Department of Computer Science
George Mason University Technical Reports

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/ 703-993-1530

GeoMason: GeoSpatial Support for MASON

Keith Sullivan ksulliv2@cs.gmu.edu
Mark Coletti mcoletti@cs.gmu.edu
Sean Luke sean@cs.gmu.edu

Technical Report GMU-CS-TR-2010-16

Abstract

MASON is a free, open-source Java-based discrete event multi-agent simulation toolkit that has been used to model network intrusions, unmanned aerial vehicles, nomadic migrations, and farmer/herder conflicts, among others. Many multi-agent models use georeferenced data which represent such things as road networks, rivers, vegetation coverage, population, and topology. However, MASON does not directly support georeferenced data. Therefore practitioners using MASON must hand craft such support, which may be difficult and error prone.

In this paper we describe newly added geospatial functionality in MASON that addresses this problem. We discuss the design of this functionality, called *GeoMASON*, and its use and limitations. Finally, we give examples on how to import and manipulate georeferenced data.

1 Introduction

MASON is an open source, multi-agent simulation library written in Java [14]. Designed as a research tool, MASON has been used in various multi-agent problem domains, including traffic control [1], network intrusion detection, ant foraging [21, 22, 9], cooperative target observation [15], and modeling civilization development [6, 5].

MASON is unusual among "swarm"-style lightweight multi-agent simulators in that it was intended to run on both front-end systems with visualization, on back-end compute servers with no visualization, and to migrate between these two modes. This enables experiments with potentially millions of agents involving many thousands of simulation runs, often for optimization or parameter-sweeping. To this end, MASON is designed to run efficiently, even in heterogeneous environments, and to have its core models entirely separated from visualization tools. This allows for MASON simulations to run on back-end machines without incurring visualization overhead, thus greatly increasing throughput; the simulations can be visualized in a separate session. In contrast, other agent-based modeling tools tightly integrate their computation and visualization parts. In addition, MASON was designed as a general purpose, extensible multi-agent simulation toolkit for tasks ranging from robotics to social networks.

In this paper we discuss extensions to MASON to provide Geographic Information Systems (GIS) facilities. Many agent-based models (ABMs) benefit from embedding the simulation in an actual environment based somewhere on the Earth. To do this, data representing surface features, otherwise known as *georeferenced data*, must be imported into a given simulation (i.e., features that correspond to roads, buildings, lakes, provinces, and the like must be incorporated into a given model). Moreover, ABMs may need non-spatial data associated with a given physical location such as demographics, pollution levels, location names, etc. A GIS is used to store, analyze, and visualize geospatial data and is typically the ultimate source for any such data used by an ABM tool.

GIS data comes in two broad categories: *raster* and *vector*. Raster data can represent such things as elevation values, a scanned-in topographic map, or satellite imagery. Vector data can represent geometry associated with land features such as buildings, roads, lakes, and so on. An ABM can use such data in a variety of ways. For example one project of ours blends slope and vegetation raster satellite data to compute crop and grazing suitability in a herder and farmer simulation for Africa [24]. In general an ABM can use geospatial data to have agents move along roads, determine building locations, locate water sources, compute political influence, etc.

Several extant ABM programs and toolkits are GIS capable [4, 12, 19]. NetLogo imports vector and raster data in ESRI shapefile format and allows user-supplied projections [27]. SeSAm imports vector data [10]. Both NetLogo and SeSAm are distributed in binary format, with no access to source code. The Kenge [3] project adds GIS support to the SWARM library [16] by using raster

data in cellular automata format. In a similar fashion, OBEUS works on the geographical automata system by importing raster data into a cellular automaton [2]. Similar to an ABM, Framsticks has some GIS support [11]. Repast [7] has full GIS support: importing vector and raster 2D and 3D data, agent manipulation of GIS data, and using user supplied projections. Repast can also communicate with ArcGIS via intermediate files. Using the Java Topology Suite, Repast can represent any spatial geometry. Repast largely imports ESRI shapefiles, but can import any filetype supported by GeoTools.

