter
  - Both of the cells cannot divide

VisIt

- VisIt is an open-source software written in C/C++
  - Uses Python scripting for visualizations
  - Can utilize Java in addition to previous mentioned languages
- It is used for the visualization of data
  - Animation is simple, time-stepped pictures
- Like RepastHPC, it can also run in parallel allowing it to handle larger data sets



Nuclei Data File

```
2, 1, 2, 2, -1, -0.858771, -5.67441, -1.13391, 2.69986, "ABarp",
0, 0, 0, 0 , , 0, 0, 0, , 0, , 0,
7, 1, 7, 7, -1, 8.99198, 3.55359, -0.279014, 2.43646, "Ep", 0, 0,
0, 0 , , 0, 0, 0, , 0, , 0,
1, 1, 1, 1, -1, 8.10036, -1.74798, 2.67941, 2.56816, "ABprp", 0,
0, 0, 0 , , 0, 0, 0, , 0, , 0,
11, 1, 11, 11, 11, -2.8105, -4.63722, 2.26936, 2.69986, "ABara",
0, 0, 0, 0 , , 0, 0, 0, , 0, , 0,
12, 1, 12, 12, 12, -10.3246, 0.902684, 1.60153, 2.69986, "ABala",
0, 0, 0, 0 , , 0, 0, 0, , 0, , 0,
5, 1, 5, 5, -1, -0.321074, 4.19406, -3.37117, 2.56816, "ABplp",
0, 0, 0, 0 , , 0, 0, 0, , 0, , 0,
9, 1, 9, 9, 9, -4.65803, 0.753715, -3.54214, 2.56816, "ABpla", 0,
0, 0, 0 , , 0, 0, 0, , 0, , 0,
6, 1, 6, 6, -1, -7.96281, 4.69369, 0.599455, 2.69986, "ABalp", 0,
0, 0, 0 , , 0, 0, 0, , 0, , 0,
14, 1, 14, 14, 14, 8.89825, 3.58228, -0.256163, 2.43646, "Ea", 0,
0, 0, 0 , , 0, 0, 0, , 0, , 0,
3, 1, 3, 3, -1, 2.27465, 3.52582, 1.57864, 2.43646, "MSp", 0, 0,
0, 0 , , 0, 0, 0, , 0, , 0,
10, 1, 10, 10, 10, 5.40742, -4.98538, 2.50359, 2.56816, "ABpra",
0, 0, 0, 0 , , 0, 0, 0, , 0, , 0,
4, 1, 4, 4, -1, 11.6828, 2.63428, 0.944602, 2.80053, "P3", 0, 0,
0, 0 , , 0, 0, 0, , 0, , 0,
13, 1, 13, 13, 13, 1.77554, 3.83512, 1.50167, 2.43646, "MSa", 0,
0, 0, 0 , , 0, 0, 0, , 0, , 0,
8, 1, 8, 8, 8, 6.15657, -1.7066, -2.69015, 2.80053, "C", 0, 0, 0,
0 , , 0, 0, 0, , 0, , 0,
```

Python Scripts

```python
import sys, os, shutil

def java(divloc, outputloc):
    def ParseDivTable():
        DivTable = []
        with open(divloc) as inDiv:
            for line in inDiv:
                data = line.split('" ')
                parent = data[0]
                parent = parent[1:]
                daughter1 = data[1]
                daughter1 = daughter1[1:]
                daughter2 = data[2]
                daughter2 = daughter2[1:]
                listofdata = [parent, daughter1, daughter2]
                DivTable.append(listofdata)
            DivTable = DivTable[1:]
            return DivTable

    def Find_Children(User_Cell, DivTable, Children):
        search_cells = []
        search_cells.append(User_Cell)
        Children.append(User_Cell)
        while search_cells != []:
            search_cell = search_cells[0]
            for cells in DivTable:
                if search_cell == cells[0]:
                    search_cells.append(cells[1])
                    search_cells.append(cells[2])
                    Children.append(cells[1])
                    Children.append(cells[2])
            search_cells.remove(search_cell)
```