MASON does not inherently support geospatial data, so practitioners have had to handcraft geospatial data support for their MASON simulations. To address this problem we have created an ancillary MASON package, *GeoMASON*, that supports such data. In the interest of following the MASON design philosophy of being lightweight, modular, and efficient, GeoMASON represents objects as general geometric shapes instead of objects that support full blown GIS functionality. GeoMASON geometry is supported via the Java Topology Suite API which allows for general geometry related operations. Currently GeoMASON only works with vector formatted data; raster data will be supported in a future release. However, MASON itself already partially supports raster geospatial data with existing 2D and 3D grids of hexagons or squares.

By following MASON's design philosophy of separating computation and visualization, GeoMASON does not add significant additional computational load to MASON. The design intent is to do computationally expensive GIS related processing, such as map projecting, outside of MASON. As such, GeoMASON does not make MASON into a general GIS tool. Rather, GeoMASON adds the ability to import and use geospatial data within MASON; GeoMASON does not presently have the ability to do complex GIS-like operations or analysis. Any changes to the GIS data require a dedicated GIS program such as Quantum GIS [23], uDIG [17], or GRASS [18].

GeoMASON is part of the contrib branch of the MASON SVN repository and can be downloaded from http://code.google.com/p/mason/.

2 Geographical Information System Overview

Generally a Geographical Information System collects, stores, analyzes, and visualizes geospatial data [13]. For example, a GIS can be used to analyze migration patterns, population distributions, and predict flood coverage. GIS data can be gathered from a variety of sources including satellite imagery, scanned topographic maps, radar, and Light Detection And Ranging (LIDAR) data. Geospatial data describes a location on the Earth's surface of which raster and vector are two broad categories. Raster data is organized as a grid of data corresponding to a patch on the Earth's surface. Each grid cell can represent an elevation value, population size, type of vegetation, slope, a photograph pixel, and so on. In vector data individual features are represented as geometric shapes such as points, lines and polygons. For example, lakes could be represented by polygons, roads as line strings, and cell towers as points.

The Earth is a three dimensional object that can be modeled as an ellipsoid. A map or most simulations of surface features is a two dimensional representation of a region on that ellipsoid. To get from the three dimensional domain to the two dimensional, the data must be projected into a two dimensional coordinate system. Unfortunately the projection process will introduce spatial distortion. There exist a variety of well known projections each with their respective distortions. Generally projections will try to minimize some subset of distance, shape, area, or shortest route distortions. When bringing together disparate geospatial data for an area of interest it is important to ensure that they all use the same coordinate reference system. Otherwise, the data from different sources will not line up when used together.

3 Design

Like MASON, GeoMASON is written in three layers: a utility layer, a model layer, and a visualization layer. The utility layer imports GIS data into MASON and also supports exporting. The model layer contains zero or more GeomFields which contain the GIS geometries and associated metadata. Finally, the visualization layer contains classes for drawing the geometries and displaying the associated metadata. Figure 1 shows a simplified UML diagram relating GeoMASON objects within the model and visualization layers, and how these objects relate to the MASON architecture.

3.1 Model Layer

The model layer in MASON is unaware of the visualization layer. The model layer contains a single instance of a MASON model class, which in turn contains a discrete-event Schedule, random number generator, and zero or more *fields*. A field relates objects or data to a specific location: for example, a 2D grid of integers may be a field; or a data structure representing continuous 3D space; or a social network. Objects may exist in more than one field.