Point3d file

```
File  Edit  Format  View  Help
x y z var
-4.33342 1.0783 -3.49675 100
6.46939 -1.52136 -2.37814 0
-0.883161 -5.72977 -1.00291 50
-8.24692 4.49588 0.658714 50
2.3284 3.71282 1.53047 75
5.65049 -4.8762 2.52926 100
11.6293 2.53173 1.01563 0
2.19009 3.79853 1.50914 75
-10.3253 1.1598 1.54054 50
-0.499773 4.11939 -3.34562 100
-2.60528 -4.8146 1.99998 50
8.82555 3.59902 -0.221151 75
8.02156 -2.02575 2.68406 100
```

VisIt Display

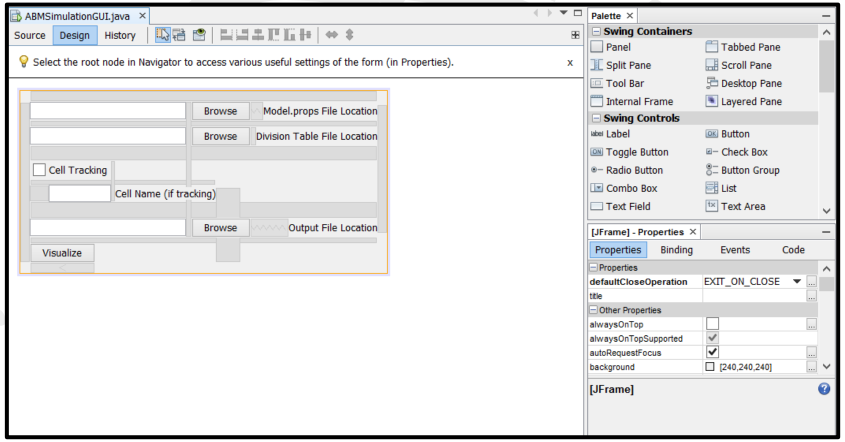
DB: data50.3d
Cycle: 0

Max: 100.0
Min: 0.0000

user: Kison
Tue Jul 07 12:41:24 2015

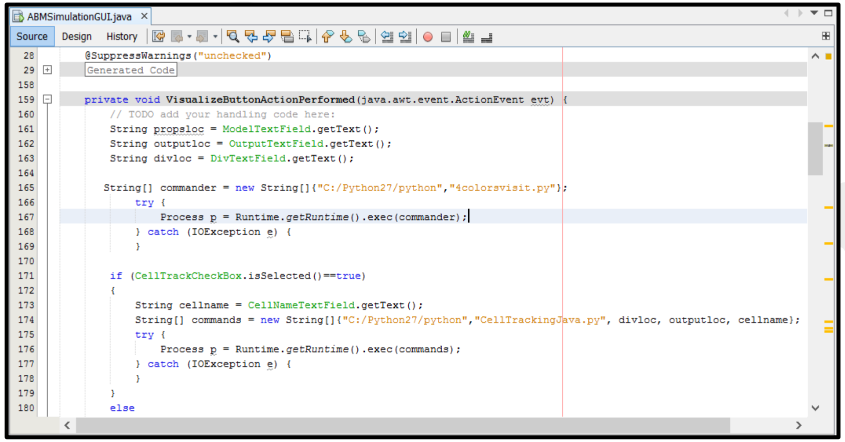
Netbeans IDE

- Open source program written in Java
- Allows for the creation of a Java GUI without being proficient in the programming language
- Streamlines the Nuclei-to-Display process



Select the root node in Navigator to access various useful settings of the form (in Properties).

```
Properties
defaultCloseOperation   EXIT_ON_CLOSE
title
Other Properties
alwaysOnTop             ☐
alwaysOnTopSupported    ✓
autoRequestFocus        ✓
background              [240,240,240]
```