GeoMASON introduces two new fields, GeomVectorField and GeomGridField, that are both subclasses of GeomField. They are described below:

GeomVectorField contains all the geometry objects from the GIS datafile. Individual geometry are stored as MasonGeometry objects. A MasonGeometry object consists of JTS (Java Topology Suite) [25]

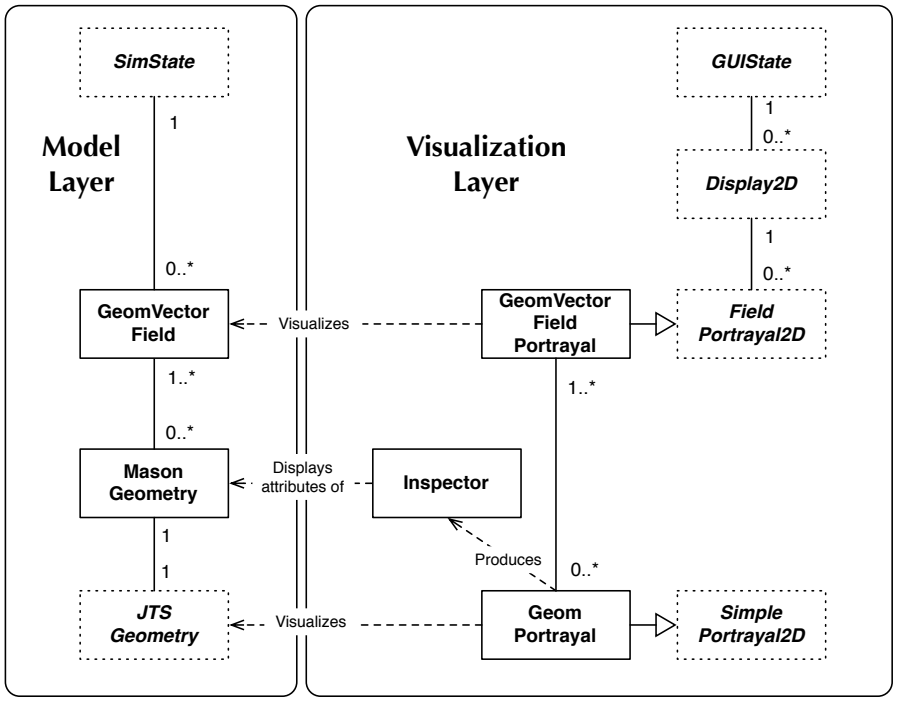


Figure 1: A simple UML diagram of how GeoMASON fits into the software architecture of MASON. Built-in MASON and JTS objects are depicted in *italics*.

geometry and option user supplied data. JTS geometries provide basic geometric operations such as intersection, union, and distance calculations. A JTS Geometry also allows for user provided data, which GeoMASON uses to attach associated attribute information.

GeomGridField embues georeference awareness to a user provided MASON Grid2D object. That is, an implementor could use, say, an IntGrid2D or DoubleGrid2D to store grid based GIS data in a GeomGridField and then define the position and ground resolution for the grid. GeomGridField provides basic grid-vector conversion services, such as returning a JTS Point that corresponds to a given grid cells centroid or returning the coordinate of a grid cell that a given Point falls in.

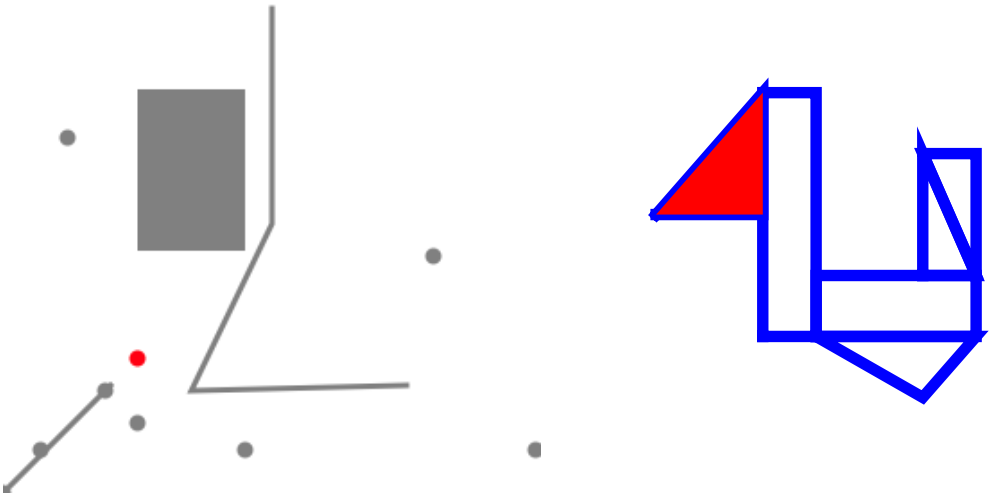
3.2 Visualization Layer

The visualization layer is responsible for displaying fields and individual objects, and allowing inspection of objects. The visualization layer also provides one or more displays for showing the model. Displays hold zero or more *field portrayals*, which are responsible for displaying all objects in field and for handling any user requests involving objects within that field. Field portrayals typically draw individual objects within the field using one or more *simple portrayals*. Each simple portrayal can build an *inspector* to display and modify user-defined information about a specific object.

GeoMASON introduces a new field portrayal, GeomVectorFieldPortrayal, for displaying GeomVectorFields. GeoMASON also provides GeomPortrayal that extends a MASON SimplePortrayal and handles portraying individual MasonGeometry objects. Since GeomGridField is Grid2D wrapper, regular MASON field portrayals can be used to render its contents.

3.3 Utility Layer

MASON's utility layer contains classes for a variety of general uses, including random number generation, data structures, and GUI widgets. GeoMASON adds the ability to import and export GIS datafiles and associated attribute files.



(a) Agent moving among arbitrary geometry (b) Highlighting a nearby polygon

Figure 2: GeoMASON examples of determining nearby objects.

The GeomImporter interface reads GIS information into a GeomField. Currently, there are four subclasses of GeomImporter, which are listed below:

- ShapeFileImporter: a pure Java importer for ESRI shape files [8], which has no dependencies on external, third-party libraries.
- GeoToolsImporter: a pure Java importer which relies on the GeoTools library [20]. This can import GIS files in the following formats: ESRI shapefile, PostGIS, Web Feature Format (WFS), Open Web Service (OWS), and various database formats including DB2 and Oracle.
- OGRImporter: a JNI interface to the OGR library [26], which supports a very large number of vector geospatial formats. This requires the user to compile and install the full OGR library on their system.
- GDALImporter: a JNI interface to OGRs sister library, GDAL, that reads in a large variety of grid-based geospatial formats. This similarly requires GDAL to be installed locally.

The GeomExporter interface writes geospatial data from a GeomField. Presently there is just one implementation, ShapeFileExporter, which writes GeomField's contents to ESRI Shape Files. We plan to add other GeomExporter GeoTools- and OGR-based implementations.