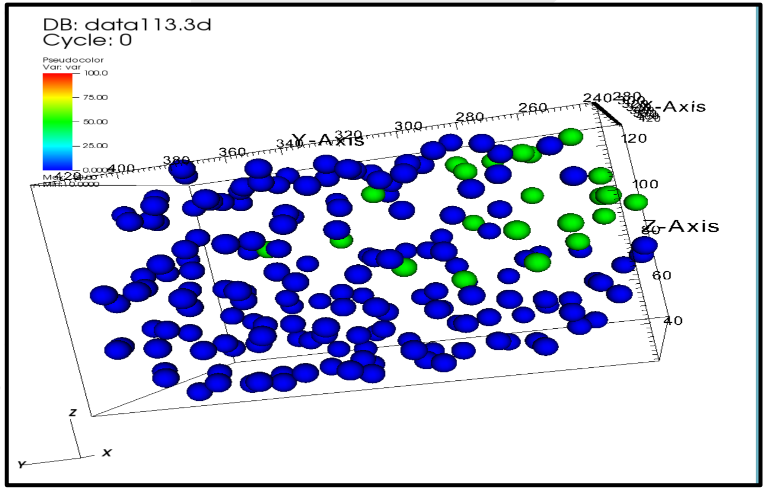
```
    @SuppressWarnings("unchecked")
    Generated Code

    private void VisualizeButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String propsloc = ModelTextField.getText();
        String outputloc = OutputTextField.getText();
        String divloc = DivTextField.getText();

       String[] commander = new String[]{"C:/Python27/python","4colorsvisit.py"};
            try {
                Process p = Runtime.getRuntime().exec(commander);
            } catch (IOException e) {
            }

        if (CellTrackCheckBox.isSelected()==true)
        {
            String cellname = CellNameTextField.getText();
            String[] commands = new String[]{"C:/Python27/python","CellTrackingJava.py", divloc, outputloc, cellname};
            try {
                Process p = Runtime.getRuntime().exec(commands);
            } catch (IOException e) {
            }
        }
        else
```

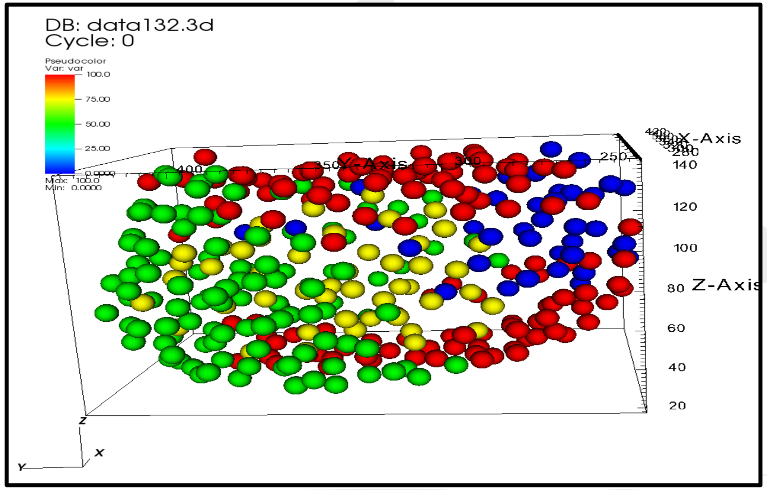
Current Status - RepastHPC

- The NetLogo simulation has been successfully implemented into RepastHPC
    - In serial, RepastHPC produces the same amount of cells as NetLogo
    - In parallel (without spatial updates), RepastHPC produces two less cells than NetLogo
    - In parallel (with spatial updates), RepastHPC produces two less cells than NetLogo