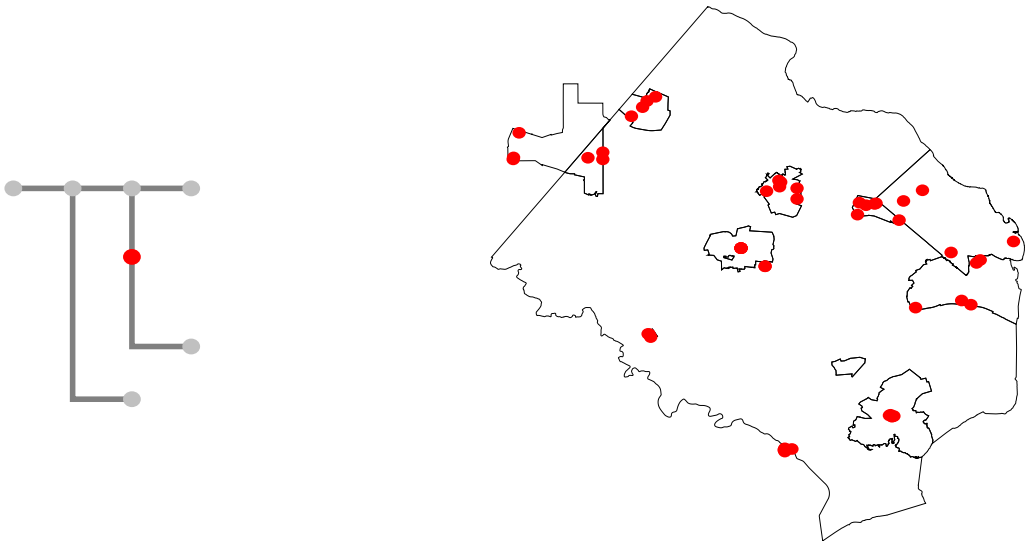
4 Examples

In a multi-agent simulation, a common functionality is to determine nearby objects. GeoMASON queries a GeomField for the closest objects using JTS primitives. This functionality exists for both simple geometric environments and geospatial environments. In Figure 2a an agent, represented by a red dot, reports the closest object, be it a string of lines, a polygon, or a point, as it moves through the environment. In addition to the closest object, GeoMASON can also determine all objects which touch a given object, i.e., determine adjacent objects. Figure 2b shows a set of adjacent polygons. A polygon is randomly selected and is colored red. In subsequent steps an arbitrary adjacent polygon is selected and colored red; that is, the notion of the current polygon "moves" to an adjacent one.

GeoMASON can also enforce geometric constraints on the agents. For example, Figure 3a shows how an agent's movement may be limited to a network. In this example, the agent moves randomly, but must stay on the gray lines. At intersections, the agent randomly chooses a new direction. An agent's movement can also be limited to a polygon. Again using the Fairfax county voting districts, Figure 3b shows agents (red dots) whose movement is limited to a single polygon.

Figure 6 shows how geometry can be colored based on criteria. A practitioner may color geometry based on inherent feature properties as well as values calculated dynamically from the simulation. In this case, the darker the shade of blue, the more agents are in a given district.

GeoMASON scales to many agents interacting with many geometry objects. Figure 4 shows 1000 agents moving along walkways on a map of George Mason University. Like the network model above, the agents move randomly along campus walkways. Figure 5 shows an inspector for the Johnson Center building at GMU. While the original attribute data contains many fields, the figure illustrates that GeoMASON can limit the number of displayed fields provided the user knows their names.



(a) Agent moving along a network

(b) Agents moving within polygon regions

Figure 3: GeoMASON examples of enforcing geometric constraints on an agent's movement.

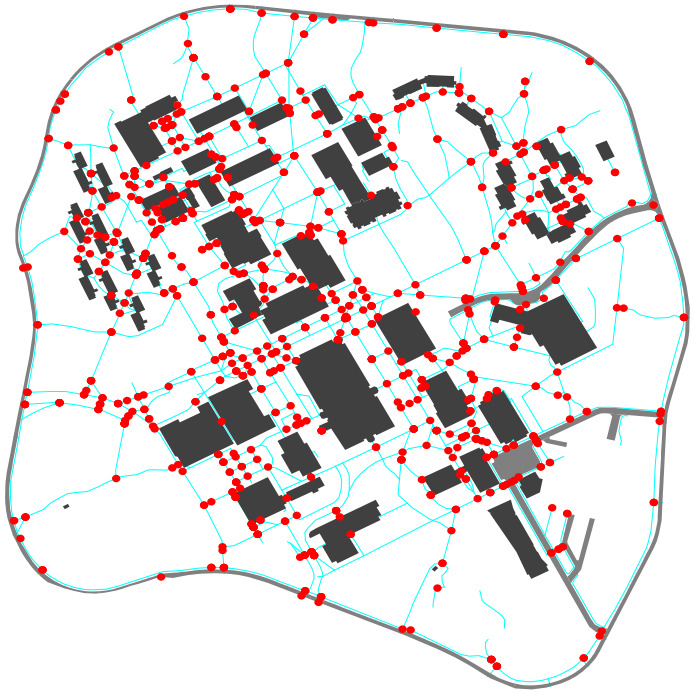


Figure 4: Agents moving on campus walkways

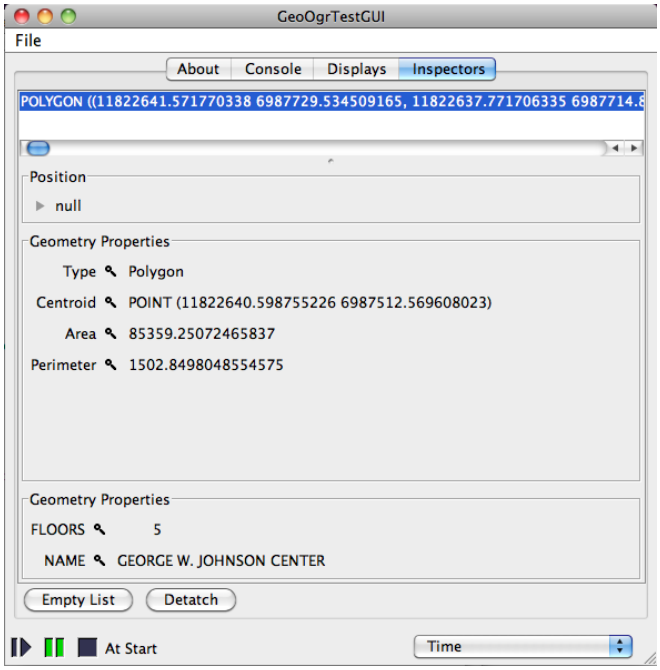


Figure 5: Inspector for campus walkways example

5 Conclusion

GeoMASON allows multi-agent modelers to use vector georeferenced data within the MASON framework. It supports many data formats, including Shape, SDTS, NTF, TIGER, S57, VRT, and GML. GeoMASON's reliance on the Java Topology Suite allows for complex geometric operations to be applied to geospatial data. It follows the MASON design philosophy of separating computation and visualization, meaning that ABM researchers can take advantage of MASON's speed while incorporating GIS information.

There are some avenues for future work for increasing GeoMASON's functionality. A convenience function for synchronizing minimum bounding rectangles between GeomFields could be provided. And, as mentioned earlier, GeoTools and OGR versions of write support can be provided.

Acknowledgements

We gratefully acknowledge the support for this research provided by the Joint Improvised Explosive Device Defeat Office (JIEDDO) J-9 Division and the Office of Naval Research (ONR) under Government Contract number N00014-09-C-0419, and also NSF grant 0916870. We would also like to thank the entire MASON development team and the GMU Center for Social Complexity. Finally, we would like to thank the Government Documents/Maps groups at the GMU library for providing the campus and Fairfax County geospatial data.

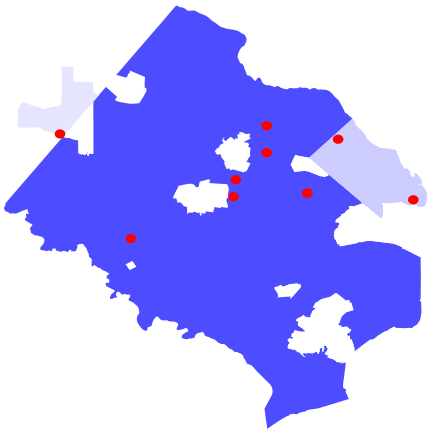


Figure 6: Showing agent population of areas ranked by color.

References

[1] Gabriel Balan and Sean Luke. History-based traffic control. In *Proceedings of Autonomous Agents and Multiagent Systems*, Hakodate, Japan, May 2006.

[2] Itzak Beneson, Slava Birfur, and VLad Kharbash. Geographic automata systems and the OBEUS software for their implementation. In *Complex Artificial Environments*, pages 137 – 153. Springer, 2006.

[3] Paul Box, Akiko Ogawa, and Alex Wells. Kenge: The Swarm GIS-CA libraries. `http://www.gis.usu.edu/swarm`, 1999.

[4] Christian J. E. Castle and Andrew T. Crooks. Principles and concepts of agent-based modelling for developing geospatial simulations. Working Papers Series, Center of Advanced Spatial Analysis, University College London, 2006.