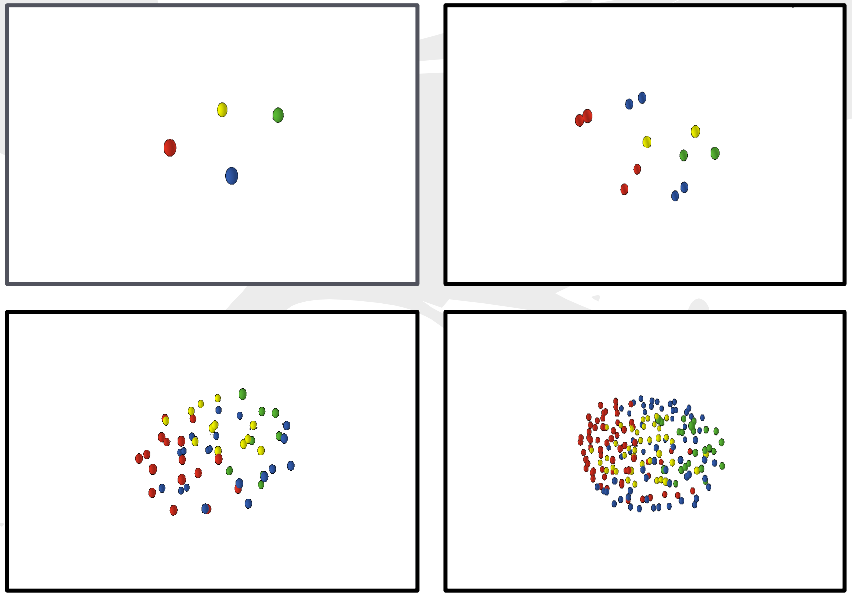
Current Status - VisIt & GUI

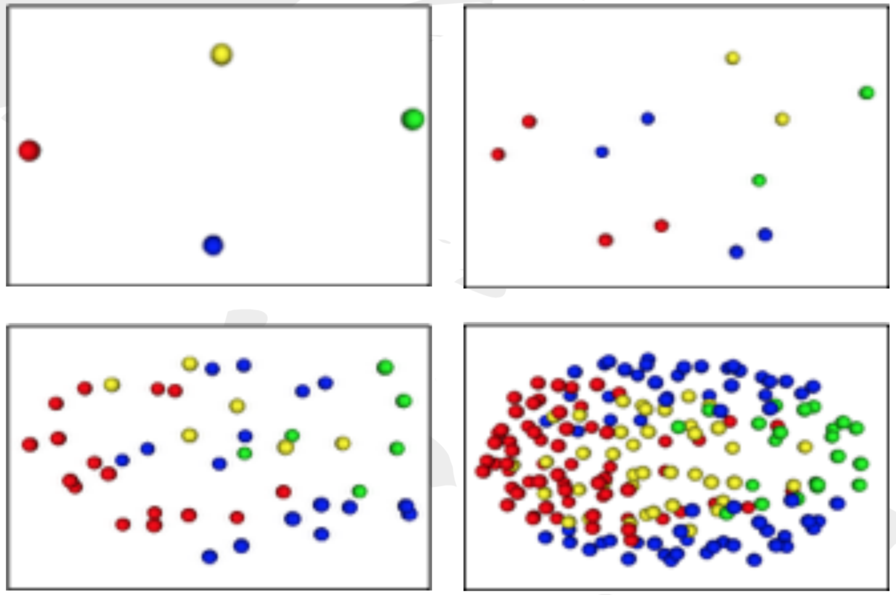
- Animation of RepastHPC results have been completed in VisIt
- GUI is almost complete
    - Working on displaying of VisIt files





RepastHPC vs NetLogo


NetLogo


RepastHPC

Future Plans

- Complete GUI for simulation
- Continue work on RepastHPC simulation
  - Enhance algorithms for more parallelized processing
  - Enhance wandering algorithms to better match the experimental data
  - Work on wandering algorithm to increase speed (parallel with spatial updates)

Special Thanks

- Ben Ramsey (University of Tennessee)
- Dali Wang (Oak Ridge National Laboratory)
- Kwai Wong (University of Tennessee)
- Scott Simmerman (Oak Ridge National Laboratory)
- Zhirong Bao (Memorial Sloan Kettering Cancer Center)
- John Murphy (Argonne National Laboratory)
- Chung Ng (Morehouse College)
- National Science Foundation
- Joint Institute For Computational Sciences

Resources

1. RepastHPC Tutorial and Download: http://repast.sourceforge.net/repast_hpc.php
2. Visit Tutorial and Manuals: https://wci.llnl.gov/simulation/computer-codes/visit/manuals
3. Caenorhabditis Genetics Center, College of Biological Sciences, University of Minnesota. “What is C. elegans?”. *College of Biological Sciences, University of Minnesota*. July 22, 2015. https://www.cbs.umn.edu/research/resources/cgc/what-c-elegans