[5] Claudio Cioffi-Revilla, Sean Luke, Dawn C. Parker, J. D. Rogers, W. W. Fitzhugh, W. Honeychurch, B. Frohlich, P. DePriest, and Chanag Amartuvshin. Computational modeling frontiers in international politics: Agent-based modeling and simulation of adaptive behavior and long-term change in inner asia. In *Proceedings of First World Congress on Social Simulation*, 2006.

[6] Claudio Cioffi-Revilla, Sean Paus, Sean Luke, James Olds, and Jason Thomas. Mnemonic structure and sociality: A computational agent-based simulation model. In *Proceedings of the Conference on Collective Intentionality IV*, 2004.

[7] Nick Collier. Repast: An agent based modelling toolkit for java, 2001. http://repast.sourceforge.net.

[8] Environmental Systems Research Institute, Inc. *ESRI Shapefile Technical Description*, July 1998.

[9] Brian Hrolenok, Sean Luke, Keith Sullivan, and Christopher Vo. Collaborative foraging using beacons. In *Proceedings of Autonomous Agents and Multiagent Systems*, 2010.

[10] Franziska Klügl, Rainer Herrler, Manuel Fehler, Conelia Triebig, and Gustavo Andriotti. SeSAm: Shell for simulated agent systems. `http://www.simsesam.de/`, 2010.

[11] Maciej Komosinski and Szymon Ulatowski. Framsticks: Creating and understanding complexity of life. In *Artificial Life Models in Software*, pages 107 – 148. Springer, 2009.

[12] Karl D. Liebert, David C. Earnest, and Andreas Tolk. Using GIS data to build virtual environments for agent based models. In *Proceedings of Agent Directed Simulation Symposium at the Spring Simulation Multiconference*, 2008.

[13] Paul Longley, Michael Goodchild, David Maguire, and David Rhind. *Geographic Information Systems and Science*. Wiley, 2002.

[14] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. MASON: A multiagent simulation envrionment. *Simulation*, 81(7):517 – 527, 2005.

[15] Sean Luke, Keith Sullivan, Liviu Panait, and Gabriel Balan. Tunably decentralized algorithms for cooperative target observation. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005)*, 2005.

[16] Nelson Minar, Roger Burkhart, Christopher Langton, and Manar Askenazi. The swarm simulation system., 1996. http://www.swarm.org.

[17] Markus Neteler and Helena Mitasova. Open source GIS: A GRASS GIS approach. `http://udig.refractions.net`, 2007.

[18] Markus Neteler and Helena Mitasova. *Open Source GIS: A GRASS GIS Approach*. Springer, 2008.

[19] Cynthia Nikolai and Gregory Madey. Tools of the trade: A survey of various agent based modeling platforms. *Journal of Artificial Societies and Social Simulation*, 12(2), 2009.

[20] Open Source Geospatial Foundation. GeoTools: The open source Java GIS toolkit. `http://www.geotools.com`, 2010.

[21] Liviu Panait and Sean Luke. Ant foraging revisited. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*, 2004.

[22] Liviu Panait and Sean Luke. Learning ant foraging behaviors. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*, 2004.

[23] Quantum GIS Development Team. Quantum GIS geographic information system. `http://qgis.osgeo.org`, 2010.

[24] M. Rouleau, M. Coletti, J. K. Bassett, A.B. Hailegiorgis, T. Gulden, and W.G. Kennedy. Conflict in complex socio-natural systems: Using agent-based modeling to understand the behavioral roots of social unrest within the mandera triangle. In *Human Behavior-Computational Modeling and Interoperability Conference*, 2009.

[25] Vivid Solutions, Inc. Java Topology Suite, September 2009. http://www.vividsolutions.com/jts/jtshome.htm.

[26] Frank Warmerdam. OGR Simple Features Library, 1998. http://www.gdal.org/ogr/.

[27] Uri Wilensky. Netlogo, 1999. http://ccl.northwestern.edu/netlogo/.